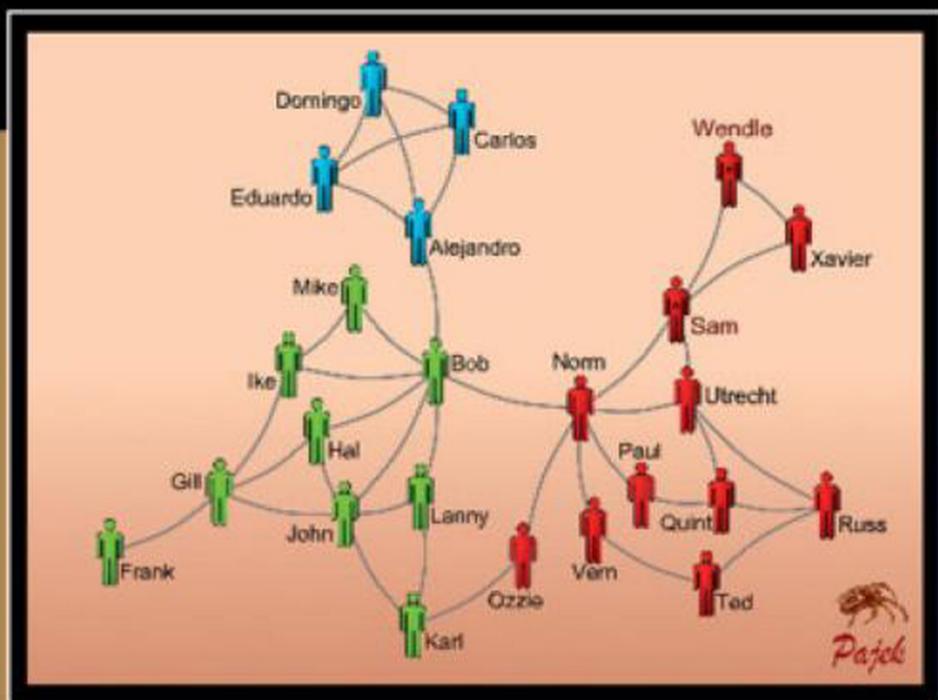


Exploratory Social Network Analysis with Pajek

REVISED AND EXPANDED EDITION
FOR UPDATED SOFTWARE

THIRD EDITION



Wouter de Nooy, Andrej Mrvar
and Vladimir Batagelj

Exploratory Social Network Analysis with Pajek

This is an extensively revised and expanded third edition of the successful textbook on analysis and visualization of social networks integrating theory, applications, and professional software for performing network analysis. The main structural concepts and their applications in social research are introduced with exercises. Pajek software and data sets are available, so readers can learn network analysis through application and case studies. In the end, readers will have the knowledge, skills, and tools to apply social network analysis across different disciplines. A fundamental redesign of the menu structure and the capability to analyze much larger networks required a new edition. This edition presents several new operations, e.g., community detection, generalized main paths searches, new network indices, advanced visualization approaches, and instructions for installing Pajek under MacOSX. This third edition is up-to-date with Pajek version 5 and it introduces PajekXXL for very large networks and Pajek3XL for huge networks.

Wouter de Nooy is Associate Professor in the Department of Communication Science at the University of Amsterdam, the Netherlands. He is a member of the Amsterdam School of Communication Research. His research interests include dynamic models for social networks and their application in culture and politics.

Andrej Mrvar is Professor of Social Science Informatics at the Faculty of Social Sciences, University of Ljubljana, Slovenia. He started developing the programs *Pajek*, *PajekXXL*, and *Pajek3XL* as part of his PhD dissertation at the Faculty of Computer and Information Science in 1996. Since then Pajek has been in constant development.

Vladimir Batagelj is Professor Emeritus of the University of Ljubljana, Slovenia. He is also a member of the Institute of Mathematics, Physics, and Mechanics (IMFM), Ljubljana, and of AMI, UP, Koper. His coauthored book *Generalized Blockmodeling* was awarded the 2007 Harrison White Outstanding Book Award by the Mathematical Sociology Section of the American Sociological Association. From the International Network for Social Network Analysis he was awarded the Georg Simmel Award (2007) and the Richards Software Award for the program *Pajek* (2013). In 2014 he coauthored a book entitled *Understanding Large Temporal Networks and Spatial Networks*.

Structural Analysis in the Social Sciences

Series Editor: Mark Granovetter

The series *Structural Analysis in the Social Sciences* presents studies that analyze social behavior and institutions by reference to relations among such concrete social entities as persons, organizations, and nations. Relational analysis contrasts on the one hand with reductionist methodological individualism and on the other with macro-level determinism, whether based on technology, material conditions, economic conflict, adaptive evolution, or functional imperatives. In this more intellectually flexible structural middle ground, analysts situate actors and their relations in a variety of contexts. Since the series began in 1987, its authors have variously focused on small groups, history, culture, politics, kinship, aesthetics, economics, and complex organizations, creatively theorizing how these shape and in turn are shaped by social relations. Their style and methods have ranged widely, from intense, long-term ethnographic observation to highly abstract mathematical models. Their disciplinary affiliations have included history, anthropology, sociology, political science, business, economics, mathematics, and computer science. Some have made explicit use of social network analysis, including many of the cutting-edge and standard works of that approach, whereas others have kept formal analysis in the background and used “networks” as a fruitful orienting metaphor. All have in common a sophisticated and revealing approach that forcefully illuminates our complex social world.

Recent Books in the Series

- Wouter De Nooy, Andrej Mrvar and Vladimir Batagelj, *Exploratory Social Network Analysis with Pajek: Revised and Expanded Edition for Updated Software. Third Edition*
- Darius Mehri, *Iran Auto*
- Navid Hassanpour, *Leading from the Periphery and Network Collective Action*
- Cheol-Sung Lee, *When Solidarity Works* (Second Edition)
- Benjamin Cornwell, *Social Sequence Analysis* (Second Edition)
- Mariela Szwarcberg, *Mobilizing Poor Voters* (Second Edition)
- Luke M. Gerdes, ed., *Illuminating Dark Networks* (Second Edition)
- Silvia Domínguez and Betina Hollstein, eds., *Mixed Methods in Studying Social Networks* (Second Edition)
- Dean Lusher, Johan Koskinen, and Garry Robins, eds., *Exponential Random Graph Models for Social Networks: Theory, Methods, and Applications* (Second Edition)
- Sean F. Everton, *Disrupting Dark Networks* (Second Edition)
- Wouter de Nooy, Andrej Mrvar, and Vladimir Batagelj, *Exploratory Social Network Analysis with Pajek* (First Edition)
- Noah E. Friedkin and Eugene C. Johnsen, *Social Influence Network Theory* (Second Edition)
- Zeev Maoz, *The Networks of Nations: The Evolution and Structure of International Networks, 1815–2002* (Second Edition)

(continued after the index)

Exploratory Social Network Analysis with Pajek

*Revised and Expanded Edition for Updated
Software. Third Edition*

WOUTER DE NOOY

University of Amsterdam

ANDREJ MRVAR

University of Ljubljana

VLADIMIR BATAGELJ

University of Ljubljana



CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE
UNIVERSITY PRESS

University Printing House, Cambridge CB2 8BS, United Kingdom

One Liberty Plaza, 20th Floor, New York, NY 10006, USA

477 Williamstown Road, Port Melbourne, VIC 3207, Australia

314–321, 3rd Floor, Plot 3, Splendor Forum, Jasola District Centre,
New Delhi - 110025, India

79 Anson Road, #06-04/06, Singapore 079906

Cambridge University Press is part of the University of Cambridge.

It furthers the University's mission by disseminating knowledge in the pursuit of education, learning, and research at the highest international levels of excellence.

www.cambridge.org

Information on this title: www.cambridge.org/9781108474146

DOI: 10.1017/9781108565691

© Wouter de Nooy, Andrej Mrvar, and Vladimir Batagelj 2018

This publication is in copyright. Subject to statutory exception and to the provisions of relevant collective licensing agreements, no reproduction of any part may take place without the written permission of Cambridge University Press.

First published 2018

Printed in the United States of America by Sheridan Books, Inc.

A catalogue record for this publication is available from the British Library.

ISBN 978-1-108-47414-6 Hardback

ISBN 978-1-108-46227-3 Paperback

Cambridge University Press has no responsibility for the persistence or accuracy of URLs for external or third-party internet websites referred to in this publication and does not guarantee that any content on such websites is, or will remain, accurate or appropriate.

Contents

<i>List of Figures</i>	page xiii
<i>List of Tables</i>	xix
<i>Preface to the Third Edition</i>	xxi
<i>Preface to the Second Edition</i>	xxiii
<i>Preface to the First Edition</i>	xxv
Overview	xxvi
Justification	xxviii
Acknowledgments	xxix

PART I FUNDAMENTALS

1	Looking for Social Structure	3
1.1	Introduction	3
1.2	Sociometry and Sociogram	3
1.3	Exploratory Social Network Analysis	5
1.3.1	Network Definition	6
1.3.2	Manipulation	12
1.3.3	Calculation	15
1.3.4	Visualization	17
1.4	Assembling a Social Network	27
1.5	Summary	30
1.6	Questions	31
1.7	Assignment	32
1.8	Further Reading	32
1.9	Answers	33
2	Attributes and Relations	36
2.1	Introduction	36
2.2	Example: The World System	36
2.3	Partitions	38

2.4	Reduction of a Network	45
2.4.1	Local View	46
2.4.2	Global View	48
2.4.3	Contextual View	51
2.5	Vectors and Coordinates	53
2.6	Network Analysis and Statistics	61
2.7	Summary	63
2.8	Questions	64
2.9	Assignment	65
2.10	Further Reading	65
2.11	Answers	66
PART II COHESION		
3	Cohesive Subgroups	73
3.1	Introduction	73
3.2	Example	73
3.3	Density and Degree	75
3.4	Components	79
3.5	Cores	83
3.6	Cliques and Complete Subnetworks	86
3.7	Summary	92
3.8	Questions	94
3.9	Assignment	96
3.10	Further Reading	96
3.11	Answers	96
4	Sentiments and Friendship	99
4.1	Introduction	99
4.2	Balance Theory	99
4.3	Example	103
4.4	Detecting Structural Balance and Clusterability	103
4.5	Development in Time	109
4.6	Summary	113
4.7	Questions	113
4.8	Assignment	115
4.9	Further Reading	115
4.10	Answers	116
5	Affiliations	119
5.1	Introduction	119
5.2	Example	120
5.3	Two-Mode and One-Mode Networks	121
5.4	Islands	127

5.5	Communities	132
5.6	The Third Dimension	135
5.7	Summary	139
5.8	Questions	140
5.9	Assignment	141
5.10	Further Reading	141
5.11	Answers	142

PART III BROKERAGE

6	Center and Periphery	149
6.1	Introduction	149
6.2	Example	149
6.3	Distance	151
6.4	Betweenness	158
6.5	Eigenvector Centrality	160
6.6	Assortativity	162
6.7	Summary	164
6.8	Questions	165
6.9	Assignment	166
6.10	Further Reading	167
6.11	Answers	167
7	Brokers and Bridges	170
7.1	Introduction	170
7.2	Example	171
7.3	Bridges and Bi-Components	172
7.4	Ego-Networks and Constraint	177
7.5	Affiliations and Brokerage Roles	184
7.6	Summary	189
7.7	Questions	190
7.8	Assignment	191
7.9	Further Reading	193
7.10	Answers	194
8	Diffusion	197
8.1	Example	197
8.2	Contagion	200
8.3	Exposure and Thresholds	204
8.4	Critical Mass	211
8.5	Summary	216
8.6	Questions	217
8.7	Assignment	219
8.8	Further Reading	219
8.9	Answers	220

PART IV RANKING

9	Prestige	225
9.1	Introduction	225
9.2	Example	226
9.3	Popularity and Indegree	227
9.4	Correlation	229
9.5	Domains	231
9.6	Proximity Prestige	235
9.7	Summary	238
9.8	Questions	238
9.9	Assignment	240
9.10	Further Reading	241
9.11	Answers	241
10	Ranking	244
10.1	Introduction	244
10.2	Example	245
10.3	Triadic Analysis	245
10.4	Acyclic Networks	253
10.5	Symmetric-Acyclic Decomposition	256
10.6	Summary	261
10.7	Questions	263
10.8	Assignment	265
10.9	Further Reading	265
10.10	Answers	266
11	Genealogies and Citations	269
11.1	Introduction	269
11.2	Example I: Genealogy of the Ragusan Nobility	269
11.3	Family Trees	270
11.4	Social Research on Genealogies	278
11.5	Example II: Citations among Papers on Network Centrality	289
11.6	Citations	291
11.7	Summary	304
11.8	Questions	304
11.9	Assignment 1	306
11.10	Assignment 2	306
11.11	Further Reading	306
11.12	Answers	307

PART V MODELING

12	Blockmodels	315
12.1	Introduction	315

12.2	Matrices and Permutation	316
12.3	Roles and Positions: Equivalence	322
12.4	Blockmodeling	331
	12.4.1 Blockmodel	332
	12.4.2 Blockmodeling	333
	12.4.3 Regular Equivalence	338
12.5	Summary	343
12.6	Questions	345
12.7	Assignment	347
12.8	Further Reading	348
12.9	Answers	348
13	Random Graph Models	353
13.1	Introduction	353
13.2	Example	355
13.3	Modeling Overall Network Structure	357
	13.3.1 Classic Uniform Models	358
	13.3.2 Small-World Models	362
	13.3.3 Preferential Attachment Models	366
13.4	Monte Carlo Simulation	373
13.5	Summary	377
13.6	Questions	379
13.7	Assignment	381
13.8	Further Reading	381
13.9	Answers	383
Appendix 1	Getting Started with Pajek	387
A1.1	Installation	387
A1.2	Network Data Formats	387
A1.3	Creating Network Files for Pajek	389
	A1.3.1 Within Pajek	389
	A1.3.2 Helper Software	391
	A1.3.3 Word Processor	392
	A1.3.4 Relational Database	394
A1.4	Limitations	400
A1.5	PajekXXL and Pajek3XL	400
A1.6	Updates of Pajek	402
Appendix 2	Exporting Visualizations	404
A2.1	Export Formats	404
	A2.1.1 Bitmap and JPEG	404
	A2.1.2 Encapsulated PostScript	405
	A2.1.3 Scalable Vector Graphics	406
	A2.1.4 VOSviewer	408
	A2.1.5 Virtual Reality Modeling Language and X3D	409

	A2.1.6 MDL MOL and Kinemages	410
A2.2	Layout Options	411
	A2.2.1 Top Frame on the Left: EPS/SVG Vertex Default	412
	A2.2.2 Bottom Frame on the Left: EPS/SVG Line Default	416
	A2.2.3 Top Frame on the Right	418
	A2.2.4 Middle Frame on the Right – Background Colors	419
	A2.2.5 Bottom Frame on the Right – EPS Border	420
Appendix 3	Installing Pajek on Mac OS X	421
Appendix 4	Shortcut Key Combinations	424
	A4.1 Main Screen	424
	A4.2 Hierarchy Edit Screen	425
	A4.3 Draw Screen	425
	<i>Glossary</i>	427
	<i>Index of Pajek and R Commands</i>	439
	<i>Subject Index</i>	445

Figures

1	Dependencies between the chapters (for the second and third editions)	<i>page</i> xxvii
2	Sociogram of dining-table partners	4
3	Partial listing of a multiple relations network data file for Pajek	10
4	Pajek Main screen	11
5	Menu structure in Pajek	13
6	Dialog box in Pajek	14
7	Report screen in Pajek	16
8	Dialog box of <i>Network> Info> General</i> command	17
9	Draw screen in Pajek	19
10	Continue dialog box	21
11	A selected option in the Draw screen	23
12	<i>Options</i> menu of the Draw screen	24
13	Textual output from <i>[Draw]Info> All Properties</i>	24
14	A 3-D rendering of the dining-table partners network	26
15	Empty network	29
16	Edit Network screen	29
17	World trade of manufactures of metal and world system position	40
18	Edit screen with partition according to world system position	41
19	Vertex colors according to a partition in Pajek	43
20	Trade ties within South America	46
21	The <i>Partitions</i> menu	47
22	World system positions in South America: (2) semiperiphery and (3) periphery	48
23	Trade in manufactures of metal among continents (imports in thousands of US dollars)	49

24	Trade among continents in the Draw screen	51
25	Contextual view of trade in South America	52
26	Geographical view of world trade in manufactures of metal, ca. 1994	54
27	<i>Vector>Info</i> dialog box	55
28	Trade, position in the world system, and GDP per capita	58
29	Aggregate trade in manufactures of metal among world system positions	67
30	Contextual view of North American trade ties and (mean) GDP per capita	68
31	Visiting ties in Attiro	74
32	A simple unconnected directed network	79
33	Strong components (contours) and family–friendship groupings (vertex colors and numbers) in the network of Attiro	82
34	k -cores in the visiting network at Attiro	84
35	k -cores	85
36	Stacking or nesting of k -cores	85
37	The complete triad and an example	87
38	A hierarchy of cliques	89
39	Viewing a hierarchy in an Edit screen	90
40	Complete triads and family–friendship groupings (colors and numbers inside vertices)	91
41	Decision tree for the analysis of cohesive subgroups	93
42	A Person-Other-Object (X) triple	100
43	P–O–X triple as a signed digraph	100
44	A balanced network	102
45	First positive and negative choices between novices at T_4	104
46	Output listing of a <i>Doreian–Mrvar Method*</i> command	108
47	Three solutions with one error	109
48	Partial listing of Sampson.net	110
49	Differences between two solutions with four classes	116
50	A fragment of the Scottish directorates network	122
51	One-mode network of firms created from the network in Figure 50	123
52	One-mode network of directors derived from Figure 50	124
53	Islands in the network of Scottish firms, 1904–1905 (contours added manually)	128
54	The islands in the network of Scottish firms (1904–1905) with industrial categories (class numbers) and capital (vertex size)	130

55	Islands in three dimensions	136
56	Coordinate system of Pajek	136
57	A landscape of islands in the Scottish firms network	138
58	Communication ties within a sawmill	150
59	Star-network and line-network	152
60	Distances to or from Juan (vertex colors: Default GreyScale 1)	156
61	Geodesics between HP-1 and EM-4	157
62	Betweenness centrality in the sawmill	160
63	Communication network of striking employees	171
64	Cut-vertices (gray) and bi-components (manually circled) in the strike network	174
65	Hierarchy of bi-components and bridges in the strike network	177
66	Three connected triads	178
67	Alejandro's ego-network	179
68	Proportional strength of ties around Alejandro	181
69	Constraints on Alejandro	181
70	Energized constraint network	183
71	Five brokerage roles of actor v	185
72	Bob's ego-network	186
73	Constraint inside groups	188
74	Two overlapping cliques	192
75	Friendship ties among superintendents and year of adoption	199
76	Adoption of the modern math method: diffusion curve	201
77	Diffusion by contacts in a random network ($N = 100$; vertex numbers indicate the distance from the source vertex)	201
78	Diffusion from a central and a marginal vertex	202
79	Adoption (vertex color) and exposure (in brackets) at the end of 1959	205
80	Modern math network with arcs pointing toward later adopters	209
81	Visiting ties and prestige leaders in San Juan Sur	226
82	<i>Partitions</i> menu in Pajek	231
83	Distances to family 47 (represented by the numbers within the vertices)	233
84	Proximity prestige in a small network	237
85	Student government discussion network	245
86	An example of a network with ranks	246
87	Triad types with their sequential numbers in Pajek	247
88	Strong components in the student government discussion network	255

89	Acyclic network with shrunk components	255
90	Clusters of symmetric ties in the student government network	257
91	Discussion network shrunk according to symmetric clusters	257
92	Symmetric components in the (modified) student government discussion network	258
93	The order of symmetric clusters according to the depth partition (acyclic)	260
94	Ranks in the student government discussion network	261
95	Three generations of descendants to Petrus Gondola (years of birth)	271
96	Ore graph	272
97	Descendants of Petrus Gondola and Ana Goce	274
98	Shortest paths between Paucho and Margarita Gondola	275
99	Structural relinking in an Ore graph	280
100	P-graph	281
101	Structural relinking in a P-graph	282
102	Fragment of relinking grandchildren	285
103	Centrality literature network in layers according to year of publication	290
104	k -cores in the centrality literature network (without isolates)	292
105	Traversal weights in a citation network	293
106	Forward local and key-route local (top), standard global and key-route global (middle), and backward local (bottom) main paths in the centrality literature network	299
107	Main path component of the centrality literature network (not all names are shown here)	303
108	Communication lines among striking employees	316
109	The matrix of the strike network sorted by ethnic and age groups	318
110	A network and a permutation	319
111	Partial listing of the strike network as a binary matrix	320
112	The strike network permuted according to ethnic and age groups	321
113	Part of the permuted strike network displayed as a binary network	322
114	Hypothetical ties among two instructors (i) and three students (s)	322

115	A dendrogram of similarities	324
116	Imports of miscellaneous manufactures of metal and world system position in 1980	325
117	Hierarchical clustering of the world trade network	328
118	Hierarchical clustering of countries in the Hierarchy Edit screen	329
119	An ideal core-periphery structure	331
120	Image matrix and shrunk network	332
121	Error in the imperfect core-periphery matrix	334
122	<i>Optimize Partition</i> dialog box	336
123	Output of the <i>Optimize Partition</i> procedure	336
124	<i>Random Start</i> dialog box	338
125	Matrix of the student government network	339
126	Image matrix and error matrix for the student government network	340
127	Assembling a blockmodel in Pajek	342
128	Random versions of a small friendship network	354
129	Political blogosphere, United States, February 8, 2005	356
130	Small-world random graph generation: Ring of local lines (left) and rewired lines (right)	363
131	Log-log degree distributions of the blogs network, absolute frequencies (left) and cumulative proportions (right)	369
132	<i>Read Network</i> dialog box	388
133	A network in Pajek matrix format	389
134	Editing vertex labels	390
135	Edit Network screen	390
136	An empty network in Pajek Arcs/Edges format	392
137	A network in the Pajek Arcs/Edges format	393
138	A network in the Pajek matrix format	393
139	A two-mode network in the Pajek Arcs/Edges format	394
140	Four tables in the world trade database (MS Access 2010)	395
141	Contents of the <i>Countries</i> table (partial)	395
142	A Lookup to the <i>Countries</i> table	396
143	Export a report to plain text	397
144	Tables and relations in the database of Scottish companies	399
145	The <i>Options</i> screen	411
146	Layout of a vertex and its label	412
147	Bezier curves and different vertex shapes (man, woman, and house)	414

148	The x/y ratio of a vertex	414
149	Visualization with Unicode symbols	416
150	The position and orientation of a line label	417
151	Gradients in SVG export: linear (left) and radial (right)	419
152	XQuartz webpage	422
153	WineHQ webpage	422
154	Pajek webpage	423
155	Pajek running on Mac OS X	423

Tables

1	Tabular output of the command <i>Partition>Info</i>	page 41
2	Distribution of GNP per capita in classes	56
3	Output of the <i>Info</i> command	62
4	Cross-tabulation of world system positions (rows) and GDP per capita (columns)	69
5	Frequency distribution of degree in the symmetrized network of visits	78
6	Error score with all choices at different moments ($\alpha = .5$)	112
7	Error score with first choices only ($\alpha = .5$)	117
8	Line multiplicity in the one-mode network of firms	126
9	Frequency tabulation of coordinator roles in the strike network	188
10	Adoption in the modern math network	203
11	Adoption rate and acceleration in the modern math diffusion curve	212
12	Fragment of Table 11	215
13	Indegree listing in Pajek	228
14	Input domain of f47	234
15	Size of input domains in the visiting relations network	235
16	Balance-theoretic models	250
17	Triad census of the example network	251
18	Triad census of the student government network	252
19	Number of children of Petrus Gondola and his male descendants	277
20	Size of sibling groups in 1200–1250 and 1300–1350	279
21	Birth cohorts among men and women	287
22	Traversal weights in the centrality literature network	297
23	Dissimilarity scores in the example network	324

24	Cross-tabulation of initial (rows) and optimal partition (columns)	337
25	Final image matrix of the world trade network	338
26	Monte Carlo simulation results: confidence intervals for the simple undirected blogs network	374
27	Names of colors in Pajek	413

Preface to the Third Edition

Two major developments in program Pajek required a new edition of this book: a fundamental redesign of the menu structure and the capability to analyze much larger networks, containing billions of vertices. A proliferation of methods for analyzing a single network necessitated a reorganization of Pajek's menu structure, in particular the former *Net* menu, now called the *Network* menu. The new *Network* menu contains submenus for commands that apply to a particular type of network: a two-mode network, multiple relations network, acyclic network, temporal network, and signed network. It is much easier now to find analyses for special networks. Because the *Network* menu is used most intensively, we had to adjust most of the commands in the Application sections of this book.

Pajek's capability to analyze much larger networks is the second major development. As a result of changes in the Windows[®] operating system, Pajek can now handle networks with nearly one billion vertices. For even larger networks, PajekXXL and Pajek3XL have been developed, which can handle up to two and ten billion vertices respectively. PajekXXL and Pajek3XL have the same user interface as Pajek; if you can work with Pajek, you can also work with PajekXXL and Pajek3XL. What you need to know is that the latter two programs offer a very limited set of analyses to describe and partition huge networks. Subnetworks extracted from a huge network can be sent directly to (regular) Pajek for further analysis. Appendix A1.5 explains all of this.

Questions on how to install and use Pajek on Mac OS X have been asked repeatedly. This new edition brings an additional appendix (Appendix 3) containing detailed instructions on installing Pajek under Mac OS X. We hope that Mac users will find out that installing and running Pajek on Mac OS X is not a difficult technical problem: Installation takes only few additional minutes compared to installation under native Windows. When Pajek is installed, running Pajek under Mac OS X is the same as running it under Windows.

Finally, we took the opportunity to include some analyses requested by Pajek users. Chapter 1 now includes Pivot MDS and VOS mapping for network layout as well as the correlation between layout coordinates and network geodesics as a measure of layout performance. Chapter 2 introduces partitions on vertex labels using regular expressions and marking partition clusters with Unicode symbols in the Draw screen. It also discusses interactive FishEye magnification and the Adjusted Rand Index. Relaxed balance is explained in Chapter 4; community detection (Louvain Method and VOS Clustering) and the E-I Index appear in Chapter 5. Chapter 6 includes (degree) assortativity and the assortativity coefficient. A collection of main path methods (including key-route searches) and preprint transformation are explained and applied in Chapter 11. Appendices A1 and A2 have been updated, now containing goodies such as dragging and dropping data to a Pajek window, sending Pajek objects to Excel[®], defining colors and transparency of vertices and lines, using Unicode symbols and additional vertex shapes (e.g., man, woman, and house), tooltips for vertex labels, drawing curved lines, and so on. We hope that you will continue enjoying social network analysis with Pajek.

The webpage to the third edition of this book (<http://mrvar.fdv.uni-lj.si/pajek/be3.htm>, mirror <http://mrvar2.fdv.uni-lj.si/pajek/be3.htm>) contains the example data sets, helper programs, and other online documents referenced in this book.

Preface to the Second Edition

I go with him out in a shed in back and see he is selling a whole Harley machine in used parts, except for the frame, which the customer already has. He is selling them all for \$125. Not a bad price at all.

Coming back I comment, “He’ll know something about motorcycles before he gets *those* together.”

Bill laughs. “And that’s the best way to learn, too.”

Robert M. Pirsig, *Zen and the Art of Motorcycle Maintenance*

To some of its readers, this book is an introduction to social network analysis; to other readers, it is a manual to Pajek software (<http://mrvar.fdv.uni-lj.si/pajek>). To us, it is both. As Patrick Doreian argued in his review of our book (In: *Social Networks* 28 [2006] 269–274), an understanding of social network analysis is required for proper use of Pajek and, vice versa, understanding the concepts and logic of Pajek fosters comprehension of network concepts. In this second edition, we have aimed to strengthen both aspects, updating the discussion of the Pajek interface and commands to include several capabilities that have been implemented since we submitted the text of the first edition, such as multiplex networks (Section 1.3.1), eigenvector centrality (Section 6.5), matrix multiplication (Section 11.3), and using Pajek output in R (Chapters 5 and 13). The new capabilities cover some important advances in social network analysis, including random graph models to which we have dedicated a new chapter.

We expanded the Further Reading sections with references to seminal, much cited texts. This should allow the reader to trace the literature on the selected topic in bibliographic and citation databases. For more comprehensive lists of literature, we refer to two other volumes in this series: S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications* (Cambridge: Cambridge University Press, 1994) and P. J. Carrington, J. Scott, and S. Wasserman, *Models and Methods in Social Network*

Analysis (Cambridge: Cambridge University Press, 2005). A concise history of social network analysis is published by L. C. Freeman, *The Development of Social Network Analysis. A Study in the Sociology of Science* (Vancouver, Canada: Empirical Press, 2004).

We hope that this second edition will continue to stimulate analysts to sharpen their understanding of social networks and expand their command of network analytic tools.

Preface to the First Edition

In the social sciences, social network analysis has become a powerful methodological tool alongside statistics. Network concepts have been defined, tested, and applied in research traditions throughout the social sciences, ranging from anthropology and sociology to business administration and history.

This book is the first textbook on social network analysis integrating theory, applications, and professional software for performing network analysis. It introduces structural concepts and their applications in social research with exercises to improve skills, questions to test the understanding, and case studies to practice network analysis. In the end, the reader has the knowledge, skills, and tools to apply social network analysis.

We stress learning by doing: Readers acquire a feel for network concepts by applying network analysis. To this end, we make ample use of professional computer software for network analysis and visualization: Pajek. This software, operating under Windows 95 and later, and all example data sets are provided on a Web site (<http://vlado.fmf.uni-lj.si/pub/networks/book/>) dedicated to this book. All the commands that are needed to produce the graphical and numerical results presented in this book are extensively discussed and illustrated. Step by step, the reader can perform the analyses presented in the book.

Note, however, that the graphical display on a computer screen will never exactly match the printed figures in this book. After all, a book is not a computer screen. Furthermore, newer versions of the software will appear, with features that may differ from the descriptions presented in this book. We strongly advise using the version of Pajek software supplied on the book's Web site (<http://vlado.fmf.uni-lj.si/pub/networks/book/>) while studying this book and then updating to a newer version of Pajek afterward, which can be downloaded from <http://vlado.fmf.uni-lj.si/pub/networks/pajek/default.htm>.

Overview

This book contains five sections. The first section (Part I) presents the basic concepts of social network analysis. The next three sections present the three major research topics in social network analysis: cohesion (Part II), brokerage (Part III), and ranking (Part IV). We claim that all major applications of social network analysis in the social sciences relate to one or more of these three topics. The final section discusses an advanced technique (*viz.*, blockmodeling), which integrates the three research topics (Part V).

The first section, titled Fundamentals, introduces the concept of a network, which is obviously the basic object of network analysis, and the concepts of a partition and a vector, which contain additional information on the network or store the results of analyses. In addition, this section helps the reader get started with Pajek software.

Part II on cohesion consists of three chapters, each of which presents measures of cohesion in a particular type of network: ordinary networks (Chapter 3), signed networks (Chapter 4), and valued networks (Chapter 5). Networks may contain different types of relations. The ordinary network just shows whether there is a tie between people, organizations, or countries. In contrast, signed networks are primarily used for storing relations that are either positive or negative such as affective relations: liking and disliking. Valued networks take into account the strength of ties, for example, the total value of the trade from one country to another or the number of directors shared by two companies.

Part III on brokerage focuses on social relations as channels of exchange. Certain positions within the network are heavily involved in the exchange and flow of information, goods, or services; whereas others are not. This is connected to the concepts of centrality and centralization (Chapter 6) or brokers and bridges (Chapter 7). Chapter 8 discusses an important application of these ideas, namely, the analysis of diffusion processes.

The direction of ties (*e.g.*, who initiates the tie) is not very important in the section on brokerage, but it is central to ranking, presented in Part IV. Social ranking, it is assumed, is connected to asymmetric relations. In the case of positive relations, such as friendship nominations or advice seeking, people who receive many choices and reciprocate few choices are deemed as enjoying more prestige (Chapter 9). Patterns of asymmetric choices may reveal the stratification of a group or society into a hierarchy of layers (Chapter 10). Chapter 11 presents a particular type of asymmetry, namely, the asymmetry in social relations caused by time: genealogical descent and citation.

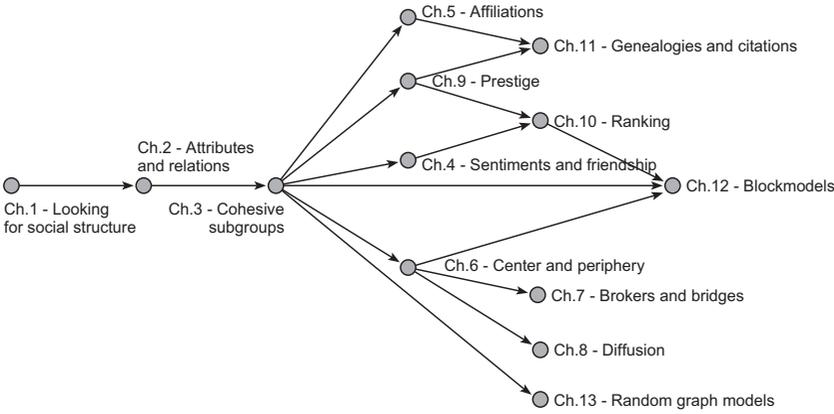


Figure 1. Dependencies between the chapters (for the second and third editions).

The final section, Part V, on roles concentrates on rather dense and small networks. This type of network can be visualized and stored efficiently by means of matrices. Blockmodeling is a suitable technique for analyzing cohesion, brokering, and ranking in dense, small networks. It focuses on positions and social roles (Chapter 12).

The book is intended for researchers and managers who want to apply social network analysis and for courses on social network analysis in all social sciences as well as other disciplines using social methodology (e.g., history and business administration). Regardless of the context in which the book is used, Chapters 1, 2, and 3 must be studied to understand the topics of subsequent chapters and the logic of Pajek. Chapters 4 and 5 may be skipped if the researcher or student is not interested in networks with signed or valued relations, but we strongly advise including them to be familiar with these types of networks. In Parts III (Brokering) and IV (Ranking), the first two chapters present basic concepts and the third chapter focuses on particular applications.

Figure 1 shows the dependencies among the chapters of this book. To study a particular chapter, all preceding chapters in this flowchart must have been studied before. Chapter 10, for instance, requires understanding of Chapters 1 through 4 and 9. Within the chapters, there are no sections that can be skipped.

In an undergraduate course, Part I and II should be included. A choice can be made between Part III and Part IV; or, alternatively, just the first chapter from each section may be selected. Part V on social roles and blockmodeling is quite advanced and more appropriate for a

postgraduate course. For managerial purposes, Part III is probably more interesting than Part IV.

Justification

This book offers an introduction to social network analysis, which implies that it covers a limited set of topics and techniques, which we feel a beginner must master to be able to find his or her way in the field of social network analysis. We have made many decisions about what to include and what to exclude, and we want to justify our choices now.

As reflected in the title of this book, we restrict ourselves to exploratory social network analysis. The testing of hypotheses by means of statistical models falls outside the scope of this book. In social network analysis, hypothesis testing is important but complicated; it deserves a book on its own. Aiming our book at people who are new to social network analysis, our first priority is to have them explore the structure of social networks to give them a feel for the concepts and applications of network analysis. Exploration involves visualization and manipulation of concrete networks, whereas hypothesis testing boils down to numbers representing abstract parameters and probabilities. In our view, exploration yields the intuitive understanding of networks and basic network concepts that are a prerequisite for well-considered hypothesis testing.

From the vast array of network analytic techniques and indices we discuss only a few. We have no intention of presenting a survey of all structural techniques and indices because we fear that the readers will not be able to see the forest for the trees. We focus on as few techniques and indices as are needed to present and measure the underlying concept. With respect to the concept of cohesion, for instance, many structural indices have been proposed for identifying cohesive groups: n -cliques, n -clans, n -clubs, m -cores, k -cores, k -plexes, lambda sets, and so on. We discuss only components, k -cores, 3-cliques, and m -slices (m -cores) because they suffice to explain the basic parameters involved: density, connectivity, and strength of relations within cohesive subgroups.

Our choice is influenced by the software that we use because we have decided to restrict our discussion to indices and techniques that are incorporated in this software. Pajek software is designed to handle very large networks (up to millions of vertices). Therefore, this software package concentrates on efficient routines, which are capable of dealing with large networks. Some analytical techniques and structural indices are known to be inefficient (e.g., the detection of n -cliques), and for others no efficient algorithm has yet been found or implemented. This limits our options; we present only the detection of small cliques (of size 3), and we cannot extensively discuss an important concept such as k -connectivity. In

summary, this book is neither a complete catalogue of network analytic concepts and techniques nor an exhaustive manual to all commands of Pajek. It offers just enough concepts, techniques, and skills to understand and perform all major types of social network analysis.

In contrast to some other handbooks on social network analysis, we minimize mathematical notation and present all definitions verbatim. There are no mathematical formulae in the book. We assume that many students and researchers are interested in the application of social network analysis rather than in its mathematical properties. As a consequence, and this may be very surprising to seasoned network analysts, we do not introduce the matrix as a data format and display format for social networks until the end of the book.

Finally, there is a remark on the terminology used in the book. Social network analysis derives its basic concepts from mathematical graph theory. Unfortunately, different “vocabularies” exist within graph theory, using different concepts to refer to the same phenomena. Traditionally, social network analysts have used the terminology employed by Frank Harary, for example, in his book *Graph Theory* (Reading: Addison-Wesley, 1969). We choose, however, to follow the terminology that prevails in current textbooks on graph theory, for example, R. J. Wilson’s *Introduction to Graph Theory* (Edinburgh: Oliver and Boyd, 1972; published later by Wiley, New York). Thus, we hope to narrow the terminological gap between social network analysis and graph theory. As a result, we speak of a vertex instead of a node or a point, and some of our definitions and concepts differ from those proposed by Frank Harary.

Acknowledgments

The text of this book has benefited from the comments and suggestions from our students at the University of Ljubljana and the Erasmus University Rotterdam, who were the first to use it. In addition, Michael Frishkopf and his students of musicology at the University of Alberta gave us helpful comments. Mark Granovetter, who welcomed this book to his series, and his colleague Sean Farley Everton have carefully read and commented on the chapters. In many ways, they have helped us make the book more coherent and understandable to the reader. We are also very grateful to an anonymous reviewer, who carefully scrutinized the book and made many valuable suggestions for improvements. Ed Parsons (Cambridge University Press) and Nancy Hulan (TechBooks) helped us through the production process. Finally, we thank the participants of the workshops we conducted at the Sunbelt International Conference on Social Network Analysis in New Orleans (XXII) and Cancun (XXIII) for their encouraging reactions to our manuscript.

Most data sets that are used in this book have been created from sociograms or listings printed in scientific articles and books. Notwithstanding our conviction that reported scientific results should be used and distributed freely, we have tried to trace the authors of these articles and books and ask for their approval. We are grateful to have obtained explicit permission for using and distributing the data sets from them. Authors or their representatives whom we have not reached are invited

Part I

Fundamentals

Social network analysis focuses on ties among, for example, people, groups of people, organizations, and countries. These ties combine to form networks, which we will learn to analyze. The first part of this book introduces the concept of a social network. We discuss several types of networks and the ways in which we can analyze them numerically and visually with the computer software program Pajek, which is used throughout this book. After studying Chapters 1 and 2, you should understand the concept of a social network and be able to create, manipulate, and visualize a social network with the software presented in this book.

Looking for Social Structure

1.1 Introduction

The social sciences focus on structure: the structure of human groups, communities, organizations, markets, society, or the world system. In this book, we conceptualize social structure as a network of social ties. Social network analysts assume that interpersonal ties matter, as do ties among organizations or countries, because they transmit behavior, attitudes, information, or goods. Social network analysis offers the methodology to analyze social relations; it tells us how to conceptualize social networks and how to analyze them.

In this book, we present the most important methods of exploring social networks, emphasizing visual exploration. Network visualization has been an important tool for researchers from the very beginning of social network analysis. This chapter introduces the basic elements of a social network and shows how to construct and draw a social network.

1.2 Sociometry and Sociogram

The basis of social network visualization was laid by researchers who called themselves sociometrists. Their leader, J. L. Moreno, founded a social science called *sociometry*, which studies interpersonal relations. Society, they argued, is not an aggregate of individuals and their characteristics, as statisticians assume, but a structure of interpersonal ties. Therefore, the individual is not the basic social unit. The social atom consists of an individual and his or her social, economic, or cultural ties. Social atoms are linked into groups, and, ultimately, society consists of interrelated groups.

From their point of view, it is understandable that sociometrists studied the structure of small groups rather than that of society at large.

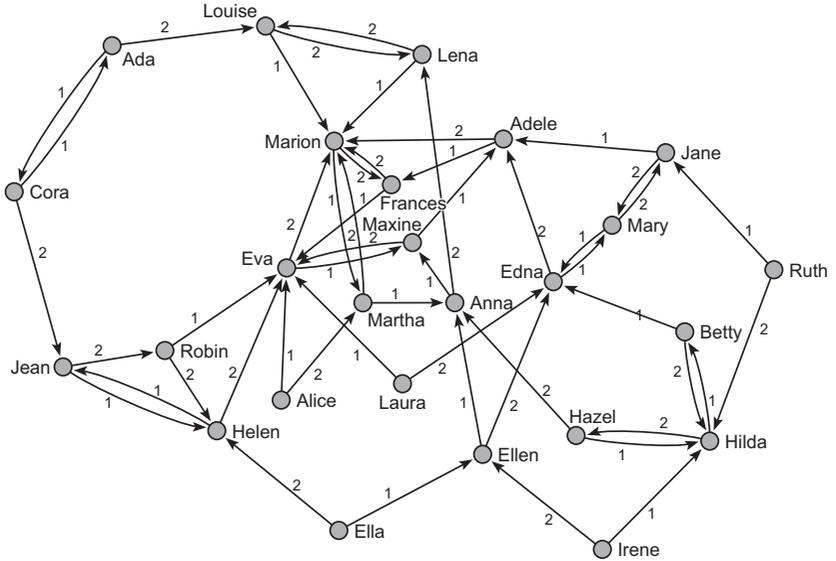


Figure 2. Sociogram of dining-table partners.

In particular, they investigated social choices within a small group. They asked people questions such as, “Whom would you choose as a friend [colleague, advisor, etc.]?” This type of data has since been known as *sociometric choice*. In sociometry, social choices are considered the most important expression of social relations.

Figure 2 presents an example of sociometric research. It depicts the choices of twenty-six girls living in one “cottage” (dormitory) at a training school in New York State. The girls were asked to choose the girls they liked best as their dining-table partners. First and second choices are selected only. (Here and elsewhere, a reference on the source of the data can be found in “Further Reading” at the end of each chapter.)

Figure 2 is an example of a sociogram, which is a graphical representation of group structure. The sociogram is among the most important instruments that originated in sociometry, and it is the basis for the visualization of social networks. You have most likely already “read” and understood the figure without needing the following explanation, which illustrates its visual appeal and conceptual clarity. In this sociogram, each girl in the dormitory is represented by a circle. For the sake of identification, the girls’ names are written next to the circles. Each arc (arrow) represents a choice. The girl who chooses a specific peer as a dining-table companion sends an arc toward her. Irene (in the bottom right of the figure), for instance, chose Hilda as her favorite dining-table partner and Ellen as her second choice, as indicated by the numbers labeling each arrow.

A sociogram depicts the structure of ties within a group. This example shows not only which girls are popular, as indicated by the number of choices they receive, but also whether the choices come from popular or unpopular girls. For example, Hilda receives four choices from Irene, Ruth, Hazel, and Betty, and she reciprocates the last two choices. However, none of these four girls is chosen by any of the other girls. Therefore, Hilda is located at the margin of the sociogram, whereas Frances, who is chosen only twice, is more central because she is chosen by Adele and Marion, two of the “popular” girls. A simple count of choices does not reveal this, whereas a sociogram does.

The sociogram has proved to be an important analytical tool that helped reveal several structural features of social groups. In this book, we make ample use of it.

1.3 Exploratory Social Network Analysis

Sociometry is not the only tradition in the social sciences that focuses on social ties. Without going into historical detail (see the “Further Reading” section for references on the history of social network analysis), we may note that scientists from several social sciences have applied network analysis to different kinds of social relations and social units. Anthropologists study kinship relations, friendship, and gift giving among people rather than sociometric choice; social psychologists focus on affections; political scientists study power relations among people, organizations, or nations; economists investigate trade and organizational ties among firms. In this book, the word *actor* refers to a person, organization, or nation that is involved in a social relation. We may say that social network analysis studies the social ties among actors.

The *main goal* of social network analysis is detecting and interpreting patterns of social ties among actors.

This book focuses on exploratory social network analysis only. This means that we have no specific hypotheses about the structure of a network beforehand that we can test. For example, a hypothesis on the dining-table partners network could predict a particular rate of mutual choices (e.g., one of five choices will be reciprocated). This hypothesis must be grounded in social theory and prior research experience. The hypothesis can be tested provided that an adequate statistical model is available.

We use no hypothesis testing in this book (except for the last chapter) because we cannot assume prior research experience in an introductory course book, and because the statistical models involved are complicated.

Therefore, we adopt an exploratory approach, which assumes that the structure or pattern of ties in a social network is meaningful to the members of the network and, hence, to the researcher. Instead of testing pre-specified structural hypotheses, we explore social networks for meaningful patterns.

For similar reasons, we pay no attention to the estimation of network features from samples. In network analysis, estimation techniques are even more complicated than estimation in statistics, because the structure of a random sample seldom matches the structure of the overall network. It is easy to demonstrate this. For example, select five girls from the dining-table partner network at random and focus on the choices among them. You will find fewer choices per person than the two choices in the overall network for the simple reason that choices directed to girls outside the sample are neglected. Even in this simple respect, a sample is not representative of a network.

We analyze entire networks rather than samples. However, what is the entire network? Sociometry assumes that society consists of interrelated groups, so a network encompasses society at large. Research on the so-called Small World problem suggested that ties of acquaintance connect us to almost every human being on Earth in six or seven steps (i.e., with five or six intermediaries), so our network eventually covers the entire world population, which is clearly too large a network to be studied. Therefore, we must use an artificial criterion to delimit the network we are studying. For example, we may study the girls of one dormitory only. We do not know their preferences for table partners in other dormitories. Perhaps Hilda is the only vegetarian in a group of carnivores, and she prefers to eat with girls of other dormitories. If so, including choices between members of different dormitories will alter Hilda's position in the network tremendously.

Because boundary specification may seriously affect the structure of a network, it is important to consider it carefully. Use substantive arguments to support your decision of whom to include in the network and whom to exclude.

Exploratory social network analysis consists of four parts: the definition of a network, network manipulation, determination of structural features, and visual inspection. In the following subsections, we present an overview of these techniques. This overview serves to introduce basic concepts in network analysis and to help you get started with the software used in this book.

1.3.1 Network Definition

To analyze a network, we must first have one. What is a network? Here, and elsewhere, we use a branch of mathematics called *graph theory* to

define concepts. Most characteristics of networks that we introduce in this book originate from graph theory. Although this is not a course in graph theory, you should study the definitions carefully to understand what you are doing when you apply network analysis. Throughout this book, we present definitions in text boxes to highlight them.

A *graph* is a set of vertices and a set of lines between pairs of vertices.

What is a graph? A graph represents the structure of a network; all it needs for this is a set of vertices (which are also called points or nodes) and a set of lines, with each line connecting two vertices.

A *vertex* (singular of vertices) is the smallest unit in a network. In social network analysis, it represents an actor (e.g., a person, such as a girl in a dormitory; an organization; or a country). A vertex is usually identified by a number.

A *line* is a link between two vertices in a network. In social network analysis, it can be any social relation. A line is defined by its two endpoints, which are the two vertices that are *incident* with the line.

A *loop* is a special kind of line, namely, a line that connects a vertex to itself. In the dining-table partners network, loops do not occur because girls are not allowed to choose themselves as dinner-table partners. However, loops are meaningful in some kinds of networks.

A line is directed or undirected. A directed line is called an *arc*, whereas an undirected line is an *edge*. Sociometric choice is best represented by arcs, because one girl chooses another and choices need not be reciprocated (e.g., Ella and Ellen in Figure 2).

A *directed graph*, or *digraph*, contains one or more arcs. A social relation that is undirected (e.g., cooperation on school projects) is represented by an edge because both individuals are equally involved in the relation. An *undirected graph* contains no arcs: All of its lines are edges.

Formally, an arc is an ordered pair of vertices in which the first vertex is the *sender* (the *tail* of the arc) and the second the *receiver* of the tie (the *head* of the arc). An arc points *from* a sender *to* a receiver. In contrast, an edge, which has no direction, is represented by an unordered pair. It does not matter which vertex is first or second in the pair. We should note, however, that an edge is usually equivalent to a *bidirectional arc*: If Ella and Ellen cooperate (undirected), we can say that Ella cooperates with Ellen and Ellen cooperates with Ella (directed). It is important to note this, as we will see in later chapters.

The dining-table partners network has no *multiple lines* because no girl was allowed to nominate the same girl as both first and second choices. Without this restriction, which was imposed by the researcher, multiple arcs could have occurred, and they actually do occur in other social networks.

In a graph, multiple lines are allowed, but when we say that a graph is *simple*, we indicate that it has no multiple lines. In addition, a simple undirected graph contains no loops, whereas loops are allowed in a simple directed graph. It is important to remember this.

A *simple undirected graph* contains neither multiple edges nor loops.

A *simple directed graph* contains no multiple arcs.

Now that we have discussed the concept of a graph at some length, it is very easy to define a network. A network consists of a graph and additional information on the vertices or lines of the graph. We should note that the additional information is irrelevant to the structure of the network because the structure depends on the pattern of ties.

A *network* consists of a graph and additional information on the vertices or the lines of the graph.

In the dining-table partners network, the names of the girls represent additional information on the vertices that turns the graph into a network. Because of this information, we can see which vertex identifies Ella in the sociogram. The numbers printed near the arcs and edges offer additional information on the links between the girls: A 1 indicates a first choice, and a 2 represents a second choice. These are called *line values* and usually indicate the strength of a relation, which is a quantity.

Lines can also have a specific quality; for example, they can be of a particular type. All lines in the dining-table partners network are of the same kind, expressing seating preferences. We can say that they express the seating preferences *relation*. A network can, however, contain more than one relation. Perhaps we also know which of the girls cooperated on class projects. Because this information involves the same set of vertices (the girls), we can add this information as a second set of lines, that is, as a second relation to the network. This creates a *multiple relations network*, which is also called a *multiplex network*.

The dining-table partners network is clearly a network and not a graph. It is a directed simple network because it contains arcs (directed) but not multiple arcs (simple). In addition, we know that it contains just one relation and no loops. Several analytical techniques we discuss assume that loops and multiple lines are absent from a network. However, we do not always spell out these properties of the network but rather indicate whether it is simple. Take care!

Application

In this book, we learn social network analysis by doing it. We use the computer program Pajek – Slovenian for spider – to analyze and draw social networks. The website dedicated to this book (<http://mrvar.fdv.uni-lj.si/pajek/>) contains the software. We advise you to download and install Pajek on your computer (see Appendix 1 for more details) and all example data sets from this website. Store the software and data sets on the hard disk of your computer following the guidelines provided on the website. When you have done so, carry out the commands that we discuss under “Application” in each chapter. This will familiarize you with the structural concepts and with Pajek. By following the instructions under “Application” step by step, you will be able to produce the figures and results presented in the theoretical sections, unless stated differently. Sometimes, the visualizations on your computer screen will be slightly different from the figures in the book. If the general patterns match, however, you know that you are on the right track.

Some concepts from graph theory are the building blocks or *data objects* of Pajek. Of course, a network is the most important data object in Pajek, so let us describe it first. In Pajek, a network is defined in accordance with graph theory: a list of vertices and lists of arcs and edges, where each arc or edge has a value. Take a look at the partial listing of the data file for the dining-table partners network (Figure 3; note that part of the vertices and arcs are replaced by [...]). Open the styled `Dormitory.net`, which you have downloaded from the website, in a word processor program to see the entire data file.

*Network data
file*

First, the data file specifies the number of vertices. Then, each vertex is identified on a separate line by a serial number, a textual label (enclosed in quotation marks [“ ”]), and three real numbers between 0 and 1, which indicate the position of the vertex in three-dimensional space if the network is drawn. We pay more attention to these coordinates in Chapter 2. For now, it suffices to know that the first number specifies the horizontal position of a vertex (0 is at the left of the screen and 1 at the right), and the second number gives the vertical position of a vertex (0 is the top of the screen and 1 is the bottom). The text label is crucial for identification of vertices, the more so because serial numbers of vertices may change during the analysis.

The list of vertices is followed by a list of arcs. The `*Arcs` statement assigns relation number 1 (the integer after the colon) to the arcs specified in the subsequent lines, and the relation is labeled “Dining-table partner choice.” Note that labels should be enclosed in double quotation marks. Each line identifies an arc by the serial number of the sending vertex, followed by the number of the receiving vertex and the value of the arc. Just as in graph theory, Pajek defines a line as a pair of vertices. In Figure 3, the first arc represents Ada’s choice (vertex 1) of Louise (vertex 3) as a

```

*Vertices 26
  1 "Ada"           0.1646    0.2144    0.5000
  2 "Cora"          0.0481    0.3869    0.5000
  3 "Louise"        0.3472    0.1913    0.5000
  4 "Jean"          0.1063    0.5935    0.5000
[...]
 25 "Laura"         0.5101    0.6133    0.5000
 26 "Irene"         0.7478    0.8087    0.5000
*Arcs :1 "Dining-table partner choice"
  1  3  2
  1  2  1
  2  1  1
  2  4  2
[...]
*Edges :2 "Cooperation"
  1  2  1  1 "Math 2a"
  2  4  1  1 "Math 2a"
  1  4  1  1 "Math 2a"
[...]

```

Figure 3. Partial listing of a multiple relations network data file for Pajek.

dining-table partner. Louise is Ada's second choice; Cora is her first choice, which is indicated by the second arc. A list of edges is similar to a list of arcs, with the exception that the order of the two vertices that identify an edge is disregarded in computations. In this data file, edges express cooperation among girls, which is coded as relation number 2 with an appropriate label. The edges are labeled with the name of the project the girls cooperated on (see Appendix 2 for directions on adding information to lines and vertices). Note that a Pajek network file can contain several **Arcs* and **Edges* statements, and the relation number and label is not mandatory.

It is interesting to note that we can distinguish between the structural data or graph and the additional information on vertices and lines in the network data file. The graph is fully defined by the list of vertex numbers and the list of pairs of vertices, which defines its arcs and edges. This part of the data, which is printed in regular typeface in Figure 3, represents the structure of the network. The vertex labels and coordinates, the relation numbers and labels, line values and labels (in italics) specify the additional properties of vertices and lines that make these data a network. Although this information is extremely useful, it is not required: Pajek will use vertex numbers as default labels and set relation numbers and line values to 1 if they are not specified in the data file. In addition, Pajek can use several other data formats (e.g., the matrix format), which we do not discuss here. They are briefly described in Appendix 1.

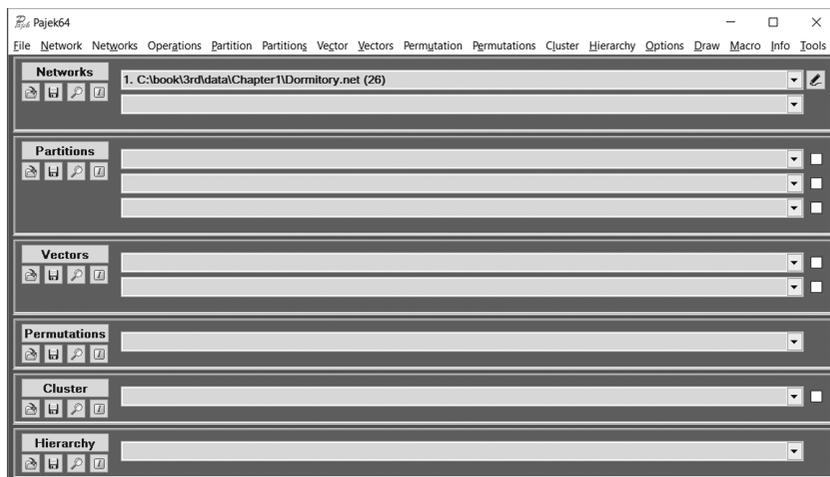


Figure 4. Main screen of Pajek.

It is possible to generate ready-to-use network files from spreadsheets and databases by exporting the relevant data in plain text format. For medium or large networks, processing the data as a relational database helps data cleaning and coding. See Appendix 1 for details.

We explain how to create a new network in Section 1.4. Let us first look at the dormitory network containing the dining-table partner choices and the cooperation among girls. First, start Pajek by double-clicking the file `Pajek.exe` on your hard disk. The computer will display the Main screen of Pajek (Figure 4). From this screen, you can open the dormitory network with the *Read* command in the *File* menu or by clicking the button with an icon of a folder under the word *Networks*. In both cases, the usual Windows® file dialog box appears in which you can search and select the file `Dormitory.net` on your hard disk, provided that you have downloaded the example data sets from the book’s website.

File>
Network>
Read

When Pajek reads a network, it displays its name in the top Network drop-down menu. This menu is a list of the networks accessible to Pajek. You can open a drop-down menu by left-clicking on the button with the triangle at the right. The network that you select in the list is shown when the list is closed (e.g., the network `Dormitory.net` in Figure 2). Notice that the number of vertices in the network is displayed in parentheses next to the name. The selected network is the *active network*, meaning that any operation you perform on a network will use this particular network. For example, if you use the *Draw* menu now, Pajek draws the dormitory network for you.

Network
drop-down
menu

The Main screen displays several more drop-down menus beneath the two network drop-down menus. Each of these menus represents a data object in Pajek: partitions (three drop-down menus), vectors (two

drop-down menus), permutations (two drop-down menus), clusters, and hierarchies. Later chapters will familiarize you with these data objects. Note that each object can be opened, saved, or edited from the *File* menu or by using the four icons to the left of a drop-down menu (see Section 1.4).

1.3.2 Manipulation

In social network analysis, it is often useful to modify a network. For instance, large networks are too big to be drawn, so we extract a meaningful part of the network that we inspect first. Visualizations work much better for small (some dozens of vertices) to medium-sized (some hundreds of vertices) networks than for large networks with thousands of vertices. When social networks contain different kinds of relations, we may focus on one relation only; for instance, we may want to study dining-table partner choices only in the dormitory network. Finally, some analytical procedures demand that complex networks with loops or multiple lines are reduced to simple graphs first.

Application

Network manipulation is a very powerful tool in social network analysis. In this book, we encounter several techniques for modifying a network or selecting a subnetwork. Network manipulation always results in a new network. In general, many commands in Pajek produce new networks or other data objects, which are stored in the drop-down menus, rather than graphical or tabular output.

Menu structure

The commands for manipulating networks are accessible from menus in the Main screen. The Main screen menus have a clear logic. Manipulations that involve one type of data object are listed under a menu with the object's name; for example, the *Network* menu contains all commands that operate on one network, and the *Networks* menu lists operations on two networks. Manipulations that need different kinds of objects are listed in the *Operations* menu. When you try to locate a command in Pajek, just consider which data objects you want to use. Pajek also groups commands that apply to a special type of network; e.g., all commands that require a multiple relations network are available in the submenu *Multiple Relations Network* of the *Network* Menu. You will learn more about other types of networks in the following chapters.

[Main]
 Network>
 Multiple
 Relations
 Network>
 Extract
 Relation(s) into
 Separate
 Network(s)

The following example highlights the use of menus in Pajek and their notation in this book. If we want to restrict the analysis to the dining partners relation (as in Figure 2), we must create a new network containing only the lines that belong to the first relation. Because this operation concerns one network and no other data objects, we must look for it in the *Network* menu. If we left-click on the word *Network* in the upper left

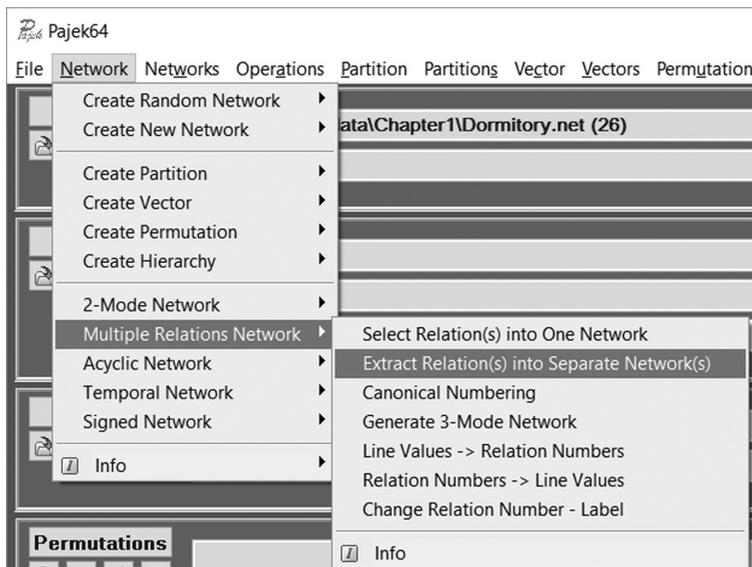


Figure 5. Menu structure in Pajek.

of the Main screen, a drop-down menu is displayed. Click on the words *Multiple Relations Network* in the drop-down menu. Thus, we reach the command allowing us to create a new network containing only one relation from the currently active network: *Extract Relation(s) into Separate Network(s)* (see Figure 5). Execute this command.

In this book, we abbreviate this sequence of commands as follows:

```
[Main] Network> Multiple Relations Network> Extract Relation(s) into
Separate Network (s)
```

The screen or window that contains the menu is presented between square brackets, and a transition to a submenu is indicated by the > symbol. The screen name is specified only if the context is ambiguous. The abbreviated command is also displayed in the margin for the purpose of quick reference. An Index of Pajek commands discussed in this book can be found near the end of this book.

In a follow-up dialog box, you can enter one relation number, several consecutive numbers (use a dash), or several nonconsecutive numbers (separate with commas). In our case, just enter 1 to obtain a new network containing only the dining-table partner relation (Figure 6). A new network named *Relation:1 [Dining-table pa]* from *N1 (26)* is added to the top *Network* drop-down menu with a serial number of 2. The original network is not changed; it is still on the list of networks in this drop-down menu.

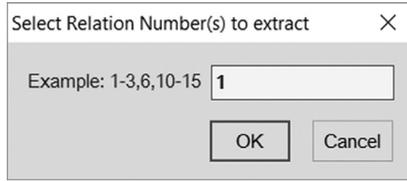


Figure 6. Dialog box in Pajek.

To demonstrate the manipulation of two networks at the same time, let us find out which girls are linked both by cooperation and dining partner choice. In other words, which lines appear on both relations? We already have a network with just the dining-table partners lines but we also need one for collaboration, which can be extracted in the same manner: Select the dormitory network in the top Network drop-down menu and follow the steps described in the previous paragraphs to extract relation number 2. If you are successful, a third network appears in this menu named Relation:2 [Cooperation] from N1 (26).

[Main]
 Network>
 Create New
 Network >
 Transform>
 Edges→Arcs

Before we can compare the two networks, we must realize that one network contains arcs (directed) whereas the other contains edges (undirected). We cannot compare arcs and edges directly, so we have to change arcs into edges or the other way around. We recommend changing the collaboration edges into *bidirectional arcs*, which can be done with the *Network> Create New Network> Transform> Edges→Arcs* command, provided that we select the collaboration network in the top drop-down menu first. When the command to change edges into arcs is executed, an information box appears asking whether a new network must be made. If the answer is “yes,” which we advise, a new network is created. Conversely, answering “no” to the question in the information box causes Pajek to change the original network.

[Main]
 Networks>
 Cross-
 Intersection>
 First

To compare two networks, we must tell Pajek which networks we want to use. There are two drop-down menus for networks (if not, click the *Networks* button on the left), so we can select one network in each. We recommend selecting the dining partners network in the top drop-down menu (to Pajek, this is the first network) and the collaboration network with edges changed into arcs in the menu directly below (the second network to Pajek). Now we can execute commands of the *Networks* menu, including basic set theoretic operations such as the union, intersection, and difference of the two sets of lines. We need the intersection, which collects all lines that appear in both networks: pairs of girls linked both by dining-table choice and collaboration. Pajek contains two versions of this command: *Intersection of Multiple Relations Networks* and *Cross-Intersection*. The *Intersection of Multiple Relations Networks* command selects lines only if they appear in both networks and have the same

relation number. The *Cross-Intersection* command ignores relation numbers, which is what we need here because our two networks have different relation numbers.

The *Cross-Intersection* submenu contains several options for combining the values of the lines that appear in both networks. Note that this set operation replaces two arcs in the original networks by one arc in a new network, so it must know what to do with the original line values. In our case, all collaboration lines have unit value, which is uninformative. Therefore, we advise retaining the values of the dining partner lines. Because the dining partner network was in the top *Network* drop-down menu when we created the intersection, we select the command *First* from the *Cross-Intersection* submenu to retain the line values of the dining partner network, which express the first and second choice for dining partners. Do you want to have a look at the result? Select the *Network* command in the *Draw* menu. This command is discussed in detail in Section 1.3.4., but why wait?

The procedure just described is characteristic for manipulating network data with Pajek. There are many basic commands for transforming and combining networks in Pajek and a multitude of results can be obtained by combining these commands. But it is up to the user (that is you) to decide on the commands that you need for obtaining the desired outcome.

Exercise I

Instead of changing the collaboration edges into arcs, change all dining partner arcs into edges with the command summarized in the left margin (select 0 in the *Remove multiple lines?* dialog box, which creates an edge for each arc in the network). Why is this command part of the *Network* menu? Have a look at Figure 2; how many pairs of girls do you expect to be linked by multiple lines in the new network? (The answers to the exercises are listed in Section 1.9.)

Network>
Create New
Network>
Transform>
Arcs→Edges>
All

1.3.3 Calculation

In social network analysis, many structural features have been quantified (e.g., an index that measures the centrality of a vertex). Some measures pertain to the entire network, whereas others summarize the structural position of a subnetwork or a single vertex. Calculation outputs a single number in the case of a network characteristic and a series of numbers in the case of subnetworks and vertices.

Exploring network structure by calculation is much more concise and precise than visual inspection. However, structural indices are sometimes abstract and difficult to interpret. Therefore, we use both visual inspection of a network and calculation of structural indices to analyze network structure.

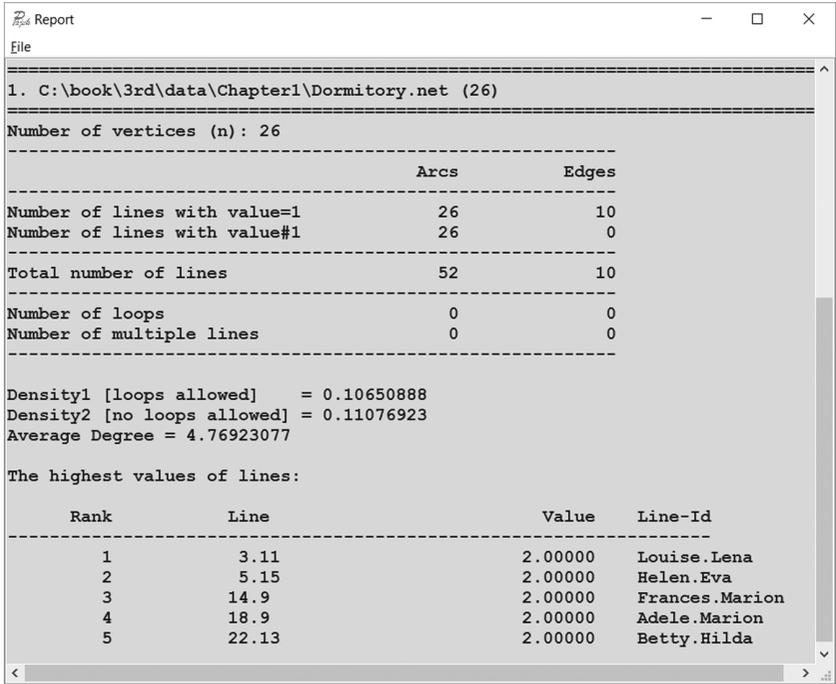


Figure 7. Report screen in Pajek.

Application

*Info> Show
Report
Window*

In Pajek, results of calculations and other kinds of feedback to the user are automatically reported in a separate window that we call the Report screen. If you closed the Report screen or if it is hidden behind other screens, you can show it again with the *Show Report Window* command in the *Info* menu of Pajek's Main screen.

The Report screen displays numeric results that summarize structural features as a single number, a frequency distribution, or a cross-tabulation. Calculations that assign a value to each vertex are not reported in this screen. They are stored as data objects in Pajek, notably as partitions and vectors (see Chapter 2). The Report screen displays text but no network drawings. The contents of the Report screen can be saved as a text file from its *File* menu.

*Network>
Info> General*

The Report screen depicted in Figure 7 shows the number of vertices, edges, and arcs in the dormitory network. This is general information on the network that is provided by the command *Network> Info> General* (as you know now, this means the *General* command within the *Info* submenu of the *Network* menu) provided that the dormitory

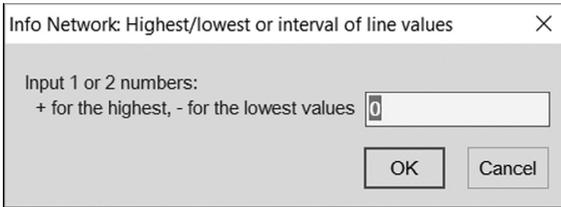


Figure 8. Dialog box of *Network > Info > General* command.

network was selected in the top *Network* drop-down menu. In addition to the number of vertices, edges, and arcs, the screen shows the number of multiple lines and loops, the average degree, and two indices of network density that are explained in Chapter 3. Also, this command displays the number of lines requested in the dialog box depicted in Figure 8.

In this example, we typed a 5 in the dialog box, so the report screen shows the five lines with the highest line values that are second choices in the dining-table partners network. In the Report screen, each line is described by its rank according to its line value, a pair of vertex numbers, the line value, and a pair of vertex labels. Hence, the first line in Figure 7 represents the arc from Louise (vertex 3) to Lena (vertex 11), who is Louise's second choice.

Basic information on different relations within a network is easily retrieved with the *Network > Multiple Relations Network > Info* command. This command produces a table with a row for each relation in the network, specifying the number of arcs and edges as well as the relation label, if any. Make sure that the original multiple relations network (*Dormitory.net*) is selected in the top *Network* drop-down menu.

Network >
Multiple
Relations
Network > Info

Exercise II

Can you see the number of first choices from the listing in Figure 7 and does this number surprise you? Which number should you enter in the dialog box depicted in Figure 8 to get a list of all arcs representing first choices?

1.3.4 Visualization

The human eye is trained in pattern recognition. Therefore, network visualizations help to trace and present patterns of ties. In Section 1.2, we presented the sociogram as the first systematic visualization of a social network. It was the sociometrists' main tool to explore and understand the structure of ties in human groups. In books on graph theory, visualizations are used to illustrate concepts and proofs. Visualizations

facilitate an intuitive understanding of network concepts, so we use them frequently.

Our eyes are easily fooled, however. A network can be drawn in many ways, and each drawing stresses different structural features. Therefore, the analyst should rely on systematic rather than ad hoc principles for network drawing. In general, we should use automatic procedures, which generate an optimal *layout of the network*, when we want to explore network structure. Subsequently, we may edit the automatically generated layout manually if we want to present it.

Some basic principles of network drawing should be observed. The most important principle states that the distance between vertices should express the strength or number of their ties as closely as possible. In a map, the distance between cities matches their geographical distance. In psychological charts, spatial proximity of objects usually expresses perceived similarity. Because social network analysis focuses on relations, a drawing should position vertices according to their ties: Vertices that are connected should be drawn closer together than vertices that are not related. A good drawing minimizes the variation in the length of lines. In the case of lines with unequal values, line length should be proportional to line value.

The legibility of a drawing poses additional demands, which are known as *graph drawing aesthetics*. Vertices or lines should not be drawn too closely together, and small angles between lines that are incident with the same vertex should be avoided. Distinct vertices and distinct lines should not merge into one lump. Vertices should not be drawn on top of a line that does not connect this vertex to another vertex. The number of crossing lines should be minimized because the eye tends to see crossings as vertices.

Application

Draw screen

Pajek offers many ways to draw a network. It has a separate window for drawing, which is accessed from the *Draw > Network* menu or by clicking the *Draw* button at the right-hand side of the *Network* drop-down menu (the picture of the drawing pencil) in the Main screen. The Draw screen has an elaborate menu of choices (see Figure 9), some of which are presented in later chapters. Use the index of Pajek commands in this book to find them. We discuss the most important commands now to help you draw your first network.

It is very important to note the following. Figure 9 is a screen shot of the Draw screen in Pajek, showing the dormitory network. This figure does not necessarily exactly match the Draw screen that you see on your computer. First, the size of the computers' Draw screens may be different. Second, on your computer screen the vertices are most probably orange

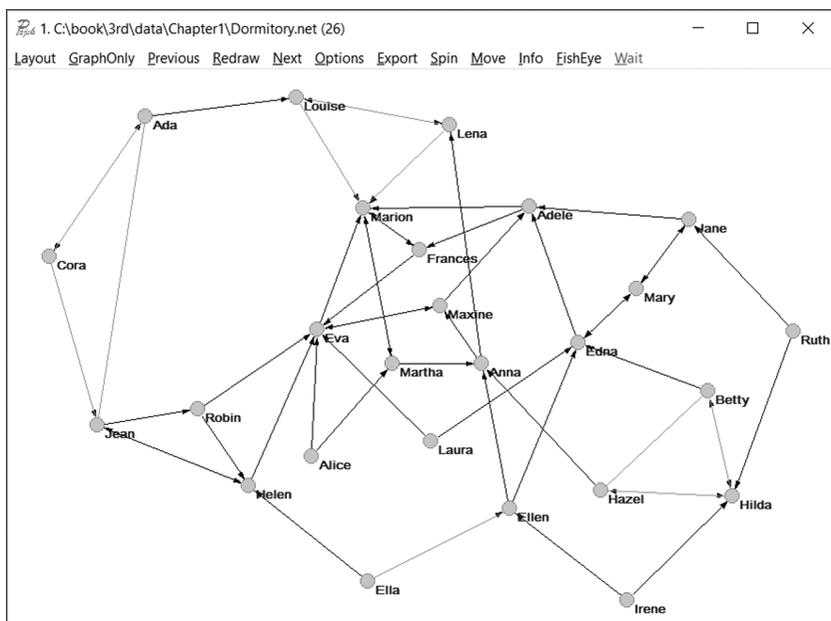


Figure 9. Draw screen in Pajek.

instead of gray. In this book, colors are replaced by grays; we discuss this matter in greater detail in Chapter 2.

Next, compare Figure 9 to Figure 2. Although the arcs in the two figures display the same network, there are differences. Reciprocal arcs, for example, are drawn as straight arcs with two heads in Figure 9, whereas they are two arcs in Figure 2. Figure 2 is exported to a format (Encapsulated PostScript) that is optimal for publication in a book. It also allows manual editing (e.g., moving the vertex labels a little bit so as not to obscure arcs or headings of arcs). In Section 1.3.4.3 and Appendix 2, you can learn more about this format and how to edit it. The Draw screen of Pajek as shown in Figure 9, however, is meant for efficient drawing and redrawing of networks, so it is less detailed.

Keep in mind that the level of detail in the book is greater than that on the screen. This does not mean that the layout of the network in the Draw screen is wrong and that you followed the wrong procedure or applied the wrong commands.

Figure 9 displays the dining-table partner choices (arcs) and collaboration (edges). The two relations have different colors (black and gray), because the options for the colors of arcs and edges are set to *Relation Number* in the Draw screen. The color for a relation number can be

```
[Draw]
Options>
Colors>
Edges>
Relation
Number
```

[Draw]
Options>
Colors> Arcs>
Relation
Number

[Draw]
Options>
Colors>
Relation Colors

picked with the *Options> Colors> Relation Colors* command, which is analogous to the *Options> Colors> Partition Colors> for Vertices* command discussed in detail in Section 2.3. As you can see, edges are drawn exactly on top of arcs and we can see only the arrowheads of the latter. Pajek draws different relations on top of each other in the order in which it encounters them in the network data file. Our data file lists the arcs first, followed by the edges (see Figure 3), so the edges are on top. Multiple lines and multiple relations require some ingenuity to be displayed in drawings.

[Draw]
Options>
Lines> Mark
Lines> with
Values, with
Labels, No

Figure 2 displays line values, whereas Figure 9 does not. Do not worry about such differences unless the text is explicitly discussing this aspect. There are many options for changing features of the layout in the Draw screen, which may be turned on or off. The line values, for instance, can be displayed by selecting the option *with Values* in the *Mark Lines* submenu of the *Lines* command in the *Options* menu of the Draw screen. Activate this option now – pressing *Ctrl-v* will have the same effect (see Appendix 4) – and you will see the line values on your computer screen. Line values will now be shown until you select another option: *with Labels* (*Ctrl-b*), which will show the line labels, or *No*, which removes line values or labels. Note the difference between a command and an option in Pajek: A command is executed once, whereas an option remains effective until it is turned off or changed.

1.3.4.1 Automatic Drawing

Automated procedures for finding an optimal layout are a better way to obtain a basic layout than manual drawing, because the resulting picture depends less on the preconceptions and misconceptions of the investigator. In addition, automated drawing is much faster and quite spectacular because the drawing evolves before your eyes.

Layout>
Energy menu

In Pajek, several commands for automatic layout are implemented. Two commands are accessible from the *Layout> Energy* menu, and we refer to them as energy commands. Both commands move vertices to locations that minimize the variation in line length. Imagine that the lines are springs pulling vertices together, though never too close. The energy commands “pull” vertices to better positions until they are in a state of equilibrium. Therefore, these procedures are known as *spring embedders*.

Layout>
Energy>
Starting
positions:
Random,
Circular, Given
xy, Given z

The relocation technique used in both automatic layout commands has some limitations. First, the results depend on the starting positions of vertices. Different starting positions may yield different results. Most results will be quite similar, but results can differ markedly. The *Starting Positions* submenu on the *Layout> Energy* menu gives you control over the starting positions. You can choose random and circular starting positions, or you can use the present positions of vertices as their starting positions: option



Figure 10. *Continue* dialog box.

Given xy , where x and y refer to the (horizontal and vertical) coordinates of vertices in the plane of the Draw screen. The fourth option (*Given z*) refers to the third dimension, which we present in Chapter 5.

The second limitation of the relocation technique is that it stops if improvement is relatively small or if the user so requests in a dialog box (Figure 10). This means that automatic layout generation outputs a drawing that is very good but not perfect. Manual improvements can be made. However, make small adjustments only to reduce the risk of discovering a pattern that you introduced to the drawing yourself. To improve a drawing, it is usually worth the effort to repeat an energy command a couple of times with given starting positions.

Because of these limitations, take the following advice to heart: *Never rely on one run of an energy command*. Experiment with both commands and manual adjustment until you obtain the same layout repeatedly.

The first energy command is named Kamada–Kawai after its authors. Suboption *Free* produces regularly spaced results for connected networks that are not very large. If the network is disconnected, use the *Separate Components* (or *Ctrl-k*) suboption that tiles the components. Kamada–Kawai seems to produce more stable results than the other energy command (Fruchterman–Reingold), but it is a little slower and it should not be applied to networks containing more than some thousands of vertices. With Kamada–Kawai, commands *Fix First and Last Vertex* and *Fix One Vertex in the Middle* enable you to fix vertices that should not be relocated. The first command fixes the two vertices with the lowest and highest serial numbers. This is very useful if these vertices represent the sender and target in an information network. The second command allows you to specify one vertex that must be placed in the middle of the drawing.

The second energy command, which is called Fruchterman–Reingold, is faster and works with larger networks. This command also separates unconnected parts of the network nicely. It can generate two- and three-dimensional layouts, as indicated by commands *2D* and *3D* in the submenu. We discuss three-dimensional visualizations in Chapter 5.

The third entry in the submenu, which is labeled *Factor*, allows the user to specify the optimal distance between vertices in a drawing energized with Fruchterman–Reingold. This command displays a dialog box

```
Layout>
Energy>
Kamada-
Kawai>
Separate
Components,
Free, Fix First
and Last
Vertex, Fix
One Vertex in
the Middle
```

```
Layout>
Energy>
Fruchterman-
Reingold> 2D,
3D
```

```
Layout>
Energy>
Fruchterman-
Reingold>
Factor
```

asking for a positive number. A low number yields small distances between vertices, so many vertices are placed in the center of the plane. A high number pushes vertices out of the center toward positions on a circle. An optimal distance of 1 is a good starting point. Try a smaller distance if the center of the drawing is quite empty, but a higher distance if the center is too crowded. Note that you have to reissue the Fruchterman–Reingold energy command to use the new distance settings.

Each energy command has strong and weak points. Sometimes it is better to start with Fruchterman–Reingold and using several optimal distances until a stable result appears. The drawing can then be improved using Kamada–Kawai with the actual location of vertices as starting positions. Finally, improve the drawing by manual editing, which we discuss in Section 1.3.4.2.

```
Layout> Pivot
      MDS>
      Random
Pivots> 2D,
      3D
```

Pajek offers two automatic drawing methods that are not based on minimizing energy. The first one is Pivot MDS. It can produce 2D and 3D layouts of connected or disconnected networks. We advise using it when you want to get some basic impression of large sparse networks; it can lay out a network containing hundreds of thousands of vertices in few seconds. When you apply this method you are asked for the number of pivots. Note that a larger number of pivots produces better drawings but demands more time. It is possible to select a set of vertices that Pajek must use as pivots (*Pivots from Cluster*) but this requires understanding of the cluster data object that we will introduce no earlier than in Chapter 12.

```
Layout> VOS
Mapping> 2D,
      3D
```

Whereas Pivot MDS is suitable for drawing large sparse networks, VOS Mapping gives better results for dense networks. This method can also handle networks where values of lines express similarities, e.g., collaboration networks where values of lines represent the number of common publications. Using VOS Mapping we can visualize connected networks having up to approximately 10,000 vertices. See additional reading for more details on Pivot MDS and VOS Mapping.

Exercise III

Extract the dining-table partners relation from the dormitory network (see Section 1.3.2) and manipulate the *Factor* option of the Fruchterman–Reingold energy command to obtain a sociogram of the dining-table partners network with a clear distinction between vertices in the center and vertices in the margin or periphery. Do you think your drawing is better than the original? Justify your answer.

1.3.4.2 Manual Drawing

Pajek supports manual drawing of a network. Use the mouse to drag individual vertices from one position to another. Place the cursor on a vertex, hold down the left mouse button, and move the mouse to drag a vertex.



Figure 11. A selected option in the Draw screen.

Notice that the lines that are incident with the vertex also move. You can drop the vertex anywhere you want, unless you constrain the movement of vertices with the options in the *Move* menu.

The *Move* menu in the Draw screen has (at least) three options: *Fix*, *Grid*, and *Circles*. The *Fix* option allows you to restrict the movement of a vertex. It cannot be moved either horizontally or vertically if you select *x* or *y*, respectively, in the drop-down menu. Select *Radius* to restrict the movement of a vertex to a circle. The options *Grid* and *Circles* allow you to specify a limited number of positions to which a vertex can be moved. In the case of small networks, this generates aesthetically pleasing results. In Pajek, when you select an option, it is marked by a dot (Figure 11).

You can zoom in on the drawing by pressing the right mouse button and dragging a rectangle over the area that you want to enlarge. Within the enlargement, you can zoom in again. Select the command *ZoomOut* if you want to zoom out to the previous zoom level (note that this option is visible only after zooming in). To return to the entire network, you must select the *Redraw* command on the Draw screen menu. The *GraphOnly* menu is similar to the *Redraw* menu, except that *GraphOnly* removes all labels as well as the heads of the arcs (press *Ctrl-l* to redisplay them). It shows vertices and lines only. This option speeds up automatic layout of a network, which is particularly helpful for drawing large networks. The *Previous* and *Next* commands on the menu allow you to display the network before or after the current network in the *Network* drop-down menu, so you need not return to the Main screen first to select another network. Note, however, that this is true only if the option *Network* is selected in the *Previous/Next*> *Apply to* submenu of the *Options* menu.

The *Options* menu offers a variety of choices for changing the appearance of a network in the Draw screen and for setting options to commands in other Draw screen menus (Figure 12). Many drawing options are self-explanatory. Options for changing the shape of the network are listed in the *Transform* and *Layout* submenus, whereas options for the size, color, and labeling of vertices and lines are found in several other submenus. Figure 12 shows the *Options*> *Mark Vertices Using* submenu and its options for changing the type of vertex label displayed. Note the keyboard shortcuts in this submenu: Pressing the *Ctrl* key along with the

Move> *Fix*,
Grid, *Circles*

ZoomOut,
Redraw,
Previous, *Next*,
GraphOnly
[*Draw*]
Options>
Previous/Next>
Apply to

[*Draw*]
Options>
Transform,
Layout, *Mark*
Vertices Using

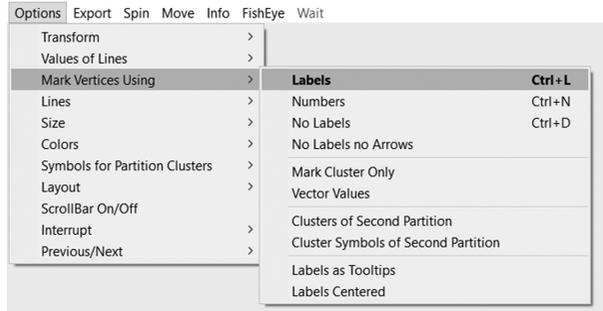


Figure 12. *Options* menu of the Draw screen.

l, *n*, or *d* key has the same effect as selecting the accompanying item in the submenu. If you check *Labels as Tooltips*, labels of vertices appear as popups when you hover over the vertex. This is particularly useful when networks are dense and labels overlap. The last option (*Labels Centered*) positions all labels at the center of vertices instead of to the right of vertices.

[Draw] Info>
Closest Vertices

To improve your drawing, Pajek can evaluate a series of aesthetic properties, such as closest vertices that are not linked or the number of crossing lines. This allows you to find the worst aspects of a drawing and to improve them manually. The *Info* menu in the Draw screen allows you to select one or all aesthetic properties. If you select one of these commands (e.g., *Closest Vertices*), Pajek identifies the vertices that perform worst on this feature: They are identified by a color in the drawing, and they are listed in the Report screen; (e.g., vertices 10 [Maxine] and 14 [Frances] are closest.) In the Draw screen, they now have a divergent color. Because they are not directly linked, it may be wise to move them apart a little.

[Draw] Info>
All Properties

When you select all aesthetic properties, each property will be associated with a color (see Figure 13 for aesthetic information on the dining-table partners network as supplied on the book's website). Sometimes, vertices violate several aesthetic indices, so they ought to be marked by several different colors. Because this is not possible, some colors may not

```

-----
Layout Info
-----
Yellow: The closest vertices: 10 and 14. Distance: 0.07267
LimeGreen: The smallest angle: 2.1.2. Angle: 0.00000
Red: The shortest line: 9.14. Length: 0.08530
Blue: The longest line: 11.20. Length: 0.29546
Pink: Number of crossings: 13
White: Closest vertex to line: 6 to 15.25. Distance: 0.03463

```

Figure 13. Textual output from [Draw] Info> All Properties.

show up in the drawing. There are no rules of thumb for optimal scores on the aesthetic properties. The *Info* command helps to identify the vertices that perform worst, but it is up to you to see whether you can improve their placement.

The last option in the *Info* menu – *Correlation (Layout, Geodesics)** – rates the layout using a single number. It computes the correlation between positions of vertices on the screen and their graph theoretical distances. Correlations range between -1 and $+1$, and the higher the value, the better the visualization. You will learn more about graph theoretical distances (geodesics) in Chapter 6. At this point an example suffices: two vertices that are directly connected by an edge or an arc (e.g., Ada and Louise in Figure 9) should be drawn approximately two times closer than two vertices that are only indirectly connected through one intermediary vertex (e.g., Ada and Lena in Figure 9). *Kamada–Kawai* optimization aims to maximize this correlation. For the visualization presented in Figure 9 this correlation is 0.84, which is a pretty good score. The asterisk in the command is a warning that the command should not be applied to very large networks (50,000 vertices is the upper limit in this case).

[Draw] Info>
Correlation
(Layout,
Geodesics)*

Exercise IV

Open the original dormitory network data file, extract the dining-table partners network (see Section 1.3.2), and improve it according to the aesthetic criterion for minimizing crossing lines. Which vertex can you move to reduce the number of crossing lines?

1.3.4.3 Saving a Drawing

To present pictures of networks to an audience, we have to save our visualizations. This subsection sketches the ways in which network drawings can be exported from Pajek.

If a researcher wants to save a layout for future use, the easiest thing to do is to save the network itself. Remember that a network consists of lists of vertices, arcs, and edges. The list of vertices specifies a serial number, a label, and coordinates in the plane for each vertex. Relocation of a vertex in the Draw screen changes its coordinates, and the latest coordinates are written to the network file on execution of the *Save* command. When the network is reopened in Pajek, the researcher obtains the layout he or she saved, that is, the general pattern because the size and colors of vertices and lines are usually not specified in the network data file.

[Main] File>
Network> Save

Pajek offers several ways to save the drawing as a picture for presentation to an audience. They are listed in the *Export* menu of the Draw screen. Four commands produce two-dimensional output (EPS/PS, SVG, JPEG, and Bitmap) and another four commands yield three-dimensional output (X3D, VRML, MDL MOLfile, and Kinemages). Although three-dimensional representations can be quite spectacular (Figure 14: a

[Draw] Export

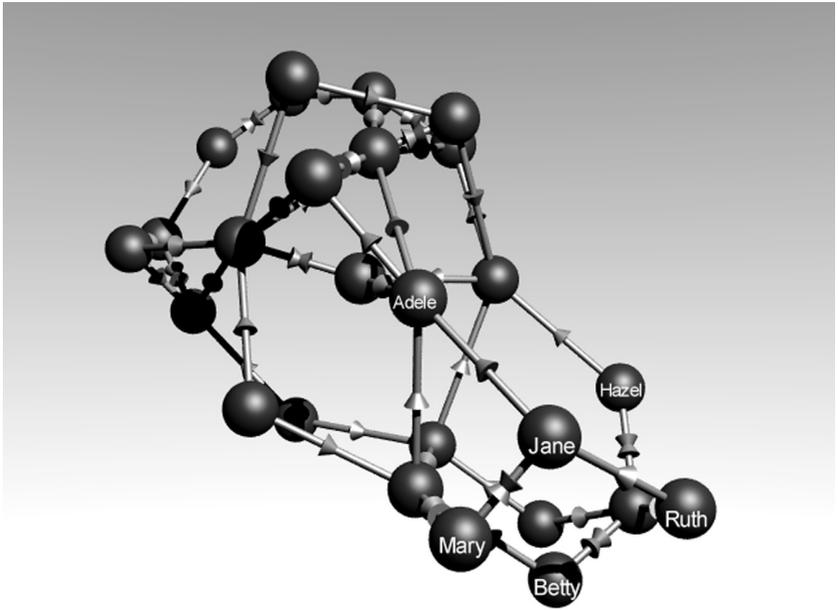


Figure 14. A 3D rendering of the dining-table partners network.

ray-traced image from a VRML model exported by Pajek), we do not discuss them in this chapter but rather in Chapter 5. Here, we briefly outline the commands for two-dimensional output (see Appendix 2 for more details).

```
Export> 2D>
  Bitmap
Export> 2D>
  JPEG
```

The *Bitmap* export command produces an image of the Draw screen: Each screen pixel is represented by a point in a raster, which is called a bitmap. You get exactly what you see; even the size of the picture matches the size of the Draw screen (e.g., see Figure 9). Every word processor and presentation program operating under Windows can load and display bitmaps. Bitmaps, however, are cumbersome to edit, they are “bumpy,” and they lose their sharpness if they are enlarged or reduced.

JPEG is similar to Bitmap, except that the exported files are smaller in size because of data compression.

```
Export> 2D>
  EPS/PS, SVG,
  Options
```

Vector graphics produce more pleasing results than bitmaps. In a vector graphic, the shape and position of each circle representing a vertex and each line, arc, loop, or label are specified. Because each element in the drawing is defined separately, a picture can be enlarged or reduced without loss of quality and its layout can be manipulated easily. Pajek can export vector graphics in two formats, which are closely related: PostScript (command *EPS/PS*) and Scalable Vector Graphics (command *SVG*). PostScript (PS) and Encapsulated PostScript (EPS) are meant for

printing, whereas the Scalable Vector Graphics format was developed for the web. The user can modify the layout of both PostScript and Scalable Vector Graphics drawings in the *Export > Options* submenu, which is covered in detail in Appendix 2.

In this book, we use vector graphics because of their high quality. For example, the sociogram shown in Figure 2 was exported from Pajek as Encapsulated PostScript. For details on using, PostScript and Scalable Vector Graphics, see Appendix 2.

1.4 Assembling a Social Network

To perform network analysis, social relations must be measured and coded. In this section, we briefly discuss data collection techniques for social network analysis and explain how to convert data to a network file for Pajek.

There are several ways to collect data on social relations. Traditionally, sociometrists focus on the structure of social choice within a group. They gather data by asking each member of a group to indicate his or her favorites (or opponents) with respect to an activity that is important to the group. For example, they ask pupils in a class to name the children next to whom they prefer to sit. In a questionnaire, respondents may write down the names of the children they choose or check their names on a list. These methods are called *free recall* and *roster*, respectively. The latter method reduces the risk that respondents may overlook people.

Sometimes, the respondent is asked to nominate a fixed number of favorites. In sociometry, it was very popular to restrict the number of choices to three. For example, in the dining-table partners network each girl was asked to make three choices. This restriction is motivated by the empirical discovery that the more choices that are allowed the more they concentrate on people who are already highly chosen. When asked for their best friends, most people mention four or fewer people. If they have to mention more people, they usually nominate people they think they should like because they are liked by many others. However, restricting the number of choices reduces the reliability of the data: Choices are less stable over time and correlate less well than other measurement techniques such as unrestricted choices, ranking (rank all other group members with respect to their attractiveness), or paired comparison (list all possible pairs of group members and choose a preferred person in each pair). A researcher who fixes the number of choices available to respondents eliminates the difference between a respondent who entertains many friendships and a loner.

Fixed and free choice, ranking, and paired comparison are techniques that elicit data on social relations through questioning. However, there

are several data collection techniques that register social relations rather than elicit them. For example, the amount of interaction between pupils in a class may be observed by a researcher; respondents may be asked to register their contacts in a diary; membership lists and files that log contacts in electronic networks may be coded; family relations or transactions can be retrieved from archives and databases. The rapid growth of electronic data storage offers new opportunities to gather data on large social networks.

Indirect data are usually better than reported data, which rely on the often inaccurate recollections of respondents. However, it is not always easy to identify people and organizations unambiguously in data collected indirectly: Is a Mr. Jones on the board of one organization the same Mr. Jones who is CEO of another firm? It goes without saying that network analysis demands correct identification of vertices in the network.

Application

When you have collected your data, it is time to create a network that can be analyzed with Pajek. In Section 1.3.1, we discussed the structure of a Pajek network file. This is a simple text file that can be typed out in any word processor that exports plain text. Do not forget to attribute serial numbers to the vertices ranging from 1 to the number of vertices. Save the network from the word processor as plain, unformatted text (DOS text, ASCII) and use the extension `.net` in the file name. If your data are stored as a relational database, the database software may be able to produce the Pajek network file; see Appendix 1 for an example.

Network>
Create New
Network>
Empty
Network

It is also possible to produce the network file in Pajek. First, make a new empty network with the command *Network> Create New Network> Empty Network*. In the dialog box, type in the number of vertices you want. The command creates a new network and adds it to the *Network* drop-down menu in the Main screen. When you draw it (*Draw> Network*), you will see that the vertices are nicely arranged in a circle or ellipse (Figure 15).

File>
Network>
View/Edit
Editing
Network screen

As a second step, add lines to the network, which may be done in the Main screen or in the Draw screen. In the Main screen, both the *View/Edit* button at the side of the *Network* drop-down menu (a picture of a magnifier) and the command *File> Network> View/Edit* open a dialog box that allows you to select a vertex by serial number or by label. Next, the Editing Network screen is shown for the selected vertex (Figure 16). In the Draw screen, you can open the Editing Network screen by right-clicking a vertex.

[Editing
Network]
Newline

Double-clicking the word *Newline* in the Editing Network screen opens a dialog box allowing the user to add a line to or from the selected vertex. To add an edge, type the number of another vertex. Type the number of

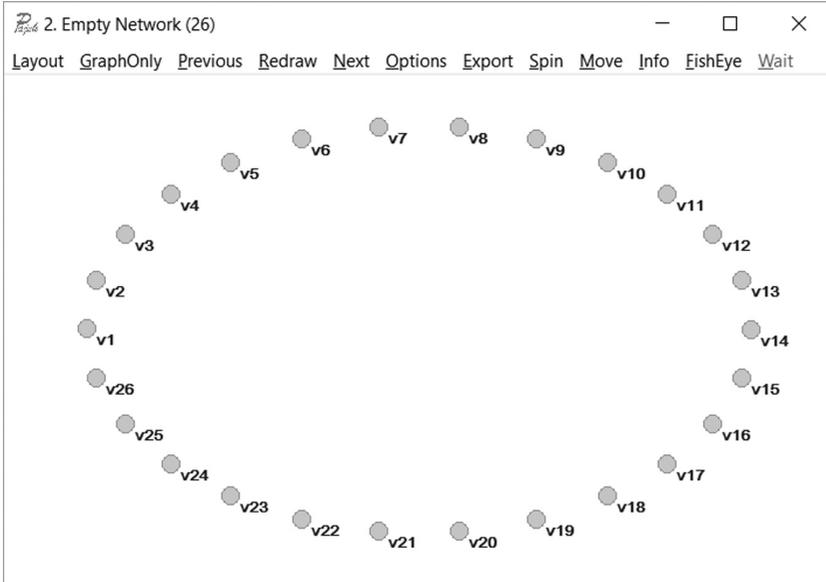


Figure 15. Empty network.

the vertex preceded by a + sign to add an arc *to* the selected vertex and use a – sign to add an arc *from* the selected vertex. Each new edge and arc is displayed as a line in the Editing Network screen. For example, Figure 16 displays an arc from vertex 4 to vertex 1, an arc from vertex 1 to 3, and an edge (indicated by a dash between vertex numbers) between vertices 1 and 2. Also, you can delete a line in the Editing Network screen: Just double-click the line you want to delete.

By default, all lines have unit line value, as indicated by the expression $val = 1.0000$ in the Editing Network screen and relation number 1, as indicated by the first number for each line. Line values and relation numbers can be changed in this screen by selecting the line (left-click) and then right-clicking it. A dialog box appears that accepts any real number (positive, zero, or negative) as input for line values and positive integers (1–999999997) for relation numbers.

Relation	Line	Value	Direction
1	4.1	val=1.00000	/ v4.v1
1	1.3	val=1.00000	/ v1.v3
1	1-2	val=1.00000	/ v1-v2

Figure 16. Edit Network screen.

File>
Network> Save

As a final step, save the network. Networks in Pajek are not automatically saved. Because network analysis usually yields many new networks, most of which are just intermediate steps, Pajek does not prompt the user to save networks. Save the new network as soon as you finish editing it. Use *Save* from the submenu *File> Network* or use the save button (the picture of a diskette) at the side of the *Network* drop-down menu. We recommend that you save a network in Pajek Arcs/Edges format for easy manual editing and for a maximum choice of layout options (see Appendix 2). Give the file a meaningful name and the extension *.net*.

Exercise V

Create some random networks with as many vertices and arcs as the dining-table partners network (without multiple lines). Energize them and inspect the number of crossing lines. Which systematic differences occur between the random networks and the original dining-table partners network?

1.5 Summary

This chapter introduced social network analysis and emphasized its theoretical interest in social relations between people or organizations and its roots in mathematical graph theory. A network is defined as a set of vertices and a set of lines between vertices with additional information on the vertices or lines. This flexible definition permits a wide variety of empirical phenomena, ranging from the structure of molecules to the structure of the universe, to be modeled as networks. In social network analysis, we concentrate on relations between people or social entities that represent groups of people (e.g., affective relations between people, trade relations between organizations, or power relations between nations).

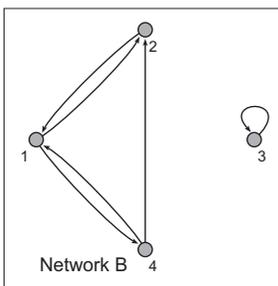
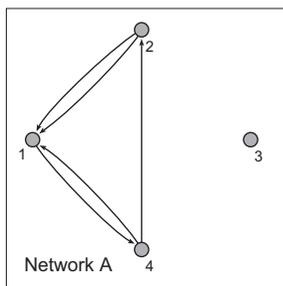
This simple definition of a network covers all types of networks encountered in this book: directed and undirected networks and networks with and without loops, multiple lines, or multiple relations. Most social networks are simple undirected or directed networks, which contain no multiple lines, and loops usually do not occur. As a result of transformations, however, networks may acquire multiple lines and loops. The results of analytic procedures may depend on the kind of network we are analyzing, so it is important to know what kind of network it is.

The mathematical roots of network analysis permit powerful and well-defined manipulations, calculations, and visualizations of social networks. Exploratory social network analysis as we present it here makes ample use of these three techniques. In exploring a social network, we first visualize it to get an impression of its structure. The sociogram, which originates from sociometry, is our main visualization technique.

We are convinced that doing social network analysis is a good way to learn about it. Therefore, the program Pajek for network analysis is an integral part of the book. We urge you to practice the commands demonstrated rather than just read about them. The data related to each example in the book are available on the book's website (<http://mrvar.fdv.uni-lj.si/pajek/>). This chapter introduced the menu structure of Pajek and some basic commands for visualizing networks. You are now ready to start analyzing social networks.

1.6 Questions

1. A sociogram is
 - a. an index of sociability.
 - a graphic representation of group structure.
 - the structure of sociometric choices.
 - a set of vertices connected by edges or arcs.
2. Which of the following definitions is correct?
 - A graph is a set of vertices and a set of lines.
 - A graph is a set of vertices and a set of edges.
 - A graph is a network.
 - A graph is a set of vertices and a set of pairs of vertices.
3. Have a look at the following networks and choose the correct statement.



- Neither A nor B is a simple directed graph.
 - A is not a simple directed graph but B is.
 - A and B are simple directed graphs.
 - A and B are networks but not graphs.
4. Social network analysis is exploratory if
 - the researcher has no specific ideas on the structure of the network beforehand.
 - it deals with social settings that are unexplored by social scientists.
 - the network is studied outside a laboratory.
 - it does not try to predict network structure from a sample.

5. Which of the following statements is correct?
- A line can be incident with a line.
 - A line can be incident with a vertex.
 - An edge can be incident with a line.
 - A vertex can be incident with a vertex.

1.7 Assignment

Your first social network analysis: Investigate the friendship network within your class as it is or as you perceive it. Choose the right kind of network to conceptualize friendship ties (directed, undirected, valued, with multiple lines and loops?), collect the data (design a questionnaire or state the friendships you observe), make a Pajek network file, and then draw and interpret it. If you use perceived friendship ties, it is worthwhile to compare your network to networks made by other students. Where do they differ?

1.8 Further Reading

- The dining-table partner choices of the girls' school dormitory is taken from J. L. Moreno, *The Sociometry Reader* (Glencoe, IL: The Free Press, 1960, p. 35). The cooperation among the girls is fictitious and for demonstration purposes only.
- A brief history of social network analysis can be found in J. Scott, *Social Network Analysis: A Handbook* (London, SAGE [4th edn. 2017], 1991, Chapter 2). L. C. Freeman's *The Development of Social Network Analysis* (Vancouver: ΣP Empirical Press, 2004) gives a comprehensive historical account and his article "Visualizing Social Networks" (*Journal of Social Structure* 1:2 [2000]) describes the history of network visualization.
- Stanley Milgram originated research on the Small World problem; for example, see S. Milgram, "Interdisciplinary thinking and the small world problem," in M. Sherif and C. W. Sherif (eds.), *Interdisciplinary Relationships in the Social Sciences* (Chicago, IL: Aldine, 1969, pp. 103–20).
- For the problem of boundary specification and sampling, see S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications* (Cambridge: Cambridge University Press, 1994, Section 2.2, pp. 30–5. Information on data collection can be found in Section 2.4, pp. 43–59). This impressive monograph is a good starting point for further reading on any topic in social network analysis.

- The references for the energy commands are T. Kamada and S. Kawai, “An algorithm for drawing general undirected graphs” (*Information Processing Letters* 31 [1989], 7–15) and T. M. J. Fruchterman and E. M. Reingold, “Graph drawing by force-directed placement” (*Software, Practice and Experience* 21 [1991], 1129–64).
- Pivot MDS and VOS mapping are explained in U. Brandes and Ch. Pich, “Eigensolver methods for progressive multidimensional scaling of large data” (*Graph Drawing* [2006], 42–53) and N. J. Van Eck, L. Waltman, R. Dekker, and J. Van den Berg, “A comparison of two techniques for bibliometric mapping: Multidimensional scaling and VOS” (*Journal of the American Society for Information Science and Technology* 61[2010], 2405–16).
- All commands in Pajek with detailed explanation are available in “Pajek, PajekXXL and Pajek3XL: Programs for analysis and visualization of very large networks. Reference manual. List of commands with short explanation. Available at: <http://mrvar.fdv.uni-lj.si/pajek/pajekman.pdf>.

1.9 Answers

Answers to the Exercises

- I. This command is part of the *Network* menu because it operates on a single network only. Each arc creates an edge in the new network, so pairs of girls linked by two or more arcs will be linked by multiple lines. The original network is simple, so there are no multiple arcs (or multiple loops). However, it contains bidirectional arcs – mutual choices – which yield two edges in the new network. This happens in ten pairs: Ada–Cora, Louise–Lena, Marion–Frances. Marion–Martha, Eva–Maxine, Jean–Helen, Hazel–Hilda, Hilda–Betty, Edna–Mary, and Mary–Jane.
- II. In Figure 7, the line starting with “Number of lines with value = 1” shows the number of lines with line value 1: 26 arcs. Because the line value of 1 represents a first choice here, the number of first choices is 26. This should not come as a surprise because each of the 26 girls was supposed to indicate one first choice among her peers. To get a list of all first choices in the dining-table partners network, we should have entered –26 or the range 27 to 52 (type the two numbers separated by a space), which contains the arcs on ranks 27 up to and including 52, in the dialog box.
- III. Setting *Factor* to 2 or higher will yield a drawing with a lot of vertices placed on a circle and few vertices in the middle, notably Anna, Eva, or Edna. This drawing is not an improvement; the selection of central

vertices is arbitrary. Repeat the energy command several times (with random starting positions), and you will find different persons in the center.

- IV. Select the command *No. of Crossings* in the *Info* menu of the Draw screen. The vertices in the center are now colored pink: Their lines cross. The Report screen shows the number of crossing lines to be 13. The easiest improvement is to drag Alice to a position above the arc from Laura to Eva. Now Alice's arcs cross no others and the number of crossing lines is reduced to 12. However, it is even better to drag Alice and Martha to the left of the arc from Francis to Eva, because Martha and Marion are connected by mutual choice, which counts twice at every crossing. If you do this, it is better to drag Maxine below the arc from Martha to Anna, because Eva and Maxine are doubly connected. Another improvement is made if Adele is placed to the left of the arc from Anna to Lena. We reach a minimum of seven crossing arcs. Can you do better?
- V. The random networks seem to be "messier" than the original dining-table partners network. This is clearly reflected by the fact that the number of crossing lines in the energized drawing is much higher: approximately 40 crossing lines compared to 13 in the dining-table partners network. There are several possible reasons. First, in the dining-table partners network, each girl makes two choices, whereas the number of arcs emanating from a vertex in the random network differs: You may find isolates next to vertices sending four or more arcs. The vertices sending many arcs are likely to "clutter" the view. Another reason pertains to the ratio of mutual choices. In the actual selection of dining-table partners, there is a fair chance that the girls will nominate each other. In the sociogram, this is represented by a bidirectional arc, which needs no more space than a single arc. See Chapter 13 for more information on working with random networks.

Answers to the Questions in Section 1.6

1. Answer b is correct because a sociogram is a specific way of drawing the structure of ties within a human group.
2. A graph is defined as a set of vertices and a set of lines between pairs of vertices (see Section 1.3.1), so answer d is correct: Lines can be defined as pairs of vertices. Answer a does not specify that the lines must have the graph's vertices as their endpoints, so this answer is not correct. Answer b has the same flaw, and it restricts the definition to undirected graphs, which is not correct according to our definitions. Answer c ignores the difference between a graph and a network.
3. Answer b is correct. A simple directed graph contains at least one arc (so it is directed) and no multiple lines; it may contain loops.

Therefore, network B is a simple directed graph but network A is not because of its multiple lines. Answer d is wrong because there is no additional information on the vertices and arcs, such as vertex labels or line values.

4. Answers a and d are correct.
5. Answer b is correct.

Attributes and Relations

2.1 Introduction

In Chapter 1, we argued that social network analysis focuses on social relations. A network is a set of vertices and lines. Both vertices and lines have characteristics that we may want to include in our analysis (e.g., the gender of people and the strength of their ties). As noted in Chapter 1, properties of relations are represented by line values in the network (e.g., first and second choices among girls in the dormitory). Now, we add characteristics of the vertices to the analysis. How can we use information on the actors to make sense of the social network?

In this chapter, we present techniques that combine relational network data and nonrelational attributes, such as psychological, social, economic, and geographical characteristics of the vertices in the network. The attributes enhance our interpretation of network structure, and they enable us to study subsections of the network. In addition, we briefly discuss how to use the network position of vertices in statistical analysis; social network analysis and statistics are two complementary sets of techniques. After having studied this chapter, you will understand the basic data used in network analysis and will be able to combine relational and nonrelational data.

2.2 Example: The World System

Social network analysis can be applied to large-scale phenomena. In 1974, Immanuel Wallerstein introduced the concept of a capitalist world system, which came into existence in the sixteenth century. This system is characterized by a world economy that is stratified into a core, a semiperiphery, and a periphery. Countries owe their wealth or poverty to their position

in the world economy. The core, Wallerstein argues, exists because it succeeds in exploiting the periphery and, to a lesser extent, the semiperiphery. The semiperiphery profits from being an intermediary between the core and the periphery.

The world system is based on a global division of labor. Countries in the core specialize in capital-intensive and high-tech production, whereas peripheral countries apply themselves to low-valued, labor-intensive products or unprocessed, raw materials. Core countries import raw and less-processed products from the periphery and turn them into expensive high-tech products that are exported to countries in the core, semiperiphery, and periphery. In consequence, there is much trade among core countries, but little trade between countries in the periphery. The core dominates the world trade in a double sense: Core countries are more often involved in trade ties than peripheral countries, and the value of exports from core countries exceeds the value of imports because their products have higher added value. This is why core countries do very well economically.

Which countries belong to the core, semiperiphery, or periphery? In political economy, several attempts have been made to answer this question, some of which are based on social network analysis. The network analysts analyzed the structure of the trade relation and classified countries according to the pattern of their trade ties; for instance, the nations that trade with almost all other countries are classified as core countries. World trade statistics, which are widely available, offer the data required for this analysis.

In this chapter, we use statistics on world trade in 1994. We included all countries with entries in the print version of the *Commodity Trade Statistics* published by the United Nations, but we had to add data on some countries for 1993 (Austria, Seychelles, Bangladesh, Croatia, and Barbados) or 1995 (South Africa and Ecuador) because they were not available for 1994. Countries that are not sovereign are excluded because additional economic data are not available, for example, the Faeroe Islands and Greenland or Macau, which belong to Denmark and Portugal, respectively. In the end, the network contains eighty countries, and most missing countries are located in central Africa and the Middle East or belong to the former USSR.

The arcs in our network represent imports into one country from another. We restrict ourselves to one class of commodities rather than total imports, and we picked miscellaneous manufactures of metal, which represent high-technology products or heavy manufacture. We use the absolute value of imports (in US\$1,000), but we did not register imports with values less than 1 percent of the country's total imports on this commodity. The network data are stored in the file `Imports_manufactures.net`.

In addition, we use several attributes of the countries in our analysis, namely their continent (`Continent.clu`), their structural world system position in 1994 (`World_system.clu`), their world system position in 1980 according to a previous analysis by Smith and White (`World_system_1980.clu`; see the reference in Section 2.10), and their gross domestic product per capita in US dollars in 1995 (`GDP_1995.vec`). Note that in this chapter, we do not determine the world system position of the countries; we use results from an advanced structural technique called blockmodeling, which is presented in Chapter 12. The three world system positions in 1994 – core, semiperiphery, and periphery – are defined such that the core countries trade a lot of manufactures of metal among themselves, and they export a lot to the countries in the semiperiphery; whereas the countries in the semiperiphery and periphery do not export a substantial amount of these manufactures.

2.3 Partitions

A partition of a network is a classification or clustering of the network's vertices. Each vertex is assigned to exactly one class or cluster (we use these words as synonyms); for example, one country is assigned to the core and another to the semiperiphery. A partition may contain a special class that collects the vertices we cannot classify because data are missing. Usually, the classes are identified by integers; for instance, a core country receives code 1 in the partition, and a country in the semiperiphery gets a 2. In this format, a partition is simply a list of nonnegative integers, one for each vertex in the network.

A *partition* of a network is a classification or clustering of the vertices in the network such that each vertex is assigned to exactly one class or cluster.

In network analysis, partitions store discrete characteristics of vertices. A property is *discrete* if it consists of a limited number of classes; for instance, we may code the continents of countries by digits such that African countries constitute class 1, Asian countries constitute class 2, and so on. Six classes will suffice. A classification contains a limited number of classes, and most classes contain several vertices because we want the classes to represent groups of actors rather than single actors or nothing at all. Partitions, therefore, are very useful for making selections from a network to reduce its size and complexity. We discuss this in Section 2.4.

In some cases, the order of class numbers in a partition is arbitrary; for instance, in the partition of nations according to continents, there is no compelling reason why African countries should have a lower class code than Asian countries. In other instances, however, the order is meaningful. For instance, it would be foolish to assign the semiperiphery to class 1, the core to class 2, and the periphery to class 3, because this would not correctly reflect the obvious ranking of the three classes. Finally, the class codes may represent “real” numbers, for instance, the number of lines incident with a vertex: All vertices in class 1 are incident with one line, vertices in class 2 are incident with two lines, and so on, so make sure that you attach the right meaning to class numbers!

Partitions may specify a *structural property* such as world system position, which is a result of network analysis or a characteristic measured independently of the network (e.g., the continent where a nation is located). We call the latter *attributes* of vertices.

Figure 17 displays the trade in manufactures of metal and their position in the world system in 1994. In line with the spatial connotations of the concepts of core, semiperiphery, and periphery, the core countries are placed in the center (black vertices), the semiperiphery constitutes the middle ring (gray vertices), and the peripheral countries (white vertices) are located on the outer ring. The intense trade ties among the countries in the core and between the core and semiperiphery are apparent, just like the relative absence of trade in manufactures of metal among countries in the periphery. We should, however, note that the impression of clear boundaries between the three classes has been deliberately created by the researcher: This layout shows the world system positions rather than proves them.

Application

In Pajek, partitions are a data object on their own, so they are accessible from a drop list once they are read (*File> Partition> Read*). Partitions are saved in files with the extension `.clu` (*File> Partition> Save*), and these files are just lists of nonnegative integers preceded by a line that specifies the number of vertices. Just open a partition file, for example, `World_system.clu`, in a text editor to see its structure. The first integer represents a property of the first vertex (e.g., world system position), the second integer belongs to the vertex with serial number 2, and so on. You should change neither the sequence of vertices in a network nor the sequence of entries in a partition because this will destroy the compatibility of the partition and the network: Vertices are then no longer associated with the corresponding classes.

Although partitions can be stored separately, we can also save them with the network to which they belong. Pajek has a special data format – the project file – that may contain all networks, partitions, and other

File>
Partition>
Read

File>
Partition> Save

Partition
drop-down
menu

File> Pajek
Project File

Table 1. *Tabular output of the command Partition> Info*

Cluster	Freq	Freq%	CumFreq	CumFreq%	Representative
1	12	15.0000	12	15.0000	Austria
2	51	63.7500	63	78.7500	Algeria
3	17	21.2500	80	100.0000	Bangladesh
SUM	80	100.0000			

first vertex in the world trade network represents Algeria, which belongs to class 2 (semiperiphery), and the fourth vertex is Austria, part of the core (class 1). You may click on the class code to change it manually, but you can also change the labels of vertices in the network. Note that the labels are displayed only if the associated network is selected in the *Network* drop-down menu.

The command *Info* in the *Partition* menu produces a frequency table of the classes in the active partition, which offers a quick way to inspect a partition. On execution, this command displays a dialog box with two user entries. The first entry allows for suppressing classes in the table that occur seldom; for instance, type 5 to exclude classes with four vertices or less from the frequency tabulation. For the second entry, which is similar to the entry in the dialog box associated with the *Network> Info> General* command (see Section 1.3.3), the user may request a listing of vertices with the highest or lowest class numbers. Type a positive integer to list vertices with the highest class numbers, and type a negative integer to list vertices with the lowest class numbers.

Partition> Info

The *Partition> Info* command presents a table that lists the number of vertices in each class of the partition. In Table 1, we can see that twelve countries belong to the first class, which is 15 percent of all countries. The number 12 is the *frequency* (abbreviated to *Freq* in the table) with which class 1 occurs among the vertices. Pajek does not “know” that this class refers to the core. It can only help the interpretation of the meaning of the class by showing a representative of the class, that is, the label of a vertex that belongs to this class. Together, the core and semiperiphery

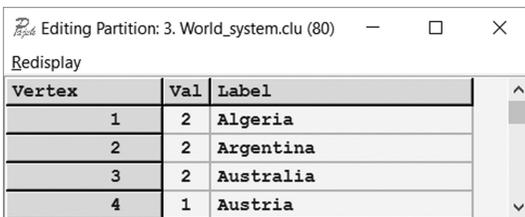


Figure 18. Edit screen with partition according to world system position.

contain sixty-three of the eighty countries (column CumFreq), which is 78.75 percent (CumFreq%).

Draw>
Network +
First Partition

Because partitions contain discrete classes, the class to which a vertex belongs can be represented by the color of the vertex. Each class is associated with a color (e.g., vertices in class 1 are yellow). It is easy to obtain a colored sociogram. First, make sure the right network and partition are selected in the drop-down menus of the Main screen, for example, the manufactures of metal network (`Imports_manufactures.net`) in the *Network* drop-down menu and the world position partition (`World_system.clu`) in the *Partition* drop-down menu. Next, execute the command *Network* + *First Partition* from the *Draw* menu (or press *Ctrl-p*). The same command can also be executed by clicking buttons and checkboxes on the right-hand side of the Pajek Main window: Select the first checkbox at the right of the *Partition* drop-down menu and then click on the drawing pencil button at the right of the *Network* drop-down menu. If the selected network and partition are compatible, that is, if the number of entries in the partition is equal to the number of vertices in the network, Pajek draws the network with vertex color determined by the partition.

Layout>
Energy>
Kamada-
Kawai> *Free*

Move> *Circles*

In Figure 17, the circular layout was created in the following way. First, the layout was energized with the Kamada–Kawai energy command. This brought most core countries to the center and most peripheral countries to the margin of the sociogram. Then, manual movement of vertices was restricted to three circles and eighty positions on each of them with the *Circles* command in the *Move* menu. Finally, the countries were manually dragged toward the nearest position on the appropriate circle.

Then, what you get is a sociogram such as in Figure 17 in color. On your screen, the core countries are yellow, the countries in the semiperiphery are green, and the peripheral countries are red. Because this book is printed in black and white, we cannot reproduce the colors here but we stored all the color illustrations in a document (`illustrations.pdf`) on the website dedicated to this book (<http://mrvar.fdv.uni-lj.si/pajek/>), so you can check the colors that you are expected to see on your screen.

Options>
Colors>
Partition
Colors> for
Vertices

For black-and-white printing, a limited number of grays can replace color. Pajek offers a command to switch between colors and grays in the *Colors* submenu of the *Options* menu in the Draw screen: the command *Partition Colors for Vertices*. On selection of this command, Pajek displays a dialog box such as that in Figure 19. It contains forty colored squares and the partition's class numbers with which they are associated. For class numbers above 39, Pajek cycles through the first forty colors again: The vertex color of class 40 is equal to the color of class 0, and so on. Press the button labeled “Default GreyScale 1” to change the first five colors (classes 0 to 4) into grays (this color scheme is represented in Figure 19) or the button “Default GreyScale 2” to change the first eleven colors

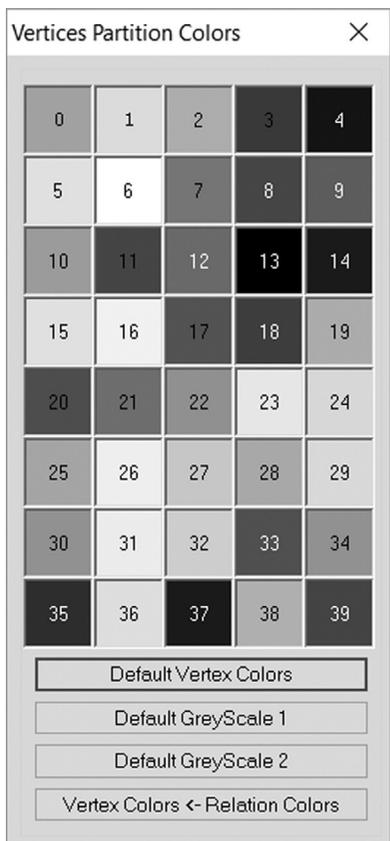


Figure 19. Vertex colors according to a partition in Pajek.

into grays. The button “Default Vertex Colors” resets the original colors. Unless stated differently, we use partition color scheme “GreyScale 1” in the sociograms printed in this book.

In addition, you can change the color of a particular class by means of the *Partition Colors* dialog box. If you want to change the color associated with a particular class, click on the square with the desired color, and type the number of the class you want this color to be associated with in the dialog box that appears; Pajek will swap the colors. Press the button labeled “Default Vertex Colors” if you want to restore the original colors of classes.

When colors or grays do not suffice, you may display the class numbers of the vertices at the center of the vertices. Select the option *Clusters of Second Partition* in the *Options> Mark Vertices Using> Clusters of Second Partition*

*Options>
Mark Vertices
Using>
Clusters of
Second
Partition*

Main Pajek window. Three drop-down menus for partitions are available. If they are not visible, keep clicking the Partitions button on the left which toggles between displaying one, two, or three menus, so that you can select one partition in each. In our case the second partition is the same as the first one, select the partition named “world_system.clu” again. Until you turn this option off, the cluster number of each vertex is displayed at the vertex center, provided, of course, that the second partition matches the network.

Options>
Mark Vertices
Using> Cluster
Symbols of
Second
Partition

It is possible to represent vertex type by symbols instead of cluster numbers. If you select *Cluster Symbols of Second Partition* in the *Options> Mark Vertices Using* submenu, symbols associated with cluster numbers are displayed at the center of the vertices. For symbols you can select any Unicode characters. The default symbols are ■ ● ◆ ★ ✱ ☼ ☼ ♀ ♂ in this order for clusters 0 to 9. You can change the list of symbols with *Options> Symbols for Partition Clusters> Change*. For example, remove all symbols except those for females and males to obtain the ♀ symbol for vertices in class 0 of the second partition and the ♂ symbol for the vertices in class 1. Thus, you can visualize a network of relations between women and men. You can also copy and paste Unicode symbols from elsewhere into this dialog. You can set the size of symbols (or clusters numbers) in *Options> Size> of Symbols*. If you select *Use Third Partition for Symbol Color* in the *Options> Colors* submenu, symbol colors will be determined by the third partition. More information on using symbols is available on the Pajek website. See Figure 149 (Appendix 2) for an example of a network visualization with symbols.

Options>
Symbols for
Partition
Clusters>
Change

Options>
Size> of
Symbols

Options>
Colors> Use
Third Partition
for Symbol
Color

Partition>
Create
Constant
Partition

Draw>
Network +
Create Null
Partition

In Pajek, you can create a new partition that can be edited manually. In the *Partition* menu, the command *Create Constant Partition* makes a new partition for the selected network. You must enter the dimension (size) of a partition and a constant – in our case 0. Pajek suggests a dimension that is equal to the number of vertices of the active network. Usually this is the right number. All vertices are placed in class 0. With the edit command, which was discussed previously (*File> Partition> View/Edit*), you can assign vertices to other classes: Just change their class numbers in the list. You can obtain the same result with the command *Network + Create Null Partition* from the *Draw* menu in the Main screen (shortcut: *Ctrl-a*). This command creates a new partition and displays it in the Draw screen.

Change the
class number of
vertices

In the Draw screen, you can raise the class number of a vertex by 1 in the following way: Click on the vertex with the middle mouse button – if available – or with the left mouse button while holding down the *Shift* key on the keyboard. If the cursor is not on a vertex, all class numbers are raised. Clicking a vertex while the *Alt* key is pressed subtracts 1 from the class number, and clicking between vertices with the *Alt* key pressed lowers the class numbers of all vertices provided that they are larger than 0,

which is the minimum value. In this way, you can easily create a partition that groups a small number of vertices.

In a sociogram with colored classes, it is very easy to move all vertices that belong to one class. Position the cursor *near* but not *on* a vertex, press the left mouse button, and drag: All vertices in the class will move simultaneously. This is a very useful technique. In addition, the Kamada–Kawai energy procedure has a special command for energizing networks with class colors; you can restrict the automatic relocation to the vertices in class 0 with the *Fix Selected Vertices* command. If the partition does not contain vertices in class 0, Pajek issues a warning and it does not change the layout of the network.

Layout>
Energy>
Kamada–
Kawai> *Fix*
Selected
Vertices

If you want to compare lines within classes and between classes visually, it is useful to have classes represented by nonoverlapping circles. Lines within circles represent ties within classes, for example, trade within a continent, and lines between circles represent interclass ties, for example, trade among continents. This can be done in Pajek. Select the `Imports_manufactures.net` network and the `Continent.clu` partition. Next, draw the network with the partition (see earlier) and issue the *Circular>using Partition* command in the *Layout* menu of the Draw screen. As described in the preceding paragraph, it is easy to move circles with your mouse. If you would like to enlarge the circles, move them toward the center and resize the drawing with the *Options>Transform>Fit Area>max(x),max(y),max(z)* command. Your drawing is probably still too dense to really see well which continents have high or low internal and external trade relations. The next section offers techniques for better results.

Layout>
Circular>
using Partition

Options>
Transform> *Fit*
Area> *max(x),*
max(y), max(z)

There are several ways to manipulate partitions, make a new partition, or combine two partitions. We encounter most of these techniques in later chapters. Suffice it to say here that the commands that involve one partition are located in the *Partition* menu, whereas commands operating on two partitions can be found in the *Partitions* menu of the Main screen.

Partition

Partitions

Exercise I

Open the original manufactures of metal trade network and energize the positions of the core countries only. What changes? Hint: Create a new partition in which the core countries belong to class 0 and the other countries to class 1 or higher, and energize it with the *Fix Selected Vertices* command.

2.4 Reduction of a Network

Partitions divide the vertices of a network into a number of mutually exclusive subsets. In other words, a partition splits a network into parts.

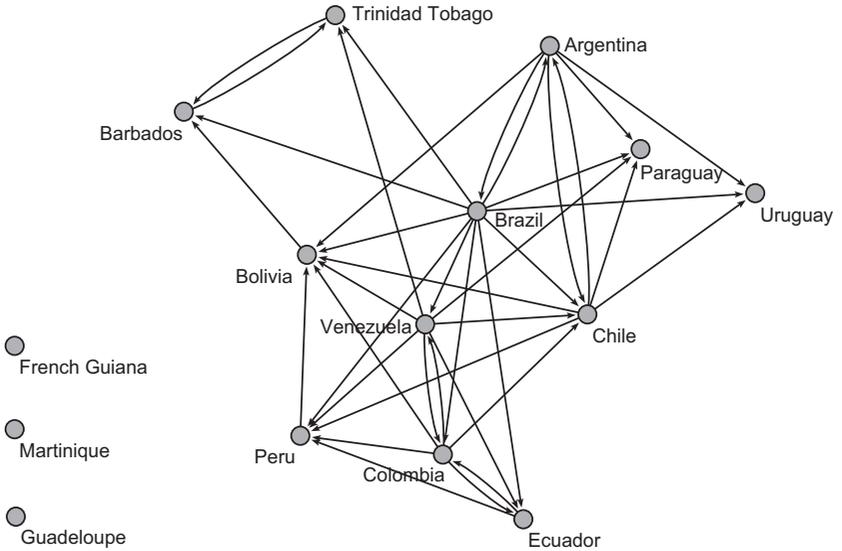


Figure 20. Trade ties within South America.

Therefore, we can use partitions to reduce a network in three ways: extract one part (local view), shrink each class of vertices into one new vertex (global view), or select one part and shrink neighboring classes to focus on the internal structure and overall position of this class (contextual view). We now discuss the three types of reduction.

2.4.1 Local View

The easiest way to reduce a network is to select one class of vertices. Of course, a mere selection of vertices is not very interesting; we must also select all lines between them – and their loops eventually – if we want to inspect the structure of ties in this part of the network. This operation is called *subnetwork extraction*, and its result is called an *induced subnetwork* that offers a local view.

To *extract* a subnetwork from a network, select a subset of its vertices and all lines that are incident only with the selected vertices.

If we want to examine the trade in manufactures of metal within South America, we can extract the subnetwork of South American countries, which is depicted in Figure 20. From the energized drawing, it is clear that Brazil occupies a central position, whereas French Guiana, Martinique, and Guadeloupe are isolated. Apparently, these islands trade with countries in other parts of the world. Finally, geography appears to be

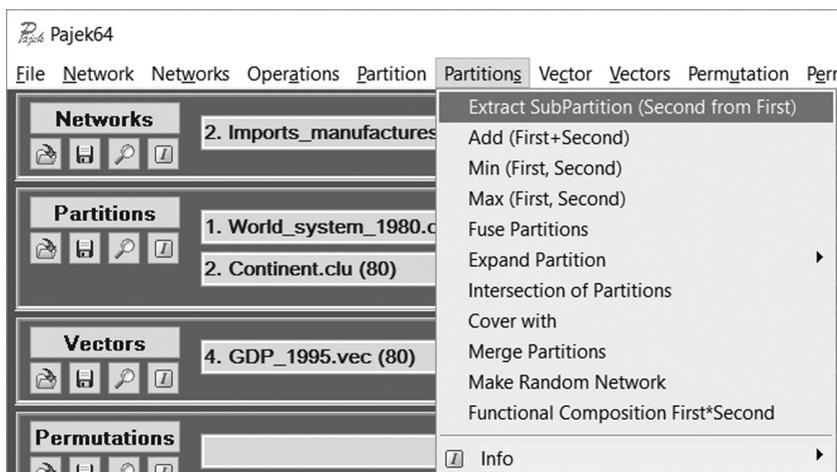


Figure 21. The *Partitions* menu.

important within the continent, because neighboring countries are close in the network.

Application

To extract a subnetwork, we need a network and a partition that defines the sets (classes) of vertices that we want to extract. Therefore, the extraction commands are located in the *Operations* menu. In this section, we use the manufactures of metal trade network (`Imports_manufactures.net`) and the continents partition (`Continent.clu`).

To obtain a local view of the South American trade network, use the command *SubNetwork Induced by Union of Selected Clusters* from the *Operations* > *Network + Partition* > *Extract* submenu. Because South America is the sixth class in the partition, select class 6 in the dialog box that appears. Pajek produces a new network, which contains fifteen countries. Energize it to obtain a drawing similar to Figure 20.

It is very important to note that the original partitions do not fit the induced subnetwork because the subnetwork contains fewer vertices than the original vertices. This is a pity, because we may want to know, for instance, the world system position of the countries in the South American trade network. However, we can extract a new partition that matches the induced subnetwork in a few steps.

First, we must select the original partition that we want to translate to the induced subnetwork as the first partition: Make sure that the world system partition is selected in the first *Partition* drop-down menu (Figure 21). Second, select the partition that you used to extract the network – in our case the continents partition – in the second *Partition* drop-down menu. You need this partition to identify the subset of vertices that must

Operations>
Network +
Partition>
Extract>
SubNetwork
Induced by
Union of
Selected
Clusters

Partitions>
Extract
SubPartition
(Second from
First)

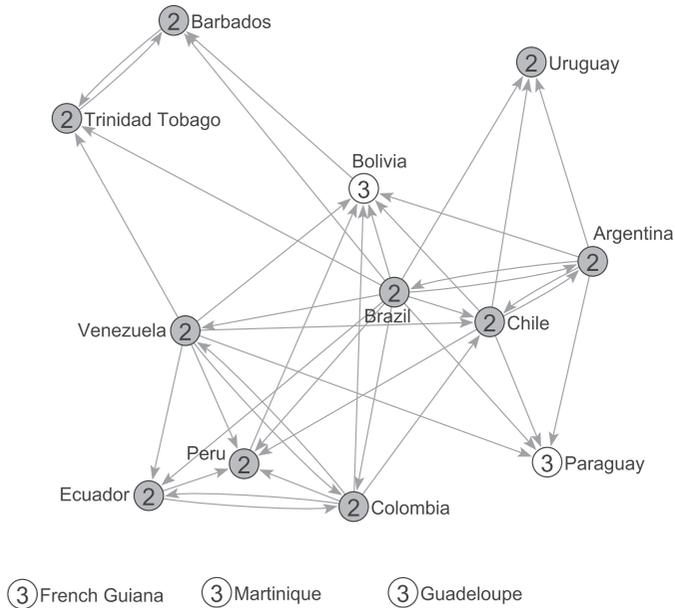


Figure 22. World system positions in South America: (2) semiperiphery and (3) periphery.

be extracted from the world positions partition. Finally, execute the command *Extract SubPartition (Second from First)* from the *Partitions* menu. A dialog box will ask for the class numbers to be extracted, and if you enter 6, Pajek produces a new partition for the fifteen South American countries. Figure 22 shows the South American trade network drawn with this partition.

2.4.2 Global View

Instead of zooming in on a particular region of the network, we may also zoom out to obtain a global view. Now, we are no longer interested in each individual vertex, but we want to study relations between classes, for instance, continents. Which continents have strong trade ties? In this example, a global view of the network covers the whole world, but we should note that a global view may also pertain to ties between groups in a local setting.

To *shrink* a network, replace a subset of its vertices with one new vertex that is incident to all lines that were incident with the vertices of the subset in the original network.

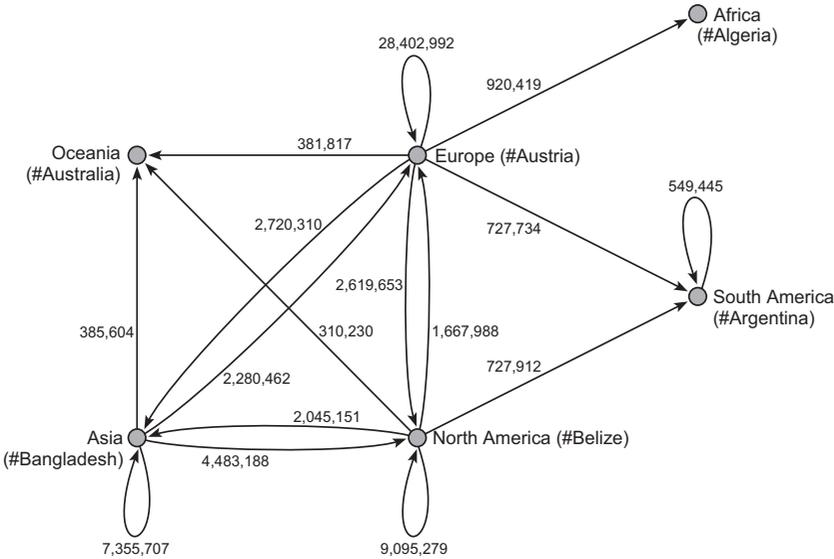


Figure 23. Trade in manufactures of metal among continents (imports in thousands of US\$).

In network analysis, we obtain a *global view* by shrinking all vertices of a class to one new vertex. In our example, we shrink all countries within a continent to a new vertex that represents the entire continent (see Figure 23). Lines incident with shrunken vertices are replaced; for instance, all imports by South American countries from European countries are replaced by one new arc pointing from Europe to South America. Its line value is equal to the sum of all original line values. In the network of trade ties, line values indicate the value of imports expressed in thousands of US dollars (US\$), so values of lines in the shrunken network represent the value of total imports of metal manufactures. Lines between shrunken vertices within one class (e.g., trade in manufactures of metal within South America) are replaced by a loop.

Figure 23 shows the world trade network that is shrunk according to continents. To obtain a clear picture, we deleted arcs (total imports) with summed values less than US\$300 million. We can see that South American countries import manufactures of metal with a total value of US\$728 million from Europe; but they do not export a substantial amount to Europe, so there is an asymmetrical trade tie between South America and Europe. The countries in Oceania and Africa are also importing rather than exporting. Trade within South America (US\$549 million) is much higher than in Oceania and Africa, but internal trade is at its highest in Europe.

Application

Operations>
Network +
Partition>
Shrink
Network

The command *Shrink Network* in the *Operations> Network + Partition* submenu shrinks classes of vertices in a network according to the active partition, in this case, the continents partition. First we are asked for the minimum number of connections between clusters. This is the minimum number of lines that must exist between shrunken vertices to obtain a new line in the shrunken network. We recommend choosing 1 (the default value). In the second entry, you may choose a class of vertices that must not be shrunk to give them a contextual view. Shrink all classes to obtain a global view, so type any class number that is not present in your partition or accept the default value (0). Finally, Pajek shrinks all classes except the selected class and adds the shrunken network to the *Network* drop-down menu.

File>
Partition>
View/Edit

Pajek's *Shrink Network* command also creates a new partition – called shrinking – along with the new network identifying the classes in the partition that was used to shrink the original network. However, Pajek does not know the meaning of these classes, so it cannot assign meaningful new labels to shrunken classes. It chooses the label of the first vertex of a class that is shrunk and adds a pound sign (#) to obtain a label for the shrunken class. For example, Argentina happens to be the first South American country in the network, so the vertex that represents this continent carries the label “#Argentina” in the shrunken network. We added the names of continents to Figure 23 manually by editing the shrunken partition (“Shrinking”) with the *File> Partition> View/Edit* command (see Section 2.3).

Network>
Create New
Network>
Transform>
Remove> Lines
with Value>
lower than

In Figure 23, we removed lines with summed values less than US\$300 million to obtain a clear picture. Lines with low values can be removed automatically with the command *Remove> Lines with Value> lower than* in the *Network> Create New Network> Transform* submenu. We entered 300000 as the threshold for this operation, because import values are measured in thousands of US dollars.

In a shrunken network, a class of vertices is replaced by one new vertex; in our example, the Latin American countries are substituted by a new vertex representing this continent. Properties of the original vertices, such as their world system position, are lost: It is impossible to assign the entire continent to a particular world system position. Therefore, it is impossible to use the data from a partition that was not used to shrink the network.

Options>
Blockmodel -
Shrink

Note that the way a network is shrunk depends on the option selected in the *Options> Blockmodel - Shrink* submenu. By default, option 0 is selected, which means that the number of links in the original network is used to decide whether a new line is added to the shrunken network. This option causes Pajek to display the dialog box mentioned at the start of this “Application” Section. We advise not to select another option from

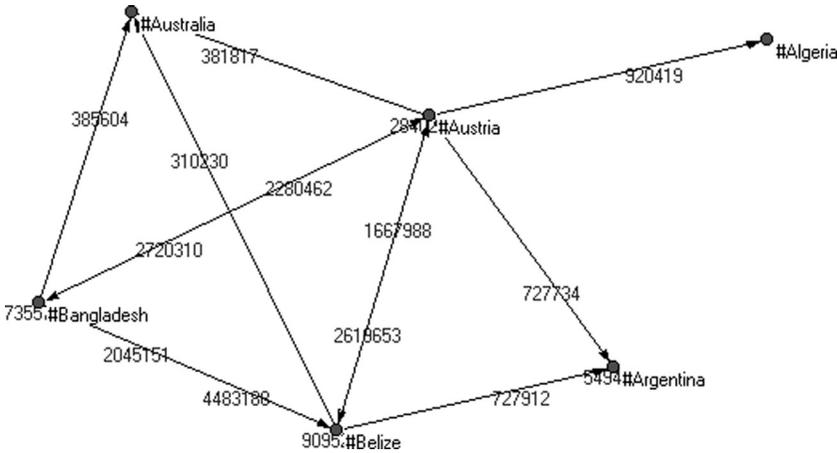


Figure 24. Trade among continents in the Draw screen.

this submenu until you understand blockmodeling, which is presented in Chapter 12.

Loops, which signify the trade within a continent in this example, cannot be drawn in the Draw screen. If the option *Options> Lines> Mark Lines> with Values* (shortcut *Ctrl-v*) has been selected, the line values are shown in the Draw screen (see Figure 24), including the values of loops, which are printed very close to the vertices. The value of a loop can be examined more closely if you right-click a vertex in the Draw screen. Among the lines, which are then listed in the *Editing Network* dialog box, you can find the loop, for instance, the line from #Argentina to #Argentina. Its line value is 549445, which means that the total value of trade in manufactures of metal among South American countries amounted to US\$549,445,000.

*Options>
Lines> Mark
Lines> with
Values*

2.4.3 Contextual View

A global view shows the position of South America in the world system, and a local view clarifies the central position of Brazil within the South American trade network. But how do countries in this regional network relate to the rest of the world? From which continents do the isolated islands import their manufactures of metal? If you want to focus on one class of vertices (e.g., countries on one continent) and take into consideration aggregated ties to the “outside world,” you need a contextual view.

In a *contextual view*, all classes are shrunk except the one in which you are particularly interested. In Figure 25, all countries are shrunk to continents (black vertices) except the South American countries (gray vertices). We removed ties between continents because we already know them from

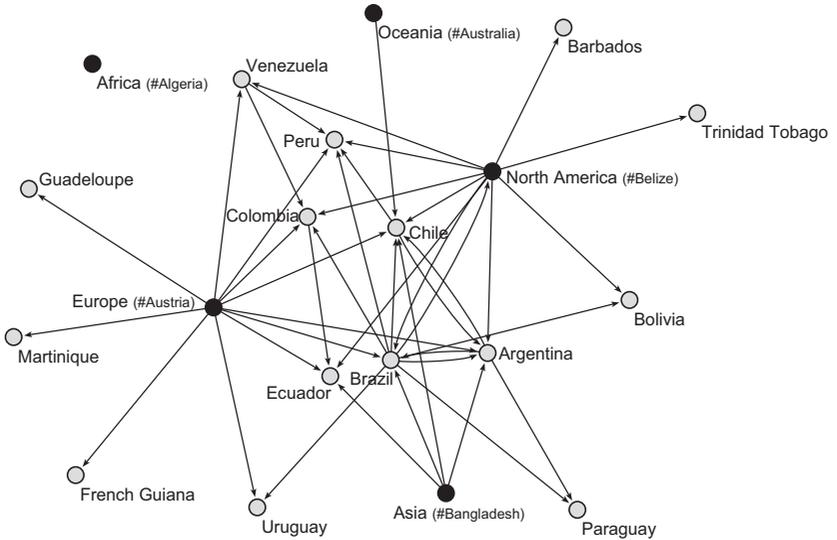


Figure 25. Contextual view of trade in South America.

Figure 23 and we discarded trade in manufactures of metal with summed values less than US\$10 million to obtain an intelligible drawing. This sociogram shows the South American countries in the context of world trade. Clearly, Africa and Oceania are not important trading partners to South American countries. The larger countries on the continent import from Europe and North America, but smaller countries are connected to either North America (Barbados, Trinidad and Tobago, and Bolivia) or Europe (Uruguay, Guadeloupe, Martinique, and French Guiana). Former colonial ties surely play a part here.

There are many ways to extract or shrink a network, and extraction and shrinking can even be combined. Subnetwork extraction and shrinking are important techniques to dissect a network and to obtain partial views of a network when its structure is too complicated to understand at a first glance.

Operations>
 Network +
 Partition>
 Shrink
 Network
 Network>
 Create New
 Network>
 Transform>
 Remove> Lines
 with Value>
 lower than

Application

A contextual view is obtained by partially shrinking the network, so we can use the *Shrink Network* command. In the dialog box asking for the class that should not be shrunk, type the appropriate class number, namely class 6 for South America. Lines with summed values less than US\$ 10 million can be removed automatically with the command *Remove> Lines with Value> lower than* in the *Network> Create New Network> Transform* submenu as discussed.

Lines between continents can be deleted automatically provided that you have a partition that identifies the continents in the shrunken network. You may create this partition using command *Vertex Labels Matching Regular Expression* in the *Network> Create Partition* submenu. Simply enter $\hat{\#}$ for selecting all labels starting with $\#$. Now, you can execute the command *Operations> Network + Partition> Transform> Remove Lines> Inside Clusters* to remove the lines between vertices in the continents class (in our case class 1, enter 1 in the dialog box).

```
Network>
Create
Partition>
Vertex Labels
Matching
Regular
Expression
Operations>
Network +
Partition>
Transform>
Remove Lines>
Inside Clusters
```

Exercise II

Create a global view of the trade relation (manufactures of metal) between the core, semiperiphery, and the periphery. Explain the structural differences between the world system positions from this network.

2.5 Vectors and Coordinates

Partitions store discrete properties of vertices, and vectors store continuous properties. In principle, a continuous property may take any value within a defined range; for instance, the surface of a country may take any value between zero square kilometers (or miles) and the total surface of Earth. When two countries have different sizes, it is always possible to imagine a country that is smaller than one and bigger than the other.

In practice, of course, we do not care about differences in sizes of countries smaller than a square kilometer, but the principle is important: Continuous properties are not meant to group vertices into classes, so they cannot be used to reduce a network by extraction or shrinking. Continuous properties express a particular and often unique value of a vertex, for instance, the wealth of a country indicated by its gross domestic product (GDP) per capita. In practice, no two countries have exactly the same GDP per capita because it is equal to the quotient of the economic production of a country and the size of its population, both of which are usually figures that characterize no other country. Most quotients yield results with decimals, so a vector is not a list of integers but a list of real numbers.

A vector assigns a numerical value to each vertex in a network.

In our discussion of partitions, we have distinguished between structural indices and attributes. This distinction applies to continuous properties or vectors as well. In later chapters, we encounter several examples of vectors that represent structural features of vertices, for instance, their centrality (Chapter 6). GDP per capita of a country is an example of a

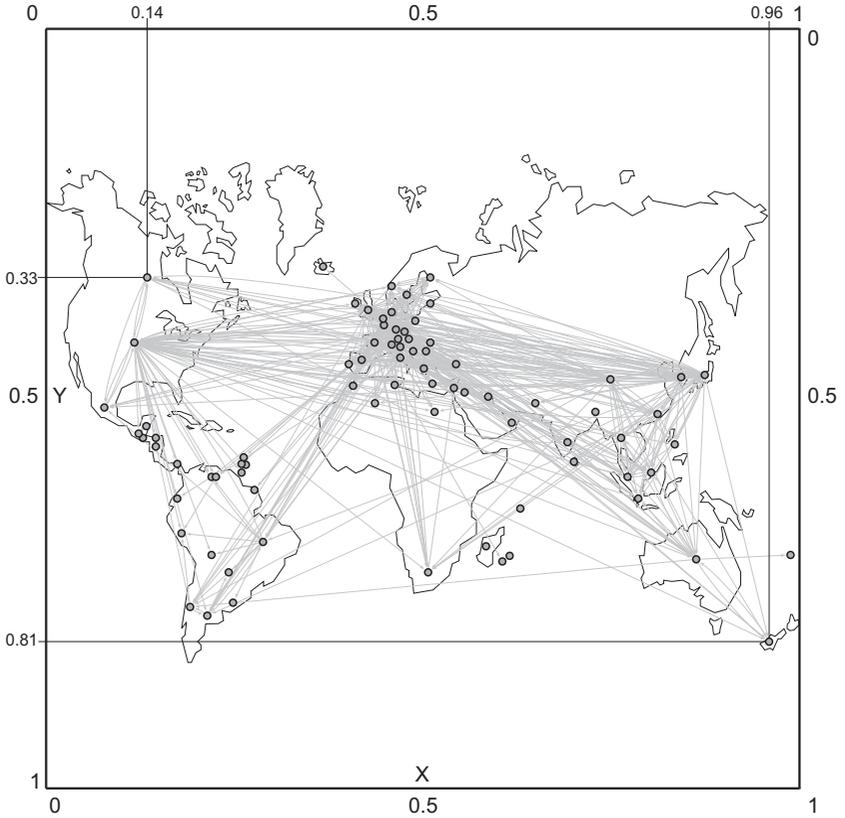


Figure 26. Geographical view of world trade in manufactures of metal, ca. 1994.

continuous attribute that is measured independently of the network and added to the graph as additional data: It is a continuous attribute.

Vectors store quantities, which are used for calculations, for example, average centrality, and displayed as sizes, for example, vertex size or line width. Some structural features that yield integer scores are used for calculations rather than for partitioning the network into subnetworks. In this case, Pajek may store integer values as a vector instead of a partition. Anyway, vectors can be transformed into partitions and the other way around, as we will see in the Application Section.

A special continuous property of a vertex is its location in a sociogram. Location is expressed by coordinates, real numbers that correspond with positions on one or more axes. For example, the two-dimensional Draw screen has two axes, a horizontal x -axis and a vertical y -axis. Both axes range from 0 to 1 (see Figure 26). The position of a vertex in the Draw

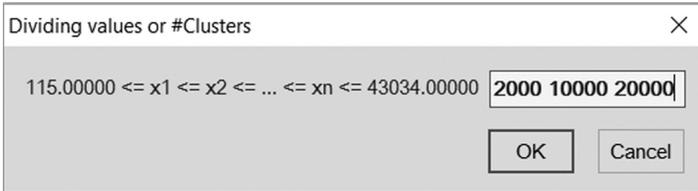


Figure 27. *Vector> Info* dialog box.

screen is determined by two coordinates (e.g., Canada is located at 0.14 on the x -axis and 0.33 on the y -axis).

Because coordinates are real numbers, they can be stored as vectors: one representing the location of vertices on the x -axis and the other representing locations on the y -axis. In Figure 26, we did something special: We made sure that locations in the Draw screen matched geographical locations on a map of the world projected behind the network (see Appendix 2 for details on how to produce this image). Now, vectors express geographical coordinates, and we can see geographical patterns in a social network.

Application

In Pajek, vectors have a drop-down menu of their own. A vector (e.g., the vector `GDP_1995.vec` containing GDP per capita in 1995) can be opened, edited, and saved like a network or a partition. Basic information can be gathered from the output of the *Vector> Info* command. When you execute this command, you can ask for a list of highest or lowest vector values. In a second dialog box, you can specify the number or boundaries of the classes of values that will be reported in the frequency distribution (Figure 27). Note that vector values are supposed to be continuous, so each value probably occurs only once. As a consequence, it is not informative to list each separate value as an entry in the frequencies table; values must be joined into classes. You may either list selected values that must be used as class boundaries (as in Figure 27) or specify the number of classes preceded by the pound (#) sign.

In the Report screen, the *Vector> Info* command outputs some statistics on the vector values; for instance, it tells us that GDP per capita ranges from US\$115 to US\$43,034, with an average of US\$10,249.94 and standard deviation US\$10,834.75. Then it prints the requested frequencies table, which is shown in Table 2. The first entry of this table contains the twenty-two countries with GDP per capita ranging from the lowest value up to and including US\$2,000. The second entry contains the twenty-seven countries with GDP per capita of more than US\$2,000 up to and including US\$10,000, and so on. In the first column, the round bracket means “from” and the square bracket means “up to and including.”

File> Vector

Vector> Info

Table 2. *Distribution of GNP per capita in classes*

	Vector Values		Freq	Freq%	CumFreq	CumFreq%
(... 2000]		22	27.5000	22	27.5000
(2000 ... 10000]		27	33.7500	49	61.2500
(10000 ... 20000]		15	18.7500	64	80.0000
(20000 ... 43043]		16	20.0000	80	100.0000
TOTAL			80	100.0000		

Partition>
Copy to Vector

As we discussed before, partitions and vectors have different applications: Partitions serve to select subsets of vertices from a network, whereas vectors specify numerical properties that can be used in calculations. Sometimes, however, you may want to change partitions into vectors or vice versa. It is very easy to convert a partition to a vector: Just use the *Copy to Vector* command in the *Partition* menu. Note, however, that conversion may be meaningful only if classes in a partition express a quantity – if they are numbers that can meaningfully be added, subtracted, and so on.

Vector> Make
Partition>
Copy to
Partition by
Truncating
(Abs)

The translation of a vector into a partition is more complicated because you have to change real numbers into integers, which can be done in several ways in the *Vector> Make Partition* submenu. Truncation is the easiest way, which means that you drop the decimals from the real numbers in the vector to obtain integers that can be stored in a partition. A GDP per capita of US\$115, up to but not including US\$116, is changed to class 115 in a new partition. In the command name, *Abs* stands for absolute, which means that negative vector values are transformed into positive class numbers because partitions cannot hold negative integers.

Vector> Make
Partition> by
Intervals> First
Threshold and
Step

Another way to convert a vector into a partition is to recode vector values into classes of fixed width; for instance, recode GDP per capita into classes of US\$10,000. Use the command *by Intervals> First Threshold and Step* in the *Vector> Make Partition* submenu and specify the upper bound of the lowest class as the first threshold (e.g., 10000 for a class including a minimum value up to and including 10,000) and class width as the step (e.g., fill in 10000 again). Note that Pajek does not accept a comma separating thousands from hundreds.

Vector> Make
Partition> by
Intervals>
Selected
Thresholds

In the case of GDP per capita, a conversion of values to classes with fixed width is not very useful. GDP per capita is unevenly distributed, so many countries are lumped together in the lowest classes, whereas higher classes contain few countries. In this example, it is better to create classes of unequal width, namely narrow classes for low values and wider classes for higher values, with the command *Selected Thresholds* in the *Make Partitions by Intervals* submenu. In the dialog box, specify the boundaries between classes (e.g., 2000, 10000, and 20000). Just type the numbers separated by spaces and do not use \leq as specified in the top

of the dialog box. The \leq sign means that all classes include the upper boundary.

In a sociogram, vector values are represented by the size of vertices if you use *Network + First Vector* or *Network + First Partition + First Vector* from the *Draw* menu. The area of a vertex is proportional to its vector value. Note that GDP per capita ranges from US\$115 to US\$43,034, but in the Draw screen vertices usually have sizes between 2 and 4. If you draw a network with vector sizes ranging from 115 to 43034, vertices are so large that they do not fit in the screen. Pajek may issue a warning that it changes the size of the vertices. If vertices are still too large, you can adjust the vector values to get reasonable sizes in a drawing. The *Options> Size> of Vertices* command of the Draw screen offers the simplest way to achieve this. In the dialog box issued by this command, enter 0 to activate the *AutoSize utility* of Pajek, which computes optimal vertex sizes automatically.

In Figure 28, countries with higher GDP per capita have larger vertices, according to their vector values. We can see that the wealthiest countries are part of the core and strong semiperiphery, but note that geographical location is also important; Scandinavian countries have similar (high) GDP per capita, whether they belong to the core (Sweden) or to the semiperiphery (Norway, Denmark, and Finland).

Representation of vector values by size of vertices is not useful for negative numbers, because negative size is meaningless. Pajek ignores negative signs of vector values when it computes the size of a vertex in a drawing. Vertices with large negative values are drawn as big as vertices with large positive values, which is quite misleading. Always check whether negative vector values exist with the *Vector> Info* command.

Note that you can also draw a network with two vectors (*Network + First Vector + Second Vector*). Select the first vector in the first *Vector* drop-down menu and the second vector in the second *Vector* drop-down menu. The first vector will define the size of the vertex in the horizontal direction and the second in the vertical direction. In this case, vertices will be drawn as ellipses instead of circles. You may, for example, use the population size (in millions) in 1995 (vector *Population_1995.vec*) as the first vector and GDP per capita in that year as the second vector. If you click the *Draw> Network + First Vector + Second Vector* command now, the resulting drawing will not be informative, because the two vectors are measured on different scales. It is better to normalize them first and in this case we advise to divide the vector values by their maximum: execute the *Vector> Transform> Normalize> Max* command on both vectors. A drawing with the two new vectors will be better, although you might want to increase vertex size (*[Draw] Options> Size> of Vertices*) to have a clearer view of countries with small populations and large GDP per capita (vertically stretched) and countries with large populations but

Draw>
Network +
First Vector

Draw>
Network +
First Partition
+ First Vector

Options>
Size> of
Vertices

Draw>
Network +
First Vector +
Second Vector

Vector>
Transform>
Normalize>
Max

[Draw]
Options>
Size> of
Vertices

Draw>
Network +
First Partition
+ First Vector
+ Second
Vector

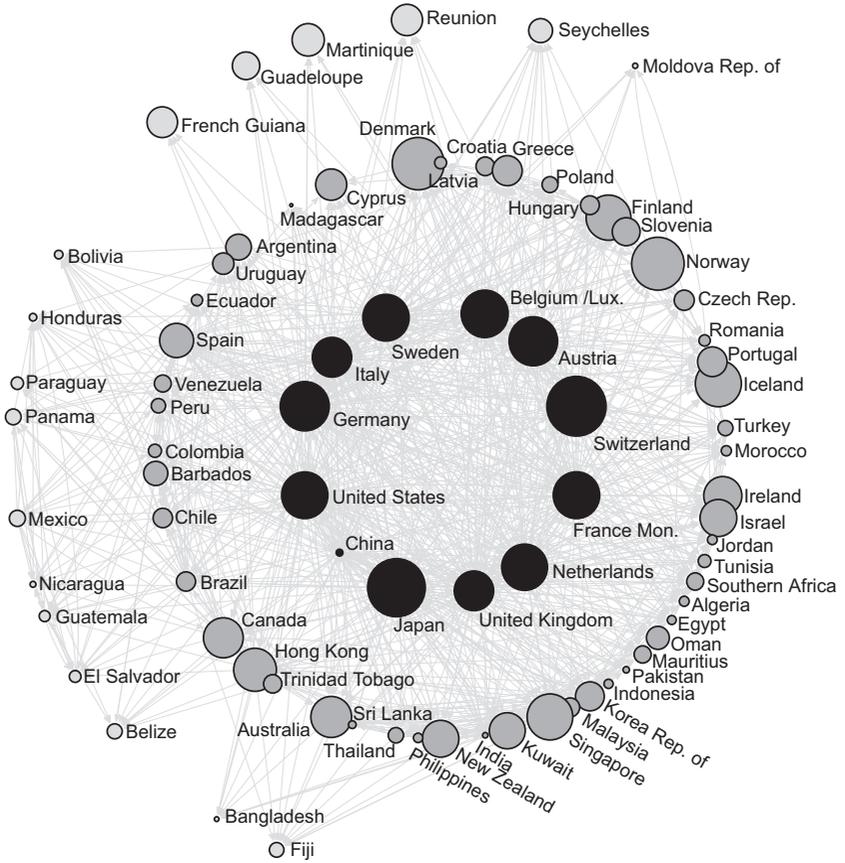


Figure 28. Trade, position in the world system, and GDP per capita.

low GDP per capita (horizontally stretched). It is possible to have a partition determine vertex color in this drawing (*Draw> Network + First Partition + First Vector + Second Vector*).

Options>
Mark Vertices
Using> Vector
Values

Vector values may also be displayed as vertex labels in the Draw screen: Select the option *Vector Values* in the *Options> Mark Vertices Using* submenu. In most networks, the result is visually not very attractive, but it is a good way to check the exact vector values of vertices in a drawing.

FishEye>
Cartesian,
Polar, Factor

In case of denser layouts (like the one in Figure 28) you may find the *FishEye* magnifier tool in the main menu of the Draw window very useful. By selecting the *Cartesian* or *Polar* transformation you can put the Draw window into a *FishEye mode*. While the Draw window is in the *FishEye mode* hovering the mouse pointer over the Draw window magnifies the area around the mouse pointer. By setting *Factor* higher or lower than the

default value (3), you can increase and decrease the magnification around the mouse pointer. When you want to quit the *FishEye mode* simply click with the mouse button somewhere in the Draw window. Pajek will ask you if you want to keep the obtained layout. By answering “Yes” you will keep the current positions of vertices. By answering “No” you will restore the positions of vertices to the ones before entering into *FishEye mode*, which is also the result of *FishEye> Exit*.

In Section 2.4.1, we learned how to select a part of a network (e.g., the trade in manufactures of metal among countries of one continent) and how to reduce a partition to fit the extracted network. In a similar way, we can adjust vectors to induced subnetworks. Recall that we need a partition to extract a network; a partition of countries according to their continent is needed for extracting countries of one continent from the network. We can use the same partition for extracting the vector values, for instance GDP per capita, of the selected countries. First, make sure that the partition that was used to reduce the network is selected in the *Partition* drop-down menu. Next, execute the command *Extract SubVector* in the *Operations> Vector + Partition* menu. Just like the command for extracting a partition, the *Extract SubVector* command asks the user to choose a class of vertices or range of classes that must be extracted: Enter 6 to select South American countries. Finally, Pajek adds the reduced vector to the *Vector* drop-down menu.

Operations>
Vector +
Partition>
Extract
SubVector

To obtain a global or contextual view, we shrink networks (see Sections 2.4.2 and 2.4.3). Recall that a class of vertices is replaced by one new vertex in a shrunken network; we replaced all countries belonging to the same continent with a vertex representing the continent. To use vector values associated with the original network (e.g., GDP per capita of a country) we must also shrink the vector. The command *Shrink Vector* in the *Operations> Vector + Partition* submenu accomplishes this provided that you have selected the relevant vector and the partition that you want to use for the shrinking, namely *Continent.clu*. The command allows you to choose among taking the sum, mean, minimum, maximum, or median of the vertices in a class that will be shrunk. In the example of GDP per capita, it makes sense to calculate the mean or median GDP per capita on a continent.

Operations>
Vector +
Partition>
Shrink Vector

As noted, coordinates of vertices can be thought of as vectors. In Chapter 1, we learned that coordinates of vertices are saved in the network data file, so we do not need vectors to store the location of vertices. However, vectors of coordinates are useful if, for instance, you want to use predefined geographical coordinates or coordinates from one network in another network with the same vertices. If you have the coordinates of all vertices (or any other vector) in a matrix with the vertices sorted in the same order as in the network, you can produce a

File> Vector>
Read

simple text file for each coordinate (or vector) or one text file containing all coordinates (or vectors) that can be read by Pajek (*File> Vector> Read*). See the files `x_coordinates.vec`, `y_coordinates.vec`, and `world_coordinates.vec` for examples. Note that the file should *not* contain tabs or other special characters.

Network>
Create Vector>
Get Coordinate

Operations>
Network +
Vector>
Transform>
Put Coordinate

You can save the present coordinates of the vertices as vectors with the *Get Coordinate* command in the *Network> Create Vector* submenu. You must create a vector of coordinates for each axis separately. The *Put Coordinate* command in the *Operations> Network + Vector> Transform* submenu loads coordinates from vectors: one vector for each axis. In our example, geographical location of countries can be read from two vectors: `x_coordinates.vec` and `y_coordinates.vec`. Note that the *Save* and *Load* commands are listed in different menus. For saving coordinates, you need only a network, but for loading coordinates from a vector you need a vector to load from and a network to add coordinates to. Therefore, the *Load* command is situated in the *Operations* menu.

Options>
Transform> Fit
Area

When you load coordinates from a file that was not created by Pajek, the Draw screen may be blank. This happens if the coordinates are not in the range between 0 and 1. In this case, use the *Options> Transform> Fit Area* command in the Draw screen, and Pajek will fit the sociogram to the size of the screen.

Options>
Layout> Real
xy Proportions

Networks>
Union of
Vertices

Now you have the knowledge to produce a drawing like Figure 26: a network against the background of a geographical map. The basic trick is to conceive of the map as a network with zero-sized vertices. The network `World.net` represents the map of the world (more or less a Mercator projection) provided that the Draw screen is square. Resizing is constrained to a square if the *Options> Layout> Real xy Proportions* is set. Vertices are invisible because their shape is set to *empty* (see Appendix 2). Now we have two networks: the map and the trade relations. If we add the geographical *x* and *y* coordinates from `x_coordinates.vec` and `y_coordinates.vec` (see earlier), we only have to combine the two networks, which is accomplished with the *Networks> Union of Vertices* command if `World.net` is selected in the first *Network* drop-down menu and the second *Network* drop-down menu contains the trade network with geographical coordinates.

Exercise III

Shrink all countries to continents except the North American countries in the network of trade in metal manufactures. Remove the lines between the continents and trade ties with values less than US\$5 million (recall that line values reflect trade in thousands of US dollars). Draw the energized network with vertex sizes reflecting (mean) GDP per capita. Describe the structure that you see.

2.6 Network Analysis and Statistics

In this chapter, we use attributes of vertices in social network analysis; for instance, we compare GDP per capita of countries to their visual positions in the trade network. We find that countries in the core of the world trade system have higher GDP per capita than countries in the periphery. Thus, social network analysis handles relational data as well as attributes of vertices.

Attributes such as GDP per capita are usually analyzed with statistical techniques. GDP per capita is compared, for instance, to population growth and level of education to find out which properties of countries are associated. GDP per capita is found to be higher in countries with a lower population growth and a higher level of education. Statistics offers a wide range of techniques to describe attributes and investigate the association between attributes. If we are able to express structural properties of vertices as attributes or properties of actors, however, they can be included in statistical analysis to determine their association. This is the case with the position of countries in the world system, which was calculated from the world trade network and stored in a partition. Continuous structural indices, such as vertex centrality (Chapter 6), can be stored as vectors.

In this book, some examples of statistical analysis are given but are restricted to basic statistical techniques that are incorporated in Pajek. After all, this is not a book on statistics. Nevertheless, the link between statistics and social network analysis should be well understood, because the two techniques make up a powerful combination.

Application

One of the basic statistical techniques implemented in Pajek is the cross-tabulation of two partitions and some measures of association between the classifications represented by two partitions. Let us look at the world trade example. Our partition of positions in the world trade network was derived from the trade in manufactures of metal around 1994. In addition, we have at our disposal a classification of countries according to their position in the world system in 1980 that was proposed by Smith and White (`World_system_1980.clu`). To analyze the transition of countries between world system positions from 1980 to 1994, we can use simple statistical techniques.

First, we must select the two partitions that we want to compare. Select the partition with world system positions in 1980 (`World_system_1980.clu`) in the first *Partition* drop-down menu. Next, select the world system positions partition in 1994 (`World_system.clu`) in the second *Partition* drop-down menu. To obtain a cross-tabulation

Table 3. *Output of the Info command*

Cramer's V, Rajsiki, Adjusted Rand Index				
Rows: 1. World_system_1980.clu (80)				
Columns: 3. World_system.clu (80)				
	1	2	3	Total
1	10	1	0	11
2	1	16	0	17
3	0	15	0	15
4	0	4	5	9
TOTAL	11	36	5	52

Warning: 28 vertices with missing values excluded from cross tabulation!

Warning: 8 cells (66.67%) have expected frequencies less than 5!

The minimum expected cell frequency is 0.87!

Chi-Square: 66.2597

Cramer's V: 0.7982

Rajsiki (C1↔C2): 0.3422

Rajsiki (C1→C2): 0.6827

Rajsiki (C1←C2): 0.4069

Adjusted Rand Index: 0.3299

and measures of association, the selected partitions must refer to the same network: the same number of vertices in the same order. It is meaningless to compare partitions that do not refer to exactly the same vertices.

Partitions>
Info> Cramer's
V, Rajsiki,
Adjusted Rand
Index

Second, select the *Info> Cramer's V, Rajsiki, Adjusted Rand Index* command from the *Partitions* menu. Pajek will show a cross-tabulation and some measures of association in the Report screen. Table 3 contains the results of the *Info* command. In the cross-tabulation, the rows contain the four classes according to world system position in 1980: core countries in class 1, strong semiperiphery in class 2, weak semiperiphery in class 3, and periphery in class 4. The columns contain the three world system positions in 1994: core countries in class 1, semiperiphery in class 2, and periphery in class 3. Countries with an unknown world system position are automatically omitted by Pajek; their numbers are reported below the table. As a result, the first column represents eleven of the twelve countries in the core of the trade system in 1994. We do not know the world system position of China in 1980, so it is placed in class 99999998 in the *World_system_1980.clu* partition, which is handled as a missing value. Note that rows and columns are swapped when the first and second partitions are exchanged in the *Partitions* menu.

From this table, we may conclude that the composition of the core has hardly changed between 1980 and 1994 (see the row and column of the cross-tabulation labeled "1"). The countries in the strong and weak semiperiphery in 1980 constitute the major part of the semiperiphery in 1994 (column 2), and four countries were promoted from the periphery in 1980 to the semiperiphery in 1994 (row 4, column 2).

Statistical indices of association tell us how strong the association is. Indices range from 0 to 1, and as a rule of thumb we may say that values between 0 and 0.05 mean that there is no association, values between 0.05 and 0.25 indicate a weak association, values from 0.25 to 0.60 indicate a moderate association, and values higher than 0.60 indicate a strong association.

In Pajek, three types of association indices are computed: Cramer's V, Rajski's information index, and Adjusted Rand Index. Cramer's V measures the statistical dependence between two classifications. It is not very reliable if the cross-tabulation contains many cells that are (nearly) empty, so Pajek issues a warning if this is the case. Rajski's indices measure the degree to which the information in one classification is preserved in the other classification. It has three variants: a symmetrical version, represented by $(C1 \leftrightarrow C2)$ in the output of Pajek, and two asymmetrical versions, which indicate the extent to which the first classification can be predicted by the second ($C1 \leftarrow C2$) or the second classification can be predicted by the first ($C1 \rightarrow C2$). The Adjusted Rand Index (ARI) is another index of the similarity between two partitions. Its values range from a minimum of 0 (independent partitions – no association) to maximum of 1 (identical partitions – maximum association).

In our example, Cramer's V is not very reliable because many cells have low expected frequencies. Rajski's information indices suggest that world system position in 1994 can be predicted quite well from the positions in 1980: Rajski $C1 \rightarrow C2$ is 0.68. The two world system positions are strongly associated. The Adjusted Rand Index (0.33) suggests a moderate association between the two partitions.

Exercise IV

Determine the statistical association between the position of countries in the world system of trade in metal manufactures in 1994 and their GDP per capita in 1995. Hint: Translate the GDP per capita vector into a partition with four classes: 0 to US\$2,000; US\$2,000 to US\$10,000; US\$10,000 to US\$20,000; and US\$20,000 and higher.

2.7 Summary

In this chapter, we used properties of vertices to find and interpret patterns of ties in a network. These properties are stored in partitions or vectors. Both partitions and vectors are lists of numbers, one number for each vertex in a network, but numbers in partitions refer to discrete classes, whereas vector values express continuous properties of vertices. Classes in partitions are represented by integers (e.g., countries on the African continent have code, and negative class numbers are not allowed). Vectors contain real numbers, which can be negative.

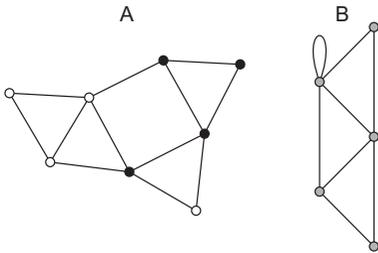
Social networks are often large and complicated. To understand network structure, it helps to study reductions of the network first. Partitions can be used to reduce a network in two ways: by extraction and by shrinking. Extraction is the selection of a subset of vertices from a network as well as the lines among these vertices. You can now concentrate on the structure of a part of the network, which is called a local view. In contrast, shrinking a network means that you lump together sets of vertices and lines incident with these vertices (e.g., you replace all African countries with one vertex representing the African continent). This yields a global view of the network if all classes are shrunk and a contextual view if one class of vertices is not shrunk. In all cases, classes of a partition define the subsets of vertices that are extracted or shrunk.

Properties of social actors that are not based on their structural position in a network are called attributes. They are added to a network to enhance the analysis and interpretation of its structure. Partitions and vectors, however, can also contain structural indices of vertices, which result from network analysis (e.g., the world system position of countries that is inferred from the trade network). These properties may be included in statistical analysis, so partitions and vectors are the nexus between social network analysis and statistics.

2.8 Questions

1. Which of the following statements is correct?
 - a. Each property of a vertex or a line is an attribute.
 - b. Each property of a vertex is an attribute.
 - c. Each nonrelational property of a vertex is an attribute.
 - d. Each nonrelational property of a line is an attribute.
2. There are two lists of numbers. Numbers in list A range from -1 to 1 , and list B contains numbers between 1 and 10 . Which statements can be correct?
 - a. A and B are partitions.
 - b. A is a partition, and B is a vector.
 - c. A is a vector, and B is a partition.
 - d. A and B are vectors.
3. Suppose that the average population growth rate between 1990 and 1995 is available for the countries in the trade network, coded as $0 =$ negative growth, $1 = 0$ to 1.0 percent, $2 = 1.0$ to 3.0 percent, and $3 = 3.0$ percent and higher. Would you use a partition or a vector for these data?
 - a. A partition because we cannot add or subtract the codes meaningfully.
 - b. A partition because percentages are recoded to integers.

- c. A vector because negative population growth is possible.
 - d. A vector because percentages are “real” numbers.
4. What happens to the sociogram of a network if the order of numbers in the accompanying partition is changed?
 - a. Nothing changes in the sociogram.
 - b. Pajek does not draw the partition any longer.
 - c. Pajek uses other colors to draw the vertices.
 - d. Vertices are drawn in the wrong color.
 5. Which of the following statements is correct about the networks in the figure below?
 - a. A is extracted from B.
 - b. B is extracted from A.
 - c. A is shrunk from B.
 - d. B is shrunk from A.



6. Which of the following statements is correct about the networks of Question 5?
 - a. B is a contextual view of A.
 - b. A is a global view of B.
 - c. B is a local view of A.
 - d. A is a local view of B.

2.9 Assignment

Analyze the manufactures of metal trade network of Asian countries (local view) and their position with regard to other continents (contextual view). Does the structure of trade ties match the world system positions of countries, their prosperity (indicated by GDP per capita), and their geographical locations? Try to apply the theory of the world system, which was outlined in Section 2.2.

2.10 Further Reading

- In *The Modern World System: Capitalist Agriculture and the Origins of the European World-Economy in the Sixteenth Century*

(New York, NY: Academic Press, 1974), Immanuel Wallerstein introduced the concept of a world system. The *Capitalist World-Economy* (Cambridge/Paris: Cambridge University Press & Editions de la Maison des Sciences de l'Homme, 1979) is a collection of essays that introduces the theory of a world economy. V. Bornschier and B. Trezzini ("Social stratification and mobility in the world system – Different approaches and recent research." *International Sociology* 12 [1997], 429–55) offer a summary of research traditions that use the concept of a world system or world economy.

- See D. Snyder and E. L. Kick ("Structural position in the world system and economic growth 1955–70." *American Journal of Sociology* 84 [1979], 1096–126) for the first (published) application of social network analysis to world systems theory, and see D. A. Smith and D. R. White ("Structure and dynamics of the global economy: Network analysis of international-trade 1965–1980." *Social Forces* 70 [2002], 857–93) for the research design followed here.
- Our data on imports are taken from the Statistical Papers. Commodity Trade Statistics (Series D Vol. XLIV, No. 1–34 [1994]) published by the United Nations, and we gathered additional economic and demographic data from the *Statistical Yearbook of the United Nations* (Ed. 43, IVATION Datasystems Inc.). Imports of miscellaneous manufactures of metal are listed as SITC code 69 in the commodity trade statistics.

2.11 Answers

Answers to the Exercises

- I. If you want to fix the locations of the countries of the semiperiphery and periphery in the Draw screen, you must create a partition in which the core countries belong to class 0 and the other countries to another class. The easiest way to achieve this is to create and draw a new empty partition with the *Draw> Network + Create Null Partition* command in the Main screen. The Draw screen opens and shows the world trade network with all vertices in class 0 (light blue). Pressing down the *Shift* key, click somewhere between the vertices to raise all class numbers to 1. Then, zoom in on the core countries, dragging your mouse around them while pressing the right mouse button. After you have zoomed in on the core countries, lower their class number back to 0 by clicking between the vertices while pressing the *Alt* key. Press Redraw in the Draw screen, and you will see that the core countries are the only ones that belong to class 0 in

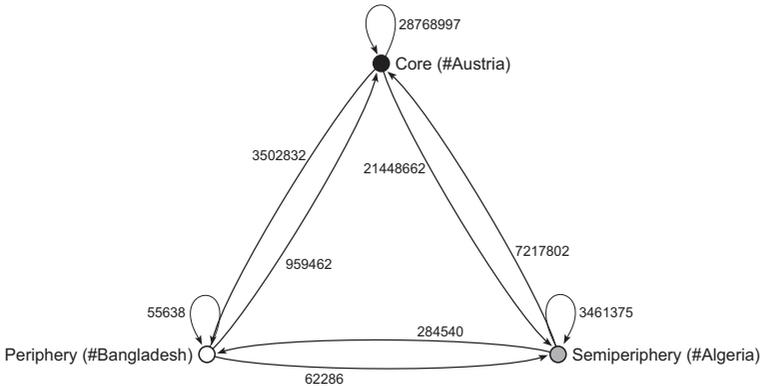


Figure 29. Aggregate trade in manufactures of metal among world system positions.

the partition. Now you have the right partition, and you can select the *Layout > Energy > Kamada-Kawai > Fix Selected Vertices* command to optimize the location of the core countries with respect to the other countries. You will probably see that Japan is pulled toward the Latin American countries, Canada, and the Philippines, whereas Austria moves in the direction of the Eastern European countries (Czech Republic, etc.).

- II. Make sure that the partition with world system positions is active in the *Partition* drop-down menu and select the original trade network in the *Network* drop-down menu. Shrink the network with the *Operations > Network + Partition > Shrink Network* command. The energized shrunken network may look like the sociogram depicted (in Figure 29), although the loops are not visible in the Draw screen. The shrunken network shows several characteristics of the world system. First, the values associated with the loops reveal that trade between countries within one position of the world system decreases from core to periphery: from US\$28 billion within the core to US\$55 million within the periphery. Second, values of arcs between world system positions show asymmetries in world trade. Core countries export goods at a much higher value to the semiperiphery than vice versa, and the semiperiphery exports more to the periphery than the other way around. The values of arcs indicate that the larger the difference in world system position, the larger the difference between exports and imports.
- III. In the partition according to continents, `Continent.clu`, the North American countries constitute class 4. Shrink the network according to this partition, entering 4 in the dialog box asking for the class that must not be shrunk. Now you have the contextual view of the North

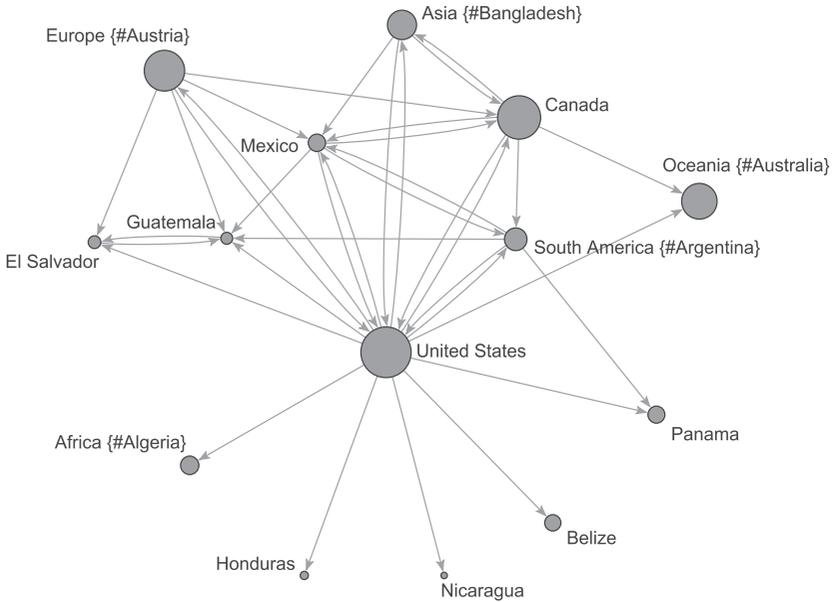


Figure 30. Contextual view of North American trade ties and (mean) GDP per capita.

American countries, from which you can remove the lines among the shrunken continents (see Section 2.4.3) and the lines with values less than 5000 (command *Network > Create New Network > Transform > Remove > Lines with Value > lower than*). Make sure that you select the original partition according to continents again in the *Partition* drop list before you apply the *Operations > Vector + Partition > Shrink Vector > Mean* command to the GDP per capita vector. Finally, draw the shrunken network with vector values (command *Draw > Network + First Vector* in the Main screen) and energize it. The result should be similar to the sociogram depicted in Figure 30. The United States occupies a central place in the trade of metal manufactures among the North American countries. South America and to a lesser extent Europe play an important part in the trade ties of the smaller countries on the North American continent.

- IV. You can extract a partition from the GDP vector with the *Make Partition > by Intervals > Selected Thresholds* command (see Section 2.5). Select the world system positions partition as the first partition and the GDP partition as the second. The *Info > Cramer's V, Rajski, Adjusted Rand Index* command produces Table 4. The table shows that the core countries have higher GDP per capita (classes 3 or 4) than countries in the semiperiphery and periphery, which are found

Table 4. *Cross-tabulation of world system positions (rows) and GDP per capita (columns)*

3. World_system.clu (80) / 4. From Vector 3 [2000 10000 20000] (80)		1	2	3	4	Total
1		1	0	2	9	12
2		13	21	10	7	51
3		8	6	3	0	17
TOTAL		22	27	15	16	80

Warning: 7 cells (58.33%) have expected frequencies less than 5!
 The minimum expected cell frequency is 2.25!
 Chi-Square: 31.5510
 Cramer's V: 0.4441
 Rajski (C1↔C2): 0.0962
 Rajski (C1→C2): 0.1460
 Rajski (C1←C2): 0.2200
 Adjusted Rand Index: 0.0782

in the lower GDP classes. The association between world system position and GDP per capita, however, is weak according to Rajski's indices and Adjusted Rand Index (0.08). The low GDP per capita of China, which is one of the core countries, is partly responsible for the moderate association.

Answers to the Questions in Section 2.8

1. Statement c is correct. Attributes (e.g., the continent of a country) are properties of vertices (viz., countries) and not lines; therefore, statements a and d are incorrect. Recall that properties of lines are called line values. Properties that express the structural position of a vertex in a network (e.g., centrality) are distinguished from social, economic, psychological, and other characteristics that do not measure network position. We restrict the concept of an *attribute* to the latter, nonrelational, kind of property.
2. Statements c and d can be correct. List A contains negative numbers, so it must be a vector. List B contains positive numbers, so it is either a partition (if it contains integers) or a vector.
3. Statement a is the right choice. Aggregation of the percentages into four classes of unequal width implies that we cannot make meaningful calculations on the class numbers; for instance, add and divide class numbers of two countries. We cannot say that two countries have average population growth of 2.5. Classes of equal width, such as rounding to the nearest integer, would yield "real" numbers, which may safely be used in calculations. But this is not the case here (statement b). The properties of the original percentages no longer matter after they have been aggregated in classes, so statements c and d are incorrect.

4. Answer d is correct. Changing the order of numbers in a partition does not change the total number of entries or the class numbers. The number of entries of a partition must match the number of vertices in the network; otherwise Pajek does not draw the partition. Answer b is incorrect. Class numbers determine the colors used. Because they do not change, the same colors are used (answer c is incorrect). The order of entries in a partition determines which class is assigned to which vertex: the class number in the first entry of a partition is assigned to the first vertex in the network, and so on. Changing the order of entries in a partition changes the links between vertices and entries with class numbers, so at least some vertices will be drawn in other colors (answer d).
5. Statement d is correct. Network B contains fewer vertices than network A, so B is extracted or shrunk from A and not the other way around (statements a and c). B is shrunk from A (statement d), because we find a loop in B but no loops in A and because there is no subset of five vertices in network A that are connected like B: three triangles that share a line with at least one other triangle. In fact, the white vertices in network A are shrunk to one new vertex, namely the vertex with the loop in network B.
6. Answer a is correct. B is shrunk from A, but not all vertices are shrunk: The black vertices are not shrunk (see answer to Question 5). A partially shrunken network offers a contextual view.

Part II

Cohesion

Solidarity, shared norms, identity, collective behavior, and social cohesion are considered to emerge from social relations. Therefore, the first concern of social network analysis is to investigate who is related and who is not. Why are some people or organizations related, whereas others are not? The general hypothesis here states that people who match on social characteristics will interact more often, and people who interact regularly will foster a common attitude or identity.

In this part of the book, which covers Chapters 3 through 5, we discuss several measures of cohesion. You will learn to detect cohesive subgroups within several types of social networks.

Cohesive Subgroups

3.1 Introduction

Social networks usually contain dense pockets of people who “stick together.” We call them cohesive subgroups, and we hypothesize that the people involved are joined by more than interaction. Social interaction is the basis for solidarity, shared norms, identity, and collective behavior, so people who interact intensively are likely to consider themselves a social group. Perceived similarity, for instance, membership of a social group, is expected to promote interaction. We expect similar people to interact a lot, at least more often than with dissimilar people. This phenomenon is called homophily or assortativity: Birds of a feather flock together. We will learn how to measure this phenomenon in Chapter 6.

In this chapter, we present a number of techniques to detect cohesive subgroups in social networks, all of which are based on the ways in which vertices are interconnected. These techniques are a means to an end rather than an end in themselves. The ultimate goal is to test whether structurally delineated subgroups differ with respect to other social characteristics, for instance, norms, behavior, or identity. Does the homophily principle work? May we conclude that a cohesive subgroup represents an emergent or an established social group?

3.2 Example

In 1948, American sociologists executed a large field study in the Turrialba region, which is a rural area in Costa Rica (Latin America). They were interested in the impact of formal and informal social systems on social change. Among other things, they investigated visiting relations between families living in *haciendas* (farms) in a neighborhood called Attiro. The network of visiting ties (*Attiro.net*, drawn in Figure 31)

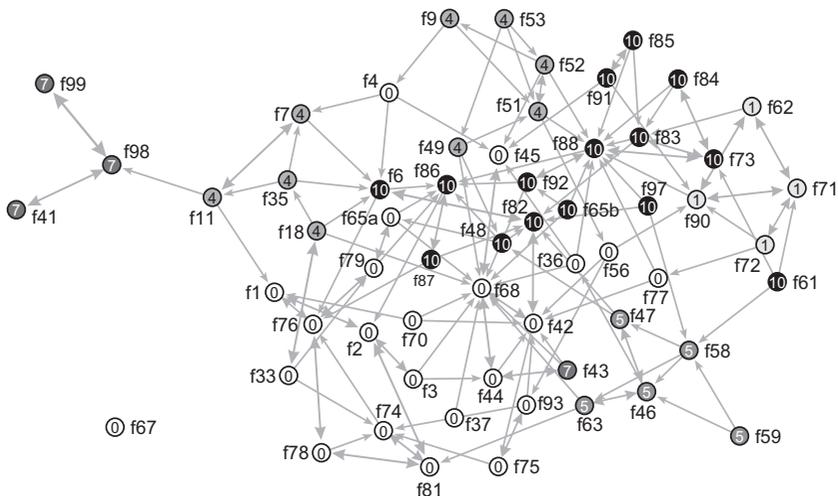


Figure 31. Visiting ties in Attiro.

is a simple directed graph: Each arc represents “frequent visits” from one family to another. The exact number of visits was not recorded. Line values classify the visiting relation as ordinary (value 1), visits among kin (value 2), and visits among ritual kin (i.e., between godparent and godchild); but we do not use them in this chapter. Loops do not occur because they are meaningless.

We compare cohesive subgroups in this network with an ethnographic classification of the families into six family–friendship groupings that were made by the researchers on substantive criteria (`Attiro_grouping.clu`; we adjusted the class numbers to get optimal graphs with the `Options> Colors> Partition Colors> for Vertices> Default Greyscale 2` option). In rural areas where there is little opportunity to move up and down the social ladder, social groups are usually based on family relations. All relevant data are collected in the project file `Attiro.paj`. Open this file now, and draw the network with its partition to obtain a sociogram like Figure 31 (`Draw> Network + First Partition` command). You may want to use real colors instead of grays for easy recognition of the family–friendship groupings.

Which cohesive subgroups can we find in the Attiro network, and do they match the family–friendship groupings? Figure 31 offers a visual impression of the kin visits network and the family–friendship groupings, which are identified by the colors and numbers within the vertices. As is shown, the network is tightly knit with family–friendship groupings 0 and 10 dominating the center. Most families that belong to one grouping are connected by visiting ties, so they are rather close in the

network. Exceptions occur, however; notably family f43 (bottom right), which is separated from the other vertices in the seventh family–friendship grouping (left). In subsequent sections, we set out this first impression in detail.

3.3 Density and Degree

Intuitively, cohesion means that a social network contains many ties. More ties between people yield a tighter structure that is, presumably, more cohesive. In network analysis, the density of a network captures this idea. It is the percentage of all possible lines that are present in a network. Maximum density is found in a complete simple network, that is, a simple network in which all pairs of vertices are linked by an edge or by two arcs, one in each direction. If loops are allowed, all vertices have loops in a complete network.

Density is the number of lines in a simple network, expressed as a proportion of the maximum possible number of lines.

A *complete network* is a network with maximum density.

In this definition of density, multiple lines and line values are disregarded. Intuitively, multiple lines between vertices and higher line values indicate more cohesive ties. Although density measures have been proposed that account for multiplicity and line values, we prefer not to present them. We count distinct lines only, which means that we treat a multiple line as one line and multiple loops as one loop. We discuss other measures that capture the contribution of multiple lines and line values to cohesion in Chapter 5.

In the kin visiting relation network, density is 0.045, which means that only 4.5 percent of all possible arcs are present. It is very common to find density scores as low as this one in social networks of this size. Density is inversely related to network size: The larger the social network, the lower the density because the number of possible lines increases rapidly with the number of vertices; whereas the number of ties that each person can maintain is limited. In a visiting relation network, there is a practical limit to the number of families you can visit. Therefore, including more families in the network will reduce network density.

This is a problem if you want to interpret or compare network density. Density in the visiting network in San Juan Sur, which is another neighborhood in the Turrialba region, is 0.036. This is slightly lower than in Attiro, but the difference may be due to a larger number of families in San

Juan Sur (seventy-five families). Therefore, we cannot draw a conclusion from this comparison.

The *degree* of a vertex is the number of lines incident with it.

Network density is not very useful because it depends on the size of the network. It is better to look at the number of ties in which each vertex is involved. This is called the degree of a vertex. Vertices with high degree are more likely to be found in dense sections of the network. In Figure 31, family f88 (a member of family–friendship grouping number 10) is connected to thirteen families by fifteen visiting ties (note that the double-sided arcs between f88 and f73, f92 indicate that these families are linked by mutual visits), so its degree is 15. The lines incident with this family contribute substantively to the density of the network near this family.

A higher degree of vertices yields a denser network, because vertices entertain more ties. Therefore, we can use the *average degree* of all vertices to measure the structural cohesion of a network. This is a better measure of overall cohesion than density because it does not depend on network size, so average degree can be compared between networks of different sizes. For example, the average degree of the Attiro network is 5.37, which is slightly higher than the average degree of the San Juan Sur network (5.28).

Two vertices are *adjacent* if they are connected by a line.

The *indegree* of a vertex is the number of arcs it receives.

The *outdegree* is the number of arcs it sends.

In a simple undirected network, the degree of a vertex is equal to the number of vertices that are adjacent to this vertex: its *neighbors*. Each line that is incident with the vertex connects it to another vertex because multiple lines and loops, which contribute to the degree of a vertex without connecting it to new neighbors, do not occur. In a directed network, however, there is a complication because we must distinguish between the number of arcs received by a vertex (its indegree) and the number of arcs sent (its outdegree). Note that the sum of the indegree and the outdegree of a vertex does not necessarily equal the number of its neighbors; for instance, family f88 is involved in fifteen visiting ties, but it has thirteen adjacent families because families f73 and f92 are counted twice (Figure 31).

In this section, we restrict ourselves to degree in undirected networks. When we encounter a directed network, we symmetrize it, which means that we turn unilateral and bidirectional arcs into edges. A discussion of indegree in directed networks occurs in Chapter 9, which presents the concept of prestige.

To *symmetrize* a directed network is to replace unilateral and bidirectional arcs with edges.

Application

Let us analyze the network of visiting ties in Attiro (*Attiro.net*), which contains neither multiple lines nor loops. In Pajek, the density of a network can be obtained by means of the *Info* submenu of the *Network* menu in the Main screen. Choose the command *General* to display basic information on the selected network, such as the number of vertices and lines as well as its density. You can also press the “*I*” (*I* stands for “Info”) button at the left of the *Network* drop-down menu. When executed, this command displays a dialog box asking the user to specify the number of lines to be displayed. When you are only interested in network density and average degree, request zero lines. Pajek computes two density indices in the Report screen. The first index allows for loops, and the second does not. Because loops are meaningless in a visiting relation network – people do not visit themselves – the second index is valid. Density in the directed network is 0.045. Finally, the average degree is reported, which is 5.37 for Attiro.

[Main]
Network>
Info> General

In undirected simple networks, the degree of a vertex is equal to its number of neighbors. This is the easiest interpretation of degree, so we concentrate on undirected simple networks in this section. The kin visiting network, however, is directed, so we must symmetrize it first. Use the *Arcs→Edges> All* command in the *Network> Create New Network> Transform* submenu to replace all arcs with edges. Pajek will ask whether you want to make a new network, and we advise you to do so because you may want to use the directed network later. Next, Pajek asks whether you want to remove multiple lines. To obtain a simple undirected network, that is, a network without multiple lines and loops, you should choose option 1 (sum the line values of lines that will be joined into a new line), 2 (count the number of lines that are joined), 3 (preserve the minimum value of joined lines), 4 (take their maximum line value), or 5 (the value of the new line will be 1) in this dialog box. It does not matter which of these five options you choose because in this chapter we pay no attention to line values. Now, the network is symmetrized, and it is simple because multiple lines are removed and there were no loops in the original network.

Network>
Create New
Network>
Transform>
Arcs→Edges>
All

File>
Network> Save

Table 5. Frequency distribution of degree in the symmetrized network of visits

Clusters	Freq	Freq%	CumFreq	CumFreq%	Representative
0	1	1.6667	1	1.6667	f67
1	3	5.0000	4	6.6667	f37
2	1	1.6667	5	8.3333	f59
3	19	31.6667	24	40.0000	f3
4	20	33.3333	44	73.3333	f1
5	4	6.6667	48	80.0000	f45
6	6	10.0000	54	90.0000	f51
7	2	2.3333	56	93.3333	f6
8	1	1.6667	57	95.0000	f86
9	1	1.6667	58	96.6667	f42
13	1	1.6667	59	98.3333	f88
14	1	1.6667	60	100.0000	f68
SUM	60	100.0000			

You may want to save it (*File* > *Network* > *Save*) for future use under a new name (e.g., *Attiro_symmetrized.net*).

Degree is a discrete attribute of a vertex (it is always an integer), so it is stored as a partition. We obtain the degree partition with a command from the *Network* > *Create Partition* > *Degree* submenu: *Input*, *Output*, or *All*. *Input* counts all incoming lines (indegree), *Output* counts all outgoing lines (outdegree), and *All* counts both. Note that an edge, which has no direction, is considered to be incoming as well as outgoing, so each edge is counted once by all three degree commands. In an undirected network, therefore, it makes no difference whether you select *Input*, *Output*, or *All*.

The command *Partition* > *Info* displays the partition as a frequency table (see Table 5). Class numbers represent degree scores, so we can see that the degree of vertices varies markedly from zero to fourteen neighbors in the symmetrized network. Clearly, family f68 is connected to most families by visiting ties. One family, family f67, is isolated in the network: It is linked to no other families by regular visits.

The average degree of all vertices can be calculated from the degree distribution. In this example, the class numbers in the degree partition represent integers, namely the number of neighbors of a vertex, but this is not true for all partitions. As a consequence, no average class number is calculated and presented by the *Partition* > *Info* command. To obtain the average degree of the symmetrized network, we can use the *Network* > *Info* > *General* command again, which will report an average degree of 4.27. Families in Attiro have regularly visiting relations with more than four families on average. Note that this average degree differs from the average degree reported for the original directed network (5.37) because the latter sums indegree (visits received) and outdegree (visits paid), which may count families twice.

Network >
Create
Partition >
Degree

Partition > *Info*

Network >
Info > *General*

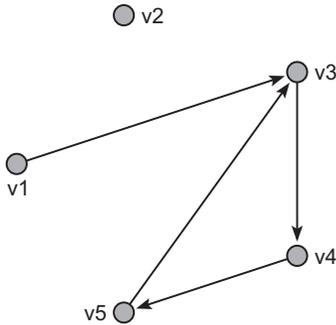


Figure 32. A simple unconnected directed network.

Exercise I

Open the visiting relation network in San Juan Sur (SanJuanSur.net), symmetrize it, and determine the average degree. Is this network more cohesive than the Attiro network in this respect?

3.4 Components

Vertices with a degree of one or higher are connected to at least one neighbor, so they are not isolated. However, this does not mean that they are necessarily connected into one lump. Sometimes, the network is cut up in pieces. Isolated sections of the network may be regarded as cohesive subgroups because the vertices within a section are connected, whereas vertices in different sections are not. The network of visits in Attiro is not entirely connected (see Figure 31). In this section, we identify the connected parts of a network, which are called components, but we must introduce some auxiliary graph theoretic concepts first.

Let us have a look at a simple example (Figure 32). Intuitively, it is clear that some vertices are connected to other vertices, whereas others are not; for instance, vertex v_2 is adjacent to no other vertex, but the other four vertices have one or more neighbors. If we consider the arcs to be roads, we can walk from vertex v_5 to v_3 and, not considering the direction of the arcs, we can proceed from vertex v_3 to v_1 . We say that there is a semiwalk from vertex v_5 to vertex v_1 . From vertex v_2 , however, we can walk nowhere.

A *semiwalk* from vertex u to vertex v is a sequence of lines such that the end vertex of one line is the starting vertex of the next line and the sequence starts at vertex u and ends at vertex v .

A *walk* is a semiwalk with the additional condition that none of its lines are an arc of which the end vertex is the arc's tail.

Imagine that the arcs represent one-way streets, so we take into account the direction of the arcs. Now, we can drive from vertex v_5 to vertex v_3 , but we cannot arrive at vertex v_1 . In graph theory, we say that there is a walk from vertex v_5 to v_3 , but there is not a walk from vertex v_5 to v_1 . In a walk, you have to follow the direction of the arcs.

Walks and semiwalks are important concepts, but we need another, related concept to define whether a network is connected. We should note that there are many – in fact, infinitely many – walks from vertex v_5 to v_3 in our example; for instance, $v_5 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5 \rightarrow v_3$ is also a walk, and we may repeat the circular route $v_5 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5$ as many times as we like. Clearly, we do not need these repetitions to establish whether vertices are connected, so we use the more restricted concepts of paths and semipaths, which demand that each vertex on the walk or semiwalk occurs only once, although the starting vertex may be the same as the end vertex. In the example, the walk $v_5 \rightarrow v_3$ is a path but the walk $v_5 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5 \rightarrow v_3$ is not because vertices v_5 and v_3 occur twice. A path is more efficient than a walk, one might say, because it does not pass through one junction more than once.

A *semipath* is a semiwalk in which no vertex in between the first and last vertex of the semiwalk occurs more than once.

A *path* is a walk in which no vertex in between the first and last vertex of the walk occurs more than once.

Now we can easily define the requirements a network must meet to be connected. A network is weakly connected – often we just say connected – if all vertices are connected by a semipath. In a (weakly) connected network, we can “walk” from each vertex to all other vertices if we neglect the direction of the arcs, provided that there are any. The example of Figure 32 is not connected because vertex v_2 is isolated: It is not included in any semipath to the other vertices.

In directed networks, there is a second type of connectedness: A network is strongly connected if each pair of vertices is connected by a path. In a strongly connected network, you can travel from each vertex to any other vertex obeying the direction of the arcs. Strong connectedness is more restricted than weak connectedness: Each strongly connected network is also weakly connected, but a weakly connected network is not necessarily strongly connected. Our example is not weakly connected, so it cannot be strongly connected.

A network is (weakly) *connected* if each pair of vertices is connected by a semipath.

A network is *strongly connected* if each pair of vertices is connected by a path.

Although the network of our example is not connected as a whole, we can identify parts that are connected; for instance, vertices $v_1, v_3, v_4,$ and v_5 are connected. In comparison with the isolated vertex v_2 , these vertices are relatively tightly connected, so we may say that they are a cohesive group. If the relation represents communication channels, all vertices except vertex v_2 may exchange information. Vertices $v_1, v_3, v_4,$ and v_5 constitute a (weak) component because they are connected by semipaths and there is no other vertex in the network that is also connected to them by a semipath.

Formally, we say that a (weak) component is a maximal (weakly) connected subnetwork. Remember that a subnetwork consists of a subset of the vertices of the network and all lines between these vertices. The word maximal means that no other vertex can be added to the subnetwork without destroying its defining characteristic, in this case connectedness. If we would add the only remaining vertex – v_2 – the subnetwork is no longer connected. In contrast, if we would omit any of the vertices $v_1, v_3, v_4,$ or v_5 , the subnetwork is not a component because it is not maximal: It does not comprise all connected vertices.

Likewise, we can define a strong component, which is a maximal strongly connected subnetwork. The example network contains three strong components. The largest strong component is composed of vertices $v_3, v_4,$ and v_5 , which are connected by paths in both directions. In addition, there are two strong components consisting of one vertex each, namely vertices v_1 and v_2 . Vertex v_2 is isolated and there are paths only from vertex v_1 but no paths to v_1 , so vertex v_1 is not strongly connected to any other vertex. It is asymmetrically linked to the larger strong component. In general, the ties among strong components are either asymmetrical or absent. In Chapter 10, we elaborate on this feature.

A (weak) *component* is a maximal (weakly) connected subnetwork.

A *strong component* is a maximal strongly connected subnetwork.

In an undirected network, lines have no direction; so each semiwalk is also a walk, and each semipath is also a path. As a consequence, there is only one type of connectedness, which is equivalent to weak connectedness in directed networks, and one type of component. In an undirected network, components are isolated from one another; there are no lines between vertices of different components. This is similar to weak components in directed networks.

In a directed network, should you look for strong or weak components? The choice depends on substantive and practical considerations. Substantive reasons pertain to the importance you attach to the direction of a relation: Does it matter to social processes whether actor A turns to

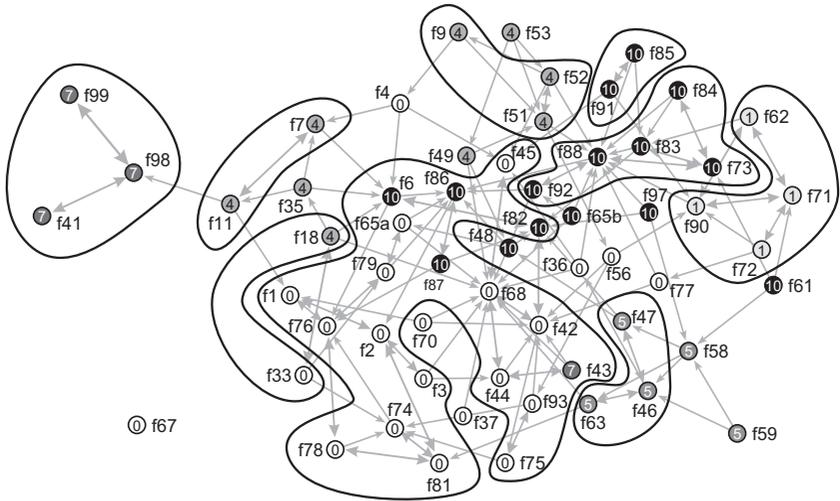


Figure 33. Strong components (contours) and family–friendship groupings (vertex colors and numbers) in the network of Attiro.

actor B, actor B turns to actor A, or both? If the flow of communication is being investigated, it probably does not matter who initiates a contact. If family f98 visits both families f11 and f99 (Figure 33, left), it may inform family f11 about family f99 and the other way around. Families f11 and f99 may share information although there is no path between them. In this case, direction of the relation is quite unimportant and weak components are preferred.

If substantive arguments are indecisive, the number and size of components may be used to choose between strong and weak components. Recall that strong components are stricter than weak components, which means that strong components usually are smaller than weak components. It is a good strategy to detect weak components first. If a network is dominated by one large weak component (e.g., the network in Attiro), we advise using strong components to break down the weak component in a next step.

Figure 33 shows the strong components in the visiting relation network. Each strong component of more than one vertex is manually delineated by a contour. Each vertex outside the contours is a strong component on its own (e.g., families f67 and f59). The original classification according to family–friendship groupings is represented by vertex colors and by the numbers inside the vertices. We see that the large weak component is split up into several small strong components, some of which approximate family–friendship groupings, for instance, family–friendship groupings 1 (at the right) and 7 (at the left).

Components can be split up further into denser parts by considering the number of distinct, that is noncrossing, paths or semipaths that connect the vertices. Within a weak component, one semipath between each pair of vertices suffices, but there must be at least two different semipaths in a *bi-component*. The concept of a bi-component is discussed in Chapter 7. This can be generalized to k -connected components: maximal subnetworks in which each pair of vertices is connected by at least k distinct semipaths or paths. A weak component, for instance, is a 1-connected component and a bi-component is a 2-connected component.

Application

With Pajek, it is easy to find components in the visiting relation network (*Attiro.net*). The *Network* menu has a submenu to find three types of components: strong, weak, and strong-periodic. Strong-periodic components will not be discussed here. When you execute commands *Strong* or *Weak*, a dialog box appears asking for the minimum size of components. Sometimes, very small components are not interesting; for instance, isolated vertices, which are counted as separate components if minimum component size is set to 1 vertex. Raise this number to exclude them. The command creates a partition in which each class represents a component. Draw the network with the strong components partition (*Draw> Network + First Partition*) to see the clusters enclosed in contours in Figure 33. Draw it with the original family–friendship groupings partition to obtain the clusters represented by vertex color in Figure 33. Figure 33 combines these two layouts.

```
Network>
Create
Partition>
Components>
Strong
```

```
Draw>
Network +
First Partition
```

In undirected networks, it makes no difference whether you select strong or weak components because the commands yield identical results. Furthermore, weak components in a directed network are equal to components in the symmetrized network. Therefore, it is not necessary to symmetrize a directed network when you want to know its components: Just compute weak components in the directed network.

```
Network>
Create
Partition>
Components>
Weak
```

3.5 Cores

The distribution of degree reveals local concentrations of ties around individual vertices, but it does not tell us whether vertices with a high degree are clustered or scattered all over the network. In this section, we use degree to identify clusters of vertices that are tightly connected because each vertex has a particular minimum degree within the cluster. We pay no attention to the degree of one vertex but to the degree of all vertices within a cluster. These clusters are called k -cores and k indicates the minimum degree of each vertex within the core; for instance, a 2-core contains all vertices that are connected by degree 2 or more to other vertices within

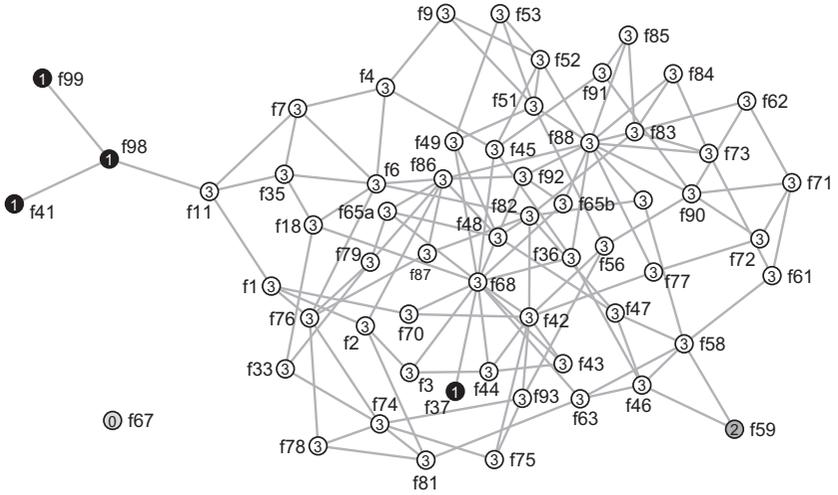


Figure 34. k -cores in the visiting network at Attiro.

the core. A k -core identifies relatively dense subnetworks, so they help to find cohesive subgroups. As is shown, however, a k -core is not necessarily a cohesive subgroup itself!

A k -core is a maximal subnetwork in which each vertex has at least degree k within the subnetwork.

The definition of a k -core is more complicated than you might think. It is easiest to explain if we apply it to a simple undirected network and, as a rule, we apply it only to this type of network. In a simple undirected network, the degree of a vertex is equal to the number of its neighbors, as discussed in Section 3.3, so a k -core contains the vertices that have at least k neighbors within the core. A 2-core, then, consists of all vertices that are connected to at least two other vertices within the core. In the definition, the word *maximal* means that we are interested in the largest set of vertices that satisfy the required property, in this case a minimum number of k neighbors within the core.

The undirected visiting relation network, which we obtained by symmetrizing the directed network, contains a large 3-core (white vertices in Figure 34). In the 3-core, each family is connected to at least three other families. In addition, there is a 2-core (dark gray), a 1-core (black), and a 0-core (light gray). Do the k -cores in the kin visits network represent cohesive subgroups? For the 3-core, this seems to be true because it is clearly a dense pocket within the network. The 2-core and the 0-core, however, consist of one vertex (families f59 and f67), and the 1-core is situated at

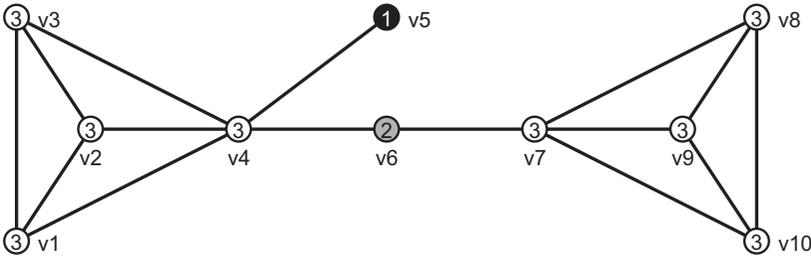


Figure 35. k -cores.

two different places in the network (at the left and at the bottom). It is silly to regard them as cohesive subgroups.

The meaning of the lower k -cores can be illustrated by the simple example in Figure 35. This little network is connected, so all ten vertices are linked to at least one other vertex. As a result, all vertices belong to the 1-core, which is drawn in black at the bottom of Figure 36. One vertex, v_5 , has only one neighbor, so it is not part of the 2-core (gray, in the middle of Figure 36). Vertex v_6 has a degree of 2, so it does not belong to the 3-core (white, in the top of Figure 36). Other vertices belong to the highest k -core, so the resulting sociogram looks like Figure 35: The different levels are stacked one on top of the other. We say that k -cores are *nested*: A vertex in a 3-core is also part of a 2-core, but not all members of a 2-core belong to a 3-core.

The example illustrates another feature of k -cores, namely, that a k -core does not have to be connected. As a result of nesting, different cohesive subgroups within a k -core are usually connected by vertices that belong to lower cores. In Figure 36, vertex v_6 , which is part of the 2-core, connects the two segments of the 3-core. If we eliminate the vertices belonging to cores below the 3-core, we obtain a network consisting of two components that identify the cohesive subgroups within the 3-core.

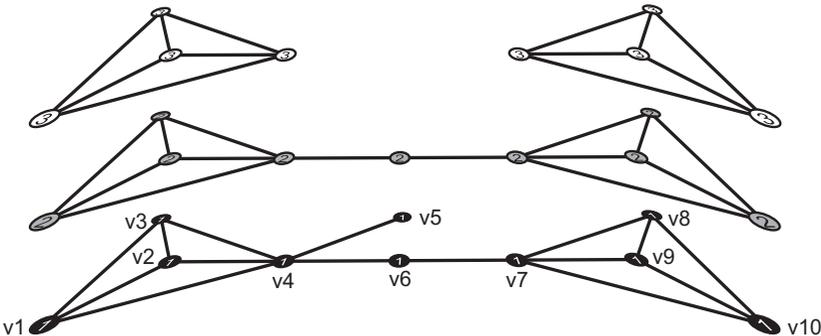


Figure 36. Stacking or nesting of k -cores.

This is exactly how k -cores help to detect cohesive subgroups: Remove the lowest k -cores from the network until the network breaks up into relatively dense components. Then, each component is considered to be a cohesive subgroup because it has at least k neighbors within the component. In (very) large networks, this is an effective way of finding cohesive subgroups. In the Attiro visiting relation network, however, this strategy does not work because there are no unconnected k -cores. Elimination of the lower k -cores does not split the network into separate components.

Application

Network>
Create
Partition>
k-Core> *Input,*
Output, All

In Pajek, k -cores are detected with the *k-Core* command in the *Network>* *Create Partition* submenu. The *Input*, *Output*, and *All* commands operate exactly in the same way as in the *Network>* *Create Partition>* *Degree* submenu, distinguishing among input cores, output cores, and cores that ignore the direction of lines. We advise using the *All* command and applying it only to simple undirected networks. The command yields a partition that assigns each vertex to the highest k -core in which it appears. Vertex colors and the numbers inside the vertices display the k -core partition in Figure 34. In this example, the k -cores do not match the ethnographic clustering into family–friendship groupings.

Operations>
Network +
Partition>
Extract>
SubNetwork
Induced by
Union of
Selected
Clusters

With the k -core partition, you can easily delete low k -cores from the network to distill the densest sections in the network. Select the k -core partition in the *Partition* drop list and execute the *Operations>* *Network + Partition>* *Extract>* *SubNetwork Induced by Union of Selected Clusters* command (see Section 2.4.1). Select the lowest and highest k -cores that you want to extract from the network, in this case the third k -core. Subsequently, use the *Network>* *Create Partition>* *Components>* *Strong* command to check whether the selected k -core levels are split into two or more components.

Network>
Create
Partition>
Components>
Strong

Exercise II

Determine the k -cores in the network `ExerciseII.net` and extract the 4-core from this network.

3.6 Cliques and Complete Subnetworks

In the visiting relation network, most vertices belong to one large 3-core. If we want to split this large 3-core into subgroups, we need a stricter definition of a cohesive subgroup. In this section, we present the strictest structural form of a cohesive subgroup, which is called a clique: a set of vertices in which each vertex is directly connected to all other vertices. In other words, a clique is a subnetwork with maximum density.

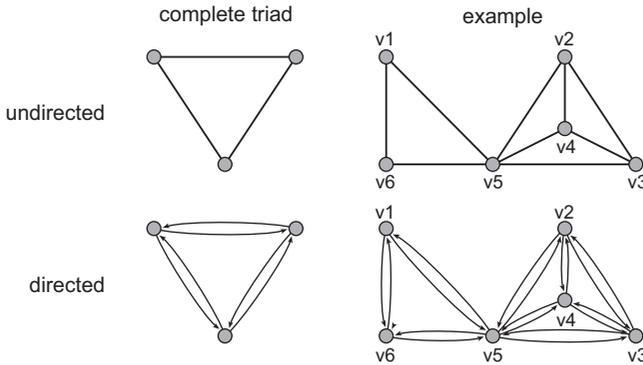


Figure 37. The complete triad and an example.

A *clique* is a maximal complete subnetwork containing three vertices or more.

The size of a clique is the number of vertices in it. Maximal complete subnetworks of size 1 and 2 exist, but they are not very interesting because they are single vertices and edges or bidirectional arcs, respectively. Therefore, cliques must contain a minimum of three vertices.

Unfortunately, it is very difficult to identify cliques in large networks: The computational method is very time consuming, and even medium-sized networks may contain an enormous number of cliques. In this book, therefore, we restrict ourselves to the analysis of small complete subnetworks, which may or may not be cliques. We concentrate on complete *triads*, that is, complete subnetworks consisting of three vertices; but the argument is easily extended to complete subnetworks of size 4 or more.

Figure 37 shows the complete undirected and directed triad as well as an example of a network that contains several complete triads. Note that the complete triad with vertices v_1 , v_5 , and v_6 is a clique because we cannot add another vertex from the network to this subnetwork such that it is still complete. This subnetwork is maximal with respect to completeness. In contrast, triad v_2 , v_4 , v_5 is not a clique because we can add vertex v_3 and the subnetwork is still complete. Vertices v_2 to v_5 constitute a clique of size 4, which, by the way, is made up of four complete triads.

Figure 37 shows a very important feature of cliques and complete subnetworks, namely, that they can overlap. The complete triad v_1 , v_5 , v_6 overlaps with the complete triad v_2 , v_4 , v_5 because they share vertex v_5 . As a consequence, it is impossible to assign all vertices unambiguously

to one clique or complete subnetwork. We cannot equate each clique or complete subnetwork with a cohesive subgroup, and this is a serious complication if we want to classify vertices into cohesive subgroups.

In social network analysis, structures of *overlapping cliques*, which are thought to represent social circles rather than individual cliques, are regarded as cohesive subgroups. Cliques or complete triads are the densest sections or “bones” of a network, so the structure of overlapping cliques are considered its “skeleton.” Sometimes, additional conditions are imposed on the overlap of cliques (e.g., a minimum number or percentage of vertices that two cliques must share), but we do not use them here.

Application

Network> Because clique detection is particularly useful for dense networks, we now
Create New analyze the symmetrized (undirected) network of visiting ties in Attiro,
Network> which has higher density (0.072) than the directed network (0.045).
Transform> Symmetrize the network with the *Network> Create New Network>*
Arcs→Edges> *Transform> Arcs→Edges> All* command and avoid multiple lines by
All selecting option 1, 2, 3, 4, or 5 in the “Remove multiple lines?” dialog
 box. This network is too dense to spot complete triads and structures
 of overlapping triads visually. Even the best energized drawing contains
 many crossing edges, which makes it difficult to see complete triads; there
 are probably many.

Networks The first step, then, is to detect all complete triads within the network.
 In other words, we have to find all occurrences of one particular network
 or fragment – in our case, a complete triad – in another network, namely,
 the original network. The command is situated in the *Networks* menu,
 which contains all operations on two networks, and it requires that the
 fragment and the original network are identified as the *First Network*
 and *Second Network*, respectively. The project file *Attiro.paj* contains
 the network *triad_undir.net*, which is a single complete undirected
 triad. Select this network in the first *Network* drop-down menu menu,
 and select the symmetrized visiting ties network in the second *Network*
 drop-down menu.

Networks> Next, we can find all complete triads in the network by executing the
Fragment (First *Find* command of the *Fragment (First in Second)* submenu. Executing this
in Second) command, Pajek reports the number of fragments it has found, and it cre-
 ates one or more new data objects, depending on the options selected in
 the *Options* window of the *Fragment* command. We recommend having
 the options *Extract subnetwork* and *Same vertices determine one frag-*
ment at most checked only. The latter option ensures that unique instances
 of the fragment are counted, for example, that three vertices are counted
 only once as a complete triad. For large networks, this check may take a
 lot of time so the option can be deselected as well as its suboption *Create*

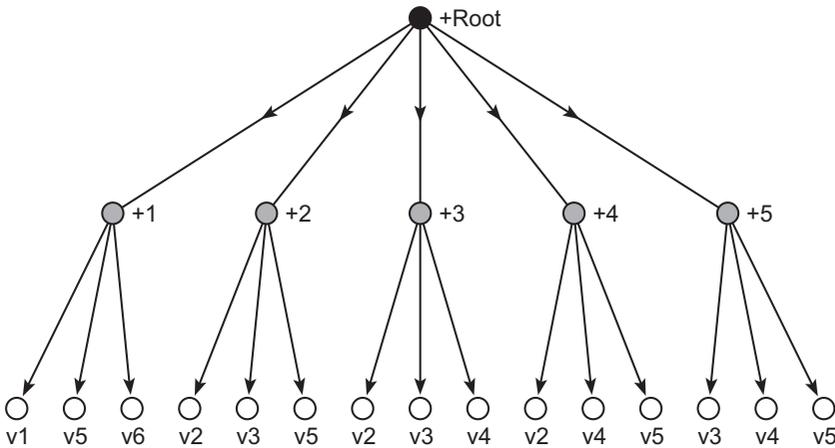


Figure 38. A hierarchy of cliques.

Hierarchy with Fragments. The search is quicker but there are multiple counts of the same fragment, depending on the structure of the fragment. For example, each undirected complete triad is counted six times, so one should divide the counts by this number. Note that the option *Same vertices determine one fragment at most* should usually not be set in searches of directed fragments because there can be different fragments for the same vertices, for example, several transitive triples in a complete directed triad.

This produces a new network labeled “Subnetwork induced by Sub fragments.” It is called *induced* because Pajek selects vertices and lines within the fragments (complete triads) only. This network contains the overlapping cliques that we are looking for, and we discuss it at the end of this section. In addition, Pajek creates a hierarchy and a partition. The partition counts the number of fragments to which each vertex belongs, and the hierarchy lists all fragments: complete triads in our example.

A *hierarchy* is a data object that we have not yet encountered. It is designed to classify vertices if a vertex may belong to several classes. In the visiting relation network, for instance, a family may belong to several complete triads. A hierarchy is a list of groups, and each group may consist of groups or vertices. Ultimately, vertices are the units that are grouped. Figure 38 shows the hierarchy for the example of overlapping complete triads of Figure 37. There are five complete triads; each of them is represented by a gray vertex in Figure 38. Each complete triad consists of three vertices (white in Figure 38). Note that most vertices appear more than once because the triads overlap. At the top of the hierarchy, one node (black) connects all groups; it is called the root of the hierarchy.

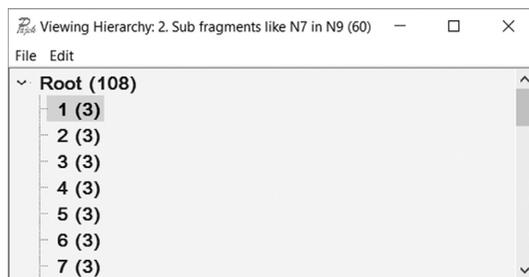


Figure 39. Viewing a hierarchy in an Edit screen.

Hierarchy
drop-down
menu

File>
Hierarchy>
View/Edit

A hierarchy is stored as a data object in the *Hierarchy* drop-down menu. You can browse a hierarchy in an Edit screen, which is opened with the *View/Edit* command in the *File*> *Hierarchy* submenu or by the *View/Edit* button to the left of the *Hierarchy* drop-down menu. On opening, the Edit screen displays the root only. Click on the plus sign preceding the root to display the (first level of) groups in the hierarchy.

Figure 39 shows part of the thirty-six complete triads in the visiting relation network of Attiro. Select a group with your left mouse button and click with the right mouse button to display its vertices in a separate window. If the original network is selected in the *Network* drop-down menu, vertex labels are displayed next to their numbers in this window. In this way, you can see which vertices belong to a complete triad.

Partition> *Info*

Now let us turn to the induced network and the partition created by the *Networks*> *Fragment (First in Second)*> *Find* command. The partition, labeled “Sub fragments,” shows the number of triads that include a particular vertex. Using the *Partition*> *Info* command in the Main screen, you can see that two vertices belong to no fewer than seven complete triads, whereas thirteen vertices are not included in any of the complete triads. The latter vertices are not part of the structure of overlapping cliques, so they are eliminated from the induced network (labeled “Subnetwork induced by Sub fragments”), which contains the remaining forty-seven vertices of the Attiro network.

Partitions >
Extract
SubPartition
(*Second from*
First)

With this partition, we can make the original partition according to family–friendship groupings (in `Attiro_grouping.clu`) match the new induced network. Select the original partition as the first partition, and select the fragments partition as the second partition. Then execute the *Partitions*> *Extract SubPartition (Second from First)* command and specify 1 as the lowest class number and 7 (or higher) as the highest class number to be extracted. Pajek creates a new partition holding the family–friendship groupings of the forty-seven vertices in the induced network of overlapping complete triads. Draw this network and partition, and energize it with Kamada–Kawai to obtain a sociogram such as Figure 40 (use

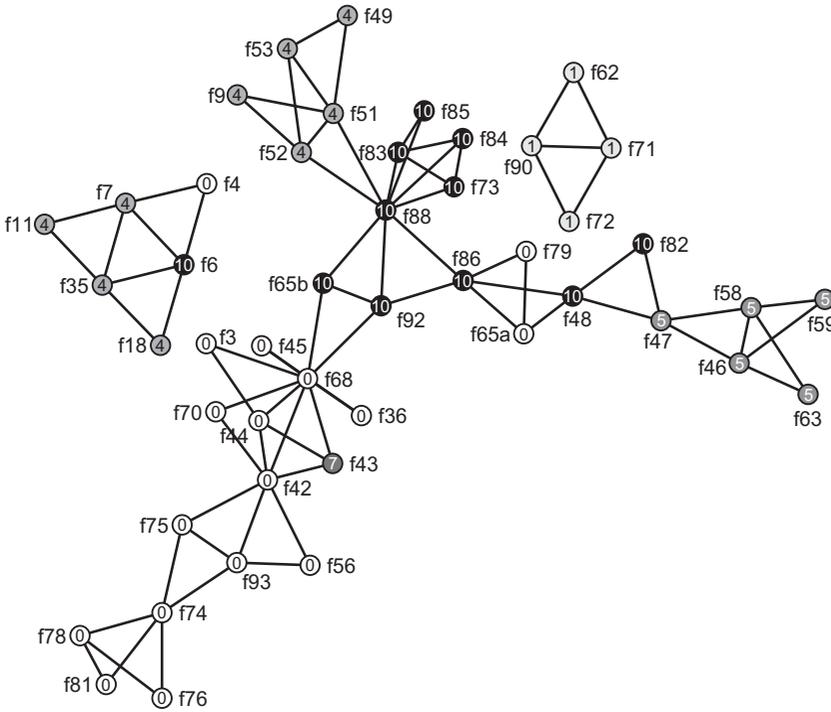


Figure 40. Complete triads and family–friendship groupings (colors and numbers inside vertices).

Default GreyScale 2 in the *[Draw] Options > Colors > Partition Colors > for Vertices* dialog screen to get the grays).

The induced subnetwork is displayed in Figure 40. It has three components of overlapping complete triads, so we say that we have found three social circles under the criterion of complete triads that share at least one member. Family–friendship grouping 1 is a separate social circle, but the other family–friendship groupings are interconnected although they are clearly clustered within the largest component. Family–friendship grouping 10 occupies a pivotal position in this structure, connecting groupings 0, 5, and part of family–friendship grouping 4.

In a directed network, you may follow the same procedure, but you have to use a complete directed triad as a fragment (e.g., `triad_dir.net`). In general, you will find fewer cliques in the directed network than undirected cliques in the symmetrized network. In the directed Attiro network, for instance, there is just one complete directed triad, containing families f62, f71, and f90, so we cannot speak of overlapping cliques in the directed network.

3.7 Summary

In this chapter, social cohesion was linked to the structural concepts of density and connectedness. Density refers to the number of links between vertices. A network is strongly connected if it contains paths between all of its vertices, and it is weakly connected when all of its vertices are connected by semipaths. Connected networks and networks with high average degree are thought to be more cohesive. This also applies to sections of a network (subnetworks). We expect local concentrations of ties in a social network to identify cohesive social groups.

There are several techniques to detect cohesive subgroups based on density and connectedness, three of which are presented in this chapter: components, k -cores, and cliques or complete subnetworks. All three techniques assume relatively dense patterns of connections within subgroups, but they differ in the minimal density required, which varies from at least one connection (weak components) to all possible connections (cliques). Two more techniques based on a similar principle (islands and bi-components) are presented in later chapters. There are many more formal concepts for cohesive subgroups, but all of them are based on the notions of density and connectedness.

Components identify cohesive subgroups in a straightforward manner: Each vertex belongs to exactly one component. The link between cohesive subgroups and k -cores or cliques is more complicated. k -cores are nested, which means that higher k -cores are always contained in lower k -cores, so a vertex may belong to several k -cores simultaneously. In addition, k -cores are not necessarily connected: The vertices within one k -core can be spread over several components. To identify cohesive subgroups, the researcher has to eliminate vertices of low k -cores until the network breaks up into relatively dense components. Cliques or complete subnetworks, such as complete triads, may overlap, that is, share one or more vertices, so a component of overlapping cliques is regarded as a cohesive subgroup rather than each clique on its own.

Because the techniques to detect cohesive subgroups are based on the same principle, substantive arguments to prefer one technique over another are usually not available. The choice of a technique depends primarily on the density of the network. In a dense network, the structure of overlapping cliques reveals the cohesive skeleton best, whereas components and k -cores unravel loosely knit networks better. In exploratory research, we recommend looking for components first and then applying k -cores and searching for complete triads to subdivide large k -cores if necessary (see the decision tree in Figure 41).

Another choice pertains to the treatment of directed relations. In general, symmetrizing directed relations yields higher density, thus more

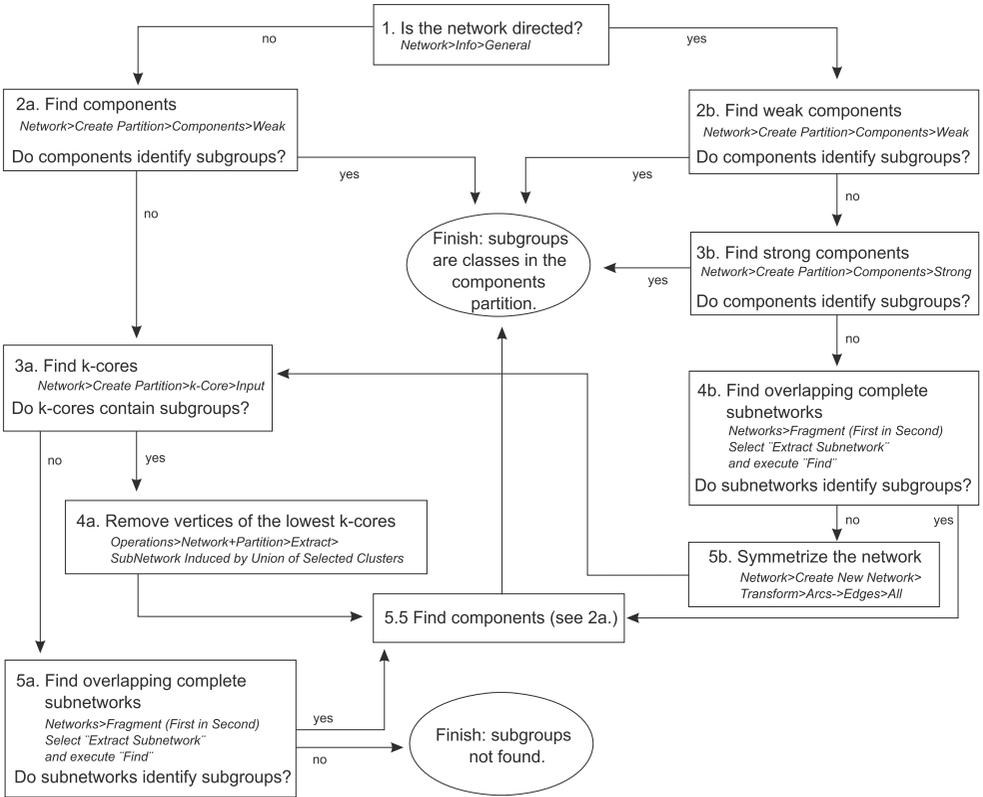


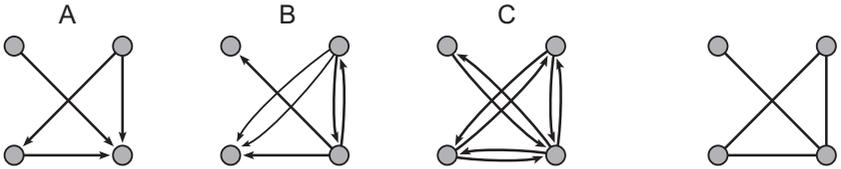
Figure 41. Decision tree for the analysis of cohesive subgroups.

or larger cohesive subgroups. For k -cores, we recommend using simple undirected or symmetrized networks to make sure that k equals the number of neighbors to which each vertex is connected in a core. In a directed network, components may be weak or strong. Strong components and complete directed triads are based on reciprocal ties, whereas weak subgroups consider unilateral ties as well.

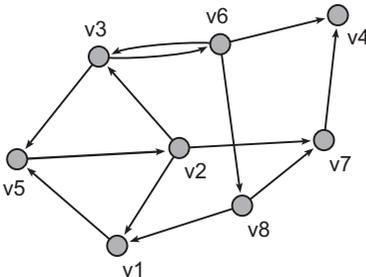
In this chapter, we use the word *subgroup*, but a cohesive subgroup is not necessarily a social group. We need to check this by comparing the structural subgroups with respect to the social characteristics, behavior, and opinions of their members. Sometimes, our prior knowledge about the entities in the network enables us to make sense of the cohesive subgroups we detect. Otherwise, we must systematically compare the partition that identifies cohesive subgroups with partitions representing social attributes of the vertices.

3.8 Questions

1. Inspect the networks that are depicted in the following figure. If we symmetrize the directed networks, which ones become identical to the undirected network?

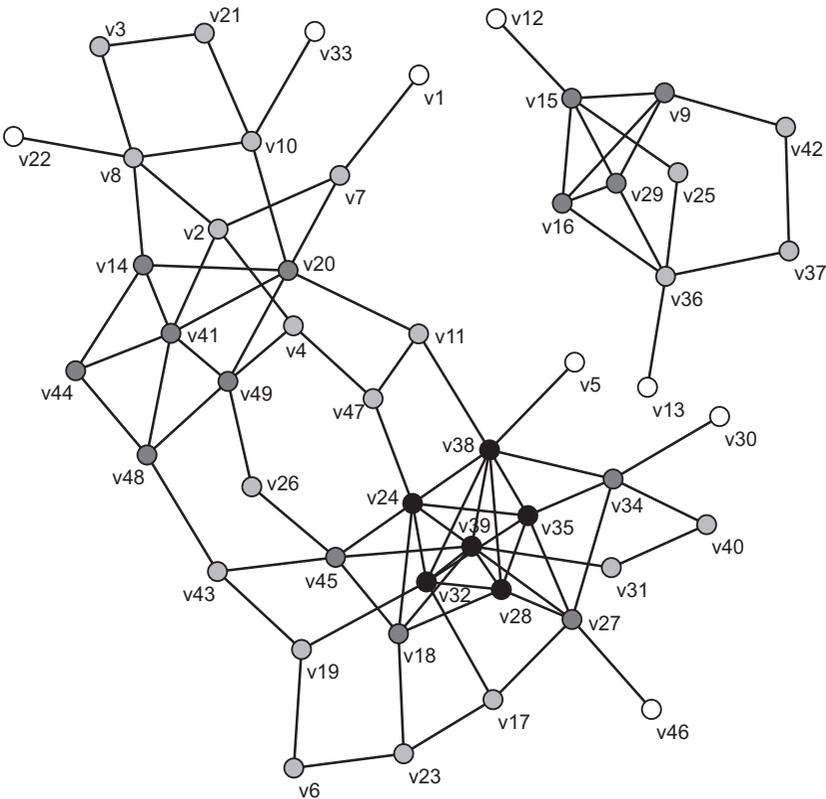


- A, B, and C
 - A and B only
 - A and C only
 - A only
2. Which of the following statements is correct?
- Density is the degree sum of all vertices in a simple undirected network.
 - Density is the degree sum of all vertices divided by the number of vertices in a simple undirected network.
 - Density is the number of arcs divided by the number of pairs of vertices in a simple directed graph.
 - Density is the number of edges divided by the number of pairs of vertices in a simple undirected graph.
3. Which of the following statements about the network below are correct?



- v_6 and v_4 are not connected by a semipath.
 - v_1 and v_4 are not connected by a path.
 - There is no path from v_7 to v_3 .
 - There is no path from v_1 to v_8 .
4. How many strong components does the network of Question 3 contain?
- No strong components
 - One strong component

- c. Two strong components
 - d. Three strong components
5. Which of the following statements is correct?
- a. A subnetwork is maximal if it cannot be enlarged without losing its structural characteristic.
 - b. A subnetwork is not maximal if it does not cover the whole network.
 - c. A subnetwork is maximal if it is connected.
 - d. A subnetwork is maximal if it is complete.
6. Which of the following statements is correct for simple undirected networks?
- a. In a 1-core, each vertex has exactly one neighbor.
 - b. A component containing two or more vertices is always a 1-core.
 - c. Each 1-core is a clique.
 - d. Each 3-core is a clique.
7. Count the number of 3-cores in the network in the following figure. Vertex colors indicate the level of k (white: $k = 1$; light gray: $k = 2$; dark gray: $k = 3$; black: $k = 4$).



- a. One 3-core
- b. Two 3-cores

- c. Three 3-cores
- d. Four 3-cores

3.9 Assignment

The researchers assigned the families of another village in the Turrialba region, San Juan Sur, to family–friendship groupings on the basis of their answers to the question: In case of a death in the family, whom would you notify first? Their choices are stored in the file `SanJuanSur_deathmessage.net`. In this file, the coordinates of families correspond with the locations of families in the original sociogram drawn by the researchers. The partition `SanJuanSur_deathmessage.clu` contains the family–friendship groupings for this network.

We would like to reconstruct the way the families were assigned to family–friendship groupings. Find out which type of cohesive subgroups match the family–friendship groupings best, and use the indices of statistical association presented in Chapter 2, Section 2.6, to assess how well they match. Do you think that the researchers used additional information to assign families to family–friendship groupings?

3.10 Further Reading

- The example is taken from Charles P. Loomis, Julio O. Morales, Roy A. Clifford, and Olen E. Leonard, *Turrialba: Social Systems and the Introduction of Change* (Glencoe, IL: The Free Press, 1953). We will also use these data in Chapter 9 on prestige.
- Chapter 7 in John Scott, *Social Network Analysis: A Handbook* (London: SAGE [4th edn. 2017], 1991) offers an overview with some additional types of connected subnetworks. Chapter 7 in Stanley Wasserman and Katherine Faust, *Social Network Analysis: Methods and Applications* (Cambridge: Cambridge University Press, 1994) is even more detailed.
- S. B. Seidman’s “Network structure and minimum degree” (*Social Networks* [1983], 269–87) introduced k -cores. R. D. Luce and A. Perry formally defined cliques in “A method of matrix analysis of group structure” (*Psychometrika* 14 [1949], 95–116).

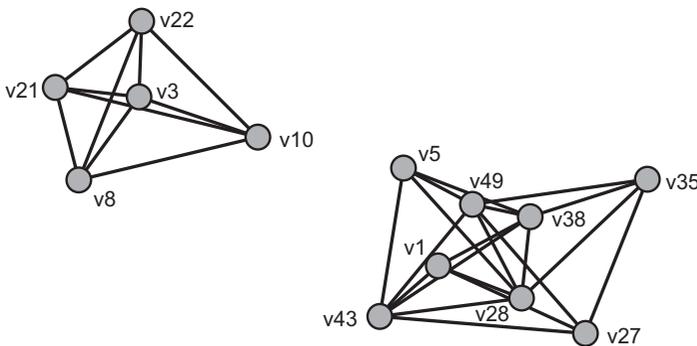
3.11 Answers

Answers to the Exercises

- I. In the symmetrized network of San Juan Sur without multiple lines, vertex degree ranges from 1 to 12 and the average degree is 4.13

(change the degree partition into a vector and inspect the vector with the *Vector> Info* command).

- II. Open the network `ExerciseII.net` and determine the k -core partition with one of the *Network> Create Partition> k-Core* commands. Inspecting the core partition with the *Partition > Info* command, you will see that 13 vertices belong to the 4-core, 33 (13 + 20) to the 3-core, 47 (33 + 14) vertices to the 2-core, and all 49 vertices (47 + 2) to the 1-core. You can extract the 4-core from the network with the *Operations> Network + Partition> Extract> SubNetwork Induced by Union of Selected Clusters* command (just extract class 4). The extracted 4-core is not connected, as shown in the following figure; it consists of two parts.



Answers to the Questions in Section 3.8

- Answer c is correct. Symmetrizing a network means that unilateral and bidirectional arcs are replaced by an edge. Multiple arcs (e.g., from the top-right vertex to the bottom-left vertex in network B) are replaced by multiple edges. Therefore, the undirected network is not a symmetrized version of network B.
- Answer d is correct. Density is defined as the number of lines in a network, expressed as a proportion of the maximum possible number of lines. In a simple undirected network, a pair of vertices can be connected by one edge because multiple lines do not occur. In addition, loops do not occur. Therefore, the number of pairs of vertices equals the maximum possible number of lines, and answer d is correct. Recall that an edge is defined as an unordered pair of vertices and an arc as an ordered pair. In a simple directed network, each pair of vertices may be connected by two arcs, so the maximum possible number of lines is twice the number of pairs. Moreover, a simple directed network may also contain loops. Thus, answer c is incorrect. Answers a and b refer to the sum of degrees and average degree, which are other kinds of indices.

3. Answer c is correct because the only path that originates in v_7 leads to v_4 , where it stops because v_4 sends no arcs. Answer a is incorrect because there are several semipaths between v_6 and v_4 (e.g., $v_6 \rightarrow v_3 \rightarrow v_2 \rightarrow v_7 \rightarrow v_4$) and each path ($v_6 \rightarrow v_4$) is also a semipath. Answer b is incorrect because there is a path from v_1 to v_4 (e.g., $v_1 \rightarrow v_5 \rightarrow v_2 \rightarrow v_7 \rightarrow v_4$). Answer d is incorrect because there is a path from v_1 to v_8 as follows: $v_1 \rightarrow v_5 \rightarrow v_2 \rightarrow v_3 \rightarrow v_6 \rightarrow v_8$.
4. Answer d is correct. Vertex v_4 is a strong component on its own because it has no path to any other vertex. Vertex v_7 is the start of one path, which ends at v_4 . Because there is no path in the opposite direction, v_7 is not a member of a larger strong component. The third strong component consists of the remaining six vertices, which are connected by paths in both ways.
5. Statement a is correct; for example, a component is a maximal connected subnetwork because no vertex can be added in such a way that the subnetwork is still connected. Connectedness in itself is not enough for a subnetwork to be maximal (statement c), nor is completeness (statement d). A maximal subnetwork does not need to include all vertices in the network (statement b is incorrect).
6. Answer b is correct for every simple undirected network. In a component of size 2 or more, vertices must be linked to at least one other vertex for the component to be connected, so each component is at least a 1-core. A star network is a 1-core because all vertices except the central vertex have just one neighbor. Nevertheless, the central vertex has more than one neighbor, so answer a is incorrect. k -cores are not necessarily cliques, so answers c and d are not correct.
7. Answer a is correct. A k -core is not necessarily connected, so all unconnected parts of the 3-core still belong to one 3-core.

Sentiments and Friendship

4.1 Introduction

In the preceding chapter, we discussed several techniques for finding cohesive subgroups within a social network. People who belong together tend to interact more frequently than people who do not. In the current chapter, we extend this idea to affective relations that are either positive or negative, for instance, friendship versus hostility, liking versus disliking. We expect positive ties to occur within subgroups and negative ties between subgroups.

Hypotheses about patterns of affective relations stem from social psychology, and they are widely known as balance theory. First, we introduce this theory and discuss how it was incorporated in network analysis. Then, we apply it to affective relations, that is, social relations that are subjective and mental rather than tangible.

4.2 Balance Theory

Social psychology is interested in group processes and their impact on individual behavior and perceptions. In the 1940s, Fritz Heider formulated a principle that has become the core of balance theory, namely, that a person feels uncomfortable when he or she disagrees with his or her friend on a topic. Figure 42 illustrates this situation: P is a person, O is another person (the Other), and X represents a topic or object. P likes O, which is indicated by a positive line between P and O; however, they disagree on topic X because P is in favor of it (positive line), whereas O is opposed to it (negative line). Note the convention of drawing negative ties as dashed lines, which is also adopted in Pajek.

Heider predicted that P would become stressed and feel an urge to change the imbalance of the situation, either by adjusting his opinion on

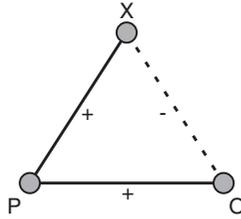


Figure 42. A Person–Other–Object (X) triple.

X, by changing his affections for O, or by convincing himself or herself that O is not really opposed to X. Research in small groups corroborated the hypothesis that people feel stressed in a situation of imbalance.

A special kind of network may represent structures of affective ties, namely a signed graph. In a signed graph, a positive or negative sign is attached to each line indicating whether the associated tie (e.g., an affection) is positive or negative.

A *signed graph* is a graph in which each line carries either a positive or a negative sign.

In a signed graph, the Person–Other–Object triple is represented by a cycle, that is, a path in which the first and last vertices coincide. All balanced cycles contain an even number of negative lines or no negative lines at all; for instance, there is one negative line in the cycle of Figure 42, which is an uneven number, so this triple is not balanced. P, and possibly O, will feel stressed in this situation.

However, affective relations do not need to be symmetrical. My feelings for you may differ from your feelings toward me. Affections are projected from a person to something or someone else. Therefore, it is usually better to represent affect ties by arcs rather than edges. It is easy to generalize balance theory to signed directed graphs: Ignore the direction of arcs, and count the number of negative arcs in each semicycle (a closed semipath). In Figure 43, the sequence of arcs from P to X, on to O, and back to

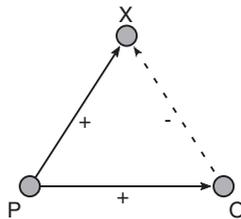


Figure 43. P–O–X triple as a signed digraph.

P constitute a semipath and a semicycle but not a path and a cycle, because not all arcs point toward the next vertex within this sequence. The semicycle is unbalanced because it contains an uneven number of negative arcs.

A *cycle* is a closed path.

A *semicycle* is a closed semipath.

A (semi-)cycle is *balanced* if it does not contain an uneven number of negative arcs.

Fritz Heider was concerned with the feelings and perceptions of one person. Therefore, Figure 43 contains affections from person P to the other (O) and to the object or topic X. Even O's tie to X is measured from the perspective of P: It is P's idea about what O thinks of X, which does not necessarily correspond to O's real opinion. In social psychology, this phenomenon is called *attribution*. Of course, O may have positive or negative affect for P as well, and if X is a human being (or an animal) rather than a topic, X may also express affections for P and O.

Network analysts are interested in the feelings of all members of a group toward each other. This has led to the notion of *structural balance*, which expects balance in the overall pattern of affect ties within a human group rather than in one person's affections and attributions.

There are exact conditions for a signed graph to be balanced. A balanced signed graph can be partitioned into two clusters such that all positive arcs are contained within the clusters and all negative arcs are located between clusters. You might say that a balanced network is extremely polarized because it consists of two factions and actors have positive ties only with members of their own faction, whereas they have negative ties with members of the other faction. Clusters group people who like each other but who dislike members of the other cluster. It is easy to check this in Figure 44, which uses gray and black to identify the clusters.

In addition, it was proven that a signed graph is balanced if and only if all of its semicycles are balanced. Find one unbalanced semicycle, and you know that the network is unbalanced.

A signed graph is *balanced* if all of its (semi-)cycles are balanced.

A signed graph is *balanced* if it can be partitioned into two clusters such that all positive ties are contained within the clusters and all negative ties are situated between the clusters.

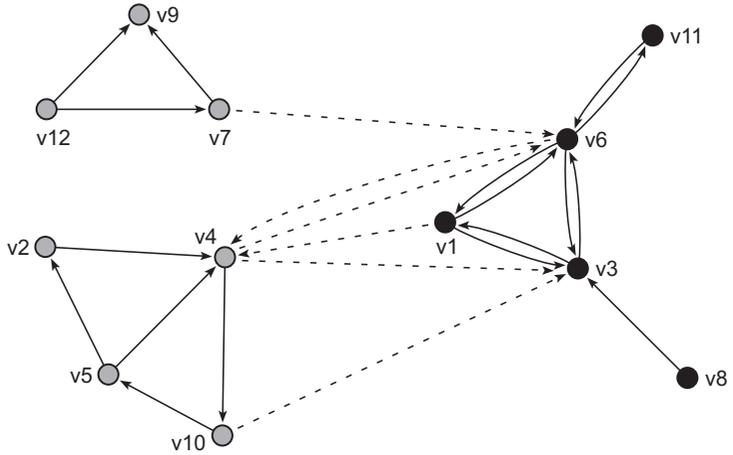


Figure 44. A balanced network.

Why would human groups consist of two clusters or factions instead of three or more? In Figure 44, for instance, vertices v_7 , v_9 , and v_{12} could very well be a cluster on their own. To allow for three or more clusters, balance was generalized to *clusterability*. A signed network is clusterable if there is a partition satisfying the criterion that positive lines connect vertices within a cluster and negative lines are incident with vertices in different clusters, no matter the number of clusters. It was shown that a network is clusterable if it contains no semicycles with exactly one negative arc. Clearly, balance is a special case of clusterability because all balanced semicycles are clusterable.

A cycle or semicycle is *clusterable* if it does not contain exactly one negative arc.

A signed graph is *clusterable* if it can be partitioned into clusters such that all positive ties are contained within clusters and all negative ties are situated between clusters.

In the course of time, balance theory has been generalized to models that incorporate hierarchy. We present these models in Chapter 10. Some of them apply to unsigned networks, but we analyze signed relations only in the current chapter. To find subgroups in unsigned networks, we advise using the techniques for tracing cohesive subgroups, which are presented in Chapter 3.

4.3 Example

In this chapter, we use a case that has been reanalyzed by network analysts many times, namely the ethnographic study of community structure in a New England monastery by Samuel F. Sampson. The study describes several social relations among a group of men (novices) who were preparing to join a monastic order. We use the affect relations among the novices, which were collected by asking them to indicate whom they liked most and whom they liked least. The novices were asked for a first, second, and third choice on both questions.

The social relations were measured for several moments in time. The file `Sampson.net` contains the affect relations at five different moments. The first choice of the most liked peer is coded with line value 3, the second choice with line value 2, and the third choice with line value 1. Least liked choices are coded with negative line values as follows: -3 for the most disliked colleague, -2 for the second choice, and -1 for the third choice. In the present section, however, we focus on the affective ties between the novices at the fourth moment in time (T4), which was one week before four of them were expelled from the monastery. For the sake of illustration, we use their first choices only, which are recoded to 1 for *most liked* and -1 for *least liked*. The data are available in the file `Sampson_T4.net`. The Pajek project file `Sampson.paj` contains all networks and partitions.

Some novices had attended the minor seminary of “Cloisterville” before they came to the monastery; they are identified as class 1 in the partition `Sampson_cloisterville_T4.clu`. Based on his observations and analyses, Sampson divided the novices into four groups, which are represented by classes in the partition `Sampson_factions_T4.clu`: Young Turks (class 1), Loyal Opposition (class 2), Outcasts (class 3), and an interstitial group (class 4). The Loyal Opposition consists of the novices who entered the monastery first. The Young Turks arrived later, during a period of change. They questioned practices in the monastery, which the members of the Loyal Opposition defended. Some novices did not take sides in this debate, so they are labeled “interstitial.” The Outcasts are novices who were not accepted into the group.

4.4 Detecting Structural Balance and Clusterability

Social networks are seldom perfectly balanced or clusterable. In some applications, researchers want to know whether a social network is more balanced or clusterable than we may expect by chance (see Chapter 13). If so, they conclude that the actors in the network adjust their ties to

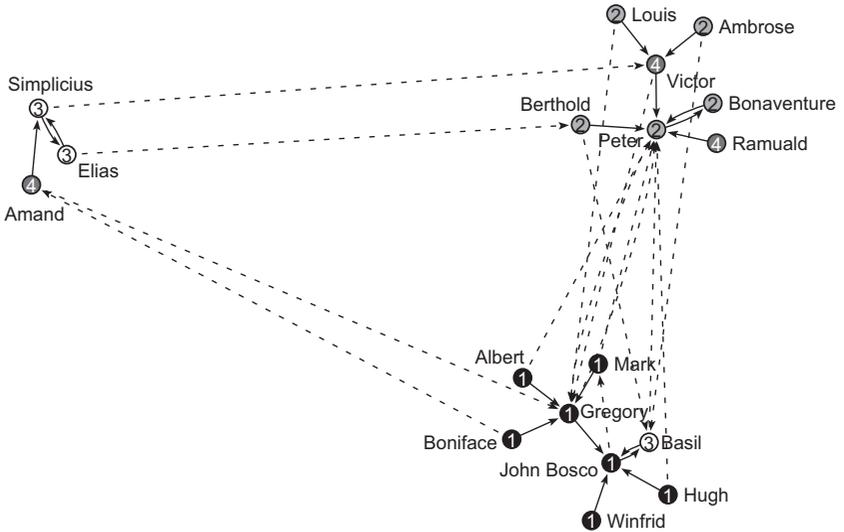


Figure 45. First positive and negative choices between novices at T4.

balance. In exploratory social network analysis, however, we are primarily interested in detecting balanced clusters, which represent cohesive subgroups within the network.

There are several ways to detect clusters in a signed network such that positive lines are within clusters and negative lines between clusters. Sometimes, clusters can be found by visual exploration. If we draw positive lines, which indicate attraction, as short as possible and negative lines, which signal repulsion, as long as possible, clusters of positive ties are clearly visible in a sociogram. In Figure 45, which is drawn in this manner (as is explained under the section “Application”), we can see three clusters in the network of novices. Because there are three clusters, the network is clusterable rather than balanced.

Because the network is highly clusterable and not very dense, we can visually check that all positive arcs are situated within clusters and almost all negative arcs are directed from one cluster to another. The only negative arc within a cluster points from John Bosco to Mark at the bottom of the sociogram. Note that the triple John Bosco, Mark, and Gregory contains exactly one negative arc, so it is unclusterable and will yield problems in any clustering we may attempt.

In Figure 45, vertex colors and class numbers indicate the factions that Sampson delineated: Young Turks (black, class 1), Loyal Opposition (light gray, class 2), Outcasts (white, class 3), and the interstitial group (dark gray, class 4). The social cleavage between the Young Turks (black) and the Loyal Opposition (light gray) is evident, but the Outcasts are not

clustered perfectly. Ramuald and Victor are clustered with the Loyal Opposition, and they probably felt somewhat related to them because they all (except Louis) came from Cloisterville.

If sociograms are not as orderly as Figure 45, we must use computational techniques to find the clustering that fits balance or clusterability best. In exploratory network analysis, a good strategy is to try many clusterings and select the one containing the lowest number of forbidden lines: positive lines between clusters or negative lines within a cluster. The number of forbidden lines (inconsistencies) is an error score that measures the degree of balance or clusterability in a network: More errors mean less balance or clusterability.

In Figure 45, there is just one forbidden line if we partition the novices into three clusters, namely, the negative arc from John Bosco to Mark in the bottom right cluster. It is up to the researcher to decide whether the degree of balance or clusterability is acceptable. Criteria cannot be specified without the use of estimation techniques, which fall outside the scope of the present chapter, because the acceptability of an error score depends on the size and density of a network. The error score allows us to pick the best fitting clustering, but it does not say whether it is good enough.

The approach of rearranging vertices into clusters over and over again and selecting the best solution is an *optimization technique* that has three features that are worth noting. First, an optimization technique may find several solutions or partitions that fit equally well. It is up to the researcher to select one or present them all.

Second, it is possible that this technique does not find the best fitting clustering, although this is expected to happen only in exceptional cases. Nevertheless, there is no guarantee that there is not a better solution, unless, of course, you find a clustering that fits perfectly. We advise repeating the procedure many times and inspecting the results visually to see whether you can find a better solution.

Third, starting options may yield different results; for instance, the procedure finds another solution if it is told to look for two clusters instead of three or four. It is usually possible to estimate the approximate number of clusters from an energized sociogram, but it is hard to tell the exact number of clusters that will yield the lowest error score. Therefore, it is important to repeat the optimization technique with different numbers of clusters.

In addition, the user may attribute different weights or penalties to forbidden positive and negative arcs. For instance, researchers have noted that negative arcs within a cluster are tolerated less than positive arcs between clusters, so we can raise the penalty on negative arcs within clusters. In the network of affect relations between novices depicted in Figure 45, this would mean that John Bosco's negative feelings for Mark are more

important than Gregory's positive affection for John Bosco. Hence, the optimization technique will split the bottom cluster between John Bosco and Gregory. Different weights may produce different results.

Application

[Draw]
Options>
Values of
Lines>
Similarities

A sociogram that minimizes the length of positive lines and maximizes the length of negative lines can be made in two steps. First, select the option *Similarities* in the *Options> Values of Lines* submenu of the Draw screen. This option tells the energy procedures that line values indicate similarity or attraction: The higher a line value, the closer two vertices should be drawn. Negative line values mean that vertices are dissimilar and must be drawn far apart. In Pajek, signs of lines are represented by the sign of the line values (e.g., 1 and -1), so positive arcs are short and negative arcs are long in an energized drawing. Note that this option remains effective until another option is selected. Second, apply an energy procedure to the sociogram. Figure 45 was created with the Kamada–Kawai energy command.

The command *Doreian–Mrvar Method**, which searches for an optimal clustering in a signed network, is located in the *Network> Signed Network* menu because it can be applied to signed networks only. The network contains the vertices and ties that must be clustered. In addition, we need a partition that specifies the number of clusters and the initial clustering that the computer tries to improve.

Partition>
Create Random
Partition>
1-Mode

If you have no partition with a meaningful initial clustering, you can easily make a random partition with the *Create Random Partition> 1-Mode* command in the *Partition* menu. This command issues a dialog box with two questions. The first question asks for the number of vertices or *dimension* of the partition. By default, Pajek shows the number of vertices in the network that is currently active, which is the right number because you want the partition to fit this network. In the second field, you enter the number of clusters you want to detect in the network. In this example, you may want to obtain three clusters.

Network>
Signed
Network>
Create
Partition>
Doreian–Mrvar
Method*

The *Doreian–Mrvar Method** command opens a new window for specifying the number of repetitions, the error weight, and the minimum number of vertices in a cluster (click to change).

Be sure that the *Relaxed balance* checkbox remains unchecked. When we select relaxed balance we treat some situations, which are violations of structural balance, as belonging to other relevant processes (e.g.; mediation, differential popularity, and internal subgroup hostility; see Further Reading.) Relaxing ideas of structural balance allows us to partition signed two-mode networks. We will introduce (unsigned) two-mode networks in Chapter 5. If you are interested in partitioning signed two-mode networks check the papers referenced in the Further Reading section.

In each repetition, the *Doreian–Mrvar Method** command starts with a new random partition. If a starting partition fits quite well, the optimization technique will not find better solutions because all changes will increase the error score initially. This could happen, for instance, with a starting clustering based on visual inspection of the energized sociogram. With several random starting partitions, the procedure is unlikely to miss a good clustering that differs greatly from your expectations, although this is not guaranteed. In a small network, 100 repetitions is a reasonable first choice, but you are advised to try many more repetitions if the computer needs little time for 100 repetitions.

Next, you have to specify the error weight of a forbidden negative arc, that is, a negative arc within a positive cluster. This weight is called α and is 0.5 by default. The error weight for an erroneous positive arc is equal to $1 - \alpha$, so negative and positive arcs are treated equally by default. If you want to penalize a forbidden negative arc more than an erroneous positive arc, raise α in the dialog box, for instance, to 0.75. In consequence, a forbidden positive arc is weighted by 0.25, which is a third of the weight attached to an out-of-place negative arc. Finally, by increasing the *Min. number of vertices in Cluster* you can avoid getting clusters containing only one or a few vertices.

Figure 46 shows the results for the novices network. We used a random partition containing three classes as the starting partition, and it was instructed to weigh positive and negative errors equally ($\alpha = 0.5$). First, the listing displays the error score and the erroneous arcs in the initial clustering. There are many errors, which are identified by their line value (1 or -1) and their vertex numbers in the listing of lines. Using a random starting partition of your own, you will probably find a different list of errors. You should, however, find final solutions that match the ones displayed here, so let us concentrate on them.

With sufficient repetitions, the *Doreian–Mrvar Method** command finds three solutions with exactly one “forbidden” arc. In the first clustering, a positive arc wrongly connects vertices 7 (Mark) and 2 (Gregory), which are apparently members of different clusters. In the second clustering, a negative arc from vertex 1 (John Bosco) to vertex 7 (Mark) is a problem because it is situated within a cluster. In the third clustering, the positive arc from vertex 2 (Gregory) to vertex 1 (John Bosco) causes problems. As expected, the unclusterable triple John Bosco–Mark–Gregory causes these problems. Nevertheless, the clustering is nearly perfect, so we may conclude that the network is clusterable. To know whether it is balanced as well, we must repeat the procedure with a starting partition containing two clusters.

All optimal solutions are saved as partitions in the *Partitions* dropdown menu. Drawing the network with these partitions, we can see that clusters at the left and at the top are correctly identified. The cluster at

```
-----
Signed Graphs: positive diagonal blocks, negative off-diagonal blocks
-----
```

```
Working...
Number of clusters: 3, alpha: 0.500, min size of clusters: 1
----- Starting partition -----
Errors:      5.50      Lines
-----
          -1.00 :      6.3
          -1.00 :     14.4
           1.00 :      1.3
           1.00 :      3.1
           1.00 :      6.4
           1.00 :      7.2
           1.00 :      8.4
           1.00 :      9.8
           1.00 :     10.4
           1.00 :     14.1
           1.00 :     15.2
-----
----- Improvements -----
      1:      1.50
      2:      0.50
----- Final partition 1-----
Errors:      0.50      Lines
-----
          1.00 :      7.2
-----
----- Final partition 2-----
Errors:      0.50      Lines
-----
          -1.00 :      1.7
-----
----- Final partition 3-----
Errors:      0.50      Lines
-----
          1.00 :      2.1
-----

3 solutions with 0.50 inconsistencies found.
Time spent: 0:00:00
```

Figure 46. Output listing of a *Doreian-Mrvar Method** command.

the bottom is split in three ways (Figure 47): Mark is added to the cluster of Simplicius, Elias, and Amand (solution 1); he is part of an undivided cluster including Gregory and John Bosco (solution 2); or he is grouped with Albert, Boniface, and Gregory, who are separated from John Bosco, Basil, Hugh, and Winfrid (solution 3). The first and last solutions are most likely if negative arcs within a cluster are considered slightly more problematic than positive arcs between clusters. For instance, try the method with α set to 0.6.

Let us conclude this section with a warning. The *Doreian-Mrvar Method** command triggers a procedure that is very time consuming, so it should not be applied to networks with more than some hundreds of vertices unless you do not need your computer for some hours or days.

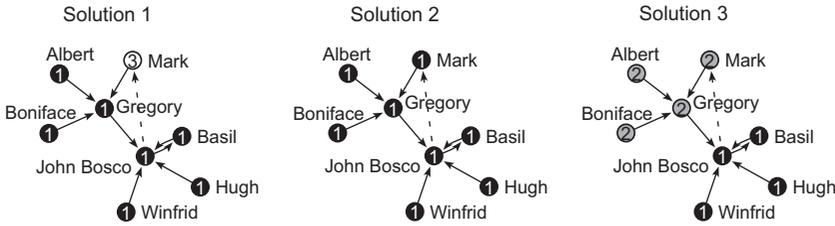


Figure 47. Three solutions with one error.

In Pajek, commands that should be applied only to small networks are marked by an asterisk in the menu.

Exercise I

Use Sampson's classification according to factions (`Sampson_factions_T4.clu`) to find four optimal clusters in the network of first choices at time four (`Sampson_T4.net`). Weight positive and negative errors equally. Do the optimal clustering(s) match Sampson's classification?

4.5 Development in Time

Balance theory expects a tendency toward balance. In the course of time, affect relations within a human group are hypothesized to become more balanced or clusterable. This raises the question of how to analyze the evolution of social networks. In this section, we discuss the simplest way to analyze longitudinal networks, namely, by comparing network structure at different points in time.

Sampson measured affect relations at the monastery at five moments. At the time of the first measurement (T_1), the group consisted mainly of novices who soon left the monastery to study elsewhere. The second measurement (T_2) concerns the period just after the arrival of a number of newcomers. The third measurement (T_3) shows affective ties around the time when one of the newcomers (Brother Gregory) organized a meeting to discuss the situation in the monastery. This meeting fueled a process of polarization, which led to the expulsion of four novices (among them Brother Gregory) one week after the fourth measurement (T_4). The expulsion triggered the voluntary departure of many novices in the next few weeks. At the time of the fifth measurement (T_5), no more than seven of the eighteen novices were still living in the monastery, so this network is difficult to compare to the previous networks. We analyze the networks at T_2 , T_3 , and T_4 only.

*Vertices	25			
1	"Leo"	0.1000	0.5000	0.5000 [1-1]
2	"Arsenius"	0.1126	0.4005	0.5000
3	"Bruno"	0.1495	0.3073	0.5000
4	"Thomas"	0.2084	0.2262	0.5000
5	"Bartholomew"	0.2857	0.1623	0.5000
6	"John Bosco"	0.3764	0.1196	0.5000 [2-4]
7	"Gregory"	0.4749	0.1008	0.5000
8	"Basil"	0.5750	0.1071	0.5000
9	"Martin"	0.6703	0.1381	0.5000 [1-1]
10	"Peter"	0.7550	0.1918	0.5000 [1-5]
11	"Bonaventure"	0.8236	0.2649	0.5000 [1-*]
12	"Berthold"	0.8719	0.3528	0.5000
13	"Mark"	0.8968	0.4499	0.5000 [1-4]
14	"Brocard"	0.8968	0.5501	0.5000 [1-1]
15	"Victor"	0.8719	0.6472	0.5000 [1-4]
16	"Ambrose"	0.8236	0.7351	0.5000 [1-*]
17	"Ramuald"	0.7550	0.8082	0.5000 [2-5]
18	"Louis"	0.6703	0.8619	0.5000 [2-*]
19	"Winfried"	0.5750	0.8929	0.5000 [2-5]
20	"Amand"	0.4749	0.8992	0.5000 [2-4]
21	"Hugh"	0.3764	0.8804	0.5000
22	"Boniface"	0.2857	0.8377	0.5000
23	"Albert"	0.2084	0.7738	0.5000
24	"Elias"	0.1495	0.6927	0.5000
25	"Simplicius"	0.1126	0.5995	0.5000
*Arcs				
6	7	2	[3]	
6	7	-2	[4]	
6	8	2	[2]	
6	8	3	[4]	
6	10	-2	[3]	
6	11	1	[3]	
6	11	3	[2]	
6	12	-2	[2]	
6	13	-1	[2]	
6	13	-3	[4]	
6	15	3	[3]	
6	17	-1	[4]	
6	17	-3	[2, 3]	

Figure 48. Partial listing of Sampson.net.

Application

Pajek has special facilities for longitudinal networks. Figure 48 shows part of the network file `Sampson.net`. By now, the structures of the lists of vertices and arcs are familiar, so the focus is on the time indicators in square brackets that are added to each vertex and arc. For instance, Brother John Bosco was at the monastery from T_2 up to and including T_4 . He left before T_5 . This is also true for Brothers Gregory and Basil;

Pajek assumes that a time indicator remains valid until it encounters a new one (e.g., with Brother Martin, who left the monastery after $T1$). Bonaventure arrived at the monastery before the first measurement, and he stayed after the last measurement. The asterisk (*) indicates infinity.

At $T3$, the arc from vertex 6 (John Bosco) to vertex 7 (Gregory) has a value of 2, indicating a positive second choice. At $T4$, however, it has turned into a second negative choice (line value -2). In Figure 48, the last line indicates that John Bosco chooses Ramuald (vertex 17) as the person he likes least (line value -3) at $T2$ and $T3$. Note that time is always represented by positive integers, and a line value must be specified before the time indication in the arcs and edges lists.

The time notation can be used to split the longitudinal network into separate networks for different moments or periods. The submenu *Network* > *Temporal Network* > *Generate in Time* offers the user several commands for generating a series of cross-sectional networks. First, you can choose to obtain a network for each period requested (option *All*) or to produce a network only if it differs from the previous one (option *Only Different*). The latter command is useful if a network does not change much over time. Whichever command you choose, you will have to specify the first and last time point you want to analyze as well as the time interval (step) between successive networks. In our example, we start at $T2$ (enter 2) and stop at $T4$ (enter 4), and we want a network for each moment in between, so we choose step value 1. Step values must be positive integers. For example, with a step value of 2, starting at the first moment in time, the command would create a network for the first, third, and fifth moments, and so on.

Note that serial numbers of vertices change in generated networks when vertices disappear from the network (as at $T5$) or when new vertices are added, because numbers of vertices must always range from 1 to the number of vertices in Pajek. As a result, a partition accompanying the original longitudinal network may not match the generated cross-sectional networks. Therefore, the *Generate in Time* command automatically creates new partitions for each generated network from the active partition, provided that it matches the original longitudinal network. The same is true for the active vector.

After generating networks for separate moments, you can easily switch from one moment to another with the commands *Previous* and *Next*, provided that the option *Network* is selected in the *Options* > *Previous/Next* > *Apply to* submenu in the Draw screen. If one of the energy options in the *Options* > *Previous/Next* > *Optimize Layouts* submenu has been selected, Pajek automatically energizes the network when you step to the next or previous network. It is sensible to inspect all generated networks to check whether the results are as intended. Errors in time indicators may have serious consequences; for instance, if Brother John

Network>
Temporal
Network>
Generate in
Time

Previous
Next
Options>
Previous/Next>
Apply to
Options>
Previous/Next>
Optimize
Layouts

Bosco is not present at $T3$ – by mistake, the indicator reads [2,4] instead of [2–4] – all arcs to and from him are deleted from the network at $T3$.

Network>
Signed
Network>
Create
Partition>
Doreian–Mrvar
Method*

Having generated networks for $T2$, $T3$, and $T4$, we can analyze the degree of balance or clusterability with the *Network> Signed Network> Create Partition> Doreian–Mrvar Method** command in each period. Table 6 shows the error scores associated with the optimal clustering for several numbers of clusters and different time points. Some optimal solutions are hard to find; you may need thousands of repetitions in some cases. Note that the network now consists of all three positive and negative choices at $T2$, $T3$, and $T4$. First choices count three times (line values are 3 and –3 for the first positive and negative choice, respectively), second choices count double (values 2 or –2), and third choices count once (1 or –1). The error score is computed from these line values, which explains why it is quite large compared to error scores in the previous section: A forbidden first choice contributes 0.5 (α) times 3 instead of 0.5 times 1 to the error score.

Table 6 helps draw a conclusion on the evolution of balance and clusterability in the network of novices. Each clustering fits better in the course of time, and the partition with three clusters fits best at any moment, so we may conclude that there is a tendency toward clusterability rather than balance. This is probably due to a process of polarization, which ends when several novices leave the monastery. Instead of a continuing trend toward balance or clusterability, human groups probably experience limited periods of polarization, which are reflected in increasingly balanced or clusterable patterns of affective ties.

Exercise II

Network>
Create New
Network>
Transform>
Remove> Lines
with Value>
within interval

Check whether the networks of first positive and negative choices display a tendency toward balance or clusterability from the second to the fourth moment in time. Hint: Use the command *Network> Create New Network> Transform> Remove> Lines with Value> within interval* to remove the lines with values from –2 to 2 from the longitudinal network

Table 6. *Error score with all choices at different moments ($\alpha = .5$)*

Number of Clusters	Time Points		
	$T2$	$T3$	$T4$
2 (balance)	21.5	16.0	12.5
3	17.5	11.0	10.5
4	19.0	13.5	12.5
5	20.5	16.0	15.0

(`Sampson.net`) before you split it into separate networks for different moments.

4.6 Summary

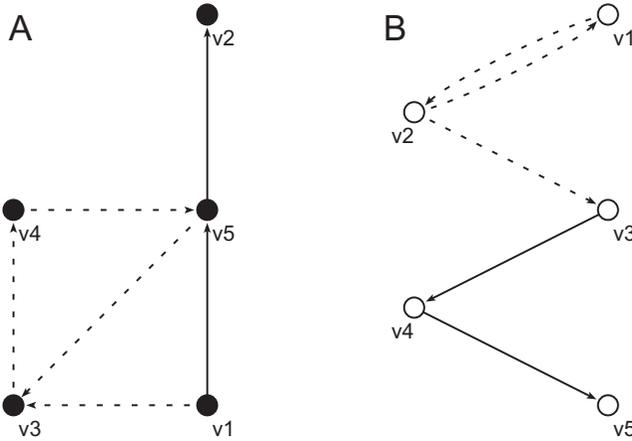
In this chapter, we discuss cohesive subgroups in signed networks, that is, in networks with positive and negative ties. If relations represent affections, people who like each other tend to huddle together, whereas negative sentiments exist predominantly between groups. This principle stems from balance theory and has been generalized to groups with three or more clusters (clusterability). Balance and clusterability occur in times of polarization between opposing factions.

We may determine whether affective ties are patterned according to balance theory by searching for a partition or clustering that satisfies the principle that positive lines are found within clusters and negative lines between clusters. If a signed network can be partitioned into two clusters according to this principle, the network is balanced. A partition with three or more clusters is a clusterable network.

The weighted number of lines that do not conform to the balance theoretic principle, namely, negative lines within a cluster and positive lines between clusters, indicates the degree of balance or clusterability in a network. This is called the error score of the best fitting clustering. For one network, we compare error scores for different numbers of clusters to find the optimal clustering. If the error score is acceptable, the clusters represent cohesive groups. In addition, we may compare error scores of a network at different moments to check whether group structure displays a tendency toward balance that is predicted by balance theory.

4.7 Questions

1. Which of the following statements is correct?
 - a. A signed graph is balanced if it is clusterable into two clusters.
 - b. A signed graph is balanced if it is not clusterable.
 - c. A signed graph is balanced if its vertices can be partitioned into two groups.
 - d. A signed graph cannot be balanced because it is undirected.
2. Have a look at the following two networks. Which of the following statements is correct?
 - a. A and B are balanced.
 - b. A is balanced, and B is clusterable.
 - c. A is clusterable, and B is balanced.
 - d. Neither A nor B is balanced.



3. How many cycles and semicycles does network A of Question 2 contain?
 - a. No cycles and two semicycles
 - b. One cycle and one semicycle
 - c. One cycle and two semicycles
 - d. Two cycles and one semicycle
4. Which of the following statements about the sequence of lines v2–v3–v4–v5 in network B (Question 2) is correct?
 - a. It is neither a semiwalk nor a semipath.
 - b. It is a semiwalk and a semipath.
 - c. It is a walk and a semipath.
 - d. It is a walk and a path.
5. An analysis of 58 alliance (a line value of 1) and 58 antagonistic (a line value of -1) ties among 16 tribes in New Guinea yields error scores reported in the following table ($\alpha = 0.5$). Which conclusion do you support? Please state your reasons.

	Number of Clusters			
	2	3	4	5
Error score	7.0	2.0	4.0	6.0

- a. The tribal network is balanced.
 - b. The tribal network is clusterable.
 - c. The tribal network is neither balanced nor clusterable.
 - d. It is impossible to draw a conclusion from these results.
6. In the best fitting clustering presented in Question 5, how many arcs violate the balance theoretic principle?

- a. Two arcs
- b. Four arcs
- c. Six arcs
- d. Seven arcs

4.8 Assignment

In 1943, Leslie D. Zeleny administered a sociometric test to forty-eight cadet pilots at a US Army Air Forces flying school. Cadets were trained to fly a two-seated aircraft, taking turns in flying and aerial observing. Cadets were assigned at random to instruction groups ranging in size from five to seven, so they had little or no control over who their flying partners would be. The sociometric test was used to improve the composition of instruction groups. Zeleny asked each cadet to name the members of his flight group with whom he would like to fly as well as those with whom he would not like to fly. The data are available in the project file `Flying_teams.paj`, which contains a network (`Flying_teams.net`) and the original (alphabetical) instruction groups (`Flying_teams.clu`).

Which instruction groups would you advise to reduce the risk of cadets flying with partners they do not want? Try to find groups of five to ten cadets.

4.9 Further Reading

- The example was taken from S. F. Sampson, *A Novitiate in a Period of Change: An Experimental and Case Study of Social Relationships* (Ph.D. thesis, Cornell University, 1968). An analysis of the data using the method presented in this chapter can be found in P. Doreian and A. Mrvar, “A partitioning approach to structural balance.” *Social Networks* 18 (1996), 149–68.
- The New Guinea data were reported in K. Read, “Culture of the central highlands, New Guinea.” *Southwestern Journal of Anthropology* 10 (1954), 1–43, and reanalyzed in P. Hage and F. Harary, *Structural Models in Anthropology* (Cambridge: Cambridge University Press, 1983). The data are also distributed with UCINET network analysis software.
- The flying team’s data are taken from J. L. Moreno, *The Sociometry Reader* (Glencoe, IL: The Free Press, 1960, pp. 534–47).
- For an overview of balance theory in social network analysis, consult Chapter 6 in S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications* (Cambridge: Cambridge University Press, 1994).

- Fritz Heider introduced the notion of balance in his article “Attitudes and cognitive organization,” which appeared in the *Journal of Psychology* 21 (1946), 107–12. A more detailed account can be found in his book, *The Psychology of Interpersonal Relations* (New York, NY: John Wiley & Sons, 1958). F. Harary, R. Z. Norman, and D. Cartwright, *Structural Models: An Introduction to the Theory of Directed Graphs* (New York, NY: John Wiley & Sons, 1965) formalized the concept of balance. The lucidity of this book makes it a pleasure to read. Clusterability was defined by J. A. Davis in the article “Clustering and structural balance in graphs.” *Human Relations* 20 (1967), 181–7. *Discrete Mathematical Models* by F. S. Roberts (Englewood Cliffs, NJ: Prentice Hall, 1976) is a good book to further your understanding of several mathematical models, including balance.
- Relaxed balance was introduced in P. Doreian and A. Mrvar, “Partitioning signed social networks.” *Social Networks* 31 (2009), 1–11. Approaches to partitioning signed two-mode networks are described in A. Mrvar and P. Doreian, “Partitioning signed two-mode networks.” *Journal of Mathematical Sociology* 33 (2009), 196–221. Links to many other papers on signed networks can be found on the Pajek webpage.

4.10 Answers

Answers to the Exercises

- With a sufficient number of repetitions, the *Doreian–Mrvar Method** command finds two optimal partitions with one mistake each (0.5 error score). In one partition, Mark (vertex 7) is considered a cluster on his own, so his choice of Gregory (vertex 2) as his most liked colleague is erroneously situated between two classes instead of within a class (Figure 49, left).

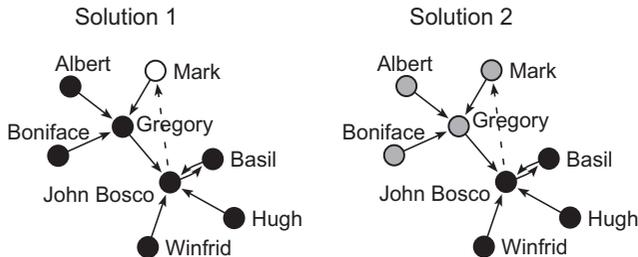


Figure 49. Differences between two solutions with four classes.

In the other partition, Mark, Albert, Gregory, and Boniface are separated from John Bosco, Basil, Hugh, and Winfrid (Figure 49, right). Now, the positive arc from Gregory (vertex 2) to John Bosco (vertex 1) is troublesome. The first solution matches Sampson's classification into factions better because he assigned Gregory, Albert, Boniface, John Bosco, Hugh, and Winfrid to the Young Turks (class 1 in Figure 45). In both partitions, however, the interstitial group (Amand, Victor, and Ramuald; class 4) is divided over the Loyal Opposition (class 2) and the Outcasts (class 3).

- II. Remove the lines with values from -2 to 2 from the longitudinal network (`Sampson.net`) with the command `Network> Create New Network> Transform> Remove> Lines with Value> within interval` (enter -2 as the lower limit and 2 as the upper limit), and then split it with `Network> Temporal Network> Generate in Time> All`. For the networks at time two to four, find the optimal balanced partitions with the `Network> Signed Network> Create Partition> Doreian-Mrvar Method*` command as described in Section 4.5. If you use many repetitions, you should find the results summarized in Table 7. We already obtained some results for time four in Section 4.4, but now the error scores are three times as high because the line values are 3 and -3 instead of 1 and -1 . The error scores tend to diminish in the course of time, and they are lowest with three or four classes: This indicates a tendency toward clusterability rather than balance.

Answers to the Questions in Section 4.7

1. Answer a is correct. Balance is a special case of clusterability, because a network is balanced if it can be partitioned into two clusters in such a way that all positive lines are within clusters and all negative lines are between clusters.
2. Answer c is correct. Semicycles $v1-v3-v4-v5$ and $v3-v4-v5$ contain three negative arcs, so network A is not balanced. Cycle $v1-v3-v5$ has two negative arcs. There is no (semi-)cycle with exactly one negative

Table 7. Error score with first choices only ($\alpha = .5$)

Number of Clusters	Time Points		
	T2	T3	T4
2 (balance)	7.5	4.5	3.0
3	4.5	3.0	1.5
4	4.5	3.0	1.5
5	4.5	3.0	3.0
6	6.0	4.5	4.5

arc, so network A is clusterable. Network B contains just one (semi-) cycle, $v1-v2$, which contains two negative arcs. Hence, network B is balanced. Network A can be partitioned according to the principle of positive arcs within clusters and negative arcs between clusters: $v1, v2$, and $v5$ are a cluster, and $v3$ and $v4$ are separate clusters. Likewise, network B can be divided into two clusters: $v2$ versus the rest.

3. Answer c is correct, see the answer to Question 2.
4. Answer d is correct. The sequence of lines is a walk because the lines are adjacent and they point in the same direction: The head of one arc is the tail of the next arc. This walk is a path because all vertices are distinct, meaning that no vertex occurs more than once in the walk.
5. Answer b is the most likely conclusion. Clearly, a partition with three clusters fits better than a partition with two clusters. Therefore, the network is clusterable rather than balanced. Strictly speaking, however, even an error score of 2.0 is too high, and you should conclude that the network is neither balanced nor clusterable (answer c).
6. Answer b is correct. Alpha is 0.5, which means that every “forbidden” arc – positive or negative – contributes 0.5 to the error score. Because the absolute line values are 1, four arcs in the wrong places yield a total error score of 2.0, which is the minimal error score in the table.

Affiliations

5.1 Introduction

Membership of an organization or participation in an event is a source of social ties. In organizations and events, people gather because they have similar tasks or interests and they are likely to interact. Members of a sports club, for instance, share a preference for a particular sport and play with or against one another. Directors and commissioners on the boards of a corporation are collectively responsible for its financial success and meet regularly to discuss business matters. Inspired by the sociology of Georg Simmel, groups of people who gather around one or more organizations and events are called *social circles*.

In previous chapters, we studied direct ties among people, such as the choice of friends, or among other social entities, for instance, trade relations between countries. Note that we studied relations among actors of one kind: relations between people or between organizations but not between people and organizations. Now, we focus on the latter type, which is called an affiliation. Data on affiliations can be obtained relatively easily, and they are very popular in data mining.

Affiliations are often institutional or “structural,” that is, forced by circumstances. They are less personal and result from private choices to a lesser degree than sentiments and friendship. Of course, membership in a sports team depends much more on a person’s preferences than detention in a particular prison ward, but even the composition of sports teams depends on circumstances and on decisions made by coaches and sports club authorities. Affiliations express institutional arrangements; and because institutions shape the structure of society, networks of affiliations tell us a lot about society. People are often affiliated to several organizations and events at the same time, so they belong with a number of social circles; or, in other words, they are the intersection of many social circles. Society may be seen as a fabric of intersecting social circles.

Although membership lists do not tell us exactly which people interact, communicate, and like each other, we may assume that there is a fair chance that they will. Moreover, joint membership in an organization often entails similarities in other social domains. If, for example, people have chosen to become members in (or have been admitted to) a particular golf club, they may well have similar professions, interests, and social status. Different types of affiliations do not overlap in a random manner: Social circles usually contain people who are clustered by affiliations to more than one type of organization. From the number or intensity of shared events, we may infer the degree of similarity of people. However, this argument can be reversed: Organizations or events that share more members are also closer socially. A country club with many members from the local business elite can be said to be part of the business sphere.

In this chapter, we present a technique for analyzing networks of affiliations that focuses on line values. In addition, we discuss three-dimensional displays of social networks.

5.2 Example

In political science, economy, and sociology, much attention has been paid to the composition of the boards of large corporations. Who are the directors of the largest companies; and, in particular, who sits on the boards of several companies? If a person is a member of the board of directors in two companies, he or she (although women are found less often in these positions) is a multiple director who creates an interlocking directorate or interlock between firms.

The network of interlocking directorates tells us something about the organization of a business sector. It is assumed that interlocking directorates are channels of communication between firms. In one board, a multiple director can use the information acquired in another board. Information may or may not be used to exercise power, depending on the role played by the director. If directors are elected because of their social prestige within a community, they serve the public relations of a firm, but they do not influence its policy: They fulfill a symbolic role. However, multiple directors who have executive power may coordinate decisions in several companies, thus controlling large sections of an economy. Then, interlocking directorates are power lines.

In this chapter, we use a historical example: the corporate interlocks in Scotland in the beginning of the twentieth century (1904–5). In the nineteenth century, the Industrial Revolution brought Scotland railways and industrialization, especially heavy industry and the textile industry. The amount of capital needed for these large-scale undertakings exceeded the means of private families, so joint stock companies were established that

could raise the required capital. Joint stock companies are owned by the shareholders, who are represented by a board of directors. This opens up the possibility of interlocking directorates. By the end of the nineteenth century, joint stock companies had become the predominant form of business enterprise at the expense of private family businesses. However, families still exercised control through ownership and directorships.

The data are taken from the book *The Anatomy of Scottish Capital* by John Scott and Michael Hughes. It lists the (136) multiple directors of the 108 largest joint stock companies in Scotland in 1904–5: sixty-four nonfinancial firms, eight banks, fourteen insurance companies, and twenty-two investment and property companies (Scotland.net). The companies are classified according to industry type (see `Industrial_categories.clu`): 1 – oil & mining, 2 – railway, 3 – engineering & steel, 4 – electricity & chemicals, 5 – domestic products, 6 – banks, 7 – insurance, and 8 – investment. In addition, there is a vector specifying the total capital or deposits of the firms in 1,000 pounds sterling (`Capital.vec`). The data files are collected in the project file `Scotland.paj`.

Exercise I

Open the project file `Scotland.paj`. To obtain the two-mode or affiliation partition, run the `Network>2-Mode Network> Partition into 2 Modes` command. Draw the network of firms and directors with the affiliation partition. What do the classes in the affiliation partition mean?

`Network>`
`2-Mode`
`Network>`
`Partition into 2`
`Modes`

5.3 Two-Mode and One-Mode Networks

By definition, affiliation networks consist of at least two sets of vertices such that affiliations connect vertices from different sets only. There are usually two sets, which are called *actors* and *events*, for example, directors (actors) and boards of corporations (events). Affiliations connect directors to boards, not directors to directors or boards to boards, at least not directly. Figure 50 shows a fragment of the interlocking directorates network in Scotland: a set of directors (“man” shape in the figure, “box” shape on your computer screen) and firms (“house” shape in the figure, “triangle” shape on your computer screen). In Chapter 2 we learned how to draw Unicode symbols like “\$” in the center of vertices in Pajek. See Appendix 2 to learn how to get additional vertex shapes like “man” and “house” as well as curved lines instead of straight lines.

The network file is available as `FragmentScotland.paj`. Note that lines always connect a vertex from the first set (drawn as a “man”) and a vertex from the second set (drawn as a “house”), e.g., director J. S. Tait to the Union Bank of Scotland. This type of network is called a *two-mode network* or a *bipartite network*, which is structurally different from the

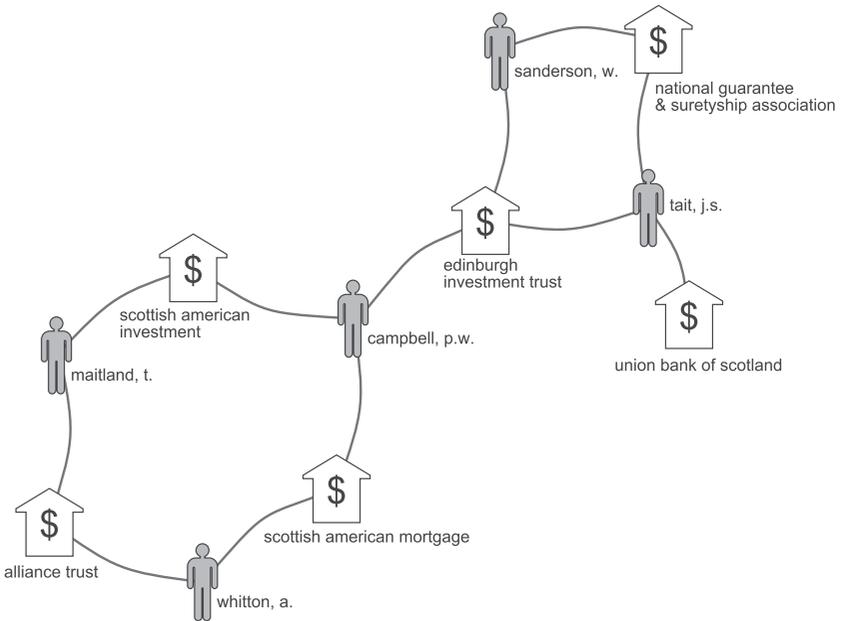


Figure 50. A fragment of the Scottish directorate's network.

one-mode networks, which we have analyzed thus far because all vertices can be related in a *one-mode network*.

In a *one-mode network*, each vertex can be related to every other vertex.

In a *two-mode network*, vertices are divided into two sets and vertices can be related only to vertices in the other set.

We can describe the two-mode network of Scottish directorships in the usual manner by its number of vertices (108 firms and 136 multiple directors – see the affiliation partition) and lines (358 affiliations or board seats), the number of components (16 isolated firms without multiple directors, 3 small components containing 2 firms, and one large component), and its degree distribution. Recall that the degree of a vertex is equal to the number of its neighbors if the network does not contain multiple lines and loops. Because this is the case in the directorships network, the degree of a firm specifies the number of its multiple directors. This is known as the *size of an event*. The degree of a director equals the number of boards he or she sits on, which is called the *rate of participation* of

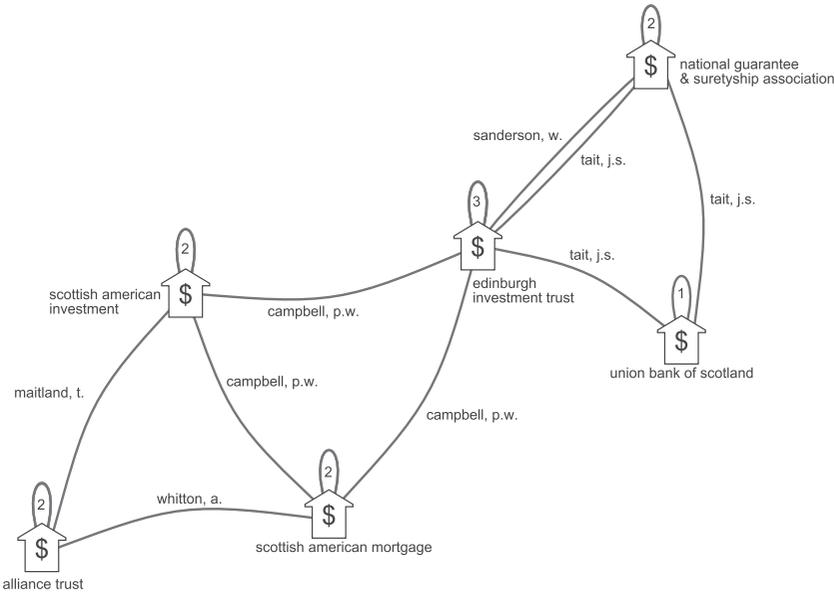


Figure 51. One-mode network of firms created from the network in Figure 50.

an actor. Have a look at Figure 50 and determine the size of events and participation rates of its vertices.

In our description of a two-mode network, we must distinguish between actors and events, because simple measures such as degree have different meanings for actors and events. There are more complications: Some structural indices must be computed in a different way for two-mode networks. Consider, for example, the concept of completeness, which we defined as the maximum possible number of lines in a network (see Chapter 3). In a one-mode network, this number is much higher than in a two-mode network because each vertex can be related to all other vertices in a one-mode network, but it can be related only to part of the vertices in a two-mode network, namely the vertices in the other mode. As a consequence, the density of a two-mode network, which is the actual number of lines divided by the maximum possible number of lines, must be computed differently for one-mode and two-mode networks.

Techniques for analyzing one-mode networks cannot always be applied to two-mode networks without modification or change of meaning. Special techniques for two-mode networks are quite complicated and fall outside the scope of this book. So, what can we do? The solution commonly used, which we will follow, is to change the two-mode network into a one-mode network, which can be analyzed with standard techniques.

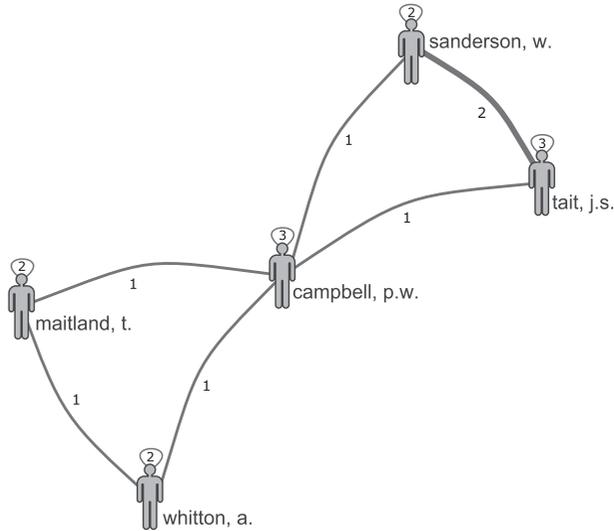


Figure 52. One-mode network of directors derived from Figure 50.

We can create two one-mode networks from a two-mode network: a network of interlocking events and a network of actors that are members of the same organization or attend common events. Figure 51 shows the one-mode network of firms (events) that is derived from the network in Figure 50. It is constructed in the following way. Whenever two firms share a director in the two-mode network, there is a line between them in the one-mode network. For instance, J. S. Tait creates a line between the Union Bank of Scotland and The Edinburgh Investment Trust because he sits on the boards of both companies. Because he is also on the board of the National Guarantee and Suretyship Association, he is responsible for lines between three firms. Each line can be labeled by his name. In addition, he creates a loop for each of these firms. The number of loops incident with a vertex shows the number of its neighbors in the two-mode network. In our example, it shows the number of multiple directors on the board of a firm: the size of the event. In short, the actors in the two-mode network become the lines and loops in the one-mode network of events.

From Figure 51, it is clear that firms can be connected by multiple lines, namely, in the case that two firms share more than one director. The derived network, therefore, usually is not a simple network. Because it may also contain loops, you must take care when you interpret the degree of a vertex in a network derived from a two-mode network.

Multiple lines can be replaced by a single line to obtain a *valued network*, with a line value indicating the original number of lines between two vertices. Such a line value is called a *line multiplicity*. Figure 52 shows

the valued network of directors (comembership) that can be derived from the example in Figure 50. Now, the events of the two-mode network are represented by lines and loops in the one-mode network of actors. J. S. Tait meets W. Sanderson in board meetings of two companies. We are confident that you can trace the firms that are responsible for the lines in this network.

Although we stated that one-mode networks derived from two-mode networks can be analyzed with standard techniques, there is a risk of interpreting the results erroneously. A direct tie in a derived one-mode network is easy to interpret: It indicates that two boards have common directors or that two directors meet at one or more boards. Absence of a direct tie implies that two boards do not share a director (e.g., Alliance Trust and Edinburgh Investment Trust in Figure 51) or that two directors do not meet in a board, for instance, A. Whitton and J. S. Tait in Figure 52.

The interpretation of subgroups consisting of three or more vertices is more complicated in derived one-mode networks. Figure 51 contains three cliques of size three. Two cliques are due to directorships of one person, namely P. W. Campbell and J. S. Tait, but one clique is not: The clique of size three with Scottish American Mortgage, Alliance Trust, and Scottish American Investment exists thanks to the memberships of three directors (Campbell, Whitton, and Maitland). In a valued network, this difference is not visible because the multiple lines labeled by names of directors are replaced by single lines with multiplicity values. When interpreting a derived valued network, restrict your conclusions about the number of shared persons (or events) to pairs of vertices. For threesomes and larger sets of vertices, you can conclude only that they share one or more actors or events, but you do not know the actual number.

Application

Pajek has special facilities for two-mode networks. We advise to use the data format for two-mode data, which is an ordinary list of vertices and list of edges with the special feature that the list of vertices is sorted: The first part contains all vertices that belong to one subset, and the remainder lists the vertices of the other subset. In our example (*Scotland.net*), vertices numbered 1 to 108 are firms and 109 to 224 are multiple directors. The first line of the data file specifies the total number of vertices and the number of vertices in the first subset (e.g., **Vertices 244 108*). Use the *Network> 2-Mode Network> Partition into 2 Modes* command to create a partition that distinguishes between the first subset of vertices (class 1) and the second (class 2). This partition is usually called “affiliation partition” or “2-Mode” partition in Pajek.

The submenu *Network> 2-Mode Network> 2-Mode to 1-Mode* contains commands for translating two-mode into one-mode networks. You can create a one-mode network on each of the two subsets of vertices.

Network>
2-Mode
Network>
Partition into 2
Modes

Network>
2-Mode
Network>
2-Mode to
1-Mode>
Rows, Columns

Table 8. *Line multiplicity in the one-mode network of firms*

Line Values	Frequency	Freq%	CumFreq	CumFreq%
(... 1.0000]	231	83.6957	231	83.6957
(1.0000 ... 2.0000]	28	10.1449	259	93.8406
(2.0000 ... 3.0000]	7	2.5362	266	96.3768
(3.0000 ... 4.0000]	3	1.0870	269	97.4638
(4.0000 ... 5.0000]	1	0.3623	270	97.8261
(5.0000 ... 6.0000]	0	0.0000	270	97.8261
(6.0000 ... 7.0000]	0	0.0000	270	97.8261
(7.0000 ... 8.0000]	6	2.1739	276	100.0000
TOTAL	276	100.0000		

By convention, vertices of the first subset are called rows, whereas the vertices of the second subset are called columns. These terms are derived from matrix notation, which we present in Chapter 12. In our example, the first subset contains the firms, so the *Rows* command in the *2-Mode to 1-Mode* submenu will create a network of firms, provided that the two-mode network is selected in the first *Network* drop-down menu of the Main screen. The *Columns* command creates a network of directors.

Network>
2-Mode
Network>
2-Mode to
1-Mode>
Include Loops,
Multiple Lines

Check the *Include Loops* option before you derive a one-mode network if you want to know the number of affiliations per vertex in the new network. Depending on the subset you choose for induction, loops specify the participation rate of actors or the size of events. When the option *Multiple Lines* is checked, the derived network will contain one line for each shared actor or event with labels of the events or actors that create the lines. If this option is not checked, a valued network without multiple lines is created with line values expressing line multiplicity. Usually, you do not want loops or multiple lines in the one-mode network, so we do not check these options now.

Network>
Info> Line
Values

There is an easy way to display the distribution of line values in Pajek; for example, line multiplicity in a derived one-mode network can be displayed by executing the command *Line Values* from the *Network> Info* submenu. In a dialog box, you may either specify custom class boundaries or you may choose a number of classes of equal width. To obtain classes of equal width, type a number preceded by a pound (#) sign in the dialog box. Usually, the number suggested by the dialog box serves the purpose.

Table 8 lists the multiplicity of the lines in the one-mode network of Scottish firms. The first class contains 231 lines with values up to and including 1. Because there are no lines with a multiplicity of less than 1, this class contains all lines with multiplicity 1: the single lines. The next class contains lines with values higher than 1, up to and including lines with a value of 2. We assume that you will understand that all 28 lines in this class have a multiplicity of 2. They refer to pairs of firms interlocked by two directors.

You can use the affiliation partition, which distinguishes between the two modes (actors and events), to select the vertices of one mode from a partition or vector associated with the two-mode network. The standard techniques for extracting one or more classes from one partition or vector according to another, which was presented in Chapter 2, can be used to this end.

Imagine, for example, that we want to know the degree of the firms in the two-mode network, which is equal to the number of their multiple directors (the size of the events). We compute the degree in the usual manner with the *Degree> Input* command in the *Network> Create Partition* submenu. The partition created by this command does not distinguish between the firms and the directors, so we must extract the firms from it. We select the degree partition in the first *Partition* drop-down menu. Next, we select the affiliation partition as the second partition because it identifies the firms within the network. Finally, we extract class 1 (the firms) of the affiliation partition from the degree partition with the *Partitions> Extract SubPartition (Second from First)* command. Now, we can make a frequency distribution (*Partition > Info*) of the degree of the firms. In the same way, we can translate partitions belonging to a two-mode network to a one-mode network derived from it.

```
Network>
Create
Partition>
Degree> Input
```

```
Partitions>
Extract
SubPartition
(Second from
First)
```

Exercise II

Derive the one-mode network of firms without loops and multiple lines from the two-mode network in *Scotland.paj*. Energize and adjust the network manually until you obtain a structure such as the one depicted in Figure 53. Pay no attention to the contours, which were added manually in special drawing software (see Appendix 2), or to the colors of the vertices.

5.4 Islands

One-mode networks derived from two-mode affiliation networks are often rather dense. They contain many cliques, so we can analyze the structure of overlapping cliques or complete subnetworks if we want to detect cohesive subgroups (see Chapter 3, Section 3.6). Chapter 12 presents additional techniques that are useful for analyzing dense networks. In the present chapter, however, we concentrate on a technique based on line multiplicity: *islands*.

Multiple lines are considered more important because they are less personal and more institutional. From this point of view, we may define cohesive subgroups on line multiplicity rather than on the number of neighbors. The larger the number of interlocks between two firms, the stronger or more cohesive their tie, the more similar or interdependent they are.

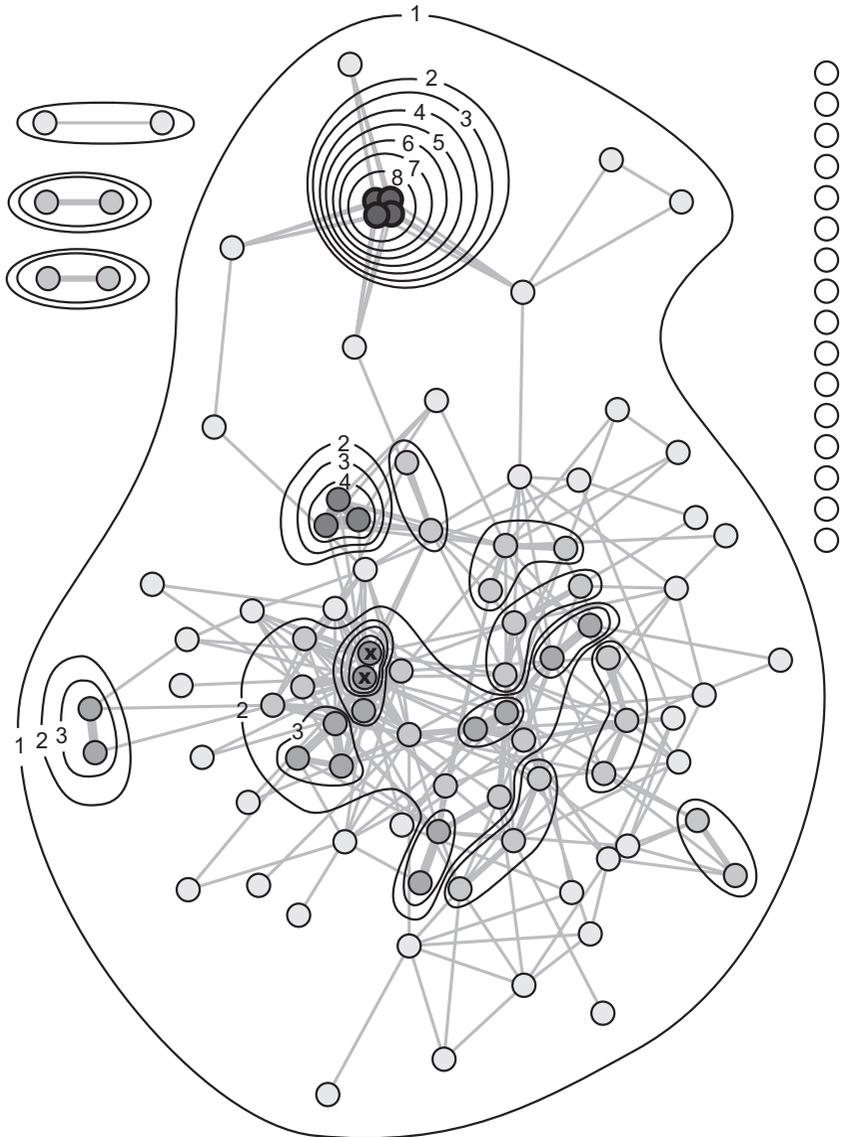


Figure 53. Islands in the network of Scottish firms, 1904–5 (contours added manually).

In Figure 53, for instance, the four gray firms share eight directors; they are connected by six lines of multiplicity level 8 (compare Table 8). These firms are connected much more tightly than other firms, which are connected at a multiplicity level of 5 or less.

This brings us to the concept of an island: a subnetwork defined by the multiplicity or value of lines. Within an island, vertices are connected directly or indirectly by lines of a particular minimum value whereas they are connected at lower values to vertices outside the island.

An *island* is a maximal subnetwork of vertices connected directly or indirectly by lines with a value greater than the lines to vertices outside the subnetwork.

An island can be thought of as a local summit in the network if we use the highest value of the lines incident with this vertex as its height. An island, then, is a set of vertices raised above its immediate surroundings. This implies that islands can be found at different altitudes: mere hillocks in a flat plain, such as the Vaalserberg in The Netherlands (alt. 1,059 feet), or pronounced peaks in a mountain range, such as the Triglav in Slovenia (9,396 feet). In Figure 53, we manually added contours to the islands in the one-mode network of Scottish firms. The contours give a feel of the height of the islands. The four dark gray vertices in the top middle of Figure 53 represent an island of four firms sharing eight directors among them. This island is sharply distinguished from its surroundings because the ties to firms outside the island have a maximum multiplicity of 2.

Figure 54 shows the islands with the industrial categories of the firms (class numbers from `Industrial_categories.clu`) and their economic value (vertex size determined by `Capital.vec`). The large component in Figure 53 is now broken down into a number of islands: several small islands of firms of one type (domestic products [class 5], railways [class 2], electricity [class 4], and investments [class 8]) and one larger island mainly connected by financial institutions (banks [class 6], insurance companies [class 7], and investment banks [class 8]). In the large island, the wealthy Caledonian Railway and the Scottish Widows Fund occupy pivotal positions. We may conclude that plural interlocks interconnect financial organizations rather than that they link the financial sphere to the heavy industries or to the production of consumer goods in this historical example.

Application

In a valued network, the `Network> Create Partition> Islands> Line Weights` command identifies the islands. It ignores the direction of lines in directed networks. When issued, this command asks for a minimum and maximum size of the islands. The default minimum size is one. If you raise this minimum, small islands will simply be ignored. Unless your network contains many uninteresting small islands, you can use one or two as the minimum island size.

```
Network>
Create
Partition>
Islands> Line
Weights
```

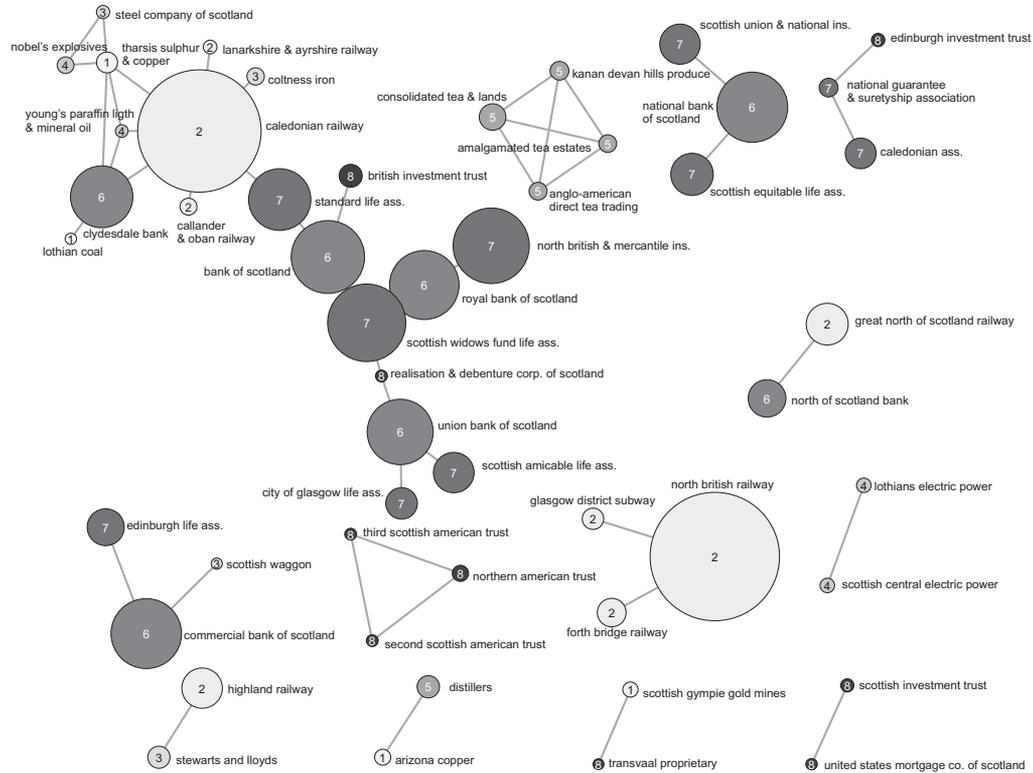


Figure 54. The islands in the network of Scottish firms (1904–5) with industrial categories (class numbers) and capital (vertex size).

The maximum size of islands is a bit trickier. It should be sufficiently large to contain large interesting islands in the network. However, it may ignore differentiated line values within a component or an island if the maximum size is greater or equal to that component or island. So if you want to check a component for local clusters of vertices connected by lines with relatively high values, you should set the maximum size below the size of that component. The only large component in the Scottish firms network contains 86 vertices, so the maximum should be set below this number, for instance at 70.

The islands command produces a partition and a vector. In the partition, each detected island is assigned a class number with vertices that are not part of an island collected in class 0. The vector assigns the line value defining the island to the vertices that are part of the island – their “height.” If the option *Network> Create Partition> Islands> Generate Network with Islands* was set before issuing the islands command, it also produces a new network containing all vertices and only the lines that define islands. In this network, the islands are weak components.

Figure 54 shows the islands in the Scottish firm’s network with industrial categories as class numbers and vertex size determined by the firm’s capital. The vertices that do not belong to an island were removed in the following manner. First, a degree partition was created for the islands network. Second, the vertices with at least one neighbor (partition classes 1-*) were extracted from the network with the *Operations> Network + Partition> Extract> SubNetwork Induced by Union of Selected Clusters* command. The new network contains only 51 of the original 108 vertices, so the industrial categories partition and capital size vector do not fit the new network. To make them fit, use *Partitions> Extract SubPartition (Second from First)* (make sure that *Industrial_categories.clu* and the islands partition are selected as the first and second partition) and *Operations> Vector + Partition> Extract SubVector* (make sure the islands partition is selected in the first *Partition* drop-down menu) commands discussed in Chapter 2 (Sections 2.4 and 2.5, respectively). If you draw the new network, partition, and vector (command *Draw> Network + First Partition + First Vector*), you obtain a sociogram similar to the one depicted in Figure 54, provided that you optimize the obtained a layout with the Kamada–Kawai command that separates components (*Layout> Energy> Kamada-Kawai> Separate Components*).

Finally, Pajek Classes offers a powerful tool to show islands interactively on the web with the *Export> 2D> SVG* submenu. Select the one-mode network of firms and the islands partition in the drop-down menu. Then, draw the network and partition and execute the command *Nested Classes* from the submenu *Export> 2D> SVG> Line Values* in the Draw screen to obtain an HTML file and SVG file containing the drawing of the network. Dialog boxes ask for the name of the HTML (and SVG) file that

```
Network>
Create
Partition>
Islands>
Generate
Network with
Islands
```

```
Network>
Create
Partition>
Degree
```

```
Operations>
Network +
Partition>
Extract>
SubNetwork
Induced by
Union of
Selected
Clusters
```

```
Partitions>
Extract
SubPartition
(Second from
First)
```

```
Operations>
Vector +
Partition>
Extract
SubVector
```

```
Draw>
Network +
First Partition
+ First Vector
```

```
Layout>
Energy>
Kamada-Kawai>
Separate
Components
```

```
Export> 2D>
SVG> Line
Values> Nested
Classes
```

stores the drawing and the number of classes from the active partition that must be represented by different layers. In the latter case, accept the default number suggested in the dialog box.

If you open this file in an Internet browser (see Appendix 2 for details), you will see the sociogram and a set of checkboxes to its right. Each checkbox is associated with a class of lines. If you deselect a checkbox, all lines with values up to and including the deselected class are removed from the picture as well as the vertices that do not belong to the remaining islands. With the checkboxes you can view the lines and vertices at different multiplicity levels interactively. If checkboxes for any reason do not function, try a different browser. Mozilla Firefox, for example, worked well at the time of writing this book.

On the book's website (<http://mrvar.fdv.uni-lj.si/pajek/>), the HTML file `islands.htm`, which loads the file `islands.svg` automatically, offers an example. Note that deselecting the `(1.00...2.00]` checkbox, part of the lines in the largest island (containing 20 vertices) disappears, so the island is broken down into five smaller islands. Because the maximum size of islands was set over 20, the command does not differentiate within the set of 20 vertices. This illustrates the impact of the maximum size settings on the results of the islands command.

5.5 Communities

In Section 5.4, we used islands to identify densely connected groups. Several other methods for identifying groups exist. Let us quickly introduce another set of methods which is widely used: community detection methods. Communities are (briefly speaking) clusters of vertices for which the density of lines inside clusters is larger than the density of lines between clusters. Values of lines are taken into account: Lines within a community are supposed to have larger values than lines between communities. Community detection, however, can also be applied to networks without line values.

Pajek offers two community detection methods: *Louvain Method* and *VOS Clustering*. The *Louvain Method* searches for the partition of vertices into clusters with the highest value of *modularity*. *Modularity* is a popular measure for comparing the density of lines and their line values inside and outside clusters. *VOS Clustering* is a similar method, which uses the *VOS quality function* instead of *modularity*. The two measures are slightly different but we will not go into details or provide formulae here. For details on both methods see Further Reading.

A much simpler but popular measure of how well clusters divide a network into cohesive subgroups is the *E-I Index*: External-Internal Index. The E-I Index subtracts the number of lines within clusters from the

number of lines between the clusters. The difference is divided by the total number of lines. Line values can be taken into account. Values of the E-I Index range from -1 to 1 . If the E-I Index is -1 , all lines are inside clusters whereas the value 1 means that all lines are between clusters. The value 0 indicates that the number of lines (or the sum of line values) between clusters equals the number of lines (or sum of line values) inside clusters. If a partition classifies vertices into cohesive groups well, lines should be within clusters rather than between clusters, so the E-I Index should be negative and preferably close to -1 .

Application

Searching for communities is available under *Network> Create Partition> Communities*. For each of the two methods (*Louvain* and *VOS*) two suboptions are available: *Multi-Level Coarsening + Single Refinement* and *Multi-Level Coarsening + Multi-Level Refinement*. The first one is a bit faster, while the second one gives more stable results. Details can be found in Further Reading. We recommend to use single refinement only when networks are really large. In this example we select multi-level refinement. After we have selected one of the methods for searching communities, we are confronted with an input box with several parameters that can be changed. Most of them need to be changed only when we deal with larger networks. The only parameter that is always important is the *resolution parameter*. The larger this parameter, the larger the number of communities that we find, so the smaller the communities are. By default, the resolution parameter is set to 1 , which is a good choice to start with.

As an example, let us search for communities in the one-mode network of the Scottish firms. Using the Louvain Method and resolution parameter set to 1 we obtain twenty-four communities with modularity 0.59 . By applying VOS Clustering we get twenty-nine communities with VOS quality function 0.78 . Note that modularity and VOS quality function values are not directly comparable; we cannot say that the method with which we obtain a higher value is a better one. If you get a lower value for modularity and/or the VOS quality function than the one we report, run the procedure again with higher number of restarts, e.g., 100 . Both community detection procedures work by optimization, so usually several repetitions are needed to get a stable solution. But the results seem quite stable in this example.

Let us now compare the Louvain and VOS community partitions. Cramer's V indicates a very strong association (0.98), which suggests that the two community detection methods yield very similar results. Expected frequencies, however, are very low, so we should check other indices as well. Adjusted Rand Index (0.59) and the three Rajske coefficients (all of them well over 0.70) also suggest strong association. If we compare

Network>
Create
Partition>
Communities>
Louvain
Method, VOS
Clustering

Partitions>
Info> Cramer's
V, Rajske,
Adjusted Rand
Index

the communities to the islands that we detected before, we get lower but still quite high values for Cramer's V: 0.59 in the comparison of islands to Louvain communities and 0.66 comparing islands to VOS Clustering. But in this case Adjusted Rand Index values are low (0.02 and 0.01 respectively). The differences are mainly due to the fact that several vertices are not assigned to islands (they are in cluster 0 of the islands partition) while vertices are always assigned to a community.

Partitions>
Extract
(Sub)Partition
(Second from
First)

We had better exclude the firms that are not part of an island if we want to compare the islands to the communities. So, let us extract the vertices with nonzero values in islands partition from the community partitions obtained with the Louvain Method and VOS Clustering (see Section 5.4 on islands): Select one of the communities partition as the first partition and the islands partition as the second. Then use *Partitions> Extract (Sub)Partition (Second from First)* and enter 1-* when asked for clusters to be extracted. After extracting subpartitions, all partitions have dimension 51, that is, clusters for fifty-one vertices. If we compute Cramer's V and Adjusted Rand Index again, we get much higher associations between the islands and communities. We can conclude that the three methods give us very similar results. Now take your time and play with searching for communities by setting the resolution parameter to different values (higher and lower than 1) until you get communities that fit your needs the best.

Operations>
Network +
Partition>
Info> E-I
Index

Operations>
Network +
Partition>
Extract>
SubNetwork
Induced by
Union of
Selected
Clusters

What are the E-I Index values for the Islands, Louvain communities, and VOS Clustering partitions? Let us use the partitions for the fifty-one vertices that are part of islands. To calculate the E-I Index, we need both a partition and a network, so we must first create a network of fifty-one vertices that matches the partitions. Select the one-mode network of the Scottish firms and the islands partition (both having 108 vertices). Extract the subnetwork belonging to islands (vertices for which values in the islands partition are larger than zero) by running *Operations> Network + Partition> Extract> SubNetwork Induced by Union of Selected Clusters* and entering 1-*. This will give us the network with fifty-one vertices that is compatible with the three partitions. The *E-I Index* is available in *Operations> Network + Partition > Info* submenu, where you can also ask for the *Modularity* and *VOS Quality* of the network. Ensure that the network with fifty-one vertices is selected in the *Network* drop-down menu and that one of the three partitions with fifty-one vertices is selected in the first *Partitions* drop-down menu, then run *E-I Index*. Answer *No* to the question "*Forget values on lines?*" because we want to include line values in the calculations. We get the following E-I indices: -0.55 for the Islands partition, -0.57 for Louvain communities, and -0.30 for VOS Clustering. All three indices are well below 0 but the values suggest that the Louvain communities (and Islands) identify cohesive groups better than VOS Clustering in this example. This need not be the case

in other examples. On the Pajek webpage, you can find several examples and comparisons of results obtained by the Louvain method and VOS Clustering. It is possible to refine the communities in steps. Detect communities with the default resolution parameter (1). Then extract each obtained community as a separate network. This can be done with a single command (*Extract > SubNetworks Induced by Each Selected Cluster Separately*), which is available as a submenu in *Operations > Network + Partition > Extract > SubNetworks Induced by Each Selected Cluster Separately*. For every community that you want to refine, that is, split into smaller communities, you can apply the community detection again to the extracted network for this community. Now use the same or a larger resolution value. You will learn how to run such repetitive operations very efficiently (using *Macro > Repeat Last Command*) in Chapter 13.

*Operations >
Network +
Partition >
Extract >
SubNetworks
Induced by
Each Selected
Cluster
Separately*

5.6 The Third Dimension

Can we model the network as a landscape in which the elevation of a point matches the value of its island? In principle, this can be done, but it involves techniques from geography that are not available in Pajek. We can, however, apply the principle of adding heights to points in a plane in different ways.

In previous chapters, we alluded to the possibility of drawing networks in three dimensions, but we restricted ourselves to two dimensions. The third dimension is called the z -axis, which points from the plane of the Draw screen toward the person in front of the computer monitor. If we use the island height of vertices as their scores on the z -axis, the highest island peaks out of the plane. If computer screens were flexible, a three-dimensional drawing of the islands in the network of Scottish firms would change its surface to a landscape and we would be able to feel islands with our fingertips.

As is, we must be satisfied with a faint sensation of depth, which is caused by the size of vertices and the darkness of vertex labels in a two-dimensional drawing. Nearby vertices are drawn larger, and distant vertex labels are gray rather than black. When we rotate the network, we get a better view of the landscape of islands (Figure 55), which is clearly dominated by the dark gray peak of the island at height 8.

The third dimension offers another opportunity to visualize social networks. Instead of using a predetermined set of values as z scores (heights) (e.g., heights of islands), we can energize a network in three dimensions, allowing the *Energy* procedure to locate vertices in a three-dimensional space to optimize the length of lines. Sometimes, a third dimension helps to detect patterns; for instance, a three-dimensional model of the Scottish firms networks separates the different islands better than a two-dimensional drawing, but the results are often disappointing. Our

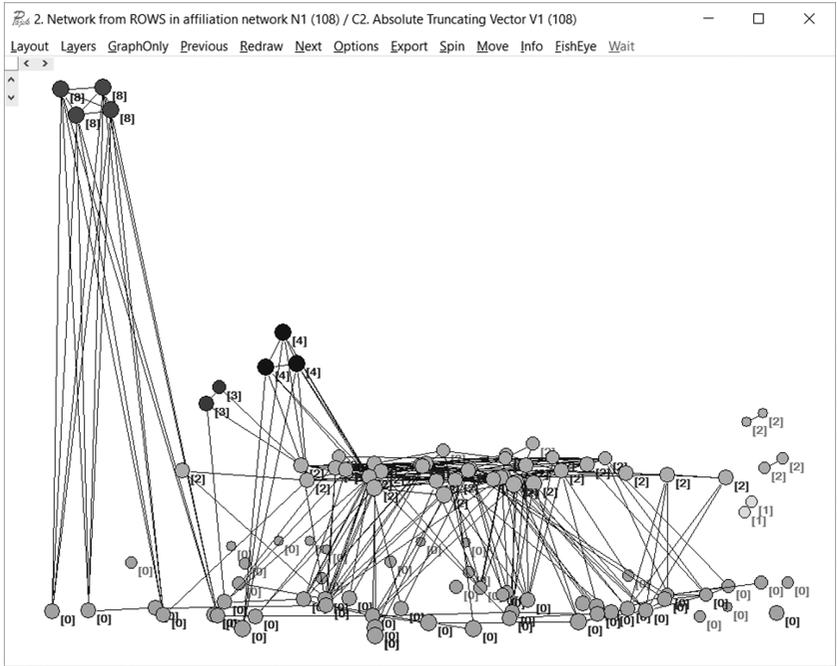


Figure 55. Islands in three dimensions.

graphical devices can handle two-dimensional representations much better than three-dimensional models.

Application

First, let us have a look at the coordinate system of Pajek (Figure 56). Imagine that the light gray square is the Draw screen. As we have seen before, the x value defines the horizontal location of a vertex (from left to right), and its y value specifies the vertical position (from top to bottom). The z value of a vertex defines its amount of protrusion from the

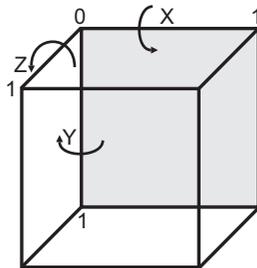


Figure 56. Coordinate system of Pajek.

background of the Draw screen. The arrows indicate the direction of positive rotations around the axes.

Pajek can only use a partition to lift vertices out of the plane of the screen, so we need to create a partition from the island height vector first. In our case, the heights are line multiplicities (number of shared board members), which are integers, so we can simply drop all decimals to obtain the integers we need for the partition [command *Vector> Make Partition> by Truncating (Abs)*]. Now draw the one-mode network of firms with the island height partition (*Draw> Network + First Partition*) and you will notice that the Draw screen contains a *Layers* menu. Choose option 3D from the *Type of Layout* submenu. The *Layers* menu now shows a command that displays the layers in the *z* direction. On execution of the *In z Direction* command, nothing seems to happen; but if you take a closer look, you will see that some vertices are drawn larger than others and some vertex labels are gray instead of black. You are looking at the peaks from above, which does not give much sense of relief.

When you rotate the structure, peaks and lowlands become more apparent. Toggle the *ScrollBar On/Off* option in the *Options* menu to add two scrollbars to the top left of the Draw screen. Press the buttons on the vertical scroll bar to rotate the network around the *x*-axis, which will raise or lower the peaks, and use the buttons on the horizontal scroll bar to rotate the network around the *y*-axis. Continue until you are satisfied with your view. Now you can see that the island at multiplicity 8 is towering high above the rest of the network. If the Draw screen is active, you can use the *x*, *y*, and *z* keys on your keyboard to spin the network around the *x*-, *y*-, and *z*-axes, respectively. Use the capitals *X*, *Y*, and *Z* for rotation in the opposite direction.

You can also rotate the three-dimensional structure in any direction you wish with the *Spin* menu, but this is slightly more complicated because you have to choose the axis of rotation (command *Normal*) and the angle of rotation, which you must enter in a dialog box displayed by the *Spin around* command. When you ask for a rotation higher than 360 degrees (or just press *s* or *S* if the Draw screen is active), you will see the network revolve before your eyes. If the rotation is too fast, lower the *Step in degrees* setting. This allows you to inspect the network from all angles.

As pointed out at the start of this section, Pajek cannot depict the islands as a landscape (Figure 57) or as a map with contours. The free software *R* can do this. It is easy to transfer data from Pajek to *R*, provided that you have downloaded (<http://www.r-project.org/>) and installed *R* on your computer. First you must tell Pajek where it can find *R* on your computer. Click the *Tools> R> Locate R* command and find the file *rgui.exe* or *rterm.exe*, which are located in the *bin* subdirectory of the folder where you installed *R*. Now Pajek is ready to transfer data to *R*.

*Vector> Make
Partition> by
Truncating
(Abs)*

*Layers> Type
of Layout*

*Layers> In z
Direction*

*Options>
ScrollBar
On/Off*

Spin menu

*Tools> R>
Locate R*

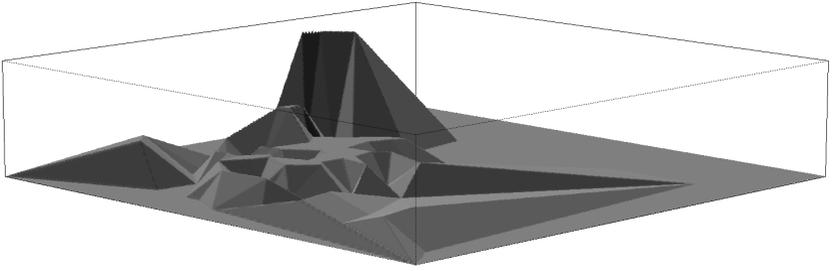


Figure 57. A landscape of islands in the Scottish firm's network.

Network>
Create Vector>
Get
Coordinate> x,
y

Tools> R>
Send to R> All
Vectors

R: File> Open
script

R: Edit> Run
all

Tools> Excel
Tools> Export
to Delimited
File

Layout>
Energy>
Fruchterman-
Reingold> 3D
Export> 3D>
X3D

What you need to produce a landscape like the one depicted in Figure 57 are the x - and y -coordinates of the vertices in the Draw screen and the island height vector. You already have the height vector and you can produce vectors with the coordinates with commands in the *Network> Create Vector> Get Coordinate* submenu. After doing this, send all vectors to R with the command *Tools> R> Send to R> All Vectors*. If R is correctly installed and made known to Pajek, the R software should start and read the vectors into R automatically.

With R's *File> Open script* command, open the script file `plot_landscape.R` that is available from the book's website (<http://mrvar.fdv.uni-lj.si/pajek>). In the script file, which is displayed in a separate window, make sure that under the line starting with `# USER:` the x , y , and island height vectors are correctly identified by the vector names (`v2`, `v3`, etc.) in your data. In addition, the `{akima}` package must be downloaded and installed, which is explained in the script file. Finally, use the *Edit> Run all* command in R to execute the script. This will produce a landscape like Figure 57 and a plot depicting the islands as a geographical map with contours.

If you are more familiar with Excel than with R you might do similar visualizations in Excel as well. Simply send one or more Pajek objects (e.g., Vectors) to Excel (*Tools> Excel> Send to Excel*), perform some further calculations in Excel, and draw different diagrams or charts. If you prefer some other statistical software you can also save Pajek objects into a Tab Delimited File (*Tools> Export to Delimited File*). Most programs for doing statistical analysis and visualization have an option to load data which is stored in a Tab Delimited File.

Three-dimensional optimization is accomplished with the command *3D* in the *Layout> Energy> Fruchterman-Reingold* submenu. For a better view, rotate the network with the scrollbars or the *Spin* commands. The sense of depth is much greater if the model is viewed in special 3D software. To accomplish this, Pajek can export the network to an X3D model: just select the *X3D* command in the *Export> 3D* menu of the

Draw screen. A dialog box asks for a name of the file that will contain the model. By default, the extension of this file is `.x3d`. X3D viewers recognize this extension, so do not change it. The X3D file can be displayed and manipulated in special programs like `instantplayer` (<http://www.instantreality.org/>) or even some Internet browsers (for details see Appendix 2). You can rotate and move through the structure as if it is part of a video game, but you need a fast computer and graphics card for smooth operations. The file `islands.x3d` (also available from the book's website), which was created by Pajek with the `Export> 3D> X3D` command, contains a model of the Scottish firms network with vertex colors indicating islands. Enjoy.

Exercise III

Create a three-dimensional energized drawing of the information network in San Juan Sur, which we analyzed in the assignment presented in Chapter 3 (`SanJuanSur_deathmessage.net`). Which family-friendship groupings (`SanJuanSur_deathmessage.clu`) are nicely clustered in this image?

5.7 Summary

Affiliation networks are typically two-mode networks, in which persons are related to organizations. The structure of these networks may be analyzed with standard network techniques, but several structural concepts have to be redefined or must be interpreted differently when applied to two-mode networks. Therefore, network analysts usually focus on the one-mode person-by-person or organization-by-organization network that can be induced from a two-mode affiliation network.

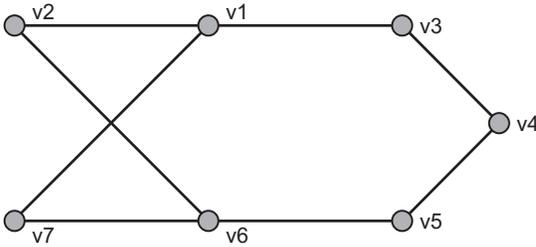
Induced one-mode networks tend to be rather dense, because all people affiliated with one organization are interrelated in the one-mode network of persons and all organizations that share a particular person are completely connected in the one-mode network of organizations. Researchers apply the techniques of overlapping cliques, islands, and communities to one-mode networks derived from affiliations. Overlapping cliques identify relatively dense sections of the network while islands and communities identify clusters of persons or organizations that are related by multiple lines (e.g., firms sharing a number of directors).

Finally, this chapter introduces three-dimensional displays. We can use the third dimension to represent predetermined values by layers (e.g., the height of islands), which turns a sociogram into a landscape, or we can use all three dimensions to energize a network. In subsequent chapters,

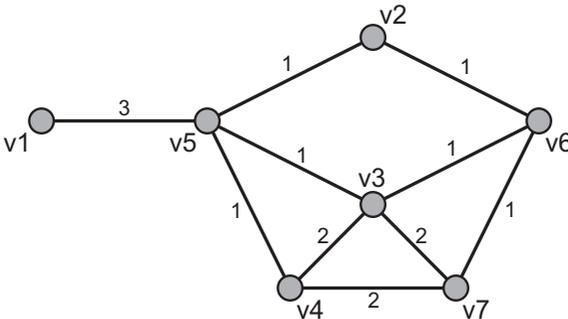
we encounter more applications of the third dimension but it should be noted that two-dimensional sociograms are often easier to interpret.

5.8 Questions

1. Could the sociogram depicted below represent a two-mode network? If so, show the way in which the vertices can be divided into two modes.



2. Add the names of the firms to the lines of the one-mode network of directors in Figure 52 (Section 5.3).
3. Which of the following statements is correct? Justify your choice.
 - a. Each affiliation network is a two-mode network.
 - b. Each two-mode network is an affiliation network.
4. What is the multiplicity of the tie between the two vertices marked by an X in the middle of Figure 53 (Section 5.4)?
 - a. We do not know.
 - b. 4
 - c. 5
 - d. 5 or higher
5. The following sociogram depicts a one-mode network of firms derived from affiliations between multiple directors and firms. Which of the following interpretations can be incorrect?



- a. Three directors sit on the boards of v1 and v5 simultaneously.
 - b. No director sits on all four boards of firms v2, v3, v5, and v6.
 - c. Two directors sit on the boards of v3, v4, and v7 simultaneously.
 - d. No director sits on the boards of v1 and v4 simultaneously.
6. Manually add contours to the islands in the sociogram of Question 5.
 7. Pajek has two options for extracting subnetworks in submenu *Operations> Network + Partition> Extract*:
 - *SubNetwork Induced by Union of Selected Clusters*
 - *SubNetworks Induced by Each Selected Cluster Separately*
 What is the difference? When should one use the first and when the second option?

5.9 Assignment

In Hollywood, composers of soundtracks work on a freelance basis. For each movie, a producer hires a composer and negotiates a fee. The earnings of composers are highly skewed: A handful of composers earn a lot, whereas most of them have moderate or low revenues. This is characteristic of artistic labor markets.

Why do some composers earn much more money than their colleagues in Hollywood? Let us assume that there are two hypotheses:

1. A successful composer works for the same producer(s) on a regular basis, whereas less successful composers do not.
2. The most successful composers all work for the top producers who are responsible for the most expensive movies.

The network `Movies.net` contains the collaboration of forty composers and the sixty-two producers who produced a minimum of five (completed, shown, and reviewed) movies in Hollywood, 1964–76. This is a two-mode network: A line between a composer and a producer indicates that the former created the soundtrack for the movie produced by the latter. The line values indicate the number of movies by one producer for which the composer created the music in the period 1964–76. The partition `Movies_top_composers.clu` identifies the five top composers, each of whom earned 1.5 percent or more of the total income of Hollywood movie score composers in the 1960s and 1970s.

Analyze the two-mode network in order to test hypothesis 1. Then, create a one-mode network of composers and see whether it corroborates or falsifies hypothesis 2.

5.10 Further Reading

- Georg Simmel stated his ideas about social circles in *Soziologie: Untersuchungen über die Formen der Vergesellschaftung* (Berlin:

- Duncker & Humblot, 1908), which was translated by Kurt H. Wolff as *The Sociology of Georg Simmel* (New York, NY: The Free Press, 1950). It contains the often-cited chapter “The web of group affiliations” (Chapter 6). Charles Kadushin used this concept and adapted it to network analysis in his book *The American Intellectual Elite* (Boston, MA: Little, Brown and Company, 1974), using the technique of overlapping cliques. Ronald Breiger wrote a seminal article on the social meaning of affiliations: “The duality of persons and groups.” *Social Forces* 53 (1974), 181–90.
- In the article “Analysing interlocking directorates: Theory and methods” (*Social Networks* 1 [1979], 1–36), Meindert Fennema and Huibert Schijf survey research on interlocking directorates. Our example is taken from John Scott and Michael Hughes, *The Anatomy of Scottish Capital: Scottish Companies and Scottish Capital, 1900–1979* (London: Croom Helm, 1980).
 - Stanley Wasserman and Katherine Faust discuss affiliation networks and some advanced techniques in Chapter 8 of their book, *Social Network Analysis: Methods and Applications* (Cambridge: Cambridge University Press, 1994).
 - Stephen Borgatti and Martin Everett’s “Network analysis of 2-mode data” (*Social Networks* 19 [1997] 243–69) discusses the translation of structural indices from one-mode to two-mode networks.
 - The data of the assignment are taken from Robert R. Faulkner, *Music on Demand: Composers and Careers in the Hollywood Film Industry* (New Brunswick, NJ: Transaction Books, 1983).
 - Louvain community detection method was introduced by V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre “Fast unfolding of communities in large networks” (*Theory and Experiment* 10 [2008]), VOS Clustering is explained in L. Waltman, N. J. Van Eck, and E. C. M. Noyon, “A unified approach to mapping and clustering of bibliometric networks” (*Journal of Informetrics* 4[2010], 629–35). The implementation of all methods in Pajek follows the instructions in R. Rotta and A. Noack, “Multi-level local search algorithms for modularity clustering” (*Journal of Experimental Algorithmics* 16 [2011]).

5.11 Answers

Answers to the Exercises

- I. Select the affiliation partition in the *Partition* drop-down menu before you draw the network. Note that the other partition

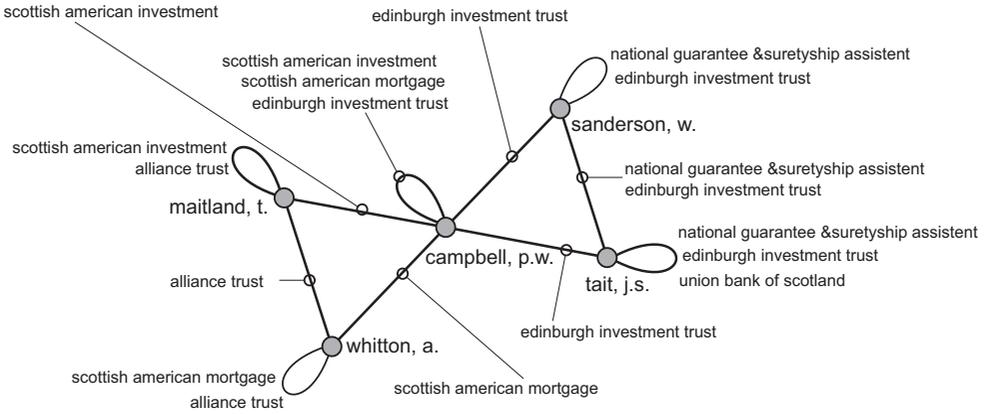
(`Industrial_categories.clu`) contains information about 108 vertices (the firms), whereas there are 244 vertices in the network. Therefore, the industrial categories partition cannot be drawn with the network.

In the circular layout of the network, you may notice that all vertices in the first class of the affiliation partition (yellow or black vertices) are grouped together. These are the firms. The second class (green or gray vertices) contains the directors.

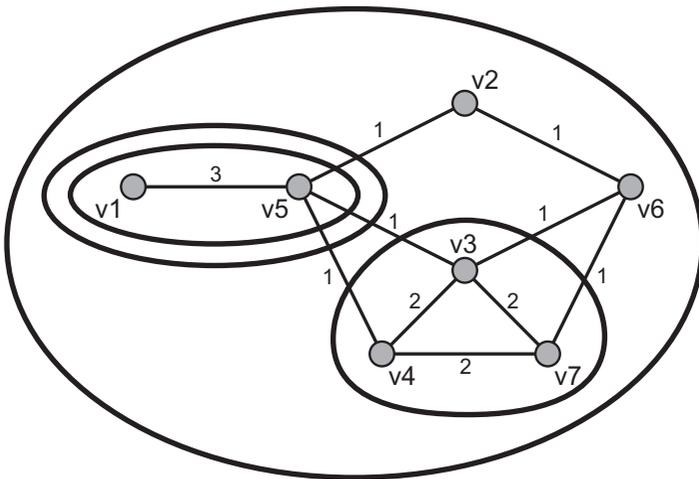
- II. As explained in the “Application” part of Section 5.3, transform the two-mode network to a one-mode network of firms with the `Network> 2-Mode Network> 2-Mode to 1-Mode> Rows` command. Check that the options *Include Loops* and *Multiple Lines* are not selected before you execute the `2-Mode to 1-Mode` command. To obtain a layout comparable to the one shown in Figure 53, make sure that line values are regarded as measures of similarity: Select the option `Options> Values of Lines> Similarities` in the Draw screen. Energy commands will now shorten lines with high multiplicity. Because the one-mode network is not connected, it is best to energize it with Fruchterman–Reingold first. Then, apply Kamada–Kawai (once or repeatedly) for refined results. Perhaps, your layout is reflected or rotated – never mind that. It is very likely that you will have to rearrange the isolates and small components by hand because the energy commands do not always set them apart nicely. It is easy to locate and drag the components if you create a partition according to components (minimum size 2) with the `Network> Create Partition> Components> Weak` command first.
- III. Energize the network with `Fruchterman–Reingold> 3D` and spin it around. You will see that all family–friendship groupings are clustered quite well in the three-dimensional model, with the exception of the light green family–friendship grouping consisting of families f5, f14, f17, f25, f58, and f69, which is like a thread through the middle of the sphere.

Answers to the Questions in Section 5.8

1. Yes, the sociogram may represent a two-mode network. The vertices can be divided into two modes such that all lines connect vertices from different modes (the network is bipartite): vertices v1, v4, and v6 would make up one mode and the remaining vertices the other. Vertices v1, v4, and v6 could, for example, represent firms and vertices v2, v3, v5, and v7 could be directors.
2. The sociogram should look like the following figure (do not forget the loops!).



3. Answer a is correct, because affiliations are ties between people and organizations or events, there must be two subsets that cannot be linked internally. Heterosexual love ties constitute a two-mode network but not an affiliation network, so answer b is not correct.
4. Answer c is correct. There are exactly five contours around the two vertices, so their multiplicity level is 5.
5. Interpretation c can be incorrect because it is possible that firms v3 and v4 share other directors than firms v3 and v7 and firms v7 and v4. The other interpretations can be correct.
6. Your answer should look like the sociogram that follows. Do not forget to draw two contours around the island at height 3 (left) and to include a contour for the island at height 1, that is, the component; otherwise the height does not correspond to the number of contours.



7. The command *SubNetwork Induced by Union of Selected Clusters* always extracts *only one* subnetwork, which contains all vertices (and lines among them) that are part of the selected clusters. On the other hand, the command *SubNetworks Induced by Each Selected Cluster Separately* extracts one subnetwork for each selected cluster. In this way, *several* subnetworks can be extracted using a single command.

Part III

Brokerage

In quite a few theories, social relations are considered channels that transport information, services, or goods between people or organizations. In this perspective, social structure helps to explain how information, goods, or even attitudes and behavior diffuse within a social system. Network analysis reveals social structure and helps to trace the routes that goods and information may follow. Some social structures permit rapid diffusion of information, whereas others contain sections that are difficult to reach.

This is a bird's-eye view of an entire social network. However, we can also focus on the position of specific people or organizations within the network. In general, being well connected is advantageous. Contacts are necessary to have access to information and help. The number and intensity of a person's ties are called his or her *sociability* or *social capital*, which is known to correlate positively with age and education in Western societies. Some people occupy central or strategic positions within the system of channels and are crucial for the transmission process. Such positions may put pressure on their occupants, but they may also yield power and profit.

In this part of the book, we focus on social networks as structures that allow for the exchange of information. In this approach, the direction of ties is not very important, so we discuss only undirected networks (with one exception). In Chapter 6, we present the concepts of centrality and centralization. In Chapter 7, we discuss the structure of the immediate network of actors, especially the pressure or power that is connected to particular structures of this ego network. In Chapter 8, we take time into account as we study the role of network structure in the diffusion of innovations and diseases.

Center and Periphery

6.1 Introduction

In this chapter, we present the concepts of centrality and centralization, which are two of the oldest concepts in network analysis. Most social networks contain people or organizations that are central. Because of their position, they have better access to information and better opportunities to spread information. This is known as the *ego-centered approach* to centrality. Viewed from a *sociocentered perspective*, the network as a whole is more or less centralized. Note that we use *centrality* to refer to positions of individual vertices within the network, whereas we use *centralization* to characterize an entire network. A network is highly centralized if there is a clear boundary between the center and the periphery. In a highly centralized network, information spreads easily but the center is indispensable for the transmission of information.

In this chapter, we discuss several ways of measuring the centrality of vertices and the centralization of networks. We confine our discussion of centrality to undirected networks because we assume that information may be exchanged both ways between people or organizations that are linked by a tie. Concepts related to importance in directed networks, notably prestige, are discussed in Part IV of this book.

6.2 Example

Studies of organizations often focus on informal communication: Who discusses work matters with whom and to whom do people turn for advice? Informal communication is important to the operation of the organization, and it does not always coincide with the organization's formal structure. For the diffusion and retrieval of information, it is crucial

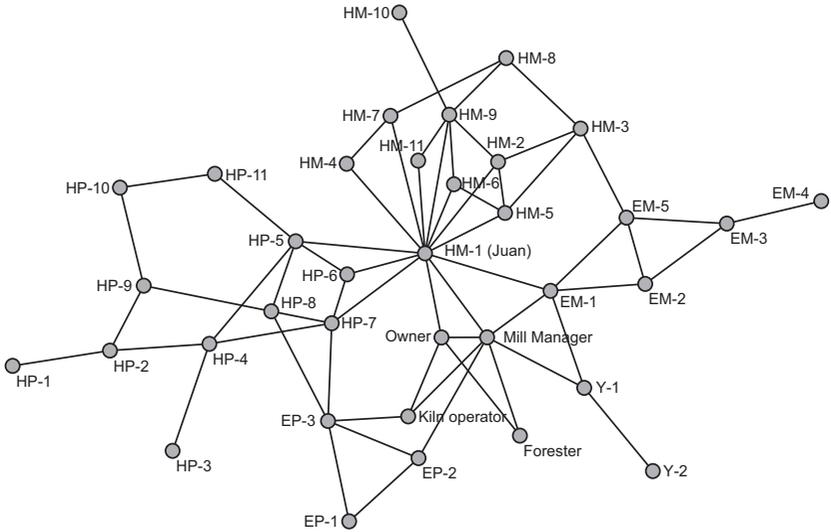


Figure 58. Communication ties within a sawmill.

to know the people who occupy central positions in the communication network.

Our example is a communication network within a small enterprise: a sawmill. All employees were asked to indicate the frequency with which they discussed work matters with each of their colleagues on a 5-point scale ranging from less than once a week to several times a day. Two employees were linked in the communication network if they rated their contact as three or more. We do not know whether both employees had to rate their tie in this way or that at least one employee had to indicate a strength of 3 or more. The network is stored in the file `Sawmill.net`.

In the sawmill, the employees are Spanish speaking (H) or English speaking (E), which, of course, is relevant to their communication. The sawmill contains two main sections: the mill (M), where tree trunks are sawn into logs, and the planer section (P), where logs are planed. Then there is a yard (Y) where two employees are working and some managers and additional officials.

Figure 58 shows the communication network in the sawmill. Note that vertex labels indicate the ethnicity and the type of work of each employee; for example, HP-10 is a Hispanic (H) working in the planer section (P). In this figure, vertex labels instead of vertex colors identify the attributes of employees. It is quite easy to see that work-related communication is structured along work section (planers at the left, sawyers at the right) and ethnicity: Hispanics at the top and English speakers at the bottom – assuming that management, forester, kiln operator, and employees in the

yard are English speakers. This is an example of homophily (Chapter 3), which is closely related to assortativity (Section 6.6).

Intuitively, HM-1 (Juan) is a central, perhaps the most central, person in this network. He communicates with many colleagues directly, and through his direct contacts it is easy for him to reach most of the people working in the sawmill. Juan seems to occupy a pivotal position in the flow of information between the planers, the mill section, and management. This chapter presents formal measures of centrality and centralization, which capture these intuitions.

6.3 Distance

One approach to centrality and centralization is based on the simple idea that information may easily reach people who are central in a communication network. Or, to reverse the argument, people are central if information may easily reach them.

The larger the number of sources accessible to a person, the easier it is to obtain information; for instance, an elderly person will acquire information about where to look for help more easily if his or her social support network is larger. In this sense, social ties constitute a *social capital* that may be used to mobilize social resources. Hence, the simplest indicator of centrality is the number of its *neighbors*, which is his or her degree in a simple undirected network (see Chapter 3). The higher the degree of a vertex, the more sources of information it has at its disposal, the quicker information will reach the vertex, so the more central it is. In the sawmill network, Juan communicates with no fewer than thirteen colleagues, whereas the manager of the mill has only seven communication ties (Figure 58). In this respect, Juan is more central than the manager, and information from the shop floor will reach him more easily than it will reach the manager. If degree is the simplest measure of the centrality of a vertex, what is the associated measure of centralization for the entire network that expresses the extent to which a network has a center? Let us first answer another, related question: Given a fixed number of lines, what is the most efficient structure to exchange information? We should note that this network must be connected; otherwise information cannot reach all vertices. In this case, the *star-network* is known to be the most efficient structure given a fixed number of lines. A star is a network in which one vertex is connected to all other vertices, but these vertices are not connected among themselves (e.g., network A in Figure 59).

Compare the star-network in Figure 59 to the line-network, containing the same number of vertices and lines (network B). It is much easier to identify the central vertex in the star-network than in the line-network because the difference between the central vertex (v_5) and the



Figure 59. Star-network and line-network.

peripheral vertices ($v_1, v_2, v_3,$ and v_4) is much more apparent than in the line-network. This leads to an idea that may be counterintuitive, namely, that a network is more centralized if the vertices vary more with respect to their centrality. More variation in the centrality scores of vertices yields a more centralized network.

Now we can define *degree centralization* as the variation in the degree of vertices divided by the maximum variation in degree that is possible given the number of vertices in the network. In a simple network of a particular size, the star-network has maximum degree variation. The division by maximum degree variation ensures that degree centralization ranges from 0 (no variation) to 1 (maximum variation) in the case of a star-network.

The *degree centrality* of a vertex is its degree.

Degree centralization of a network is the variation in the degrees of vertices divided by the maximum degree variation that is possible in a network of the same size.

Variation is the summed (absolute) differences between the centrality scores of the vertices and the maximum centrality score among them. In network A (Figure 59), for instance, one vertex (v_5) has degree 4, which is the maximum degree in a simple undirected network of this size because this vertex is connected to all other vertices. The other four vertices have minimum degree, which is 1 in a connected undirected network. Hence, the degree variation amounts to 12: (vertices v_1 to v_4 contribute) $4 \times (4 - 1)$ and (vertex v_5 contributes) $1 \times (4 - 4)$. In a simple undirected network, the degree of vertices cannot vary more than this, so 12 is the maximum variation; and dividing 12 by itself, of course, yields a degree centralization of 1.00.

In network B, two vertices have a degree of 1 (v_1 and v_2) and the other vertices have a degree of 2. Because 2 is the maximum degree in this network, the degree variation equals $2 \times (2 - 1)$ (for vertices v_1, v_2) and

$3 \times (2 - 2)$ (for vertices v_3 to v_5), which is 2. To obtain the degree centralization of network B, we divide 2 by 12, which is the maximum variation in a simple undirected network, and we obtain 0.17. If we add a line between v_1 and v_2 , degree centralization becomes minimal (0.00) because all vertices have equal degree, so variation in degree is 0.00 and degree centralization is 0.00.

We should issue a warning here. In a network with multiple lines or loops, the degree of a vertex is not equal to the number of its neighbors. Therefore, the star-network does not necessarily have maximum variation, and we may obtain centralization scores of more than 1.00 if we compare the variation in a network with multiple lines or loops to the variation in a simple star-network of the same size. In this case, we advise not using degree centralization.

In a simple undirected network, degree centrality is just the number of neighbors of a vertex. In some cases, this is all we know about the network position of people, for instance, when data are collected by means of a survey in which people are asked to indicate the size of their personal network. If we want to analyze the communication structure of the network, however, we need to know who is connected to whom in the entire network; and we must pay attention to indirect ties because information can flow from one person to the next and on to other people. In a communication network, information will reach a person more easily if it does not have to “travel a long way.” This brings us to the concept of *distance* in networks, namely, the number of steps or intermediaries needed for someone to reach another person in the network. The shorter the distance between vertices, the easier it is to exchange information.

In Chapter 3, we defined paths as a sequence of lines in which no vertex in between the first and last vertices occurs more than once. Via a path, we can reach another person in the network: We can inform our neighbor, who passes the information on to his neighbor, who in turn passes it on, until the information finally reaches its destination. We say that a person is *reachable* from another person if there is a path from the latter to the former. Note that two persons are mutually reachable if they are connected by a path in an undirected network, but that two paths (one in each direction) are needed in a directed network.

In an undirected network, the distance between two vertices is simply the number of lines or steps in the shortest path that connects the vertices. A shortest path is also called a *geodesic*. In a directed network, the geodesic from one person to another is different from the geodesic in the reverse direction, so the distances may be different. This sounds strange if you are used to geographic distances, but think of a directed network as a system of one-way streets: It is easy to imagine that the route from A to B differs from the journey back. In this chapter, however, we use only undirected networks, so you do not have to worry about this now.

A *geodesic* is the shortest path between two vertices.

The *distance* from vertex u to vertex v is the length of the geodesic from u to v .

Distance is important in social network analysis. Recall the Small World problem (Section 1.3), which states that network distance between all people is very low, claiming an average of about six. With the concept of distance, we can also define another index of centrality, which is called *closeness centrality*. The closeness centrality of a vertex is based on the total distance between one vertex and all other vertices, where larger distances yield lower closeness centrality scores. The closer a vertex is to all other vertices, the easier information may reach it and the higher its centrality.

Just like degree centralization, we can conceptualize closeness centralization as the amount of variation in the closeness centrality scores of the vertices. Again, we compare the variation in centrality scores to the maximum variation possible, that is, the variation in closeness centrality in a star-network of the same size.

The *closeness centrality* of a vertex is the number of other vertices divided by the sum of all distances between the vertex and all others.

Closeness centralization is the variation in the closeness centrality of vertices divided by the maximum variation in closeness centrality scores possible in a network of the same size.

In star-network A (Figure 59), vertex v_5 has maximum closeness centrality because it is directly linked to all other vertices. The sum of distances to the other vertices is minimal, namely, four geodesics of length 1 combine into a summed distance of 4. Because there are four vertices other than v_5 , the closeness centrality of vertex v_5 is maximal: $4/4 = 1.00$. The other vertices of network A have a closeness centrality score that is considerably lower (0.57) because three vertices are two steps away from them.

In network B, v_5 also has the highest closeness centrality because it is in the middle, but now its closeness centrality is not maximal (0.67) and it differs less from the other vertices, which have closeness centrality 0.57 (vertices v_3 and v_4) and 0.40 (v_1 and v_2). Because the variation of closeness centrality scores in network B is less than in network A, network B is less centralized. Its closeness centralization is 0.42, whereas the maximum centralization of network A is 1.00.

Note that complications arise if the network is not (strongly) connected. If an undirected network is not connected or a directed network is not strongly connected, there are no paths between all vertices, so it is impossible to compute the distances between some vertices. The solution to this problem is to take into account only the vertices that are reachable to or from the vertex for which we want to calculate closeness centrality and weight the summed distance by the percentage of vertices that are reachable. This solution works fine for the closeness centrality of vertices. However, it does not allow us to compute the closeness centralization of the entire network because the star-network does not necessarily have the highest variation in closeness centrality scores if the network is not (strongly) connected. Therefore, we do not use closeness centralization in the case of a network that is not (strongly) connected.

Application

In Chapter 3, we explained how to compute the degree of vertices in Pajek. Note that the *Network> Create Partition> Degree> All* command counts edges only once, which is fine if the network is undirected. In a network containing edges and arcs, however, you may want to count the edges as incoming and outgoing arcs. If so, replace the edges with bidirectional arcs (*Network> Create New Network> Transform> Edges→Arcs*) before you calculate degree with the *All* command. In addition, we advise removing multiple lines (*Network> Create New Network> Transform> Remove> Multiple Lines*) and loops (*Network> Create New Network> Transform> Remove> Loops*) from the network before you compute degree centrality and degree centralization. The degree partition tells us the degree centrality of all vertices. For degree centralization, we have to calculate the variation of degree scores. In Pajek philosophy, partitions are classifications assigning vertices to clusters. The cluster numbers should not be used for calculations; vectors should be used for calculations. For this reason, degree centralization of a network is not reported together with the degree partition. We must use the *Create Vector> Centrality> Degree* submenu to obtain degree centralization in the Report screen. This command stores degree centrality scores of vertices as a vector.

Note that the degree centralization is reported only if the network contains no multiple lines and loops. Otherwise Pajek reports a message explaining that degree centralization is not valid for this network. In the sawmill network, the degree centralization is 0.289. This centralization value is meaningful only in comparison to other networks.

If you want to know the distance between one vertex (e.g., Juan) and all other vertices in the network, you can use the commands in the *Network> Create Partition> k-Neighbours* submenu, which create a partition of

Network>
Create
Partition>
Degree

Network>
Create New
Network>
Transform>
Edges→Arcs

Network>
Create New
Network>
Transform>
Remove>
Multiple Lines

Network>
Create New
Network>
Transform>
Remove>
Loops

Network>
Create Vector>
Centrality>
Degree

Network>
Create
Partition>
k-Neighbours

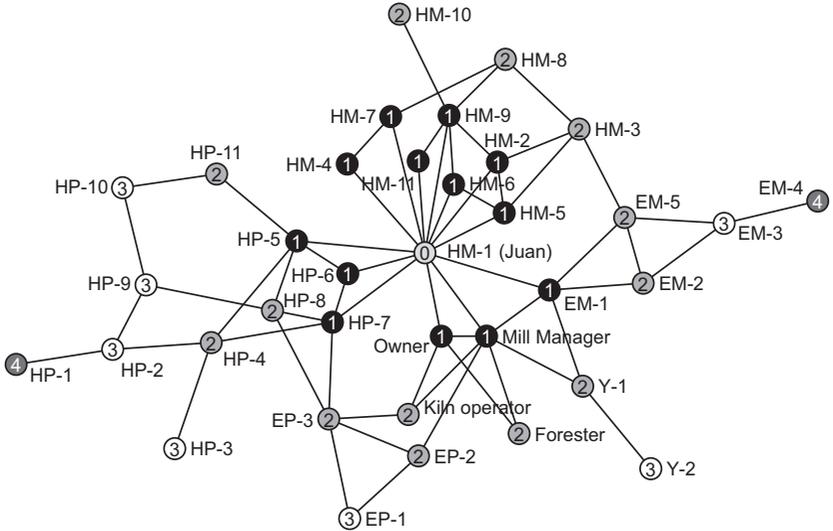


Figure 60. Distances to or from Juan (vertex colors: Default Grey-Scale 1).

classes containing the distances between one vertex and all other vertices. The *Input* command calculates the distances *to* the selected vertex, whereas the *Output* command computes distances *from* the vertex. The *All* command disregards the direction of the lines. In an undirected network, you may choose the *Input*, *Output*, or *All* commands: They yield the same results.

When you execute a *k-Neighbours* command, you must first specify the vertex number or the label of the vertex from which distances will be computed. In the case of Juan, enter 12 (his vertex number) or HM-1 (the start of his vertex label). Next, you can set a limit to the maximum distance that will be computed. In very large networks, setting a limit may speed up computation considerably. In this dialog box, 0 means that you want all distances, which is usually the right choice in the case of a small network. The distances are stored in a partition; and unreachable vertices or vertices further away than maximum distance are placed in class number 999999998, which indicates that their distance is not known.

In Figure 60, vertex colors and class numbers indicate the distances between Juan and other employees. Most employees are directly connected to Juan (black or yellow) or indirectly connected with one intermediary (light gray or green, distance 2). Two employees are four steps away from Juan, namely, HP-1 and EM-4.

Employees HP-1 and EM-4 seem to be farthest apart in the communication network because their distance to Juan is 4. However, their geodesic

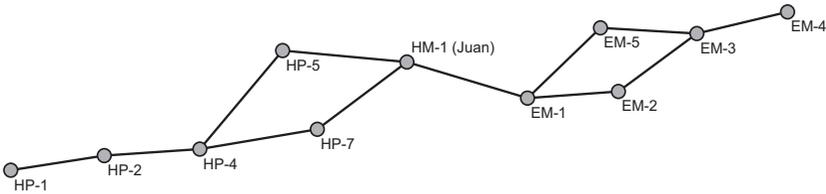


Figure 61. Geodesics between HP-1 and EM-4.

does not necessarily include Juan, so they may be connected in fewer than eight steps. In Pajek, the geodesics between two vertices can be found with the command *Network> Create New Network> SubNetwork with Paths> All Shortest Paths between Two Vertices*. First enter the vertex number or label of HP-1 and then enter EM-4 and subsequently answer *Yes* to the question “Forget values of lines?” because you do not want to weight the lines by their values. This is the right thing to do unless line values indicate distances, for instance, geographical distances. Finally, a dialog box asks whether the paths must be identified in the source network. If you answer *Yes* to this question, Pajek produces a partition for the original network that assigns vertices on the geodesics to class 1 and other vertices to class 0. Regardless of your choice in this dialog box, Pajek creates a new network with the vertices and lines that constitute the geodesics (Figure 61). In addition, it prints the distance in the Report screen. In our example, all geodesics between HP-1 and EM-4 include Juan (see Figure 61), so the distance between HP-1 and EM-4 cannot be less than 8, which is the sum of their distances to Juan.

```
Network>
Create New
Network>
SubNetwork
with Paths> All
Shortest Paths
between Two
Vertices
```

For a general description of a network, it is often useful to have a frequency distribution of the distances between all pairs of vertices. Is a network characterized by short geodesics or rather by long ones? The command *Network> Create Vector> Distribution of Distances** reports the average and maximum distance in the network and it produces a vector containing the distance distribution. Note that this is not an ordinary vector with one entry for each vertex in the network. Instead, it contains one entry for each distance and the vector value specifies the number of geodesics having this length in the network. Edit the distance distribution vector (*File> Vector> View/Edit*) to inspect the distance distribution in the sawmill network: 124 pairs of vertices are connected by a path of length one, 308 pairs are linked by a path of length two, and so on.

```
Network>
Create Vector>
Distribution of
Distances*
```

In Pajek, the computation of closeness centrality is straightforward. Because closeness centrality scores are continuous rather than discrete, the centrality commands are located in the *Network> Create Vector* submenu. The *Network> Create Vector> Centrality* submenu has commands to compute closeness centrality for all vertices in the network. For undirected networks, you may choose the commands *Input*, *Output*, or *All*,

```
Network>
Create Vector>
Centrality>
Closeness
```

which yield the same results. If the network is not (strongly) connected, Pajek creates a vector with closeness centrality scores but it does not compute closeness centralization, which is undefined in such a network. Closeness centrality of vertices that are not reachable to or from all other vertices is set to 0. For medium-sized and large networks, closeness centrality demands a lot of computing time, so it should be applied with care.

Pajek creates a vector with the closeness centrality scores of the vertices. You may inspect this vector or use it for computations in the ways explained in previous chapters. In our example, closeness centrality scores range from 0.20 to 0.51 and Juan (0.51) turns out to be more central than the manager (0.42). In addition, Pajek computes the closeness centralization of the network, which is printed in the Report screen. The sawmill communication network has a closeness centralization score of 0.38, which, again, must be interpreted in comparison to other networks.

Exercise I

What will happen to the network if Juan (HM-1) disappears? Remove the vertex, compute closeness centrality and centralization, and interpret the results.

6.4 Betweenness

Degree and closeness centrality are based on the reachability of a person within a network: How easily can information reach a person? A second approach to centrality and centralization rests on the idea that a person is more central if he or she is more important as an intermediary in the communication network. How crucial is a person to the transmission of information through a network? How many flows of information are disrupted or must make longer detours if a person stops passing on information or disappears from the network? To what extent may a person control the flow of information due to his or her position in the communication network?

This approach is based on the concept of *betweenness*. The centrality of a person depends on the extent to which he or she is needed as a link in the chains of contacts that facilitate the spread of information within the network. The more a person is a go-between, the more central his or her position in the network. If we consider the geodesics to be the most likely channels for transporting information between actors, an actor who is situated on the geodesics between many pairs of vertices is very important to the flow of information within the network. This actor is more central.

Juan, for instance, is important to the communication between HP-1 and EM-4 in the sawmill because all (four) geodesics include Juan (Figure 61). In contrast, HP-5 and HP-7 or EM-2 and EM-5 are less important

because if one fails to pass on information, the other may fulfill this role and the communication chain between HP-1 and EM-4 is still intact.

Each pair of vertices may contribute to the betweenness centrality of a vertex. HP-5 and EM-1, for example, contribute to the betweenness centrality of Juan, because their geodesic includes Juan. In contrast, the pair HP-4 and HP-5 does not contribute to Juan's betweenness centrality, because he is not included in their geodesic. In general, we may say that the *betweenness centrality* of a vertex is the proportion of all geodesics between other vertices in the network that include this vertex. *Betweenness centralization* is the ratio of the variation in betweenness centrality scores to the maximum variation.

The *betweenness centrality* of a vertex is the proportion of all geodesics between pairs of other vertices that include this vertex.

Betweenness centralization is the variation in the betweenness centrality of vertices divided by the maximum variation in betweenness centrality scores possible in a network of the same size.

It is easy to see that the center of a star-network (vertex v5 in Figure 59) has maximum betweenness centrality: All geodesics between pairs of other vertices include this vertex. In contrast, all other vertices have minimum betweenness centrality (0) because they are not located between other vertices. The centrality scores of vertices in a star have maximum variation, so the betweenness centralization of the star is maximal: Remove its central vertex and all communication ties are destroyed. In the line-network (B in Figure 59), removal of a vertex may also break the flow of information, but parts of the chain remain intact. Therefore, centrality indices vary less than in the star-network, and betweenness centralization is lower.

Application

The *Betweenness* command in the *Network> Create Vector> Centrality* submenu creates a vector of betweenness centrality scores for the vertices in the network. In addition, the betweenness centralization of the network is printed in the Report screen. In directed networks, the procedure automatically searches for directed paths, so there are no separate commands for *Input*, *Output*, and *All*. Even in unconnected networks, betweenness centrality can be computed.

Network>
Create Vector>
Centrality>
Betweenness

The sawmill communication network has a betweenness centralization of 0.55. Betweenness centrality scores of employees range from 0.00 to 0.59. In Figure 62, vertex size indicates betweenness centrality.

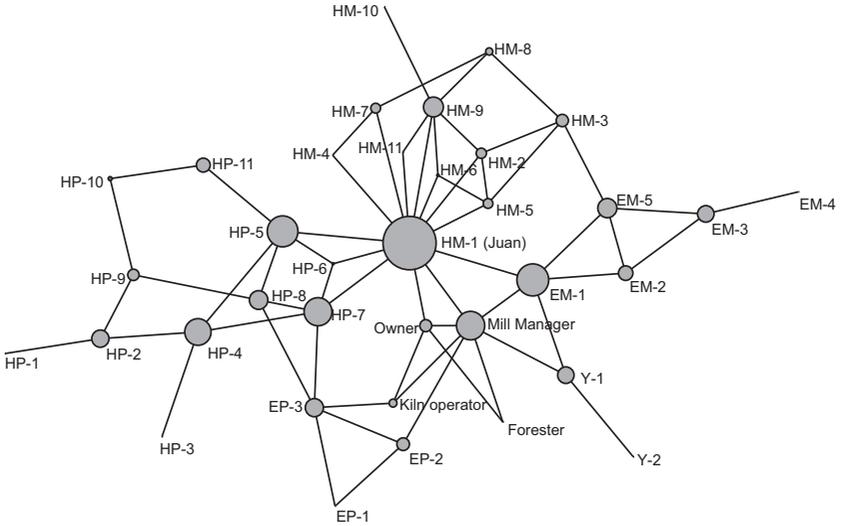


Figure 62. Betweenness centrality in the sawmill.

Several vertices are invisible because they have zero betweenness centrality: They do not mediate between other vertices. In this example, the betweenness centrality of vertices varies more than their closeness centrality because vertices at the outer margin of the network have zero betweenness, whereas they are still close to part of the network. As a consequence, betweenness centralization is higher than closeness centralization.

It is interesting to note that Juan (0.59), EM-1 (0.21), and HP-5 (0.20) are more central than the manager of the mill (0.17). Each ethnic group within the mill's departments – with the exception of the English-speaking planers – seems to have an informal spokesperson who is taking care of the communication with other departments or ethnic groups. Juan, who is the spokesman of the Hispanic employees at the mill, is clearly most central.

Exercise II

Compute betweenness centrality on the sawmill network, and draw it with vertex sizes corresponding to the betweenness centrality like Figure 62.

6.5 Eigenvector Centrality

In addition to degree, closeness, and betweenness, there is a fourth perspective on centrality. The basic assumption here is that you are more

central if you have more contacts – as in degree centrality – and especially if your contacts are more central, that is, if they have many central contacts. It is important to know people but it matters who you know. If you know influential people, you are more likely to exert influence through them. The direction of lines is irrelevant here, so this type of centrality is applied only to undirected networks.

As you may have noted from the description of this type of centrality, there is a circularity inherent in this approach: To calculate the centrality of a vertex, you must first calculate the centrality of its neighbors, which can be calculated only from the centrality of their neighbors, including the vertex we started with. There is, however, an elegant arithmetic solution to this problem. If we represent the network as a matrix (see Section 12.2), the first normalized eigenvector calculated on this matrix yields the sought-after centrality scores. For this reason, this type of centrality is called eigenvector centrality, even though it may not always be calculated by means of eigenvalue analysis.

The *eigenvector centrality* of a vertex is the extent to which it is linked to vertices with high eigenvector centrality.

Eigenvector centralization is the variation in the eigenvector centrality of vertices divided by the maximum variation in eigenvector centrality scores possible in a network of the same size.

Application

The Pajek command *Hubs-Authorities* applies the principle that the importance of a vertex is determined by the importance of its network neighbors to a directed network. It distinguishes between hubs, vertices that are important senders (connected with important authorities), and authorities, vertices that are important receivers (connected with important hubs). Pajek calculates a hub and an authority weight for each vertex and stores it as vectors. In addition, Pajek creates a partition with the highest scoring authorities (class 1), the top hubs (class 3), and vertices that are both a top hub and top authority (class 2). The user has to determine the number of authorities and hubs to be selected but these choices do not affect the calculation of the weights, so all choices are safe. The last two entries in the dialog box (“*adding loops to a network*” and “*selecting number of valid decimal places in computations*”) are not important for our purposes, so they do not need to be changed.

If applied to an undirected network, there is no distinction between hubs and authorities, so the hub and authority weight vectors are identical and they are the same as the eigenvector centrality scores of the vertices. In short, both vectors created by the *Hubs-Authorities* command on an

Network>
Create Vector>
Centrality>
Hubs-
Authorities

undirected network contain the eigenvector centrality scores. In the sawmill network, Juan has the highest eigenvector centrality (.54), followed by the mill owner with a score of .28. The Eigenvector centralization of the sawmill network, printed in the Report screen, is 0.72.

6.6 Assortativity

The matter of the centrality of your contacts, which is central to Eigenvector centrality (Section 6.5), brings us to another interesting phenomenon of social networks (see the paper by M. E. J. Newman in Further Reading): People with high degree tend to be linked to people with high degree whereas people with low degree tend to be linked to people with low degree. This phenomenon is called *degree assortativity*.

Degree assortativity is a special case of assortativity or *assortative mixing*, namely, a preference to relate to similar others. Assortativity resembles homophily – “Birds of a feather flock together” – and the two concepts are sometimes used as if they have the same meaning. In this book, however, assortativity refers to similarity with respect to a numeric property of the vertices such as a person’s network degree, age, or weight. We reserve homophily for similarity with respect to a categorical property of vertices such as gender, race, or social class. Similarity, that is, association, between numeric variables requires other measures than similarity between categorical variables.

Assortativity is a preference of vertices to attach to other vertices which are similar to them according to a numeric property.

Assortativity coefficient is the correlation between one or two numeric properties of vertices that are directly connected.

Assortativity in a network is measured by the *assortativity* coefficient, which is the Pearson correlation between a numeric property of the vertices, for example their degree, and a numeric property of the vertices to which they are directly connected. The maximum value is 1, indicating that vertices with high scores are linked to vertices with high scores whereas vertices with low scores are linked with other vertices with low scores. The minimum value is -1 , indicating that high degree vertices are linked to low-degree vertices. A negative assortativity coefficient indicates *disassortativity*. Disassortative behavior has been observed, for example, in animal networks with respect to body size. Small animals want to attach to big animals expecting that they will protect them, while big animals enjoy being powerful to control small animals. In biology such close

and long-term interaction between two different species is called symbiosis (see Further Reading). If degree assortativity is 0, the degree of a vertex does not tell us anything about the degree of its contacts.

In an undirected network, we cannot distinguish between the two vertices that are linked. As a consequence, the assortativity coefficient must use the same numeric property for both vertices. In contrast, we can distinguish between the sending and receiving vertex in a directed network. We may theorize that different properties of the sender and receiver are relevant to a link. In a heterosexual marriage system in which the bridegroom's family must ask the bride's family for a match, the number of sons is probably an important property of the sender whereas the number of daughters is an important property of the receiver.

Application

The degree assortativity coefficient is a characteristic of a network, so the command is found in the *Network* menu: *Network*> *Info*> *Degree Assortativity*. The suboptions *Input-Input*, *Input-Output*, *Output-Input*, *Output-Output* are relevant only for directed networks. *Input-Output*, for example, calculates the correlation between the indegree of the sending vertex and the outdegree of the receiving vertex. In an undirected network, such as the Sawmill network, indegree equals outdegree and we cannot distinguish between sender and receiver, so all four possible suboptions yield the same results. Any suboption is fine. Degree assortativity coefficient for the Sawmill network is -0.07 , which is close to zero, suggesting that the degree of an employee is not an important factor when selecting discussion partners.

Network>
Info> *Degree*
Assortativity

Calculation of the assortativity coefficient, other than degree assortativity, requires both a network and a numerical property of the vertices, which is stored as a vector. Let us turn to the example of Chapter 5, the two-mode network of directors and firms. Open the Pajek project file *Scotland.paj*, transform the two-mode network *Scotland.net* into a one-mode network of firms (*Network*> *2-Mode Network*> *2-Mode to 1-Mode*> *Rows* with the option *Multiple Lines* not selected). Select the vector containing the firm's capital (*Capital.vec*) in the first *Vector* dropdown list. The one-mode network of firms is undirected, so we must use the same numeric property for both vertices that are linked and one vector suffices. Run the command *Operations*> *Network + Vector*> *Info*> *Assortativity* and the Report screen will show that the assortativity coefficient is -0.08 . Assortativity is negative but quite near zero, so firms with large capital tend to share directors with firms with little rather than large capital. Note that the assortativity coefficient does not take into account the line values.

Operations>
Network +
Vector> *Info*>
Assortativity

In a directed network, the assortativity coefficient is requested in a similar way. Let us use the imports of manufactures network

(open the example of Chapter 2: `World_trade.paj`). Do countries import manufactures from countries with a similar Gross Domestic Product (GDP) in 1995 (vector `GDP_1995.vec`)? Select the network `Imports_manufactures.net` in the *Network* dropdown list, select `GDP_1995.vec` in the first *Vector* dropdown list and in the second *Vector* dropdown list or ensure that the second dropdown list is empty. Now, the assortativity coefficient (*Operations* > *Network* + *Vector* > *Info* > *Assortativity*) calculates the correlation between GDP of directly linked countries. Its value is 0.16, so countries tend to import from countries with a similar GDP.

If the first and second *Vector* dropdown list contains a vector, the first vector is used as the numeric property of the sending vertex and the second vector is used as the property of the receiving vertex. Although it does not make sense, let us find out whether the countries with large GDP import manufactures from countries with large populations. The sender is the exporting country, so we must select the population size vector (`Population_1995.vec`) in the first *Vector* dropdown list. The receiver is the importing country, so we must select the GDP vector in the second *Vector* dropdown list. The assortativity coefficient is around -0.03 , so rich countries import manufactures from smaller countries rather than the other way around.

The assortativity coefficient requires a numeric property of both linked vertices but there is one exception. A categorical property with only two categories (a dichotomy) can also be used. As an example we use the language vector (`language.vec`) that accompanies the Sawmill network. This vector contains two values that indicate the main language of the workers: 1 – English and 2 – Hispanic. The assortativity coefficient for language in the Sawmill network is very high, namely 0.79. We can conclude that membership in ethnic group is strongly related to the selection of colleagues with whom the workers discuss work matters. Note that you can use any pair of numbers to code the two categories. Pajek interprets a vector with exactly two different numbers as a dichotomy or dummy variable. In contrast, a vector with three or more different numbers is treated as a truly numeric property of the vertices. Don't use the assortativity coefficient if you have more than two categories; use the External-Internal (E-I) Index (Section 5.5) instead.

6.7 Summary

The concepts of vertex centrality and network centralization are best understood by considering undirected communication networks. If social relations are channels that transmit information between people, central people are those who either have quick access to information

circulating in the network or who may control the circulation of information.

The accessibility of information is linked to the concept of distance: If you are closer to the other people in the network, the paths that information has to follow to reach you are shorter, so it is easier for you to acquire information. If we take into account direct neighbors only, the number of neighbors (the degree of a vertex in a simple undirected network) is a simple measure of centrality. Eigenvector centrality extends this with the importance of the neighbors. If we also want to consider indirect contacts, we use closeness centrality, which measures our distance to all other vertices in the network. The closeness centrality of a vertex is higher if the total distance to all other vertices is shorter.

The importance of a vertex to the circulation of information is captured by the concept of betweenness centrality. In this perspective, a person is more central if he or she is a link in more information chains between other people in the network. High betweenness centrality indicates that a person is an important intermediary in the communication network. Information chains are represented by geodesics, and the betweenness centrality of a vertex is simply the proportion of geodesics between pairs of other vertices that include the vertex.

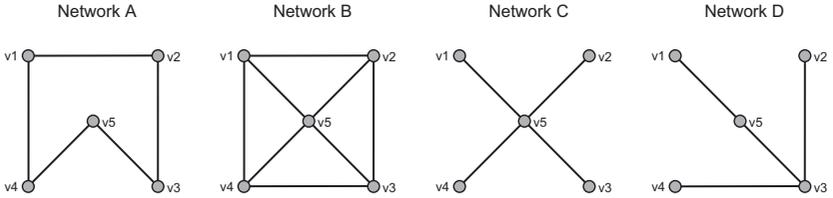
The centralization of a network is higher if it contains very central vertices as well as very peripheral vertices. Network centralization can be computed from the centrality scores of the vertices within the network: More variation in centrality scores means a more centralized network. There is an index of network centralization for each measure of centrality, but some centralization measures need special networks: Degree centralization is applicable only to networks without multiple lines and loops, and closeness centralization requires a (strongly) connected network.

In this book, we apply centrality and centralization only to undirected networks. It is easy to devise centrality measures for directed networks. We could base degree centrality on the outdegree of vertices, compute closeness centrality from the distances from a vertex to all other vertices (and not in the reverse direction), and consider only shortest directed paths in the case of betweenness centrality. In fact, other books on social network analysis advocate such an approach. We think, however, that it is conceptually clearer to restrict centrality and centralization to undirected networks and to apply other concepts (e.g., prestige) to directed networks.

6.8 Questions

1. Which of the following statements is correct?
 - a. The centrality of a network equals the degree of its vertices.
 - b. Centralization depends on the variation of centrality scores.

- c. A single vertex is always the center of a network.
 - d. The center of a network is always a cohesive subgroup.
2. Put these four networks in order of ascending centralization.



- a. B, A, C, D
 - b. A, D, B, C
 - c. D, A, B, C
 - d. A, B, D, C
3. Manually compute the closeness centrality of vertices $v1$ and $v3$ in network D of Question 2.
4. The betweenness centrality of vertex $v3$ in network D of Question 2 is 0.83. List the geodesics that include $v3$ and the geodesics that do not.
5. The file `question5.net` contains the nominations made by thirty-two employees of an organization who were asked to name the colleagues with whom they discuss work matters. Note that not all nominations are reciprocated. Omit the unilateral nominations, which are less reliable, from the network and find out who is most central in this communication network.

6.9 Assignment

In the 1970s, Rogers and Kincaid studied the diffusion of family planning methods in twenty-four villages in the Republic of Korea. The files `Korea1.net` and `Korea2.net` contain the communication networks among women in two villages: a village with a successful family planning program (`Korea1.net`) and a village in which family planning was not adopted widely (`Korea2.net`). In both networks, a line indicates that two women discussed family planning. In addition, we know which women adopted family planning methods at least temporarily (class 1 in the partitions `Korea1_adopters.clu` and `Korea2_adopters.clu`) and which women were members of the local Mothers' Club, which played an important role in the diffusion of family planning methods (class 1 in the partitions `Korea1_members.clu` and `Korea2_members.clu`) in both networks. The project file `Korea.paj` contains all files.

Analyze the networks and find whether centrality and centralization are associated with the success of the family planning program in one

village and its relative failure in the other village. Explain the effects of centrality and centralization by discussing the role of communication in the adoption of family planning methods.

6.10 Further Reading

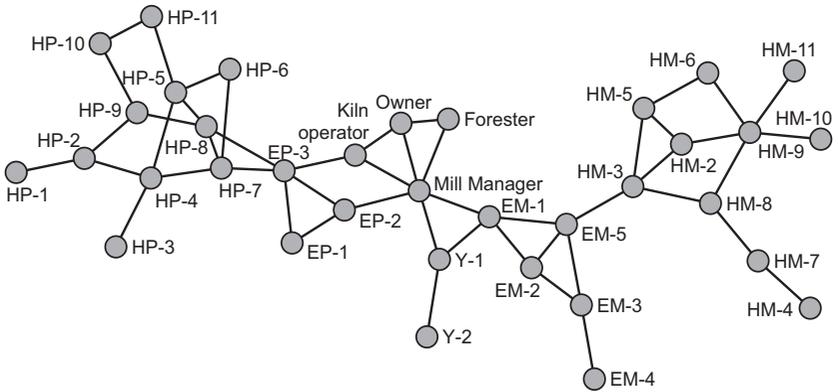
- The sawmill example is taken from J. H. Michael and J. G. Massey, “Modeling the communication network in a sawmill.” *Forest Products Journal* 47 (1997), 25–30.
- The data on the Korean villages stem from E. M. Rogers and D. L. Kincaid, *Communication Networks: Toward a New Paradigm for Research* (New York, NY: The Free Press, 1981), which offers an overview over network analysis from the perspective of communication studies. Note that some of the methods and software packages discussed in the book are obsolete.
- Many more measures of centrality have been proposed, notably information centrality, which considers all paths and not just geodesics in computing a betweenness score. Read more about these and other measures of centrality in A. Degenne and M. Forsé, *Introducing Social Networks* (London: SAGE, 1999, Chapter 6); J. Scott, *Social Network Analysis: A Handbook* (London: SAGE [4th edn. 2017], 1991, Chapter 6); and S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications* (Cambridge: Cambridge University Press, 1994, Chapter 5).
- The seminal paper on centrality is Linton Freeman’s “Centrality in social networks conceptual clarification.” *Social Networks* 1 (1979), 215–39.
- The method for calculating closeness centrality on networks that are not (strongly) connected was proposed by G. Sabidussi in “The centrality index of a graph.” *Psychometrika* 31 (1966), 581–603.
- Eigenvector centrality was proposed by Phil Bonacich in “Factoring and weighting approaches to clique identification.” *Journal of Mathematical Sociology* 2 (1972), 113–20.
- Assortativity is discussed in more detail in M. E. J. Newman, “Mixing patterns in networks.” *Physical Review E* 67.2 (2003), 026126.

6.11 Answers

Answers to the Exercises

- I. Remove Juan by creating a partition identifying him in one class and then extracting the other vertices from the network. When Juan’s

vertex is deleted, the network changes considerably (see the figure that follows). Now, the mill's manager is most central (his or her closeness centrality score is 0.29). The network's closeness centralization decreases to 0.16, and the planners (left) are clearly separated from the mill (right). It is interesting to see that the English-speaking employees are more central, because they are closer than the Spanish-speaking employees to the manager. Juan seemed to function as the informal Hispanic manager.



II. Section 6.4 explains how to compute betweenness centrality. To obtain a sociogram such as that in Figure 62, draw it with the command `Draw> Network + First Vector` (or `Ctrl-u`) from the Main screen.

Answers to the Questions in Section 6.8

1. Answer b is correct: Network centralization measures the variation of the centrality of the vertices within the network. The more variation, the easier it is to distinguish between the center and the periphery, the more centralized the network. Answer a is incorrect because centrality is a property of a vertex, not of a network. Answers c and d are incorrect, because the center of a network, if it has one, may either be a single vertex or a cohesive subgroup (e.g., a clique) consisting of a number of vertices that are equally central.
2. Answer d is correct. The star-network is most central, so answer a is not correct. In the circle-network (network A), all vertices have degree 2, each vertex is equally distant from all other vertices, and each vertex is situated on one geodesic between pairs of other vertices, so there is no variation in centrality, hence minimum centralization. Network A is least centralized, so answer c is not correct. Network D is more centralized than network B, so answer d is correct.
3. The distances between vertex v_1 and vertices v_2 , v_3 , v_4 , and v_5 are 3, 2, 3, and 1, respectively. The sum distance is 9, so the closeness centrality

of vertex v_1 is 4 (the number of other vertices) divided by 9, which is 0.44.

The sum distance of vertex v_3 to v_1 , v_2 , v_4 , and v_5 is $2+1+1+1 = 5$, so its closeness centrality is $4/5 = 0.8$.

4. The geodesics between vertex v_1 and v_2 ($v_1-v_5-v_3-v_2$), v_1 and v_4 ($v_1-v_5-v_3-v_4$), v_5 and v_2 ($v_5-v_3-v_2$), v_5 and v_4 ($v_5-v_3-v_4$), and v_2 and v_4 ($v_2-v_3-v_4$) include vertex v_3 , whereas the geodesic between v_1 and v_5 (v_1-v_5) does not. Five of six geodesics include v_3 , so its betweenness centrality is 0.83.
5. The easiest way to omit all unilateral nominations is to change bi-directional arcs into edges (*Network> Create New Network> Transform> Arcs→Edges> Bidirected only*) and remove the remaining arcs subsequently (*Network> Create New Network> Transform> Remove> all Arcs*). Now you have an undirected network, and it is easy to compute the three kinds of centrality (degree, closeness, and betweenness) and display the vertices with highest scores with the *Partition> Info* and *Vector> Info* commands. Vertex v_9 has highest degree (17), highest closeness centrality (0.67), and highest betweenness centrality (0.20). Person v_9 consistently ranks highest on the three centrality indices.

Brokers and Bridges

7.1 Introduction

A person with many friends and acquaintances has better chances of getting help or information. Therefore, social ties are one measure of social capital, an asset that can be used by actors for positive advantage. Network analysts, however, discovered that the kind of tie is important in addition to the sheer number of ties. Their general argument is that strong (i.e., frequent or intense) ties with people who are themselves related yield less useful information than weak ties with people who do not know one another. Having a lot of ties within a group exposes a person to the same information over and over again, whereas ties outside one's group yield more diverse information that is worth passing on or retaining to make a profit.

As a consequence, we have to pay attention to the ties between a person's contacts. A person who is connected to people who are themselves not directly connected has opportunities to mediate between them and profit from his or her mediation. The ties of this person bridge the structural holes between others. It is hypothesized that people and organizations that bridge structural holes between others have more control and perform better.

In this chapter, we first discuss bridges at the level of the entire network (Section 7.3). Which ties (bridges) and which vertices (cut-vertices) are indispensable for the network to remain connected? If a network contains such ties and vertices, it contains bottlenecks and the flow of information through the network is vulnerable. In the remaining sections, we focus on brokerage at the level of individuals. Who is in the best position to profit from his or her social ties (Section 7.4), and how is this affected by group membership (Section 7.5)?

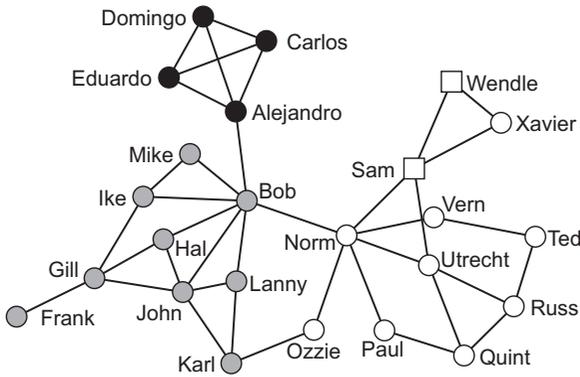


Figure 63. Communication network of striking employees.

7.2 Example

The example in this chapter shows the importance of informal communication structures within a firm. In a wood-processing facility, a new management team proposed changes to the workers' compensation package, which the workers did not accept. They started a strike, which led to a negotiation stalemate. Then, management asked an outsider to analyze the communication structure among the employees because it felt that information about the proposed changes was not effectively communicated to all employees by the union negotiators.

The outside consultant asked all employees to indicate the frequency with which they discussed the strike with each of their colleagues on a 5-point scale, ranging from *almost never* (less than once per week) to *very often* (several times per day). The consultant used 3 as a cutoff value. If at least one of two persons indicated that they discussed work with a frequency of 3 or more, a line between them was added to the informal communication network (*Strike.net*).

The network displays fairly stringent demarcations between groups defined on age and language (Figure 63). The Spanish-speaking young employees, who are of age thirty or younger (class 1 in the partition *Strike_groups.clu*, black or yellow vertices), are almost disconnected from the English-speaking young employees (class 2, gray or green vertices), who communicate with no more than two of the older English-speaking employees (thirty-eight years old or older, class 3: white or red vertices). These divisions mirror the homophily principle discussed in Chapters 3, 5, and 6: People tend to relate to those who are similar: the External-Internal (E-I) Index of a partition defined by the three groups is large and negative (-0.84).

All ties between groups have special backgrounds. Among the Hispanics, Alejandro is most proficient in English and Bob speaks some Spanish, which explains their tie. Bob owes Norm for getting his job, and probably because of this, they developed a friendship tie. Finally, Ozzie is the father of Karl.

Sam and Wendle are the union negotiators; they are represented by boxes in Figure 63 (open the network data file `Strike.net` in a text editor to see how we created boxes for Sam and Wendle in the list of vertices). They were responsible for explaining the new program proposed by the managers. When the informal communication structure among employees was reported to the management, they approached two of the other employees directly (Bob and Norm) to explain the reforms to them personally. Then, they gave them some time to discuss the plans with their colleagues. Within two days, the younger and older employees were willing to strike a deal with the management, and they persuaded the union representatives to reopen negotiations. Soon, the labor dispute was reconciled, and the strike ended.

7.3 Bridges and Bi-Components

The example shows the importance of social ties to the diffusion of information. Information is the key to the exploitation of social ties as social capital. Social ties offer access to information, which can be used to reduce uncertainty and risk and to create trust, as for instance, when information is confirmed from several sources. People in crucial positions in the information network may also spread or retain information strategically because they have control over the diffusion of information.

In a social system, for instance, an organization, the overall structure of informal ties is relevant to the diffusion of information. Can information reach all members of the organization, or is it more likely to circulate in one segment of the network? Are there any bottlenecks that are vital to the flow of information, which may prohibit the spread of information because of information overload or because people pursue their private, strategic goals?

In Figure 63, the tie between Alejandro and Bob is clearly a bottleneck because it is the only channel for information exchange between the Hispanic employees and all other employees. Removing this single line will cut off the Hispanics from information circulating among the other employees. Formally, this line is a *bridge* in the network because its removal creates a new component that is isolated from other components. The strike network consists of one component (recall that a component is a maximal connected subnetwork; see Chapter 3), so information may travel to each employee via social ties. When you remove the line between

Alejandro and Bob, you disconnect the Hispanic workers from the communication network, so they become a component on their own.

A *bridge* is a line whose removal increases the number of components in the network.

Note that there is one more bridge in the information network of striking employees: the tie between Frank and Gill. If you remove this tie, Frank becomes an isolate. Because an isolate is a component, the network consists of two components after removing Frank's tie with Gill.

The removal of a line may annul the connectedness of a network or component, but the deletion of a vertex may have the same effect, because you remove the lines that are incident with the deleted vertex. After all, you cannot have a line with a single endpoint! When Bob refuses to discuss the strike any further, he is lost to the communication network and all of his ties disappear, including the bridge to Alejandro. Therefore, Bob is a *cut-vertex* or *articulation point*: Its deletion disconnects the network, or it disconnects a component of the network. Just as with a bridge, a cut-vertex is crucial to the flow of information in a network. It is a bottleneck in the network that controls the flow from one part to another part of the network. Norm, for example, is indispensable for exchanging information between the older and younger employees.

Deleting a vertex from a network means that the vertex and all lines incident with this vertex are removed from the network.

A *cut-vertex* is a vertex whose deletion increases the number of components in the network.

In Figure 64, all cut-vertices are gray. Note that vertices incident with a bridge may or may not be cut-vertices. Alejandro and Bob are cut-vertices, but Frank is not, because removal of Frank and his bridge to Gill does not increase the number of components.

Now that we have defined cut-vertices, it is easy to define sections of a network that are relatively invulnerable to the withdrawal or manipulation of a single vertex, namely bi-components. A *bi-component* is simply a component – a maximal connected subnetwork – of minimum size 3 without a cut-vertex. In a bi-component, no person can control the information flow between two other persons completely because there is always an alternative path that information may follow. In a bi-component, each person receives information from at least two sources (in an undirected network), so he or she may check the information. We may say that a

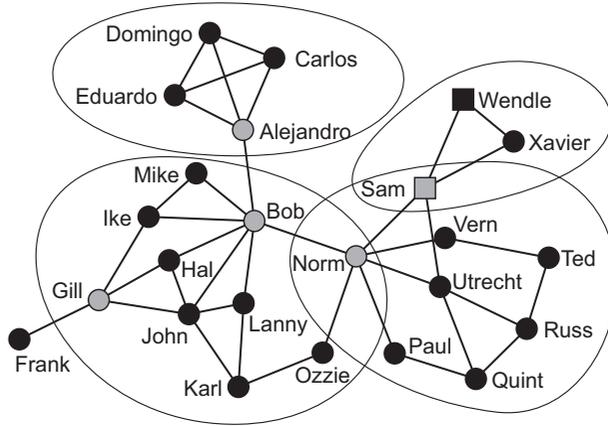


Figure 64. Cut-vertices (gray) and bi-components (manually circled) in the strike network.

bi-component is more cohesive than a strong or weak component because there are at least two different paths between each pair of vertices; that is, two paths that do not share a vertex in between their starting point and endpoint.

A *bi-component* is a component of minimum size 3 that does not contain a cut-vertex.

The strike network as a whole clearly is not a bi-component because it contains five cut-vertices. Within the network, however, there are four bi-components, which are manually circled in Figure 64. These are, clockwise and starting at the top, (1) a Hispanic bi-component, consisting of Alejandro, Carlos, Domingo, and Eduardo; (2) a bi-component with the two union representatives and Xavier; (3) a bi-component with all older English-speaking employees except Ozzie, Wendle, and Xavier; and (4) a bi-component with Ozzie, Norm, and all younger English speakers except Frank.

You should be puzzled now. Didn't we define a bi-component as a component (of minimum size 3) without a cut-vertex? Then, how is it possible that each of the listed bi-components contains at least one gray vertex in Figure 64, that is, at least one cut-vertex? The answer is that a bi-component does not contain cut-vertices if you look at the bi-component only and ignore the rest of the network. Concentrate on the Hispanic employees, for instance: If you remove Alejandro, the other three Hispanics remain connected into one component, so the removal of Alejandro does not increase the number of components among the Hispanic

employees. Looking at the entire network, however, Alejandro is a cut-vertex because he connects the Hispanic bi-component to Bob.

In other words, a cut-vertex always connects different bi-components or bridges. Norm, for example, belongs to two bi-components: to the majority of older English speakers and to the bi-component of the young English-speaking employees. In a similar way, Sam connects two bi-components. Alejandro, Bob, and Gill, however, connect a bi-component to a bridge, namely, the bridge between Alejandro and Bob or the bridge between Gill and Frank. Cut-vertices indicate the borders of bi-components and bridges. A component usually consists of overlapping bi-components and bi-components connected by bridges.

It is interesting to note that the two union representatives among the employees, Wendle and Sam, are part of a bi-component that is connected to the bi-component of older employees by Sam. So we may say that Sam controls the information exchange between the union representatives and all other employees except Xavier. If Sam does not want to strike a deal with the management of the firm, he can manipulate the information to and from the other employees.

In numerous applications, it has been shown that people with strong ties belong to cliques, and strong ties tend to be located in or develop into cliques; for example, family ties are usually strong in the sense that they are intense, and family ties display cliques: Several or all members of a family maintain strong ties among themselves. As a consequence, family ties are not very useful in finding new jobs because they relate you to people with whom you are already related. Usually, they do not supply information about new jobs of which you have not already heard. In contrast, less intense and irregular contacts such as former colleagues or acquaintances are better sources of information on new job opportunities. These weak ties are more likely to be bridges to distant information networks, hence the concept of “the strength of weak ties,” meaning that weak ties are often more important for the dispersion of information than strong ties. The strength of a tie may be taken as a proxy of its chances of being a network bridge.

This hypothesis could apply to the Spanish-speaking employees. Strong ethnic ties develop into a clique, and the only nonethnic tie (between Alejandro and Bob) is a bridge to the rest of the network. In this example, however, family ties connect employees outside cliques: Gill is Frank’s cousin, and Ozzie is Karl’s father. We should note that the strength of weak ties depends on the situation: On the shop floor, family ties, which are usually considered strong, may fulfill the bridging role of weak ties because family ties are uncommon and they will not develop into cliques within the firm (a firm is not the natural setting for raising a family).

Remember, however, that we consider only the stronger communication ties because irregular communication ties are disregarded (scores 1 and 2

on the 5-point scale) in this example. The strength-of-weak-ties argument predicts that strong ties will constitute cliques, which is clearly the case. Perhaps, the weaker ties cross group boundaries more often. Note that the strength of a tie may be defined in several ways, for instance, frequency versus social intensity, which is important to consider when you apply the strength-of-weak-ties hypothesis.

Application

Network> Create New Network> with Bi-Connected Components stored as Relation Numbers You may use the *with Bi-Connected Components stored as Relation Numbers* command in the *Network> Create New Network* submenu for finding bi-components, bridges, and cut-vertices in a network. On selection of this command, you are prompted to specify the minimum size of the bi-components to be identified. The default value 3 will identify the bi-components within the network, and it will report only cut-vertices that connect two or more bi-components. A minimum size of 2 will trace all bi-components, bridges, and all cut-vertices, including cut-vertices connecting bridges. Note that a bridge and its incident vertices constitute a component of size 2 without a cut-vertex in an undirected network.

Vector> Make Partition> Copy to Partition by Truncating (Abs)

Pajek's *Bi-Connected Components* command treats directed networks as if they were undirected, which means that it identifies weak instead of strong components without cut-vertices in directed networks. If you symmetrize a directed network before you execute the *Bi-Connected Components* command, you will obtain exactly the same results.

In this example, we want to identify the bi-components and the bridges, so we issue the *Bi-Connected Components* command with a minimum component size of 2. The output of the *Bi-Connected Components* command consists of a partition, a vector, and a hierarchy. The partition ("Vertices belonging to exactly one bicomponent") indicates the sequential number of the bridge or bi-component to which a vertex belongs. Vertices that do not belong to a bridge or bi-component (e.g., isolates) are collected in class 0, and vertices that belong to two or more bridges or bi-components – cut-vertices – are placed in class number 999999998.

The vector ("Articulation points") indicates the number of bridges or bi-components to which a vertex belongs: 0 for isolates, 1 for a vertex that belongs to exactly one bridge or bi-component, 2 for vertices that belong to two bridges or bi-components, and so on. We can represent the number of bridges or bi-components to which a vertex belongs by vertex size. But if we want to show this property with different colors, we have to copy the vector into a partition by truncation (run *Vector> Make Partition> Copy to Partition by Truncating (Abs)* or simply press F6) and draw the network with the new partition as in Figure 64. Note that the contours in this figure were added manually. Finally, the hierarchy shows the bridges or bi-components to which each vertex belongs. We need a hierarchy to store the bridges and bi-components because cut-vertices belong to two or

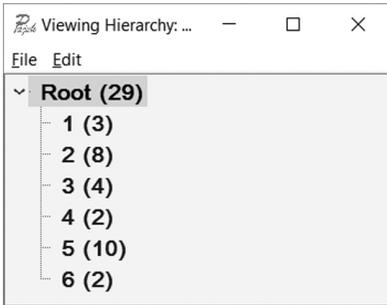


Figure 65. Hierarchy of bi-components and bridges in the strike network.

more bi-components. Note that bridges are not counted if the minimum size has been set to 3.

Although vertices may belong to more than one bi-component, each line belongs to exactly one bi-component. As a consequence, bi-components can also be stored as a characteristic of the lines. In Pajek, the sequential number of the bi-component to which a line belongs can be stored as the relation number of a line. When running bi-components Pajek asks if you want to generate a new network with relation numbers indicating bi-components or store relation numbers in the current network. Be careful if you choose the latter option: If the network is already multirelational, old relation numbers will be lost. Because bridges are bi-components of size 2 in an undirected network without multiple lines, you can easily find the bridges in the hierarchy of bi-components: Open the Edit screen with the hierarchy of bi-components (see Figure 65) with the command `File> Hierarchy> View/Edit` or with the View/Edit button on the left of the *Hierarchy* drop-down menu. Figure 65 lists the six bridges and bi-components in the communication network among striking employees. The size of each subnetwork is reported between brackets, so it is easy to find the two bridges in this example: subnetworks 4 and 6. Double-click them to see their vertices.

*File>
Hierarchy>
View/Edit*

Exercise 1

Detect only the bi-components in the strike network by setting the minimum component size to 3 in the *Bi-Connected Components* command. How many cut-vertices does Pajek identify now? And what happens to Frank?

7.4 Ego-Networks and Constraint

In the previous section, we analyzed the structure of the entire network, which is a *sociocentered approach*. Now we turn to the *ego-network* and

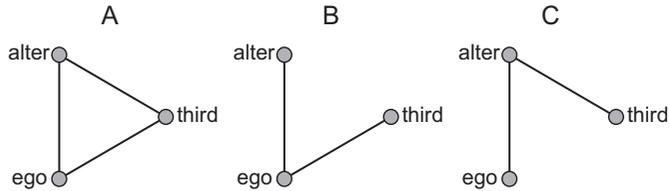


Figure 66. Three connected triads.

ego-centered approach: We focus on the position of one person in the network and his or her opportunities to broker or mediate between other people.

Let us first have a look at a *triad*, which consists of a focal person (ego), an alter, a third person, and the ties among them. The triad is the smallest network that contains more than two persons, and it highlights the complexities of ties within a group. According to the sociologist Georg Simmel, a complete triad (A in Figure 66) reduces the individuality of its members. When three people are fully connected, they share norms and information, they create trust with feedback, and conflicts between two members may be resolved or moderated by the third person. In other words, complete connections between three persons make them behave as a group rather than as a set of individuals.

In an undirected triad that is connected but incomplete, for instance, networks B and C in Figure 66, people are considered less bound by group norms. One person is in an advantageous, powerful position because he or she may broker between the other two. The person in the middle (the ego in B and alter in C, Figure 66) may profit from the competition between the other two; for example, the ego negotiates the price of a good or service to be delivered by either alter or the third party in network B. The ego makes them compete, which would not be possible if the alter and third would agree about a price among themselves. This is known as the *tertius gaudens* (“the third who benefits”) or the *tertius* strategy: Induce and exploit competition or rivalry between the other two, who are not directly related. The absence of a tie between an alter and the third party is known as a *structural hole*, which may be exploited by the ego.

A more malicious variant is known as *divide et impera* or the divide-and-rule strategy, in which a person creates and exploits conflict between the other two to control both of them; for example, the ego tells alter unpleasant things about the third party and the third party about alter, which results in hostility among them. This would not be possible if they could directly check the information and find out the ego’s subversive strategy. Again, the structural hole allows the ego to apply this strategy.

In both strategies, an individual’s advantage or power is based on his or her control over the spread of information, goods, or services, which

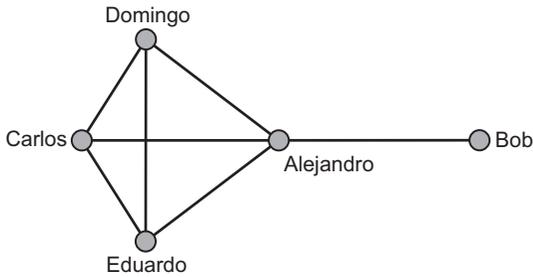


Figure 67. Alejandro's ego-network.

stems from the structure of his or her network. We want to stress that brokerage is related to the absence of ties (i.e., the presence of holes) between neighbors, whereas we concentrated on the presence of ties in our chapters about cohesive subgroups in Part II.

The opportunities that a structural hole offers in an incomplete triad have a reverse side: They imply constraint in a complete triad. A complete triad is not just a triad without opportunities because it has no structural holes. The situation is even worse from the perspective of brokerage, because you cannot withdraw from any of these unrewarding ties without creating a structural hole around yourself. In network A (Figure 66), the ego is more or less obliged to maintain both ties, because if the ego ends ties with one (e.g., with the third in A, so triad C evolves), there is a structural hole around the ego that the alter may take advantage of.

The *ego-network* of a vertex contains this vertex, its neighbors, and all lines among the selected vertices.

Now, let us focus on the *ego-network*, which consists of an ego, the ego's neighbors, and the ties among them. Alejandro's ego-network is displayed in Figure 67. The ego-network of a person contains all connected triads that include this person, so we can analyze it as a set of triads. For each triad, we can determine whether it constrains ego or whether it contains a structural hole that the ego may exploit; for instance, Alejandro (ego) has an opportunity to broker between Bob (alter) and Domingo (third) because Bob and Domingo are not directly connected. For the same reason, Alejandro may broker between Bob and Carlos or Eduardo. There are three triads in Alejandro's ego-network that give him an opportunity to broker for Bob.

In a similar way, we can compute the constraint on Alejandro that is exercised by his tie with Bob: the number of complete triads containing Alejandro, Bob, and another neighbor of Alejandro. Because no other

neighbor of Alejandro is directly connected to Bob, there is no constraint on Alejandro because of his tie with Bob. A low constraint indicates many structural holes, which may be exploited. In contrast, the constraint on Alejandro's ties with Carlos, Domingo, and Eduardo is very high because these ties are involved in three complete triads. When Alejandro withdraws from any of these ties, they may start brokering for him.

The higher the constraint, the fewer the opportunities to broker and the more dangerous it is to withdraw from a tie. This constraint is known as the *dyadic constraint* associated with a tie from ego's point of view. Note that the constraint of a tie on ego may differ from the constraint experienced by alter on the same tie. The tie between Alejandro and Carlos, for instance, is more constrained for Carlos than for Alejandro because all triads in Carlos' ego-network are complete.

In our discussion of structural holes and constraint, something is still missing: We ought to take into account the importance of a tie to a person. If a tie is very cheap in terms of investment (money, network time, and energy), it is not really a problem to be obliged to maintain it. If a tie is just one among many (low exclusivity), ego does not depend on this tie much and it is no big deal if alter threatens to break it. Besides, if the tie between the alter and the third party is not important to them, it may function like an absent tie, which can be exploited.

The *proportional strength* of a tie with respect to all ties of a person is a simple indicator of the importance or exclusivity of a tie. It is computed as the value of the line(s) representing a tie, divided by the sum of the values of all lines incident with a person. If line values express costs, time, or energy, the proportional strength of a tie is the portion of an actor's total expenditure that is invested in the ties with an alter. Just like dyadic constraint, it makes a difference from which standpoint you look at the network; for instance, the tie between Alejandro and Bob is one out of Alejandro's four ties (0.25), but it is only one of Bob's seven ties (0.14), so the proportional strength of a tie must be represented by a directed network (Figure 68). Note that the original network may contain multiple lines, directed and undirected lines, and line values, but the network with proportional strength ties is always simple directed and contains only bidirected arcs.

The *dyadic constraint* on vertex u exercised by a tie between vertices u and v is the extent to which u has more and stronger ties with neighbors who are strongly connected with vertex v .

This definition describes the ideas behind dyadic constraint rather than the exact computation. For those interested in the exact computation: Add the proportional strength of the tie from the ego to the alter

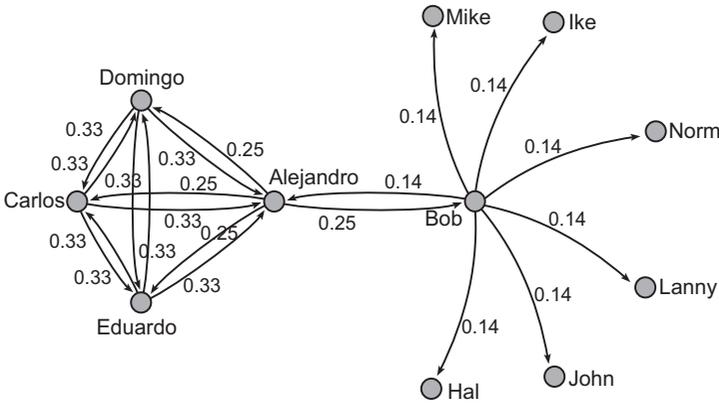


Figure 68. Proportional strength of ties around Alejandro.

(investment of the ego in the alter) to the products of the proportional strength of the two arcs in each path from the ego to the alter via another neighbor of the ego, and take the square of this sum.

The constraint on Alejandro attached to his tie with Eduardo is equal to the square of the following sum: 0.25 (Alejandro’s investment in Eduardo), plus 0.25×0.33 (Alejandro’s tie to Carlos times Carlos’ tie to Eduardo), plus 0.25×0.33 (Alejandro’s tie to Domingo times Domingo’s tie to Eduardo). All numbers are proportional strengths that can be read from Figure 68, from which we omitted arcs toward Bob that are not relevant to the constraints on Alejandro. The sum is 0.415, and the square of this sum is 0.17 (see Figure 69). As you may have expected, the constraint on Alejandro that is attached to his ties with Domingo or Carlos is also 0.17. The constraint on his tie with Bob is just the square of the proportional strength of this tie (0.0625) because there are no indirect paths from Alejandro to Bob in Alejandro’s ego-network.

We may conclude that the constraint on Alejandro’s tie with Bob is about one-third of the constraint of his ties within the Hispanic cluster. Clearly, the structural holes in Alejandro’s network are attached to his

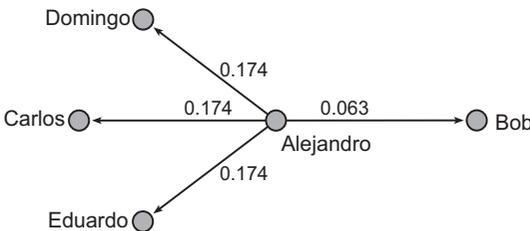


Figure 69. Constraints on Alejandro.

link with Bob: He is able to play the *tertius gaudens* strategy between Bob and the other Hispanics because he may act as a representative of the Hispanics to Bob.

If we have the dyadic constraint on all ties of a person, we can simply add them to obtain the *aggregate constraint* on this person. The aggregate constraint is a nonnegative number that is usually between 0 and 1, but it can be greater than 1. The aggregate constraint on Alejandro, for instance, is $0.174 + 0.174 + 0.174 + 0.063 = 0.585$. The higher the aggregate constraint, the less “freedom” a person has to withdraw from existing ties or to exploit structural holes.

In general, more direct links between an ego’s neighbors yield a higher aggregate constraint on that ego because each link between neighbors creates a complete triad with the ego. For this reason, network analysts have used the *density of the ego-network* without the ego as an indicator of the constraint on an ego. In the case of Alejandro, there are three lines among his four neighbors (see Figure 67) (viz., between Carlos, Domingo, and Eduardo). The *egocentric density* is 0.5: Three of the six possible lines exist among Alejandro’s four neighbors. In contrast, Carlos’ egocentric density is maximal because all of his neighbors (Domingo and Eduardo) are directly linked.

People or organizations with low aggregate constraint are hypothesized to perform better. It has been shown that employees with low constraint in an organization have more successful careers and that business sectors with lower constraint on firms are more profitable. In general, researchers compare the constraint on an actor to one or more indicators of its (economic) success. In our example, this could be the success in resolving the conflict between employees and management or personal influence on the conditions specified in the final agreement. Bob and Norm negotiated the proposal with the management before they called in the union representatives, so they may have been successful in changing the conditions according to their interests.

Application

Network>
Create Vector>
Structural
Holes

In Pajek, one command computes the proportional strength of ties, the dyadic constraint, and the aggregate constraint for all vertices in a network. This command is aptly called *Structural Holes*, and you can find it in the *Network> Create Vector* submenu. The proportional strength of ties is output as a new network, and so is dyadic constraint. In these networks, the line values express the strength and constraint on ties, respectively. Note that these networks are always directed and that all arcs are reciprocated, no matter whether the original network is directed or undirected and valued or unvalued or contains multiple lines and loops.

Options>
Values of
Lines>
Similarities

There is an easy way to visualize the structural holes in a network. Take the network of dyadic constraint and, using the line values as similarities

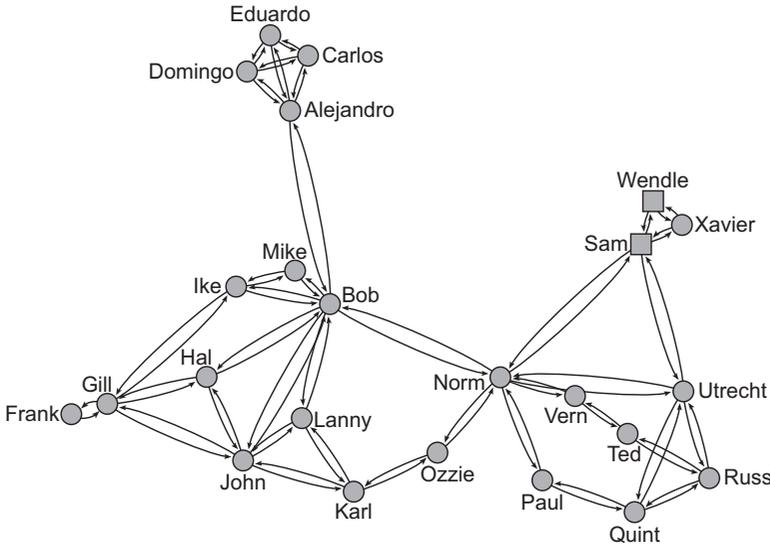


Figure 70. Energized constraint network.

(option *Options*> *Values of Lines*> *Similarities* in the Draw screen), energize it. Now, vertices that are tied by links of high constraint are drawn closely together, whereas ties of low constraint are long, so they create a lot of space between the vertices, which looks like a hole (Figure 70 – energized with Kamada–Kawai).

Aggregate constraint is output as a vector. You may inspect this vector in the usual ways with the *Vector*> *Info* command or by editing it. If you want the size of the vertices to represent their aggregate constraint in the Draw screen, we advise multiplying the vector by 10 (command *Vector*> *Transform*> *Multiply by*) or using the *Autosize* option in the *Options*> *Size*> of *Vertices* submenu of the Draw screen; otherwise the vertices are drawn too small.

To calculate the egocentric density of a vertex, that is, the density of ties among its neighbors, you must extract the subnetwork of the neighbors from the overall network. First, select the neighbors of a particular vertex with the *Network*> *Create Partition*> *k-Neighbours*> *All* command. In the first entry, specify the number or label of the vertex for which you want to compute egocentric density (e.g., Alejandro). For the second answer, enter 1 as the maximum distance. The command now creates a partition with the ego in class 0 and its neighbors in class 1. Second, extract the neighbors from the network with the *Operations*> *Network + Partition*> *Extract*> *SubNetwork Induced by Union of Selected Clusters* command, selecting class 1 as the only class to be extracted. Now that you have

Vector> *Info*

Vector>
Transform>
Multiply by

[*Draw*]
Options>
Size> of
Vertices

Network>
Create
Partition>
k-Neighbours>
All

Operations>
Network +
Partition>
Extract>
SubNetwork
Induced by
Union of
Selected
Clusters

created the network of neighbors, you can inspect its density with the *Network> Info> General* command in the Main screen.

[Main]
Network>
Info> General

Network>
Create Vector>
Clustering
Coefficients>
CC1

For a simple undirected network, you can compute the egocentric density of all vertices with the *Network> Create Vector> Clustering Coefficients> CC1* command. This command produces two vectors; the first vector – “Clustering Coefficients CC1” but *not* “Clustering Coefficients CC1” – contains the egocentric density. The results for directed networks represent egocentric density only if the network does not contain loops or bidirectional arcs.

Exercise II

Compute the aggregate constraint on Norm and Bob in the strike network as well as their egocentric density. Do aggregate constraint and egocentric density match in this case?

7.5 Affiliations and Brokerage Roles

Group affiliation is often important in brokerage processes. A union representative mediates between the management and the workers. He or she can negotiate with one manager or another and choose whom of his or her colleagues to consult. To some extent, his or her contacts are replaceable by someone else from the same group. Moreover, the union representative himself must belong to a particular group, namely, the workers. In our example, the union representatives Sam and Wendle are a subgroup of workers, and the management is supposed to negotiate with them. This restricts the managers’ choice of negotiation partners enormously, so they have little opportunity to play off one negotiator or worker against another. In this case, the opportunity to broker depends not only on the position of people in the network but also on their group affiliations.

The easier it is to replace your contact with someone else from his or her group, the stronger your position is to negotiate and the higher the chance of striking a good deal or of getting things done your way. The replacement does not have to be one of your present contacts; it may be someone outside of your present ego-network whom you include at the expense of someone else. Is there someone else in your contact’s group who is at least as central as your contact but who is not directly linked to your contact so including him or her in your ego-network would create a structural hole between your present contact and the new contact? Such a structural hole is called a *secondary structural hole*.

Let us illustrate this with an example. Suppose Alejandro wants to play a divide-and-rule strategy against Bob because he feels too constrained by him. Bob is in a good structural position to negotiate on behalf of the English-speaking younger employees because he is directly connected to

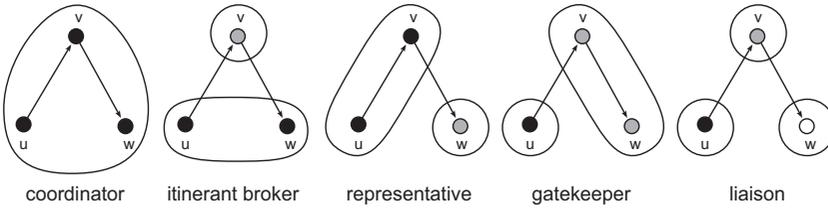


Figure 71. Five brokerage roles of actor v .

most members of this group. It is very difficult to spread discord among members of his group by spreading rumors about Bob, because the other members of the group are likely to inform Bob when they maintain direct ties. Alejandro's best choice seems to be Gill, because he is not directly connected to Bob and it is very likely that Frank will team up with him, after which Gill may try to play Ike versus Hall and John, although these colleagues are constrained by Bob. Frank and Karl, the only other English-speaking younger employees who are not directly related to Bob, are less suited as an alternative to Bob because they are less central in the group.

Because secondary structural holes concern the ties within one group, namely, the opportunities to exploit structural holes within that group, the aggregate constraint within a group seems to be a useful indicator of whom to contact and persuade as an alternative to your present contact in the group: the person who is least constrained and who is not constrained by your present contact because he is not directly tied to him or her (see Figure 73 in the "Application" section).

For another approach to brokerage and affiliations, we have to turn to triads again. A triad in which person v mediates transactions between persons u and w can display five different patterns of group affiliations, which are indicated by vertex color as well as contours in Figure 71. Each pattern is known as a *brokerage role*. Research into brokerage roles is concerned with describing the types of brokerage roles that dominate a transactional or exchange network. In addition, individual positions within the network may be characterized by the dominant type of brokerage role, and hypotheses may be tested about the personal characteristics of individuals with certain types of brokerage roles.

Two brokerage roles involve mediation between members of one group. In the first role, the mediator is also a member of the group. This is known as the *coordinator role*. In the second role, two members of a group use a mediator from outside, an *itinerant broker*. The other three brokerage roles describe mediation between members of different groups. In one role, the mediator acts as a *representative* of his group because he regulates the flow of information or goods from his or her own group. In

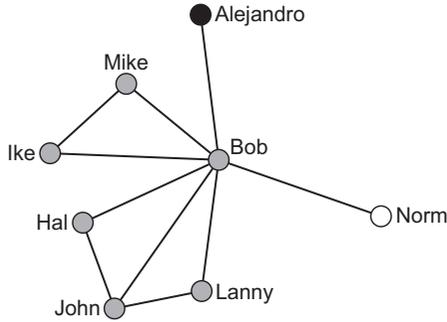


Figure 72. Bob's ego-network.

another role, the mediator is a *gatekeeper*, who regulates the flow of information or goods to his or her group. Finally, the *liaison* is a person who mediates between members of different groups but who does not belong to these groups himself or herself.

The five types of brokerage roles have been conceived for directed networks, namely, transaction networks. Note, however, that the direction of relations is only needed to distinguish between the representative and the gatekeeper. The other brokerage roles are also apparent in undirected relations, so we can apply the brokerage roles to undirected networks if we do not distinguish between representatives and gatekeepers. In an undirected network, each representative is also a gatekeeper and vice versa. We should note, however, that the other three brokerage roles (coordinator, itinerant broker, and liaison) always yield even counts in an undirected network because the edge is interpreted as a bidirectional arc. If, for example, the ego (v) coordinates between alter (u) and the third person (w), it also coordinates between the third person (w) and alter (v).

Now, let us have a look at the brokerage roles in the strike network. We use the groups according to language and age (see Figure 63), and we assume that a line is equivalent to a bidirectional arc: Discussing work implies the possibility of disseminating and receiving information. Employees who are isolated or whose ties are contained within a clique (e.g., Carlos, Domingo, and Eduardo, but also Wendle and Xavier) have no opportunity to mediate because all of their contacts are directly connected. As a result, none of the brokerage roles apply to them.

Most of the other employees have ties only within their own group, so they can play only the coordinator role. In the network, brokerage is clearly dominated by the coordinator role. It is easy to see that Alejandro, Bob, Norm, and Ozzie are the only employees who also have other types of brokerage roles because they are the only ones who are connected to members of different groups.

Let us have a closer look at Bob (Figure 72), who combines several types of brokerage roles. There are several structural holes among Bob's ties

within the group of English-speaking younger employees (e.g., between Ike and Mike, on the one hand, and Hal, John, and Lanny, on the other hand). To them, Bob plays the coordinator role. In addition, Bob bridges many structural holes between his group of English-speaking younger employees and the Hispanic workers or the older employees. For information about his group, Bob is a representative; and for information flowing toward members of his group, he is a gatekeeper. Finally, he may mediate between Alejandro and Norm, that is, between the Hispanics and the older workers. In this role, he is a liaison.

The only brokerage role that Bob cannot play given the ties in the network is the role of an itinerant broker because he has no ties with two or more members of any group other than his own. Actually, none of the employees can play this role – this role is absent in the strike network.

Bob was the first employee whom the management contacted directly. Perhaps, this was justified not only by the amount of structural holes in his ego-network but also by the variety of brokerage roles that Bob may play.

Application

Secondary structural holes are related to constraint within a group, so we may delete the ties between groups and calculate the constraint within each group. If there is another member of the group with equal or less constraint than the one you are already connected to, you may play him or her off against your present contact provided that they are not strongly and directly connected. In a similar manner, you may evaluate your position within your own group to see whether you may easily be replaced by someone else.

The detection of secondary structural holes consists of two steps. In the first step, we delete the lines between groups. Because the groups are defined as classes in a partition (in our example `Strike_groups.clu`), we must make sure that this partition is selected in the *Partition* dropdown menu. Then we can remove the lines between clusters with the command `Operations> Network + Partition> Transform> Remove Lines> Between Clusters`.

In the second step, we apply the *Structural Holes* command to the network without lines between clusters to obtain the constraint of vertices within their groups (Figure 73). We can see that Gill is even less constrained within the class of younger English-speaking employees than Bob. Because there is no direct tie between Gill and Bob, Gill seems to be a good candidate to be played off against Bob. In the Hispanic group, there is no real alternative to Alejandro because he is directly connected to all others. Among the older employees, Norm is clearly less constrained than any other employee, so there is no good alternative in this group. Judging from their structural positions and ignoring their linguistic abilities or special relationships, we conclude that Norm and Alejandro are

```
Operations>
Network +
Partition>
Transform>
Remove Lines>
Between
Clusters
```

```
Network>
Create Vector>
Structural
Holes
```

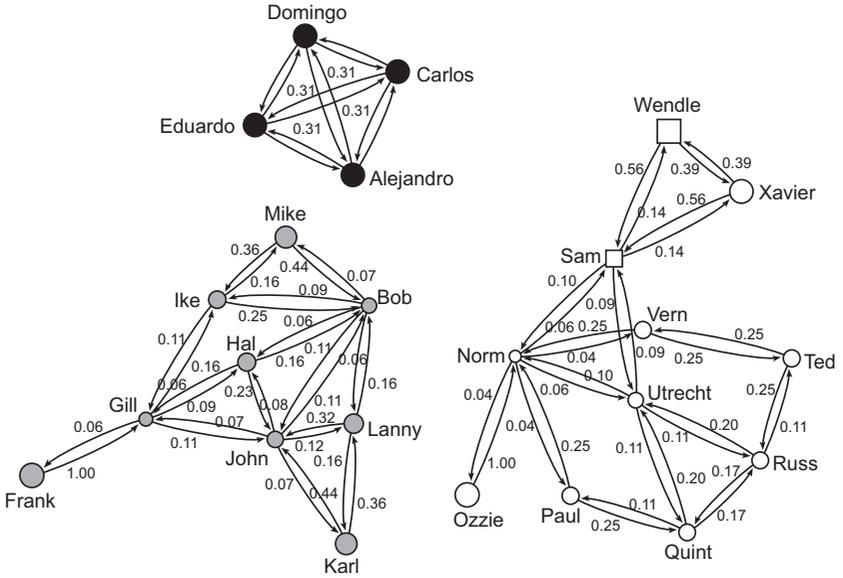


Figure 73. Constraint inside groups.

less likely to be replaced as representatives or gatekeepers of their groups than Bob because there is a good alternative to Bob only.

```

Operations>
Network +
Partition>
Brokerage
Roles
Vector> Make
Partition>
Copy to
Partition by
Truncating
(Abs)
Partition> Info
    
```

Pajek contains a command that counts the brokerage roles in a network. Make sure that the strike network and the appropriate partition (`strike_groups.clu`) are selected in the drop-down menus of the Main screen. Then, execute the command *Operations> Network + Partition> Brokerage Roles* to obtain five new vectors, one for each brokerage role, which are added to the *Vector* drop-down menu. The value of a vertex in a vector specifies the number of incomplete triads in which this vertex plays the corresponding brokerage role. Table 9 shows the results for the coordinator role, which is stored in the vector labeled

Table 9. Frequency tabulation of coordinator roles in the strike network

Cluster	Freq	Freq%	CumFreq	CumFreq%	Representative
0	10	41.6667	10	41.6667	Frank
2	5	20.8333	15	62.5000	Hal
4	3	12.5000	18	75.0000	Ike
8	2	8.3333	20	83.3333	Utrecht
10	1	4.1667	21	87.5000	Gill
12	1	4.1667	22	91.6667	John
14	1	4.1667	23	95.8333	Bob
18	1	4.1667	24	100.0000	Norm
SUM	24	100			

“Coordinators in N1 according to C1.” Since all values in these vectors are positive integers, we can again transform them into partitions (*Vector*> *Make Partition*> *Copy to Partition by Truncating (Abs)* or F6) and apply the *Partition*> *Info* command which is more suitable for presenting few integer values than the *Vector*> *Info* command.

We can see that ten employees have no coordinator roles. The number of coordinator roles per person is unevenly distributed: Some employees have two or four coordinator roles, whereas Norm has no less than eighteen coordinator roles. If we sum the roles (five employees with two coordinator roles plus three with four roles, etc.), we count ninety-two coordinator roles. Note that there are only edges in this network, which are equivalent to bidirectional arcs; therefore, the number of coordinator roles is twice what you may have expected. In the representative roles partition, we count twenty-one roles, and there are just two liaison roles (connected with Bob). As noted, the coordinator role occurs most frequently in this network because most employees have direct ties only within their own subgroup.

7.6 Summary

In a connected social network, information may reach anybody through their ties. Holes in this network, that is, absent ties, are obstacles to flows of information. Information is less likely to reach anybody easily in a connected network with large holes. In this chapter, we focus on the holes in a network. The information flow is especially vulnerable in networks with bridges and cut-vertices because the removal of a bridge or cut-vertex disconnects the network. Actors who are cut-vertices in a network control the flow of information from one part of the network to another. They may decide to retain information when it suits their personal purposes.

From the perspective of the communication system as a whole, bridges and cut-vertices are undesirable. From the individual point of view, however, being a cut-vertex is attractive because it offers opportunities for brokering information and for profiting from brokerage in one way or another. The advantage of the broker position in an ego-network, which is the network of one actor and its neighbors, is that you can play a *tertius* strategy: You can induce competition or conflict between neighbors who are not linked directly. The gap between the neighbors is called a structural hole, and each structural hole represents an opportunity to broker.

There is a drawback; however, you must avoid becoming the object of a *tertius* strategy yourself. This implies that you cannot end ties with neighbors who are directly linked. This is called the constraint on your tie with a neighbor. The constraint on a tie is inversely related to the structural holes associated with it: Low constraint means many structural holes and vice

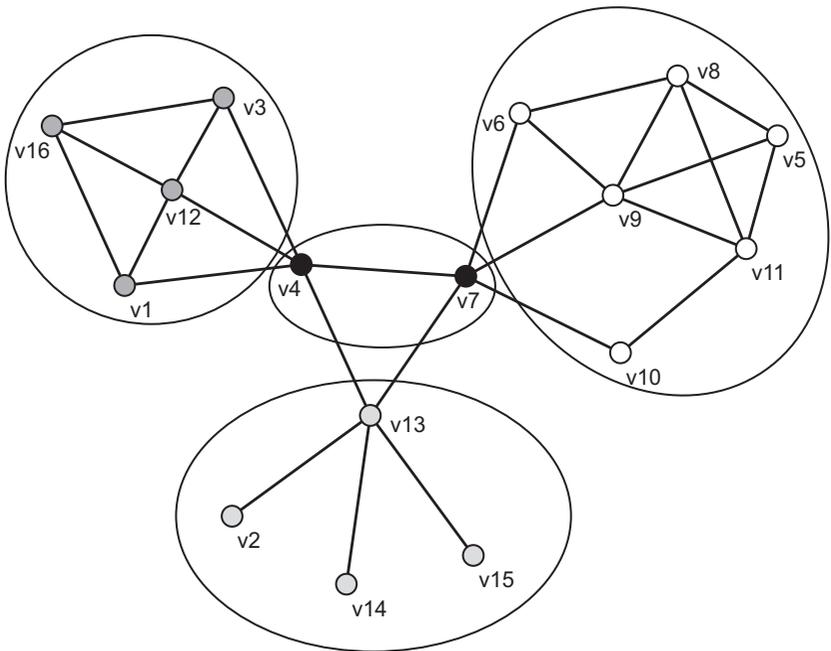
versa. The constraint on each tie as well as on each vertex in the network may be calculated to find the segments that have the most opportunities to broker, which are hypothesized to be more successful and profitable.

In many social contexts, brokerage is connected to group affiliations, and the people involved can be replaced only by other persons from their groups. Threatening a contact to replace him or her with another contact is also a *tertius* strategy. This strategy is successful only if there is a good alternative to the present contact in the group, someone who is also quite central in the group but who is not directly linked to the present contact so the (secondary) structural hole between them can be exploited.

When we consider brokerage in the context of group membership, there are five brokerage roles. We can characterize a network or an actor in a network by the kinds of brokerage roles that occur. The brokerage signature of a network or actor can be compared to other characteristics to determine whether certain types of actors or types of social relations develop particular brokerage roles.

7.7 Questions

1. List the bridges in the network depicted in the following figure (ignore contours and vertex colors).



2. How many bi-components and cut-vertices does the network of Question 1 contain?
 - a. Three bi-components and two cut-vertices
 - b. Three bi-components and three cut-vertices
 - c. Six bi-components and two cut-vertices
 - d. Six bi-components and three cut-vertices
3. Let the network of Question 1 represent the communication network within an organization. If you would like to reduce the power of cut-vertices to control the flow of information, which pair of vertices would you urge to establish a communication tie? Justify your answer.
4. Which of the following statements is correct?
 - a. Vertices incident with a bridge are cut-vertices if and only if they have two or more neighbors.
 - b. Vertices that are part of a bi-component cannot be cut-vertices.
 - c. A bi-component is a subnetwork without a cut-vertex.
 - d. If there are two paths between all pairs of vertices in a component, this component is a bi-component.
5. Which of the following statements about the strength-of-weak-ties hypothesis is correct?
 - a. Strong family ties cannot be bridges in a communication network.
 - b. A tie is weak if and only if it is a bridge in a communication network.
 - c. In general, weak ties are more likely to be bridges in a communication network.
 - d. A tie is strong if and only if it is part of a clique in a communication network.
6. In the network of Question 1, which vertex is least constrained: v_4 , v_7 , or v_{13} ? Justify your answer.
7. When vertex v_7 wants to reduce the number of ties that he is maintaining in the network of Question 1, which tie would you advise him to end?
8. In the network of Question 1, vertex colors and contours indicate the group to which a vertex is affiliated. Count each brokerage role that vertex v_7 may play in this network.

7.8 Assignment

In our discussion of the triad, we compared complete and nearly complete triads. We stressed the opportunities that the incomplete structure offers to the person in the middle and the constraint exercised by the complete triad. Especially in the case of public behavior, that is, behavior that cannot be concealed from people in other groups, it has been argued that membership in several different cliques is very stressful because it obliges a person to conform to the (supposedly different) sets of norms

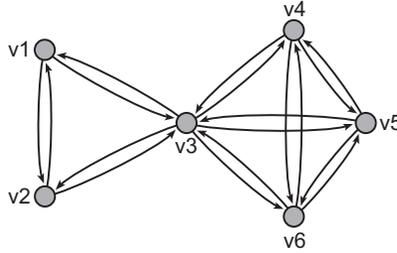


Figure 74. Two overlapping cliques.

of the different cliques. In this position, a person has very little room to maneuver.

In particular instances, this hypothesis contradicts the structural holes argument. If a person is a member in several different cliques, there are structural holes between the cliques, which may be exploited. In the network depicted in Figure 74, for example, vertex v_3 may exploit structural holes between the other members in the 3-clique (v_1 and v_2) and the other members in the 4-clique (v_4 , v_5 , and v_6). According to the structural holes hypothesis, vertex v_3 is least constrained. According to the hypothesis of overlapping cliques, however, v_3 is most constrained because it is a member in two cliques.

Having two competing hypotheses, it is interesting to see which one applies in a particular situation. The case is a small hi-tech computer firm that sells, installs, and maintains computer systems. The data file `hi-tech.net` contains the friendship ties among the employees, which were gathered by means of the question: Who do you consider to be a personal friend? Three employees (Fran, Quincy, and York) did not return the questionnaire. Note that most friendship nominations are reciprocated, but not all (112 of 147).

Some months later, employees tried to unionize the firm: They sought support among the employees to let the union have a say in the firm. The three top managers (class 3 in the partition `hi-tech.union.clu`) and three employees who were not directly involved (class 2) were opposed to union certification of the firm. Five employees (class 1) were pro-union, but two of them (Chris and Ovid) did not actively advocate the pro-union position. At the end, the proposal to unionize the firm was voted down. Chris resigned from the firm ten days before the vote because he did not want to participate in it. He rejoined the firm two days after the vote.

Analyze these data, which are joined in the project file `Hi-tech.paj`, and argue whether they support the structural holes argument or the overlapping cliques hypothesis. For the analysis of cliques, review Section 3.6 in Chapter 3. Pay attention to the position and behavior of Chris in

particular. In addition, analyze the brokerage roles if the groups are defined by their stance toward unionization (`hi-tech.union.clu`) and find out whether this explains Chris's behavior.

7.9 Further Reading

- The example is taken from J. H. Michael, "Labor dispute reconciliation in a forest products manufacturing facility." *Forest Products Journal* 47 (1997), 41–5.
- G. Simmel explains his ideas about triadic configurations in "Individual and society." In *The Sociology of Georg Simmel* (New York, NY: The Free Press, 1950, a translation of *Soziologie: Untersuchungen über die Formen der Vergesellschaftung*, Berlin: Duncker & Humblot, 1908). A recent approach from a different angle can be found in H. C. White, *Identity and Control: A Structural Theory of Social Action* (Princeton, NJ: Princeton University Press, 1992).
- Read more on social capital in Chapter 5 of A. Degenne and M. Forsé's *Introducing Social Networks* (London: SAGE, 1999) or N. Lin's *Social Capital: A Theory of Social Structure and Action* (Cambridge: Cambridge University Press, 2001).
- Mark Granovetter's article "The strength of weak ties" (*American Journal of Sociology* 78 [1973], 1360–80) is the source of the strength-of-weak-ties hypothesis. The second edition of his book *Getting a Job: A Study of Contacts and Careers* (Chicago, IL: University of Chicago Press, 1974, 1995) includes an appendix with surveys and analyses research based on this hypothesis.
- The theory of structural holes was introduced by R. S. Burt in his book *Structural Holes: The Social Structure of Competition* (Cambridge, MA: Harvard University Press, 1992), which contains applications of this theory to careers of managers and to the profitability of business sectors. We use the formulae presented in the second chapter of this book.
- The five types of brokerage roles were proposed by R. V. Gould and R. M. Fernandez in "Structures of mediation: A formal approach to brokerage in transaction networks." In *Sociological Methodology* (San Francisco, CA: Jossey-Bass [1989], pp. 89–126).
- The example used in the assignment as well as the theory of constraint by overlapping cliques stems from D. Krackhardt's, "The ties that torture: Simmelian tie analysis in organizations" (*Research in the Sociology of Organizations* 16 [1999], 183–210).

7.10 Answers

Answers to the Exercises

- I. When Pajek detects the bi-components but ignores bridges, it identifies a cut-vertex only if it belongs to two or more bi-components, that is, if bi-components intersect at this vertex. In the strike network, this is the case only for Norm and Sam, as one can tell from the contours in Figure 64. Frank is not part of a bi-component, so he is put in class 0 in both partitions produced by the *Bi-Connected Components* command, and he is not included in the hierarchy of bi-components.
- II. The aggregate constraint on Norm and Bob can be determined with the *Network > Create Vector > Structural Holes* command. This command produces a vector containing the aggregate constraint on vertices. Open this vector in a View/Edit screen (e.g., use the magnifier button left of the *Vector* drop-down menu) and look for Norm and Bob. Their aggregate constraint is 0.20 and 0.24, respectively.

Extracting the subnetwork of Norm's neighbors, we find one link among six neighbors, so a density of 0.07. The subnetwork of Bob's seven neighbors contains three links: a density of 0.14. Both Bob's constraint and his egocentric density are higher than Norm's, so they match in this respect. Norm has more opportunities to broker.

Answers to the Questions in Section 7.7

1. The lines between v13 and v2, v14, and v15 are bridges. If you remove one of these lines, v2, v14, or v15 becomes an isolate, so the network contains two components instead of one.
2. Answer b is correct. The network contains three cut-vertices: v4, v7, and v13. Removal of any of these vertices disconnects the network. Cut-vertex v4 belongs to two bi-components: One bi-component contains vertices v1, v3, v4, v12, and v16, and the other bi-component consists of v4, v7, and v13. Cut-vertex v7 belongs to the latter bi-component and to the third bi-component with vertices v5, v6, v7, v8, v9, v10, and v11. Cut-vertex v13 is part of the second bi-component and links it with the bridges (see Question 1) toward vertices v2, v14, and v15. These bridges and the vertices incident to them, however, are not bi-components, so the total number of bi-components is three, not six.
3. To reduce the control of a cut-vertex, you must join two or more bi-components into one bi-component. A tie between members of the two largest bi-components, which are not cut-vertices, produces this effect, for instance, between vertices v3 and v6. Now v4 and v7 are no longer cut-vertices.

4. Statement a is correct. A vertex that is incident to a bridge can have fewer than two neighbors only if its only neighbor lies at the other side of the bridge. The vertex, then, is a “hanger,” such as v2 in the network of Question 1. If you remove this vertex, you create no new components. If the vertex has two or more neighbors, however, it mediates between the vertex at the other side of the bridge and its other neighbor(s). When you remove the vertex, the latter neighbor is disconnected from the network, so the number of components in the network increases. Statement b is not correct because a vertex in a bi-component may well be a cut-vertex in the network at large (e.g., vertices v4, v7, and v13 in the network of Question 1). Statement c is not correct because a subnetwork is not necessarily connected. It may consist of several components that are not one bi-component by definition. Finally, statement d is not correct because the two paths between a pair of vertices may share a vertex between the endpoints, for instance, the paths v16–v3–v4–v7 and v16–v1–v4–v13–v7 in the network of Question 1. This vertex (v4) is a cut-vertex, so the component is not a bi-component. Statement d would be correct if it read “two distinct paths,” meaning that the two paths share no vertex between the endpoints.
5. Answer c is correct. It is not ruled out that a strong family tie is a bridge in an information network, for instance, the tie between Frank and Gill in the strike network, so answer a is not correct. Strong and weak ties are defined on the basis of their frequency or intensity, not on their structural features, so it is not ruled out that strong ties occur outside cliques and weak ties occur inside cliques; therefore, answers b and d are incorrect. However, there is a statistical association between the property (strength) of a tie and its structural location, so we may say that weak ties are more likely to be bridges in general, which is answer c.
6. Vertex v13 is less constrained than vertices v4 and v7. Note, first, that these three vertices have the same degree: Each has five neighbors. As a consequence, the proportional strength of their ties is equal, namely, 0.20. Therefore, we do not have to bother with the proportional strength of ties, and we can simply count the number of structural holes around each vertex to find out who is least constrained. We count only one direct tie among the five neighbors of vertex v13, namely, between v4 and v7, so nine of the ten possible pairs of neighbors are not directly linked: They are separated by structural holes. Among the neighbors of v7, two pairs are directly linked, which leaves eight structural holes. Finally, three pairs of v4’s neighbors are directly linked, so there are seven structural holes. More structural holes means less constraint, so vertex v13 is least constrained.

7. It is not wise to withdraw from ties that are part of a complete triad because that allows a neighbor to play the *tertius* strategy against you. Vertex v_7 has only one tie outside a complete triad, namely, the tie with vertex v_{10} . He may safely withdraw from this tie because he is already connected to vertex v_9 , which is the most central member of the white group.
8. Vertex v_7 does not broker between members of its own group, so he cannot play the coordinator role. Vertex v_7 is connected to three vertices in the white group, two of which are directly connected, so he is an itinerant broker between v_{10} on the one hand and v_6 and v_9 on the other hand (four roles if we treat edges as bi-directional arcs). Vertex v_7 may mediate between v_4 , who is a member of his group, and three vertices in the white group: v_6 , v_9 , and v_{10} . Here, v_7 plays the representative or gatekeeper role, both of which occur three times. Finally, v_7 may mediate between three white neighbors and v_{13} , who is in the light gray group, and vice versa, which yields six liaisons.

Diffusion

Diffusion is an important social process. Administrators are interested in the diffusion of information and opinions, manufacturers seek the adoption of new techniques and products, and all of us have a vivid interest in not acquiring contagious diseases. Diffusion processes are being studied in the communication sciences, social psychology and sociology, public administration, marketing, and epidemiology.

In this chapter, we present diffusion processes from a network point of view. Diffusion is a special case of brokerage, namely, brokerage with a time dimension. Something – a disease, product, opinion, or attitude – is handed over from one person to another in the course of time. We assume that social relations are instrumental to the diffusion process: They are channels of social contagion and persuasion.

If personal contacts are important, then the structure of personal ties is relevant to the diffusion process and not just the personal characteristics that make one person more open to innovations than another. We investigate the relation between structural positions of actors and the moment at which they adopt an innovation.

8.1 Example

Educational innovations have received a lot of attention in the tradition of diffusion research. Our example is a well-known study into the diffusion of a new mathematics method in the 1950s. This innovation was instigated by top mathematicians and sponsored by the National Science Foundation of the United States as well as the US Department of Education. The diffusion process was successful because the new method was adopted in a relatively short period by most schools.

The example traces the diffusion of the modern math method among school systems that combine elementary and secondary programs in

Allegheny County, Pennsylvania. All those school superintendents who were in office at least two years were interviewed. They are the gatekeepers to educational innovation because they are in the position to make the final decision. The researchers obtained data from sixty-one of sixty-eight superintendents, fifty-one of whom had adopted the method by 1963 (84 percent).

Among other things, the superintendents were asked to indicate their friendship ties with other superintendents in the county with the following question: Among the chief school administrators in Allegheny County, who are your three best friends? The researcher analyzed the friendship choices among the thirty-eight interviewed superintendents who adopted the method and were in position at least one year before the first adoption, so they could have adopted earlier. Unfortunately, the researcher did not include the friendship choices by superintendents who received no choices themselves; they are treated as isolates. In the original network, some friendship choices are reciprocated and others are not (`ModMath_directed.net`), but we use the symmetrized network (`ModMath.net`), which is depicted in Figure 75. A line in this network indicates that at least one superintendent chooses the other as his friend.

As you may infer from Figure 75, adoption started in 1958 and all the schools researched had adopted by 1963. The year of adoption by a superintendent's school is coded in the partition `ModMath_adoption.clu`: 1958 is class (time) 1, 1959 is class (time) 2, and so on. The first adopter (`v1`) is a superintendent with many contacts outside Allegheny County but few friends within the county. He is a "cosmopolite" and cosmopolites usually are early adopters, but they are often too innovative to be influential in a local network.

Application

Draw>
Network +
First Partition

For a first visual impression of a diffusion process, open the Pajek project file `ModMath.paj` and draw the sociogram in the order of adoption time (see Figure 75; we manually added the years to the top of this figure).

Layers> In y
Direction

To do this, the adoption time of vertices must be specified in a partition (e.g., `ModMath_adoption.clu`). Draw the sociogram with vertex colors defined by the partition (*Draw> Network + First Partition* or *Ctrl-p*) and select the command *Layers> In y Direction* to arrange the vertices by adoption time. Note that this procedure is available only when a network with partition is drawn.

Layers>
Optimize
Layers in x
Direction

In most cases, the vertices are not optimally placed within each level. To improve their positions, use the *Optimize Layers in x Direction* command. You can let this command adjust all levels (i.e., classes), or you can restrict the optimization to a range of levels. Play around with the options (*Forward, Backward, Complete*) until you obtain a layout without lines that cross vertices with which they are not incident. In Figure 75, this was

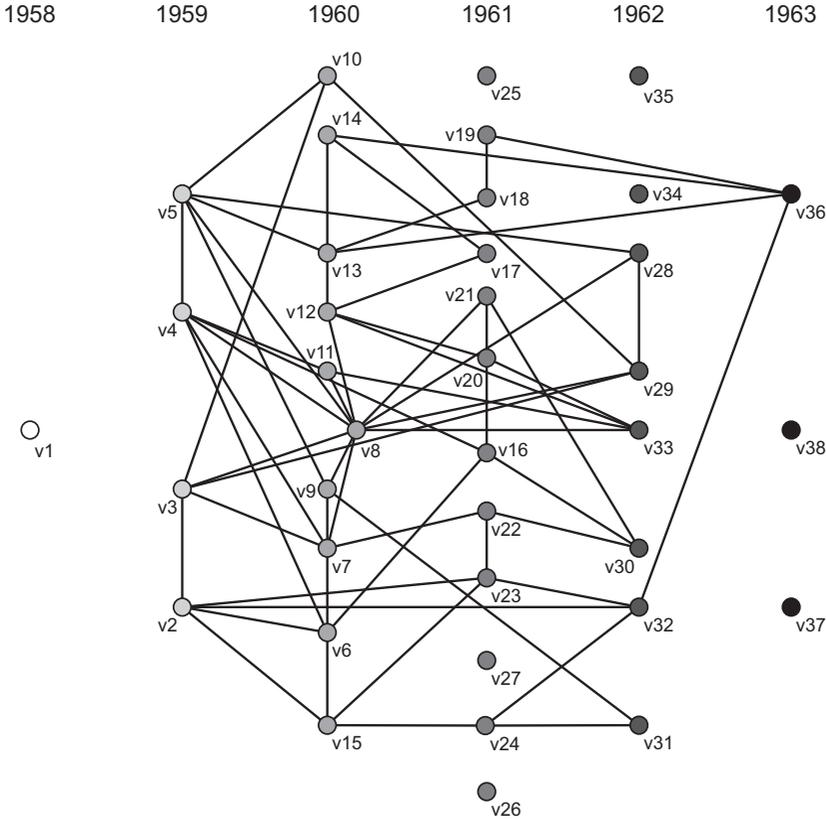


Figure 75. Friendship ties among superintendents and year of adoption.

not possible because superintendent v8 is connected to too many vertices in his adoption class, so we decided to move him away from the line of his class. Even for our small diffusion network, the sociogram needs a lot of fine-tuning, so you should not expect a clear picture if you are working with large networks.

Sometimes it helps to rearrange the vertices within a layer by hand. If you do this but you want to be sure that the vertices within a class remain aligned, activate the option *y* in the *Fix* menu of the Draw screen. Now, you can move vertices horizontally only.

The layers are drawn in the *y* direction: from the top down. In Figure 75, however, time flows from left to right on the *x*-axis, which is the standard way to represent time. We obtained this figure by rotating the standard layout of layers by 90 degrees. Select the command *Rotate 2D* from the *Options> Transform* submenu in the Draw screen. Type 90 in the dialog box captioned *Angle in degrees*, and press the OK button.

Move> Fix> y

[Draw]
Options>
Transform>
Rotate 2D

8.2 Contagion

Information is important to the diffusion of new opinions, products, and the like. In most societies, the mass media are central to the spreading of information, so we ought to pay attention to mass communication. Several models have been proposed for the process of mass communication, one of which is consistent with a network approach: the two-step flow model. According to this model, mass communication consists of two phases. In the first phase, mass media inform and influence opinion leaders. In the second phase, opinion leaders influence potential adopters within their communities or social systems.

Network models of diffusion focus on the second phase, assuming that opinion leaders use social relations to influence their contacts. Social ties are thought to be important because innovations are new, hence, risky. Personal contacts are needed to inform and persuade people of the benefits associated with the innovation. Note that salient social relations for spreading information may be different from relations used for persuasion. The relations most commonly investigated are advice and friendship relations.

Basically, network models see diffusion as a process of contamination, just like the spread of an infectious disease. Therefore, passing on an innovation via social ties is called social contagion. This perspective is backed by the empirical fact that many innovations diffuse in a pattern that is similar to the spread of infectious diseases. First, an innovation is adopted by few people, but their number increases relatively fast. Then, large numbers adopt, but the growth rate decreases. Finally, the number of new adopters decreases rapidly, and the diffusion process slowly stops. This diffusion pattern is characteristic for a chain reaction in which people contaminate their contacts, who contaminate their contacts in the next step, and so on.

The adoption of the modern math method is represented by a *diffusion curve* (Figure 76). The x -axis shows the moment of adoption, and the y -axis represents the *prevalence* of the innovation, which is the percentage of all interviewed superintendents who have adopted the modern math method by that year. Note that prevalence is represented by cumulative percentages, that is, the sum of all percentages of previous adopters: In 1958, 3 percent of the superintendents adopt and in 1959 another 10 percent adopt, so the cumulative percentage of adopters is 13 percent in 1959.

The diffusion curve has the logistic S-shape, which is characteristic of a chain reaction. We find a similar curve when we take a random network and choose a vertex as a source of contamination (the white vertex in Figure 77). When we assume that a vertex contaminates its neighbors at time 1, who contaminate their neighbors at time 2, and so on, we obtain the typical diffusion curve of Figure 78 (bold line). Note that the number

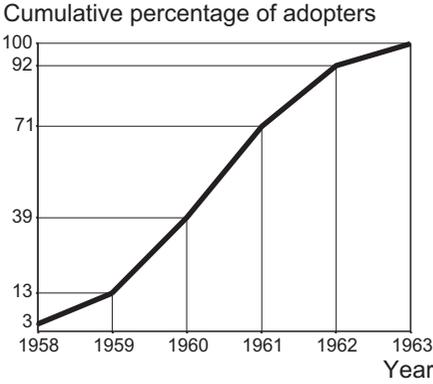


Figure 76. Adoption of the modern math method: diffusion curve.

of new adopters increases faster and faster in the first three steps (vertices with numbers 1, 2, and 3) and that the absolute number of new adopters decreases sharply after the fourth step. This example illustrates that contagion through network ties may explain the logistic spread of an innovation or a disease. If we find a diffusion curve that does not have the typical shape, it is quite unlikely that network ties are important to the diffusion process, and diffusion is probably propelled predominantly by other forces such as mass media campaigns.

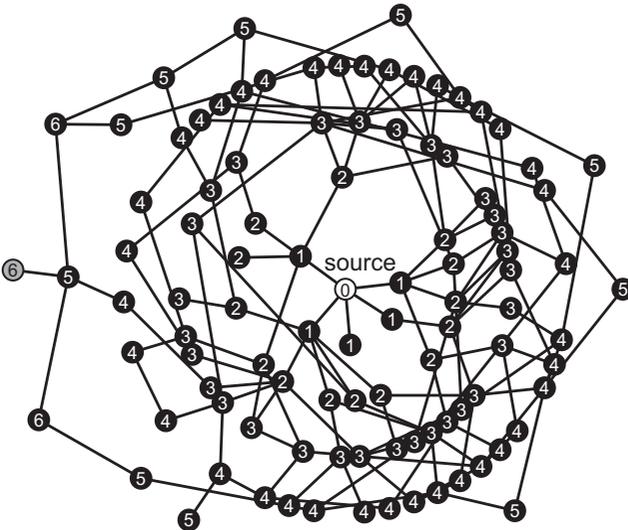


Figure 77. Diffusion by contacts in a random network ($N = 100$, vertex numbers indicate the distance from the source vertex).

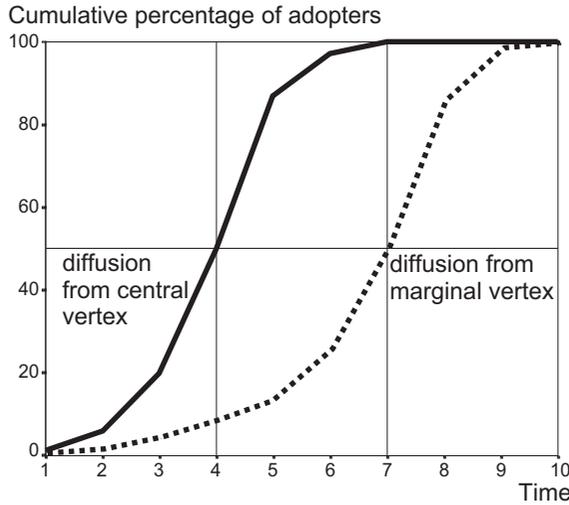


Figure 78. Diffusion from a central and a marginal vertex.

When contagion drives the diffusion process, the structure of an information or contact network conditions the diffusion of information, innovations, diseases, and so on. Using the measures introduced in previous chapters, some broad hypotheses are easily derived as follows:

- In a dense network, an innovation spreads more easily and faster than in a sparse network.
- In an unconnected network, diffusion will be slower and less comprehensive than in a connected network.
- In a bi-component, diffusion will be faster than in components with cut-points or bridges.
- The larger the neighborhood of a person within the network, the earlier she or he will adopt an innovation.
- A central position is likely to lead to early adoption.
- Diffusion from a central vertex is faster than from a vertex in the margins of the network.

The *adoption rate* is the number or percentage of new adopters at a particular moment.

The speed of the diffusion process is measured by the *adoption rate*, which is the number or percentage of new adopters at a particular moment. It is easy to see that the adoption rate is higher when an innovation spreads from a central vertex than when it starts at a marginal vertex. Figure 78 shows the diffusion curves for the diffusion from the

Table 10. *Adoption in the modern math network*

Cluster	Freq	Freq%	CumFreq	CumFreq%	Representative
1	1	2.63	1	2.63	v1
2	4	10.53	5	13.16	v2
3	10	26.32	15	39.47	v6
4	12	31.58	27	71.05	v16
5	8	21.05	35	92.11	v28
6	3	7.89	38	100.00	v36
SUM	38	100.00			

central white source vertex in Figure 77 (bold line) and from the peripheral gray vertex (dotted line). Both curves have the typical shape, but it takes considerably more time for a diffusion to reach half or all of the population when it is triggered by a vertex in the periphery.

The hypotheses presented in the list highlight the impact of network structure on the diffusion process. We should note, however, that personal characteristics and the type of innovation also influence the rate of adoption. The perceived risk of an innovation, its perceived advantage over alternatives, and the extent to which the innovation complies with social norms that govern the target group determine whether it is adopted quickly, reluctantly, or not at all. A risky innovation, for instance, will diffuse slower regardless of the network’s density and connectivity.

Application

The diffusion curve is constructed from a simple frequency tabulation of adoption time, which is displayed by the *Partition> Info* command (Table 10). The table shows the cumulative relative frequencies that are plotted on the *y*-axis in the chart of Figure 76. The cluster numbers represent the moments that are displayed on the *x*-axis. Note that the table and chart are basic statistical techniques, which may be produced in any statistical software package or spreadsheet.

Partition> Info

Exercise I

Create a simple random network with fifty vertices that have an outdegree of 1 or 2 (use the *Network> Create Random Network> Vertices Output Degree* command with a minimum outdegree of 1 and a maximum outdegree of 2 and no multiple lines). Pick a vertex as the source of a diffusion process, and determine the adoption time of all vertices and the adoption rate at each point in time, assuming that a vertex will adopt at the first time point after it has established direct contact with an adopter. Note that the adoption time of a vertex is equal to its distance (see Chapter 7) from the source vertex under this assumption. Ignore the direction of the lines in the network.

*Network>
Create Random
Network>
Vertices Output
Degree*

8.3 Exposure and Thresholds

In the previous section, we assumed that every person is equally susceptible to contagion. One infected neighbor is enough to get infected; friendship with one adopter is enough to persuade someone to adopt. This is not very realistic because some people are more receptive to innovations than other people. There are two different ways to conceptualize the innovativeness of people, namely, relative to the system and relative to their personal networks: adoption categories and threshold categories.

Adoption categories classify people according to their adoption time relative to all other adopters. These typologies are very popular in product marketing. A standard classification distinguishes between the early adopters (the first 16 percent who adopt), the early majority (the next 34 percent), the late majority (the next 34 percent), and late adopters or laggards (the last 16 percent to adopt). To classify people, we have to know only their adoption time. Then, we can simply mark the first 16 percent of all adopters as early adopters, and so on. This classification is useful for marketing purposes because it enables the marketing manager to identify the social and demographic characteristics of early adopters.

In the modern math example, early adopters are characterized by higher professionalism ratings and more accurate knowledge about the spread of educational innovations in their district. In addition, the superintendents who adopted early were not recruited from the school staff; they came from outside.

We concentrate on the second approach to innovativeness, *threshold categories*, which considers the personal network of actors. The network model of diffusion is based on contagion: An adopter spreads the innovation to his or her contacts. It is quite natural to assume that the chance that a person will adopt increases when he or she is linked to more people who already have adopted, that is, when he or she is exposed to more adopters. Hearing about the benefits of an innovation from different sources will persuade a person to adopt. The amount of exposure varies over time and among individuals, which explains that some people adopt early although they are not close to the sources in a diffusion process. The exposure of a person is expressed as a proportion, so it may be thought of as a chance to adopt.

The *exposure* of a vertex in a network at a particular moment is the proportion of its neighbors who have adopted before that time.

Figure 79 shows the modern math network with the exposure of vertices in 1959 indicated by vertex size and by the numbers in brackets. Note that invisible vertices have 0.00 exposure: None of their neighbors

we assume that some people are easily persuaded (e.g., they need only one contact with an adopter), whereas others are talked into adopting an innovation with difficulty. Some people are more susceptible than others, which is an established fact for media exposure as well as social exposure.

In the network model of diffusion, the *innovativeness* of a person is perceived as his or her threshold to exposure. An individual's *threshold* is the degree of exposure that he or she needs to adopt an innovation. Now, differences between individual thresholds may account for the fact that only some of the people adopt who are equally exposed.

The *threshold* of an actor is his or her exposure at the time of adoption.

In our example, four superintendents (gray vertices in Figure 79) adopted the new math method in 1959. They exposed thirteen superintendents (white vertices) to their experience with this method, and eight of them adopted the method in the next year. However, five superintendents adopted two or three years later. Why? Each of the exposed superintendents who adopted after 1960 has one or two friends among the colleagues who adopted in 1960 or 1961. By the time they adopted, these friends had also adopted, so their exposure was higher than at the end of 1959. According to the threshold hypothesis, their exposure had not reached the required threshold in 1959, but it did in 1960 or 1961. This explains why they adopted later.

At the end of 1959, for example, one of the four friends of superintendent v23 had adopted the modern math method, so his exposure was 0.25. In 1960, one more friend (v15) adopted, and his exposure increased to 0.5. Then, superintendent v23 adopted, so we assume that his threshold was 0.5 or somewhere between 0.25 and 0.5.

We should note that individual thresholds are computed from the diffusion network after the fact: They are predictions with hindsight, and they are not very informative by themselves. It is important to make sense of them or to validate them, which means that they should be associated with other indicators of innovativeness, for instance, adoption time or personal characteristics.

Thresholds indicate personal innovativeness, a lower threshold means more innovative, and we expect innovative people to adopt an innovation earlier than noninnovative people. Therefore, individual thresholds must be related to adoption time: Innovative people have low thresholds and adopt early. If we find such a relation, we obtain some support for the assumption that individual thresholds indicate innovativeness.

At least to some extent, however, a positive relation between adoption time and individual thresholds is an artifact of the contagion model that we use. The first adopters cannot be exposed to previous adopters, so

their thresholds are zero by definition. Within the network of adopters, the last adopters are very likely to be connected to previous adopters, so their exposure and thresholds are high at the time of adoption. When measurement of adoption time is restricted to a small number of moments, this will automatically produce a relation between individual thresholds and adoption time.

Therefore, it is also important to compare individual thresholds to external characteristics of the actors that usually indicate innovativeness. In general, innovativeness and low thresholds are supposed to be related to broad media use, many cosmopolitan contacts (contacts outside your local community), a high level of education, and high socioeconomic status.

Application

Let us compute exposure levels in the modern math network at one moment, for instance, at time 2, 1959 (see Figure 79). The procedure consists of several steps, which illuminate the calculation and exact meaning of the exposure concept. We assume that the network is undirected. If not, symmetrize it (*Network> Create New Network> Transform> Arcs→Edges> All* and remove any multiple lines, e.g., take the sum or minimum line value).

First, we identify the adopters in the network at the selected time, which is 1959 or time 2 in our example. Make a binary partition from the adoption time partition where adoption times 1 and 2 are assigned a score of 1 (adopted), and others are assigned a score of 0 (not adopted yet) with the *Partition> Binarize Partition* command, selecting classes 1–2 in the dialog box. In Figure 79, the adopters are gray and the nonadopters are white. Then, turn this partition into a vector to use it for computation (*Partition> Copy to Vector* or press F5).

Second, compute the number of adopters in each actor's neighborhood with the command *Operations> Network + Vector> Neighbours> Sum> Input, Output, or All*. A dialog box appears that asks whether a vertex should be included in its own neighborhood; answer no. Pajek does not count the number of neighbors, but it sums the class numbers of the neighbors of a vertex. Because we use a binary partition in which an adopter has class number 1 and a nonadopter has class number 0, this sum is equal to the number of adopters in the neighborhood. It is a little trick, but it works.

Third, the number of adopters in the neighborhood of a vertex must be divided by its total number of neighbors because we defined exposure as the percentage of neighbors who have adopted. The division can be done in the *Vectors* menu. The vector we just made must be selected as the first vector. Next, we must make a vector with the total number of neighbors of a vertex. Recall that the degree of a vertex in a simple undirected

Network>
Create New
Network>
Transform>
Arcs→Edges >
All

Partition>
Binarize
Partition

Partition>
Copy to Vector

Operations>
Network +
Vector>
Neighbours>
Sum

Vectors>
Divide
(First/Second)

network specifies the number of neighbors of a vertex, so we can make a degree vector in the usual way (*Network*> *Create Vector*> *Centrality*> *Degree*). This vector must be used as the second vector. Finally, we divide the number of adopters in a vertex's neighborhood by the total number of neighbors with the *Vectors*> *Divide (First/Second)* command. Now, we obtain a vector with the exposure of vertices at the end of 1959 (time 2).

[Main]
Options> Read
- Write> 0/0

Note that the computation of exposure is not straightforward if the network contains isolated vertices. An isolated vertex has no neighbors, so its degree is 0. The division described in the previous paragraph would ask Pajek to divide by 0, which is mathematically incorrect. In this case, Pajek assigns the value 0 to the exposure of the vertex, that is, if the default setting of 0/0 to 0 was not changed in the *Options*> *Read - Write* submenu of the Main screen.

Macro> Play

The calculation of exposure consists of a considerable number of steps. If you want to compute exposure at several points in time, you have to repeat these steps over and over again. This is not very efficient, so Pajek contains the possibility of executing a number of steps in one command, which is called a *macro*. A macro is a file that consists of a list of commands that are executed when you play the macro in Pajek. We prepared the macro `exposure.mcr`, which is located in the directory with the data accompanying this chapter. You can execute the macro by dragging and dropping this file to a Pajek screen or clicking on the *Play* command in the *Macro* menu and selecting the file `exposure.mcr`. Make sure that the original undirected network and the adoption time partition are selected before you execute the macro. When you open this file, Pajek starts to execute the commands. It displays the dialog box *Select clusters* that allows you to select the first time of adoption and the time for which you want to compute exposure (enter 1–3). On completion of the macro, several new partitions and vectors have been created, and the last vector contains the exposure at the requested time.

Macro>
Record

Macro> Add
Message

Creating a macro yourself is fairly simple. In essence, Pajek records all commands that you execute between the first and second time you click the *Record* command in the *Macro* menu. It prompts for a filename with the extension `.mcr` in which to store the recorded commands. While recording, you can add messages to the macro (*Macro*> *Add Message*) that will be displayed in the Report screen when the macro is played afterwards. Make sure that you have the relevant network, partition, and vector selected in the drop-down menus before you record the macro, and check the results when you play it for the first time.

Now that we have computed the exposure at one time, let us turn our attention to the calculation of thresholds. The *threshold* of a vertex is the proportion of its neighbors who have adopted before ego does, so we have to divide the number of prior adopters among the neighbors of a vertex by the size of its neighborhood. The computation of thresholds

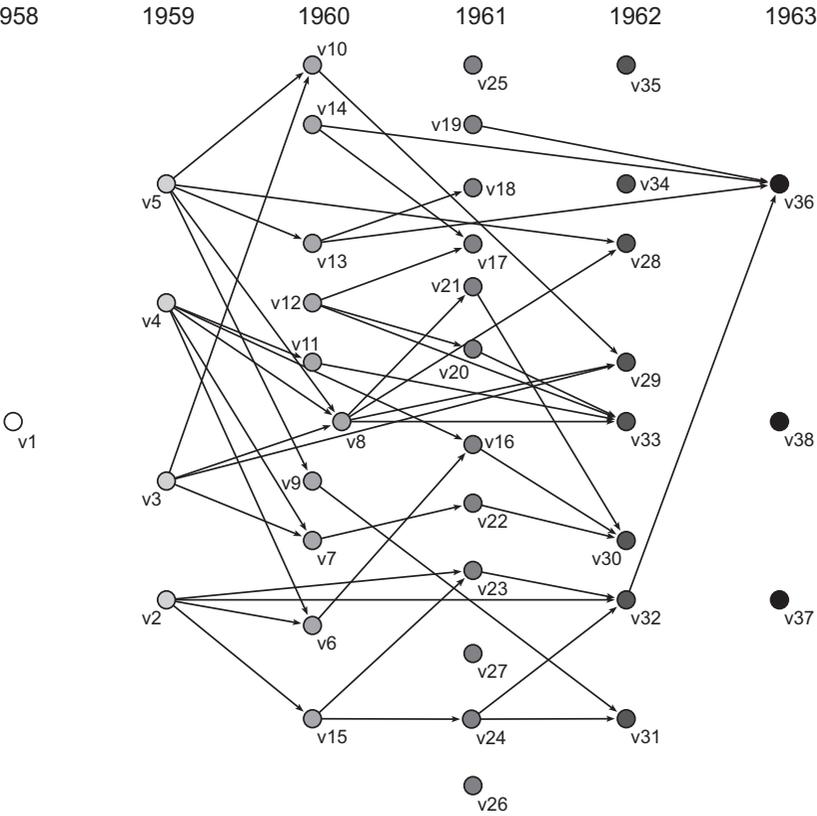


Figure 80. Modern math network with arcs pointing toward later adopters.

is fairly simple once you realize that the number of neighbors who have adopted prior to ego is equal to the indegree of ego in a directed network in which each line points from an earlier adopter to a later adopter. Figure 80, for instance, shows the modern math network if edges are replaced by arcs that point to later adopters. If social ties are used to spread the innovation, the arcs represent the direction of the spread. Note that ties within an adoption class are omitted because they are not supposed to spread the innovation.

In Pajek, we can change an undirected network (e.g., the modern math network) into a directed network with all arcs pointing from an earlier adopter to a later adopter with the commands in the *Operations*> *Network + Partition*> *Transform*> *Direction* submenu. The commands are located in the *Operations*> *Network + Partition* submenu, so you need a network and a partition that specifies the (adoption) classes to which the vertices belong. There are two commands: *Lower*→*Higher*

Operations>
Network +
Partition>
Transform>
Direction

and *Higher*→*Lower*. The first command replaces an edge in an undirected network by an arc that points toward the vertex with the higher class number. In our case, the partition contains adoption time classes so the *Lower*→*Higher* command produces arcs toward later adopters. This command issues a dialog box asking whether lines within classes must be deleted. In a directed diffusion network, we do not want to have lines within adoption classes normally, so answer yes. Now, we obtain the network shown in Figure 80. Applied to a directed network, the *Direction* procedure selects the arcs that conform to the selected option (lower to higher or higher to lower).

```
Network>
Create Vector>
Centrality>
Degree> Input
```

```
Vectors>
Divide
(First/Second)
```

```
Cluster>
Create Empty
Cluster
```

```
Partition>
Make Cluster>
Vertices from
selected
Clusters
```

```
Operations>
Network +
Vector> +
Cluster>
Diffusion
Partition
```

Now, we can simply compute the thresholds of all vertices in the diffusion network by dividing the indegree of vertices in the transformed directed network by their degree in the original undirected network provided that both networks contain neither multiple lines nor loops. Select the vector with indegree in the directed network as the first vector, and select the corresponding vector for the undirected network as the second vector, and divide the first by the second to obtain a vector with the individual thresholds. Make sure that a division of 0 by 0 (no neighbors) yields 0 in the *Options*> *Read – Write* menu.

If you have threshold values for adoption on the vertices in a network and one or more vertices that introduce the innovation in the network, it is possible to predict at what time vertices will adopt. That is, if one assumes that diffusion is solely driven by network ties, and a vertex adopts once the proportion of previous adopters among its neighbors exceeds its threshold. The Pajek command *Operations*> *Network + Vector*> *+Cluster*> *Diffusion Partition* accomplishes this by calculating an adoption partition from a network, a vector of individual thresholds, and a cluster specifying the vertices that adopted first. A cluster is merely a set of vertex numbers that you can create by hand (command *Cluster*> *Create Empty Cluster*) or extract from a partition (command *Partition*> *Make Cluster*> *Vertices from selected Clusters*). In the modern math network, it makes sense to mark vertices v1 to v5 as the initial adopters, which is accomplished with the *Partition*> *Make Cluster*> *Vertices from selected Clusters* command, selecting the clusters 1 and 2 from the `ModMath_adoption.clu` partition. If you use the original undirected network and the threshold vector described earlier (see also Exercise II), you can simulate the diffusion process. The diffusion partition calculated by the command can be drawn in the usual way. You will notice that some vertices are expected to adopt earlier or later than they actually did. Network ties and the calculated thresholds do not completely account for the empirical diffusion process.

Exercise II

Compute the thresholds of the vertices in the modern math diffusion network as explained in this section. Is the threshold higher for vertices that adopt later as one would expect when thresholds really matter?

8.4 Critical Mass

Some diffusion processes are successful because almost everybody in the target group adopts the innovation. For instance, the modern math method was adopted by fifty-one of sixty-one superintendents in Allegheny County within a period of six years. Diffusion, however, may also fail because too few people adopt and spread the innovation. Once again, a biological metaphor is illuminating: A bacteria may either succeed to overcome the resistance of the human body and develop into a disease or does not gain the upper hand and is oppressed and finally eliminated by antibodies. The spread of a disease has a critical limit: Once it is exceeded, the bacteria multiply quickly.

The *critical mass* of a diffusion process is the minimum number of adopters needed to sustain a diffusion process.

In the diffusion of innovations theory, a similar limit is hypothesized to exist. It is called the critical mass of a diffusion process, and it is defined as the minimum number of adopters needed to sustain a diffusion process. In the first stage of a diffusion process, outside help is needed (e.g., an advertisement campaign); but once a sufficient number of opinion leaders have adopted, social contagion fuels the process and causes a chain reaction that ensures wide and rapid diffusion. Then, no more outside input to the diffusion process is required.

The critical mass of a particular diffusion process is difficult to pinpoint, so it is hard to prove that it exists and when it occurs. Recall that the two-step flow model combines contagion with external events. We need detailed information about the effects of external events, such as media campaigns, versus the effect of social contagion on the diffusion process to know when critical mass is reached. Only afterward, we may evaluate whether a diffusion process was successful. We present some approaches that try to overcome this problem.

There is an empirical rule of thumb that tells us something about the number of people who will eventually adopt an innovation. In many diffusion processes, a particular phenomenon occurs when the innovation has been adopted by 16 (or 10 to 20) percent of all people who will adopt eventually: The acceleration of the adoption rate decreases although the adoption rate still increases in absolute numbers. This is known as the first *second-order inflection point* of the S-curve.

In the modern math network, for instance, the number of new adopters (adoption rate) rises from 1 to 4 from 1958 to 1959 (see the fourth column in Table 11), which is three more than the number of adopters in 1958, so there is an acceleration of 3 (see the fifth column in Table 11). Note that

Table 11. *Adoption rate and acceleration in the modern math diffusion curve*

Time	Cum% of Adopters	Cum # of Adopters	Adoption Rate	Acceleration
↓ 1958 (1)	2.63	1	1	3
↓ 1959 (2)	13.16	5	4	6
↓ 1960 (3)	39.47	15	10	2
↓ 1961 (4)	71.05	27	12	-4
↓ 1962 (5)	92.11	35	8	-5
↓ 1963 (6)	100.00	38	3	-3
↓			0	

adoption rates are placed between the moments because they reflect the change between two measurements. In the next year, ten superintendents adopt, which is an even larger acceleration, but in 1961 the acceleration drops to 2 because the number of new adopters grows only from 10 to 12; the number of new adopters still rises, but it rises less sharply. In 1959, we may conclude, the acceleration of the adoption rate is highest, and we can see that 13 percent of all adopters have adopted (see the column “Cum% of adopters” in Table 11) as predicted by the rule of thumb.

Because of this empirical relation between the first second-order inflection point of the diffusion curve and the final spread of an innovation, diffusion analysts say that critical mass is attained when the diffusion curve reaches this inflection point. In this approach, any diffusion process in which the adoption rate first accelerates and then declines is thought to be driven by the chain reaction characteristic for contagion models. Social contagion is assumed to take over the diffusion process at this point, so we may conclude that the process has reached its critical mass.

A similar argument has been made for the first-order inflection point of the logistic diffusion curve, which is the period with the highest adoption rate, that is, the largest absolute increase in new adopters. Usually, the first-order inflection point occurs when approximately 50 percent of all eventual adopters have adopted. In the modern math network, the highest adoption rate is 12, and it was realized between 1960 and 1961. In this period, the percentage of adopters rose from 39 to 71 percent.

We ought to realize that this approach completely presupposes the relation between contagion and critical mass; it does not prove that critical mass occurs, it merely assumes so. Nevertheless, it is useful for

practical purposes. We may monitor the diffusion process and watch out for the moment in which the first decline in growth acceleration occurs (but we should ignore incidental declines). When it occurs, we may estimate the final number of adopters at about five to ten times the number of adopters at the time of the largest increase because about 10 to 20 percent has adopted then. If this estimated number of adopters is not enough according to our target, we can try to boost the diffusion process with additional media campaigns and the like. If this leads to acceleration of the diffusion, the critical mass becomes larger and the diffusion process will probably reach more people in the end. However, we have no guarantee that this will actually happen. After all, we are working with a simple rule of thumb.

In another perspective, a diffusion process is assumed to attain its critical mass when the most central people have adopted. Once they have adopted, so many actors in the network are exposed to adopters that many individual thresholds have been reached and an avalanche of adoptions occurs. Betweenness centrality seems to be associated with critical mass in particular. Targeting the actors with highest betweenness centrality is a good strategy for launching an innovation. In general, the position of the first adopters in the network is relevant to the diffusion process. If the first adopters are central and directly linked, their neighbors have higher exposure rates, so they are more likely to adopt.

Why does critical mass boost the diffusion of an innovation? On the one hand, the reason may be purely quantitative: Once a sufficient number of well-connected people have adopted, enough people are exposed to the innovation to adopt, after which even more people are exposed. This is the mechanism we described for the case that the central actors adopt. On the other hand, reaching the critical mass has been thought of as a qualitative change to the system, namely, a sudden lowering of individual thresholds. During the diffusion process, individual thresholds may be lowered as a consequence of the rate of adoption in the entire social system. People are supposed to monitor their social system. If they perceive wide acceptance of an innovation, they feel confident or even obliged to adopt it. Lower thresholds lead to easier adoption, so the diffusion process strengthens itself, and it will most probably not wither away.

The lowering of thresholds is expected to occur particularly when actors are interdependent with regard to an innovation. New communication technology products (e.g., buzzers or Short Message Service [SMS]) are a case in point. When more people have one, their benefits and value increase. The first adopters can reach few people with the new communication products, but the late majority can contact many more users. This kind of innovation is called an *interactive innovation*. Even in the case of noninteractive innovations, such as the modern math method, the qualitative mechanism may be operative. Superintendents may be persuaded

to adopt the new method because they know that most of their peers have adopted, regardless of the number of adopters in their circle of friends.

A *threshold lag* is a period in which an actor does not adopt although he or she is exposed at the level at which he or she will adopt later.

The lowering of thresholds when critical mass is attained in the diffusion process may explain the occurrence of a threshold lag, that is, a period in which the exposure has reached the individual threshold but the individual does not adopt. In this case, adoption occurs after the critical mass is reached, and the individual's threshold is lowered. In the modern math network, in 1960, superintendents v28 and v29 reached the level of exposure at which they would eventually adopt because all of their friends had adopted by that year. However, they did not adopt immediately in 1961. There is a delay of one year, which is their threshold lag. Perhaps, the diffusion process reached its critical mass in 1961, which lowered their thresholds and induced them to adopt in 1962.

We should note that this approach to thresholds and threshold lags does not prove that individuals have certain thresholds and threshold lags; it merely defines them in a particular way. In an empirical diffusion network, we can always compute an actor's exposure at the moment of adoption (threshold) and how long this actor had been exposed at this level before he or she adopted (threshold lag). However, this does not rule out the possibility that the individual threshold was actually lower and his or her threshold lag was longer. We should also consider the possibility that the individual's original threshold was even higher than the exposure at the time of adoption, so there was no threshold lag at all, namely, when the diffusion process reached a critical mass lowering individual thresholds or when outside events (e.g., a media campaign) convinced individuals to adopt before they reached their thresholds. The launching of *Sputnik I* in October, 1957, for example, is known to have spurred a wave of innovations in science and education in the United States.

Therefore, we need empirical data supplementing the diffusion network data to validate the actors' thresholds, notably, psychological information, relevant social characteristics, or a record of past adoptions. Then we can estimate the most likely adoption time and compare it to the actual adoption time to determine threshold lags and critical mass effects. If threshold lags coincide with external events, it is likely that these events have an impact on the diffusion process. In contrast, if a media campaign does not coincide with the end of relatively many lags, it is probably not very influential.

Table 12. *Fragment of Table 11*

Time	Cum% of Adopters	Cum # of Adopters	Adoption Rate	Acceleration
↓ 1958 (1)	2.63	1	1	3
↓ 1959 (2)	13.16	5	4	6
↓ 1960 (3)	39.47	15	10	2

Application

The absolute adoption rates and their acceleration can be calculated from the frequency tabulation of adoption times, discussed in Section 8.2. The absolute growth or adoption rate is just the number of new adopters between two moments (e.g., ten superintendents adopt the modern math method between the end of 1959 and the end of 1960). In Table 12, adoption rates are again placed between the moments because they reflect the change between two measurements. The acceleration of the adoption rate at a particular moment is the difference between the adoption rate directly before and after this moment: subtract two successive adoption rates. In 1959, the acceleration is 6, because ten schools adopted in the year after the end of 1959 and four adopted in the year before. It is easy to spot the moment in which the acceleration starts to decrease while the absolute growth (adoption rate) is still increasing.

If the critical mass is the first moment at which the most central vertices have adopted, we may simply calculate the betweenness centrality of the vertices and check at which time all or most of the central actors have adopted (*Network> Create Vector> Centrality> Betweenness*). We advise computing betweenness centrality in the undirected network, so symmetrize a directed network first (*Network> Create New Network> Transform> Arcs→Edges> All*, avoid multiple lines). List the most central vertices with the *Vector> Info* command by entering a positive number in the dialog box captioned *Highest/lowest or interval of values*. You can check their adoption time in the adoption time partition or in the layered sociogram (see Section 8.1). In our example, the most central superintendents (v8, v13, and v12) are found among the adopters in 1960 or 1959 (v5), so critical mass was reached in 1960. Note, however, that not all central actors adopted then: The fifth (v36) and sixth (v32) most central superintendents adopted as late as 1963 and 1962.

For the calculation of threshold lags, it is important to note that a vertex reaches its threshold when all prior adopters in its neighborhood have

Partition> Info

*Network>
Create Vector>
Centrality>
Betweenness*

*Network>
Create New
Network>
Transform>
Arcs→Edges>
All*

Vector> Info

adopted. In the modern math network, for instance, superintendent v28 reached his threshold of 0.67 in 1960, when superintendent v8 adopted. At the end of 1960, all prior adopters in his neighborhood had adopted (superintendents v5 and v8). The third contact in his friendship network, superintendent v29, is irrelevant to the threshold of v28 because he or she adopted at the same time as v28. The threshold lag is calculated as the difference between the time of adoption of an actor (his or her class number in the adoption partition) and the maximum adoption time of the neighbors that adopted before him or her. We have to subtract 1 from the threshold lag if we consider exposure at one moment to cause adoption at the next moment. The threshold lag of v28 is equal to $1962 - 1960 - 1$, which is one year.

Threshold_lag
.mcr

Options> Read
- Write>

Ignore Missing
Values in menu
Vector and
Vectors

In Pajek, the last contact of a vertex to adopt prior to this vertex is easily found in the directed network that we introduced to calculate thresholds. When all lines point from earlier to later adopters, the neighbor with the highest adoption time on the input side of a vertex is its closest predecessor. However, the computation of thresholds includes some tricks we do not want to discuss here, so we prepared a macro (*Threshold_lag.mcr*) you can use to obtain threshold lags from the original undirected network and adoption time partition. When you play this macro (see Section 8.3), some new networks, partitions, and vectors are created. The last vector contains the threshold lags. Be careful to check *Options> Read - Write> Ignore Missing Values in menu Vector and Vectors* before running this macro.

In the modern math network, we find threshold lags only for superintendents v28 and v29 (one-year lag). This small number of lags does not suggest that external events or critical mass influenced the diffusion process. Most vertices adopted right after one or more of their friends had adopted, which is in line with the simple exposure and threshold model.

8.5 Summary

Innovations and infectious diseases diffuse in a particular manner that is represented by the typical shape of the diffusion curve. At first, few actors adopt the innovation, but the adoption rate accelerates. When 10 to 20 percent of the actors have adopted, the acceleration levels off while the absolute number of new adopters is still increasing, causing a sharp rise of the total number of adopters. Finally, the number of new adopters decreases, and the diffusion process slowly reaches its end.

This growth pattern is typical for a chain reaction caused by contagion. Therefore, network models approach diffusion as a contagion process in

which personal contacts with adopters expose people to an innovation. They learn about the innovation, and their contacts persuade them to adopt. Once exposure reaches their threshold for adopting the innovation, which depends on their personal characteristics and on characteristics of the innovation, they will adopt the innovation and start infecting others. As a consequence, the network structure and the positions of the first adopters in the network, who are usually opinion leaders, influence the rate at which an innovation diffuses. This is a very likely mechanism, but it is difficult to prove that diffusion actually works this way.

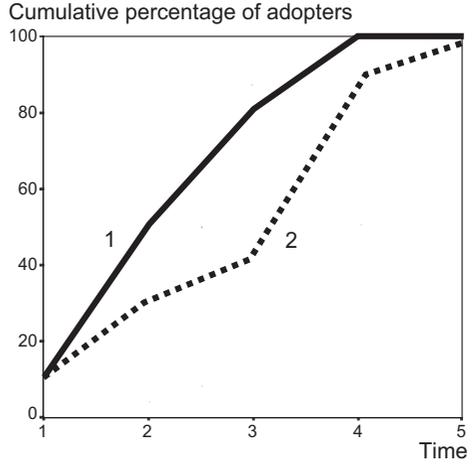
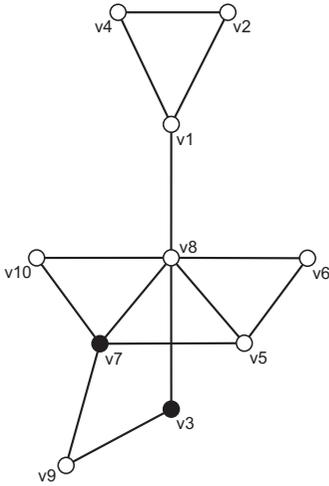
At a particular moment in time, a successful diffusion process is hypothesized to reach a critical mass, which means that the diffusion process can sustain or even accelerate itself without help from outside (e.g., media campaigns). Even with hindsight it is not easy to pinpoint the moment when critical mass is reached, but according to an empirical rule of thumb this happens when the innovation has spread to 10 to 20 percent of the actors who adopt eventually. This is the first second-order inflection point of the S-shaped diffusion curve: the moment when the adoption rate no longer accelerates although it is still increasing. Alternatively, the critical mass may be placed at the moment when the most central actors have adopted or when relatively many actors adopt although their exposure is not increasing. In the latter case, the critical mass or external events are thought to lower individual thresholds.

We are not sure whether critical mass occurs and whether it has the hypothesized impact on the diffusion process. Ongoing research in the diffusion of innovations tradition must clarify this matter. Nevertheless, the concept offers some practical tools for monitoring and guiding a diffusion process.

In theory, knowledge about other people's adoption without personal contact may count as exposure too, especially in the case of status similarities. Knowing that people with similar network positions have adopted although you are not directly linked to them may persuade you to also adopt. In this chapter, we have presented contagion by contact only and not by status imitation. Structural approaches to status and roles will be introduced in Part V of this book.

8.6 Questions

1. If we use a simple contagion by contact model without individual thresholds and one source of contamination, which vertex is the source in diffusion curve 1, and which vertex is the source in curve 2 in the network that follows?



- a. v8 is the source in curve 1, and v1 is the source in curve 2.
 - b. v7 is the source in curve 1, and v2 is the source in curve 2.
 - c. v6 is the source in curve 1, and v3 is the source in curve 2.
 - d. v5 is the source in curve 1, and v4 is the source in curve 2.
2. In the network of Question 1, which vertex would you choose to introduce an innovation (e.g., a new product)? Justify your choice.
 3. In the network of Question 1, assume that the black vertices have adopted an innovation at time 1. Calculate the exposure of the remaining vertices by hand.
 4. Take the situation of Question 3 as your starting point. At time 2, vertices v9 and v10 adopt. What can you say about the thresholds of vertices v10, v9, and v8?
 5. Take the situation of Question 4 as your starting point. If vertices v5, v6, and v8 adopt at time 3, which vertices have threshold lags?
 - a. None of the three vertices.
 - b. Only vertex v5.
 - c. Vertices v5 and v6.
 - d. All three vertices.
 6. Considering your answer to Question 5, can you identify the critical mass of this diffusion process? Justify your answer.
 7. Estimate the total number of adopters from the following data on the start of a diffusion process.

Time	1	2	3	4	5	...
Cumulative number of adopters	21	74	251	635	1176

8.7 Assignment

In a famous study, known as the Columbia University Drug Study, the diffusion of a new drug (gammanym) was investigated. The researchers collected data on the first subscription of this drug by physicians in several communities. In addition, they investigated friendship ties and discussion links between the physicians, asking them to name three doctors they considered to be personal friends and to nominate three doctors with whom they would choose to discuss medical matters.

The file `Galesburg.net` contains a network of discussion (relation 1) and friendship ties (relation 2) between seventeen physicians who adopted the new drug in Galesburg, Illinois, in the 1950s. The partition `Galesburg_adoptiontime.clu` specifies the number of months between the introduction of the new drug and the point at which the physician first prescribed the drug. This is considered to be their adoption time.

Analyze the diffusion process visually and numerically. Do you think the innovation diffused by a simple contagion process or by a contagion process with individual thresholds? Is it probable that external events or a critical mass effect influenced the diffusion process?

8.8 Further Reading

- The example is taken from R. O. Carlson, *Adoption of Educational Innovations* (Eugene, OR: University of Oregon, Center for the Advanced Study of Educational Administration, 1965). Friendship choices and year of adoption are coded from the sociogram on p. 19 of the book.
- The Columbia University Drug Study was reported in J. S. Coleman, E. Katz, and H. Menzel, *Medical Innovation: A Diffusion Study* (Indianapolis, IN: Bobbs-Merrill, 1966). The data, however, are taken from D. Knoke and R. S. Burt, "Prominence." In R. S. Burt and M. J. Minor (Eds.), *Applied Network Analysis: A Methodological Introduction* (Beverly Hills, CA: SAGE, 1983, pp. 195–222).
- E. M. Rogers's *Diffusion of Innovations*, 4th edn. (New York, NY: The Free Press, 1995) offers a general overview of diffusion theory and research. He uses the modern math case as an example (see pp. 65–6 and Chapter 8).
- M. Granovetter introduced the concept of threshold as the percentage of previous adopters in a person's ego-network in "Threshold models of collective behavior." *American Journal of Sociology* 83 (1978), 1420–43.

- T. W. Valente's *Network Models of the Diffusion of Innovations* (Cresskill, NJ: Hampton Press, 1995) presents and evaluates different diffusion network models. He discusses more advanced models in "Network models and methods for studying the diffusion of innovations." In: P. J. Carrington, J. Scott, and S. Wasserman (eds.), *Models and Methods in Social Network Analysis* (Cambridge: Cambridge University Press, 2005, pp. 98–116).

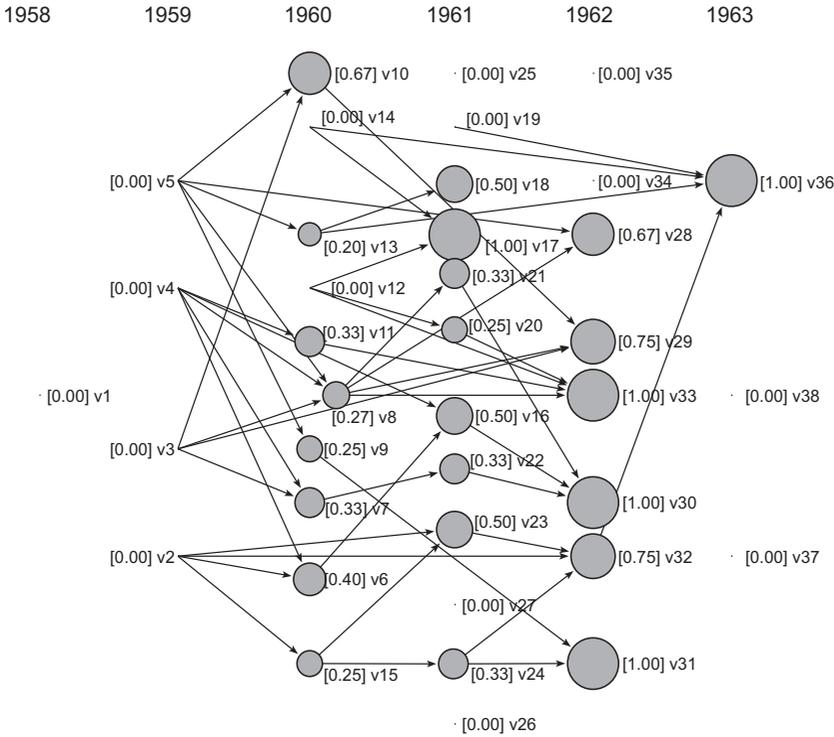
8.9 Answers

Answers to the Exercises

- I. After producing the random network, you must use the *Network> Create Partition> k- Neighbours> All* command to obtain a partition with the distances between the selected source vertex and all other vertices ignoring the direction of lines. Under the current assumptions, the distance to the source vertex is the adoption time of a vertex, so this partition specifies all adoption times; and a tabulation of this partition, which may be obtained with the *Partition> Info* command, shows the adoption rate at each moment (see the columns *CumFreq* or *CumFreq%*). It is worthwhile playing around a little with central and peripheral vertices as source vertices and with different random networks (e.g., with outdegree fixed at 1 or outdegree varying from 1 to 3 or more).
- II. In the sociogram that follows, the sizes of the vertices (*Draw> Network + First Vector* command) and the numbers in brackets (*Options> Mark Vertices Using> Vector Values* in the Draw screen) show the thresholds of all vertices. If we ignore the actors without contacts to previous adopters, which cannot have adopted by direct contagion, the thresholds seem to increase from left to right. The mean thresholds per year (excluding the actors without links to prior adopters) illustrate this: 0.34 (1960), 0.47 (1961), 0.86 (1962), and 1.00 (1963).

Answers to the Questions in Section 8.6

1. Answer b is correct. Recall that the simple contagion model without individual thresholds assumes that an adopter persuades his neighbors to adopt at the next time, who transmit the innovation to their neighbors, and so on. This means that the adoption time of a vertex is directly related to its distance from the source of contagion. If we assume that the source adopts at time 1, the cumulative percentage of adopters is one of ten (the number of vertices) or 10 percent. This is correct because both diffusion curves start at 10 percent. At time 2, the neighbors of the source adopt. In the solid line, 50 percent or five of ten vertices have adopted at the second moment. One of them is



the source, so there are four new adopters who are neighbors of the source. There is only one vertex in the network with four neighbors, namely, vertex v7, so this must be the source of the diffusion process represented by the solid curve.

From the dotted curve, we can infer that two vertices adopt at time 2, one vertex adopts at time 3, five vertices adopt at time 4, and one vertex adopts at time 5. The source has two neighbors, so it can be vertex v2, v4, v6, v3, v9, or v10. At distance 2, it is connected to one vertex. This is true for vertices v2 and v4 but not for the other possible sources. As is shown, the diffusion process is identical whether it starts from v2 or v4, so the dotted line may represent the diffusion from both vertices.

2. For optimal diffusion, you must choose the most central vertex in a network because it exposes most vertices to the innovation. Vertex v8 is most central and has the largest degree, so it is a good choice.
3. Vertices v1, v2, v4, and v6 are not directly linked to the adopters (v3 and v7) in the network, so their exposure is 0.00. Vertex v9 is only linked to adopters, so its exposure is maximal: 1.00. One of v10's two neighbors is an adopter, so its exposure is 0.5. One of v5's three neighbors and two of v8's six neighbors are adopters, so they have an exposure of 0.33.

4. If we calculate the exposure of a vertex as the number of vertices that adopted previously, the threshold of v_9 and v_{10} , who adopt at time 2, is equal to their exposure to the vertices that adopted at time T_1 . Hence, the threshold of v_9 is 1.00, and the threshold of v_{10} is 0.5. The exposure of vertex v_8 is 0.33 at T_2 ; but this vertex does not adopt, so we may conclude that its threshold is higher than 0.33.
5. Answer c is correct. At time T_2 , vertices v_9 and v_{10} adopt (see Question 3). This changes the exposure of vertex v_8 : Now half of its neighbors are adopters. The new degree of exposure may surpass its threshold, hence, v_8 adopts at the next time. In contrast, the exposure of vertices v_5 and v_6 does not change at T_2 because neither vertex is directly linked to v_9 or v_{10} , so they could have adopted earlier. These vertices have threshold lags.
6. Threshold lags may be caused by critical mass. When a substantive number of vertices adopt at a moment when their exposure does not increase, their thresholds may be lowered due to critical mass. If we think that two threshold lags ending at time T_3 is a substantive amount for this small network, we may propose that the diffusion process reached its critical mass at T_2 .
7. To estimate the total number of adopters, we need to know the number of adopters at the first second-order inflection point of the diffusion curve, that is, the last time that the acceleration of the adoption rate increases. Because this moment often occurs when 10 to 20 percent of all adopters have espoused the innovation, the total number of adopters may be estimated as five to ten times this number.

Between successive measurements, the number of new adopters specifies the adoption rate, which increases over all five moments (see the third row in the table below). To calculate the acceleration, we must subtract the number of new adopters by the number of new adopters at the previous moment. The results are presented in the fourth row.

Now, we can see that the acceleration decreases after T_3 , so we assume that T_3 represents the inflection point. The total number of adopters may be estimated at five to ten times the (cumulative) number of adopters at T_3 , which is five to ten times 251:1255 to 2510 adopters.

Time	→ 1	→ 2	→ 3	→ 4	→ 5	...
Cumulative number of adopters	21	74	251	635	1176	
Adoption rate (number of new adopters)	21	53	177	384	541	
Acceleration		31	124	207	157	

Part IV

Ranking

Previous chapters paid little attention to the direction of social relations. In matters of cohesion or brokerage, it is more important to know that a tie exists than to know who initiates it. In this part, however, direction is central, especially asymmetry in social relations. Which choices are not reciprocated? Asymmetry in social relations points to social prestige and ranking.

Chapter 9 presents the concept of structural prestige as the reception of direct and indirect positive choices. We compare structural prestige scores with social prestige of persons measured separately to find that there is some but not perfect overlap. In Chapter 10, we discuss techniques to infer informal or latent ranking from the occurrence of different types of triads and acyclic components in a directed network. Finally, Chapter 11 focuses on networks with arcs indicating the passing of time: genealogical networks and citation networks.

Prestige

9.1 Introduction

In directed networks, people who receive many positive choices are considered to be prestigious. Prestige becomes salient especially if positive choices are not reciprocated, for instance, if everybody likes to play with the most popular girl or boy in a group but he or she does not play with all of them or, in the case of sentiments, if people tend to express positive sentiments toward prestigious persons but receive negative sentiments in return. In these cases, social prestige is connected to social power and the privilege of not having to reciprocate choices.

In social network analysis, prestige is conceptualized as a particular pattern of social ties. We discuss techniques to calculate the *structural prestige* of a person from his or her social ties, notably sociometric choices. We do not compute a prestige score for an entire network.

Structural prestige is not identical to the concept of *social prestige* in the social sciences or in ordinary speech. For example, the medical profession is thought to be prestigious, but it is difficult to consider professions as a network in which many arcs point toward the medical profession. The prestige of an art museum may depend on the value and origins of its collection rather than on the number of art works it attracts (receives) from other museums. However, social prestige is probably related to structural prestige. In community studies, for example, a physician is more often nominated in advice-seeking relations than members of many other professions, and a prestigious art museum receives more attention from art critics than less prestigious ones.

In this chapter, we compare the structural prestige of families within a network of visiting ties to their social prestige. As is shown, the two kinds of prestige are related but far from identical. Therefore, be careful not to equate structural prestige to social prestige. Instead, find out whether structural prestige scores on a social relation match indicators of social

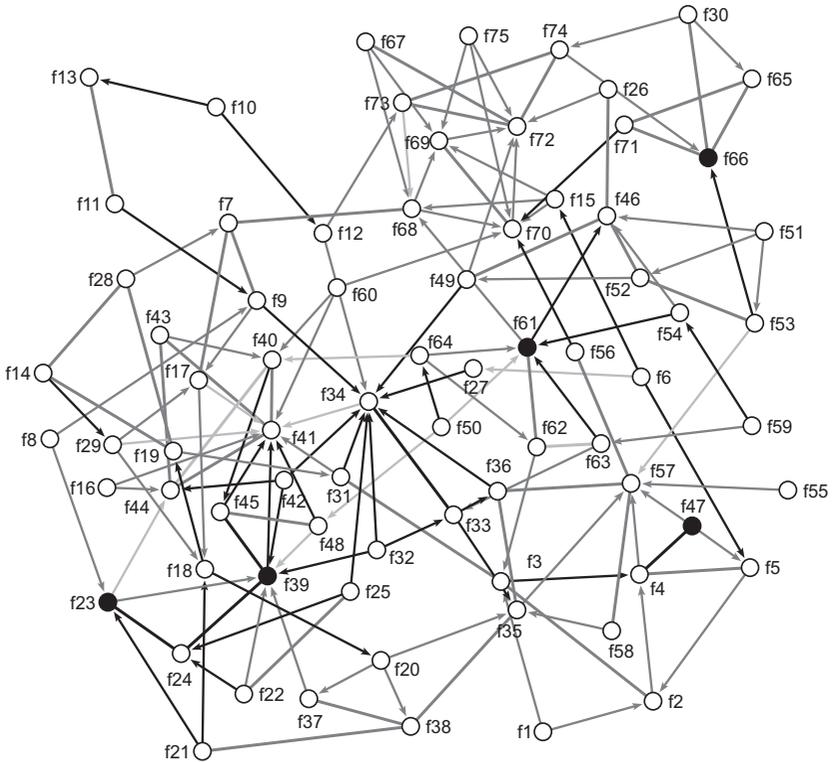


Figure 81. Visiting ties and prestige leaders in San Juan Sur.

prestige that are measured by external variables. In a particular setting, which social relation is connected to social prestige?

9.2 Example

Let us have a look at the visiting ties in another village in the Turrialba region of Costa Rica: the village of San Juan Sur containing seventy-five *haciendas* (`SanJuanSur.net`). In Chapter 3, we analyzed cohesive subgroups in the network of Attiro. Now, we concentrate on status and prestige. Members of the San Juan Sur community who were well informed about its population were asked to rank all heads of households according to their importance to the community. Social status was computed for each family farm in this area as the average importance of its inhabitants and grouped into fourteen classes (partition `SanJuanSur_status.clu`). Prestige leaders were identified as those people who received more than ten nominations within the community on the question: Which persons would you pick to represent you and the people of this place on a commission? In Figure 81, the prestige leaders

are colored black (partition `SanJuanSur_leaders.clu`, the data are collected in `SanJuanSur2.paj`). In this chapter, we use these scores as indicators of social prestige or status.

Figure 81 depicts the simple network of visiting ties. Note that bidirectional arcs are replaced by edges for the sake of clarity. There are three types of visiting ties, indicated by the color of lines that are defined by the relation numbers in the data file: dark gray (or red) arcs (relation number 2) represent visits among kin, light gray (or blue) arcs (relation number 3) are visits among families bound by godparent or godchild ties (church relations), and other types of ties are drawn in black arcs (relation number 1). Note that the grays in Figure 81 do not show the different types of visiting ties as clearly as the colors on your computer screen. In this chapter, we calculate the structural prestige of the families from these visiting relations to find out whether social prestige (status) matches structural prestige.

9.3 Popularity and Indegree

At a first glance, this sociogram tells us little about the structural positions of prestige leaders. The leaders are dispersed over the network. They are situated in dense areas (e.g., family f39) as well as in the margins (families f23, f47, and f66). We need some calculations to get a better view of structural prestige.

The *popularity* or indegree of a vertex is the number of arcs it receives in a directed network.

The simplest measure of structural prestige is called popularity, and it is measured by the number of choices a vertex receives: its indegree. Nominations on a positive social relation (e.g., liking) express prestige; more nominations indicate higher structural prestige, for example, in an election or a popularity poll. In this example, receiving more visitors indicates higher structural prestige. Note that the indegree of a vertex can be determined only in a directed network. In undirected networks, we cannot measure prestige; instead, we use degree as a simple measure of centrality (see Chapter 6). In fact, several centrality measures are equal or similar to prestige measures applied to undirected networks.

Of course, a high indegree on a relation such as “lend money to someone” does not reflect the popularity of an actor; it merely identifies someone who owes money to many persons. We should note that indegree does reflect prestige if we *transpose* the arcs in such a network, that is, if we reverse the direction of arcs. In the transposed network, arcs represent the “owe money to” relation, and someone with a large indegree has lent

Table 13. *Indegree listing in Pajek*

Cluster	Freq	Freq%	CumFreq	CumFreq%	Representative
0	13	17.3333	13	17.3333	f1
1	17	22.6667	30	40.0000	f11
2	11	14.6667	41	54.6667	f13
3	15	20.0000	56	74.6667	f2
4	5	6.6667	61	81.3333	f3
5	6	8.0000	67	89.3333	f35
6	2	2.6667	69	92.0000	f44
7	1	1.3333	70	93.3333	f70
8	3	4.0000	73	97.3333	f39
10	1	1.3333	74	98.6667	f34
12	1	1.3333	75	100.0000	f41
SUM	75	100.0000			

money to many other people. Probably, this actor is quite rich compared to the other actors and more prestigious.

In the original network, the direction of the arcs depends on the way the researcher has defined the relation and worded the sociometric question. In the analysis, it is sometimes better to change the direction of the arcs. You are allowed to do this, because no information is lost in the transposed network: Just transpose it again, and you obtain the original network. It is interesting to note that several structural properties of a network do not change when the arcs are transposed (e.g., the components remain unchanged) and other properties are just swapped (e.g., outdegree becomes indegree and vice versa).

Application

Network>
Create
Partition>
Degree> Input

In Chapter 3, you learned to compute the indegree of vertices in a directed network by means of the *Input* command in the *Network> Create Partition> Degree* submenu. This command creates a new partition that can be displayed with *Partition> Info*. Table 13 shows the frequency count of the indegree of family farms in San Juan Sur. Thirteen families were not visited, so their indegree is 0. They have minimal structural prestige. Family number f41 is most popular because it is visited by twelve families (see entry of class 12 in Table 13). Note that the indegree is equal to the number of visiting families because there are no multiple arcs. In Figure 81, we can see the high number of visits that family f41 receives. This simple frequency tabulation summarizes the distribution of popularity better than the sociogram. The table shows that half of the families receive two visits at most. No more than one-fifth of all families receive five or more visits (see column CumFreq%).

File>
Partition>
View/Edit

How about the prestige leaders? May we conclude that families containing prestige leaders are structurally prestigious? Inspecting the

sociogram or the indegree partition (use *File*> *Partition*> *View/Edit*), we note that prestige leaders f23, f39, f47, f61, and f66 have indegree 3, 8, 1, 5, and 5, respectively. All prestige leaders except for family f47 have indegree above average and three of five families belong to the top 20 percent because they receive five or more visits. Therefore, we conclude that the prestige leaders are visited quite often, but there are other families that receive even more visits. Structural prestige measured by indegree does not distinguish completely between prestige leaders and other frequently visited families in this example.

If the relation points from more to less prestigious vertices, as in the case of “lend money to,” you should change the direction of all ties in a network. This can simply be done by means of the *Network*> *Create New Network*> *Transform*> *Transpose*> *1-Mode* command.

```
Network>
Create New
Network>
Transform>
Transpose>
1-Mode
```

9.4 Correlation

Does structural prestige indicated by indegree match social status as it was rated by experts within the community? To answer this question, we have to apply standard statistical analysis to the results from our network analysis, which are the structural prestige scores. Because this is not a course in statistics, we keep it as simple as possible. It is our primary goal to show that social network analysis and statistical analysis are two sets of techniques that work very well together in social research.

In statistics, the association between two phenomena is usually measured by *correlation coefficients*. Correlation coefficients range from 1 to -1 . A positive coefficient indicates that a high score on one feature is associated with a high score on the other (e.g., high structural prestige occurs in families with high social status). A negative coefficient points toward a negative or inverse relation: A high score on one characteristic combines with a low score on the other (e.g., high structural prestige is found predominantly with low social status families). As a rule of thumb, we may say that there is no correlation if the absolute value of the coefficient is less than 0.05. If the absolute value of a coefficient is between 0.05 and 0.25, association is weak, coefficients from 0.25 to 0.60 (and from -0.25 to -0.60) indicate moderate association, and 0.60 to 1.00 (or -0.60 to -1.00) is interpreted as strong association. Usually, a coefficient of 1 or -1 is said to display perfect association, but it is very unlikely that you will find this unless you correlate a characteristic to itself.

In Pajek, two kinds of correlation coefficients can be computed: Spearman’s rank correlation and Pearson’s correlation. *Spearman’s rank correlation* determines whether the ranking of vertices on one characteristic (e.g., indegree) matches the ranking on another characteristic (e.g., status). The magnitude of differences between ranks is unimportant. Of

course, both characteristics must have scores that can be ranked. Spearman's rank correlation is a robust measure of association provided that few cases have equal ranks.

Pearson's correlation coefficient uses the exact numerical scores on both characteristics. It assumes a linear association between two characteristics, which means that a unit increase in one characteristic will be associated with a fixed increase (or decrease) in the other. In our example, Pearson's correlation assumes that one extra indegree of structural prestige is accompanied by a fixed amount of additional social status (e.g., 2.4 extra points of social prestige).

Pearson is more precise and more sensitive than Spearman. This can be an advantage as well as a disadvantage. If a linear association exists among two features of vertices in the network, Pearson's correlation coefficient describes it more accurately than that of Spearman. However, the assumption that unit change on one feature is associated with a fixed change in another is very strict and often not met. For example, one extra indegree may involve substantial extra social status for families in the classes with low indegree, whereas it may be associated with little extra status for families in the middle or upper indegree classes. In this case, Pearson's coefficient underestimates the actual association, whereas that of Spearman does not. Therefore, it is important to use Pearson's correlation coefficient only if its results do not diverge too much from Spearman's coefficient. If results are very different, the data contain irregularities.

Application

To compute a correlation coefficient, we need two characteristics of each vertex in the network. As learned in Chapter 2, features of vertices are stored in partitions and vectors. A partition contains integers; a vector is a list of numbers with decimals. Because Spearman's rank correlation coefficient takes only the (discrete) rank order of scores into account, it operates on partitions. To calculate Spearman, you need two partitions. Hence, Spearman can be found in the *Partitions* menu. However, Pearson's correlation coefficient uses the exact magnitude of scores. In Pajek, Pearson needs two vectors as input data, and the procedure is to be found in the *Vectors* menu.

Partitions>
Info>
Spearman Rank

Social status scores are available as a partition (`SanJuanSur_status.c1u`) that must be opened in Pajek to compute its correlation with the indegree partition. Select the two partitions in the *Partitions* drop-down menus. It does not matter which partition is first. When both partitions are selected, choose the command *Spearman Rank* on the *Info* submenu (see Figure 82), and Pajek computes the rank correlation coefficient. In this case, it is 0.40, meaning that there is a moderate positive rank correlation between indegree and social status. Families with larger indegree tend to

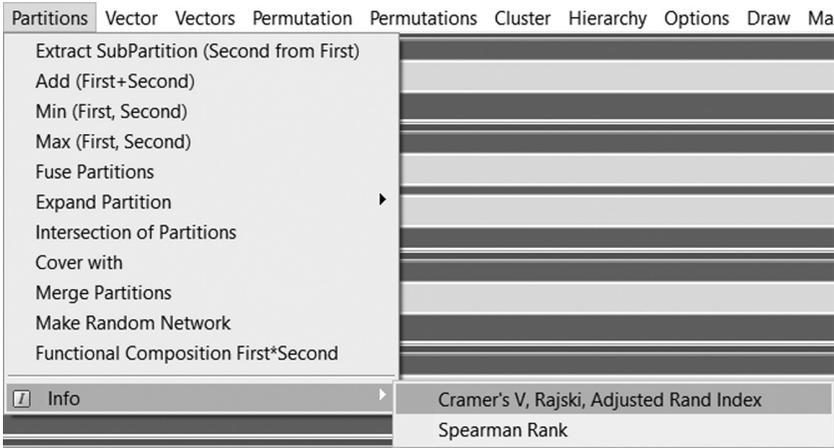


Figure 82. *Partitions* menu in Pajek.

be families with higher status. Hence, we may conclude that structural prestige is moderately associated with status in this example.

Pearson's correlation coefficient is computed in a similar way. Select a first and second vector and choose the *Info* submenu from the *Vectors* menu, which has no options other than Pearson's coefficient. In this example, you may use the input degree partition and the status partition (`SanJuanSur_status.clu`), but you have to translate both partitions to vectors first with the *Partition> Copy to Vector* command. Pearson's correlation coefficient is 0.35, which is slightly lower than Spearman's correlation, indicating that the association is not linear. Using the rule of thumb previously specified, however, we reach the same conclusion about the association between indegree and social status.

Vectors> Info

Exercise I

Split the visiting relations network into three separate networks: one for each type of visiting relation. For each new network, determine the popularity of the vertices and the correlation between popularity and status (`SanJuanSur_status.clu`). Which type of visiting relation matches the status hierarchy best?

9.5 Domains

Popularity is a very restricted measure of prestige because it takes only direct choices into account. With popularity it does not matter whether choices are received from people who are not chosen themselves or from popular people. The overall structure of the network is disregarded.

Several efforts have been made to extend prestige to indirect choices. The first idea that comes to mind is to count all people by whom someone is nominated directly or indirectly, that is, without or with go-betweens. This is the *input domain* of an actor, which has been called the influence domain because structurally prestigious people are thought to influence people who regard them as their leaders. The larger the input domain of a person, the higher his or her structural prestige.

The *input domain* of a vertex in a directed network is the number or percentage of all other vertices that are connected by a path to this vertex.

Note that the output domain is more likely to reflect prestige in the case of a relation such as “lend money to.” It is easy to define the output domain of a vertex, and we guess that you understand that the output domain of a vertex is identical to the input domain of the vertex in the transposed network. In fact, we may distinguish between three domains: input domain, output domain, and (overall) domain, which is the union of the input and the output domain.

Let us have a look at the visiting relations network again to understand the concept of an input domain. Figure 83 contains the vertices in the input domain of family f47 and the paths toward this family. The numbers inside the vertices indicate the distance to family f47. Clearly, family f47 has 0 distance to itself. This family is visited only by family f4: Its distance to family f47 is 1. Families f2, f3, and f5 visit family f4, so they can reach family f47 via family f4 (distance 2). They are visited by four families (distance 3), and so on. Ultimately, family f47 can be reached by sixty-four of the remaining seventy-four families (86 percent) in San Juan Sur. The input domain of family f47 equals sixty-four vertices, or 86 percent.

The ten families outside the input domain of family f47, which are not drawn in Figure 83, include prestige leaders f23 and f39 and several families from the densest part of the network (e.g., f40, f43, f44, f45, and f48), among them family f41 with highest indegree. Family f47, which is also a prestige leader, turns out to be unreachable for the prestige leaders in the center of the network. This family was probably nominated as a representative by a relatively isolated group of families, including families f2, f3, f4, and f5. In this case, prestige leadership does not necessarily imply high overall social or structural prestige. The prestige leader is probably just a little more prestigious than the subgroup he or she represents.

In a well-connected network with many reciprocal ties, vertices are reachable from most other vertices. Hence, input domain scores display little variation. In this case, it is more interesting to capture the network structure in a prestige index that does not consider the entire input

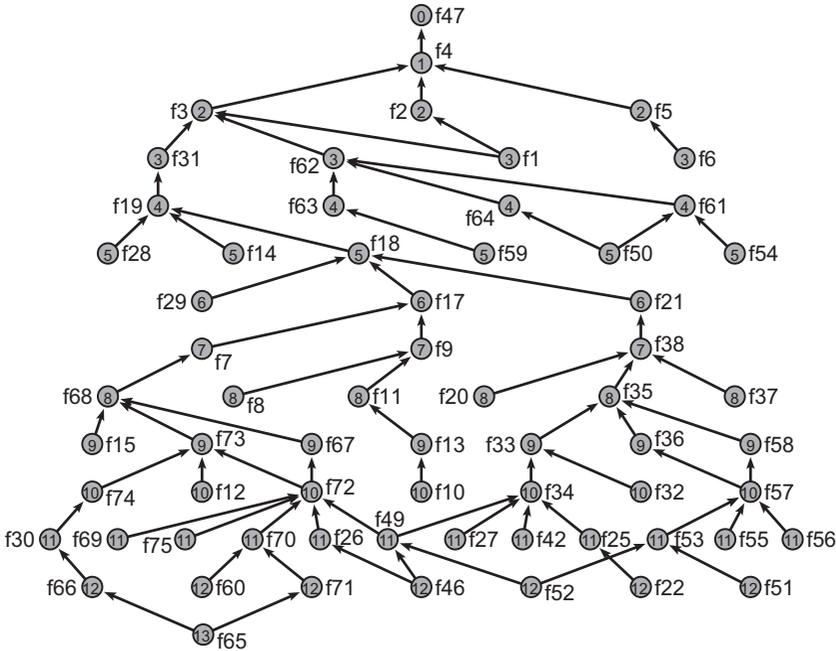


Figure 83. Distances to family 47 (represented by the numbers within the vertices).

domain. For example, we can count the vertices that are able to reach a person in one or two steps: direct choices and indirect choices with one go-between. This *restricted input domain* takes into account only the direct popularity of the people by whom one is nominated. The input domain of family f47 restricted to two steps (distance 2) is four (or 5%): one family at distance 1 (f4), and three families at distance 2 (f2, f3, and f5) (see Figure 83).

Application

The input domain of a particular vertex can be found with the `Network> Create Partition> k-Neighbours> Input` command, which is discussed in Chapter 6. In the first dialog box, enter the number or label of a vertex (e.g., f47), and in the second dialog box accept the default value (0) to compute all distances. Then, the command creates a partition specifying the distances of all vertices to the selected vertex. From a frequency tabulation, created with the `Partition> Info` command (Table 14), you can calculate the number of vertices (CumFreq) in the input domain of the selected vertex at a particular maximum distance; for instance, the input domain at maximum distance 2 contains four vertices: the five vertices at maximum distance 2 minus family f47 itself. The entry identified by

```
Network>
Create
Partition>
k-Neighbours>
Input
Partition> Info
```

Table 14. *Input domain of f47*

Cluster	Freq	Freq%	Valid%	CumFreq	CumFreq%	CumValid%	Representative
0	1	1.3333	1.5385	1	1.3333	1.5385	f47
1	1	1.3333	1.5385	2	2.6667	3.0769	f4
2	3	4.0000	4.6154	5	6.6667	7.6923	f2
3	4	5.3333	6.1538	9	12.0000	13.8462	f1
4	4	5.3333	6.1538	13	17.3333	20.0000	f19
5	6	8.0000	9.2308	19	25.3333	29.2308	f14
6	3	4.0000	4.6154	22	29.3333	33.8462	f17
7	3	4.0000	4.6154	25	33.3333	38.4615	f7
8	6	8.0000	9.2308	31	41.3333	47.6923	f8
9	7	9.3333	10.7692	38	50.6667	58.4615	f13
10	7	9.3333	10.7692	45	60.0000	69.2308	f10
11	12	16.0000	18.4615	57	76.0000	87.6923	f25
12	7	9.3333	10.7692	64	85.3333	98.4615	f22
13	1	1.3333	1.5385	65	86.6667	100.0000	f65
SUM	65	86.6667	100.0000				
UNKNOWN	10	13.3333					
TOTAL	75	100.0000					

“Unknown” in the table shows the number of vertices that are not connected by a path to the selected vertex: They do not belong to its input domain. In our example, ten of seventy-four vertices (do not count the selected vertex itself!) are outside the input domain of family f47, which is 14 percent; the remaining 86 percent of the vertices are inside its input domain. Note that you cannot find these percentages in the table because all percentages there include family f47.

Network>
Create Vector>
Centrality>
Proximity
Prestige> Input

It is quite cumbersome to repeat this command for each vertex in a network, so Pajek contains a command that calculates the size of the input domains of all vertices in one go: *Network> Create Vector> Centrality> Proximity Prestige> Input*. Use the command *Input* to restrict the analysis to incoming arcs only. A dialog box, which is similar to the one displayed by *k-Neighbours*, allows you to specify a maximum distance for the input domain.

The *Proximity Prestige> Input* command produces four new data objects: one partition and three vectors (that is why the command is located in *Create Vector* submenu). The partition specifies the number of vertices within the input domain of each vertex. The vector labeled “Normalized Size of Input Domain” lists the size of input domains as a proportion of all vertices (minus the vertex itself), and the second vector gives the average distance to a vertex from all vertices in its input domain. Of course, it is impossible to compute average distance in the case of a vertex with an empty input domain, that is, a vertex that is not chosen at all. In this case, average distance is set to 99999998, which represents infinity.

Table 15. *Size of input domains in the visiting relations network*

Cluster	Freq	Freq%	CumFreq	CumFreq%	Representative
0	13	17.3333	13	17.3333	f1
1	7	9.3333	20	26.6667	f12
2	2	2.6667	22	29.3333	f11
6	3	4.0000	25	33.3333	f61
12	5	6.6667	30	40.0000	f26
64	36	48.0000	66	88.0000	f2
74	9	12.0000	75	100.0000	f23
SUM	75	100.0000			

Table 15 lists the size of input domains in the visiting relations network. Nine families have maximal input domains; they are reachable from all seventy-four other vertices. Prestige leaders f23 and f39 are among them. As noted, the third prestige leader, family f47, is situated in the class of families with an input domain of size 64. Inspecting the partition with input domain sizes with the *File> Partition> View/Edit* procedure, we find that prestige leader f66 belongs to this class, too. Family f61 is the only prestige leader that has a small input domain of size 6. We may conclude that most prestige leaders have large input domains, but many families with equally large input domains are not prestige leaders.

The rank correlation between structural prestige measured as the size of the input domain and social prestige indicated by social status scores can easily be computed (see Section 9.4). Spearman's rank correlation coefficient is 0.36, which is a little less than the rank correlation between popularity (indegree) and social status. Nevertheless, it points to a positive, moderate association between input domain and social status: Larger input domains occur among families of higher social status.

Exercise II

Produce a sociogram such as that in Figure 83 from the network of visiting relations in San Juan Sur. Let vertex colors indicate the distance to family f47 in Pajek's Draw screen. Note that this sociogram requires many steps. You must combine several techniques that you have learned in the current and previous chapters.

9.6 Proximity Prestige

In the previous section, we noted that the input domain of a vertex is not a perfect measure of prestige. In a well-connected network, the input domain of a vertex often contains all or almost all other vertices, so it does not distinguish very well between vertices. In this case, we proposed

to limit the input domain to direct neighbors or to neighbors at maximum distance 2 on the assumption that nominations by close neighbors are more important than nominations by distant neighbors. An indirect choice contributes less to prestige if it is mediated by a longer chain of intermediaries.

Of course, the choice of a maximum distance from neighbors within a restricted input domain is quite arbitrary. The concept of proximity prestige overcomes this problem. This index of prestige considers all vertices within the input domain of a vertex, but it attaches more importance to a nomination if it is expressed by a closer neighbor. In other words, a nomination by a close neighbor contributes more to the proximity prestige of an actor than a nomination by a distant neighbor, but many “distant nominations” may contribute as much as one “close nomination.”

To allow direct choices to contribute more to the prestige of a vertex than indirect choices, proximity prestige weights each choice by its path distance to the vertex. A higher distance yields a lower contribution to the proximity prestige of a vertex, but each choice contributes something. In the calculation of proximity prestige, this is accomplished by dividing the input domain of a vertex (expressed as a proportion of all vertices that may be part of the input domain) by the average distance from all vertices in the input domain. A larger input domain (larger numerator) yields a higher proximity prestige because more vertices are choosing an actor directly or indirectly. In addition, a smaller average distance (smaller denominator) yields a higher proximity prestige score because there are more nominations by close neighbors.

Maximum proximity prestige is achieved if a vertex is directly chosen by all other vertices. This is the case, for example, in a star-network in which all choices are directed to the central vertex. Then, the proportion of vertices in the input domain is 1 and the mean distance from these vertices is 1, so proximity prestige is 1 divided by 1. Vertices without input domain get minimum proximity prestige by definition, which is 0.

The *proximity prestige* of a vertex is the proportion of all vertices (except itself) in its input domain divided by the mean distance from all vertices in its input domain.

In Figure 84, all vertices at the extremes of the network (v2, v4, v5, v6, and v10) have empty input domains; hence, they have a proximity score of 0. The input domain of vertex v9 contains vertex v10 only, so its size is 1 out of 9 (0.11). Average distance within the input domain of vertex v9 is 1, so the proximity prestige of vertex 9 is 0.11 divided by 1. You can see that the proximity prestige of vertices increases if they have a longer “tail” from vertex v10 to v1. Vertex v1 has a maximal input domain, because

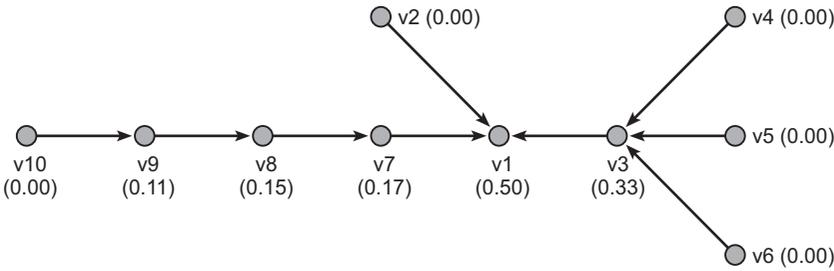


Figure 84. Proximity prestige in a small network.

it can be reached by all nine vertices (a proportion of 1.00). Average distance is 2.0, so proximity prestige amounts to 1.00 divided by 2.0, which is 0.5.

Application

In the previous section, we learned how to compute the size of input domains and average distance from all vertices within the input domain. The same command (*Network> Create Vector> Centrality> Proximity Prestige> Input*) returns, as the caption suggests, also a third vector called “Input Proximity Prestige.” Inspect the proximity prestige scores of all vertices with the command *Vector> Info* or browse with *File> Vector> View/Edit*. Proximity prestige scores range from 0 to 1.

In the network of visiting ties at San Juan Sur, proximity prestige ranges from 0.0 to 0.33. Family f41 has the highest proximity prestige. Three of five prestige leaders have a proximity prestige above average (0.12). However, the proximity prestige of families f47 (0.11) and f61 (0.07) is below average. We must conclude that prestige leaders are not characterized by high proximity prestige. In Section 9.5, we noted that family f47 occupies a special position in the network. Inspection of the average distances confirms this: Family f47 has the largest average distance (8.03). This family is difficult to reach in the network.

Finally, let us see whether proximity prestige is associated with social status in San Juan Sur. Before we can compute Spearman’s rank correlation, we must turn the vector with proximity prestige scores into a partition. As learned in Chapter 2, this can be done in several ways. In this case, the easiest way to convert the vector into a partition is to create classes of equal width with the procedure *Vector> Make Partition> by Intervals> First Threshold and Step*. Specify 0.01 as the first threshold (the upper limit of the lowest class), and enter this number also as the step (the class width) to obtain a partition with classes between 0 and 100.

The newly created partition with proximity prestige scores can be correlated to the existing partition with social status (*SanJuanSur_status.clu*) in the manner described in Section 9.4. Spearman’s

*Vector> Make
Partition> by
Intervals> First
Threshold and
Step*

*Partitions>
Info>
Spearman Rank*

correlation coefficient is 0.26, indicating a low or moderate association between proximity prestige within the network and social status rated separately by members of the community. In this example, social status is related less to proximity prestige than to popularity (indegree), which has a rank correlation of .40 (see Section 9.4).

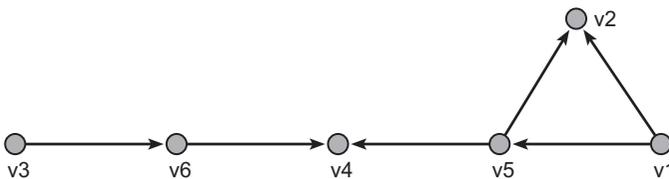
9.7 Summary

This is the first chapter of the book to deal with asymmetry in social networks. We present the simplest way to take the direction of ties into account, which is to pay attention to incoming ties only. Structural indices that do this are called measures of prestige. Actors who receive a lot of choices are popular, provided of course that the choices express a positive social relation. Popularity, which is measured as the indegree of a vertex, is the first index of prestige we discuss. More advanced measures of prestige also take indirect choices into account. We present two advanced measures: the input domain of a vertex and proximity prestige.

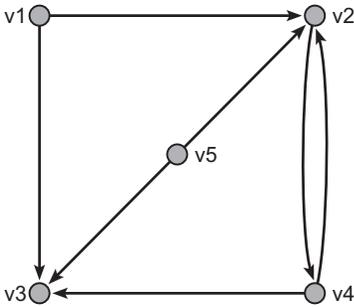
It is important to distinguish between structural prestige and social prestige. The indices introduced in this chapter assess structural prestige, that is, a pattern of ties that network analysts call prestige. They are called prestige because actors in prestigious network positions often enjoy high social prestige. However, the example we use shows that structural prestige and social prestige do not match perfectly; we find moderate association only. We use correlation coefficients to establish the strength of the association between structural prestige and social status scores measured independently from the network. This is an example of an important research strategy, namely, using structural indices such as prestige scores in statistical analysis.

9.8 Questions

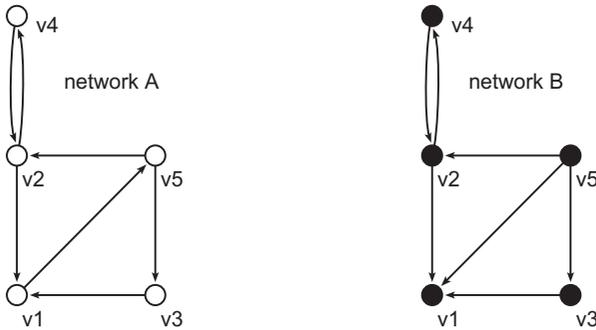
1. In the network that follows, which vertex or vertices have the highest proximity prestige?



- a. Vertex v2
 - b. Vertex v4
 - c. Vertices v2 and v4 have equal proximity prestige.
 - d. It is impossible to tell from this sociogram.
2. In the network that follows, which vertex or vertices have minimal structural prestige?



- a. Vertex v5
 - b. Vertices v5 and v1
 - c. All vertices have equal structural prestige.
 - d. It is impossible to tell from this sociogram.
3. In the network presented in Question 2, which vertex or vertices have the highest social prestige?
- a. Vertex v3
 - b. Vertices v2 and v3
 - c. Vertices v2, v3, and v4 have approximately equal prestige.
 - d. It is impossible to tell from this sociogram.
4. What is the correct interpretation of a correlation coefficient of size $-.20$?
- a. Weak negative association
 - b. Medium association
 - c. Medium positive association
 - d. No association
5. Which prestige indices take indirect ties into account?
- a. Proximity prestige only
 - b. Proximity prestige and input domain
 - c. Proximity prestige and popularity
 - d. Input domain and popularity
6. For which of the networks that follow is it useless to compute structural prestige as the full input domain of a vertex?



- a. A
 - b. B
 - c. A and B
 - d. A nor B
7. In an undirected network, is proximity prestige equal to closeness centrality?
- a. Yes, because proximity prestige is equal to the mean distance to all other vertices in an undirected network.
 - b. Yes, because a prestige index applied to an undirected network is equal to a centrality index.
 - c. No, unless the network is connected.
 - d. No, because the calculation of an input domain is meaningless in an undirected network.

9.9 Assignment

In Chapter 8, we learned that the diffusion of innovations resembles a contagion process because social contacts are needed to persuade people to adopt innovations. It is hypothesized, therefore, that prestige is associated with adoption time: Less prestigious actors adopt later because they wait for more prestigious opinion leaders to adopt first.

The file `Galesburg_discussion.net` contains a network of discussion ties among thirty-one physicians in Galesburg, Illinois, in the 1950s. The researchers asked each physician to name three doctors with whom they would choose to discuss medical matters. For seventeen physicians, the date that they first prescribed a new drug (gammanym) was recorded. The partition `Galesburg_adoptiointime31.clu` measures the adoption time as the number of months since the introduction of the drug. Note that the adoption time is unknown (code 999999998) for fourteen physicians. For most of them, the new drug was not relevant.

Investigate whether adoption time is associated with the structural prestige rather than the centrality of doctors in the discussion network. Compute the indices of prestige presented in this chapter (indegree, restricted input domain with a maximum distance of 2, and proximity prestige) as well as the corresponding centrality measures in the undirected network. Use rank correlation, and note that adoption time is higher when a doctor adopts later.

Another hypothesis states that friendship relations are more important than discussion relations for the adoption of a new drug because it is easier to persuade friends than people you know only professionally. Physicians with many direct or indirect friends would adopt sooner than physicians with less central positions in the friendship network. The file `Galesburg_friends.net` contains the friendship network between the doctors. Is the adoption time of the new drug related to prestige or centrality in the friendship network rather than in the discussion network?

9.10 Further Reading

- The data on San Juan Sur are taken from Charles P. Loomis, Julio O. Morales, Roy A. Clifford, and Olen E. Leonard, *Turrialba: Social Systems and the Introduction of Change* (Glencoe, IL: The Free Press, 1953). Consult this book to learn more about the research project.
- The medical innovation project is taken from James S. Coleman, Elihu Katz, and Herbert Menzel, *Medical Innovation: A Diffusion Study* (Indianapolis, IN: Bobbs-Merrill, 1966). David Knoke and Ronald S. Burt reanalyzed the data in a chapter on prominence in R. S. Burt and M. J. Minor (eds.), *Applied Network Analysis: A Methodological Introduction* (Beverly Hills: SAGE, 1983, pp. 195–222). This article contains the basic argument to distinguish between directed prestige and undirected centrality.
- The proximity prestige concept is attributed to Nan Lin: *Foundations of Social Research* (New York, NY: McGraw-Hill, 1976).
- To learn more about prestige indices, use Chapter 5 of Stanley Wasserman and Katherine Faust, *Social Network Analysis: Methods and Applications* (Cambridge: Cambridge University Press, 1994).

9.11 Answers

Answers to the Exercises

- I. Extract one relation at a time from the network with the `Network> Multiple Relations Network> Extract Relation(s) into Separate`

Network(s) command as you learned in Chapter 1. Compute the indegree partition, and select it as the first partition. Select the status partition (`SanJuanSur_status.clu`) as the second partition and calculate Spearman's rank correlation by means of the *Partitions> Info* submenu. Spearman's rank correlation between the status partition and the indegree partition is .22 in the network of other visiting relations (relation number 1), .33 in the network of visits among kin, and .52 in the network of visits among church relations.

Visits among godparents and godchildren kin seem to follow status differences best in that the less prestigious family comes to visit the more prestigious one; but note that the network contains very few lines, so we should not rely too much on this correlation coefficient.

- II. When you compute the input domain of family f47 with the *Network> Create Partition> k-Neighbours> Input* command, as shown in Section 9.5, you will find that the families outside the input domain are attributed to class 999999998 in the input neighbors partition. Extract all classes (do not forget class 0) except 999999998 from the network (*Operations> Network + Partition> Extract> SubNetwork Induced by Union of Selected Clusters* – see Chapter 2, Section 2.4.3) to obtain a new network without the vertices outside f47's input domain. Compute the distances toward f47 again for the new network. Now you can draw this network in layers according to the vertices' distances to family f47 and optimize it as you learned in Section 8.1 (Chapter 8).

In Figure 83, all arcs that do not point up toward f47 have been eliminated because they are not relevant to the shortest paths to f47. This was achieved with the *Operations> Network + Partition> Transform> Direction> Higher→ Lower* command, deleting lines within clusters, too. Note that the input *k-Neighbours* partition must be selected in the *Partition* drop-down menu. After deleting these lines, optimize the positions of the vertices within the layers again (see Section 8.1).

Answers to the Questions in Section 9.8

1. Answer b is correct. Vertex v4 has the largest input domain, which contains all vertices except for v2, and paths to v4 are quite short for all vertices in its input domain. A large numerator (size of input domain) and a relatively small denominator (average distance) yield high proximity prestige. Vertex v2 is second best because it has an even lower average distance from vertices in its input domain (it is directly chosen by both vertices in its input domain), but its input domain is a lot smaller.
2. Answer b is correct. Vertices v1 and v5 are not chosen. They have zero indegree, hence no input domain and minimum proximity

prestige. Both vertices have minimal scores on all prestige indices presented in this chapter.

3. Answer d is correct. Because structural prestige is not necessarily equal to social prestige, we cannot tell which actor has the most social prestige from this sociogram.
4. Answer a is correct. According to the rule of thumb presented, the association is weak. The sign of the coefficient tells us that there is a negative or inverse relation between the two characteristics.
5. Answer b is correct. Input domain counts direct choices as well as indirect choices of vertices at distance two or higher, so it definitely takes indirect ties into account. Proximity prestige uses the input domain, so it uses indirect ties too. Popularity is just the indegree of a vertex, the direct choices it receives. Clearly, it does not use indirect ties.
6. Answer a is correct. In network A, each vertex is reachable for all other vertices, so each vertex has an input domain of size 4. In other words, the network is one strong component. Because there are no differences between vertices with respect to the size of their input domain, this prestige index is useless. Network B differs from network A in the tie between v_1 and v_5 . Changing the direction of this tie breaks the strong component: Vertex v_5 is no longer reachable for any other vertex, and as a consequence v_1 can no longer reach v_2 and v_3 . Now, the size of the input domain varies between vertices; it is a useful prestige index.
7. Answer c is correct. Proximity prestige is the size of the influence domain as a proportion of the maximum number of vertices it can hold divided by the average distance from all vertices in the influence domain of a vertex. Closeness centrality is similar to the denominator of this fraction: average distance. In an undirected network, proximity prestige is only equal to closeness centrality if the numerator of the fraction is 1, which means that all other vertices are part of the input domain. This is the case if the network is connected because each vertex is reachable from all other vertices in a connected undirected network.

Ranking

10.1 Introduction

In the social sciences, society is regarded as a set of social layers or strata. Instead of ranking people, groups, or organizations on a continuous scale of prestige, they are usually classified into a limited set of discrete ranks, for instance, working class, lower middle class, upper middle class, and upper class. Within a group of humans, discrete ranking also occurs, for instance, leaders, followers, and outcasts. The stratification of art worlds into stars, settled artists, and mediocre artists is likely another example. In this chapter, we discuss techniques to extract discrete ranks from social relations.

Social ranking may be formal or informal, and the two types of ranking may coexist. In a formal ranking, it is written down who commands whom, and insignia or symbols minimize the ambiguity of the ranking and preclude any confusion about a person's rank. The army is an obvious example with its elaborate hierarchy. In contrast, an informal ranking is neither written down nor expressed by official symbols. It manifests itself in the opinions and behavior of people toward one another: respect and acts of deference versus disrespect and dominance.

The creation and maintenance of an informal ranking is a very important social process. Social network analysis is needed to investigate it and to assess the positions that individuals occupy within the informal ranking. If a formal ranking exists, it is interesting to compare it to the informal ranking because they do not need to match, just like informal communication patterns often deviate from the official communication structure.

The structural concept of social ranking is an extension to balance theory presented in Chapter 4. Balance theory assigns people to clusters that are not ranked with respect to one another. Within a cluster, people tend to like each other but do not like members of other clusters. Within clusters as well as between clusters, ties are supposed to be symmetric: You are

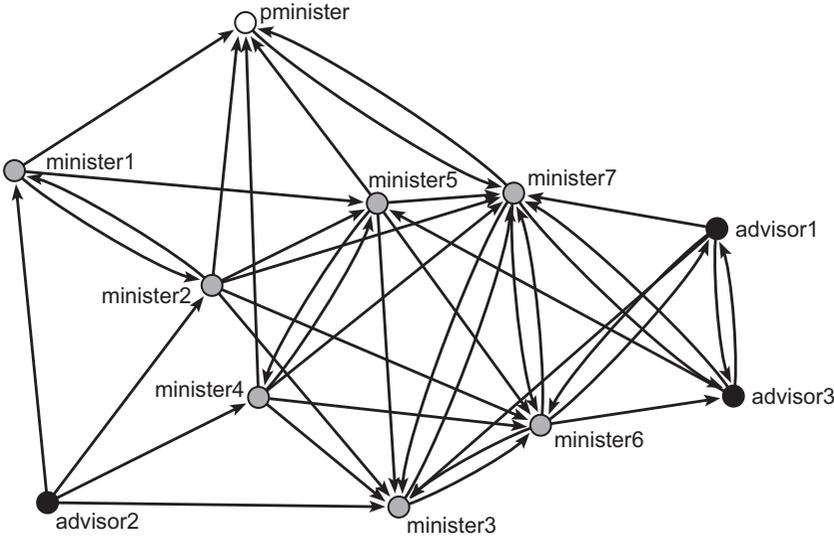


Figure 85. Student government discussion network.

supposed to reciprocate the sentiment or choices that you receive. Elaborating on this perspective, asymmetric ties, for instance, A reports to B but B does not report to A, indicate ranking: B is ranked over A.

10.2 Example

Our example is a network of a discussion relation among the eleven students who were members of the student government at the University of Ljubljana in Slovenia (`student_government.net`). The students were asked to indicate with whom of their fellows they discussed matters concerning the administration of the university informally. We suppose that this relation indicates esteem: Students will choose fellows whom they respect. Therefore, we expect this network to display informal ranking.

Within the parliament, students have positions that convey formal ranking: the prime minister, the ministers, and the advisors. In Figure 85, vertex color indicates the formal position of a student in the parliament (partition `student_government.clu`). We compare the formal ranking to the informal ranking we derive from network analysis of the discussion relation.

10.3 Triadic Analysis

Before we can analyze the ranks in the student government discussion network, we must discuss balance theory once more. In Chapter 4, we

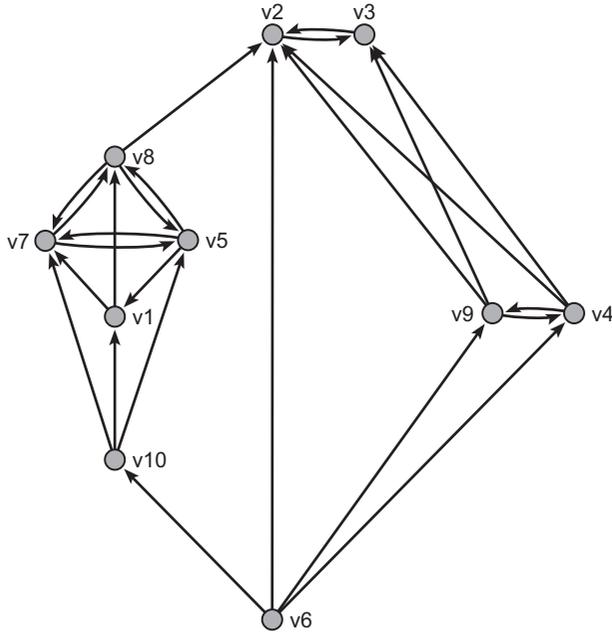


Figure 86. An example of a network with ranks.

learned that a balanced or clusterable network can be partitioned into clusters such that all positive choices occur within clusters and all negative choices are found between clusters. If we replace negative choices with absent choices, it follows that positive choices are found within clusters but choices do not occur between clusters: People tend to choose members from their own group instead of members from other groups. Because absent choices should not occur within a group, each positive choice must be reciprocated.

As a consequence, we can rephrase balance theory for the type of tie between two vertices (dyads) in a simple directed network: Mutual choices indicate group membership, and mutual absent or null choices indicate membership of different groups. Of course, this presupposes that the social relation under investigation implies a positive choice and that an absent choice is equivalent to a negative choice.

A *dyad* is a pair of vertices and the lines between them.

In the directed network of Figure 86, vertices v5, v7, and v8 constitute a cluster because they are connected by mutual choices (*complete dyads*),

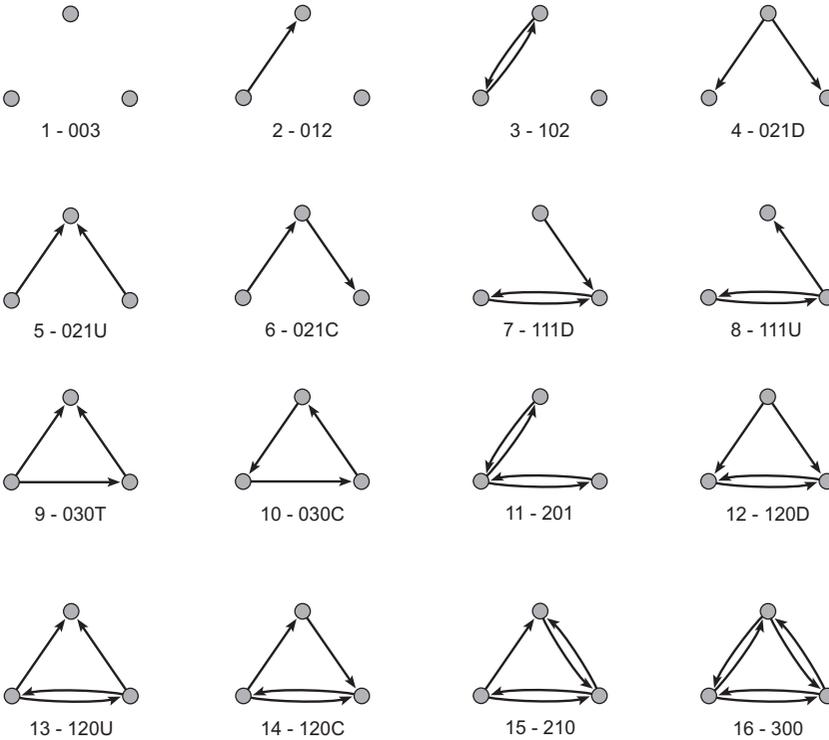


Figure 87. Triad types with their sequential numbers in Pajek.

and vertices v_4 and v_9 constitute another cluster. The two clusters are separated by absent lines or *null dyads*.

Both mutual choices and mutual absent choices are symmetric: You give as good as you get. Symmetric dyads indicate equivalence, so we assume that vertices that are linked by symmetric ties belong to the same rank. The third type of dyad, however, is the *asymmetric dyad*: One person chooses the other, but this choice is not reciprocated. Asymmetric dyads indicate ranking. In an asymmetric dyad, it is assumed that the receiver of the positive choice is ranked over the sender provided that being chosen expresses esteem or appreciation: The former can afford not to reciprocate the choice of the latter. In Figure 86, vertex v_6 is ranked under v_9 and v_4 among others, which are ranked under v_2 and v_3 .

To capture the structure of a directed network, we must proceed from dyads to *triads*. In a simple directed network, sixteen types of triads may occur, which are listed in Figure 87. As proposed by Davis, Holland, and Leinhardt, triad type is identified by an M-A-N number of three digits and, occasionally, a letter. The first digit indicates the number of mutual positive dyads (M), the second digit is the number of asymmetric dyads

(A), and the third digit is the number of null dyads (N). Sometimes a letter that refers to the direction of the asymmetric choices is added to distinguish between triads with the same M–A–N digits: D for down, U for Up, C for cyclic, and T for transitive (which is explained later).

It has been shown that the overall structure of a directed network (or a complete signed network) can be inferred from the types of triads that occur. It is very important to understand the consequences of this discovery: It suffices to analyze small subnetworks (of size 3) to understand the structure of the overall network! We do not have to check all semicycles to determine whether a network conforms to a balance theoretic model as we did in incomplete signed networks (Chapter 4).

If a directed network is balanced, for example, only two of the sixteen types of triads occur, namely, triads 300 and 102. Each cluster is a clique, so each subset of three vertices from a cluster is a complete triad like triad 300 (e.g., vertices v_5 , v_7 , and v_8 in Figure 86). If two vertices belong to one cluster and the third belongs to the other cluster, we encounter a triad in which two vertices are symmetrically linked because they belong to the same clique (e.g., vertices v_5 and v_8 in Figure 86), but they are not connected to the third vertex, which belongs to the other clique (e.g., vertex v_9 in Figure 86). This is represented by triad 102. There are no other possibilities, so the two triad types identify the *balance model* for the structure of the entire network. If a network contains just these two types of triads, then we know that the network consists of two cliques that are not interrelated.

In the course of time, four additional models for the overall structure of a directed network have been discovered, which we present now. These models have a very important property, namely, that they progressively allow for more types of triads to occur. In other words, each model that we present is less restrictive than the previous one. The second model, the *model of clusterability*, for instance, relaxes the demand of the balance model that the network consists of no more than two cliques. A clusterable network may contain three or more clusters. As a consequence, triad 003 is allowed to occur in a clusterable network because it contains vertices that belong to three different clusters. The two balanced triads (300 and 102) are also permitted because they still refer to vertices within one cluster or vertices of two clusters. The clusterability model is more permissive than the balance model: It allows for one more triad type.

In a similar way, the *model of ranked clusters* extends the clusterability model because it allows clusters to be spread over different ranks. Clusters at different ranks are connected by asymmetric dyads: Each vertex in a lower cluster sends unilateral choices toward all vertices in a higher cluster. As a consequence, five triads are permitted that contain asymmetric dyads: 120D, 120U, 021D, 021U, and 030T. In triad 120U, for instance, the bottom two vertices belong to one cluster because they are linked by

mutual choices. The top vertex is connected to them by asymmetric ties: It is chosen but does not reciprocate the choices, so it must belong to a higher rank.

If a network contains these triads in addition to balanced or clusterable triads, the network can be partitioned into ranks and clusters according to the criteria that mutual choices are found within clusters, asymmetric choices point up to a higher rank, and null choices occur between clusters within a rank. In our example of a ranked network (Figure 86), vertices v_4 , v_6 , and v_9 constitute a 120D triad, and a 030T triad contains vertices v_1 , v_5 , and v_{10} .

There are two models that relax the criteria of the ranked clusters model. The first model is the *transitivity model*. In a transitive triad, each path of length 2 is closed by an arc from the starting vertex to the end vertex of the path. If actor A obeys actor B, and actor B obeys actor C, actor A also obeys actor C in a transitive triad. In a hierarchy, relations are usually transitive, but transitivity is an effect that is also found in many other social relations, for instance, the relation “to know someone” is often transitive.

The balanced, clusterable, and ranked clusters triads are transitive, but the 012 triad, which contains a single asymmetric choice, is also transitive because it does not violate the rule that an indirect choice is paralleled by a direct choice; the 012 triad simply does not contain an indirect choice. Under the ranked clusters model, the 012 triad would mean that the three vertices belong to different clusters within a rank because of the two null dyads and, at the same time, that two vertices are ranked as a result of the asymmetric dyad. Clearly, this is a contradiction. Under the transitivity model, however, null choices are allowed between ranks. It is not necessary that someone at a lower rank chooses all people of higher rank; for instance, boys and girls may have separate rank systems that are perfect ranked clusters, but boys may ignore girls and vice versa.

The other model that relaxes the criteria of the ranked clusters model is the *hierarchical clusters model*, which is also called the hierarchical \bar{M} -clusters model. This model permits asymmetric dyads within a group as long as they are acyclic. Within a cluster, asymmetric dyads are supposed to express a mild form of ranking within a group and, like any kind of ranking, this ranking must be acyclic. The set of vertices v_1 , v_5 , v_7 , and v_8 (Figure 86) is an example of a hierarchical cluster. Vertex v_1 is connected to the other vertices by two 120C triads. Vertex v_7 does not reciprocate the choice by v_1 , who does not reciprocate v_5 's choice, so these three vertices are ranked; but vertices v_5 and v_7 are also part of a cluster with vertex v_8 because of the symmetric dyads.

The remaining five types of triads do not occur under any of the models. We may say that they are forbidden: They contradict all balance theoretic models and the assumptions about symmetric and asymmetric dyads on

Table 16. *Balance-theoretic models*

Model	Ties within a Cluster	Ties between Ranks	Permitted Triads
Balance	Symmetric ties within a cluster, no ties between clusters max. two clusters	None	102, 300
Clusterability	Idem no restriction on the number of clusters	Idem	+003
Ranked clusters	Idem	Asymmetric ties from each vertex to all vertices on higher ranks	+021D, 021U, 030T, 120D, 120U
Transitivity	Idem	Null ties may occur between ranks	+012
Hierarchical clusters	Asymmetric ties within a cluster allowed provided that they are acyclic	Idem	+120C, 210
No balance theoretic model (“forbidden”)			021C, 111D, 111U, 030C, 201

Note: A + sign indicates that all triads in previous rows are also permitted.

which the models are based. If these triads occur often, we ought to doubt whether we can cluster and rank the data according to balance theoretic principles. Table 16 summarizes the models.

Let us apply the balance theoretic models to the example network (Figure 86). Table 17 shows the number of triads found in this network arranged by the balance theoretic model to which they belong. Such a distribution is known as the *triad census*. The models are less restrictive in the order in which they are listed in Table 17, and it is standard practice to characterize the overall structure of a network by the least restrictive model that applies. After all, a less restrictive model covers all more restrictive models because it also permits their triads.

Unfortunately, social networks hardly ever conform perfectly to a balance theoretic model. Each type is likely to occur at least once, so the presence of one triad does not mean that the associated model must apply. We must compare the triad census to the distribution of triad types, which is expected by chance. If a particular triad type occurs clearly more often than expected by chance, the corresponding model may be said to guide or influence the relations: There is a tendency toward balance, clusterability, ranked clustering, transitivity, or hierarchical clusters in the network. If the models explain network structure, the forbidden triads should occur less frequently than predicted by chance.

Table 17. *Triad census of the example network*

	Type	Number of Triads	Expected	Model
3	102	22	7.56	Balance
16	300	1	0.06	
1	003	7	17.0	Clusterability
4	021D	3	7.56	Ranked
5	021U	3	7.56	Clusters
9	030T	4	5.81	
12	120D	5	1.12	
13	120U	2	1.12	
2	012	58	39.3	Transitivity
14	120C	2	2.24	Hierarchical
15	210	0	0.86	Clusters
6	021C	7	15.12	Forbidden
7	111D	4	5.81	
8	111U	2	5.81	
10	030C	0	1.94	
11	201	0	1.12	
TOTAL		120		

Table 17 shows the triad census of the example network. The column headed “Number of Triads” shows the triad counts in the example network, and the column “Expected” lists the numbers of triads that are expected by chance in a network of this size containing this number of arcs. If the actual frequencies are close to the expected frequencies, the network conforms to none of the balance theoretic models, and we may conclude that its structure is random from the point of view of balance theory. This, however, does not seem to be the case in our example.

In the example network, some types of triads occur substantially more often or less frequently than expected by chance. These frequencies are printed in italics in the table. The example network seems to contain relatively few clusterable triads but many balanced ones, some ranked clustering (120D) although other ranked clusters triads occur less often than expected (021D and 021U), and a tendency toward transitivity but not a surprising number of hierarchical cluster triads. The forbidden triads occur at chance level or less (021C and 111U), so we do not have to discard all balance theoretic models. The most appropriate model for this network seems to be the transitivity model, which allows for clustering and ranking but that does not require that all ties between ranks are asymmetric.

We should note that our expected frequencies take into account only the number of vertices and arcs in the network. Standard statistical tests of the triad census condition on indegree, outdegree, and number of mutual choices, which expresses the tendency to reciprocate choices at the

Table 18. *Triad census of the student government network*

	Type	Number of Triads (ni)	Expected (ei)	(ni-ei)/ei	Model
3	102	20	10.65	0.88	Balance
16	300	1	0.44	1.26	
1	003	10	10.05	-0.01	Clusterability
4	021D	9	10.65	-0.15	
5	021U	15	10.65	0.41	Clusters
9	030T	7	12.65	-0.45	
12	120D	14	3.76	2.72	
13	120U	6	3.76	0.60	
2	012	37	35.84	0.03	Transitivity
14	120C	1	7.52	-0.87	
15	210	5	4.47	0.12	Clusters
6	021C	16	21.29	-0.25	
7	111D	13	12.65	0.03	Forbidden
8	111U	6	12.65	-0.53	
10	030C	1	4.22	-0.76	
11	201	4	3.76	0.06	

Chi-square = 55.7613***.

Six cells (37.50%) have expected frequencies less than 5.

The minimum expected cell frequency is 0.44.

level of the dyad. These statistical tests may produce different results (see Chapter 13).

The triad census is an example of a research strategy that concentrates on local structure because it accounts only for ties within triads. The implications for the overall structure of the network are usually taken for granted and not much effort is made to assign vertices to clusters and ranks. Triadic analysis is the basis of statistical models that test hypotheses about the ties of individual actors: Why do they establish some ties and not others? Are their choices motivated by balance, transitivity?

Application

Network>
Info> Triadic
Census

In Pajek, it is very easy to compute the triad census: simply use the *Triadic Census* command in the *Network> Info* submenu. A dialog box asks whether the models should be reported, and if you choose this option, the triad types (“Type”), their actual frequencies (“Number of triads [ni]”), and the frequencies expected by chance (“Expected [ei]”) are reported. In addition, the relative difference between the actual and the expected number of triads is shown (“[ni-ei]/ei”), as well as a chi-square statistic testing the hypothesis that the actual frequencies are equal to the expected frequencies. This statistic is not reliable if expected frequencies are low. The triad counts and expected frequencies are also stored in two vectors.

Table 18 contains the triad census for the student government network. Three of the five forbidden triads appear less frequently than expected

by chance in the student government network (triads 021C, 111U, and 030C), which is also signaled by the negative value of the actual versus expected ratio, so there is some support that the underlying ideas of symmetric and asymmetric ties apply here.

Then, which structure characterizes the network? The student government network contains more between-groups triads (triad 102) than expected by chance, but the number of clusterability triads (003) is predicted by chance, so a partition into two clusters seems to suffice. Some ranked clusters triads appear as often as expected by chance; but the number of 120D triads, which signal asymmetric choices toward mutually connected pairs, is much higher than the expected frequency (the ratio of the difference (actual – expected) to expected number of triads is 2.72), so we should conclude that the network is ranked. Finally, the number of triads identifying the transitivity model (012) matches the amount expected in a random network, and the hierarchical cluster triads also do not appear more often than expected by chance.

A ranked clusters model seems to be the best choice for this data set because it permits triads 120D, 120U, and 021U, which appear substantially more often than chance; but it also permits the triads associated to more restrictive models (viz., the two balanced triads 300 and 102). In this way, the ranked clusters model contains all types of triads that occur clearly more often than expected by chance in the student government network.

The chi-square statistic is highly significant: The three stars indicate that it is statistically significant at the 0.001 level (one star represents statistical significance at the 0.05 level and two stars signal the 0.01 level), so the tendency toward ranked clustering is higher than expected by chance given the density of the network. We should note, however, that the expected frequency of six triads is less than five, and one triad (300) is expected to occur less than once. This casts some doubt on the reliability of the chi-square measure.

10.4 Acyclic Networks

In directed networks, ranking is associated with asymmetry: Arcs that represent an “ego obeys alter” relation point up, not down. Triadic analysis applies this principle to triads, that is, to local structure, but it can also be applied to the overall structure of a directed network. In a network that perfectly reflects a hierarchy, all arcs should point up and no arc should point down from a higher rank to a lower rank. This is called an acyclic network. It is important to note that such a network cannot contain cycles because a cycle would include arcs pointing up *and* arcs pointing down to return to its starting point.

An *acyclic network* contains no cycles.

We associate ranking with acyclic structures; for example, the soldier is subordinate to the sergeant, who is subordinate to the captain, who is subordinate to the colonel, and so on. An arc pointing in the wrong direction (e.g., the colonel obeys the soldier for whatever mysterious reasons) contradicts our idea of a hierarchy. This arc creates a cycle in the network, and it may even make the whole network cyclic in the sense that in the end everyone obeys everybody.

When acyclic structures point to ranking, cyclic structures are associated with clusters within one rank because they suggest equality among its vertices. In the short run (e.g., in a symmetric dyad) or in the long run (e.g., in a feedback loop that includes many vertices) a choice is reciprocated. From this point of view, we may partition a directed network into ranks: Cyclic subnetworks represent a cluster within a rank, and acyclic structures link ranks into a hierarchy.

Fortunately, it is easy to detect the cyclic parts of a network, and you have already mastered the technique to do it. Recall that a strong component is a maximal subnetwork in which each vertex is reachable to every other vertex (Chapter 3). There are paths in both directions between all pairs of vertices within a strong component, so a strong component is a cyclic (sub)network by definition. The arcs that are not part of a strong component cannot belong to a cycle, so they are part of an acyclic structure. In fact, if we shrink the strong components of a network, the network becomes acyclic.

Figure 88 shows the strong components in the student government discussion network (for the sake of clarity we replaced bidirectional arcs with fat edges). There are three components and, if you look carefully, the arcs between strong components all point in the same direction: from the white (red) component to the gray (green) component to the black (yellow) component. This is very clear in Figure 89, which shows the network with shrunk strong components. Note that the prime minister is included in the strong component in the top of the hierarchy, which is in line with his formal position.

Application

Network> In Chapter 3, we learned to identify the strong components in a network
Create with the command *Network> Create Partition> Components> Strong*.
Partition> This command creates a partition with a class for each strong component.
Components> We advise to set the minimum size of a component to 1; otherwise, the
Strong “red” component of advisor2 is not recognized.

Chapter 2 presented the command to shrink a network. The present case offers no complications: Make sure the student government

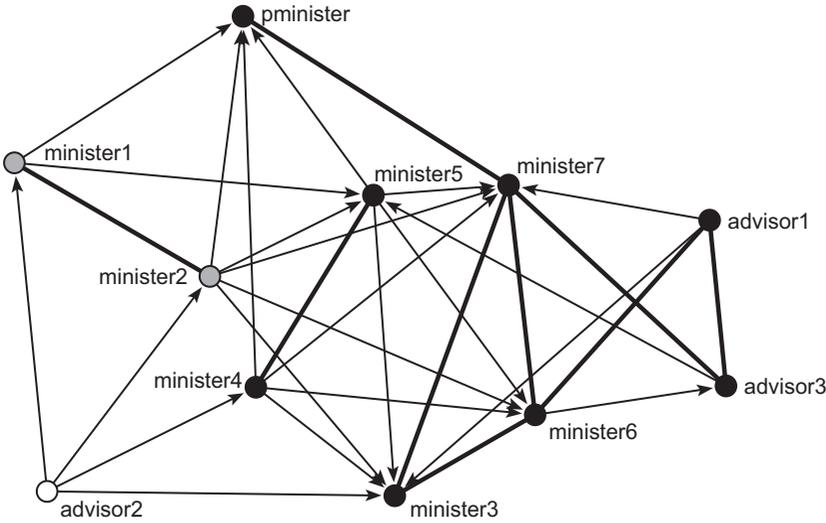


Figure 88. Strong components in the student government discussion network.

discussion network and the strong components partition are selected in the drop-down menus and execute the *Operations > Network + Partition > Shrink Network*. In the dialog boxes, require a minimum of one connection between clusters and do not shrink cluster number 0 or any other nonexistent cluster. If the network is shrunk according to a strong components partition, we obtain three vertices as shown in Figure 89.

Operations > Network + Partition > Shrink Network

Exercise I

Remove the arc from advisor3 to minister5 in the student government discussion network (right-click the vertex of advisor3 in the Draw screen and double-click the arc toward minister5 in the list). Determine the strong components in the modified network, and shrink the components. Which new rank appears now?

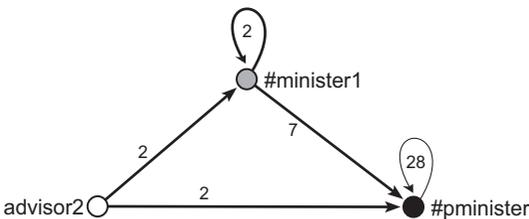


Figure 89. Acyclic network with shrunk components.

10.5 Symmetric-Acyclic Decomposition

In triadic analysis (Section 10.3), clusters within a rank must be complete. In many social networks, this criterion is too strict. Usually, social networks contain a limited number of choices made by each individual as a result of measurement techniques and social or cognitive limitations on the part of the investigated people. Respondents who are asked to recall with whom they discussed a particular matter informally, for instance, are likely to mention their most salient contacts rather than everyone with whom they have merely touched on the subject. A network constructed from these data will not yield complete clusters.

Conversely, the strong components do not seem to be sufficiently strict to identify a cluster within a rank (Section 10.4). In the student government discussion network, it would be nice if we could subdivide the black (yellow) component, which contains a heterogeneous group of actors at present: advisors, ministers, and the prime minister.

The *symmetric-acyclic model* is a suitable alternative. It uses a version of the symmetry versus asymmetry principle that is less strict than the balance theoretic assumptions but stricter than the acyclic character of strong components. It assumes that vertices that are linked by symmetric (i.e., mutual) choices directly or indirectly belong to one cluster, hence to one rank. Clusters that are linked by asymmetric ties only are ranked.

This model is especially less restrictive with respect to the internal structure of clusters because it allows for asymmetric and null dyads within a cluster; for example, if vertex u is linked to vertices v and w by symmetric ties, they belong to one cluster regardless of the tie between v and w , which may be symmetric, asymmetric, or null. Balance theoretic models never allow null dyads within a cluster, and asymmetric dyads may occur only under special conditions in the hierarchical clusters model.

It is easy to identify clusters of vertices that are connected by mutual choice: Just delete all unilateral arcs from the network and compute components. Each component is a cluster of vertices, which are linked by symmetric dyads. Figure 90 shows the four clusters in the student government network. Note that the dark gray (blue) and black (yellow) symmetric clusters of Figure 90 are strong components in the overall network (Figure 88).

The largest strong component of the original network, however, combines two symmetric clusters: the cluster of minister4 and minister5 with the cluster of the prime minister. The two symmetric clusters are linked into one strong component because the arcs between these clusters do not point in the same direction. In Figure 91, we can see that the symmetric cluster of minister4 and minister5 predominantly sends asymmetric ties to the cluster of the prime minister, but they receive one asymmetric choice from that cluster (viz., the arc from advisor3 to minister5; check

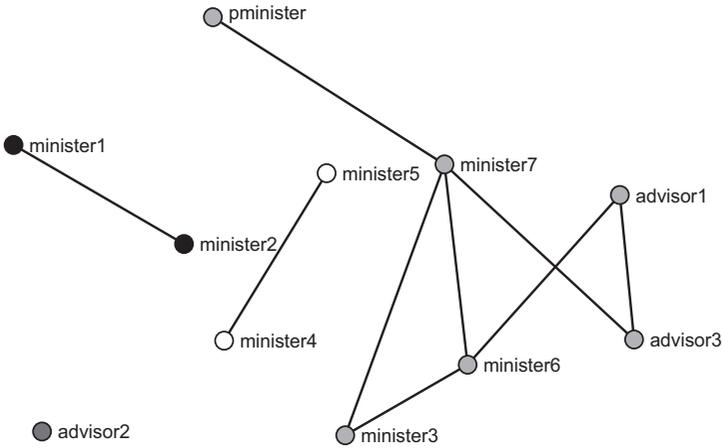


Figure 90. Clusters of symmetric ties in the student government network.

Figure 88). When we ignore this arc, we obtain smaller strong components that match the symmetric clusters perfectly.

Clusters of vertices that are reachable through symmetric ties are preferable over strong components because mutual choice is a clear indication of group membership and equality with respect to ranking. We therefore recommend paying close attention to strong components in which not all vertices are linked by paths of mutual choices. Elimination of a single arc may split this component into smaller clusters that are asymmetrically ordered.

A stricter interpretation of the symmetric-acyclic model forbids all asymmetric ties inside clusters. In other words, vertices within a cluster are

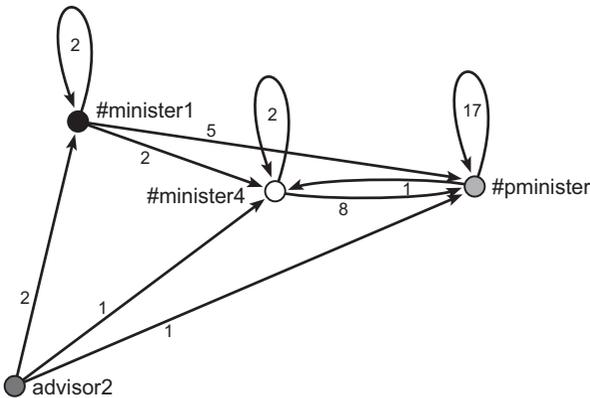


Figure 91. Discussion network shrunk according to symmetric clusters.

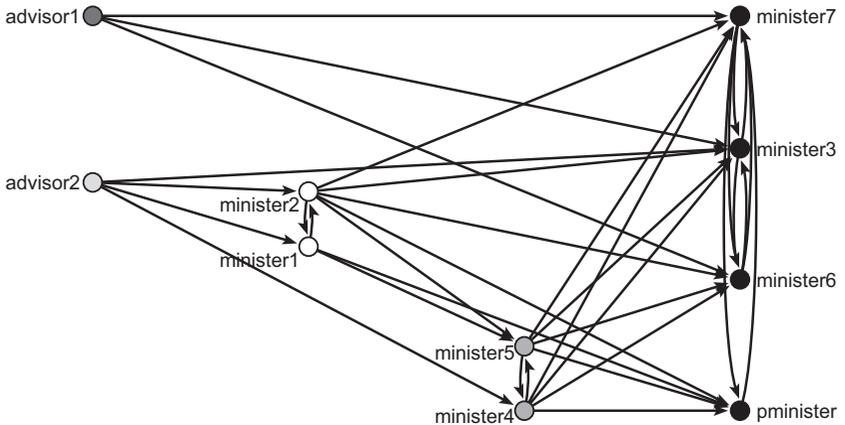


Figure 92. Symmetric components in the (modified) student government discussion network.

either tied by a symmetric tie or no tie at all (a null tie), and all asymmetric choices are situated between clusters. In the largest cluster of the student government network, which contains the prime minister, three asymmetric ties are found and all of them involve the advisors. The ministers and the prime minister are linked by symmetric ties only. If we would delete advisor3, who is offending the ranking between two symmetric clusters and who is also involved in an asymmetric tie within the top cluster, and we ignore the arc from minister6 to advisor1, we obtain a decomposition that satisfies the strictest criteria of the symmetric-acyclic model (see Figure 92).

Note that this decomposition nicely reflects the formal positions of the students: The advisors are on the lower ranks, the prime minister is on the highest rank, and the ministers are in the middle or top ranks. If the prime minister had not chosen minister7, he or she would have had the top rank for himself or herself. In this case, the informal ranking is more differentiated than the formal ranking because the ministers are spread over three ranks.

Figure 92 illustrates an important characteristic of the symmetric-acyclic model. Often, the order of the symmetric clusters is not completely determined. Advisor1, for example, must be ranked below the black cluster containing the prime minister. It is unclear, however, whether this advisor should be ranked with advisor2, minister2, or minister5 or whether it occupies a rank of its own. There is no path of asymmetric ties between advisor1 and advisor2, minister2, or minister5 that defines his or her position with respect to them. This is called a *partial order*: We know the order of some pairs of vertices but not of all pairs. As a result, the classification

of vertices according to rank does not necessarily yield a single result. In Figure 92, advisor1 could have been drawn at several other levels.

Application

Pajek contains a command to find clusters of symmetrically linked vertices and ranks: *Network> Create Hierarchy> Symmetric-Acyclic*. This command follows the logic outlined earlier.

First, the command finds the components of symmetrically linked vertices. It produces a new network with edges instead of bidirected arcs like Figure 88. Then it creates a network without the remaining (unilateral) arcs, and it computes a partition of weak components. Each weak component is a cluster of vertices that are reachable through symmetric ties. When you draw the network and partition, you obtain Figure 90.

Second, the procedure shrinks the clusters of symmetrically linked vertices. The shrunk network is very convenient for finding symmetric clusters that are linked by asymmetric ties in both directions. If you draw this network, you obtain a sociogram that is similar to that in Figure 91. In this drawing, you may detect symmetric clusters that are nearly asymmetrically linked, such as the #minister4 and #pminister clusters.

Finally, the procedure repeats the first and second steps until it does not encounter any symmetrically linked vertices or clusters. Then, all strong components have been shrunk, and the network is acyclic by definition. After the first shrinking of the student government discussion network (see Figure 91), the #pminister and #minister4 clusters are connected by a bidirectional arc: They are symmetrically linked. In the next step, they are concatenated into one new symmetric cluster and shrunk. In the resulting network, no vertices or clusters are symmetrically linked, so the procedure stops. Note that in this example the last network created by the *Symmetric-Acyclic* command contains no lines. You should select the last shrunk network (labeled something like “Shrinking N5 according to C3”) as the final result of the analysis.

The shrunk network that results from the symmetric-acyclic decomposition is acyclic, so we may determine the order of the ranks with the *Depth Partition> Acyclic* command from the *Network> Acyclic Network> Create Partition* submenu provided that you delete the loops first (*Network> Create New Network> Transform> Remove> Loops*), which are created when the network is being shrunk. Draw the shrunk network and its depth partition according to layers (*Layers> In y Direction*) to obtain a graphical representation of the ranks. It will look like Figure 93: Advisor2 advises the symmetric cluster of minister1, who advises the symmetric cluster of the prime minister. When you move the gray vertex manually in the Draw screen, you will see that there is also a direct arc from the advisor to the cluster of the prime minister.

```
Network>
Create
Hierarchy>
Symmetric-
Acyclic
```

```
Network>
Create New
Network>
Transform>
Remove>
Loops
```

```
Network>
Acyclic
Network>
Create
Partition>
Depth
Partition>
Acyclic
```

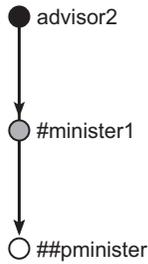


Figure 93. The order of symmetric clusters according to the depth partition (acyclic).

Network>
Create
Partition>
Components>
Strong

If you want to draw the original network in layers that represent the ranks, you have to expand the depth partition to the original network. Because the shrunk vertices in the acyclic network are the strong components (size 1 and larger) in the original network, you can use a strong components partition of the original network to expand the depth partition. Create this partition with the *Network> Create Partition> Components> Strong* command, making sure that the original network is selected in the *Network* drop-down menu.

Partitions>
Expand
Partition> First
according to
Second (Shrink)

Now, you can expand the depth partition of the shrunk network to the original network. Select the depth partition of the shrunk network as the first partition, and select the strong components partition as the second partition. Then choose the *First according to Second (Shrink)* command from the *Partitions> Expand Partition* submenu. Pajek asks which class in the strong components partition was not shrunk (zero or a number that does not occur in the strong components partition will do), and it creates a new partition that assigns each vertex in the original network to its depth in the symmetric-acyclic decomposition. You may draw this partition in layers and move vertices within each layer to obtain an image of the ranking. In Figure 94, we replaced all bidirected arcs with edges (*Network> Create New Network> Transform> Arcs→Edges> Bidirected only*) and we rotated the layout (*[Draw] Options> Transform> Rotate 2D*), so the ranks increment from left to right. Note that we cannot see all arcs within the white rank, but it is clear that all arcs between ranks point in the same direction (right).

Network>
Create New
Network>
Transform>
Arcs→Edges>
Bidirected only

[Draw]
Options>
Transform>
Rotate 2D

In the symmetric-acyclic decomposition, the resulting strong components are not necessarily symmetric clusters. In the student government network, for example, a strong component combines the two symmetric clusters #pminister and #minister4 (Figure 91). We have found a decomposition that satisfies the weak version of the symmetric-acyclic model.

The stronger version of this model does not allow asymmetric ties within clusters, so we have to inspect the ties within each cluster to find out whether the stronger model applies. Because the strong components

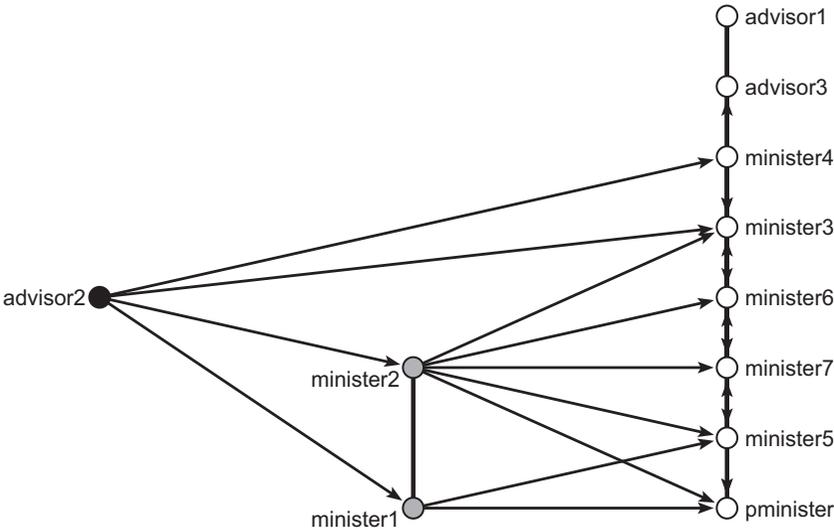


Figure 94. Ranks in the student government discussion network.

are the clusters, we may simply remove all lines between strong components and check whether the resulting network contains bidirected arcs only. Select the original network and the strong components partition to this network and remove the lines between components with the *Operations> Network + Partition> Transform> Remove Lines> Between Clusters* command. Now, replace bidirected arcs with edges (command *Network> Create New Network> Transform> Arcs→Edges> Bidirected only*), and check the number of arcs in the network (*Network> Info> General*). If there are no arcs, all strong components are symmetric clusters. Strong components containing arcs, however, are not symmetric clusters, so they do not satisfy the stronger version of the symmetric-acyclic model. In our example, the strong component containing the prime minister contains several unilateral arcs. This network does not satisfy the requirements of the strong symmetric-acyclic model.

*Operations>
Network +
Partition>
Transform>
Remove Lines>
Between
Clusters*

*Network>
Create New
Network>
Transform>
Arcs→Edges>
Bidirected only*

*Network>
Info> General*

Exercise II

In Exercise I, you removed the arc from advisor3 to minister5. In this network, do the symmetric clusters conform to the weak or strong symmetric-acyclic model?

10.6 Summary

Society and, in more detail, the human group is characterized by clustering and ranking. Like-minded people cluster into cohesive groups on

the basis of mutual positive ties. Rivalry between groups is expressed by negative or absent ties. In addition, social groups are usually ranked such that dominant groups occupy higher ranks or strata. Asymmetric ties indicate ranking: A positive choice received from a lower ranked group is not reciprocated.

Society and the social group are generally considered to contain a limited number of discrete ranks. In this chapter, we present structural models of discrete ranks that have evolved from balance theory. The first two balance theoretic models – balance and clusterability (see Chapter 4) – are confined to the clustering of social entities; they tacitly assume that there is no ranking, so asymmetric ties and unclusterable semicycles are not allowed. A third model, the ranked clusters model, regards a social system as a set of ranks where each rank contains one or more clusters. Positive arcs connect entities within a cluster, but no arcs connect different clusters at one level, as in the clusterability model. In addition, asymmetric dyads connect clusters at different ranks, where arcs point from lower to higher levels.

The ranked clusters model represents a simple hierarchy in which each pair of clusters or vertices is unambiguously ranked. Often, social systems are more complicated, containing incomplete hierarchies or even different hierarchies that are not compatible. The social cleavage between girls and boys is a simple example. There is a hierarchy of boys and a hierarchy of girls, but nobody is interested in the members of the other gender regardless of their ranking. This phenomenon is captured in the fourth balance theoretic model, which is known as the transitivity model. A fifth model, called the hierarchical clusters model, is even more permissive because it allows for ranking within a group. Asymmetric dyads within a cluster of otherwise symmetrically connected people indicate ranking in this model.

In a simple directed network, a balance theoretic model is identified by the types of triads that it permits, so we may count the number of times each triad type occurs in the network – this is called the triad census – and find the appropriate model. Unfortunately, social networks seldom fit a balance theoretic model perfectly, so we need statistical tests to determine which triad types and models occur more often than expected by chance. Triadic analysis is the basis for statistical modeling rather than exploring the structure of clusters and ranks.

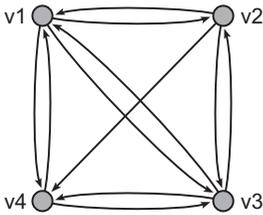
By definition, ranking is acyclic, so cyclic parts of the network either represent clustering within a rank or they contain complicated or imperfect ranking. Recall that a strong component contains vertices that are connected by paths in both directions, so strong components are cyclic subnetworks. If we shrink the strong components, the resulting network is acyclic and can be partitioned into ranks. Next, we inspect each strong

component for clusters and complicated or imperfect ranking. In a simple directed network, mutual (positive) choices are the backbones of clusters, so we look for clusters of vertices that are directly or indirectly linked by symmetric ties. The ties between the clusters tell us whether they belong to one rank or to different ranks.

This is an exploratory procedure for detecting the clusters and ranks that best fit a network, but it does not tell us whether the fit is satisfactory. With enough effort and modification, we can probably find clusters and ranks that even fit a random network. As elsewhere in this book, we must make sense of our results. The clusters and ranks should be meaningful with respect to other information that we have about the social entities in the network.

10.7 Questions

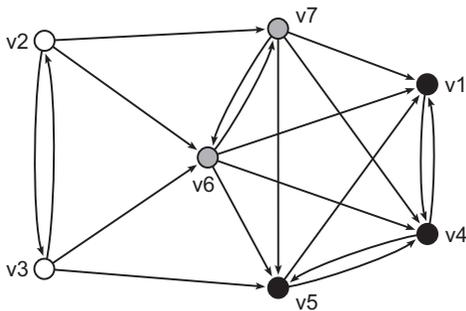
1. How many dyads does the following network contain and how many types of dyads?



- a. Six dyads and one type
 - b. Six dyads and two types
 - c. Eleven dyads and one type
 - d. Eleven dyads and two types
2. Assemble the triad census (type of triad and frequency of occurrence) of the network shown above by hand.
 3. Which balance theoretic model characterizes the network of Question 1?
 - a. The balance model
 - b. The hierarchical clusters model
 - c. The balance and hierarchical clusters models
 - d. No balance theoretic model fits this network
 4. The table that follows shows the triad census of a directed network. Choose the appropriate balance theoretic model for this network, and justify your choice.

No.	Type	Number of Triads (ni)	Expected (ei)	(ni-ei)/ei	Model
3	102	41	10.7	2.83	Balance
16	300	2	0.1	15.63	
1	003	8	19.4	-0.59	Clusterability
4	021D	0	10.7	-1.00	
5	021U	3	10.7	-0.72	Clusters
9	030T	7	9.2	-0.24	
12	120D	14	2.0	6.13	
13	120U	6	2.0	2.05	
2	012	77	49.9	0.54	Transitivity
14	120C	0	3.9	-1.00	
15	210	0	1.7	-1.00	Clusters
6	021C	2	21.4	-0.91	
7	111D	3	9.2	-0.67	Forbidden
8	111U	2	9.2	-0.78	
10	030C	0	3.1	-1.00	
11	201	0	2.0	-1.00	

5. Explain why triad 201 is not allowed under the hierarchical clusters model.
6. Assign the vertices of the following network to clusters and ranks. Vertex colors indicate strong components. Does your decomposition satisfy the weak or the strong symmetric-acyclic model?



7. Which of the following statements about symmetric-acyclic decomposition is correct?
 - a. In the strong symmetric-acyclic model, clusters do not contain asymmetric dyads.
 - b. In the strong symmetric-acyclic model, vertices in different ranks are always connected by asymmetric dyads.
 - c. In the weak symmetric-acyclic model, clusters contain mutual or null dyads only.

- d. In the weak symmetric-acyclic model, all asymmetric dyads point from a lower rank to a higher rank.

10.8 Assignment

In 1976, a literary critic published an essay about contemporary Dutch prose. In his essay, he distinguished among four trends or movements: narrators, including the authors Donkers, Kooiman, Matsier, and Meijssing; alienators, including Van Marissing, Robberechts, and Vogelaar; petty realism, including Hart, Hiddema, Luijters, Meinkema, Plomp, and Sijtsma; and decadence, including Siebelink and Joyce & Co. Find out whether this classification matches the ranks and clusters in the networks of critical attention in 1976. The simple directed network `literature_1976.net` contains an arc between two people if the first has paid attention to the second in an interview or review. Hint: Create a partition reflecting the classification of the authors according to literary movement.

10.9 Further Reading

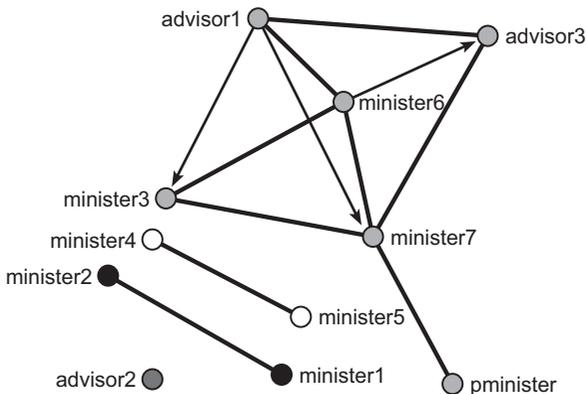
- Chapters 6 and 14 of S. Wasserman and K. Faust's *Social Network Analysis: Methods and Applications* (Cambridge: Cambridge University Press, 1994) provide an excellent overview over balance theoretic models and the analysis of triads. An overview over the work of Davis, Holland, and Leinhardt on triads can be found in J. A. Davis' article "The Davis/Holland/Leinhardt Studies: An Overview." In P. W. Holland and S. Leinhardt (eds.), *Perspectives on Social Network Research* (New York: Academic Press, 1979).
- C. Flament, in *Applications of Graph Theory to Group Structure* (Englewood Cliffs, NJ: Prentice Hall, 1963), proved that triads suffice for detecting balance in complete signed graphs and Davis ("Clustering and structural balance in graphs." *Human Relations*, 20 [1967], 181–7) did this for clusterability. For more information on the hierarchical \bar{M} -clusters model, see E. C. Johnsen, "Network macrostructure models for the Davis–Leinhardt set of empirical sociomatrices." *Social Networks* 7 (1985), 203–24.
- Symmetric-acyclic decomposition of directed networks was introduced in P. Doreian, V. Batagelj, and A. Ferligoj, "Symmetric-acyclic decompositions of networks." In *Journal of Classification* 17 (2000), 3–28.

- For more information on the student government data, consult V. Hlebec, “Recall versus recognition: Comparison of two alternative procedures for collecting social network data.” In A. Ferligoj and A. Kramberger (eds.), *Developments in Statistics and Methodology* (Ljubljana: FDV, 1993). Results of an analysis of the Dutch literary criticism data are reported in W. de Nooy, “A literary playground: Literary criticism and balance theory.” *Poetics* 26 (1999), 385–404.

10.10 Answers

Answers to the Exercises

- I. When the arc from advisor3 to minister5 has been removed, four strong components are found. Now, minister4 and minister5 constitute a strong component on their own, which is separated from the component containing the prime minister. This may be regarded as a new rank in the student government discussion network, between the rank of minister1 and minister2, who send arcs to minister4 and minister5, and the rank including the prime minister, three ministers, and two advisors, who receive only arcs from them. The arc from advisor3 to minister5 created a cycle containing minister4 and minister5 in the original network.
- II. Without the arc from advisor3 to minister5, you obtained four strong components. When you remove the lines between clusters with the *Operations > Network + Partition > Transform > Remove Lines > Between Clusters* command in the Main screen and change the bidirectional arcs into edges (*Network > Create New Network > Transform > Arcs → Edges > Bidirected only*), you can easily see that three components contain edges only, but the largest component (including the prime minister) still contains three arcs (see the figure that follows). All



arcs involve ties between a minister and an advisor. We must conclude that the weak symmetric-acyclic model applies.

Answers to the Questions in Section 10.7

1. Answer b is correct. A dyad is a pair of vertices and the lines among them. In a network with four vertices, such as the example, there are six different pairs of vertices, so there are six dyads. In a simple directed network, a dyad is mutual (arcs in both directions), asymmetric (an arc in one direction), or null (no arcs). In the example, five dyads are mutual and the sixth (v2 and v4) is asymmetric, so there are two types of dyad.
2. The table below shows the triad census.

No.	Type	Number of Triads	Model
3	102	0	Balance
16	300	2	
1	003	0	Clusterability
4	021D	0	Ranked
5	021U	0	Clusters
9	030T	0	
12	120D	0	
13	120U	0	
2	012	0	Transitivity
14	120C	0	Hierarchical
15	210	2	Clusters
6	021C	0	“Forbidden”
7	111D	0	
8	111U	0	
10	030C	0	
11	201	0	
TOTAL		4	

3. Answer b is correct. In Question 2, you have found two balanced triads (300) and two hierarchical cluster triads (210). The hierarchical clusters model allows for balanced triads, but the reverse is not true. Therefore, the hierarchical clusters model is the appropriate model for this network.
4. The transitivity model is appropriate here. The forbidden triads do not occur (030C and 201) or occur less often than in random networks, so a balance theoretic model characterizes this network. The hierarchical cluster triads do not occur, but the network contains far more transitivity triads (012) than expected by chance. Two ranked clusters triads (120D and 120U) and both balanced triads appear more often than expected by chance, but they are also permitted by the

transitivity model, so we may conclude that the transitivity model characterizes this network.

5. Triad 201 contains two symmetric choices and one null dyad. In the hierarchical clusters model, vertices connected by symmetric ties belong to one (hierarchical) cluster. A null dyad means that two vertices belong to different clusters. Therefore, two vertices belong to different clusters because of the null dyad and to the same cluster because of the path of symmetric choices at the same time. This is a contradiction, so this triad is not allowed.
6. Arcs between strong components point from the white to the gray component and from the gray to the black component. Clearly, there are three ranks; the black rank is the top rank and the white rank is at the bottom. The vertices in the white and gray component are connected by mutual arcs, but two black vertices (v1 and v5) are connected by an asymmetric tie, so the decomposition does not satisfy the criteria of the strong symmetric-acyclic model for all strong components.
7. Statement a is correct. In the strong symmetric-acyclic model, clusters contain no asymmetric dyads; hence, all asymmetric dyads are found between ranks. This is not the case in the weak symmetric-acyclic model, where asymmetric dyads may occur within clusters (answers c and d). Answer b is not correct because vertices at different ranks can also be connected by null dyads.

Genealogies and Citations

11.1 Introduction

Time is responsible for a special kind of asymmetry in social relations, because it orders events and generations in an irreversible way. Social identity and position is partially founded on common ancestors, whether in a biological sense (birth) or in an intellectual manner: citations by scientists or references to predecessors by artists. This is social cohesion by common descent, which is slightly different from cohesion by direct ties (see Part II). Social communities and intellectual traditions can be defined by a common set of ancestors, by structural relinking (families that intermarry repeatedly), or by long-lasting cocitation of papers.

Pedigree is also important for the retrospective attribution of prestige to ancestors. For example, in citation analysis the number of descendants (citations) is used to assign importance and influence to precursors. Genealogy is the basic frame of reference here, so we discuss the analysis of genealogies first.

11.2 Example I: Genealogy of the Ragusan Nobility

Ragusa, which is now known as Dubrovnik (Croatia), was settled on the coast of the Adriatic Sea in the seventh century. For a time, it was under Byzantine protection, becoming a free commune as early as the twelfth century. Napoleon, having destroyed the Venetian Republic in 1797, put an end to the Republic of Ragusa in 1806. It came under Austrian control until the fall of the Austro-Hungarian monarchy in 1918.

In Ragusa, all political power was in the hands of male nobles older than eighteen years. They were members of the Great Council (*Consilium majus*) who had the legislative authority. Every year, eleven members of the Small Council (*Consilium minus*) were elected. Together with a

duke, the Small Council had both executive and representative authority. The main power was in the hands of the Senate (*Consilium rogatorum*), which contained forty-five members elected for one year. This organization prevented any single family, such as the Medici family in Florence, from prevailing. Nevertheless historians agree that the Sorgo family was among the most influential.

The Ragusan nobility evolved from the twelfth century to the fourteenth century and was finally established by statute in 1332. After 1332, no new families were accepted until the large earthquake in 1667. A major problem facing the Ragusan noble families was that, because of their decreasing numbers and the lack of noble families in the neighboring areas, which were under Turkish control, they became more and more closely related – marriages between third and fourth removed relatives were frequent. It is interesting to analyze how families of a privileged social class organized their relations by marriage and how they coped with the limited number of potential spouses for their children.

The file `Ragusan.ged` contains the members of the Ragusan nobility from the twelfth to the sixteenth centuries, their kinship relations (parent–child); their marriages; and their (known) years of birth, marriage, and death. Note that this is not an ordinary network file, because it contains attributes and ties of vertices. The extension `.ged` indicates that it is a GEDCOM file, which is the standard format for genealogical data, as explained in the next section. The genealogy is large; it contains 5,999 persons. For illustrative purposes, we selected the descendants of one nobleman, Petrus Gondola, in the file `Gondola_Petrus.ged` (336 persons).

11.3 Family Trees

Across the world, many people are assembling their family trees. They visit archives to collect information about their ancestors in registers of births, deaths, and marriages. Because in most Western societies family names are the usual entries in these registers and family names are the father's surname, a patrilineal genealogy is reconstructed, in which father–child relations rather than mother–child relations connect generations. In addition, marriages are included in the family tree.

Figure 95 shows a part of the Gondola family tree, which includes three generations of descendants of Petrus Gondola, who was born in 1356. Note that children born to a Gondola father are included because they receive the Gondola surname. Children of a Gondola mother are not included because their surname assigns them to another family in this historiography of a family name. An exception would be a Gondola mother

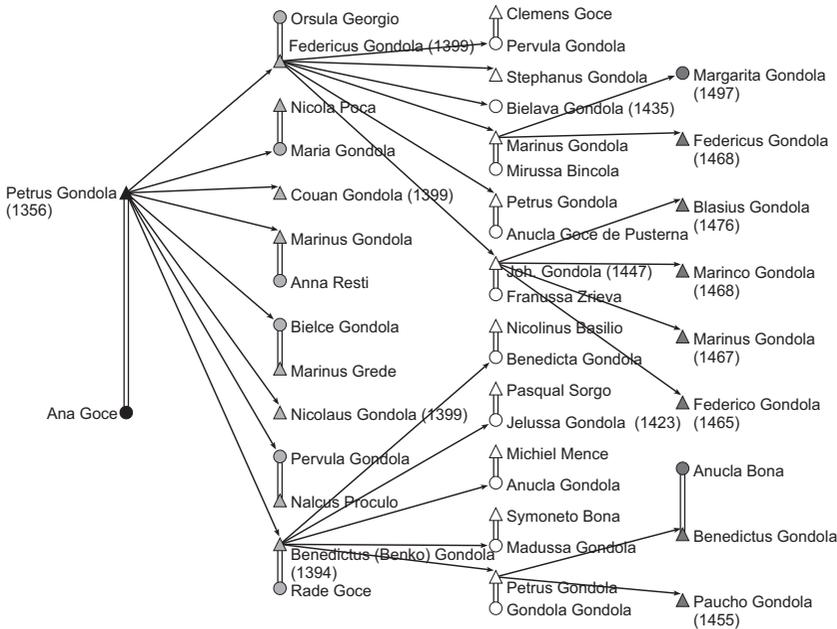


Figure 95. Three generations of descendants to Petrus Gondola (years of birth).

who married a Gondola father, but, as shown in Figure 95, this does not occur among the descendants.

In principle, genealogies contain persons as units and two types of relations among persons: birth and marriage. A person may belong to two nuclear families: a family in which he or she is a child and a family in which it is a parent. The former is called the *family of child or orientation* (FAMC) and the latter is the *family of spouse or procreation* (FAMS). Petrus Gondola’s family of procreation, for example, contains his wife and eight children, and it is identical to the family of orientation of each of his children. A husband and wife have the same family of procreation, but they have different families of orientation unless they are brother and sister.

The standard data format for genealogies (GEDCOM) uses the double coding according to family of orientation and family of procreation. In addition, it has facilities to store all sorts of information about the persons and events (e.g., about their marriage), so we advise using this data format for the collection and storage of genealogical data. On the Internet, excellent free software and several databases of genealogical data are available (see “Further Reading”).

In a representation of a genealogy as a network, family codes are translated to arcs between parents and children. In a sociogram of kinship ties

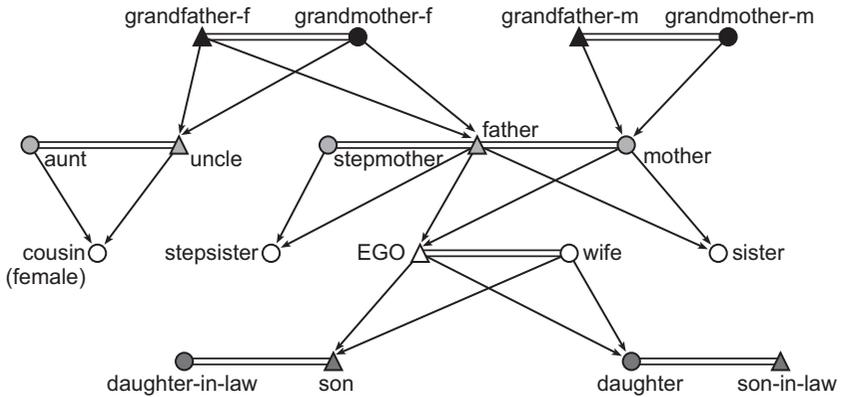


Figure 96. Ore graph.

that is known as the *Ore graph* (Figure 96), men are represented by triangles, women by ellipses, marriages by (double) lines, and parent–child ties by arcs. Note that the arcs point from parent to child following the flow of time.

In contrast to the family tree, fathers and mothers are connected to their children in an Ore graph. This greatly simplifies the calculation of kinship relations because the length and the direction of the shortest semipath between two individuals define their kinship tie; for instance, my grandparents are the vertices two steps up from me in the Ore graph. They are relatives in the second remove because two births are included in this path. In a patrilineal family tree, relatives from my mother’s side (e.g., her parents and brother) are not included, so it is impossible to establish my kinship tie with them. In the Ore graph, it is possible to distinguish between blood relations and marriage relations, so we may calculate the remove in a strict sense, that is, ignoring marital relations, or in a loose sense, including them and considering them relations with zero distance.

In the standard display of a kinship network, marriages and siblings are drawn at the same layer, and layers are either top-down (Figure 96) or ordered from left to right (Figure 95). A layer contains a *genealogical generation*: grandparents versus parents, uncles and aunts versus children, nieces, and nephews. Such are the generations that we experience during our lives. From a social point of view, however, we define generations as birth cohorts (e.g., the generation of 1945–60). In contemporary Western societies, *social generations* contain people who were born within a period of approximately fifteen years. Genealogical generations overlap with social generations to a limited extent. For four or more generations, genealogical generations may group people of very different ages as a result of early marriage and childbearing in one branch of the family and late marriage in another branch. The birth years of the

great-grandchildren of Petrus Gondola, for instance, range from 1455 (Paucho) to 1497 (Margarita; see Figure 95). Biologically, the former could have been the latter's grandfather. As a consequence, Paucho's grandson could have married Margarita, causing a *generation jump* in the genealogy because it would connect a third-degree descendant of Petrus Gondola (viz., Margarita) to a fifth-degree descendant (Paucho's grandson).

The Ore graph is a very useful instrument for finding an individual's *ancestors* (*pedigree*) and *descendants* from both the father's and the mother's side. In addition, it is easy to count *siblings* and trace the *closest common ancestor* of two individuals.

Application

Genealogical data in GEDCOM format can be read directly by Pajek. To obtain the Ore graph, make sure that the option GEDCOM – *Pgraph* in the *Options> Read – Write* submenu is *not* selected before you open the GEDCOM file. When you check the option *Ore: Different relations for male and female links*, marriages receive line value and relation number 3 (drawn as double lines), father–child ties have a line value and relation number 1 (solid lines), and mother–child ties have a value and relation number 2 (dotted lines). This is particularly useful if you want to extract patrilineal ties from the Ore graph. In all other cases, it is better not to check this option, so all parent–child ties have line value and relation number 1. Then, open a GEDCOM file in the usual way with the *File> Network> Read* command, but select the option *Gedcom files (*.ged)* in the *File Type* drop-down menu of the Read dialog screen.

Reading the GEDCOM file, Pajek translates family numbers to parent–child ties and it creates two partitions and four vectors. The first obtained partition identifies vertices that are brothers and sisters, that is, children born to the same father and mother. Stepbrothers and stepsisters from a parent's remarriage are grouped separately, and vertices without parents in the network are collected in class 0. The second partition is the gender partition (class 1 for men, 2 for women). The vectors contain each person's sequential number in the GEDCOM file and his or her year of birth, marriage, and death. Unknown dates are represented by vector value 999999998. You may inspect the dates with the *Vector> Info* procedure in the usual way (see Section 2.5 in Chapter 2).

The *genealogical generations* of the Ore graph can be obtained with the command *Genealogical* from the *Network> Acyclic Network> Create Partition> Depth Partition* submenu. An acyclic depth partition is not possible because the marriage edges are cyclic: A husband is married to his wife, and a wife is married to her husband at the same time. Draw the network in layers according to the genealogical depth partition (*Layers> In y Direction* in the Draw screen) and optimize it in the usual way (*Layers>*

[Main]
Options> Read
– Write>
GEDCOM –
Pgraph, Ore:
Different
relations for
male and
female links

Vector> Info

Network>
Acyclic
Network>
Create
Partition>
Depth
Partition>
Genealogical

Layers> In y
Direction

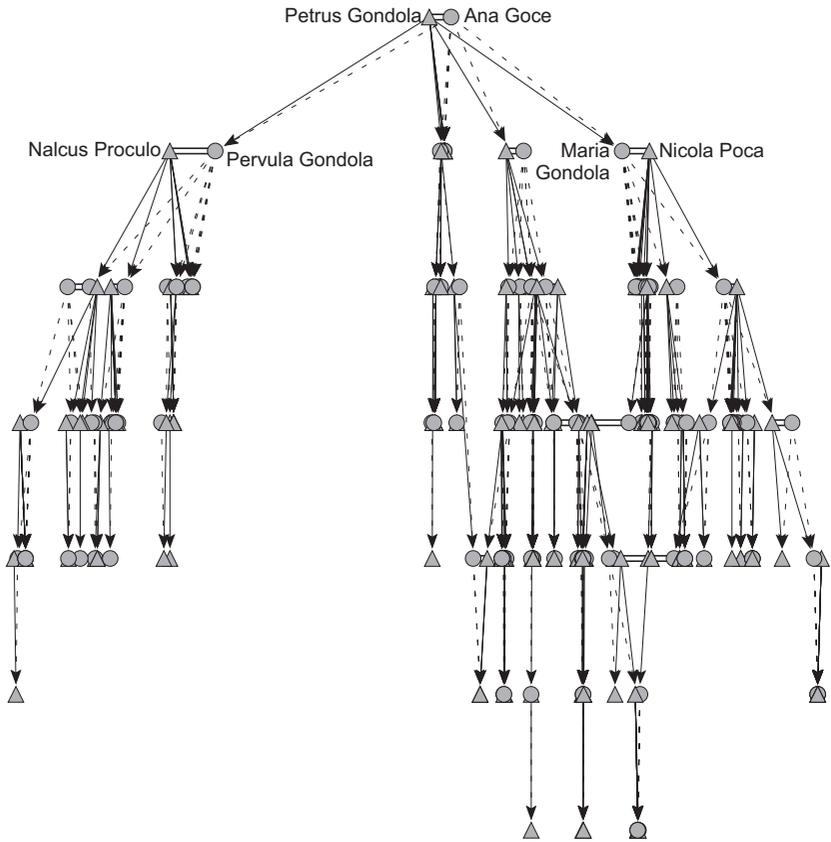


Figure 97. Descendants of Petrus Gondola and Ana Goce.

Layers>
Optimize
Layers in x
Direction
Layers>
Averaging x
Coordinate
[Main]
Options> Read
- Write> Ore:
Different
relations for
male and
female links
Network>
Create New
Network>
Transform>
Line Values

Optimize Layers in x Direction). To focus on the distinct branches in the genealogy rather than the vertices, use the *Averaging x Coordinate* command from the *Layers* menu. Usually, the *Forward* option works well, but you may have to apply it more than once to clearly separate distinct branches as in Figure 97.

The length of the shortest semipath in a symmetrized Ore graph is the *remove* or *degree of a family relation*, provided that all parent-child ties have a line value of 1 and marriage lines have a line value of 0. Therefore, you must open the GEDCOM file with the option *Ore: Different relations for male and female links* not checked in the *Option> Read - Write* submenu. Marriage lines have value 3 (not 0 as in older versions of Pajek), so you must replace all line values 3 with value 0 before you calculate the degree of a family relation. You can do that in three steps all involving menu *Network> Create New Network> Transform> Line Values*. First use the *Add Constant* option and enter -3. In this way marriage links get

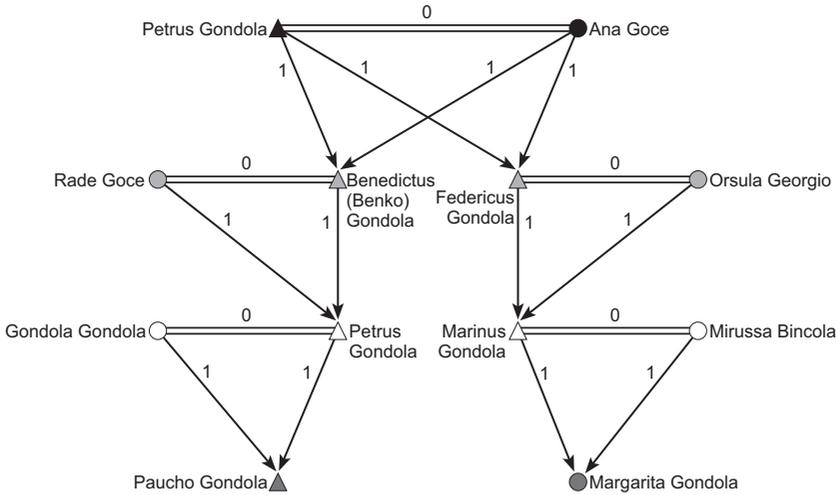


Figure 98. Shortest paths between Paucho and Margarita Gondola.

value 0, but now parent–child links have value -2 ; therefore, apply commands *Absolute* and subsequently *Multiply* by (0.5) to set all values of parent–child links back to 1 (marriage links stay 0).

First, decide whether you want to include marital relations in the calculation of the degree of family relations. If not, remove the edges from the network (*Network* > *Create New Network* > *Transform* > *Remove* > *all Edges*). Then, symmetrize the Ore graph (*Network* > *Create New Network* > *Transform* > *Arcs* → *Edges* > *All*; do not remove multiple lines), and use the *All Shortest Paths between Two Vertices* command to obtain the geodesics between two individuals in the network. When asked, do not ignore (forget) the values of the lines, because a marriage link should not contribute to the length of the semipath and hence to the remove of the relation. The length of the shortest paths, which is the distance between the vertices in the symmetrized network, is printed in the Report screen. Among the descendants of Petrus Gondola (Figure 95), for instance, Paucho Gondola (born in 1455) is a relative of Margarita Gondola (born in 1497) in the sixth remove.

Pajek creates a new network of the geodesics it has found and a partition that identifies the vertices on the geodesics in the original network provided that you requested this in one of the dialog boxes. If we extract these vertices from the original directed network (*Operations* > *Network* + *Partition* > *Extract* > *SubNetwork Induced by Union of Selected Vertices* and choose class 1) and relocate the vertices, we obtain the network shown in Figure 98. Note the triangles containing two

```

Network >
Create New
Network >
Transform >
Remove > all
Edges

Network >
Create New
Network >
Transform >
Arcs → Edges >
All

Network >
Create New
Network >
SubNetwork
with Path(s) >
All Shortest
Paths between
Two Vertices

```

parents and one child. The direct path from child to father is just as long as the indirect path via the child's mother because a marriage line counts as 0 distance. If we had ignored line values, the shortest paths would not have included the mothers (except for Ana Goce) in this example.

In Figure 98, it is easy to see that Petrus Gondola and his wife Anna Goce are the closest common ancestors of Paucho and Margarita. Of course, we could already see that in the original family tree (Figure 95), but we need the shortest paths command in large networks such as the genealogy of the entire Ragusan nobility, because this is too complicated to analyze by eyeballing.

Network>
Create
Partition>
k-Neighbours

The *ancestors (pedigree)* or *descendants* of a person are easily found with the *k-Neighbours* procedure in the Ore graph. Ancestors are connected by paths toward an individual, so they are its input neighbors. Descendants are reachable from the individual: They are output neighbors in the Ore graph. You may restrict the selection of ancestors to a limited number of generations in the *Maximum distance* dialog box of the *k-Neighbours* procedure. Note that the number of generations that you select is one more than the largest distance that you specify because the selected person, who also represents a generation, is placed in class 0. For example, the family tree in Figure 95 contains a number of output neighbors (descendants) of Petrus Gondola at maximum distance of 3.

Partition> Info

The Ore graph is most suited for finding brothers and sisters and for counting the size of sibling groups in a genealogical network. Pajek automatically creates a brothers/sisters partition, which identifies children of the same parental couple. Each class is a sibling group, except for class 0, so the number of vertices within a brothers and sisters class represents the size of a sibling group. Unfortunately, it is not easy to obtain a frequency distribution of the size of sibling groups from this partition in Pajek because the *Partition> Info* command lists each sibling group (class) separately.

It is possible, however, to obtain a frequency distribution of the size of sibling groups that have the same father or the same mother. In the Ore graph, the outdegree of a vertex is equal to the number of its children provided that marriage lines are disregarded. Ideally, every child has a father and a mother in the genealogical network, so we may count the number of children for each father or mother. In the case of a single marriage, the father and mother have the same number of children; but these numbers may differ in the case of remarriages. In the little example (Figure 96), my father remarried: He has three children (my stepsister, sister, and me), whereas my mother has only two children (my sister and me). Therefore, we must look at the outdegree of fathers or mothers, not at both.

Table 19. Number of children of Petrus Gondola and his male descendants

Cluster	Freq	Freq%	CumFreq	CumFreq%	Representative
0	131	67.5258	131	67.5258	4
1	14	7.2165	145	74.7423	15
2	15	7.7320	160	82.4742	3
3	11	5.6701	171	88.1443	1
4	7	3.6082	178	91.7526	2
5	4	2.0619	182	93.8144	29
6	1	0.5155	183	94.3299	120
7	3	1.5464	186	95.8763	23
8	4	2.0619	190	97.9381	13
9	1	0.5155	191	98.4536	114
11	2	1.0309	193	99.4845	85
12	1	0.5155	194	100.0000	171
SUM	194	100.0000			

This is achieved in the following way. First, remove the marriage lines (*Network> Create New Network> Transform> Remove> all Edges*) from the Ore graph. Now, the outdegree of a vertex is equal to an actor's number of children, so create an outdegree partition with the *Network> Create Partition> Degree> Output* command and select it as the first partition. Next we need the gender partition that was generated when reading the GEDCOM file as an Ore graph. In this partition, men are in class 1, women in 2. Select this partition as the second partition and execute the command *Partitions> Extract SubPartition (Second from First)*. In the dialog box, choose the class identifying the gender that you want to select, and Pajek will create a new partition with the outdegree of the selected vertices (e.g., the men).

*Network>
Create New
Network>
Transform>
Remove> all
Edges*

*Network>
Create
Partition>
Degree>
Output*

*Partitions>
Extract
SubPartition
(Second from
First)*

Partition> Info

The *Partition> Info* command will produce the desired frequency tabulation (see Table 19). Among Petrus Gondola's descendants, one man had twelve children and the others had fewer. Two-thirds (67.5 percent) of the men did not have children. Note, however, that they include the youngest men of the genealogy, who may have had children who were not included in the data set.

From the parent-child and marriage relations in the Ore graph, several other types of family relation can be inferred. If, for example, we know that someone's child is a third person's parent, we know that the first person is a grandparent of the third person. We may create a network with arcs expressing grandparent relations if we want to analyze this type of family relation. With matrix multiplication, the grandparent and many other types of family ties can be created. Matrix multiplication is a standard operation in linear algebra, which requires two matrices and

produces a new matrix. We can conceptualize a network as a matrix (see Chapter 12), so we can apply this technique to networks.

Network>
Multiple
Relations
Network>
Extract
Relation(s) into
Separate
Networks
Networks>
Multiply
Networks
Network>
Multiple
Relations
Network>
Change
Relation
Number –
Label
Network>
Create New
Network>
Transform>
1-Mode to
2-Mode
Macro> *Play*

The grandparent relation can easily be derived from an Ore graph. First, select the parent relation from the Ore graph (*Network> Multiple Relations Network> Extract Relation(s) into Separate Networks*). Second, multiply the parent's network by itself: select the parent's network in the first and second Network drop-down menus and issue the *Networks> Multiply Networks* command. The new network will contain the parents of parents of vertices: their grandparents. You may want to renumber and rename this relation with the *Network> Multiple Relations Network> Change Relation Number – Label* command (see earlier). Family relations networks are one-mode networks. It is also possible to multiply two-mode networks and one-mode with two-mode networks provided that the vertices of the one-mode network constitute one of the modes in the two-mode network and the one-mode network is changed into a two-mode network with the *Network> Create New Network> Transform> 1-Mode to 2-Mode* command.

In a similar way, many types of family relations can be established. Sometimes gender selections must be added – for example, if you want to tell grandfathers apart from grandmothers. The subdirectory *Macro>Kinship* in the directory where you installed Pajek contains macros for creating these networks. They require that you read the Ore graph with different relations for male and female links (see earlier). Use the *Macro> Play* command to select and execute these macros.

Exercise 1

From the genealogical data in *Gondola_Petrus.ged*, construct a network containing Petrus Gondola (born in 1356) and all his descendants who received the Gondola surname at birth. In other words, create a patrilineal genealogy for Petrus Gondola's offspring.

11.4 Social Research on Genealogies

Kinship is a fundamental social relation that is extensively studied by anthropologists and historians. In contrast to people who assemble their private family trees, social scientists are primarily interested in the genealogies of entire communities, such as the nobility of Ragusa.

These genealogies, which are usually very large, enable the study of overall patterns of kinship ties that, for instance, reflect cultural norms for marriage: Who are allowed to marry? Property is handed over from one generation to the next along family lines, so marriages may serve to protect or enlarge the wealth of a family; family ties parallel economic exchange. Demographic data on birth, marriage, and death reflect

Table 20. *Size of sibling groups^a in 1200–1250 and 1300–1350*

Size of Sibling Group	1200–1250		1300–1350	
	Freq	Freq%	Freq	Freq%
0 (no children)	18	16.4	386	54.5
1	22	20.0	87	12.3
2	18	16.4	73	10.3
3	19	17.3	53	7.5
4	11	10.0	38	5.4
5	10	9.1	29	4.1
6–10	12	10.9	40	5.6
11–21	–	–	2	0.3
TOTAL (no. of sibling groups)	110	100	708	100

^a The number of children from one father.

economic and ecological conditions (e.g., a famine or deadly disease causes high mortality rates).

The number of marriages and the age of the marital couple and the size of sibling groups, nuclear families, or extended families are determined and compared across different societies or different periods. Differences are related to external conditions and internal systems of norms or rules.

Table 20 compares the number of children of Ragusan noblemen across two periods: men born in 1200–50 and 1300–50. Unfortunately, many birth dates are unknown, so we added the parents' children and the children's in-laws from the kinship network assuming that they belong to the same generation. In the Ore graph, the simple outdegree of a vertex specifies the number of children of a person. Table 20 summarizes the output degree frequencies. In the first half of the fourteenth century and in comparison to the previous century, a large proportion of the noblemen had no children. Perhaps fewer men got married because no new families were admitted to the nobility as of 1332. Conversely, some men may have died young as a consequence of the Black Death epidemic, which struck the town in 1348.

This type of research may use network analysis, but it can also be done by database counts, for instance, calculations on a GEDCOM genealogy database. A second type of research, however, is inherently relational and must use network analysis as a tool. It focuses on structural relinking between families and the economic, social, and cultural reasons or rules for structural relinking. *Structural relinking* refers to the phenomenon that families intermarry more than once in the course of time. Intermarriage or *endogamy* is an indicator of social cohesion within a genealogy. If families are linked by more kinship ties, they are more likely to act as a

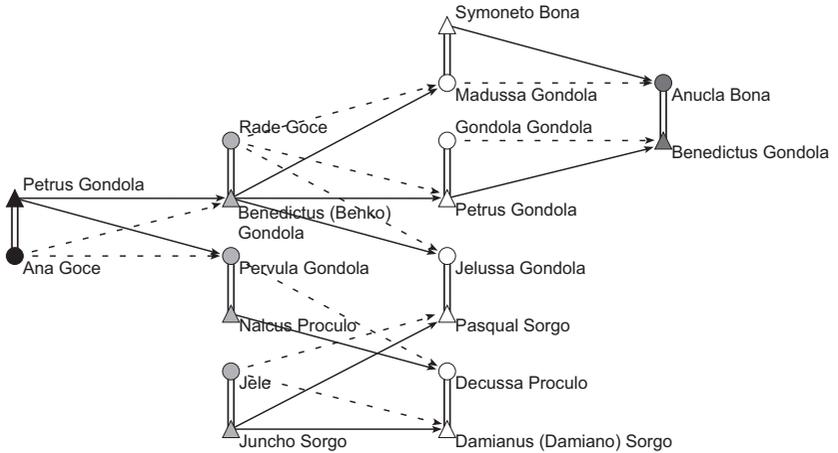


Figure 99. Structural relinking in an Ore graph.

clan: sharing cultural norms, entertaining tight relations, and restricting ties to families outside the clan.

There are two types of structural relinking: *blood marriages* and *non-blood relinking*. A blood marriage is the marriage of people with a close common ancestor, for instance, a marriage between a brother and sister or between a granddaughter and a grandson. The occurrence of this type of relinking tells us which types of intermarriages are culturally allowed and which are not. In the Ragusan nobility, a grandson of Benko Gondola (Benedictus Gondola) married a granddaughter (Anucla Bona), who was a fourth-degree relative (see Figure 99). Blood marriages between closer relatives – a son who married a daughter, a child who married a grandchild – did not occur among the Ragusan nobility. Apparently, these marriages were not allowed.

Nonblood relinking refers to multiple marriages between families without a close common ancestor. This type of relinking often serves economic goals, namely, to keep the wealth and power within selected families. Figure 99 shows nonblood interlinking between the Gondola and Sorgo families: two granddaughters of Petrus Gondola and Ana Goce (Jelussa and Decussa) marry brothers from the Sorgo family (Pasqual and Damianus), who were acknowledged to be the most influential family among the Ragusan nobility.

Structural relinking produces semicycles within a genealogical network; for instance, the blood marriage between Benedictus Gondola and Anucla Bona closes the paths from Benko Gondola to his granddaughter Anucla and his grandson Benedictus (Figure 99). The nonblood relinking between the Gondola and Sorgo families also yields a semicycle

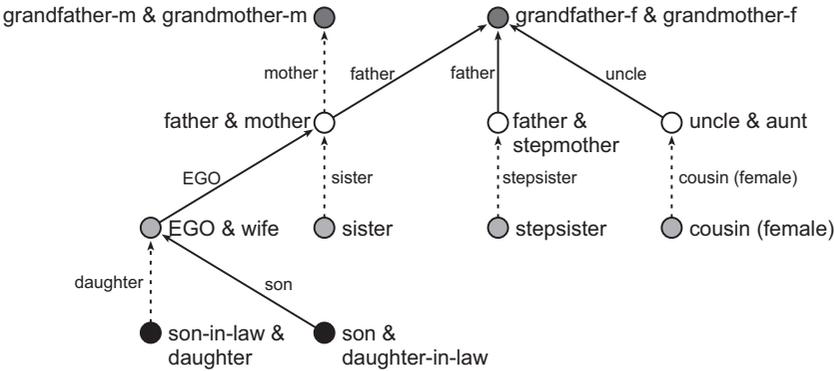


Figure 100. P-graph.

(Petrus Gondola–Benko–Jelussa Gondola–Pasqual Sorgo–Jele–Damianus Sorgo–Decussa Proculo–Pervula Gondola–Petrus Gondola, among other semicycles).

However, in the Ore graph not all semicycles represent structural relinking. A father, mother, and child also create a semicycle (e.g., Ana Goce–Petrus Gondola–Pervula Gondola in Figure 99). In addition, parents and two or more children create larger semicycles (e.g., Ana Goce–Pervula Gondola Petrus Gondola–Benko Gondola–Ana Goce). Remarriages yield even more complicated semicycles that do not point to structural relinking.

Because it is troublesome to distinguish between semicycles that represent structural relinking and semicycles that do not, a special kind of genealogical network was developed: the *parentage graph* or *P-graph*. In the P-graph, couples and unmarried individuals are the vertices and arcs point from children to parents. The type of arc shows whether the descendant is male (full arc) or female (dotted arc). In Figure 100, for instance, my son and his wife are connected by a full arc to me and my spouse; my daughter and her husband are connected by a dotted arc.

The P-graph has several advantages. It contains fewer vertices, but the path distance in a symmetrized P-graph still shows the remove of a relation, although it is not possible to exclude marital ties from the calculation. The main advantage of the P-graph, however, is the fact that it is acyclic – there are no edges between married people – and there are no separate arcs from mother and father to child. As a result, every semicycle and bi-component indicates structural relinking, which is either a blood marriage or another type of relinking. Figure 101 shows the P-graph associated with the Ore graph of Figure 99. The two semicycles represent structural relinking: the blood marriage of Benedictus Gondola

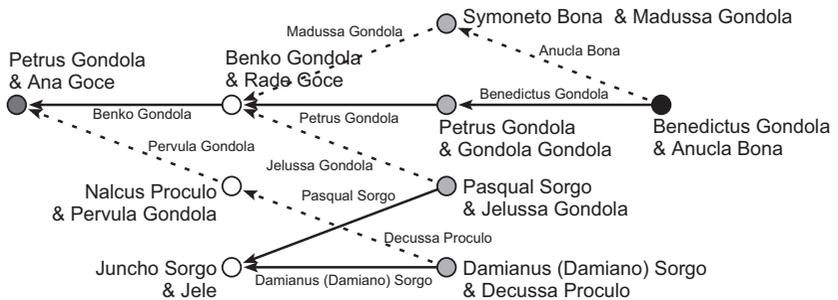


Figure 101. Structural relinking in a P-graph.

and Anucla Bona and the nonblood relinking between the Gondola and Sorgo families.

Apart from specific cases of relinking, social network analysts are interested in the amount of relinking in a genealogy. In a P-graph, this is measured by the *relinking index*. To understand this index, we must introduce the concept of a tree in graph theory: a connected graph that contains no semicycles. A tree has several interesting properties, but for our purposes the fact that it does not contain cycles and semicycles is most important.

A *tree* is a connected graph that contains no semicycles.

In a P-graph, every semicycle indicates structural relinking because the people or couples on the semicycle are linked by (at least) two chains of family ties (e.g., common grandparents on the father's side and on the mother's side). As a consequence, a P-graph that is a tree or a set of distinct trees (a *forest*) has no relinking, and its relinking index is 0. Given the number of people and the assumption that a marriage links exactly one man and one woman, the maximum amount of relinking within the P-graph of a genealogy can be computed so the actual number of relinking can be expressed as a proportion of this maximum. This is the relinking index, which is 1 in a genealogy with maximum relinking and 0 in a genealogy without relinking.

We advise calculating the relinking index on bi-components within the P-graph rather than on the entire P-graph. Genealogies have no natural borders; kinship ties extend beyond the boundaries of the data collected by the researcher, but boundary setting is important to the result of the relinking index. The largest bi-component within a genealogy is a sensible boundary because it demarcates families that are integrated into a system by at least one instance of relinking. In general, structural relinking may be used to bound the field of study, which means that you

limit your analyses to the families within the largest bi-component of a genealogy.

Let us calculate the amount of structural relinking among the Ragusan nobility in the period 1200–1350, in which new families were admitted to the nobility, and 1350–1500, when the nobility was chartered and no new families were admitted. Because we lack birth dates, we add the parents' children and children's in-laws to the couples in which at least one spouse is known to be born in the selected period. Between 1200 and 1350, a small number of the couples (137 of 1412 vertices, or 9.7 percent) were connected by two or more family ties, so the relinking index is low for the network in this period (0.02). Within this bi-component, the relinking index is higher (0.24), so there is a small core of families, the Sorgo family among them, who are tightly related by intermarriages.

In the period 1350–1500, the bi-component is larger, containing 476 couples (23.7 percent) and featuring many members of the Goce, Bodacia, and Sorgo families. The relinking index of the entire network is 0.20, and within the bi-component the proportion of relinking is 0.69. Both values are much larger than in the period before 1350, which shows increased endogamy among the Ragusan nobility.

In the P-graph, each person is represented by one arc except in the case of multiple marriages: remarriages and polygamy. Because each marriage is a separate vertex (e.g., my father and mother or my father and stepmother in Figure 100), men and women who remarry are represented by two or more arcs. In the P-graph, it is impossible to distinguish between a married uncle and a remarriage of a father or between stepsisters and (female) cousins. This problem is solved in the bipartite P-graph, which has vertices for individuals and vertices for married couples. However, the bipartite P-graph has the drawback of containing considerably more vertices and lines than the P-graph, and path distance does not correspond to the remove of a kinship relation. We do not use bipartite P-graphs in this book.

Application

The format of a genealogy that is read from a GEDCOM data file depends on the options checked in the *Options> Read – Write* menu. As noted, Pajek transforms a GEDCOM data file into an Ore graph if the option *GEDCOM – Pgraph* is *not* checked. A regular P-graph is created if this option is checked but the option *Bipartite Pgraph* is not. If the option *Pgraph + labels* is also checked, the name of a person is used as the label of an arc. All P-graphs have line value and relation number 1 for male lines and value 2 for female lines.

Pajek does not create a brothers and sisters partition in conjunction with a P-graph because siblings can easily be identified as the input neighbors (remember: arcs point from children to parents!) of a vertex

```
Options> Read
- Write>
GEDCOM –
Pgraph
```

```
Options> Read
- Write>
Bipartite
Pgraph
```

```
Options> Read
- Write>
Pgraph + labels
```

representing a married couple or an unmarried mother or father. It stores the years of birth of men and women in separate vectors because a couple has two birth dates. This also applies to the years of death. In addition, Pajek lists the year of marriage (999999998 for unmarried individuals), the family of spouse number (FAMS) for each couple, the family of child number (FAMC), and the sequential number (INDI) for the men and women separately.

We advise opening the entire Ragusan nobility genealogy (Ragusan.ged) as a P-graph (check option *GEDCOM-Pgraph* in the *Options> Read – Write* submenu) and making sure that names are used as labels of the arcs (also check the option *Pgraph + labels*). Note that reading the arc labels takes more time and uses more computer memory, so you may want to omit them if your network is very large and you do not really need the labels.

[Draw] As can be seen in Figure 101, the labels of vertices can be very long in P-graphs. This can make the layout difficult to read. Instead of showing all labels, you can show labels of selected vertices by selecting the option [Draw] *Options> Mark Vertices Using> Mark Cluster Only*. The same result can be obtained by selecting the checkbox at the right of the *Cluster* drop-down menu in the Main window following by clicking the drawing pencil button at the right of the *Network* drop-down menu. While this option is in effect, the vertices listed in the current cluster (if any), are labeled in the Draw screen. So create a cluster from a partition (see Section 8.3) or create an empty cluster (command *Cluster> Create Empty Cluster*) and manually edit the cluster so it contains the vertices for which labels should be visible. Another possibility is to display labels in several lines: insert $\backslash n$ at the position where the newline should occur. For example, change the label “Petrus Gondola & Ana Goce” to “Petrus Gondola $\backslash n$ Ana Goce.” Vertex labels can be changed if you manually edit a partition (*File>Partition> View/Edit*) that belongs to the network.

The relinking index is calculated by the *Network> Acyclic Network> Info* command, and it is printed in the Report screen. Note that the index is valid only for P-graphs. On request, Pajek will compute it for any acyclic network, but then its value is meaningless. In the P-graph with the entire Ragusan nobility, the relinking index is 0.23.

If you want to calculate the relinking index for the largest bi-component in this P-graph, you have to identify the bi-components and extract the largest bi-component first. The *Network> Create New Network> with Bi-Connected Components stored as Relation Numbers* command, introduced in Chapter 7, identifies the bi-components. Make sure that the minimum size of a bi-component is set to 3 in the dialog box issued by this command. Recall that in Pajek bi-components are stored as relation numbers. Because relation numbers are already used in a P-graph to distinguish between male and female lines, store relation numbers in a new

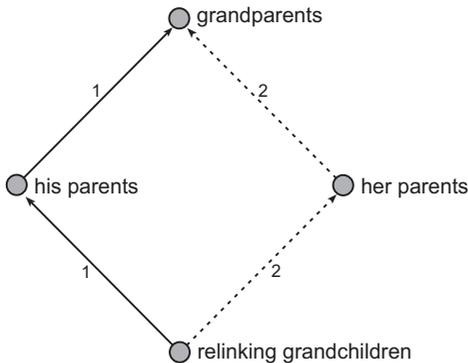


Figure 102. Fragment of relinking grandchildren.

network; do not overwrite old relation numbers because we may need them for other analyses. As you learned in Chapter 7, bi-components are stored also as a hierarchy, so inspect the hierarchy (*File* > *Hierarchy* > *View/Edit*) to find the sequential number and size of the largest bi-component. In the Ragusan nobility genealogy, we find two bi-components: the first contains 5 vertices and the second 1,446.

Extract the second bi-component from the network in the following way: Translate the required class of the hierarchy into a cluster with the *Hierarchy* > *Extract Cluster* command, specifying the sequential number of the bi-component in the hierarchy, and execute the *Extract SubNetwork* command from the *Operations* > *Network + Cluster* menu. Finally, calculate the relinking index with the *Network* > *Acyclic Network* > *Info* command. The relinking index is 0.74, which is quite high. If you would like to draw this bi-component in layers, remember that the arcs point from children to parents in a P-graph, so the oldest generations are drawn at the bottom of the Draw screen.

Particular types of relinking can be found with the *Fragment* commands in the *Networks* menu, which we also used to trace complete subnetworks (Chapter 3). Create a network that represents the relinking structure that you want to find (e.g., a marriage between two grandchildren of the same grandparents, see Figure 102), with the *Network* > *Create New Network* > *Empty Network* command and manual editing in the Draw screen. This fragment is also available in the file `relinking grandchildren.net`. Select this fragment as the first network, and select the P-graph of the Ragusan nobility genealogy as the second network. In the *Networks* > *Fragment (First in Second)* window, make sure that *Induced* is not checked because additional lines among the vertices in the fragment are allowed now. Finally, find the fragments with the *Find* command. Pajek encounters three instances of this fragment, among which is the marriage of the two grandchildren of Benko Gondola and Rade Goce.

Hierarchy >
Extract Cluster

Operations >
Network +
Cluster >
Extract
SubNetwork

Network >
Acyclic
Network > *Info*

Networks >
Fragment (First
in Second) >
Induced

Networks >
Fragment (First
in Second) >
Find

Networks>
Fragment (First
in Second)>
Check values of
lines

If you want to find a fragment with a particular pattern of male and female lines, make sure that the lines have the right values in the fragment (1 for male and 2 for female; the female lines do not have to be dotted) and select the *Check values of lines* option in the *Networks> Fragment (First in Second)* window. The same result can be obtained by matching the relation number (select the option *Check relation numbers* in the *Networks> Fragment (First in Second)* window). Recall that a line receives the same value as its relation number and line value when reading a GEDCOM file in Pajek, so you can use line values or relation numbers as a criterion for finding fragments (but do not forget to define line values and/or relation numbers also in the fragment.) In the Ragusan network, there are only two instances of a marriage among grandchildren of the same grandparents where the grandson is a descendant along patrilineal lines and the granddaughter descended along matrilineal lines as in the fragment of Figure 102.

Networks>
Fragment (First
in Second)>
Check relation
numbers

When you want to restrict your analysis to a particular birth cohort, you need a network with a selection of the genealogical data. Because the vertices of a P-graph may represent couples, you have to take into account the years of birth of the men and women, which are stored in separate vectors. You may decide either that both husband and wife must be born in the selected period or that at least one of them must be in that period. We should note, however, that vertices may also represent unmarried individuals, in which case husband or wife is irrelevant. In addition, missing birth dates, which are to be expected in historical data, may cause problems if you demand that both husband and spouse are known to be born in the selected period. Given these complexities, we advise to select the right period in your genealogical database software, produce a separate GEDCOM data file, and have Pajek translate it into a P-graph. Then, skip the remainder of this section.

Vector> Make
Partition> by
Intervals>
Selected
Thresholds

If this is not possible, however, you may extract the subnetwork in Pajek by combining information from different vectors. First, translate the vectors with birth dates of men and women to partitions with the *Vector> Make Partition> by Intervals> Selected Thresholds* command. In the dialog box, enter the limits of the required period (e.g., 1349 and 1500 if you are interested in the people born in 1350 up to and including 1500). Note that each threshold is included as the upper limit of the interval. In addition, include the threshold 99999997 to obtain a separate class with the 99999998 code, which represents either unknown or irrelevant birth dates (e.g., the male birth date in the case of an unmarried woman). Separate the thresholds by a blank.

Partition> Info

If we execute the command, Pajek creates a partition with four classes. If we inspect the partition with male birth dates (*Partition> Info*), we see that 1,025 men were born before 1350, 1,493 were born between 1350 and 1500, and 46 were born after 1500, and we have no information on

Table 21. *Birth cohorts among men and women*

Rows: 10. From Vector 1 [1349 1500 999999997] (4376)					
Columns: 11. From Vector 2 [1349 1500 999999997] (4376)					
Crosstabs					
	1	2	3	4	Total
1	51	1	0	973	1025
2	3	83	0	1407	1493
3	0	0	0	46	46
4	268	317	19	1208	1812
TOTAL	322	401	19	3634	4376

1,812 couples or individuals. The partition with female birth dates shows that 401 women are known to have been born between 1350 and 1500.

The four classes in the men and women partitions yield sixteen combinations, which are listed in Table 21. This table is part of the output produced by the *Partitions> Info> Cramer's V, Rajski, Adjusted Rand Index* command after selecting the male birth dates partition as the first partition and the female birth dates partition as the second. Note that the men are in the rows and the women in the columns and that the second class represents the period 1350–1500, whereas the fourth class contains the unknown and irrelevant birth dates.

In Table 21, the second row contains the men who were born between 1350 and 1500 (1,493 in total), and the second column shows the (401) women born in this period. Only 83 couples are known to consist of a husband and wife born in the selected period. In a majority of cases, we deal with unmarried men or unknown birth date of the wife (1,407 cases) and unmarried women or unknown birth date of the husband (317 cases). In very few cases, one spouse is known to be born in the right period, whereas the other is born in another period, namely, before 1350 (period 1): In one case, the husband was born before 1350, and in three cases the wife was born before 1350.

It seems reasonable to select all vertices in which either the man or the woman was born in the right period. This can be done if we create a new partition identifying the vertices for which the male birth date and/or the female birth date is coded as class 2. First we have to binarize the two birth dates partitions such that the period 1350–1500 (class 2 in these partitions) becomes class 1 in the new partitions, whereas all other classes become 0. Simply execute the *Partition> Binarize Partition* command on each of the birth dates partitions and select class 2 in the dialog box. Do this for both partitions: male and female birth dates.

Then select the two binarized partitions as first and second partition and sum them (*Partitions> Add (First + Second)*). The resulting partition

*Partitions>
Info> Cramer's
V, Rajski,
Adjusted Rand
Index*

*Partition>
Binarize
Partition*

*Partitions>
Add (First +
Second)*

has three classes: class 0 containing (2,565) individuals or couples without known birth between 1350 and 1500, class 1 containing (1,728) individuals and couples containing a husband or wife born in this period, and class 2 with (83) couples with both spouses known to be born between 1350 and 1500. Now we can extract the desired subnetwork from the Ragusan nobility genealogy by executing the *Operations> Network + Partition> Extract> SubNetwork Induced by Union of Selected Clusters* command, selecting clusters from one to two in the dialog box. This subnetwork contains 1,811 vertices.

Macro> Play In the Ragusan nobility genealogy, many birth dates are missing. Assuming that all children of the same parents and all parents and in-laws of children belong approximately to the same birth cohort, we may add them to the people we know were born in the required period. We need these indirect neighbors to preserve the structure of the genealogical network. The procedure is stored in the macro *expand_generation.mcr*, which can be executed with the *Macro> Play* command. A genealogical network (Ore graph or P-graph) must be selected in the *Network* drop-down menu and a binary partition identifying the selected birth cohort must be selected in the *Partition* drop-down menu. Note that the partition that we used to extract the birth cohort is not binary, because it contains classes 0, 1, and 2. We must first binarize it such that all selected couples and individuals are in class 1. Execute the *Partition> Binarize Partition* command and select classes 1 and 2 in the dialog box if you want to expand this birth cohort. The macro creates a new partition with the extended birth cohort in class 1: in our example 2,007 bachelors and couples.

Operations> Vector + Partition> Extract SubVector The macro can be executed several times to increase the number of selected vertices, but *generation jumps* may extend the range of birth dates enormously. We advise applying the macro only once and checking the range of known birth years among the selected vertices afterward. To this end, extract the vertices selected in the expanded partition from the year of birth vector(s): Make sure the expanded birth cohort partition is selected in the *Partition* drop-down menu and a year of birth vector is selected in the *Vector* drop-down menu and execute the *Operations> Vector + Partition> Extract SubVector* command (select class 1 only). You may inspect the extracted years with the *Vector> Info* command, which reports the lowest and highest values: There should not be years that fall widely outside the selected period. In the case of a P-graph, you must check the birth dates of men and women separately. With the men, the known birth dates range from 1280, which is seventy years before the selected period, to 1500. The women were born between 1298 and 1498. Even in its first step, the expansion macro lengthens the range of birth dates considerably.

Exercise II

What kind of structural relinking does the small bi-component in the Ragusan nobility genealogy represent: a blood marriage or nonblood relinking? Extract this bi-component from the network, and draw it to find the answer to this question.

11.5 Example II: Citations among Papers on Network Centrality

In several social domains, genealogical terminology is used as a metaphor for nonbiological affinity. Artists who were trained by the same master or who are influenced by the same predecessors are considered to belong to the same family or tradition. A work of art has a pedigree: a list of former owners. In a similar way, scientists are classified according to their intellectual pedigree: the theories and theorists they use as a frame of reference in their work.

In science, citations make explicit this frame of reference, so they are a valuable source of data for the study of scientific development and scientific communities in scientometrics, history, and the sociology of science. They reveal the impact of articles and their authors on later scientific work, and they signal scientific communities or specialties that share knowledge.

In this chapter, we analyze the citations among articles that discuss the topic of network centrality. In 1979, Linton Freeman published an article that defined several kinds of centrality. His typology has become the standard for network analysis, so we used it in Chapter 6 of this book. Freeman, however, was not the first to publish on centrality in networks. His article is part of a discussion that dates back to the 1940s. The network depicted in Figure 103, (centrality_literature.net) shows the articles that discuss network centrality and their cross-references until 1979. Arcs represent citations; they point from the cited article to the citing article.

In principle, articles can cite only articles that appeared earlier, so the network is *acyclic*. Arcs never point back to older articles just as parents cannot be younger than their children. However, there are usually some exceptions in a citation network: articles that cite one another (e.g., articles appearing at about the same time and written by one author). We eliminate these exceptions by removing arcs that are going against time or by shrinking the articles by an author that are connected by cyclic citations. In the centrality literature network, we used the latter approach (e.g., two publications by Gilch denoted by #GilchSW-54 in Figure 103).

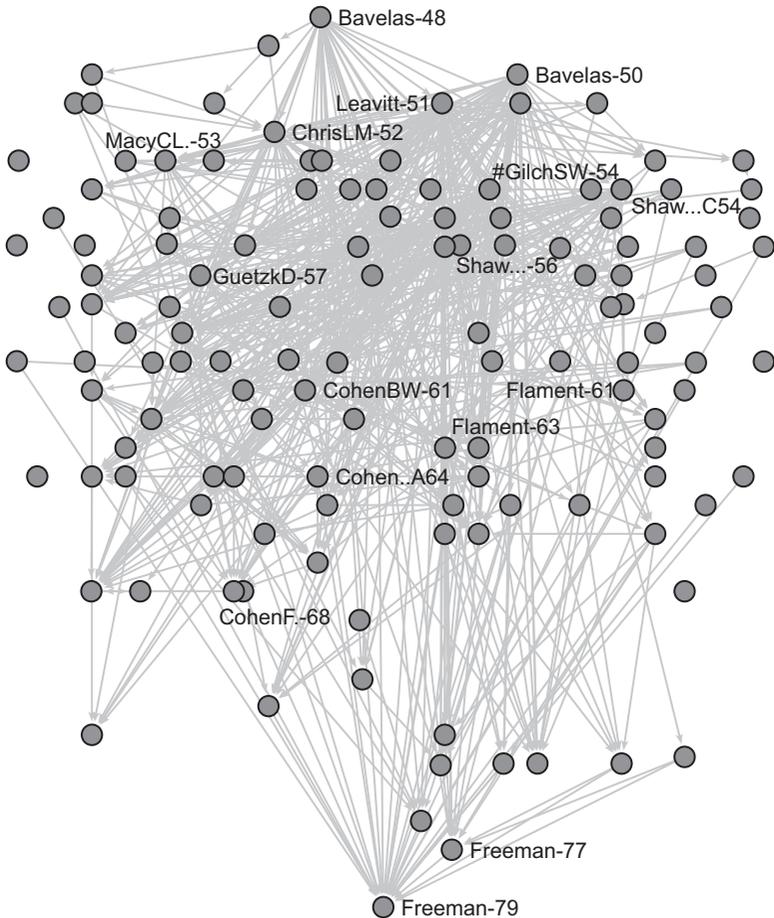


Figure 103. Centrality literature network in layers according to year of publication.

There are important differences between genealogical data and citation data. A citation network contains one relation, whereas genealogical data concern two relations: parenthood and marriage. In addition, an article may cite all previous articles notwithstanding their distance in time. In a genealogical network, children have two (biological) parents and parenthood ties always link two successive generations. The concept of a generation is not very useful in the context of a citation network, so we order the articles by publication date. In Figure 103, layers represent the year of publication (the `centrality_literature_year.clu` partition), which is also indicated by the last two digits in the label of a vertex.

11.6 Citations

Nowadays, citations are being used to assess the scientific importance of papers, authors, and journals. In general, an item receiving more citations is deemed more important. Bibliographic databases, for instance, the *Web of Science* compiled by the Institute for Scientific Information (ISI), list the citations in a large number of journals. Simple calculations yield indices of scientific standing, for instance, the *impact factor* of a journal (the average number of citations to papers in this journal) and the *immediacy index* (the average number of citations of the papers in a journal during the year of its publication). In each year, journals are ranked by their scores on these indices. Compared over longer periods, these indices show differences between scientific disciplines. In the liberal arts, for instance, it is rare for authors to cite recent publications, whereas this is very common in the natural sciences.

Citation analysis is not exclusively interested in the assessment of scientific standing. It also focuses on the identification of specialties, the evolution of research traditions, and changing paradigms. Researchers operating within a particular subject area or scientific specialty tend to cite each other and common precursors. Citation analysis reveals such cohesive subgroups, and it studies their institutional or paradigmatic background. Scientific knowledge is assumed to increment over time: Previous knowledge is used and expanded in new research projects. Articles that introduce important new insights are cited until new results modify or contradict them. Citation analysis, therefore, may spot the articles that influence the research for some time and link them into a research tradition that is the backbone of a specialty. Scientific revolutions, that is, sudden paradigmatic changes resulting from new insights, are reflected by abrupt changes in the citation network.

Network analysis is the preferred technique for extracting specialties and research traditions from citations. Basically, specialties are cohesive subgroups in the citation network, so they can be detected with the usual techniques. Weak components identify isolated scientific communities that are not aware of each other or who see no substantial overlap between their research domains. Within a weak component, a bi-component identifies sections where different lines of citations emanating from a common source text meet again. This is similar to the concept of relinking in genealogical research.

In most citation networks, however, these criteria are not strong enough because almost all articles are linked into one bi-component. k -Cores (Chapter 3) offer a more penetrating view. The centrality literature network, for example, contains one large weak component and eleven isolates. There is one large bi-component, and twelve vertices are connected by one citation. The network contains a 10-core of twenty-nine papers

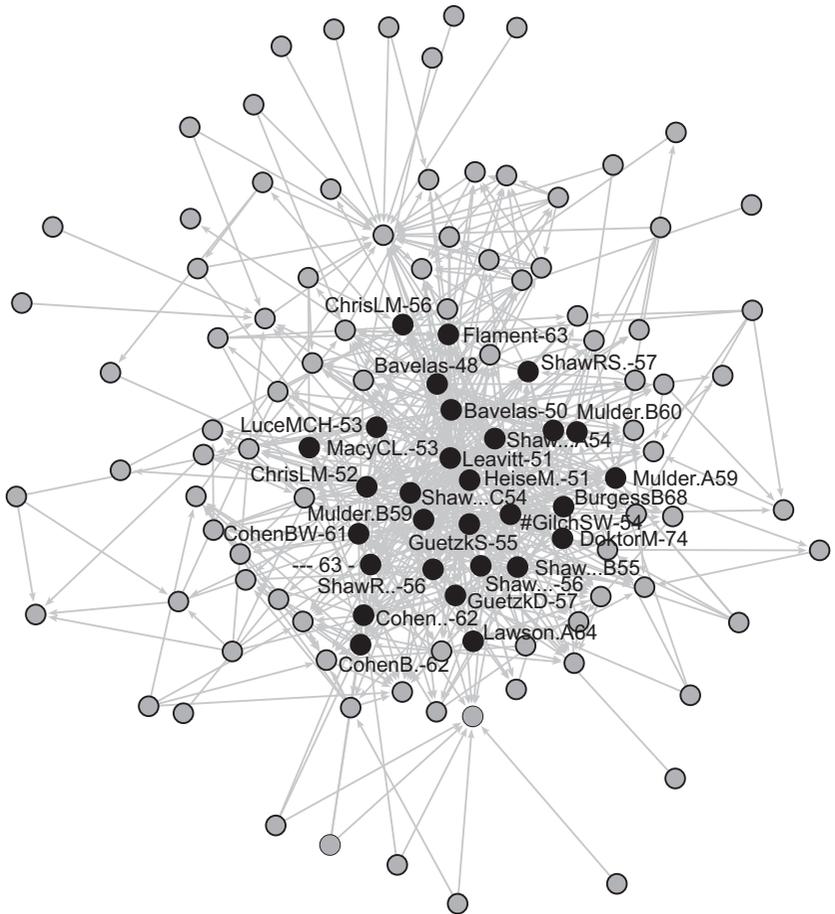


Figure 104. k -Cores in the centrality literature network (without isolates).

that is the central summit of this network (the black vertices in Figure 104). Each of the articles in this core is connected to at least ten other articles by citations, but we do not know which are cited often and cite others often.

The cohesion concept (as discussed in Chapters 3–5) does not take time into account. It does not reflect the incremental development of knowledge, nor does it identify the articles that were vital to this development. Therefore, a special technique for citation analysis was developed that explicitly focuses on the flow of time. It is called *main path analysis*.

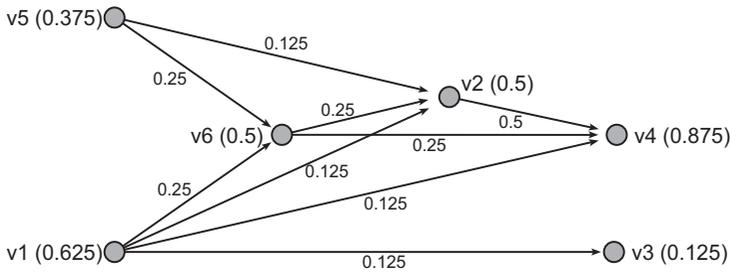


Figure 105. Traversal weights in a citation network.

Let us think of a citation network as a system of channels that transport scientific knowledge or information. An article that integrates information from several previous articles and adds substantial new knowledge receives many citations, and it will make citations to previous articles more or less redundant. As a consequence, it is an important junction of channels and a great deal of knowledge flows through it. If knowledge flows through citations, a citation that is needed in paths between many articles is more crucial than a citation that is hardly needed for linking articles. The most important citations constitute one or more main paths, which are likely to be the backbones of a research tradition.

Main path analysis calculates the extent to which a particular citation or article is needed for linking articles, which is called the traversal count or traversal weight of a citation or article. First, the procedure counts all paths from each source (an article that is not citing within the data set) to each sink (an article that is not cited within the data set), and it counts the number of paths that include a particular citation. Next, it divides the number of paths that use a citation by the total number of paths between source and sink vertices in the network. This proportion is the traversal weight of a citation. In a similar way, you can obtain the traversal weight of each article.

In an acyclic network, a *source vertex* is a vertex with 0 indegree.

In an acyclic network, a *sink vertex* is a vertex with 0 outdegree.

The *traversal weight* of an arc or vertex is the proportion of all paths between source and sink vertices that contain this arc or vertex.

For example, Figure 105 shows a citation network of six articles ordered in time from left to right. There are two sources (v1 and v5) and two sinks (v3 and v4). One path connects source v1 and sink v3, but

there is no path from v_5 to v_3 . Four paths reach v_4 from v_1 and three paths from v_5 . In sum, there are eight paths from sources to sinks. The citation of article v_1 by article v_3 is included in one of the eight paths, so its traversal weight is 0.125. The citation of v_2 in article v_4 is contained in exactly half of all paths. The traversal weights of the vertices, which are reported between brackets, are calculated in a similar way.

Now that we have defined and calculated the traversal weights of citations, we may extract the paths or components with the highest traversal counts on the lines, the main paths or main path components, which are hypothesized to identify the main stream of a literature. We can analyze their evolution over time and search for patterns that reflect the integration, fragmentation, or specialization of a scientific community.

In a citation network, a *main path* is the path from a source vertex to a sink vertex with the highest traversal weights on its arcs. Several methods have been proposed to extract main paths from the network of traversal weights. The method we will explain first is called *forward local main path search*. It consists of choosing the source vertex (or vertices) incident with the arc(s) with the highest weight, selecting the arc(s) and the head(s) of the arc(s), and repeating this step until a sink vertex is reached. In the example of Figure 105, the main paths start with vertex v_1 and vertex v_5 because both source vertices are incident with an arc carrying a traversal weight of 0.25. Both arcs point toward vertex v_6 , which is the next vertex on the main paths. Then, the paths proceed either to vertex v_2 and on to vertex v_4 or directly from vertex v_6 to vertex v_4 . We find several main paths, but they lead to the same sink, so we conclude that the network represents one research tradition.

Instead of starting with one or more source vertices, we may start with one or more sink vertices incident with the arc(s) with the highest weight and travel against the direction of the arcs. Thus, we get the *backward local main path search*. In the example of Figure 105, we find the backward local main paths starting with sink vertex v_4 , which is incident to an arc with traversal weight 0.5, because the other sink vertex (v_3) is linked to an arc with a much lower traversal weight (only 0.125). From vertex v_4 we proceed backwards to vertex v_2 , from there to vertex v_6 , finally reaching both vertex v_1 and vertex v_5 .

In *key-route local main path search*, we select a limited number of arcs, for example, 10. These are usually the arcs with the highest traversal weights. Selected arcs are called *key-routes*. Note that key-routes do not need to be arcs attached to source or sink vertices. They can be located anywhere in the acyclic network, so this is a main difference with forward and backward local searches. For each key-route, we find the main path from source vertices to the key-route and from the key-route to sink

vertices. We search forward from the terminal vertex of the key-route until a sink vertex is reached. In each step we select the arc(s) with the highest traversal weight(s). Then we search backward from the initial vertex of the key-route until a source vertex is reached. The resulting *Key-route local main path* consists of the main paths obtained for all key-routes. If we run the key-route local main path search with the highest arc as the only key-route for the example network of Figure 105, we get the same result as with forward or backward local main path search.

The three methods defined so far are called *local main path methods* because we search only locally for the current arc in each step; that is, we just check arcs that are incident with the current vertex and have the right direction. In local main path searches, we may relax the rule that we select only the arc(s) with the highest value in each step. If we also accept arcs with weights slightly below the highest value, we introduce *tolerance* in the search. If tolerance is set to 0, we select only arcs with the highest value. But if tolerance is set to some positive value, for example, 0.1, and the highest arc value is 0.5, all arcs with values larger than 0.4 are selected. It is usually a good idea to start with zero tolerance to avoid finding very extensive (“broad”) main paths.

In contrast to local search methods, *global main path methods* search for paths with the overall highest sum of traversal weights. Allowing for tolerance here is not a good idea because the search can become computationally demanding. We present two global methods: *Standard global main path search* and *Key-route global main path search*.

Standard global main path is the path from source to sink vertices with the overall highest sum of traversal weights on the path. This method is widely used across scientific disciplines. In project planning, for example, it is called *Critical Path Method (CPM)*. CPM is an algorithm for scheduling a set of project activities. For the example in Figure 105, *standard global main path* yields the same result as the one obtained using forward or backward local main path search. The overall highest sum of traversal weights is 1.

In *Key-route global main path* search, we again start with some arcs as *key-routes*. For each key-route we search for the main path that contains the key-route from source to sink vertices with the overall highest traversal weight. The *Key-route global main path* unites the main paths obtained for all key-routes. In the example of Figure 105, key-route global main path search with the highest arc as the only key-route gives us the same result as forward and backward local main path search because the arc from vertex v2 to v4 is both the key-route (traversal weight is 0.5) and part of the forward and backward main paths.

Usually some arcs (citations) with the highest traversal weights are selected as key-routes. But this is not necessary. In citation network

analysis, it may make sense to select one or more papers (not citations) that are of particular interest, for example some papers that you wrote yourself, and search for the main path containing these papers. For more details on main path searches check the references in Further Reading.

A *main path component* is extracted in the following way. Choose a cutoff value between 0 and 1, and remove all arcs from the network with traversal weights below this value. The components in the extracted networks are called main path components. Usually, we look for the lowest cutoff value that yields a component that connects at least one source vertex to one sink vertex. This value is equal to the lowest traversal weight on the main paths. In our example, this cutoff value is 0.25, and we obtain a main path component that includes all articles except v3, which is a marginal article in the research tradition represented by this data set.

Of course, article v3 may be very important in another research tradition. The choice of the articles to be included in the data set restricts the number and size of research traditions that can be found. Like a genealogy, a citation network is virtually endless, so it cannot be captured entirely in a research project. The researcher has to set limits to the data collection, but this should be based on sound substantive arguments.

Citation networks are usually created from bibliographic databases such as the *Web of Science*. The bibliographic data stored in these databases also allow the creation of other types of networks: coauthorship networks, bibliographic coupling networks, cocitation networks, keyword networks, and so on. See the Further Reading section for references and for a link to software that transforms downloads from bibliographic databases into Pajek networks.

Application

In Chapters 3 and 7, we discussed the commands for detecting components, bi-components, and k -cores, which identify cohesive subgroups in a network. In principle, a citation network is directed and acyclic, so you should search weak components instead of strong components and find k -cores on input and output ties (command *All* in the *Network> Create Partition> k-Core> All* submenu).

Network>
Create
Partition>
k-Core> All

Main path analysis is very easy in Pajek. The commands in the *Network> Acyclic Network> Create Weighted Network + Vector> Traversal Weights* submenu compute the traversal weights for lines and vertices in an acyclic network. There are three commands: *Search Path Count (SPC)*, *Search Path Link Count (SPLC)*, and *Search Path Node Pair (SPNP)*. The *Search Path Count (SPC)* command counts the paths between all source and sink vertices as explained previously. The *Search Path Link Count (SPLC)* command traces paths from all vertices to the sink vertices. In the latter procedure, citations of early articles receive lower weights because they cannot be part of paths emanating from later

Network>
Acyclic
Network>
Create
Weighted
Network +
Vector>
Traversal
Weights

Table 22. *Traversal weights in the centrality literature network*

Line Values	Frequency	Freq%	CumFreq	CumFreq%
(... 0.0000]	90	14.68	90	14.68
(0.0000... 0.0515]	465	75.86	555	90.54
(0.0515... 0.1030]	45	7.34	600	97.88
(0.1030... 0.1545]	8	1.31	608	99.18
(0.1545... 0.2059]	2	0.33	610	99.51
(0.2059... 0.2574]	2	0.33	612	99.84
(0.2574... 0.3089]	0	0.00	612	99.84
(0.3089... 0.3604]	0	0.00	612	99.84
(0.3604... 0.4118]	1	0.16	613	100.00
TOTAL	613	100.00		

articles, so we advise to use it only in special cases where early articles are relatively unimportant. In the *Search Path Node Pair (SPNP)* command, each vertex is considered as a source and as a sink. As a result, vertices and edges in the middle will receive higher traversal weights.

There are several ways of normalizing the traversal weights of lines and vertices in a citation network. Previously we discussed the normalization according to flow (*Network> Acyclic Network> Create Weighted Network + Vector> Traversal Weights> Normalization of Weights> Normalize-Flow*) for the *Search Path Count* method: the number of paths that include a line or vertex divided by the total number of paths between sources and sinks. This normalization yields the percentage of all paths between sinks and sources that include a vertex or line, and it is the recommended normalization. Other options include dividing the number of paths containing a vertex or line by the maximum found among the vertices or lines (option *Normalize-Max*), which is useful when all traversal weights according to flow are low, and taking the logarithm of the number of paths containing a vertex or line (option *Logarithmic Weights*), which is useful when the variation among traversal weights is very high. Finally, it is possible not to normalize the raw counts (option *Without Normalization*). Note, however, that normalization does not affect the main paths that are later retrieved from the citation network with traversal weights computed. It merely changes the range and variation among traversal weights.

The traversal weights of the papers (the original vertices) are stored in a vector, and the weights of the citations (lines) are saved as line values in a new network (labeled “Citation weights”), which can be inspected with the *Network> Info> Line Values* command.

When we apply the *Search Path Count (SPC)* command to the centrality literature network, about 90 percent of the lines have a traversal weight of 0.05 or less and thirteen lines have a value exceeding 0.103 (Table 22:

*Network>
Acyclic
Network>
Create
Create
Weighted
Network +
Vector>
Traversal
Weights>
Normalization
of Weights*

*Network>
Info> Line
Values*

be sure the network labeled “Citation weights (SPC)” is selected in the drop-down menu when you execute the *Network> Info> Line Values* command and request #9 clusters). Clearly, one citation is very important to the development of the centrality literature: It has an extremely high traversal weight of 0.41. This is the citation of Bavelas’ 1948 article by Leavitt in 1951. Bavelas (1948) and Leavitt (1951), as well as Freeman (1979) and Flament (1963), are the vertices with the highest traversal weights. These are the crucial articles in the centrality literature.

Vector> Info

Sometimes we want to know which articles (not citations) are the most important in the citation network. As we have already mentioned traversal weights of articles are stored as a Vector. Be sure that Vector labelled “Citation weights (SPC)” is selected in the drop-down menu when you execute the command *Vector> Info*. Input 10 in the first dialog box to get 10 most important articles according to traversal weights. In the centrality literature network the result is not surprising: the most important article is the article by Bavelas (1948) with traversal weight 0.71, following by Freeman (1979) with traversal weight (0.56), Flament (1963) with traversal weight 0.46, and Leavitt (1951) with traversal weight 0.41. All other articles have traversal weights lower than 0.40.

The *Traversal Weights* commands do not automatically identify the main paths in the citation network. After computing the traversal weights, we must apply commands available in *Network> Acyclic Network> Create (Sub)Network> Main Paths* to obtain the main paths. Note that a network with traversal weights must be selected before searching for main paths.

Network>
Acyclic
Network>
Create
(Sub)Network>
Main Paths>
Local Search>
Forward

We find the forward local main path with the *Network> Acyclic Network> Create (Sub)Network> Main Paths> Local Search> Forward* command. A dialog box allows us to set the tolerance, which we usually leave at 0. The command creates a partition identifying the vertices on the main paths (cluster 1) in the original citation network, and it produces a new network labeled “Forward Local Main Path” that contains the main paths. In the centrality literature, the main paths start with Bavelas (1948), proceed to Leavitt (1951), and end with Freeman (1977 and 1979); see the top main path in Figure 106. As an exercise, find the local forward main path with nonzero tolerance. How does the extracted main path differ from the one in Figure 106 (top)?

Network>
Acyclic
Network>
Create
(Sub)Network>
Main Paths>
Global Search>
Standard

With *Standard global main path search*, we get a slightly different result; see the main path in the middle of Figure 106. The main path still starts with Bavelas (1948), proceeds to Leavitt (1951), and HeisseM (1951), but then proceeds to ChrisLM (1952) instead of proceeding to Shaw...C (1954). If we follow the standard global main path further, we get four more articles that are not present in the forward local main path (MacyCL (1953) – LuceMCH (1953) – ChristiB(1954) – LanzetR

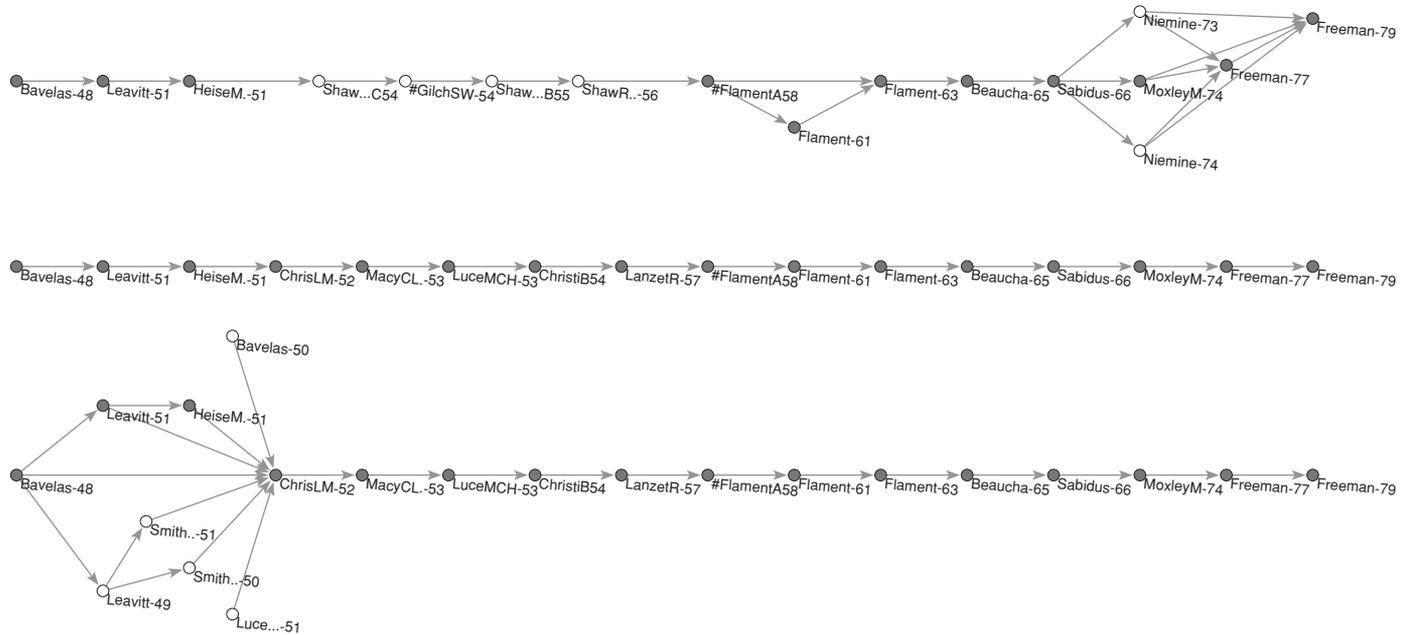


Figure 106. Forward local and key-route local (top), standard global and key-route global (middle), and backward local (bottom) main paths in the centrality literature network.

(1957)), but after that we reach the article FlamentA (1958), which is present also in the forward local main path. The rest of the standard global main path is the same as one single path that is part of the forward local main path, namely: Flament (1961) – Flament (1963) – Beaucha (1965) – Sabidusi (1966) – MaxleyM (1974) – Freeman (1977) – Freeman (1979).

Network>
Acyclic
Network>
Create
 (Sub)Network>
 Main Paths>
 Local Search>
 Key-Route

Note that both main paths obtained so far contain the citation with the highest traversal weight (Bavelas' 1948 article cited by Leavitt in 1951). The Bavelas 1948 article is also the source paper – it does not cite any other paper. As a consequence, key-route local main path search with this citation as the only key-route yields the same result as forward local main path search (the top main path in Figure 106). We select the Bavelas 1948 paper citation by entering 1 behind *Select rank numbers of key-routes* in the dialog box of the *Network> Acyclic Network> Create (Sub)Network> Main Paths> Local Search> Key-Route* command. This tells Pajek to use the arc (citation) with the highest traversal weight as the only key-route or, more accurately, to use the arc at rank 1 in the list of line values. We can select the ten arcs with highest traversal weights by entering 1–10 in the dialog box.

Network>
Info> General

Network>
Info> Line →
Rank of its
Value

If several vertices have maximum traversal weight, Pajek selects the one reported first in *Network> Info> General* (when entering some positive number in the dialog box). To be sure that we have the citation of Bavelas 1948 by Leavitt in 1951, we should check the rank of this arc in the network with traversal weights. We can find the rank of a particular arc with the *Network> Info> Line → Rank of its Value* command. Enter the initial and terminal vertex number or label and the Report window displays the rank of this line according to its line value. In this way, we can find the rank of any special citation that we would like to use as a key-route. Enter the arc's rank number in the dialog box of the *Key-Route* command.

Network>
Acyclic
Network>
Create
 (Sub)Network>
 Main Paths>
 Global Search>
 Key-Route

In this example, the key-route global main path search yields the same result as the standard global main path search if we select Leavitt's (1951) citation of the Bavelas 1948 article as the only key-route (the middle main path in Figure 106). Note that this need not be the case in other networks.

Network>
Acyclic
Network>
Create
 (Sub)Network>
 Main Paths>
 Local Search>
 Backward

Finally let us apply the last search method: backward local main path search. The main path obtained by the backward local main path search contains the standard global main path as a subnetwork (see gray vertices in the bottom main path in Figure 106).

If we compare the results of forward local and backward local main path searches, we notice that forward local search adds more articles and citations at the end of the path – when we approach the Freemans articles – while backward local search adds some articles and

citations at the beginning of the path, soon after the Bavelas (1948) paper. If we would use some nonzero tolerance in searching for local main paths even more articles would be added to the local main paths.

If we need to select only one solution, we would probably select the one that appears most often. The main path obtained by the standard global main path search is obtained also using key route global main path search, and it is a subnetwork of the backward local main path search (gray vertices in the bottom main path). Several articles and citations in the forward local main path (and key-route local main path) are also present in the standard global main path (gray vertices in the top main path). Therefore the standard global main path seems to be the most appropriate for this example. But this needs not to be a general rule. For larger citations networks, we suggest to use nonzero tolerance and more key-routes to get more main paths, which may represent different research traditions.

Instead of searching for main paths containing selected key-routes (these are arcs representing citations) we can also search for main paths containing selected articles (represented by vertices). This can be done easily: first select one or more interesting articles (vertices numbers) in a Cluster and run the last command in this menu called: *Through Vertices in Cluster*. Again we can search for local and global main paths. For an example let us find main path containing both Leavitt (1949) and #GilchSW (1954) articles. As we can see in Figure 106 these two articles do not belong together to any main path. First create an empty Cluster (command *Cluster> Create Empty Cluster*) and manually edit the cluster so it contains the vertices 2 and 21 (2 and 21 are vertices numbers of Leavitt and #GilchSW articles respectively). Finally run command *Through Vertices in Cluster (Local or Global)*. In case of global search the obtained main path contains 22 articles and in case of local search (zero tolerance) it contains 32 articles. Both main paths start in Bavelas (1948) article and finish in Freeman (1979) article what is again no surprise.

If *Mark Main Paths as Multirelational Network* is checked while searching main paths, the original network containing traversal weights is transformed into a multirelational network, where arcs belonging to the main path get relation number 2, while relation numbers for all other arcs are set to 1. A message in a Report window explains this. Be careful with this option: If the original network already contains multiple relations, these relations are overwritten.

The lowest traversal weight of the arcs in the main path is 0.05, but it is interesting to use a slightly lower cutoff value to obtain the main path component here. Let us delete all arcs with traversal weights lower

```
Network>
Acyclic
Network>
Create
(Sub)Network>
Main Paths>
[Global, Local]
Search>
Through
Vertices in
Cluster
```

```
Cluster>
Create Empty
Cluster
```

```
Network>
Acyclic
Network>
Create
(Sub)Network>
Main Paths>
Mark Main
Paths as
Multirelational
Network
```

```
Network>
Create New
Network>
Transform>
Remove> Lines
with Value>
lower than
```

Network> *Create Partition*> *Components*> *Weak* than 0.03. This can be done with the *Remove*> *Lines with Value*> *lower than* command in the *Network*> *Create New Network*> *Transform* sub-menu. Now, determine the weak components of minimum size 2 with the *Network*> *Create Partition*> *Components*> *Weak* command. The network contains two weak components, one large component with forty-six articles, a small component with three articles by Lawson and Burgess, and eighty isolated vertices.

Operations> *Network* + *Partition*> *Extract*> *SubNetwork Induced by Union of Selected Clusters* Let us concentrate on the largest component and extract it with the *Operations*> *Network* + *Partition*> *Extract*> *SubNetwork Induced by Union of Selected Clusters* command, using the citation network with lines of minimum value 0.03 and the partition according to weak components. If we also extract the publication years of the forty-six articles in this component from the publication years partition (*centrality_literature_year.clu*) – select this partition as the first partition and the weak components partition as the second partition, and extract the first weak component (cluster 1) with the *Partitions*> *Extract SubPartition (Second from First)* command – we can draw this component into layers.

Layers> *Averaging x Coordinate* The resulting sociogram may look like Figure 107 if we optimize it with the *Layers*> *Averaging x Coordinate* command (*Forward* or *Backward*). This figure reveals that the literature on network centrality was split into two lines between 1957 and 1979. One line was dominated by Cohen and the other by Flament and Nieminen. In 1979, Freeman integrated both lines in his classic article. If labels of vertices cannot be read because they overlap, put the Draw window into a FishEye mode (*FishEye*> *Cartesian, Polar*) as we have learned in Chapter 2.

Operations> *Network* + *Partition*> *Transform*> *Direction*> *Lower* → *Higher* Mutual references among articles appearing at approximately the same time (e.g., two 1954 articles by Gilch in the original centrality network) or erroneous references to later articles by mistakes during data collection and coding may prevent the citation network from being acyclic. Then, the *Traversal Weights* commands issue a warning and stop; the network must first be made acyclic. References to later publications can be removed with the *Operations*> *Network* + *Partition*> *Transform*> *Direction*> *Lower* → *Higher* command (do not delete lines within clusters) provided that the partition according to publication dates was selected in the *Partition* drop-down menu.

Network> *Create Partition*> *Components*> *Strong* In the centrality literature network, however, this solution did not work because both articles by Gilch appeared in 1954. In this case, we had to merge the articles. We computed the strong components of minimum size 2 (*Network*> *Create Partition*> *Components*> *Strong*) because they contain cyclically connected vertices in a directed network (see Chapter 10). We shrank each strong component to one vertex in a new network with the *Operations*> *Network* + *Partition*> *Shrink Network* command

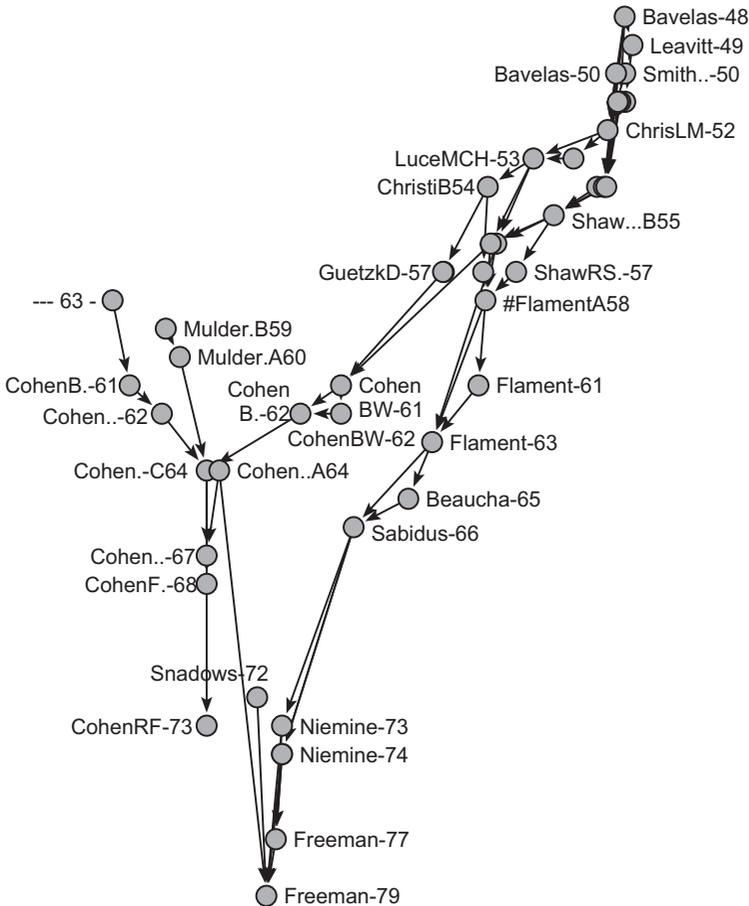


Figure 107. Main path component of the centrality literature network (not all names are shown here).

selecting 0 as the class that should not be shrunk because that class contained the vertices outside the strong components. We removed the loops with the *Network> Create New Network> Transform> Remove> Loops* command to obtain an acyclic network that allows the computation of citation weights.

In the latest Pajek versions, however, it is much easier to transform a nearly acyclic network into an acyclic one: Just apply the *Preprint Transformation* command, which is available in menu *Network> Acyclic Network> Transform> Preprint Transformation*.

```
Network>
Create New
Network>
Transform>
Remove>
Loops
```

```
Network>
Acyclic
Network>
Transform>
Preprint
Transformation
```

inside a strong component is duplicated into a “preprint” version. Vertices (papers) inside each strong component cite “preprints.”

11.7 Summary

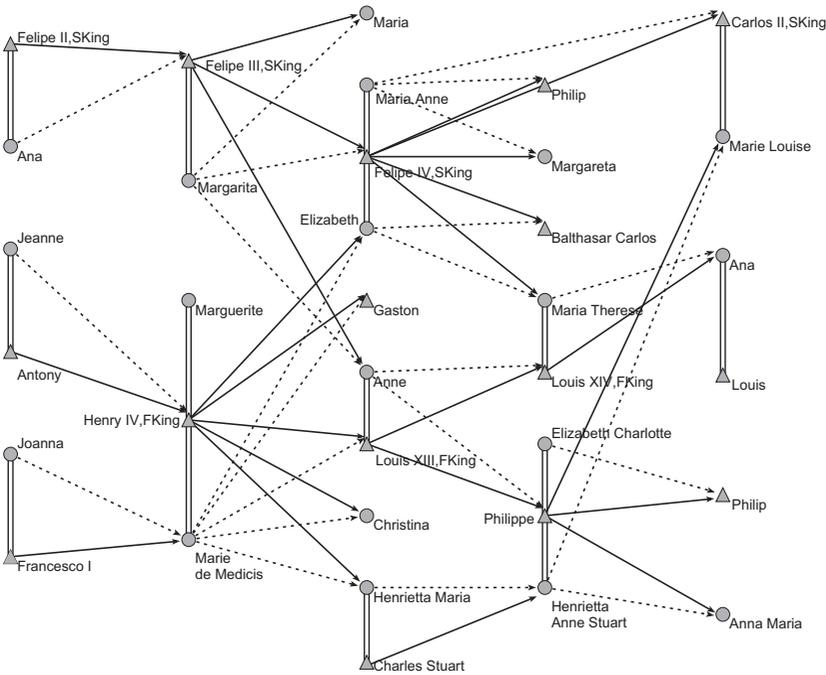
This was the last chapter that presented methods that cope with the dynamics of time in network analysis. Over time, social relations branch off into a gamut of independent strands. Kinship relations, for instance, create family trees that expand rapidly over generations. Sometimes, however, these strands merge after some time, for instance, people with common ancestors marry. This is called structural relinking, which is a measure of social cohesion over time. A social system with much relinking is relatively cohesive because relinking shows that people are oriented toward members of their own group or family.

In a genealogy, the amount of structural relinking can be assessed provided that we use a special kind of network: the P-graph. In contrast to an Ore graph, which represents each person by a vertex, parenthood by arcs, and marriage by (double) lines, couples and bachelors are vertices and individuals are arcs in a P-graph. Because symmetric marriages and parallel mother–child and father–child arcs are not represented by lines in the P-graph, each bi-component is an instance of structural relinking.

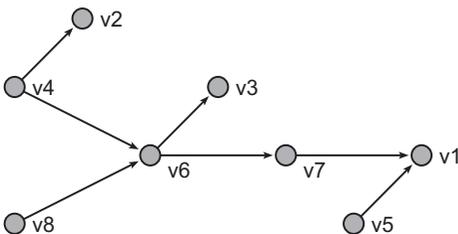
Methods for analyzing citation networks handle the time factor in a slightly different way. Here, we want to identify the publications that are the crucial links in the literature on a particular topic. Scientific articles contain knowledge, and citations indicate how knowledge flows through a scientific community. Each flow follows a path of citations, and citations that occur in many paths are important to the transmission of knowledge: They have high traversal weights. Citations with high traversal weights are linked into main paths, which represent the main lines of development in a research area. The articles and authors connected by citations of some minimum traversal weight constitute main path components, which are hypothesized to identify scientific specialties or subspecialties.

11.8 Questions

1. The Ore graph depicted below shows part of the family ties of Louis XIII, king of France (1601–43). Calculate the remove of his relation with Henrietta Anne Stuart.



2. Which people constitute the family of orientation of Louis XIII, and what is his family of procreation?
3. What is a generation jump? Indicate one in the Ore graph of Question 1.
4. Draw a P-graph that contains the same information as the Ore graph of Question 1.
5. How can we distinguish between a blood marriage and a relinking nonblood marriage in a P-graph? Give an example of both types of relinking in the genealogy of Louis XIII.
6. Explain why the relinking index of a tree is 0.
7. List all paths from sources to sinks in Figure 105, and show that the citation weight of the arc from v2 to v4 is correct.
8. Identify the source and sink vertices, the paths between them, and the traversal weight of the arcs in the following citation network below. What is the main path?



11.9 Assignment 1

The GEDCOM file `Isle_of_Man.ged` contains the combined genealogies of approximately twenty families from the British Isle of Man. Describe the overall structure of this network and the sections with structural relinking. Which types of relinking do occur?

11.10 Assignment 2

Publications and citations pass on scientific knowledge and traditions; so do advisors to their students. The file `PhD.net` contains the ties between Ph.D. students and their advisors in theoretical computer science; each arc points from an advisor to a student. The partition `PhD_year.clu` contains the (estimated) year in which the Ph.D. was obtained. Search for separate research traditions in this network and describe how they evolve.

11.11 Further Reading

- The genealogical data of the Ragusan nobility example were coded from the Ph.D. thesis of Irmgard Mahnken (1960): *Das Ragusanische Patriziat des XIV. Jahrhunderts*. For an analysis of a part of the genealogy, see V. Batagelj, “Ragusan families marriage networks.” In A. Ferligoj and A. Kramberger (eds.), *Developments in Data Analysis* (Ljubljana: FDV, 1996, pp. 217–28).
- For the collection and storage of genealogical data, we advise to use the GEDCOM 5.5 standard (<http://homepages.rootsweb.ancestry.com/~pmcbride/gedcom/55gctoc.htm>). Good free software is Brothers Keeper available at <http://www.bkwin.org>, and Personal Ancestral File, which is produced and distributed by the Church of Jesus Christ of Latter-Day Saints (www.familysearch.org). This organization compiles a large database of genealogical information from which downloads can be made.
- The P-graph was presented by D. R. White and P. Jorion in “Representing and analyzing kinship: A network approach.” *Current Anthropology* 33 (1992), 454–62; and in “Kinship networks and discrete structure theory: Applications and implications.” *Social Networks* 18 (1996), 267–314.
- For additional reading on the analysis of kinship relations in the social sciences, we refer to T. Schweizer and D. R. White, *Kinship, Networks, and Exchange* (Cambridge: Cambridge University Press, 1998).

- The centrality literature example was taken from N. P. Hummon, P. Doreian, and L. C. Freeman, “Analyzing the structure of the centrality-productivity literature created between 1948 and 1979.” *Knowledge-Creation Diffusion Utilization* 11 (1990), 459–80. The different types of main path analysis stem from N. P. Hummon and P. Doreian, “Connectivity in a citation network: The development of DNA theory.” *Social Networks* 11(1989), 39–63. E. Garfield’s *Citation Indexing: Its Theory and Application in Science, Technology, and Humanities* (New York, NY: John Wiley & Sons, 1979) is a classic text on citation analysis.
- Details on Key-Route main path searches are explained in J. S. Liu and L. Y. Y. Lu, “An integrated approach for main path analysis: Development of the Hirsch index as an example.” *Journal of the American Society for Information Science and Technology*, 63 (2012), 528–42. See also Wikipedia: https://en.wikipedia.org/wiki/Main_path_analysis
- The analysis of networks that can be derived from bibliographic data are discussed in V. Batagelj and M. Cerinšek, “On bibliographic networks.” *Scientometrics*, 96 (2013) 3, 845–64 and in Chapters 3 and 4 of V. Batagelj, P. Doreian, A. Ferligoj, and N. Kejžar, *Understanding Large Temporal Networks and Spatial Networks: Exploration, Pattern Searching, Visualization and Network Evolution* (New York, NY: John Wiley & Sons, 2014). Software for transforming bibliographic data into Pajek networks is available at <https://github.com/bavla/biblio>.

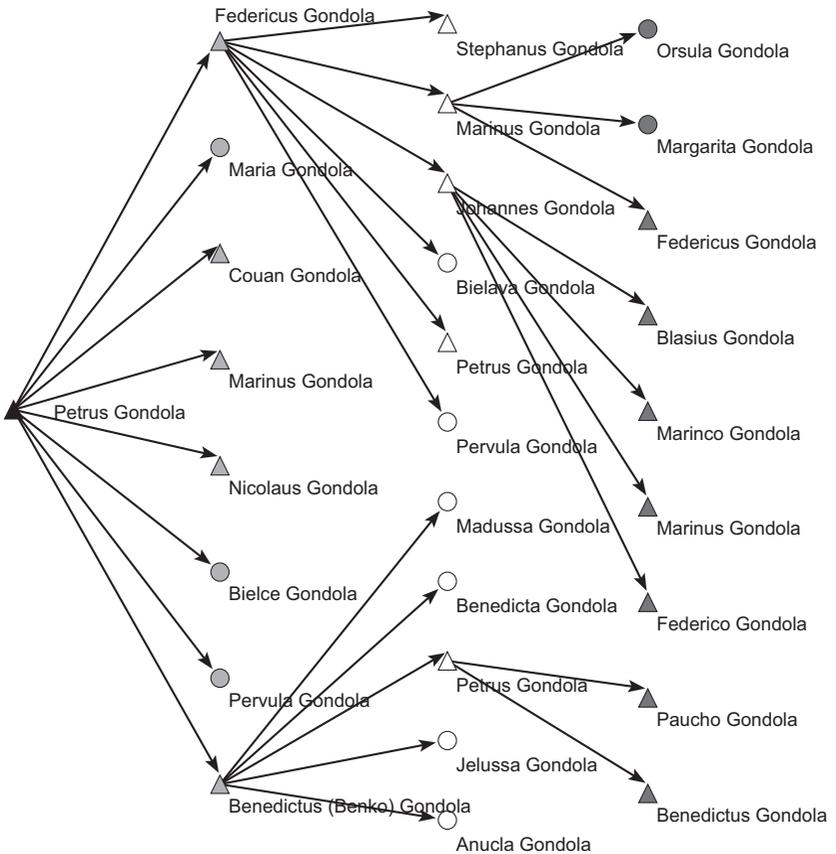
11.12 Answers

Answers to the Exercises

- I. You should realize that a surname was passed on from father to child in Ragusa. Mother–child ties and marriages do not matter (we are concerned only with the name given at birth). Therefore, you should eliminate all marriages and mother–child ties from the `Gondola_Petrus.ged` data. This can be done easily if you open the GEDCOM file with the option `Options> Read – Write> Ore: Different relations for male and female links` selected. In this network, you must select the father–child relation (`Network> Multiple Relations Network> Extract Relation(s)` into Separate Networks, enter relation number 1).

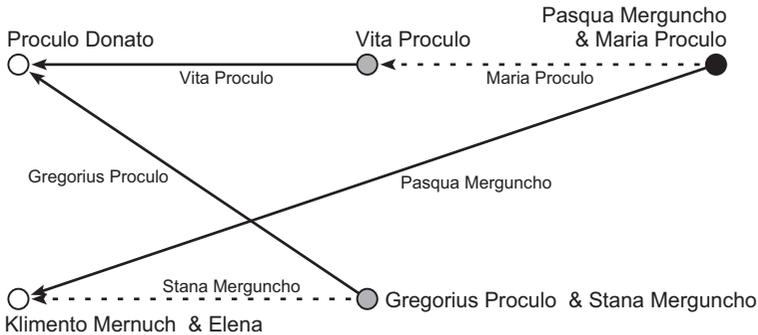
In the resulting network, the descendants of Petrus Gondola are all people who received his surname. Identify them with the `k-Neighbours> Output` command (Petrus Gondola has a vertex number of 94) and extract them from the network with the `Operations> Network + Partition> Extract> SubNetwork Induced by Union of`

Selected Clusters command (in the dialog box enter 0–3), making sure that the *k-Neighbours* partition is selected in the *Partition* drop-down menu. If you determine the genealogical depth partition of the new network (*Network > Acyclic Network > Create Partition > Depth Partition > Genealogical*), draw it in layers (*Layers > In y Direction*), optimize it (*Layers > Optimize Layers in x Direction > Forward*), and rotate it 90 degrees (*[Draw] Options > Transform > Rotate 2D*), it should look like the sociogram that follows.



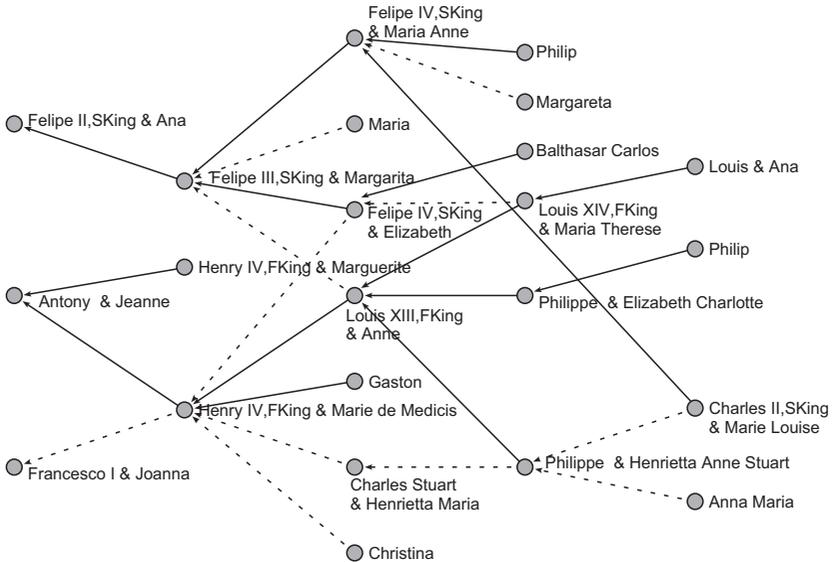
- II. In the “Application” part of Section 11.4, you learned how to determine the bi-components in a P-graph (command: *Network > Create New Network > with Bi-Connected Components stored as Relation Numbers*), how to create a cluster from a bi-component in the hierarchy of bi-components (*Hierarchy > Extract Cluster*), and how to extract this component from the original P-graph (*Operations > Network + Cluster > Extract SubNetwork*). In this way, you can obtain

the subnetwork of the small bi-component in the Ragusan nobility genealogy consisting of five vertices. With the vertices relocated, this network may look like the sociogram that follows. From this layout, it is clear that the marriages closing the semicycle are not blood marriages. Neither Pasqua Merguncho nor Maria Procuro has a close common ancestor, nor have Gregorius Procuro and Stana Merguncho. This structural relinking is an instance of repeated marriages between two families: The Procuro and Mernuch/Merguncho families swap a son.



Answers to the Questions in Section 11.8

1. Louis XIII is the uncle (mother's brother) of Henrietta Anne Stuart, so she is a relative in the third degree if we restrict ourselves to blood relations. Louis XIII is also her stepfather, so the degree is 1 if we include marital ties.
2. The family of orientation of Louis XIII includes his parents Henry IV and Marie de Medicis, his brother Gaston, and his sisters Elizabeth, Christina, and Henrietta Maria. Marguerite, the other wife of Henry IV, may or may not belong to the family of orientation. His family of procreation contains his wife Anne and their children Louis XIV and Philippe.
3. A generation jump in a genealogy refers to a relinking marriage that connects people of different genealogical generations, which are calculated from the point of view of their common ancestor. The marriage between Carlos II and Marie Louise creates a generation jump, because Carlos is a grandson of Felipe III and Margarita (second remove) and Marie Louise is the granddaughter of the daughter (Anne) of Felipe III and Margarita (third remove).
4. The P-graph should look like the following figure. Do not forget to draw different arcs for men and women and to reverse the direction of arcs.



- In a P-graph, the husband and wife involved in a blood marriage share at least one ancestor: There are two paths from the blood marriage to an ancestor, for instance, from Philippe and Henrietta Anne Stuart to Henry IV, king of France, and his spouse Marie de Medicis. Both Philippe and Henrietta Anne Stuart are their grandchildren. A relinking nonblood marriage is a marriage between descendents of families that are already linked by intermarriage; for example, the Spanish king Felipe III and the French king Henry IV are linked by two marriages among their children: Felipe IV and Elizabeth, Louis, XIII, and Anne. In a P-graph, this type of relinking is characterized by two semipaths (or one path and one semipath) between couples.
- Structural relinking involves semicycles: Vertices are connected by two paths or semipaths. Because trees contain no semicycles by definition, there is no relinking and the relinking index is 0.
- The eight paths are as follows: (1) $v_1 \rightarrow v_3$, (2) $v_1 \rightarrow v_4$, (3) $v_1 \rightarrow v_2 \rightarrow v_4$, (4) $v_1 \rightarrow v_6 \rightarrow v_4$, (5) $v_1 \rightarrow v_6 \rightarrow v_2 \rightarrow v_4$, (6) $v_5 \rightarrow v_6 \rightarrow v_4$, (7) $v_5 \rightarrow v_6 \rightarrow v_2 \rightarrow v_4$, and (8) $v_5 \rightarrow v_2 \rightarrow v_4$. Four paths include the arc $v_2 \rightarrow v_4$ (viz., paths 3, 5, 7, and 8), which is half of all paths, so the traversal weight of this arc is 0.5.
- The source vertices are v_4 , v_8 , and v_5 ; v_2 , v_3 , and v_1 are sink vertices. There are six paths from sources to sinks as follows: (1) $v_4 \rightarrow v_2$, (2) $v_4 \rightarrow v_6 \rightarrow v_3$, (3) $v_4 \rightarrow v_6 \rightarrow v_7 \rightarrow v_1$, (4) $v_8 \rightarrow v_6 \rightarrow v_3$, (5) $v_8 \rightarrow v_6 \rightarrow v_7 \rightarrow v_1$, and (6) $v_5 \rightarrow v_1$. The arcs $v_4 \rightarrow v_2$ and $v_5 \rightarrow v_1$ are

included in one of these paths, so their traversal weight is 1 divided by 6 as follows: 0.167. The other arcs are included in two paths, so their traversal weights are 0.333. There are four main paths: (1) from v4 to v3, (2) from v4 to v1, (3) from v8 to v3, and (4) from v8 to v1.

Part V

Modeling

In the final part, we shift our attention from purely exploratory description of network structure to modeling. Two types of modeling are presented: blockmodels (Chapter 12) and random graph models (Chapter 13).

Cohesion, brokerage, and ranking are connected to social roles: being a member of a group, being a mediator, or being a superior. Each of these roles is associated with a particular pattern of ties. A blockmodel describes the social roles and associated patterns of ties in the network at large. Blockmodels offer a different perspective on the concepts discussed in previous chapters.

Could the structure of our network be random? Network exploration yields values for structural properties of networks, such as cohesion, centrality, and ranking. We assume that the structural properties result from choices made by the people represented in the network – whom to relate to, whom to avoid – and that structural properties are consequential to their attitudes and behavior. The social causes and consequences of network structure give meaning to structural indices. Random graph models show us how confident we can be that network properties and effects result from social rather than random processes.

Blockmodels

12.1 Introduction

In previous parts of this book, we have presented a wide range of techniques for analyzing social networks. We have discovered that one structural concept can often be measured in several ways (e.g., centrality). We have not encountered the reverse, that is, a single technique that is able to detect different kinds of structures (e.g., cohesion and centrality). In this final chapter, we present such a technique, which is called blockmodeling.

Blockmodeling is a flexible method for analyzing social networks. Several network concepts are sensitive to exceptions; for instance, a single arc may turn a ranking into a rankless cluster (Chapter 10). Empirical data are seldom perfect, so we need a tool for checking the structural features of a social network that allows for exceptions or error. Blockmodeling and hierarchical clustering, which are closely related, are such tools.

Although blockmodeling is a technique capable of detecting cohesion, core-periphery structures, and ranking, it does not replace the techniques presented in previous chapters. At present, blockmodeling is feasible and effective only for small dense networks, whereas the other techniques work better on large or sparse networks. In addition, blockmodeling is grounded on different structural concepts: equivalence and positions, which are related to the theoretical concepts of social role and role sets. Blockmodels group vertices into clusters and determine the relations between these clusters (e.g., one cluster is the center and another is the periphery). In contrast, the techniques discussed in previous chapters, such as the measures of centrality, compute the structural position of each vertex individually.

Blockmodeling uses matrices as computational tools and for the visualization of results. Therefore, we introduce the matrix as a means for representing social networks before we will proceed to the concept of equivalence and the technique of blockmodeling.

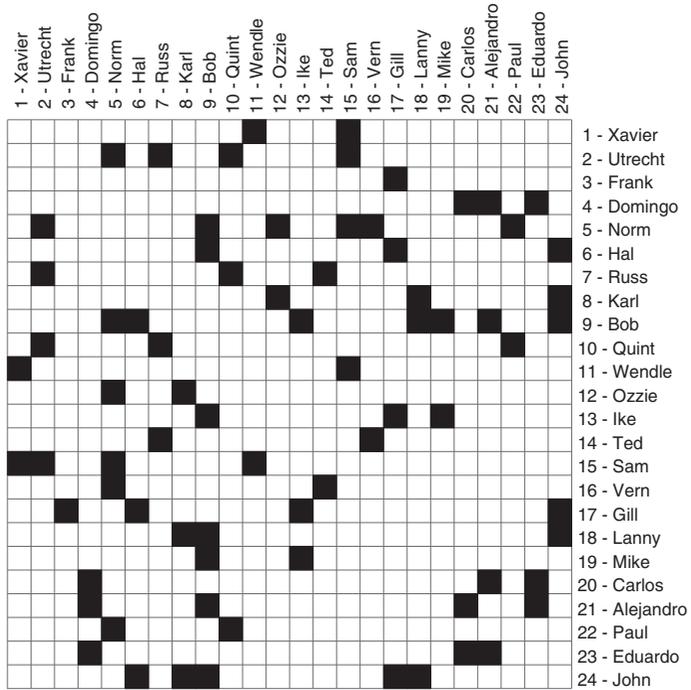


Figure 108. Communication lines among striking employees.

12.2 Matrices and Permutation

In social network analysis, matrices have been used in addition to sociograms for a long time. A matrix is an efficient tool for representing a small social network and for computing results on its structure. In addition, matrices offer visual clues on the structure of small and dense networks, which is what we use them for in the present section.

A matrix is a two-way table containing *rows* and *columns*. The intersection of a row and a column is called a *cell* of the matrix. Figure 108 displays the matrix of the communication network of striking employees in a wood-processing facility (see Chapter 7). In this matrix, each row and column represent one vertex of the network, for instance, the first (highest) row and the first (left) column feature Xavier. The cells in this row or column show Xavier’s ties. In Figure 108, a black cell indicates that Xavier communicates with another employee (or with himself), and a white (empty) cell means that there is no communication. Note that a matrix usually contains numbers, for instance, 1 for the presence of a tie and 0 if there is no tie. In Figure 108, we replaced the numbers with black or white squares to highlight the pattern of communication ties.

This type of matrix is called an *adjacency matrix* because we can tell from it which vertices are neighbors (adjacent) in the network; for instance, the black cells in the first row mean that Xavier (vertex 1) communicates with Wendle (vertex 11) and Sam (vertex 15). To be more precise, these black cells indicate that there is a tie *from* Xavier *to* Wendle and Sam. The row entry contains the sender of the tie and the column entry its recipient, so the first row contains ties from Xavier and the first column shows the ties *to* Xavier (e.g., from Wendle and Sam). It is not a coincidence that Xavier has the same neighbors in his row and column: The network is undirected, so Xavier's communication with Wendle implies that Wendle communicates with Xavier, and so on. An edge is equivalent to a bidirectional arc, so an edge is represented by two arcs in an adjacency matrix. In general, the adjacency matrix of an undirected network is symmetric around the diagonal running from the top left to the bottom right of the matrix, which is usually referred to as the diagonal of the matrix.

The adjacency matrix of Figure 108 contains no black cells on the diagonal because these cells represent the relation of a vertex to itself and the employees were not considered to communicate with themselves. Cells on the diagonal of an adjacency matrix often receive special treatment because they feature loops.

Because the same vertices define the rows and columns of an adjacency matrix, the adjacency matrix is square by definition. In contrast, a two-mode network such as the network of multiple directors in Scotland (Chapter 5) is represented by a matrix that is rectangular but not necessarily square. We can place the firms in the rows and the directors in the columns and still include all ties in the cells of this matrix because firms can only be directly related to directors. Such a matrix is called an *affiliation matrix*. In an affiliation matrix, diagonal cells do not represent loops.

The pattern of black cells in a matrix offers visual clues on the structure of the network because we see which lines are present (black) or absent (white). Just like a sociogram, however, a matrix discloses network structure only if its vertices are carefully placed. Figure 108, for instance, shows a seemingly random pattern of black cells. It does not reveal the structure of the network because the employees are listed in an arbitrary order. If we order them by their language (English or Spanish) and age (less than or more than thirty years), the black cells display a much more regular pattern (Figure 109). Now, it is easy to see that lines occur predominantly within the ethnic and age groups: No more than three lines (Karl–Ozzie, Bob–Norm, and Bob–Alejandro) exist between the groups.

A reordering or sorting of vertices is called a permutation of the network. Essentially, a permutation is a list with an entry for each vertex in

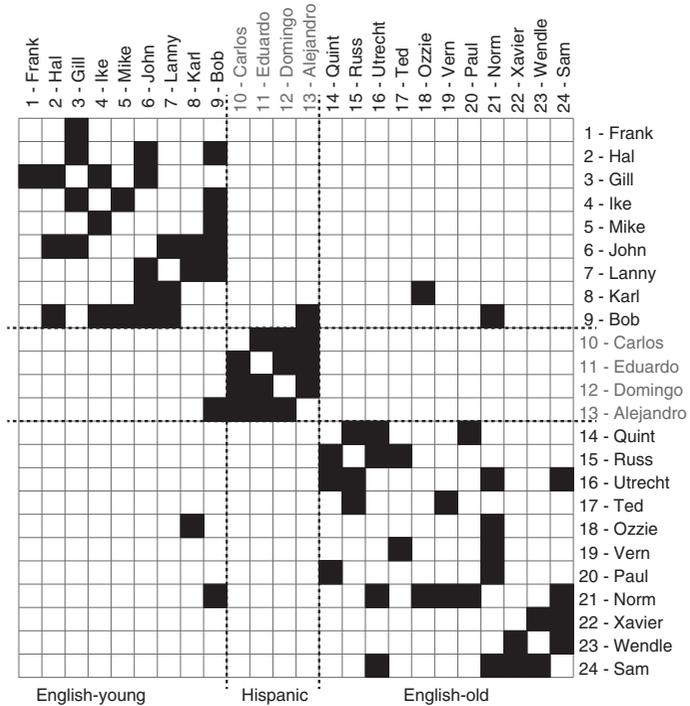


Figure 109. The matrix of the strike network sorted by ethnic and age group.

the network, specifying its new vertex number. In other words, a permutation is a renumbering of the vertices in the network.

A *permutation* of a network is a renumbering of its vertices.

If we assign new numbers to the vertices, the structure of the network does not change. Compare, for example, networks A and B in Figure 110. We exchanged the numbers of vertices 2 and 4, but *this does not affect the structure of the network*: Networks A and B are *isomorphic*; that is, they have the same structure. In the matrix, we exchanged vertex numbers in the rows *and* the columns, and we reordered the matrix obtaining a different matrix for the same structure.

The matrices look different, but they describe the same structure. This means that we can represent the same network by a number of different matrices, just as we can draw many different sociograms for one network. A permutation rearranges a matrix just like an energy command redraws

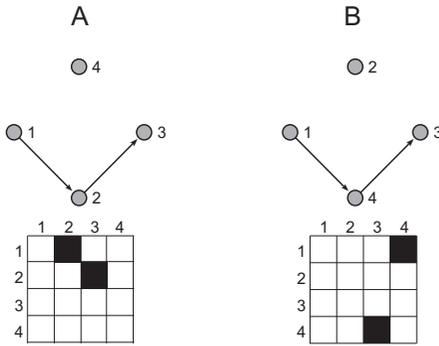


Figure 110. A network and a permutation.

a sociogram. Therefore, we can use permutations to find matrices that reveal the structure of a network. Subsequent sections show how to do this.

The strike network permuted by ethnic and age groups (Figure 109) shows the pattern that characterizes cohesive subgroups: The black (nonempty) entries cluster around the diagonal of the matrix, where they form clumps. The clumps identify subgroups of actors who maintain ties predominantly within their groups. In our example, the clumps nicely reflect the ethnic and age groups.

Application

Because a matrix can represent a network, it is possible to store network data in matrix format. For small networks, a matrix is a traditional and useful alternative to the lists of arcs and edges that we have used so far as our format for network data files. Pajek can read data in matrix format; for details, see Appendix 1 (Sections A1.2 and A1.3), which also discusses some disadvantages of the matrix format.

In Pajek, you can display the matrix of a network by double-clicking its name in the *Network* drop-down menu. In the dialog box that appears, enter a 1 if you want to display a binary matrix, that is, a matrix that tells only whether a line is present (#) or absent (.). Figure 111 shows part of the original strike network (included in the Pajek project file `strike.paj`) displayed as a binary matrix. Note that the listing consists of raw unformatted text, so it should be displayed in a fixed wide type font such as Courier. If you want to display the line values in the adjacency matrix, type a 2 in the dialog box to obtain a valued matrix. In a valued matrix, an absent line is represented by a 0 in the cell.

In Pajek, networks of 100 vertices or more cannot be displayed in this way, because they yield enormous matrices. Therefore, the options “binary” and “valued” are not available for larger networks, which are

*Network
drop-down
menu*

	1	2	3	4	5	6	7	8	9	0	1	1	1	1	1	1	1	1	2	2	2	2	2	
Xavier	1.	#	.	.	.	#
Utrecht	2.	.	.	.	#	.	#	.	.	#	#
Frank	3.	#
Domingo	4.	#	#	.	#	.
Norm	5.	.	#	#	.	.	#	.	#	#	#	.

Figure 111. Partial listing of the strike network as a binary matrix.

automatically reported as lists of arcs and edges. In these lists, each line represents a vertex, which is identified by its number and label followed by the numbers of all vertices that receive a line from it. This type of listing is also the third option (“Lists”) for displaying small networks.

File>
Network>
Export as
Matrix to
EPS> Original

The raw text matrices are not suited for high-quality printing. To this end, the matrix can be saved with the command *File> Network> Export as Matrix to EPS> Original* in PostScript format. Line values are automatically translated to the darkness of cells. Larger networks can be exported in this way, but large matrices are usually not very helpful visualizations for detecting the structure of a network.

Partition>
Make
Permutation

As we have argued, a matrix is usually more informative if it is reordered. In the example of the strike network, we must reorder the vertices according to their membership of the ethnic and age groups, which is available to us as a partition (`strike_groups.clu` included in the project file `strike.paj`). It is easy to derive a permutation from a partition such that vertices in the same class receive consecutive numbers: Select the partition in the *Partition* drop-down menu and execute the *Make Permutation* command, which is located in the *Partition* menu.

Given that the command is frequently used, you can run it also by pressing *F3*. Pajek creates a new permutation assigning the lowest vertex numbers to the vertices in the first class of the partition, and so on.

Permutation
drop-down
menu

The permutation is displayed in the *Permutation* drop-down menu of Pajek’s main screen. You may inspect and edit a permutation in the usual ways. When you edit a permutation, you will see one line for each vertex containing two numbers. The first number is the new vertex number, and the second number is the original vertex number. If a compatible network is active in the *Network* drop-down menu, the vertex label is also displayed.

When the network, the partition, and the permutation are selected in their respective drop-down menus, you can export the adjacency matrix to an Encapsulated PostScript file with the command *File> Network> Export as Matrix to EPS> Using Permutation + Partition* (or shortcut *F4*). A dialog box prompts for a name of the file in which the matrix

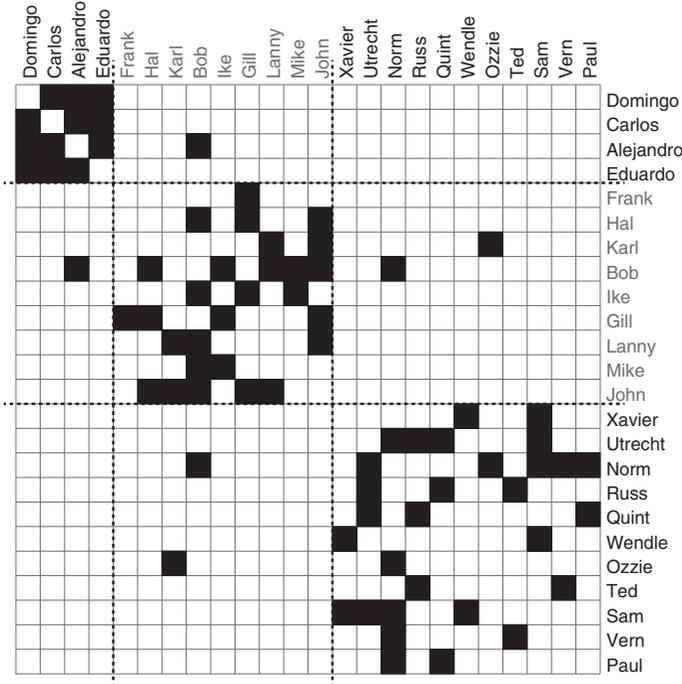


Figure 112. The strike network permuted according to ethnic and age groups.

must be saved. In a viewer capable of reading PostScript (see Appendix 2), the result should look like that in Figure 112. Check out also some other options available in the *File> Network> Export as Matrix to EPS> Options* submenu, e.g., drawing negative values as diamonds (for signed graphs), drawing in grayscale, displaying labels on top/right, displaying labels using partition colors.

The permutation of ethnic and age groups can also be used to reorder the network itself. If the network and permutation are selected in their drop-down menus, the *Operations> Network + Permutation> Reorder Network* command creates a new permuted network. Display the reordered network as a binary matrix by double-clicking its name in the drop list, and you will see that the four Hispanic employees have received vertex numbers from 1 to 4 (Figure 113). Note that the original partition according to ethnicity and age is not compatible with the permuted network, but it can also be reordered: Make sure that the original partition and permutation are active in their drop-down menus and execute the command *Operations> Partition + Permutation> Reorder Partition*. In the same way vectors can be reordered.

*File>
Network>
Export as
Matrix to
EPS> Using
Permutation +
Partition*

*File>
Network>
Export as
Matrix to
EPS> Options*

*Operations>
Network +
Permutation>
Reorder
Network*

*Operations>
Partition +
Permutation>
Reorder
Partition*

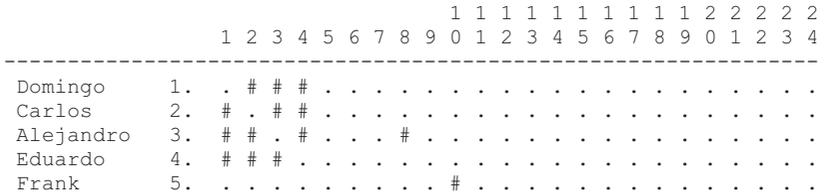


Figure 113. Part of the permuted strike network displayed as a binary network.

12.3 Roles and Positions: Equivalence

In social theory, positions and roles are important and related theoretical concepts. A position, for instance, the position of being an instructor at a university, is usually connected to a social role or a role set, namely, tutoring students and conferring with colleagues. It is hypothesized that this role or role set involves a particular pattern of ties and relations toward students, colleagues, and superiors. Sociologists, social psychologists, and other social scientists investigate the nature of social roles and role sets by observing interactions and by interviewing people about their motives and their perceptions of the roles they play.

In social network analysis, we concentrate on the patterns of ties. We want to identify actors that have similar patterns of ties to find whether they are associated with a particular role or role set, or we want to check whether people with similar role sets are involved in characteristic patterns of ties. In social network analysis, a *position* is equated to a particular pattern of ties. Actors with similar patterns of ties are said to be relationally *equivalent*, to constitute an *equivalence class*, or to occupy *equivalent positions* in the network.

Figure 114 offers a simple example illustrating these ideas. Two instructors (i1 and i2) within one department supervise three students (s1 to s3). They contact the students, and they are contacted by the students. The instructors interact, so they are a cohesive subgroup and their interaction

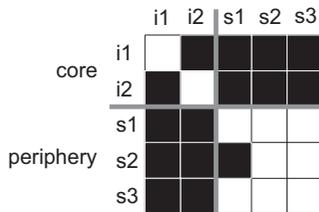


Figure 114. Hypothetical ties among two instructors (i) and three students (s).

may cause them to behave in a similar way. The three students, however, do not necessarily interact. Nevertheless, they are in the same position with respect to the supervisors; hence, they may act similarly toward them. They are relationally equivalent although they are not a cohesive subgroup. It is important to note that the external ties to members of other positions are just as important to the concept of equivalence as internal ties within a position.

Figure 114 is an example of a small core-periphery structure in which the two instructors constitute the core (one position) and the students constitute the periphery (the other position). Ties occur predominantly within the core and between the core and the periphery, so we see a horizontal and a vertical strip of ties in the permuted matrix.

So far, we have loosely described the concept of equivalence. Now let us define one type of equivalence formally, namely, structural equivalence: Two vertices are structural equivalent if they have identical ties with themselves, each other, and all other vertices. This definition implies that structural equivalent vertices can be exchanged with no consequences to the structure of the network.

Two vertices are *structural equivalent* if they have identical ties with themselves, each other, and all other vertices.

In our example, in which arcs are either present or absent, let us compare the two vertices in the core (instructors i1 and i2). Clearly, the two instructors have identical ties to themselves and to each other: None of them communicates with himself or herself (no loops), and the tie among them is symmetric. In addition, their ties with vertices in the other position – the students – are also identical. If instructor i1 is connected to a student (e.g., student s2), then the other instructor is also connected to this student. As a consequence, the rows of the two instructors are identical, except for the cells on the diagonal because they are not supposed to contact themselves. The same is true for their columns, which represent the ties received by the instructors. We may exchange the two instructors without changing the structure of the network.

In general, we can say that vertices that are structural equivalent have identical rows and columns (except for the cells on the diagonal) in the adjacency matrix. With this in mind, it is easy to see that the three students in the periphery (s1, s2, and s3) are not completely structural equivalent because vertex s2 is related to vertex s1, but the reverse is not true, so they have no identical ties to each other. Student s3 is not related to s1, so he or she is not structural equivalent to s2.

Structural equivalence is based on the similarity or dissimilarity between vertices with respect to the profile of their rows and columns

Table 23. Dissimilarity scores in the example network

	i1	i2	s1	s2	s3
i1	0.0000	0.0000	0.1875	0.1875	0.2500
i2	0.0000	0.0000	0.1875	0.1875	0.2500
s1	0.1875	0.1875	0.0000	0.1250	0.0625
s2	0.1875	0.1875	0.1250	0.0000	0.0625
s3	0.2500	0.2500	0.0625	0.0625	0.0000

in the adjacency matrix. The dissimilarity of two vertices can be calculated and expressed by an index that ranges from 0 (completely similar) to 1 (completely different). In Figure 114, the row and column of instructor i1 is perfectly similar to the row and column of instructor i2, so their dissimilarity score is 0 (see Table 23). Students s1, s2, and s3 are not completely similar in this respect, so their dissimilarity score is larger than 0 (ranging from 0.0625 to 0.125), but they are more similar to each other than to the instructors in the core (dissimilarities of 0.1875 or 0.25).

Knowing the dissimilarities between all pairs of vertices, how can we cluster vertices that are (nearly) structural equivalent into positions? This can be achieved with a well-known statistical technique, which is called *hierarchical clustering*. First, this technique groups vertices that are most similar. In our example, instructors i1 and i2, who are completely similar with respect to their ties, are merged into a cluster. Then, hierarchical clustering groups the next pair of vertices or clusters that are most similar, and it continues until all vertices have been joined.

Figure 115, which is called a *dendrogram*, visualizes the clustering process. You must read it from left to right. First, instructors i1 and i2 are joined because they are perfectly similar: Their dissimilarity is 0. Then, students s1 and s3 are joined (at dissimilarity level 0.06, see Table 23). In the third step, student s2 is added to the cluster of s1 and s3. Finally, this cluster is merged with the cluster of core vertices (i1 and i2) in the last step of the clustering process.

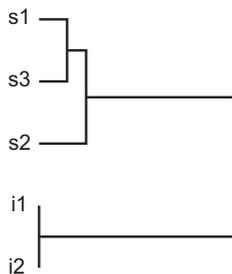


Figure 115. A dendrogram of similarities.

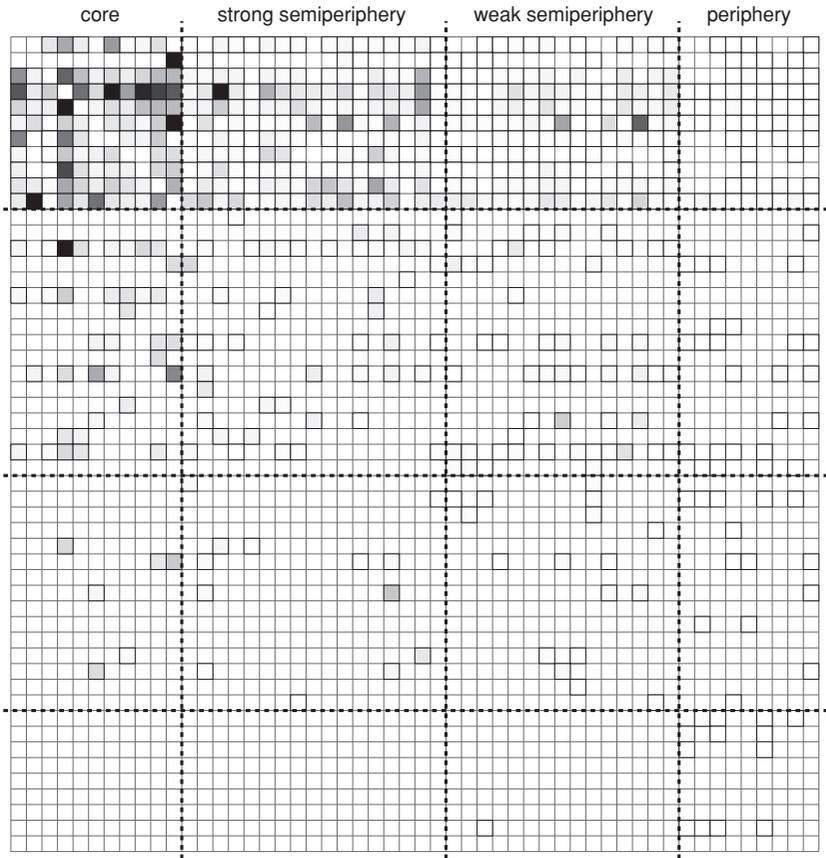


Figure 116. Imports of miscellaneous manufactures of metal and world system position in 1980.

In the dendrogram, the length of a horizontal branch represents the dissimilarity between two vertices or clusters at the moment when they are joined, so you can see that the last step merges two very different clusters. If you want to partition the vertices into two clusters, you should separate instructors i1 and i2 from the students. In general, a hierarchy of clusters is split at the place or places where the branches make large jumps. In this way, you can detect clusters of vertices that are structural equivalent or nearly structural equivalent.

Application

Let us apply the concept of structural equivalence to the world trade network, which we introduced in Chapter 2. The Pajek project file `world_trade.paj` contains the network and a partition identifying world system positions in 1980. Figure 116 shows the matrix containing

Operations>
 Network +
 Partition>
 Extract>
 SubNetwork
 Induced by
 Union of
 Selected
 Clusters

the countries with known world system position in 1980 (we extracted classes 1 to 4 of the partition from the network with the *Operations> Network + Partition> Extract> SubNetwork Induced by Union of Selected Clusters* command). Line values indicate the gross value of imports of miscellaneous manufactures of metal; they are represented by the color of the cells in the PostScript matrix: Higher values are represented by darker cells. The distribution of gross imports is highly skewed because a couple of countries trade very high volumes of goods. We changed all imports more than US\$1 billion to US\$1 billion to obtain slightly darker cells for trade ties with lower gross value. Note that these adjustments are made only for a better display of the matrix. We use the original trade network in the remainder of this section.

The network is directed, so the matrix is not symmetric although the values of imports are often in the same range as the values of exports. The matrix reveals some characteristics of a core-periphery structure that we have noted before: many and strong ties within the core and between the core and the semiperiphery but few and weak ties within the semiperiphery and the periphery. As a result, the ties concentrate in the horizontal and vertical strip that is associated with the core countries.

Cluster>
 Create
 Complete
 Cluster

Now, let us calculate the dissimilarity of the rows and columns of the countries in the original trade network. First, we must make a preliminary step. The dissimilarity method is computationally complex, so it should be used for small networks or for a small part of a large network. Therefore, the method requires that we indicate which vertices it should use. We must identify them in a special data object, which is called a *cluster*. In our example, we want to include all countries, so we create a cluster containing all vertices with the *Cluster> Create Complete Cluster* command. The total number of vertices in the network is shown by default in the dialog box issued by this command, so you may simply press the OK button. The cluster created by this command is listed in the *Cluster* drop-down menu, and it can be edited in the usual way.

Partition>
 Make Cluster>
 Vertices from
 selected
 Clusters

If you want to restrict your analysis to a part of the network, however, identify the vertices for which you want to compute dissimilarities in a partition and translate the desired class or classes from this partition into a cluster with the *Partition> Make Cluster> Vertices from selected Clusters* command. Dialog box will prompt you for the class number or range of class numbers of the partition that must be selected. For example, you may restrict the calculation of dissimilarities to the core countries of 1980 by translating class 1 of the world system positions partition to a cluster. In this case, the *Dissimilarities* command, which we discuss next, calculates dissimilarities for the core countries only, but it takes into account the ties of the core countries to noncore countries.

Because we need a network and a cluster to compute dissimilarities in Pajek, the *Dissimilarity** commands are located in the *Operations* menu.

Dissimilarities are computed from network structure: Vertices are similar if they have many common neighbors. The other possibility, which is computing dissimilarities according to vector values (*Vector based*), will not be discussed here. There are several dissimilarity indices, but we present and use only the index *d1*. Consult a handbook on numerical taxonomy to learn more about the other indices (see “Further Reading” at the end of this chapter). The dissimilarity *d1* of two vertices is simply the number of neighbors that they do not share (normalized to the interval 0–1). This index may be restricted to input neighbors (*Input*; the columns are compared) or output neighbors (*Output*; the rows are compared), or it may consider both input and output neighbors (*All*). Choose the *All* command unless you have good reasons for concentrating on either input or output neighbors.

The *d1* dissimilarity index examines the neighborhoods of vertices, so it does not consider the values of lines. If you want the dissimilarity scores to reflect line values, you should select the Euclidean or Manhattan distance indices (*d5* and *d6*). In the world trade example, using Euclidean or Manhattan distance would require structural equivalent countries not merely to export to and import from the same countries but, on top of that, have trade ties of comparable intensity. This, however, might be too harsh a criterion because the value of imports varies dramatically among countries. We therefore recommend the *d1* index here.

Now execute the *Operations> Network + Cluster> Dissimilarity*> Network based> d1> All* command. Pajek calculates the dissimilarities and reports them in the Report screen if the option *Operations> Network + Cluster> Dissimilarity*> Network based> Options> Report Matrix* has previously been selected (note: do not select the other options in this submenu). The command stores the dissimilarities as line values in a new network, which you may list or print in the usual ways (see Section 12.2). Note that this network is directed and very dense because each pair of vertices that are not completely similar (hence: have a dissimilarity larger than 0) are connected by a pair of arcs. As a rule, do not attempt to draw and energize this network.

While executing a dissimilarities command, Pajek automatically tries to apply hierarchical clustering to the newly created network of dissimilarities. It prompts the user to specify the name of a file in which the dendrogram of the clustering is stored. The dendrogram is saved and not shown because it is in Encapsulated PostScript format. You can view it (Figure 117) with a PostScript interpreter (see Appendix 2), insert it in Word or some other text editor, or print it on a PostScript printer. In addition, the results of the hierarchical clustering are saved as a permutation and a hierarchy.

The dendrogram of the world trade network, which is depicted in Figure 117, shows two very dissimilar clusters of countries in the world

```
Operations>
Network +
Cluster>
Dissimilarity*>
Network
based> d1>
All

Operations>
Network +
Cluster>
Dissimilarity*>
Network
based>
Options>
Report Matrix
```

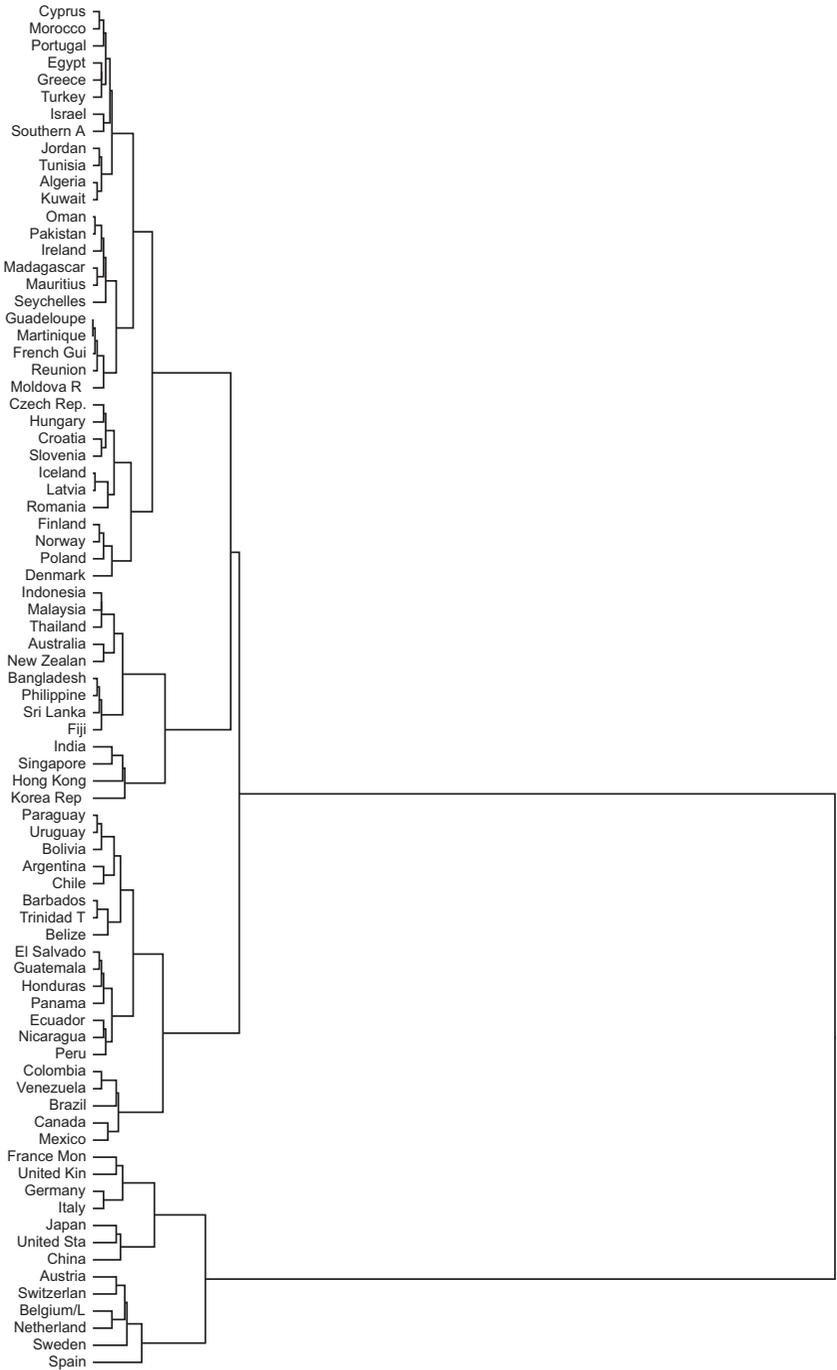


Figure 117. Hierarchical clustering of the world trade network.

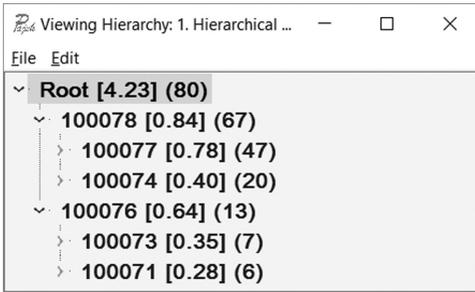


Figure 118. Hierarchical clustering of countries in the Hierarchy Edit screen.

trade network: ten Western European countries, the United States, Japan, and China are clearly separated from the remaining sixty-seven countries, most of which are poorer countries.

This can also be inferred from the hierarchy created as a result of the *Dissimilarity** command, which is labeled “Hierarchical Clustering [Ward].” Note that “Ward” refers to the default hierarchical clustering method in Pajek. Other methods are accessible from the *Network> Create Hierarchy> Clustering** dialog screen. Open the hierarchy in an Edit screen (click on the button with the magnifier at the left of the *Hierarchy* drop-down menu), and expand the root as well as the next layer of clusters by clicking on them to obtain the listing depicted in Figure 118. The root unites the two principal clusters. The figures in square brackets tell you the dissimilarity of the clusters or vertices that are joined; a larger value means that they are more dissimilar. The cluster of thirteen countries is internally more similar (0.64) than the larger cluster (0.84), which corresponds to the fact that the first split within the larger group is more to the right than the split within the smaller group in the dendrogram (Figure 117).

How do we know which countries belong to a particular cluster? We can find the names of the countries in the Hierarchy Edit screen in the following way, provided that a compatible network is active in the *Network* drop-down menu. First, make sure that the option *Show Subtree* is selected in the *Edit* menu of the hierarchy’s Edit screen. Otherwise, Pajek displays only the names of the vertices that were added to the cluster in the present step of hierarchical clustering. Second, select a cluster in the Edit screen by left-clicking (to select it) and right-clicking it subsequently. In a new window, the numbers and labels of the vertices in this cluster and all of its subclusters are listed. If you apply this to the cluster labeled “100071,” for example, you will see that it contains Austria, Switzerland, Belgium/Luxembourg, the Netherlands, Sweden, and Spain.

Network>
Create
Hierarchy>
*Clustering**
Hierarchy Edit
screen

[Hierarchy Edit
screen] Edit>
Show Subtree

Hierarchical clustering gradually merges vertices into clusters and small clusters into larger clusters. Which clusters represent structural equivalence classes and which do not? Under a strict approach to structural equivalence, vertices with zero dissimilarity are structural equivalent. In real social networks, however, such vertices are seldom found, so we consider clusters of vertices that are not very dissimilar to represent structural equivalence classes.

Which vertices are not very dissimilar? There is no general answer to this question. It is up to you to decide on the number of equivalence classes that you want, that is, how many times you want to cut up the dendrogram, but you should always cut it from “right to left”: Separate the most dissimilar clusters first. In the world trade example, you should separate the thirteen rich countries from the other countries first. Then, you could make a subdivision within the latter cluster because these countries are more dissimilar (0.84) than the thirteen rich countries (0.64), and so on, until you reach the desired number of equivalence classes or further subdivisions seem to be arbitrary or meaningless.

Let us divide the trade network into four structural equivalence classes, because we have a partition into four world system positions (core, strong semiperiphery, weak semiperiphery, and periphery). We split the cluster of sixty-seven countries (dissimilarity is 0.84) and its largest subcluster (dissimilarity is 0.78). Now, we can create a partition from the hierarchy that identifies these four clusters. This is done in two steps.

[Hierarchy Edit
screen] Edit>
Change Type

First, we must close the clusters in the hierarchy that we want to split no further. Select a cluster by left-clicking it in the Hierarchy Edit screen and select *Change Type* from the *Edit* menu of the Hierarchy Edit screen or press *Ctrl-t*. Now, the message (close) appears behind the selected cluster. Repeat this for the other clusters that must be closed, but do *not* apply it to any cluster that must be subdivided.

Hierarchy>
Make Partition

Second, execute the *Make Partition* command from the *Hierarchy* menu in the Main screen. This command creates a partition in which each closed cluster is represented by a class. When you draw this partition in the original world trade network, you will notice that the equivalence classes represent a mixture of trade position and geography; the core countries, which are Western European countries, the United States, Japan, and China, are delineated from three regional positions: the Americas, Asia with Oceania, and Europe (including former colonies) with the Middle East.

File>
Network>
Export as
Matrix to
EPS> Using
Permutation +
Partition

So far, we have discussed the dendrogram and the hierarchy created by the *Dissimilarities* command but not the permutation. The permutation is labeled “Hierarchical Clustering Permutation [Ward],” and it identifies the order of the vertices as represented in the dendrogram. When you want to print the matrix reordered by the results of hierarchical clustering, you can use this permutation. It is compatible with the partition that you

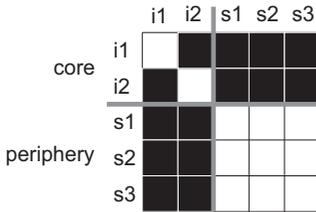


Figure 119. An ideal core-periphery structure.

created from the hierarchy, so you can obtain a matrix with blue lines indicating the splits that you have made in the hierarchy of clusters (see Section 12.2).

Exercise I

Apply hierarchical clustering to the strike network (Pajek project file `strike.paj`), and delineate the most likely clusters. To what extent do these clusters reflect the groups according to age and ethnicity?

12.4 Blockmodeling

In previous sections, we have drawn adjacency matrices with (blue) lines demarcating classes of vertices, for example, ethnic/age groups among the striking employees (Section 12.2), advisors versus students in a small example network, and world system positions of countries in the trade network (Section 12.3). By now, we should note that these lines divide the adjacency matrix into rectangles, and these rectangles are called *blocks*.

A *block* contains the cells of an adjacency matrix that belong to the cross section of one or two classes.

We can describe the structure of the network (within and between positions) by analyzing the blocks of the adjacency matrix. The blocks along the diagonal express the ties within a position. In an ideal core-periphery structure (e.g., Figure 119), vertices are linked within the core (vertices *i1* and *i2*), whereas the peripheral vertices (*s1* through *s3*) are not directly linked. The blocks off the diagonal represent the relations between classes, namely, the relations between the core and periphery. The students derive their identity from their dependence on the instructors but not from their internal ties (instead, their identity is based on the absence of internal ties).

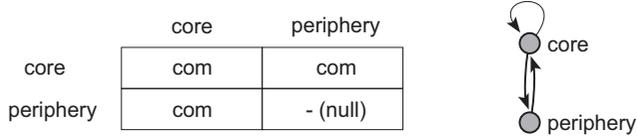


Figure 120. Image matrix and shrunk network.

12.4.1 Blockmodel

Adjacency matrices of networks containing structural equivalence classes have a very remarkable feature, namely, their blocks are either complete or empty (null blocks), if we disregard cells on the diagonal. This results from the criterion of structural equivalence that equivalent vertices have identical rows and columns.

To understand this, imagine that there is one tie among the students of Figure 119, for instance from s_2 to s_1 . Structural equivalent vertices must have identical ties to each other, so s_1 must also be connected to s_2 . If all students are structural equivalent, s_3 must have identical ties as s_1 and s_2 , so it must be linked with s_1 and s_2 . Now, the block is complete, except for the diagonal. This is also true for ties between positions.

Now that we know that the adjacency matrix of a network with structural equivalence classes contains only complete and null blocks, we may simplify the adjacency matrix by shrinking each class of vertices to one new vertex (entry in the matrix) and mark the block type of each cell in the new matrix, which is either complete (com) or empty (– or null) in the case of structural equivalence. This shrunken matrix is called an *image matrix*, and it contains all information that was present in the original adjacency matrix. Figure 120 shows the image matrix of a simple core-periphery structure and a graphical representation of the relations within and between equivalence classes (positions) in which an arc indicates a complete block and the absence of an arc signifies a null block.

A *blockmodel* assigns the vertices of a network to classes, and it specifies the permitted type(s) of relation within and between classes.

The image matrix is the last ingredient we need to define a blockmodel. A *blockmodel* for a network consists of a partition and an image matrix. The partition assigns vertices to equivalence classes, and it divides the adjacency matrix of the network into blocks. The image matrix specifies the types of relations within and between the classes because it says which kinds of blocks are allowed and where they may occur. The blockmodel of the core-periphery structure of Figure 119, for instance, consists of a partition that assigns instructors i_1 and i_2 to one class and the three

students (s1, s2, and s3) to another class and the image matrix specifying the relations between the blocks shown in Figure 120.

A blockmodel describes the overall structure of a network and the position of each vertex within this structure. In the example of the instructors and students, the image matrix shows the type of equivalence that applies to the network. This network contains structural equivalence classes because there are only complete and empty blocks. In addition, the image matrix reveals the core-periphery structure of the network because the complete blocks are arranged within one horizontal strip and one vertical strip. Class 1 represents the core, which is internally linked, and class 2 identifies the periphery. Finally, the partition tells us which actors are part of the core (the two instructors, who constitute class 1) and which actors belong to the periphery (the three students in class 2). A blockmodel is an efficient device for characterizing the overall structure of a network and the positions of individual vertices.

12.4.2 Blockmodeling

Until now, we have assumed that we knew the blockmodel of a network, that is, the partition of vertices into classes and the image matrix specifying the permitted types of blocks. In a research project, naturally, we work the other way around: We have a network, and we want to find the blockmodel that captures the structure of the network. The technique to obtain this blockmodel is called *blockmodeling*.

In general, blockmodeling consists of three steps. In the first step, we specify the number of classes in the network, for instance, two classes or positions if we hypothesize a simple core-periphery structure. In the second step, we choose the types of blocks that are permitted to occur and, optionally, the locations in the image matrix where they may occur. In the case of structural equivalence, for instance, we permit only complete and empty blocks to occur and we expect one complete block (the core) and one empty block (the periphery) along the diagonal. Finally, the computer partitions the vertices into the specified number of classes according to the conditions specified by the model and, if necessary, it chooses the final image matrix for the model. In this third step, the blockmodel is completed.

The first two steps define the image matrix: We fix the number of classes and the types of blocks (relations), but we do not yet know which vertices belong to a particular class and sometimes we do not know exactly which block type will be found in which part of the image matrix. That is settled in the third step. It goes without saying that we must have some knowledge or expectations about the network to choose an appropriate number of classes and to specify types of relations among classes that make sense. We should have reasons or clues for expecting a core-periphery structure

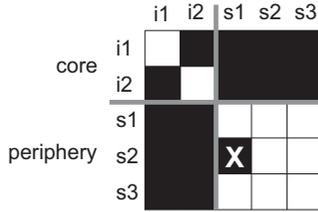


Figure 121. Error in the imperfect core-periphery matrix.

and structural equivalence in the example of contacts between instructors and students.

Empirical networks, however, seldom match the ideal represented by the image matrix. Errors occur, but they can be checked easily. Suppose you know which vertices belong to each class, then you can check whether each block of the adjacency matrix is of the right type according to the image matrix. In fact, you compare an ideal matrix (Figure 119) to the real matrix (Figure 114). In the case of structural equivalence, count the missing lines within the blocks that should be complete (none in this example), and count the number of lines that occur in the blocks that should be empty (one error: the arc from student *s1* to student *s2*, see Figure 121) to obtain an error score that indicates how well the ideal matrix fits the real network.

In this approach, the third step of blockmodeling boils down to finding the partition of vertices into equivalence classes that yields the lowest error score, that is, that fits the ideal matrix best. First, the computer assigns vertices at random to the specified number of classes. Then, it calculates the error score of this solution by comparing the actual matrix to an ideal matrix represented by the image matrix. Next, it tries to decrease the error score by moving a randomly selected vertex from one to another cluster or by interchanging two vertices in different clusters. It continues this process until it can no longer improve the error score.

This optimization approach to blockmodeling has the advantages and disadvantages of all optimization techniques (e.g., the *Doreian-Mrvar method**), namely, if applied repeatedly, it is likely to find the optimal solution, but most of the time you cannot be sure that no better solution exists. In addition, you must be aware that another number of classes or other permitted types of blocks may yield blockmodels that fit better. Usually, it is worthwhile to apply several slightly different blockmodels to the data set, namely, with another number of classes or other constraints on the relations within or between blocks. This underlines the importance of careful considerations on the part of the researcher concerning the image matrix that is hypothesized. Moreover, trees are troublesome in exploratory blockmodeling because they contain many vertices that may

be exchanged between classes without much impact on the error score, so apply blockmodeling only to rather dense (sections of) networks.

In this optimization technique, errors can be weighted and line values can be used. We do not go into details here, but it should be noted that lower error scores indicate better fit, and an error score of 0 always represents a perfect fit.

Application

As noted, blockmodeling consists of three steps. In the first two steps, the image matrix is specified: the number of classes and the types of blocks or relations within and between classes that are allowed. Then, the computer completes the blockmodel by searching the partition of vertices into classes that match the hypothesized image matrix best. If several image matrices are possible, it chooses the one that fits best. The error score shows how well the selected image matrix fits the network.

The blockmodeling commands of Pajek reflect these three steps. Before we discuss these commands, however, we must warn you that the method, like all optimization techniques, is time-consuming, so it should not be applied to networks with more than some hundreds of vertices, in which case the computer may need a full day to execute the command. For this reason the command is marked by a star in the menu.

In Pajek, there are two blockmodeling methods: One searches for the best fitting partition from scratch (*Random Start*), whereas the other only tries to improve an existing partition (*Optimize Partition*). Let us start with the latter method and apply it to the world trade network using the world system positions in 1980 as the starting partition. Both files are available in the Pajek project file `world_trade.paj`. Delete the countries with unknown world system position in 1980 from the network (*Operations*> *Network* + *Partition*> *Extract*> *SubNetwork Induced by Union of Selected Clusters* classes 1–4, see Section 12.3). Thus, we have selected fifty-two of eighty countries. The command also creates a new partition with the world system positions of the remaining fifty-two countries.

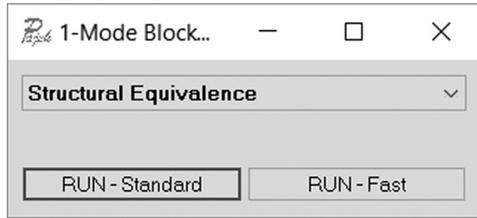
We will explain here only the basic options for blockmodeling. Therefore, check *Network*> *Create Partition*> *Blockmodeling**> *Restricted Options* and *Network*> *Create Partition*> *Blockmodeling**> *Short Report* to get the same dialog screens as the ones reproduced in this chapter. You can uncheck these options later when you become more familiar with blockmodeling. When you select the *Optimize Partition* command from the *Network*> *Create Partition*> *Blockmodeling** sub-menu, the active partition specifies the number of equivalence classes you are looking for, which is the first step of blockmodeling. On selection of the command, a dialog box opens (Figure 122). The selection box shows the last selected type of equivalence. We want to apply structural

Operations>
Network +
Partition>
Extract>
SubNetwork
Induced by
Union of
Selected
Clusters

Network>
Create
Partition>
*Blockmodeling**>
Restricted
Options

Network>
Create
Partition>
*Blockmodeling**>
Short Report

Network>
Create
Partition>
*Blockmodeling**>
Optimize
Partition

Figure 122. *Optimize Partition* dialog box.

equivalence so select this type of equivalence if the list box does not yet read “Structural Equivalence.” Change none of the other options; just press the *RUN – Standard* button to execute the command. We will not discuss the second possibility (*RUN – Fast*) here.

Pajek lists the initial settings in the Report screen, as well as the initial image matrix, the initial error matrix, and the error score of the initial partition. In our example, 366 initial errors are reported: in 366 (of the $52 \times 51 = 2652$) cells, imports are absent where they should be present and vice versa. By default, Pajek does not take into consideration line values, so it pays no attention to the value of imports here. Next, Pajek tries to improve the partition and creates the best fitting partition it has found and reports the final image matrix, the final error matrix, and the associated error score (see Figure 123). The optimal partition fits a little bit better than the world system positions in 1980 because the error score has decreased from 366 to 339. However, we do not know whether this

Image Matrix:

	1	2	3	4
1	com	com	com	com
2	-	-	-	-
3	-	-	-	-
4	-	-	-	-

Error Matrix:

	1	2	3	4
1	7	28	32	39
2	36	50	55	28
3	6	13	17	14
4	0	0	1	13

Final error = 339.000

Figure 123. Output of the Optimize Partition procedure.

Table 24. *Cross-tabulation of initial (rows) and optimal partition (columns)*

	1	2	3	4	Total
1 (core)	10	1	0	0	11
2 (strong semiperiphery)	0	17	0	0	17
3 (weak semiperiphery)	0	0	15	0	15
4 (periphery)	0	0	0	9	9
TOTAL	10	18	15	9	52

is a small or large error score for a network like this, and maybe another number of classes or other permitted block types would yield a better solution.

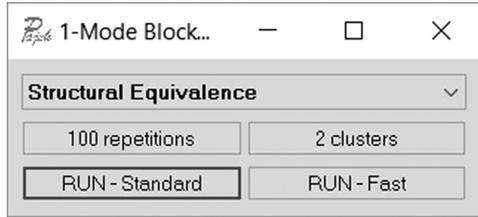
Note that the final image matrix has a very clear structure: The cells in the first row are all complete, whereas all other cells are empty. This means that every core country (class 1) exports miscellaneous manufactures of metal to all other countries, but no other country exports these products in the blockmodel: Their rows only contain empty (null) cells. The error score indicates that some of these countries do export miscellaneous manufactures of metal, but the blockmodel assumes that they are not.

The best fitting partition is equal to the initial world system partition except for one country that has been moved from the core to the strong periphery. You may check this by selecting the initial partition and the new partition as the first and second partition in the *Partitions* menu, respectively, and executing the *Partitions> Info> Cramer's V, Rajski, Adjusted Rand Index* command. Table 24 shows the cross-tabulation of the original (rows) and optimized partition (columns). Almost all countries are on the diagonal, indicating that they remain in their original class. Just one country moves from the first row (core) to the second column (strong semiperiphery).

The second method searches for the best fitting partition without taking into account an initial partition provided by the user of the program. Therefore, no initial partition is needed for the *Random Start* command. The dialog box displayed by this command offers the possibility to specify the number of classes (step 1), the kind of equivalence or blockmodel (step 2), and the number of repetitions (see Figure 124). Each repetition uses a new, random partition as a starting point to avoid settling on a local optimum. Change these choices by clicking on the buttons and entering the required numbers; for instance, change the number of clusters to 4 (core, strong semiperiphery, weak semiperiphery, and periphery) and the number of repetitions to 100.

*Partitions>
Info> Cramer's
V, Rajski,
Adjusted Rand
Index*

*Network>
Create
Partition>
Blockmodeling* >
Random Start*

Figure 124. *Random Start* dialog box.

Applied to the fifty-two classified countries in the world trade network, looking for four clusters and structural equivalence, the *Random Start* command finds a partition with 281 errors. This is quite an improvement in comparison to the solution with the world system positions in 1980 as equivalence classes. Now, the procedure does not settle on the image matrix fitting the initial partition best (Figure 123), but it finds another image matrix (Table 25; note that you may get a permutation of this image matrix) in which countries of class 1 export miscellaneous manufactures of metal to all other countries except for the countries in class 3, whereas class 2 countries export to all other countries. Countries of classes 3 and 4 are just importing and not exporting miscellaneous manufactures of metal.

Exercise II

Draw the image matrix that you expect to fit the strike network (*strike.paj*) using structural equivalence. Then, check your expectations by fitting a structural equivalence blockmodel to this network.

12.4.3 Regular Equivalence

Structural equivalence requires that equivalent actors have the same neighbors. In several applications of social network analysis, this criterion is too strict because it does not cluster actors who fulfill the same

Table 25. *Final image matrix of the world trade network*

	1	2	3	4
1	com	com	–	com
2	com	com	com	com
3	–	–	–	–
4	–	–	–	–



Figure 125. Matrix of the student government network.

role in different locations, for instance, teachers at different universities who have different students, so they have ties with similar people but not with the same people.

For these situations, another type of equivalence has been defined: *regular equivalence*. Vertices that are regular equivalent do not have to be connected to the same vertices, but they have to be connected to vertices in the same classes. This sounds like a circular argument, but it is not. In the student government discussion network (Chapter 10), for instance, all advisors are expected to choose ministers for discussing student politics because they are supposed to advise the ministers. However, they do not have to advise the same ministers, and they do not have to advise all ministers (e.g., advisor2 chooses minister1 to minister4 but advisor3 selects minister5 and minister7) (Figure 125). In reverse, each minister is supposed to use the services of at least one advisor, but he or she is not obliged to take advice from all advisors. This is also true for ties within a class: If one minister selects another minister, each minister must select a peer and must be selected by a peer. One peer, however, suffices: They do not have to be related to all peers, so their block is not necessarily complete.

We can detect regular equivalence by means of blockmodeling because there is a special block type associated with regular equivalence, which is called a *regular block*. A regular block contains at least one arc in each row (everyone selects at least one actor) and in each column (everyone is selected at least once). Regular equivalence allows regular blocks and null blocks. Note that a complete block is always a regular block, so structural

image matrix				error matrix					
	1	2	3		1	2	3		
1	-	(null)	-	(null)	1	0	1	0	
2	com	reg	-	(null)	2	2	0	3	
3	-	(null)	reg	-	(null)	3	0	0	2

Figure 126. Image matrix and error matrix for the student government network.

equivalence is a special kind of regular equivalence or, in other words, regular equivalence is more general than structural equivalence.

A *regular block* contains at least one arc in each row and in each column.

In the student government network with three classes (one class for each formal position – see Figure 125), two blocks are regular: the choices of advisors among ministers and the choices among ministers. The block containing choices from ministers to the prime minister would have been complete (thus, regular) if minister3 and minister6 had also chosen the prime minister. The two missing choices are represented by black crosses in Figure 125; they contribute two units to the error score of the regular equivalence model for this network.

In Figure 125, two blocks are empty: the choices from advisors to the prime minister and vice versa. The social distance between these two classes seems to be too large to be spanned by direct consultation. The remaining three blocks are neither null nor regular, so they contain at least one violation against the regular equivalence model. The number of errors is minimal if we assume these blocks to be empty, so all six choices in these blocks are errors (white crosses), and we assume that the ideal matrix contains null blocks here. In our image matrix, we merely specified that all blocks should be empty or regular. While evaluating the error score, we discover that it is least erroneous to expect empty blocks here. Thus, we fix the type of these blocks to null blocks.

Figure 126 shows the image matrix and the number of errors in each block (the error matrix), which summarize the results. Class 1 contains the prime minister, class 2 contains the ministers, and the advisors are grouped in class 3.

The student government discussion network is an example of a ranked structure that entails a particular location of block types. In a ranked structure, actors are supposed to choose up. If the ranks are ordered such that the highest rank is in the first rows (and columns) and the lowest rank occupies the last rows (and columns), we should not encounter choices in

the blocks above the diagonal of the matrix because they would point from a higher rank (rows) toward a lower rank (columns). Indeed, we find empty (null) blocks only above the diagonal in the image matrix of the student government network, which is a general property of a ranked structure.

Instead of using a particular type of equivalence to define the block types that are allowed, we may use any combination of permitted block types to characterize a network by specifying the type(s) allowed for each individual block, for instance, a complete block for the ministers to prime minister block, a regular block for the ministers themselves, and an empty block for the ministers to advisors block. This is known as *generalized blockmodeling*. Note that there are more block types than the three presented here. Some patterns of block types are known to contain classes of networks, namely, core-periphery models and models of ranks. These classes have a particular substantive meaning, so it is easy to interpret them. In the near future, further applications to empirical social networks will probably reveal more classes of blockmodels.

In exploratory social network analysis, we are mainly interested in detecting the blockmodel that fits a particular network. The blockmodel tells us the general structure of the network, and the equivalence classes that we find can be used as a variable in further statistical analysis. We should issue a warning here. We will always find a best fitting blockmodel, even on a random network that is not supposed to contain a regular pattern. Therefore, we should restrict ourselves to blockmodels that are supported by theory or previous results. We should start out with a motivated hypothesis about the number and types of blocks in the network. As in other cases of exploratory network analysis, we should try to validate the result, for example by linking the equivalence classes to external data, such as actor attributes. If equivalence classes of actors have different properties, tasks, or attitudes, this corroborates the interpretation that the blockmodel identifies social roles or role sets.

Application

In Pajek, a blockmodel satisfying *regular equivalence* is found in the same way as a structural equivalence blockmodel (see the previous section): Just replace structural equivalence with regular equivalence in the *equivalence type* drop-down menu (see Figures 122 and 124). If we apply the *Random Start* blockmodeling procedure to the student government discussion network (available in the Pajek project file `student_government.paj`), we find eight solutions with seven errors (under regular equivalence with three classes and hundreds of repetitions). This is a minimal improvement in comparison to the solution with the formal roles as equivalence classes, discussed previously, and it has the disadvantage that a choice must be made among seven alternative solutions. None of the solutions matches

```
Network>
Create
Partition>
Blockmodeling* >
Random Start
```

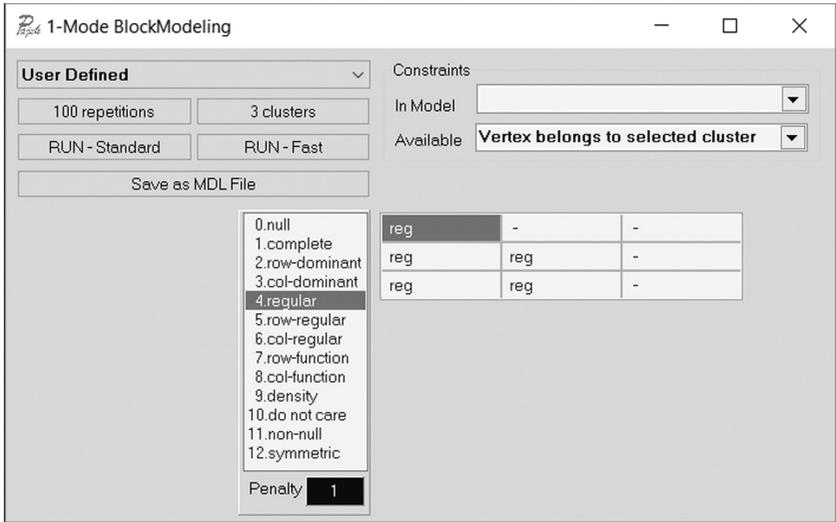


Figure 127. Assembling a blockmodel in Pajek.

the formal roles, but the image matrix resembles the image matrix in Figure 126 or one of its permutations.

Note that another number of classes and another type of equivalence may yield even better solutions, for example, we find four errors in regular equivalence solutions with two classes but the interpretation is difficult: In one solution, advisor2 is separated from the rest of the network, which seems to be a trivial solution; in the other solution, advisor1 and advisor2 are joined by minister4. Therefore, we prefer the original classification according to formal role within the student government.

In blockmodeling, the option *Structural Equivalence* tells Pajek that each block must be either complete (com) or empty (null). In regular equivalence, each block must be either complete, empty, or regular. The user has no control over the location of complete, empty, and regular blocks in the image matrix. In contrast, *generalized blockmodeling* offers the possibility of specifying (and fixing) the equivalence type of each block in the image matrix. For example, we may want to test whether a regular equivalence blockmodel matches the student government discussion network with three classes in which each class advises higher classes (if any) and all except the lowest class advise members in their own class.

The required image matrix is depicted in Figure 127; and it is shown underneath the “Save as MDL File” button if the *User Defined* option is picked in the selection box. If you click on one of the cells (blocks) of this matrix, a list is opened showing thirteen kinds of equivalence. In this list, you can select one or more (press the *Alt* key to add another choice) types

of equivalence that you prescribe for the selected cell. In the example, five cells are forced to be regular equivalent, and the remaining four cells must be empty. In addition, you may raise or lower the penalty of an error in the selected cell if you think that errors in one cell are more or less important than errors in another cell. Just click on the number after “Penalty” and enter a new number.

In the top right corner of Figure 127 you can see that additional constraints can be added to the blockmodel. The constraints concern a priori knowledge about vertices or pairs of vertices, which you can assign to a particular block or prohibit being included in a particular block, and constraints on the minimum and maximum size of blocks (clusters). To add a constraint, double click it in the *Available Constraints* drop-down menu. You will have to provide parameters and a penalty for overriding the constraint, after which the constraint will be added to the *In Model Constraints* drop-down menu. For details, see the reference on generalized blockmodeling in the Further Reading Section.

When you have defined your own blockmodel, you may save it for future use. Press the “Save as MDL File” button and enter a name for the file in which the model must be stored. By default, Pajek gives these files the extension `.mdl` (model), and we strongly advise using this file name extension. In another blockmodeling session, you can open this file by picking the *Load MDL File* option in the selection box. After loading the model, you can inspect it by selecting the *User Defined* option again. Finally, you can run the blockmodeling command.

The number of blockmodels that you can try to fit to a network is immense, especially when you design your own generalized blockmodels. Therefore, we advise the following strategy for exploratory blockmodeling: (1) use the results of other analyses and theoretical considerations to assemble an image matrix; (2) try stricter blockmodels and block types first (structural equivalence is more strict than regular equivalence); and (3) try a smaller number of classes first. Select the blockmodel with the lowest error score, but if a model with a slightly higher error score yields a single solution that is easy to interpret, you should prefer the latter.

Exercise III

Apply the generalized blockmodel to the student government discussion network and evaluate the results.

12.5 Summary

In this chapter, the families of networks presented in previous parts of this book were reviewed once more: cohesive subgroups, core-periphery

structures (brokerage), and systems of ranks. We presented a technique capable of detecting each of these structures, namely, blockmodeling.

In the case of blockmodeling, we need a new representation for networks: the matrix. The adjacency matrix of a network contains its structure; each vertex is represented by a row and a column, and arcs are located in the cells of the matrix: The first row and column belong to the first vertex, the second row and column to the second vertex, and so on. When sorted in the right way, the adjacency matrix offers visual clues on the structure of the network. Such a sorting is called a permutation of the network, which is actually a renumbering of the vertices.

Blockmodeling is not an easy technique to understand. Basically, this technique compares a social network to an ideal social network with particular structural features: a model. The researcher must suggest the model, and the computer checks how well this model fits the actual data.

The model, which is called a blockmodel, contains two parts: a partition and an image matrix. The partition assigns the vertices of the network to classes, which are also called equivalence classes or positions. In the adjacency matrix of the network, the classes demarcate blocks: rectangles of cells. Blocks along the diagonal of the adjacency matrix contain ties within classes, whereas off-diagonal blocks represent relations between classes.

In the image matrix, which is the second part of a blockmodel, each cell represents a block of the adjacency matrix. It is a shrunk and simplified model of the adjacency matrix. If the vertices within a class are structurally similar – equivalent, we say – the blocks in the adjacency matrix have particular features: They are empty, complete, or regular, which means that there is at least one tie from and to each vertex in a block. More types of blocks exist, but we do not present them here.

The image matrix shows which block types are allowed and, possibly, where to expect them. In addition, the distribution of nonempty blocks in the image matrix reveals the overall structure of the network. If the network contains cohesive groups, the nonempty blocks are found along the diagonal of the image matrix. If the network is dominated by a core-periphery structure, we find all nonempty blocks in one horizontal and one vertical strip in the image matrix. Finally, if there is a system of ranked clusters and the vertices are sorted according to their ranks, we find the nonempty blocks in the lower or upper half of the image matrix.

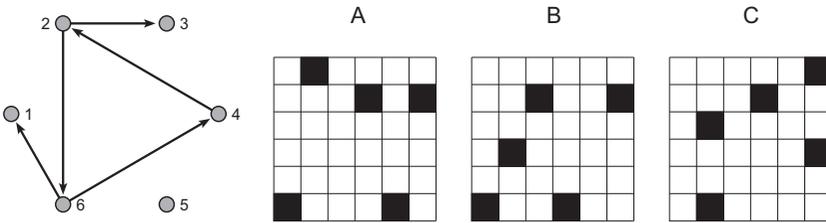
In exploratory blockmodeling, we search for the partition and image matrix that fit a social network best. Empirical social networks seldom match a blockmodel perfectly: Arcs that should be present are absent, or some absent arcs should be present. The number of errors expresses how

well a blockmodel fits the network. This error score is used to evaluate different blockmodels for the same network.

Blockmodeling is a powerful technique for analyzing rather dense networks, but it needs the right input from the researcher to produce interesting results. The number of blockmodels that may be fitted to a social network is large, so it is not sensible to embark on blockmodeling without clear conceptions of and expectations on the overall structure of the network. A researcher needs an informed hypothesis about network structure for a fruitful application of blockmodeling. In this sense, blockmodeling is used for hypothesis testing rather than exploration.

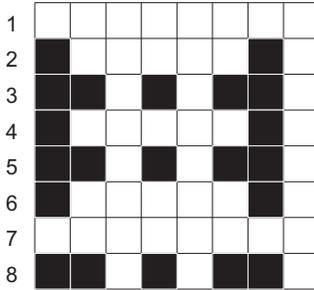
12.6 Questions

1. In the following figure, one sociogram and three adjacency matrices are presented. Which adjacency matrix belongs to the sociogram?

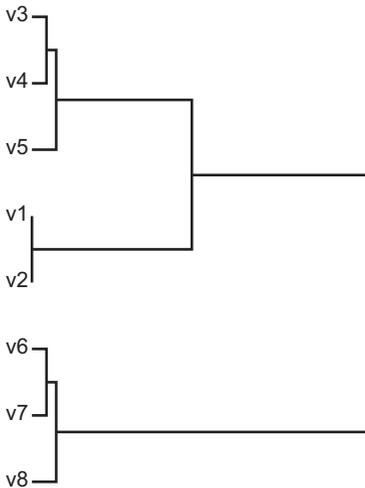


- Matrix A
 - Matrix B
 - Matrix C
 - Each matrix
2. Which of the following statements is correct?
- An adjacency matrix may contain more rows than columns.
 - An adjacency matrix is always symmetric with respect to the diagonal.
 - An affiliation matrix may contain more columns than rows.
 - An affiliation matrix is always symmetric with respect to the diagonal.
3. Of the three adjacency matrices in Question 1, which are isomorphic? It may help to draw the sociograms of the matrices.
- Matrices A and B
 - Matrices A and C
 - Matrices B and C
 - All three matrices are isomorphic.

4. Write down the permutation that reorders one matrix of Question 1 into another matrix of that question.
5. According to the adjacency matrix that follows, which vertices are structural equivalent?



6. The dendrogram that follows displays the results of hierarchical clustering. Which equivalence classes would you make?



7. Which statement is correct?
 - a. Regular equivalence allows for regular and empty blocks, but not complete blocks.
 - b. Structural equivalence allows for complete and empty blocks, but no kind of regular blocks.
 - c. Regular equivalence is a special case of structural equivalence.
 - d. Structural equivalence is a special case of regular equivalence.
8. Assign the vertices of the adjacency matrix depicted in the following figure to a minimum number of regular equivalence classes.

v1									
v2									
v3									
v4	■	■	■						
v5	■	■	□						
v6	□	□	■	■	■				
v7	■	□	■	■	■				
v8	■	■	■	■	■				

9. What kind of structure does the adjacency matrix of Question 8 represent?
- No particular structure
 - Cohesive subgroups
 - A core-periphery structure
 - A system of ranks

12.7 Assignment

In Mexico throughout most of the twentieth century, political power has been in the hands of a relatively small set of people who are connected by business relations, family ties, friendship, and membership of political institutions. A striking case in point is the succession of presidents, especially the nomination of the candidates for the presidential election. Since 1929, each new president was a secretary in the previous cabinet, which means that he worked closely together with the previous president. Moreover, the candidates always entertained close ties with former presidents and their closest collaborators. In this way, a political elite has maintained control over the country.

The network `mexican_power.net` contains the core of this political elite: the presidents and their closest collaborators. In this network, edges represent significant political, kinship, friendship, or business ties.

Notwithstanding the fact that one political party (the Partido Revolucionario Institucional) won all elections in the period under consideration, two (or more) groups within this party have been competing for power. The main opposition seems to be situated between civilians and members of the military (`mexican_military.clu`: the military in class 1 and civilians in class 2). After the revolution, the political elite were dominated by the military, but gradually the civilians assumed power. The partition `mexican_year.clu` specifies the first year (minus 1900) in which the actor occupied a significant governmental position. All data are available in the project file `mexican_power.paj`.

Draw the network into layers according to the year of “accession to power,” and use it to see when the civilians assume power. Use hierarchical clustering and blockmodeling to assess whether the political network consists of two (or more) cohesive subgroups, and check whether these subgroups match the distinction between military and civilians or whether they cover a particular period.

12.8 Further Reading

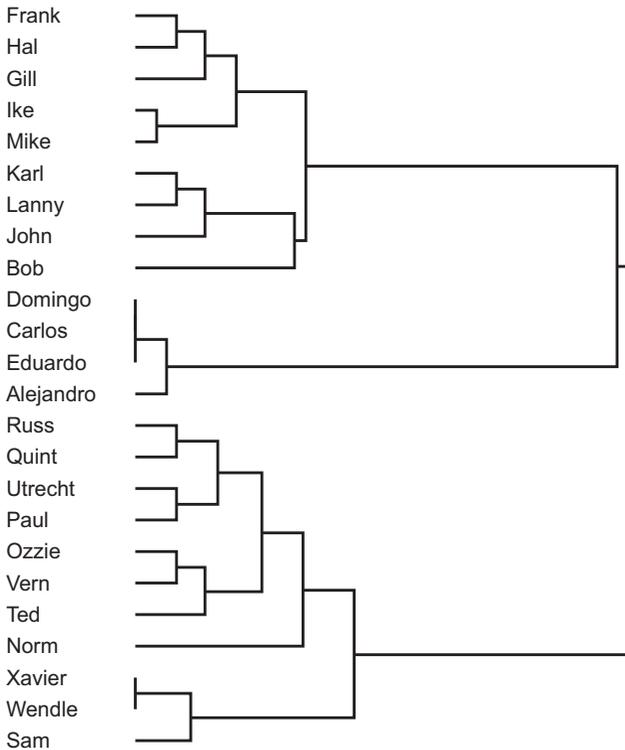
- For an introduction to matrices, see Chapter 4 in J. Scott, *Social Network Analysis: A Handbook* (London: SAGE, 2017); Chapter 3 in A. Degenne and M. Forsé, *Introducing Social Networks* (London: SAGE, 1999); and Section 4.9 in S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications* (Cambridge: Cambridge University Press, 1994).
- M. S. Aldenderfer and R. K. Blashfield’s *Cluster Analysis* (London: SAGE, 1984) offers a helpful introduction to hierarchical clustering. A comprehensive overview of (dis-)similarity measures can be found in P. Sneath and R. Sokal, *Numerical Taxonomy* (San Francisco, CA: Freeman, 1973).
- H. C. White, S. A. Boorman, and R. L. Breiger introduced the technique of blockmodeling in “Social structure from multiple networks. I. Blockmodels of roles and positions.” *American Journal of Sociology* 81 (1976), 730–80.
- Blockmodeling is explained in Chapters 9, 10, and 12 in S. Wasserman and K. Faust (see earlier reference) and in Chapter 4 of A. Degenne and M. Forsé (see earlier reference).
- For generalized blockmodeling, consult P. Doreian, V. Batagelj, and A. Ferligoj, *Generalized Blockmodeling* (Cambridge: Cambridge University Press, 2005).
- The example analyzed in the assignment is taken from J. Gil-Mendieta and S. Schmidt, “The political network in Mexico.” *Social Networks* 18 (1996), 355–81.

12.9 Answers

Answers to the Exercises

- I. Create a complete cluster (*Cluster> Create Complete Cluster*) and determine the dissimilarities in the strike network (*Operations> Network + Cluster> Dissimilarity*> Network based> d1> All*). The dendrogram is depicted in the figure that follows; inspect the hierarchy in the Hierarchy Edit screen if you have no PostScript viewer for

the dendrogram. The largest split distinguishes between the eleven older English-speaking employees (Russ to Sam, cluster 100021 in the hierarchy) and the other employees (cluster 100022). The latter cluster is internally more heterogeneous than the former, according to their values in the hierarchy (3.58 versus 1.62). Therefore, the next great split occurs in the first cluster, and it separates the young English speakers (cluster 100020) from the Hispanic employees (cluster 100005). These two splits exactly yield the three age-ethnic groups. Further splits occur at a far lower level (more to the left in the dendrogram) and may be ignored.



II. In previous analyses, we found three cohesive groups according to age and ethnicity in the strike network, so we may assume that there are three equivalence classes in the blockmodel. In addition, we may expect a cohesive group to be a complete block: Everyone is connected rather than not connected to everyone else within the group. Links between groups are sparse, so we may expect off-diagonal blocks to be empty (null) rather than complete in the image matrix. The hypothesized image matrix is depicted as follows.

	1	2	3
1	com	-	-
2	-	com	-
3	-	-	com

Now, execute the *Network> Create Partition> Blockmodeling*> Random Start* command, choosing structural equivalence, three clusters, and a sufficiently large number of repetitions (some hundreds). Pajek finds two partitions with an error score of 56. Both are saved as a partition, but none of them nicely delineates the three groups according to age and ethnicity. The final image matrix resembles the hypothesized image matrix except for the fact that the third equivalence class is internally not connected (null block) rather than completely connected. In the error matrix, we can see that most mistakes (thirty-eight of fifty-eight) occur here. The young and older English-speaking employees are simply connected too loosely to be recognized as a complete subnetwork (clique).

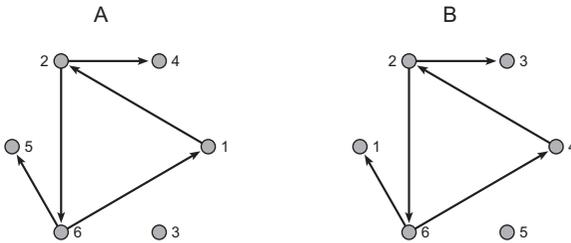
- III. Following the steps outlined in the application part of Section 12.4.3, the special ranked generalized blockmodel can be constructed. Run this model many times on the student government discussion network, and you will obtain two optimal partitions with seven errors each. The model fits the network just as well as the regular equivalence model computed in Section 12.4.3, but now there are just two optimal partitions compared to eight under the regular equivalence model. That is a step forward. Drawing the network and the partitions, we see that two advisors are placed in the third equivalence class, and the first class contains the prime minister and ministers 3 and 7. The results suggest a split among the ministers.

Answers to the Questions in Section 12.6

1. By convention, the first (top) row and the first (left) column represents the vertex with number 1. Vertex 2 is identified by the second row and column, and so on. The sociogram contains an arc from vertex 2 to vertex 3, so the cell at the intersection of the second row and the third column must contain an arc. In adjacency matrix B, this is the case so this is the only matrix that may represent the sociogram. Check the remaining arcs and black cells: They match. Answer b is correct.
2. Answer c is correct. In an affiliation matrix, the rows represent actors, and the columns contain the events to which the actors can be affiliated. The number of actors (rows) is not necessarily equal to the number of events (columns), so the number of columns may be larger than the number of rows (and vice versa). In an adjacency matrix, the rows as well as the columns represent all vertices in the network, so their

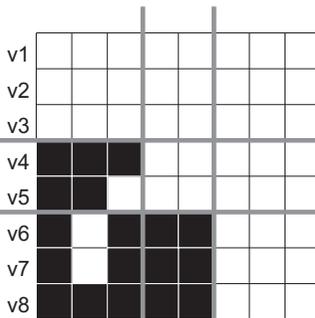
numbers must be equal (answer a is not correct). The adjacency matrix of an undirected network is always symmetric with respect to the diagonal, but this is not necessarily so in the case of a directed network (answer b is incorrect). Finally, it is a coincidence and not a necessity if for each actor u who is affiliated to event v , there would be an actor v who is affiliated to event u . Affiliation matrices are seldom symmetric (answer d is incorrect).

- In matrices A and B, three vertices do not send arcs (their rows are empty), one vertex sends one arc, and two vertices send two arcs. These matrices may describe isomorphic networks. The vertices of matrix C, however, have different outdegree: One vertex has zero outdegree, and the remaining five have one outdegree. The network of matrix C cannot be isomorphic to the networks of matrices A and B. Are the networks of matrices A and B isomorphic? If we draw them, we can see that their structures are identical. Answer a is correct.



- The permutation that transforms matrix A into matrix B can be read from the sociograms drawn in the answer to Question 3 (clockwise): vertex 2 remains vertex 2, 4 becomes 3, 1 becomes 4, 3 becomes 5, 6 remains 6, and 5 becomes 1.
- Vertices with identical rows and columns are structurally equivalent. In the adjacency matrix, vertices 1 and 7 are structurally equivalent, because their rows are empty and their columns contain six arcs sent by the remaining six vertices. Vertices 2, 4, and 6 are also structurally equivalent, and this is also true for vertices 3, 5, and 8.
- In the first step, vertices v_6 , v_7 , and v_8 must be separated from the rest. Then, vertices v_1 and v_2 can be split from vertices v_3 , v_4 , and v_5 . At this stage, the dissimilarities between vertices within a cluster are low, so it is not sensible to further subdivide the three equivalence classes.
- Answer d is correct. A complete block is also a regular block because each row and each column contains at least one entry (arc) within the block. Structural equivalence is a special type of regular equivalence (answer d). The reverse is not true (answer c is incorrect). Therefore, complete blocks are allowed under regular equivalence (answer a is

- incorrect), and a special type of regular block, namely, the complete block, is allowed under structural equivalence (answer b is incorrect).
8. A regular equivalence block is regular (at least one arc in each row and column) or empty. The rows of vertices v_1 , v_2 , and v_3 are empty, so their horizontal blocks are null blocks. No other vertex has an empty row, so we cannot add a vertex to this cluster because we would get blocks in the rows of these vertices that are not empty and not regular because not every row contains an arc. For similar reasons, we may cluster vertices v_6 , v_7 , and v_8 : They have empty columns. If we cluster the remaining vertices v_4 and v_5 , we obtain a solution with three regular equivalence classes: (1) v_1 , v_2 , and v_3 ; (2) v_4 and v_5 ; and (3) v_6 , v_7 , and v_8 . In the adjacency matrix with gray lines separating classes, we can easily check that each block is either empty or regular.



9. Answer d is correct because all entries are situated at one side of the diagonal, which means that vertices consistently choose up (or down). It is clearly not a structure of cohesive subgroups or a core-periphery structure because the blocks on the diagonal are empty.

Random Graph Models

13.1 Introduction

The main purpose of social network analysis is detecting and interpreting patterns of social ties among actors (Chapter 1). A pattern of social ties is meaningful if it expresses choices by social actors or the impact of the social system on actors' behavior and attitudes. Until now, we have implicitly assumed that the observed network expresses choices or social constraint, although we have pointed out that our behavioral interpretations should be checked by comparing them with other indicators – for example, see the discussion on structural and social prestige in Chapter 9.

In the current chapter, we accept the idea that at least part of the structure of the observed network is random. As a consequence, we should not assume that every pattern found in a network is meaningful. Statistical inference should tell us whether a network characteristic is random. We do not use statistical inference in the classic sense, assuming that the observed network is a random sample from a larger network (design-based inference). For some basic network properties, statistical inference based on a random sample is possible, but that is not what we pursue here. Instead, we present statistical network models that tell us which network characteristics to expect if the lines are assigned to pairs of vertices according to a random process (model-based inference). This approach assumes that network structure could have been different; for example, the line between actors v and u (Figure 128, network C) could have been replaced by a line between v and w (Figure 128, network D), but not every network structure is necessarily equally probable.

Suppose that we have a transitivity hypothesis for the friendship relation: People are more likely to make friends with their friends' friends than with other people. If person v is a friend of person z and the latter is befriended by persons u and w , then v is more likely to be a friend of u and

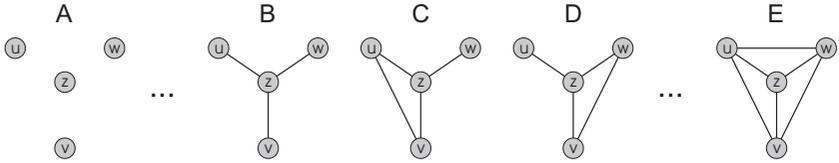


Figure 128. Random versions of a small friendship network.

w than a person who is not befriended by person z . In social theory, we do not usually formulate deterministic hypotheses – for example, we do not think it is realistic to assume that everyone is a friend of all his/her friends’ friends. If we would, all people linked by paths of friendship ties would also be direct friends, which would raise a person’s number of friends to an incredibly high value for a larger network.

We prefer to think of tendencies or probabilities. If we encounter a network like C in Figure 128, perhaps v has not met w yet or v currently thinks s/he has enough friends. There can be many different and ad hoc reasons that the observed network contains a line between v and u but not between v and w . We do not care about these ad hoc reasons because we consider their effects random or noise. We are interested in systematic effects on line formation, such as transitivity, which could also have produced a different network, for example, containing (also) a line between v and w or between u and w .

As a consequence of randomness, many networks are possible and could have been observed. In principle but usually not in practice, we could list all networks that are possible provided that we formulate conditions such as the number of vertices in the network. Figure 128 shows some of the simple undirected networks with four vertices that could have occurred, ranging from the empty network (A) to the complete network (E). Note that networks containing only intransitive triples may also occur (e.g., Network B in Figure 128), but if our transitivity hypothesis is true, they should have a lower probability to appear.

In addition to the set of all possible networks, then, we need the probabilities for each of these networks to occur. The probabilities associated with all possible networks constitute a probability distribution. The probability distributions that we will encounter belong to families, that is, sets of probability distributions that resemble one another. Within each family, one probability distribution differs from the other on one or more characteristics, which are called parameters, such as the average probability of a line to occur within a pair.

In the transitivity example, we might assume that each line has the same baseline probability to occur, but every transitive triple created by a line, for example, the transitive triple v – z – u , raises the probability by a fixed

amount. In this case, we have a probability distribution with two parameters: a general baseline probability and a transitivity bonus probability on top of that. If we define the collection of possible networks and an accompanying probability distribution in a mathematical way, we have a statistical model for the network, which describes the random process that we assume to have shaped the network.

A statistical network model is a mathematical description of a collection of possible networks and a probability distribution on this set.

In this chapter, we present statistical models for overall network structure. This approach is mainly interested in global characteristics of networks and it specifies random processes for generating networks with typical structural features such as a particular degree of clustering or a characteristic average path distance. Section 13.3 presents the most popular network models: classic Bernoulli and conditional uniform models, small-world models, and preferential attachment or power-law, scale-free models. We will use these models to construct confidence intervals for network properties in Section 13.4.

There is another kind of statistical network model focusing on local network structure rather than overall network structure. These models test hypotheses on tie formation: how do actors adapt their lines to other lines in which they or their peers are involved? The models, notably exponential random graph models (ERGM) for cross-sectional data and continuous-time Markov process models for panel data (e.g., as implemented in SIENA software), offer statistical tests on network data that are comparable to hypothesis tests on attribute data that are the mainstay of empirical research in the social sciences. Their discussion, however, deserves a separate book and their application requires other software than Pajek; we refer the reader to the Further Reading section.

13.2 Example

The rise of social media such as blogs on the Internet offers new opportunities for the analysis of large social networks. Owing to the formalized links between actors – for example, hyperlinks between blogs – it is relatively easy to extract network data from social media in large quantities. As a consequence, we can now study interactions at a scale unimaginable to the original sociometrists (see Chapter 1).

This chapter's example is a network of hyperlinks among 1,490 political blogs in February 2005 discussing the 2004 election in the United States, compiled by L. A. Adamic and N. Glance. The network contains

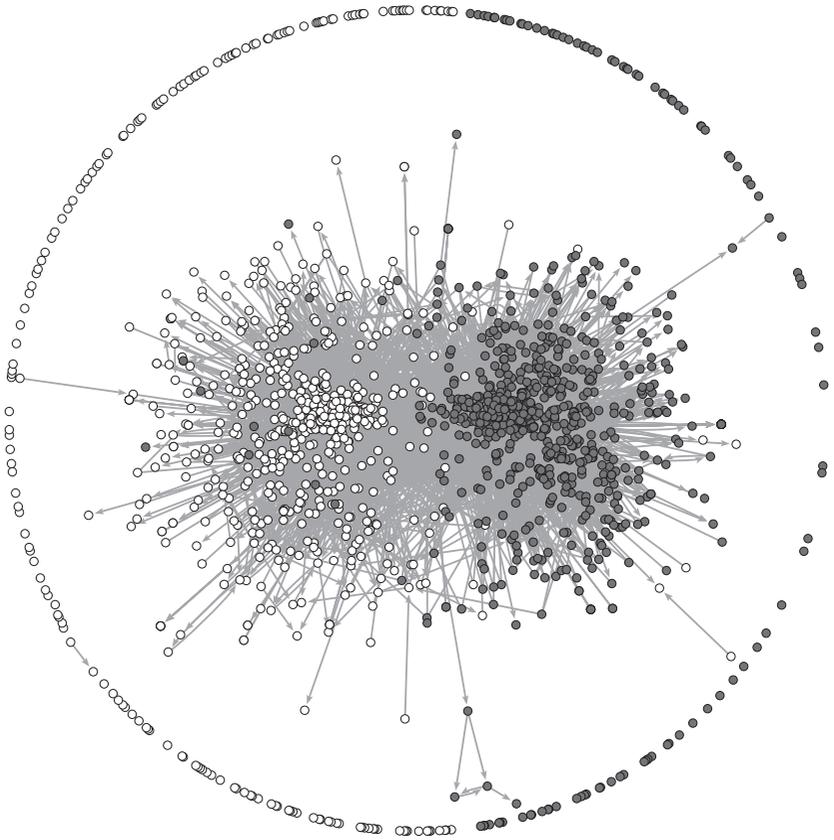


Figure 129. Political blogosphere, United States, February 8, 2005.

a large collection of political blogs with a liberal (Democrats) or conservative (Republican) signature. An arc between two blogs represents a reference in a blogroll or in a post on the blog's front page on February 8, 2005. There are some multiple lines for multiple references to the same blog and a negligible number of loops for self-references.

Figure 129 shows the overall structure of the blogs network, optimized with the Fruchterman–Reingold algorithm. White vertices represent blogs with a liberal leaning; dark gray vertices represent conservative blogs. The majority of blogs are linked into a single component displaying a clearly partisan structure (liberals on the left, conservatives on the right), which is found more often in political blogs networks. As we have learned in Chapter 6, we can consider the partition into liberals and conservatives as a dummy variable and use it for computing the assortativity coefficient. As expected we get high value of this index (0.81) what is in line with our visual perception of the partisan structure. The External–Internal (E–I)

Index, which is -0.81 , tells us the same story. Remember that in case of E-I Index high negative value means that most of the lines connect vertices inside clusters.

13.3 Modeling Overall Network Structure

Several scientific disciplines are interested in the overall structure of random networks, including mathematics, physics, and the social sciences. As a consequence, the aims of studying random networks vary. The mathematician wants to prove that random networks generated in a particular way have certain overall network characteristics such as one large component or a particular value for its diameter. In contrast, the physicist is looking for general laws or constants for large sets of observed networks. In this section, we present models from the point of view of the social scientist interested in the following two research questions:

1. Does my social network have the characteristics of a particular type of random network?
2. If so, do other characteristics of my network appear more or less often than expected by chance in this type of random network?

The first research question is of interest to a social scientist because if the observed network resembles a particular type of random network, the random process generating the random networks may also apply to the observed network. The random process can usually be interpreted as a behavioral tendency, so a fitting random network model may tell something about the behavior of the social actors in the network. Note, however, that the behavioral interpretation was not always intended or implied in the original use of the model. Sections 13.3.1 through 13.3.3 present the most popular random network models.

The second research question boils down to statistical inference: If I can assume that a random network model fits my observed network, the significance of network characteristics can be tested against that model. Section 13.4 uses Monte Carlo simulation to produce a sampling distribution that can be used for statistical tests and for the construction of confidence intervals for indices of overall network structure. Note that this technique can also be used to answer the first research question, that is, for deciding which random network model fits the observed network best.

Finally, a note on terminology. In this chapter, we model only the lines in the network. Additional information, such as vertex and line attributes, is irrelevant to the random network models, so we are actually dealing with graphs rather than networks. For this reason, they are called random graph models. From now on, *network* refers to an observed network, that

is, a network constructed from collected data, whereas we use (random) graph for a randomly drawn network.

13.3.1 Classic Uniform Models

The simplest random graph model considers all graphs with the same number of vertices and the same number of lines, loops, and multiple lines are not allowed, assuming that every graph has the same probability to occur. This model is commonly referred to as the Erdős–Rényi random graph model because the mathematicians P. Erdős and A. Rényi proved several important theorems on this type of random graph.

The most common implementation of this model assigns a line to each pair of vertices independently, with a fixed probability. Independence means that the probability that a pair of vertices is connected by a line is independent from the presence or absence of lines among other pairs. Each pair of vertices can be regarded as a random binary variable, taking the value 1 (line present) or 0 (line absent). The random graph is equivalent to a series of independent random binary variables because each vertex pair is assigned a line with the same probability and independent of all other pairs. Such a series is called a Bernoulli process and for this reason the model is also known as the Bernoulli random graph model.

The original Erdős–Rényi model fixes the exact number of lines, so this characteristic does not vary among random graphs; it is a condition just like the number of vertices. In contrast, the Bernoulli random graph may contain slightly more or fewer lines due to the random process. The number of lines and consequently network density is a parameter that could be estimated if one would have a substantive hypothesis expecting a particular density in a network. Note that density defines average degree, which is simply density times the number of vertices minus one if there are no loops and no multiple lines, so the average degree can also be used as the parameter or condition for the model.

What behavioral hypotheses does the random process imply? All actors are equally likely to be involved in lines – in other words, they are likely to be involved in the same number of lines, and lines occur at random among pairs of actors: Actors don't care with whom they are linked. From a systems perspective, the system distributes lines at random among its members. These hypotheses are not very plausible for social networks, which we expect to have more structure, for example, clustering or centralization. The model should be seen as an absolute baseline and we should not be surprised if the observed network has different structural features; perhaps we should rather be surprised if the model does fit an observed network.

The overall structure of Bernoulli random graphs has some interesting expected characteristics, that is, characteristics that do not necessarily

apply to each generated Bernoulli random graph but that appear if we average over a large number of random graphs in a collection. The characteristics are more likely to occur if a generated random graph is larger.

The most surprising characteristic of Bernoulli graphs concerns the size of the weak components in relation to average degree. An average degree greater than 1 produces with high probability a graph containing one large component while all other components are about equally small. The size of the large component grows with both the size of the graph and the average degree; with average degree 1.5, it is already expected to contain more than 50 percent of all vertices. In a large graph, the main component is also large, for which reason it is sometimes called the giant component. In contrast, average degree below unity is expected to create random graphs containing only small components. In addition, the diameter of the graph is relatively small, somewhere in the order of magnitude of the logarithm of the number of vertices (n) divided by the logarithm of the average degree (c) [formula: $\ln(n)/\ln(c)$].

Giant components and relatively small diameters are regularly encountered in social networks. The expected degree distribution of Bernoulli random graphs, which is a Poisson distribution (the model is also called a Poisson random graph model), is less common in social networks, and notably the clustering in Bernoulli random graphs is much lower than in social networks, especially if the network is not small. Clustering in a graph is measured with a clustering coefficient showing the transitivity of the graph: the proportion of all two-paths in the network that are closed. This can be interpreted as the average probability that two vertices with a common neighbor are also directly linked. In a Bernoulli random graph, the expected proportion of transitive triples over connected triples is the average degree of vertices (c) divided by the number of vertices (n) minus 1 [formula: $c/(n - 1)$], which tends toward 0 in larger sparse networks.

The *clustering coefficient* or transitivity of a network is the proportion of all two-paths in the network that are closed.

Taking into account only the number of lines, the Bernoulli random graph model ignores any additional conditions that the observed network may have had to satisfy. Data collection procedures such as a fixed number of choices, for example, name your two best friends, add important restrictions to the resulting network – in other words, constant outdegree. In this case, the random graphs should be forced to have the same outdegree distribution as the observed network because the degree distribution of vertices is important to the structure of a graph. Models extending the Bernoulli random graph model with this type of conditions are called generalized random graph models or conditional uniform random graph

models. The associated probability distribution is conditional uniform: Every graph or line has the same (= uniform) probability to occur on the condition that the resulting graph has, for example, the predefined degree distribution.

Technically, it is possible to condition on degree distribution even if the data collection design does not pose constraints. This would imply that some actors are just more likely to engage in lines (degree), initiate arcs (outdegree), or be targets of arcs (indegree). The model then assumes that given these individual capacities, actors assign their lines at random. Note that these capacities are conditions, not parameters of the model, so they cannot be estimated or tested; it is assumed that the network could only have this outdegree distribution. If the degree distribution is copied from the observed network, the individual capacities are implicitly assumed to be measured error-free, which is questionable.

In the social network analysis literature, important work has been done on probability distributions for directed graphs conditioning on the number of mutual, asymmetric, and null dyads, that is, pairs with a reciprocated arc, a single arc, or no arc at all. The objective here is to test and estimate effects among triples such as transitivity. This model is a forerunner to the exponential random graph models for cross-sectional network data, mentioned at the end of Section 13.1. Simple undirected Bernoulli graphs conditioned on degree distribution and simple directed Bernoulli graphs also have known characteristics but they are more complicated and more difficult to calculate, so we do not discuss them here.

Application

Simple undirected Bernoulli graphs have characteristic component size, diameter, degree distribution, and clustering. If we want to determine whether a Bernoulli Random Graph Model fits an observed network, we must first check whether it displays these characteristics. The political blogs network, the example in this chapter, is stored in the Pajek project file `Political_blogs.paj`. This file contains the original directed network with arcs representing links from one blog to another ("`Political blogosphere Feb. 2005`".net), but we will use the symmetrized version without loops (`Blogosphere_undirected.net`).

Network>
Create
Partition>
Components>
Weak

Component sizes can be determined with the command *Network> Create Partition> Components> Weak* as discussed in Section 3.4. This command reports the absolute and relative size of the largest component in the Report screen. The blog network has one large component containing 1,222 vertices (82 percent) while all other components are either isolates or pairs of vertices. Clearly, this network contains a giant component. The average degree of the blog network being much higher than 1 (it is 22.4; use the *Network> Info> General* command), the existence of a giant component is predicted by the Bernoulli random graph model.

Network>
Info> General

The network's diameter is reported in the Report screen by the *Network> Create New Network> SubNetwork with Paths> Info on Diameter* command. Note that execution of the command may take some time for large networks. The blog network's diameter turns out to be 8 while the model predicts a diameter of about $\ln(n)/\ln(c) = \ln(1490)/\ln(22.4) = 2.4$. There clearly is a discrepancy here, but with the observed result being only about three times the expected result, one could maintain that they are of the same order of magnitude, assuming that a factor of 10 defines different magnitudes.

The *degree sequence* of the network – that is, a list of the degree of all vertices sorted by the vertex's sequential number in the network – is obtained with the *Network> Create Partition> Degree> All* command. The degree distribution, namely the counts of each degree in the network, can be inspected with the *Partition> Info* command. Degree frequencies can also be stored as a new partition with the *Partition> Count, Min-Max Vector* command applied to the degree partition. Note that cluster numbers represent degree in the partition created by this command. The degree distribution is very skewed to the right, so it does not resemble the Poisson distribution expected in the Bernoulli random graph model. We will inspect the degree distribution more closely in Section 13.3.3.

Finally, let us inspect the clustering in the network. The clustering coefficient of the network can be calculated from the clustering coefficients of the vertices in the network. The local clustering coefficient of a vertex is the proportion of the pairs among its neighbors that are directly linked. Note that the local clustering coefficient is meaningful only for vertices with at least two neighbors in the network. A weighted average of the local clustering coefficients over all vertices with minimum degree 2 yields the clustering coefficient or transitivity of the network. An unweighted average, known as the Watts–Strogatz clustering coefficient, is also being used but it does not yield the precise proportion of closed two-paths.

In Pajek, the *Network> Create Vector> Clustering Coefficients> CC1* command outputs two vectors of local clustering coefficients for an undirected network without multiple lines, the first of which (labeled CC1 instead of CC1') contains the desired local clustering coefficients. Note that the clustering coefficients of vertices with degree below 2 are set to 99999998, which is one of Pajek's missing values. In addition, a partition is created containing the number of lines among a vertex's neighbors, which is set to 0 for vertices with less than two neighbors. Finally, the Report screen shows the Watts–Strogatz clustering coefficient and the transitivity or clustering coefficient of the network. The clustering coefficient (transitivity) is 0.226 for the blog network, which is an order of magnitude higher than the value $c/(n - 1) = 22.4/(1490 - 1) = 0.015$ predicted by the Bernoulli random graph model. As noted earlier, social networks are usually more clustered than Bernoulli random graphs.

*Network>
Create New
Network>
SubNetwork
with Paths>
Info on
Diameter*

*Network>
Create
Partition>
Degree> All*

*Partition>
Count,
Min-Max
Vector*

Partition> Info

*Network>
Create Vector>
Clustering
Coefficients>
CC1*

Network>
Create Random
Network>
Total No. of
Arcs

It is possible to generate a random graph according to several classic random graph models in Pajek. A random Erdős–Rényi graph with a fixed number of lines can be drawn with the *Network> Create Random Network> Total No. of Arcs* command. Dialog boxes ask for the number of vertices, the number of arcs in the graph, and whether it should exclude multiple lines. Note that the command can only produce directed graphs; symmetrizing the random graphs may reduce the number of lines because bidirectional arcs are replaced by one edge. Therefore, the command is not suited for generating an undirected random graph.

Network>
Create Random
Network>
Bernoulli/
Poisson

The much more widely used Bernoulli random graph can be generated in the *Network> Create Random Network> Bernoulli/Poisson* submenu. Both undirected and directed random graphs can be created as well as bipartite and two-mode random graphs, the last two being basically the same. The command will ask for the total number of vertices and the average degree. For a bipartite or two-mode network, the user must also supply the number of vertices in the first mode.

Network>
Create Random
Network>
Vertices Output
Degree

Conditions on the degree distribution can be set in different ways. A quite loose way is a specification of the range of the degree or outdegree of vertices in the random graph. The command *Network> Create Random Network> Vertices Output Degree* creates a random directed network with output degrees more or less uniformly spread over the specified range. The exact degree distribution of the random graph can be set with the commands in the *Partition> Make Network> Random Network* submenu and the command *Partitions> Make Random Network*. The first set of commands requires one partition, which it uses to fix the degree, input degree, or output degree sequence for the random graph. If, for example, the first vertex in the partition has cluster number 5, the first vertex in the random graph will have (in/output) degree 5. The partition also determines the number of vertices in the random graph, so the user need not specify this property. The second command requires two partitions using the partition in the first *Partition* drop-down menu as the input degree sequence and the one in the second drop-down menu for output degree sequence. Note that it is not always possible to create a network satisfying the conditions on the degree sequence if the partitions used in the commands are not derived from the degree sequence of an actual network. If this happens, Pajek will report it.

Partition>
Make
Network>
Random
Network

Partitions>
Make Random
Network

13.3.2 Small-World Models

An obvious shortcoming to the Bernoulli random graph model is its low and network-size-dependent clustering. The models presented in the current section are designed to remedy this problem, yielding random graphs

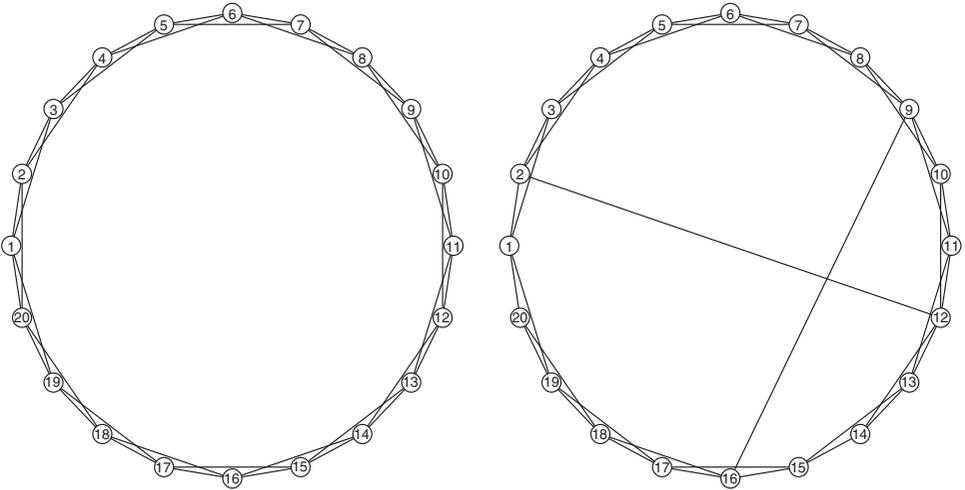


Figure 130. Small-world random graph generation: Ring of local lines (left) and rewired lines (right).

of any size with clustering coefficients in the range usually found in social networks, roughly put between 0.05 and 0.50.

The solution to the clustering problem is relatively simple: just link each vertex to a number of its nearest neighbors. The original small-world model puts all vertices on a circle and connects each vertex to a fixed number of its neighbors in a literal, spatial sense: the vertices nearest in the plane. If the number of connected neighbors exceeds 2, triangles appear because each vertex is linked to its neighbor and its neighbor's neighbor (see Figure 130, left). The expected clustering coefficient (network transitivity) is determined only by the number of neighbors each vertex is linked to at each side (r) and it is easy to calculate: $(3r - 3) / (4r - 2)$. If the number of neighbors on either side is set to 1, the clustering coefficient reaches its minimum 0 (assuming that the clustering coefficient is meaningless for lower values of r), whereas the coefficient tends toward 0.75 for a very large number of neighbors.

Here, the implicit behavioral hypothesis is that actors tend to have lines with their neighbors' neighbors. In technical terms, they have a preference for transitive closure. Substantively, transitivity can be motivated by the fact that social actors are near each other geographically (I know the people living next door to me, they know their neighbors, whom I know as well), or organizationally (I know people and the people they know in my organization), or in another sense, for example, because actors share interests (some people are linked because they share an interest in social

network analysis). In other words, transitivity implies that context matters to the formation and preservation of social ties.

This approach, however, raises a new problem: A large graph containing only local lines has an average path distance much higher than the ones encountered in social networks, whereas the small-world phenomenon argues that even in the network containing the entire world population, people are acquainted in a maximum of six steps (see Chapter 1, Section 1.3). This problem is solved by replacing one endpoint of a small proportion of the local lines by a random vertex (Figure 130, right), which suffices to obtain a graph with low average path length. This replacement is called rewiring, and rewiring as little as 1 to 10 percent of the local lines suffices to obtain the small-world phenomenon of low average path distance.

A low proportion of rewired lines does not change the density and average degree of the graph. It hardly changes its clustering, so the high clustering characteristic of social networks is preserved. Of course, when you would rewire all lines, the clustering disappears and tends toward 0 for large graphs as in the Bernoulli random graph model. Because rewiring is random, there is not a clear behavioral interpretation; some actors are accidentally tied to actors outside their local neighborhood.

The statistical model for small-world random graphs fixes the number of vertices and the number of nearby neighbors on the ring to which each vertex is linked. This effectively fixes the number of lines in the original small-world model. The probabilistic part of the model only concerns the rewiring of lines, selecting a line at random, and selecting a vertex at random to which the line is rewired. Each line and each vertex has equal probability of being selected, so the procedure can be interpreted as two Bernoulli processes with one parameter setting the probability that a line is rewired. Note that the probability that a vertex is selected to receive the rewired line is fixed by the size of the network because all vertices have equal probability.

In a small-world random graph, average path distance among vertices is short but it is difficult to say how short. Average path distance is known to increase logarithmically with the number of vertices, but that does not tell us much if we have just one network. In addition, the clustering coefficient is relatively high, so if we would like to quantify the small-world character of a social network, we may divide the average path length by the clustering coefficient. The lower this value, the more a network contains low average path length and high clustering.

The original small-world model is named after the two scientists that proposed it: the Watts-Strogatz model. Alternative small-world models have been proposed. One alternative model replaces both vertices of the selected line at random: You take away a line and replace it by a random line, as in the Bernoulli random graph. Replacement of all lines would

then produce a classic Bernoulli random graph, so the Bernoulli random graph model is a special case of this small-world model.

Another alternative model just adds random lines to the graph of local lines, that is, without changing the local lines. This boils down to combining a graph of local lines with a Bernoulli random graph. This model always preserves the clustering inherent in the local lines, so the minimum clustering is usually higher than in the other models, provided of course that the average degree in the local graph exceeds two and the probability of a random line is higher than zero. Finally, models have been proposed starting with vertices arranged on a lattice – informally: a grid in which vertices linked only to their nearest neighbors – rather than a circle.

Application

The clustering coefficient of the political blogs network is 0.226 (see previous section), which is well in the range expected for small-world random graphs. The average degree in the blogs network is 22.4, so we can compare this with a small-world random graph with vertices linked to the eleven nearest neighbors (r), which has about the same average degree. The expected value of the clustering coefficient for this random graph is $(3r - 3) / (4r - 2) = (33 - 3) / (44 - 2) = .71$ in the limit of no rewiring. The observed value is quite a bit lower, the reason probably being that vertex degree is not more or less equal for all vertices as assumed by the small-world random model; instead, it is highly skewed.

The average path length or average distance among vertices in the network is calculated by the *Network> Create Vector> Distribution of Distances** command. Take care: This command may take much time for large networks. It creates a special vector containing the distance distribution: The entry number in the vector represents the distance, and the associated vector value is the frequency of this distance in the network. In the Report screen, the command outputs the average distance, which is 2.74 for the blogs network, the diameter (8 in the example network), and the number of unreachable pairs. The ratio of average path length to the clustering coefficient is $2.74 / .226 = 12.12$. Is this low, indicating a small-world network? To answer this question, we should compare it to ratios for other networks.

The *Small World* command in the *Network> Create Random Network* menu creates an undirected random graph according to the Watts–Strogatz small-world model. In dialog box, first enter the number of vertices in the random graph, then specify the number of neighbors a vertex is linked to at each side – for example, entering 3 in this dialog box creates a ring with vertices linked to the three closest neighbors at each side, so average vertex degree is 6. Finally, enter a probability between 0 and 1 that a line is going to be rewired, that is, randomly receives one new endpoint. Low values for the rewiring/adding probability – in the order

```
Network>
Create Vector>
Clustering
Coefficients>
CC1
```

```
Network>
Create Vector>
Distribution of
Distances*
```

```
Network>
Create Random
Network>
Small World
```

of 0.01 to 0.10 – suffice for obtaining relatively short paths in the random graph.

Exercise I

Determine the effect of rewiring probability on the average path length of small-world random graphs with 1,000 vertices, linking vertices to the three nearest neighbors at each side. Use rewiring probabilities 0.001, 0.01, 0.10, and 0.20.

13.3.3 Preferential Attachment Models

There is a problem with both the Bernoulli and small-world models: The degree distribution is not as skewed to the right as in many social networks. Social networks often contain few vertices with very high degree along with many vertices with low degree. Preferential attachment models solve this problem by simply assuming that vertices prefer to link to vertices with higher degree. This is a network variant of popular sayings such as “the rich get richer” or “success breeds success.” The behavioral hypothesis is that actors prefer popular peers. In social networks, preferential attachment may represent contact probabilities – if many people know a person, I am more likely to be informed about this person – or it may capture social mechanisms in which selections by others are interpreted as an indication of an item or actor’s quality, which induces people to select it themselves, thus enhancing its popularity.

Preferential attachment models are network growth models, constructing a random graph by adding vertices and one or more lines one at a time. In the well-known model proposed by A. L. Barabási and R. Albert for undirected networks, each new line is added between the new vertex and a randomly selected vertex that was added previously to the network. No multiple lines are allowed. With each new vertex, a fixed number of lines are added, so the initial network must contain at least this number of vertices to avoid multiple lines.

The statistical model for preferential attachment fixes the number of vertices and lines of the random graph directly or indirectly by way of the number of repetitions and the number of new lines added in each repetition, assuming that each repetition adds one new vertex. The probabilistic part involves the selection of vertices for the new lines. In the Barabási–Albert model, the probability that a vertex is selected as an endpoint of the new line is proportional to its degree. This is the part that expresses preferential attachment.

Note that this model assumes that vertices and links remain in the network during the entire network growth process. If we want to use this model for a social network that has grown historically, the model is more

likely to fit if this condition also holds for the observed network. This would be the case in the political blogs network example if blogs tend to be preserved so they can always be referenced and if previous references are preserved. For certain types of networks, this clearly is not the case – for example, a friendship network measured at a particular moment is unlikely to preserve all past friendships. Abandoned friendships do not contribute any longer to a person’s popularity.

A highly skewed degree distribution is the primary characteristic of preferential attachment models. The right-hand tail of the degree distribution follows a power law: The relative frequency (proportion) of a particular degree (p_k) is more or less equal to a negative power of the degree (k) [formula: $p_k = Ck^{-\alpha}$ where C is a normalizing constant.] For this reason, the model is also known as the power-law model, but note that the power law applies only to the right tail of the distribution, that is, the higher degree values. Interest usually focuses on the (absolute) value of the exponent alpha, which is expected to be 3 for very large graphs in the Barabási–Albert model.

Power-law distributions are scale-free: At different magnitudes of degree, the frequency distribution has the same shape. More specifically, if we compare the proportion or probability of a degree to the proportion of the degree multiplied by a fixed factor, we obtain the same ratio, whether we compare rather low degrees, such as 10 and 20, or much higher values, such as 1,000 and 2,000. In other words, the scale or order of magnitude does not matter. The scale-free characteristic, however, is only meaningful for networks with degrees spread over several orders of magnitude, which can only occur in large networks.

The Barabási–Albert model turned out to be a special case of an older model introduced by D. de Solla Price for directed citation networks. In citation networks, arcs point back in time toward older publications. Preferential attachment concerns the indegree of vertices, highly cited texts are more likely to receive new citations, so the indegree distribution tends to have a power law tail in large citation networks. Network growth is simulated as in the Barabási–Albert model but if the probability of receiving an arc (being cited) is only proportional to the indegree of vertices, new vertices will never receive arcs because their initial indegree is 0. Therefore, De Solla Price added a constant baseline probability of receiving a line, each vertex in the graph is assigned a fixed number of virtual incoming arcs. This number does not have to be an integer but it must be greater than 0.

In De Solla Price’s model, the random assignment of a head to a new arc can be conceptualized as a two-step process: In the first step, one chooses at random whether or not to take into account vertex indegree, and in the second step, one selects a vertex at random according to the result of the

first choice, either with preferential attachment or with constant (uniform) probability for all vertices. The second step is the same as specified earlier for the Barabási–Albert model and the Bernoulli model, respectively. In the first step, the probability to select the endpoint proportional to its indegree (p_c) is equal to the proportion of the average number of new lines (c) in each step over the number of new (c) and virtual lines (a): $p_c = c / (c + a)$. Likewise, the probability of uniform vertex selection (p_a) is the proportion of virtual lines: $p_a = a / (c + a)$. If, for example, the average number of new lines added with each new vertex (c) equals 4 and the constant (a) is set to 1 virtual line, the probability of selecting a head with indegree weighting is 0.8 versus 0.2 for selecting a vertex ignoring indegree. In other words, on average, 4 out of each 5 draws of line heads take into account the indegree of vertices.

The constant (a) and average degree (c) in the graph define the expected exponent alpha of the power-law tail of the degree distribution in a simple manner: $\alpha = 2 + a/c$. If the constant is less than the average degree, preferential attachment has a stronger effect on the random process than uniform or “equal opportunity” selection. Then, the quotient is less than 1, so the expected power-law exponent is between 2 and 3. Empirical power-law networks tend to have exponents in this range. Note again that the expectation is valid only for large networks.

Alternatively, if the constant is equal to the average degree of the graph, meaning that draws according to preferential attachment are just as likely as draws ignoring indegree, the quotient is exactly 1 and the exponent is expected to be 3. This is the value predicted in the Barabási–Albert model, which is not a coincidence. In the Barabási–Albert model for undirected graphs, each new vertex is linked to a constant number of new lines, which are the virtual lines (a) in the model of De Solla Price. At the same time, the other endpoint of these lines is selected with probabilities weighted by vertex degree, so we also have the number of new lines added in each step as the number of endpoints selected with preferential attachment (c). Therefore, the Barabási–Albert model is equivalent to De Solla Price’s model with equal probabilities (0.5) for uniform and preferential attachment selection of vertices for new lines.

The main characteristic of preferential attachment random graphs is the shape of the degree distribution, which follows a power law especially for higher degree. If a power-law distribution is plotted on a log-log scale, that is, with the logarithm of degree on the horizontal axis and the logarithm of degree frequency on the vertical axis, the dots fall along a straight line, at least for higher degree values. Figure 131 (left panel) shows the log-log plot of the degree frequency distribution for the political blogs network. The dots seem to follow a straight descending line but they fan out for larger degree values, which means that there is quite some variation in the frequencies of high degrees. This usually occurs because the highest

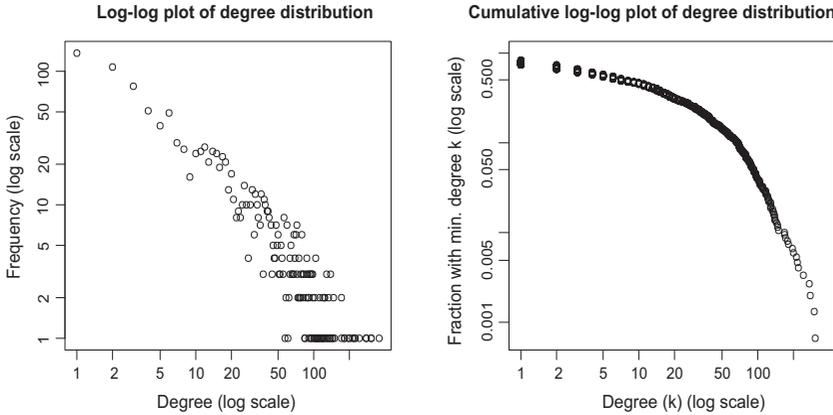


Figure 131. Log-log degree distributions of the blogs network, absolute frequencies (left) and cumulative proportions (right).

degrees occur sporadically, so we may accidentally find three vertices with degree 128 whereas we find only one with degree 129.

To get rid of this noise in the tail, a cumulative plot on a log-log scale is preferred (Figure 131 [right panel]). Here, the proportion of all vertices having a particular degree or higher is plotted on the vertical axis. Now we see more clearly a straight line in the right-hand tail of the distribution, but we can also conclude that this line applies only to the range of degrees starting at about 50. With degrees below 50, the plot bends away from a straight line, perhaps to form a new line in the range of 1 to 10 degrees.

If we would like to determine the power-law exponent α for this network, we should restrict it to the 50–300 interval, which probably includes too few orders of magnitude (300 is only 6 times 50) to speak of a scale-free distribution. Fitting a power law to this range is not something we recommend to do, but if we would do so, we find a value of α of 2.85, which is quite close to the value predicted by the Barabási–Albert model.

Preferential attachment random graphs are rather compact in terms of connectedness and average distance among vertices due to the vertices with high (in)degree, which link vertices into one component and offer short paths between most pairs of vertices. In these respects, the preferential attachment models are similar to the classic models and the small-world models. In contrast to the latter, however, clustering is relatively low.

Several alternative preferential attachment models have been proposed as well as different procedures for generating random graphs from these models, including models combining preferential attachment with

rewiring as in small-world models. All models tend to have a degree distribution with a power-law tail, but other characteristics may depend very much on the model and settings. Some models, for example, yield random graphs that are necessarily connected or connected with very high probability. For these models, the existence of a giant component should not be presented as an interesting result.

Application

Random graphs of the preferential attachment model are primarily distinguished by a highly skewed degree distribution following a negative power law. A degree partition represents the degree sequence of the network, but Pajek has no facilities for plotting frequency distributions, so we will show how to export the degree sequence to other software and create plots there. We will use the free software R, so you must first download and install the R package if you have not yet done so and ensure that Pajek can locate R (see the Application part to Section 5.5). We use the Windows version of R; the user interface may be different under other operating systems.

Network>
Create Vector>
Centrality>
Degree> All

Tools> R>
Send to R>
Current Vector

In Pajek, create a degree vector of the undirected blogs network with the *Network> Create Vector> Centrality> Degree> All* command. Export it to R with the *Tools> R> Send to R> Current Vector* command. If Pajek has been linked to R, the R software should now start and load the vector containing the degree distribution.

R: File> Open
script

R: Edit> Run
all

The R Console window shows, among other things, the name of the vector, which is the letter *v* followed by a sequential number, for example, *v2*. With R's *File>Open script* command, open the script file *Scale_free.R* which is available from the book's website (<http://mrvar.fdv.uni-lj.si/pajek/>). In the script file, which is displayed in a separate window, make sure that under the line starting with *# USER: ASSIGN THE RIGHT PAJEK VECTOR TO x*, the correct vector name is specified in the *x <- v2* statement. In addition, the *{igraph}* package must be downloaded and installed, which is explained in the script file. Finally, use the *Edit> Run all* command in R to execute the script. Note that this command is available only if the Script window is selected.

If the script has run successfully, the two log-log plots of Figure 131 should be shown in an R Graphics window. If the graphics window is selected, a button with a picture of a photo camera is shown on the main menu. Pressing this button copies the plots into computer memory, so it can be pasted in other software.

R: {igraph}
power.law.fit()

R: Edit> Run
line or selection

The last command in the script, *power.law.fit(x, xmin = 50)*, estimates a maximum likelihood value of the power-law exponent α taking into account only frequencies of degrees 50 and up. α for this part of the degree distribution of the blogs network is 2.85. You can adjust the minimum degree by changing the number behind the *xmin* parameter

in the script file and rerun this command with the *Edit> Run line or selection* command.

Preferential attachment random graphs can be generated with the *Network> Create Random Network> Scale Free* commands in Pajek. On execution of one of these commands, the user has to specify several settings, starting with the total number of vertices and the maximum number of lines. We recommend leaving the maximum number of lines unconstrained. Then the average degree of vertices must be specified, which is the number of lines added with each new vertex. If you want to compare an observed network to a random graph, use the network's average degree here. The dialog box also asks for the network to start with, the size of the initial network, and the probability of lines within the network. We recommend setting the size of the initial network slightly higher than the average degree. The probability of lines within the initial network is not very consequential, but you should enter a high value if you want to ensure that the random graph is connected.

*Network>
Create Random
Network>
Scale Free*

The last entries in the dialog box are perhaps the most crucial because they fix the probabilities that a new line is added with uniform or degree-dependent probability. The entry captured Alpha (not to be confused with the power-law exponent alpha [in lowercase]) requests the probability of line assignment proportional to vertex indegree. If the network is undirected, Pajek assumes that the same probability applies to weighting the outdegree, because each edge is treated as a bidirectional arc, so the value of Alpha cannot be larger than 0.5. For a directed network, the probability of weighting the outdegree (called Beta in Pajek) must be specified in a separate entry. Based on the probabilities of weighting indegree and outdegree, Pajek calculates the probability of uniform selection of the line's endpoint, which is simply 1 minus Alpha and Beta.

De Solla Price's model for directed networks uses both a fixed probability and a probability proportional to the indegree of the vertex. For this model, a random graph can be generated with the *Network> Create Random Network> Scale Free> Directed* command. De Solla Price's model does not take into account the outdegree of vertices, so Beta in Pajek should be set to 0. Alpha can be set within the range from 0 to 1, but settings between 0 and 0.5 are most likely to generate random graphs that resemble scale-free social networks. Alpha, which is the probability that vertex indegree is weighted (p_c), is related to average degree (c) and the number of "virtual" incoming arcs (a) in the following way: $p_c = c/(c + a) \Leftrightarrow a = c(1 - p_c)/p_c$. In a random graph with average degree $c = 22$ and a constant $a = 10$ virtual incoming arcs, the required probability of Alpha in Pajek is $p_c = c/(c + a) = 22/(22 + 10) = .6875$. Vice versa, if the probability is set to 0.25 and average degree is 22, the constant number of virtual incoming arcs (a) is $a = c(1 - p_c)/p_c = 22(1 - .25)/.25 = 66$.

*Network>
Create Random
Network>
Scale Free>
Directed*

Network>
Create Random
Network>
Scale Free>
Adding> Free

Note that the *Scale Free* command only produces the De Solla Price and Barabási–Albert models if each new edge starts at the newly added vertex. This requires that the option *Adding>Free* is not selected. If this option is selected, both the tail and head of new arcs will be selected at random and vertices may remain isolated in the resulting random graph, which may or may not be what you want. As argued, the Barabási–Albert model is a special version for undirected networks of De Solla Price’s model, namely where $a = c$. A vertex should be drawn with 0.5 probability proportional to its degree and 0.5 uniformly from all vertices. In the *Scale Free* command, Alpha must then be set to 0.25 for undirected networks because Beta is automatically set to the same value, so their sum is the desired 0.5.

Network>
Create New
Network>
Transform>
Remove>
Multiple Lines

Random graphs generated by the *Scale Free* commands may contain multiple lines. If these are not desired – for example, because the clustering coefficient cannot handle multiple lines – they should be removed with the *Network> Create New Network> Transform> Remove> Multiple Lines* command.

Network>
Create Random
Network>
Extended
Model

The *Network> Create Random Network> Extended Model* command can also be used to create preferential attachment random graphs, albeit only directed ones. This model combines the addition of vertices and lines with rewiring. All adding and rewiring probabilities are degree-dependent, so there is always a preferential attachment aspect to the random graph. The command is quite similar to the *Scale Free* commands, except that the size of the initial network cannot be smaller than the number of lines added in each new step. The user has to specify separate probabilities for adding new lines and rewiring existing lines in each step. The probability of adding a new vertex in each step is set to the difference between 1 and the sum of the first two probabilities.

Note that the probability of adding a new vertex should not be too low; otherwise the network may become too dense and never reach the desired number of vertices. In general, random preferential attachment graph generation may not be successful, coming to a premature end because some of the required properties cannot be attained. Sometimes, repeating the command will produce the desired result. In other cases, average degree settings, probability of adding new vertices, or size of the starting network may have to be increased before a random graph can be successfully generated.

Exercise II

Check whether a random undirected graph (10,000 vertices, average degree 20) generated by Pajek according to the Barabási–Albert model yields the expected value of the power-law exponent alpha, namely 3. If not, what is your explanation? Compare results when the new lines always originate at the newly added vertex to when they are not.

13.4 Monte Carlo Simulation

The preceding section presented several types of models for overall network structure. The models differ with respect to structural features of the random graphs, notably connectedness, degree distribution, clustering, and the diameter or average path distance. We noted, however, that these characteristics are difficult to predict exactly for a graph generated from these models. On the one hand, this is because they are sometimes related to the conditions and parameters in a complex way and known only for infinitely large networks, so we can merely predict the order of magnitude of the characteristic; on the other hand, this is because parameter values naturally vary among different random graphs drawn from the same model. We usually study only one social network, which is not necessarily very large, so we should not expect the observed network to exactly match the typical random graph. How can we determine that a random graph model fits our network?

Our approach is to perform a Monte Carlo simulation, which proceeds as follows. Generate a large number of random graphs using a model with conditions set to reflect the characteristics of the observed network. The generated graphs approximate the sample space from which the observed network would have been drawn if the random process of the model applies to the network. Then calculate a network characteristic of interest for each random graph, for example, average path distance. The relative frequency distribution of this characteristic approximates its sampling distribution. The average value of the sampling distribution is an approximation of the expected value, and the standard deviation approximates the characteristic's standard error. A confidence interval can be constructed using the expected value and either the standard error, if we assume a particular probability distribution (e.g., the normal distribution), or by simply looking at the limits that separate the 90% or 95% observations in the middle from the observations in the tails.

Thus, Monte Carlo simulation offers a benchmark for the features of the observed network, basically telling us how likely it is to find the observed value or a more extreme value in a random graph drawn from the specified model. If this probability is large, we conclude that our network resembles a random graph from this model – at least regarding this property – and the random process associated with the model may tell us something about social tie formation in the observed network. In contrast, a very low probability tells us that there is probably another structuring principle at work in the network because our network is unlikely to be generated by the random graph model.

Let us illustrate this with the political blogs network. The undirected network contains 1,490 vertices; loops and multiple lines are removed.

Table 26. Monte Carlo simulation results: Confidence intervals for the simple undirected blogs network

	Blogs network	Bernoulli		Degree conditional		Small world		Preferential attachment	
		2.5%	97.5%	2.5%	97.5%	2.5%	97.5%	2.5%	97.5%
No. of components	268	1	1	1	1	1	1	96	134
% Largest component	82.0	100	100	100	100	100	100	.91	.94
Diameter	8	4	4	7	9	4	5	7	9
Av. distance	2.74	2.61	2.63	3.29	3.36	2.98	3.01	3.07	3.14
Clustering	0.226	0.017	0.018	0.029	0.031	0.355	0.372	0.095	0.107
Betweenness	0.065	0.002	0.003	0.010	0.021	0.003	0.004	0.038	0.064
Triad census:									
003 ^b	526.6	522.0	522.8	522.8	522.8	526.1	526.1	527.7	529.5
102 ^b	22.49	26.99	27.70	26.61	26.66	23.81	23.82	1.16	1.37
201 ^a	1038.4	464.2	490.3	757.3	767.4	219.6	226.0	9.95	13.79
300 ^a	101.0	2.62	2.94	7.56	8.17	41.39	43.38	1.99	3.01

^a In thousands.^b In millions.

Network density is 0.015 and average degree is 22.4. We must try to reproduce these characteristics in the random graphs that we will generate. We generate 1,000 random graphs with these characteristics for the Bernoulli model, the conditional uniform random graph model with fixed degree, the small-world model, and the preferential attachment model. The degree sequence in the conditional uniform random graph model is constrained to be equal to the degree sequence in the observed network. For the small-world model, we link each vertex to 11 local neighbors on each side to obtain an average degree of 22. We use a relatively high rewiring probability of 0.20, which is rather a random choice, unfortunately, because we cannot deduce a particular rewiring probability from the nature of blog links. In the preferential attachment model, we set the probability of selecting vertices with degree preferences to 0.5, which mimics the Barabási–Albert model, but the selection of both the head and tail of new lines is random to allow for isolates, which also occur in the blogs network.

Overall, the results in Table 26 show that there is very little variation in the number and size of components, diameter, average distance, and clustering among random graphs generated with the same model. This should not come as a surprise because the models are known to have a characteristic value on these aspects of graph structure. We also notice that the blogs network resembles different random graph models for different network properties. All in all, it is difficult to decide for one random graph

model. With respect to diameter, the blogs network is most similar to the general random graph model conditioned on outdegree and the preferential attachment model. Its average distance among vertices is closest to the Bernoulli random graph model, although the observed averaged distance (2.74) is outside the 95% confidence interval (2.61; 2.63) for this model.

The blogs network is much more clustered than all models but it is closest to the small-world model, as we could have expected. The small-world model that we used, however, is too clustered, so more lines must be rewired than the (on average) 20% used here. We could have formulated this as a hypothesis test: Our null hypothesis states that the rewiring probability is 0.20. Assuming this rewiring probability, the clustering coefficient should range between 0.355 and 0.372 with 95% confidence for any network conforming to this model. The blogs network's clustering is clearly much lower, so we reject the null hypothesis.

Finally, if we have a look at the betweenness centralization of the network, the preferential attachment random graph model yields centralization scores that are closest to the centralization of the political blogs network, but centralization is still too low in the random graphs. Perhaps preferential attachment should have a stronger impact than 0.5 to obtain the desired amount of centralization.

Monte Carlo simulation can be used to construct a confidence interval for any overall network property. As an illustration, we also constructed confidence intervals for the frequencies of triad types (triad census). Small network configurations that occur with significantly high frequency are called *network motifs*. Triads with two (201) or three (300) edges appear much more frequently in the blogs network than expected in any of the random graph models used here. This indicates a popularity effect (some blogs are linked to several other blogs that are not linked themselves; triad 201) and a clustering effect (blogs are linked to their network neighbors' neighbors; triad 300). Note that the clustering effect is also apparent from the significantly high clustering coefficient of the observed network. Political proximity among blogs – for example, the Republicans versus the Democrats (Figure 128), is likely to play a role here.

Application

In previous sections, we learned how to generate one random graph for each model. The first step in Monte Carlo simulation boils down to repeating random graph generation many times with the *Macro> Repeat Last Command* command in Pajek. First, create one random graph, then select *Macro> Repeat Last Command*, ensure that none of the checkboxes in the Options dialog window is checked, press the *Repeat Last Command* button, and enter the desired number of random graphs minus one (you already have one random graph).

*Macro> Repeat
Last Command*

The second step in Monte Carlo simulation consists of calculating the desired network properties for all generated random graphs. Select the first random graph in the *Networks* drop-down menu and calculate an overall network property, for example, the clustering coefficient. Repeat this command the same number of times as in the first step. The network clustering coefficients calculated by the *Network> Create Vector> Clustering Coefficients> CC1* command for each network will be stored in vectors, one for each clustering coefficient. Note that the entries of these vectors refer to random graphs, not to vertices. The *Vector> Info* command prints the 2.5%, 5%, 95%, and 97.5% quantiles in the Report screen. These quantiles can be interpreted as boundaries of the 95% and 90% confidence intervals of the network characteristic, that is, the network clustering coefficients in this example.

Network characteristics may yield values over 999,999,997, which are ordinarily treated as missing values in Pajek. In the political blogs network, the counts of empty triads (003) or triads containing just one arc (012) may exceed this value, but they should not be treated as missing. To this end, select the option *Ignore Missing Values in menu Vector and Vectors* in the *Options> Read – Write* submenu.

Repeating the creation of a random graph with fixed-degree sequence requires special attention. This type of random graph is generated from one or two partitions and the same partition(s) must be used in all repetitions. To accomplish this, check the boxes before *Fix (First) Partition* and *Fix (Second) Partition* in the Options dialog window of the *Macro> Repeat Last Command* command. If these options are not checked, Pajek assumes that for each new repetition the next partition(s) must be used.

As with several other commands, the calculation of clustering coefficients produces vectors or partitions for each network in addition to a global network index. Quite a lot of storage space is needed for the partitions and vectors to each generated random graphs, especially if the graphs are not small. However, they are not needed for calculating confidence intervals on the global network characteristic. With a view to memory requirements, it is wise to remove all unwanted partitions and vectors. This can be done, again, with the *Repeat Last Command* command: dispose the first partition or vector (*File> Partition> Dispose* or *File> Vector> Dispose*) and repeat this command for all remaining partitions or vectors. Note that the boxes before *Fix (First) Partition* and *Fix (Second) Partition* should not be checked in the Options dialog window now because every iteration should use the next partition or vector in the drop-down menu.

With *Info> Memory (F11)* we can check at any time how much computer memory is still available for Pajek objects (Networks, Partitions, Vectors...). We recommend to check the available memory

whenever a command generates some larger network. If available memory is low, dispose some Pajek objects that you do not need for future operations. Dispose networks, for example, with the *File> Network> Dispose* command.

The Report screen contains some lines for each generated network, which also decreases the available memory. It is helpful to empty the Report screen (command *File> Empty Report* in the Report screen's menu) after generating all random graphs or all network statistics. The generation of random graphs took minutes, the generation of partitions or vectors took minutes for components to hours for betweenness centrality. There are clearly limits to the number and size of the random graphs that can be generated and analyzed.

[Report] *File>
Empty Report*

Exercise III

Perform a Monte Carlo simulation for the network of dining-table partner choice network (*Dormitory.net*) of Chapter 1. Simulate 1,000 directed networks for each relevant random graph model. Motivate your model choice and parameter settings and look for network motifs using the triad census.

13.5 Summary

This chapter introduced the concept of randomness in social network analysis. We accept the idea that the observed network could have been different, notably that some of the actual lines might not have been present and some absent lines might have been present, if our measurement method had been different or history had taken a slightly different course. To think about how different the network could have been, we must formulate a statistical network model, that is, a mathematical description of a collection of possible networks and a probability distribution for this set. We constrain the set of possible networks by fixing some network characteristics, notably the number of vertices and network density or average degree, but additional constraints can be imposed, such as the degree distribution. We assume that the network must have had these characteristics. Because we do not take into account vertex attributes or other additional data beyond the structure of lines, we prefer to speak of (random) graphs rather than random networks.

The probability distribution tells us how likely it is that we observe any of the possible graphs. If we list all possible graphs, we can determine the probability that each graph occurs. However, it is practically feasible to enumerate all possible graphs only for very small graphs. Most statistical network models therefore specify probabilities for the presence of single lines rather than probabilities of entire graphs. Randomness is included

in the generation of lines, and random generation of lines creates random graphs. The social network analyst may interpret this random process as an assumption about the behavior of the pair, the actor sending the line, or the social system. If line probability depends on the pair's context or the network position of the actors, actors are assumed to take into account context or network position when they establish social ties.

Three types of random graph models were presented, each with a characteristic random process. The first type of random graph model assumes that every line and hence every alter from the perspective of the sender is equally likely. Among the random graph models with uniform probability distributions, we distinguish between the Bernoulli model, which constrains only the number of vertices and the density of the graph, and general or conditional uniform random graph models, which also constrain other network properties, for example, the degree distribution. If the average degree exceeds one, these random graphs are very likely to contain one large component, just like many social networks.

Bernoulli random graphs tend to feature much less clustering than social networks. The second type of random graph model, namely the small-world model, remedies this problem, yielding high clustering and small average distance among vertices both of which characterize social networks. This model assumes that vertices are linked to a fixed set of geographically or otherwise close neighbors, so context is deemed relevant to line formation. In addition, the model assumes that some vertices are randomly linked to vertices that are not close, which ensures that average distance remains low even in large networks.

The third type of model assumes that actors have a preference for linking to popular actors, that is, actors who are already involved in many links. This is usually called preferential attachment. The probability that a vertex is incident with a new line depends on its (in)degree. This is a network growth model, building the network by adding new vertices and new lines step by step. In large preferential attachment random graphs, the vertex degree tends to be distributed according to a power law, especially if the lowest degrees are excluded. This implies that the degree distribution is extremely skewed – few vertices have very high degree – or scale-free, which is often found in social networks. Just like the Bernoulli random graph, however, clustering in a preferential attachment random graph is usually lower than in social networks.

We use random graph models in an exploratory way, fitting different random graph models to the observed network, possibly trying out a range of parameter values. Once we find a fitting random graph model, we can proceed to confirmatory network analysis, that is, testing hypotheses about network properties that are not part of the random graph model,

for example, whether there are significantly more transitive triads than expected under the Bernoulli random graph model. Monte Carlo simulation creates a probability distribution of the network property from which a confidence interval can be construed. If the observed value of the network property falls outside the confidence interval, its value is significantly large or small.

As our example illustrated, finding one random graph model that fits the observed network on all key characteristics (density, connectedness, compactness, and clustering) is not guaranteed. Selecting among imperfectly fitting random graph models may be difficult, so this may not be the best approach to hypothesis testing on social networks. The last decade has witnessed a rapid development of new statistical techniques for hypothesis testing on social network data. The most popular and powerful techniques are exponential random graph models (ERGM) for cross-sectional data and continuous-time Markov process models for longitudinal data, notably panel data. In contrast to the approaches presented in this chapter, these models do not focus on overall network structure. Instead, they aim to predict the occurrence or quality of each single line, regarding overall network structure as the mere result of local tie formation. The truly confirmatory character of these approaches and their complexity goes beyond the domain that this book on explanatory social network analysis intends to cover.

13.6 Questions

1. Does the Bernoulli random graph model have a uniform probability distribution?
 - a. Yes, because a uniform probability distribution means that each vertex has the same degree, which is the case in a Bernoulli random graph model.
 - b. Yes, because a uniform probability distribution means that all lines have equal probability, which is the case in a Bernoulli random graph model.
 - c. No, because a uniform probability distribution means that each vertex has the same degree, which is not the case in a Bernoulli random graph.
 - d. No, because a uniform probability distribution means that all lines have equal probability, which is not the case in a Bernoulli random graph.
2. Suppose a conditional uniform random graph model with fixed out-degree distribution fits an observed network. Which network property should *not* be tested with Monte Carlo simulation?

- a. Average degree
 - b. Betweenness centralization
 - c. Diameter
 - d. Transitivity
3. Suppose a small-world model fits an observed network. Which network property can be tested with Monte Carlo simulation?
 - a. Average degree
 - b. Clustering
 - c. Diameter
 - d. Transitivity
 4. Which statement is correct for every random graph generated with the Barabási–Albert model, starting with a network consisting of one vertex?
 - a. It does not contain isolates.
 - b. It contains a giant component and some small components.
 - c. Its degree distribution follows a power law.
 - d. Its degree distribution is scale-free.
 5. According to the Monte Carlo simulation results in the table that follows (1,000 random graphs), which random graph model fits the observed undirected network best?
 - a. The Bernoulli random graph model
 - b. The general random graph model conditioned on degree
 - c. The small-world random graph model
 - d. The preferential attachment random graph model

	Observed network	Bernoulli		Degree conditional		Small world		Preferential attachment	
		2.5%	97.5%	2.5%	97.5%	2.5%	97.5%	2.5%	97.5%
No. of components	1	1	1	1	1	1	1	1	2
% Largest component	100	100	100	100	100	100	100	99.2	100
Diameter	5	4	5	4	4	5	7	5	7
Av. distance	2.98	2.44	2.62	2.53	2.57	3.09	3.45	2.61	2.82
Clustering	0.337	0.053	0.080	0.040	0.067	0.439	0.523	0.075	0.138
Triad census:									
003 ^a	233.6	229.9	238.4	234.8	236.4	233.2	233.4	238.3	250.3
102 ^a	51.65	46.50	53.72	48.60	50.00	52.13	52.50	35.25	45.49
201	2371	3012	4200	2936	3155	1644	1967	2281	4133
300	401	60	113	43	70	511	602	66	210

^a In thousands.

6. Can any of the triad types be considered a network motif in the table accompanying Exercise 5?
- No
 - Yes, triad 102
 - Yes, triad 201
 - Yes, triad 300

13.7 Assignment

Citation networks tend to have extremely skewed indegree distributions, with some papers cited many times whereas most papers are cited sporadically. This is likely to be valid also at the level of authors: Few authors are cited by very many other authors. The network `Sciencemetrics_1978_2000.net` contains the authors who published articles in the journal *Scientometrics* in the period between 1978 and 2000 and the authors that cite one or more of these articles. Line values indicate the number of times the sender cites an article written by the receiver of the arc.

Determine whether the indegree distribution follows a power law and use the estimated value of the exponent α to simulate random graphs with the same power-law indegree distribution. Does the clustering in the observed network differ significantly from the clustering in the random graphs?

13.8 Further Reading

- The political blogs network stems from L. A. Adamic and N. Glance, “The political blogosphere and the 2004 US Election.” In *Proceedings of the WWW-2005 Workshop on the Weblogging Ecosystem* (2005). The citation data on the journal *Scientometrics* were collected by E. Garfield using HistCite software. This network results from searches in the Web of Science database and it is available as a courtesy of ISI (Philadelphia).
- Section 13.3 closely follows treatment of these topics in E. D. Kolaczyk, *Statistical Analysis of Network Data* (New York, NY: Springer Science+Business Media, 2009) and M. E. Newman, *Networks. An Introduction* (Oxford: Oxford University Press, 2010), which are recommended texts for reading more on these and other network models.
- Classic texts on random graph models: P. Erdős and A. Rényi, “On random graphs I.” (*Publicationes Mathematicae*

- 6 [1959], 290–7) and “The evolution of random graphs” (*Magyar Tudományos Akadémia. Matematikai Kutató Intézet Közleményei* 5 [1960], 17–61); E. N. Gilbert, “Random Graphs.” (*The Annals of Mathematical Statistics* 30 [1959], 1141–4) on the Bernoulli model; L. Katz and J. H. Powell, “Probability distributions of random variables associated with a structure of the sample space of sociometric investigations.” (*Annals of Mathematical Statistics* 28 [1957], 442–8) on uniform models conditioned on indegree and outdegree; P. W. Holland and S. Leinhardt, “A method for detecting structure in sociometric data.” (*American Journal of Sociology* 70 [1970], 492–513) and “Local structure in social networks.” (*Sociological Methodology* 7 [1976], 1–45) on uniform probability models conditional on the dyad distribution; D. J. Watts and S. H. Strogatz, “Collective dynamics of ‘small-world’ networks.” (*Nature* 393 [1998], 440–2) on the small-world model; D. de Solla Price, “A general theory of bibliometric and other cumulative advantage processes.” (*Journal of the American Society for Information Science* 27 [1976], 292); A. L. Barabási and R. Albert, “Emergence of scaling in Random Networks.” (*Science* 286 [1999], 509–12); and B. Bollobás, O. Riordan, J. Spencer, and G. Tusnady, “The degree sequence of a scale-free random graph process.” (*Random Structures and Algorithms* 18 [2001], 279–90) on preferential attachment.
- For an introduction to recent statistical models focusing on the local context of actors or ties, see S. Wasserman and G. Robins, “An introduction to random graphs, dependence graphs, and p^* ” as well as T. A. B. Snijders, “Models for longitudinal network data.” In P. J. Carrington, J. Scott, and S. Wasserman (eds.), *Models and Methods in Social Network Analysis* (Cambridge: Cambridge University Press [2005], 148–61 and 215–47) or G. Robins, P. Pattison, Y. Kalish, and D. Lusher, “An introduction to exponential random graph (p^*) models for social networks.” (*Social Networks* 29 [2007], 173–91) and T. A. B. Snijders, G. G. van de Bunt, and C. E. G. Steglich, “Introduction to stochastic actor-based models for network dynamics.” (*Social Networks* 32 [2010], 44–60.)
 - The algorithms for generating random graphs implemented in Pajek are described in V. Batagelj and U. Brandes, “Efficient generation of large random networks.” (*Physical Review E* 71 036113 [2005], 1–5. The maximum likelihood estimation of the power-law exponent is described in M. E. J. Newman, “Power laws, Pareto distributions and Zipf’s law.” (*Contemporary Physics* 46 [2005], 323–51.)

13.9 Answers

Answers to the exercises

- I. We found average path distances (output in the Report screen by the *Network> Create Vector> Distribution of Distances**) 82.6, 17.5, 6.0, and 5.1, respectively, for the random graphs. Because we are working with random graphs, your results may of course be different but they should be comparable. We may conclude that average distance drops sharply between rewiring probabilities 0.001 and 0.01 and substantially between 0.01 and 0.10 but not very much between 0.10 and 0.20. As stated in Section 13.3.2, a low proportion of rewired lines yield low average distance.
- II. It is quite likely that the power-law exponent diverges from 3 and even sharply so for several reasons. First, your graph is random, so it is likely to deviate at least some from the expectation. Note that the expectation specifies the average over a large number of random graphs generated with the same settings. Second, the expected value is valid in the limit of infinitely large graphs; although your graph is large, it is far from infinitely large. Third, the estimated value of alpha depends on where you cut of the tail, that is, your choice of the minimum degree to take into account. When we generated a random graph, alpha was 3.91 if we selected the tail with degree at least 50, whereas alpha was 2.82 in the tail including degrees 20 and up. Adding lines freely yielded lower values of alpha and produced a peculiar bump at the right end of the cumulative log-log distribution.
- III. Extract the dining-table partner choice relation. This network contains 26 vertices; average indegree and outdegree are each 2. Each girl was required to make a first and second choice, so this is clearly an example of a network with constrained outdegree. Therefore, a Bernoulli random graph, which may have outdegree variation, is not a suitable model for this network. Instead, use a conditional uniform model with outdegree fixed to 2 (*Network> Create Random Network> Vertices Output Degree* with minimum and maximum degree set to 2) and possibly a random graph conditioned on both indegree and outdegree (*Partitions> Make Random Network*).

Pajek cannot create directed small-world random graphs, so this option is not pursued here. Scale-free random graphs are created with the option *Adding> Free* not selected because each new vertex should be the sender of the two new lines added to the network. The initial network is set to size 2 with an initial probability of a line of 0.08, which is the density of the dining-table partners network. Alpha is set to 0.25, beta to 0, so the tendency for preferential attachment is only

one-third of the tendency to choose at random. This is a shot in the dark; any other value can be used just as well.

For calculating clustering coefficients, multiple lines must be removed, which may occur in the random graphs conditioned on input and output degree and in the preferential attachment random graphs. This is accomplished with the *Network > Create New Network > Transform > Remove > Multiple Lines > Single Line* command. Note that you should not create a new network, otherwise the *Repeat Last Command* will not work properly.

The dining-table partners network's clustering coefficient falls within the confidence intervals of all three models. Its diameter is more in line with the two conditionally uniform models; the preferential attachment model predicts a slightly shorter diameter. The dining-table partners' network's average distance, however, is slightly lower than in the two conditionally uniform models, whereas it is higher than the average distances in the preferential attachment model. The triad containing just one reciprocated choice (102) appears much more often than predicted by any of the random models used here, which indicates a tendency toward reciprocity. This also holds for triads 111D and 111U, which contain one mutual choice and one asymmetric choice.

Answers to the questions in Section 13.6

1. Answer b is correct. A uniform probability distribution means that all outcomes of a random variable have equal probability. In a Bernoulli random graph model, the presence of a line is the random variable and each line has the same probability to be present.
2. Answer a is correct. A conditional uniform random graph model with fixed outdegree distribution also constrains the density and average degree of the graph. Note that the average degree is the number of lines divided by the number of possible lines; the number of lines is equal to the sum of the outdegree (or indegree) of all vertices in a directed graph. Therefore, average degree cannot vary between random graphs with fixed outdegree distribution, so it is pointless to create a confidence interval for this characteristic with Monte Carlo simulation.
3. Answer c is (most) correct. The number of neighbors on the ring (or lattice) to which a vertex is connected on either side fixes the number of lines and average degree (a), the network's clustering (b), and transitivity (d). The small percentage of random rewiring that is usually applied to obtain relatively small average path length does not affect clustering and transitivity substantially. Of course, when the percentage of rewiring is very high, clustering and transitivity may vary from one random graph to another.

	Dining-table	Outdegree conditional		Out- and indegree		Preferential attachment	
		2.5%	97.5%	2.5%	97.5%	2.5%	97.5%
No. of components	1	1	1	1	1	1	1
% Largest component	26	26	26	26	26	26	26
Diameter	7	6	11	6	12	4	6
Av. distance	2.84	3.05	4.07	2.97	4.27	1.75	2.48
Clustering	0.081	0.022	0.101	0.020	0.099	0.026	0.136
Triad census:							
003	1705	1519	1611	1562	1688	1497	1857
012	592	758	921	723	874	648	891
102	195	0	96	0	80	0	17
021D	7	13	23	11	21	9	47
021U	26	25	54	29	51	28	79
021C	32	58	94	48	85	35	89
111D	25	0	17	0	18	0	7
111U	10	0	8	0	7	0	0
030T	1	0	7	0	7	2	20
030C	0	0	5	0	5	0	0
201	3	0	1	0	1	0	0
120D	1	0	1	0	1	0	1
120U	1	0	1	0	1	0	0
120C	2	0	2	0	2	0	0
210	0	0	0	0	0	0	0
300	0	0	0	0	0	0	0

- Statement a is correct. In the Barabási–Albert model, each new vertex is incident with the new lines added in the same step, so it must be connected to other vertices. There are no isolates in the network. Because the initial network contains one vertex, the network is very likely to be connected into one component. Statement b does not have to be true. Statements c and d need not be true either because only the right tail of the degree distribution is expected to follow a power-law and have the scale-free property. Note that this is an expectation for very large graphs so a single graph or a small graph need not have these characteristics.
- Answer c, the small-world model, is the best choice. The number and size of components as well as the diameter does not discriminate among the four random graph models; the observed network fits all models in these respects. The average distance is outside the confidence intervals for all models, being closest to the lower limit of the confidence interval for the small-world model. This is also true for network clustering, which is clearly closest to the small-world model.

6. Answer c is correct. If the small-world model is the reference model (see the answer to Exercise 5), triads 102 and 300 appear slightly less than expected. Network motifs are configurations that appear more often than expected by chance, so these triads clearly are not network motifs. Triad 201 appears more often (2,371 times) than expected under the small-world model (upper limit of the confidence interval: 1,967 times), so this can be considered a network motif.

Appendix 1

Getting Started with Pajek

A1.1 Installation

Pajek software for network analysis can be installed on all computers operating under Windows[®] 32 and 64. To use computer memory more efficiently, different versions of Pajek for Windows 32 and 64 are available. Pajek runs also under Linux, Linux 64 or Mac OS X[®] via Wine, WineBottler, and similar software. Details on installing and running Pajek on Mac OS X are provided in Appendix 3.

To install Pajek software on the computer's hard disk no administrative user privileges are needed. You can also run Pajek directly from the Pajek webpage as a Web Start application. Thus, you can first test a new version of Pajek to check if it is worth migrating to the newer version.

When you install Pajek on the computer's hard disk, two additional executable files are installed: PajekXXL and Pajek3XL. They are proposed to be used to analyse huge networks that cannot be handled by regular Pajek. While Pajek accepts networks containing up to 1 billion vertices, PajekXXL can handle up to 2 billion vertices, and Pajek3XL can handle networks containing up to 10 billion vertices. We recommend learning regular Pajek very well before using PajekXXL or Pajek3XL. That is why the two programs have not been mentioned so far. At the end of this appendix (Section A1.5), you will find some basic instructions on using both programs and combining them with regular Pajek.

A1.2 Network Data Formats

Pajek can read network data in several plain text formats, that is, files containing unformatted text (ASCII). Labels of vertices, lines, and relations in Unicode UTF8 with BOM (Byte Order Mask) are supported in Pajek as well. This means that labels can be written in any alphabet (see

*File>
Network>
Read*

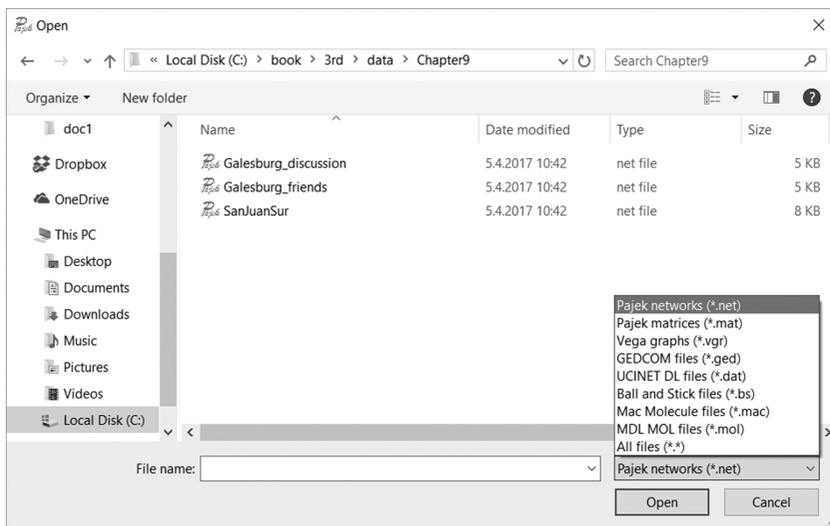


Figure 132. Read Network dialog box.

the Pajek webpage for information on using Unicode). We briefly discuss the network data formats in the order in which they appear on the dialog box issued by the *File > Network > Read* command (Figure 132).

The first two data formats are indigenous Pajek formats: Pajek networks and Pajek matrices. The *Pajek network* format (the filename extension is `.net`) is explained in Chapter 1 (Section 1.3.1), and additional features for longitudinal networks are discussed in Chapter 4 (Section 4.5). Basically, it is a list of vertices followed by a list of arcs and edges. It is the most flexible format because it allows for multiple lines, and many (layout) properties of the vertices and lines can be specified, which are explained in the file `pajekman.pdf` on the Pajek website. The Pajek network format is in line with the logic of relational databases. See Section A1.3.3 for more information on organizing your network data as a relational database and exporting from database software to Pajek. There are three network formats: Arcs/Edges, ArcsList/EdgesList, and ArcsList/EdgesList (min). In Section A1.3, we discuss only the first type (Arcs/Edges) because it is most flexible.

The *Pajek matrix* format (with the filename extension `.mat`) is slightly different. The list of vertices is the same as in the Pajek network format, but the list of edges and arcs is replaced by a matrix consisting of integers or real numbers, which are separated by blanks (Figure 133). This is an ordinary adjacency matrix (see Chapter 12, Section 12.2). The Pajek matrix format is useful for importing network data in matrix format.

The third data format is the *Vega* format devised by Pisanski. The fourth format, *GEDCOM*, is the standard data format for genealogical

```

*Vertices      6
  1 "Ada"      0.1646  0.2144  0.5000
  2 "Cora"     0.0481  0.3869  0.5000
  3 "Louise"   0.3472  0.1913  0.5000
  4 "Jean"     0.1063  0.5935  0.5000
  5 "Helen"    0.2892  0.6688  0.5000
  6 "Martha"   0.4630  0.5179  0.5000
*Matrix
0.000 1.000 1.000 0.000 0.000 0.000
1.000 0.000 0.000 1.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 1.000 0.000
0.000 0.000 0.000 1.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000

```

Figure 133. A network in Pajek matrix format.

data, which is discussed in Chapter 11 (Section 11.3; see also “Further Reading”). The fourth format, *UCINET DL files*, is the raw data format of another widely used program for network analysis: UCINET. Both UCINET and Pajek can read networks exported in this format, so this is the format for exchanging data between the two software packages. For more information about UCINET, consult the website www.analytictech.com/.

The last three data formats (*Ball and Stick*, *Mac Molecule*, and *MDL MOL*) were developed for chemistry. These formats are not widely used in social network analysis.

A1.3 Creating Network Files for Pajek

There are several ways to create data files that can be read by Pajek. We distinguish between four methods: creating data files manually within Pajek, using helper software, creating them in a word processor, and exporting them from a relational database. We present the methods in this order, from the simple and inflexible to the complicated and versatile.

A1.3.1 Within Pajek

In Pajek, you can create new networks, partitions, and vectors manually by means of a number of commands and tricks, which have been presented in the chapters of this book. Let us start with creating a new network. We suppose that you have the data on the vertices and lines at hand (e.g., on paper). Start Pajek and use the *Network> Create New Network> Empty Network* command to create a new empty network (see

```

Network>
Create New
Network>
Empty
Network

```

Vertex	Val	Label
1	0	v1
2	0	v2
3	0	v3
4	0	v4

Figure 134. Editing vertex labels.

Chapter 1, Section 1.4). Just enter the desired number of vertices in the dialog box.

Partition> As a first step, add labels to the vertices so they can be identified more easily. Here, the trick is to create a new partition with the *Partition> Create Constant Partition> 0* command (see Chapter 2, Section 2.3) and then open it in an Edit screen (the *File> Partition> View/Edit* command; see Chapter 2, Section 2.3). The result may look like that in Figure 134. In this screen, you can manually edit the labels of the vertices. Now you have defined the labels of the vertices.

File> As a next step, you must add lines to the network. This can be done in the Editing Network screen (Figure 135), which can be opened either from the Main screen with the *File> Network> View/Edit* command specifying a vertex number or label or from the Draw screen by right-clicking a vertex. Double-click the word *Newline* to add an edge (enter the number of the vertex that must be connected to the selected vertex) or an arc (enter the vertex number preceded by a plus sign [arc toward the selected vertex]) or a minus sign [arc from the selected vertex]. This is discussed in more detail in Chapter 1, Section 1.4.

Partition> If you have a discrete characteristic of the vertices (e.g., the sex of persons), you can store this in a new partition. Create a new empty partition, and open it in an Edit screen as discussed previously. Then give each person (vertex) the appropriate sex code in the Edit Partition screen. Just enter the right code in the “Val” column (see Figure 134 and Chapter 2, Section 2.3). Note that Pajek accepts only numerical codes. As an alternative, you can draw the network with a new empty partition by executing the *Draw> Network + Create Null Partition* command (see

Vertex	Val	Label
1:	4.1	val=1.00000 / v4.v1
1:	1.3	val=1.00000 / v1.v3
1:	1-2	val=1.00000 / v1-v2

Newline

Figure 135. Edit Network screen.

Chapter 2, Section 2.3). Now, you can respectively raise and lower the class associated with a vertex by left-clicking the vertex while pressing the *Shift* key, which is equivalent to clicking with the middle mouse button, or left-clicking it while holding down the *Alt* key (see Chapter 2, Section 2.3).

Vectors, which specify continuous characteristics of vertices, may be created in a similar way. Create a new vector with the *Vector> Create Constant Vector* command (enter 1), and edit it manually in the Editing Vector screen, which opens on execution of the *File> Vector> View/Edit* command. Initially, all vertices have 1 as their vector value. You can manually change it to the desired number.

*Vector> Create
Constant
Vector*
*File> Vector>
View/Edit*

Don't forget to save the data files before you exit from Pajek; otherwise your work will be lost. Pajek does not save data files automatically. Save the data files by means of the diskette button to the left of the drop-down menus or with the *Save* command in the appropriate submenu of the *File* menu: *Network*, *Partition*, or *Vector*. You can save the network in any of Pajek's formats as well as in UCINET's DL format.

*File>
Network> Save*
*File>
Partition> Save*
*File> Vector>
Save*

A1.3.2 Helper Software

Jürgen Pfeffer wrote three Windows software programs for creating Pajek network files: *createpajek.exe*, *txt2Pajek.exe* and *txt2Pajek3.exe* (www.pfeffer.at/txt2pajek/index.php). All programs require a list of lines, each line identified by two vertex names. *createpajek.exe* converts from an MS Excel[®] spreadsheet file with one network line per row, one column containing the name or label of the line's tail, and another column with the name or label of the head. This program can output a one-mode (directed and undirected) and two-mode network in Pajek Arcs/Edges format, assigning sequential numbers to vertices. It cannot read and export line values or line properties.

Txt2Pajek.exe and *Txt2Pajek3.exe* read data from a plain text file with one line in the text file containing one edge or arc, specifying the name or label of the tail and head at a minimum. The labels must be separated by a special character, for example, a Tab character. We suggest you use *Txt2Pajek3.exe* which is a newer and more powerful version of the program *Txt2Pajek.exe*. The two programs are more versatile and much faster than *createpajek.exe*. They can output one-mode (directed and undirected) and two-mode networks and they can also read and output line values as well as properties of lines, for example, line color and its time stamp. The user must specify the correct Pajek markers (prefix and suffix), e.g., 1 " and " for line label or [and] for time stamps. See Section A2.2 and the file *pajekman.pdf* for details.

```

*Vertices          6
  1 "v1"           0.1000   0.5000   0.5000
  2 "v2"           0.3000   0.1536   0.5000
  3 "v3"           0.7000   0.1536   0.5000
  4 "v4"           0.9000   0.5000   0.5000
  5 "v5"           0.7000   0.8464   0.5000
  6 "v6"           0.3000   0.8464   0.5000
*Arcs

```

Figure 136. An empty network in Pajek Arcs/Edges format.

A1.3.3 Word Processor

Network>
Create New
Network>
Empty
Network

It is possible to create a network data file in Pajek, but adding lines by means of the Editing Network screen (Figure 135) is quite tedious. Usually, it is more efficient to create a network with the right number of vertices but without lines in Pajek with the *Network> Create New Network> Empty Network* command and to change vertex labels and add lines in a word processor.

Figure 136 shows an empty network consisting of six vertices, which was created with the *Network> Create New Network> Empty Network* command and saved from Pajek as a Pajek Arcs/Edges network file. It is a plain text file, so it can be opened in any word processor including those that support Unicode UTF-8 with BOM format (e.g., NotePad, Textpad, WordPad, NotePad++, BabelPad). Its first line specifies the number of vertices. This number may never be lower than the highest vertex number in the network.

The first line is followed by the list of vertices: Each vertex has one line containing the vertex number (here: 1 to 6); the vertex label, which is the letter *v* followed by a number by default; and the *x*, *y*, and *z* coordinates of the vertex in a layout. In a word processor, you can easily change these properties of the vertices, but it is wise not to meddle with the vertex numbers even though Pajek allows vertices to be out of order or some vertex numbers to be absent in the list. Empty lines, however, are not allowed in the list of vertices.

In Figure 136, the last line reads **Arcs*, which signals the start of the list of arcs. In this example, the list is still empty. You may add arcs manually, using one line for each arc. First, specify the number of the sending vertex. Next, add one or more spaces and the number of the receiving vertex. Optionally, you can add the line value after one or more spaces. If you do not specify the line value, it is considered to be 1 (e.g., the arc from vertex 1 to vertex 2 in Figure 137). For undirected lines, first add the line **Edges* and then use a line for each edge just like the arcs. The only difference is that it does not matter which vertex is specified first (when saving, Pajek always writes the vertex with the smaller number first). There is no limit

```

*Vertices      6
  1 "v1"      0.1000  0.5000  0.5000
  2 "v2"      0.3000  0.1536  0.5000
  3 "v3"      0.7000  0.1536  0.5000
  4 "v4"      0.9000  0.5000  0.5000
  5 "v5"      0.7000  0.8464  0.5000
  6 "v6"      0.3000  0.8464  0.5000

*Arcs
  1 2
  2 5 2.5
  4 5
*Edges
  3 4 -1
  5 6

```

Figure 137. A network in the Pajek Arcs/Edges format.

to the number of arcs or edges, and multiple arcs and edges are allowed. The order of the lines does not matter.

If you prefer to enter the data as a matrix, as in Figure 133, replace the line `*Arcs` with `*Matrix` and type in a square matrix, that is, a matrix with as many rows as columns. Use one line for each row and at least one space between the cells in different columns. You may use any number of spaces between two numbers within a row. Now, the network data file may look like that in Figure 138: There is an arc from vertex 1 to vertex 2 (second number or cell in the first row of the matrix) with a line value of 1. Line values may have decimal places and negative signs.

There are some disadvantages to the matrix data format. First of all, Pajek cannot distinguish between an edge and a bidirectional arc. Therefore, a matrix read by Pajek contains only arcs. Second, Pajek cannot distinguish between a line with a value of 0 and an absent line, so Pajek replaces all zeros with arcs with a line value of 0 unless it is instructed

[Main]
Options> Read
- Write>
Threshold

```

*Vertices      6
  1 "v1"      0.1000  0.5000  0.5000
  2 "v2"      0.3000  0.1536  0.5000
  3 "v3"      0.7000  0.1536  0.5000
  4 "v4"      0.9000  0.5000  0.5000
  5 "v5"      0.7000  0.8464  0.5000
  6 "v6"      0.3000  0.8464  0.5000

*Matrix
  0  1  0  0  0  0
  0  0  0  0  2.5 0
  0  0  0 -1  0  0
  0  0 -1  0  1  0
  0  0  0  0  0  1
  0  0  0  0  1  0

```

Figure 138. A network in the Pajek matrix format.

```

*Vertices      6 2
  1 "org1"      0.1000    0.5000    0.5000
  2 "org2"      0.3000    0.1536    0.5000
  3 "person1"   0.7000    0.1536    0.5000
  4 "person2"   0.9000    0.5000    0.5000
  5 "person3"   0.7000    0.8464    0.5000
  6 "person4"   0.3000    0.8464    0.5000

*Arcs
  1 3
  2 5 2.5
  1 4
*Edges
  2 4 -1
  1 6

```

Figure 139. A two-mode network in the Pajek Arcs/Edges format.

to ignore lines with a value of 0 in the *Threshold* field of the *Options>Read-Write* dialog box. The threshold must be set to 0 before a network is opened. Note that the threshold applies to absolute values: Negative line values (e.g., -1 in Figure 138) are not eliminated if the threshold is 0. Finally, it is impossible to have multiple lines in matrix format.

Until now, we have considered only one-mode networks. Section 5.3 in Chapter 5, however, introduced *two-mode networks*: networks consisting of two sets of vertices such that all lines are found between the sets (e.g., affiliations of people to organizations). Pajek network data files have a special arrangement for two-mode networks. You can split the list of vertices into two sets or modes in the **Vertices* statement by specifying the total number of vertices followed by one or more spaces and the number of vertices in the first mode. This requires that the list of vertices is sorted such that the vertices in the first mode have the lowest vertex numbers and the vertices in the second mode have the highest numbers. Figure 139 shows the data file of a two-mode network consisting of two organizations and four persons. Note that Pajek will issue a warning if it encounters lines within a mode, which should not occur in a two-mode network.

A1.3.4 Relational Database

Data on large networks are optimally stored as relational databases. In a relational database, two tables can represent a *one-mode network* (Figure 140): One table contains the vertices of the network (e.g., countries), and another table stores the lines between the vertices (e.g., trade relations such as imports). This matches the basic structure of the Pajek Arcs/Edges network format: a list of vertices and a list of lines.

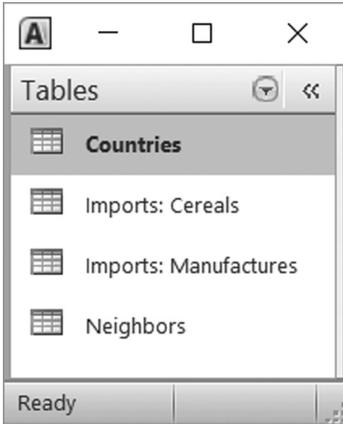


Figure 140. Four tables in the world trade database (MS Access[®] 2010).

This book's website contains a Microsoft Access[®] database with the world trade data of Chapter 2 (see Section 2.10 for the data sources): `world_trade_94.mdb`. The database is made with MS Access version 2010, so you have to convert the database if you open it in a newer version of this software. The *Countries* table is the heart of the database, containing one record (row) for each country (Figure 141). Each country has a numerical code, which is its vertex number in the network. In addition to the country's name [never use double quote characters (") in labels], several properties are listed, including its gross domestic product (GDP),

vertexn	vertexlabel	x	y	continent	trade bloc	position
1	Algeria	0,4380682	0,4957265	1	0	999998
2	Argentina	0,2159091	0,7749288	6	3	2
3	Australia	0,8636364	0,7008547	5	0	2
4	Austria	0,46875	0,411396	3	0	2
5	Barbados	0,2670454	0,5766382	6	4	999998
6	Bangladesh	0,7301137	0,5071225	2	8	999998
7	Belgium, Lux.	0,45	0,3931624	3	1	1
8	Belize	0,1352273	0,5259259	4	4	999998
9	Bolivia	0,2215909	0,6951567	6	3	999998
10	Brazil	0,2897727	0,6780627	6	3	2
11	Canada	0,1363636	0,3304843	4	2	1
12	Chile	0,1931818	0,7635328	6	3	3

Figure 141. Contents of the *Countries* table (partial).

tail	head	value	valperc
Belgium, Lux.	Southern Africa	9931	2,136122
France	Southern Africa	12892	2,773022
Germany	Spain	29006	6,239084
United Kingdom	Sri Lanka	20081	4,319349
Australia	Sweden	7514	1,616234
Canada	Switzerland	25318	22,98711
United States	Thailand	70774	64,25822
Chile	Trinidad Tobago	1902	1,726893
Colombia	Tunisia	4836	4,390775
Germany	Ecuador	5062	4,595969

Figure 142. A Lookup to the *Countries* table.

geographic location, and population growth. The table provides the data for the list of vertices in a network data file and the data for partitions and vectors (e.g., GDP and population size of the countries). In principle, any relevant characteristic of the countries can be added to this table. For instance, we added the coordinates of the country in a map (fields x and y), which we used to create the geographical layout of the trade network in Figure 26 (see Chapter 2).

The network lines are stored in separate tables. We collected the information on two kinds of trade: imports of miscellaneous manufactures of metal (table *Imports: Manufactures*) and imports of cereals (table *Imports: Cereals*). Both tables contain one record for each trade tie with the code of the exporting country (field *tail*) and the code of the importing country (field *head*). The value of imports (in thousands of dollars US) and the share in a country's total imports of this commodity have been registered. These tables are very similar to the arcs list in a Pajek network file. In addition, we created a table of undirected ties (viz., countries that share a border [table *Neighbors*]). This table contains a record for each pair of neighboring countries, and it is the source of a list of edges in a network data file.

The *Imports: Cereals* table exemplifies a feature of relational database software that is very useful for efficient data entry. Once the table of vertices has been created and filled with data, references to this table can be made from another table. In a table containing lines between vertices, you can look up the (vertex) number of a country directly. In MS Access, for example, the countries can be displayed as a list, from which you can easily pick the right country: Start typing the country's name and the software will lead you to the country you are looking for (Figure 142). When you pick a country from this list – press *Enter* when the country is

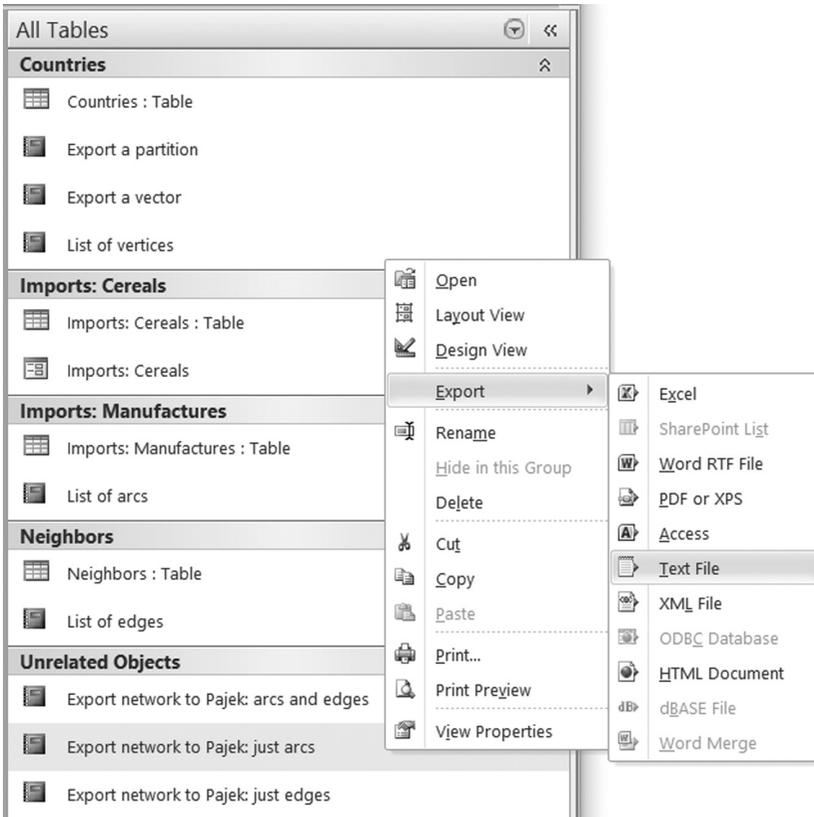


Figure 143. Export a report to plain text.

found – the country's (vertex) number is added to the table of ties although its name is displayed. This helps avoiding data entry errors.

Once all data have been entered into the database, Pajek network data files can be created. This boils down to combining the table of the vertices (*Countries*) with one or more tables of ties (e.g., *Imports: Manufactures* and *Neighbors*) in the right layout. In Microsoft Access, the Report utility is most suited for that. If you open the list of reports in the example database, you will see several reports (Figure 143). For exporting a one-mode network, two reports are needed. The first report (*List of vertices*) lists the vertices in the right layout and adds the *Vertices line. The second report (*List of arcs*) contains the formatted list of arcs with the *Arcs line. In the case of an undirected relation, a report is needed starting with *Edges instead of *Arcs (report *List of edges*). The two reports (or three reports in case there are arcs and edges simultaneously) are combined in a master report (*Export network to Pajek: just arcs*, *Export network to*

Pajek: just edges, and *Export network to Pajek: arcs and edges*), which merely concatenates the partial reports.

The master report must be exported to plain text: text without special layout characters. Unfortunately, Microsoft Access is a little uncooperative at this point. It is possible to export the report in plain text format. Right-click the report in the report list (Figure 143) select *Export*, and finally save it by selecting *Text Files* and providing a name for the file. Sometimes, however, the saved file has an empty line every four lines, which renders it unreadable to Pajek.

As an alternative, the report can be exported to Microsoft Word[®] (in Rich Text format) with the *Export> Word RTF File* command. Access adds page breaks, *which must be deleted* before the file is saved in plain text format. The Rich Text format file is automatically opened in MS Word, where you can delete the page breaks with the command *Replace* (or *Ctrl-h*). Enter \hat{m} , which denotes a page break, in the *Find what:* line and replace it with nothing (leave the *Replace with:* line empty). Finally, use the command *File> Save As* and select the *Plain text (.txt)* type to save the network data file. Give it the *.net* extension so Pajek can recognize it as a network data file. To avoid Windows adding a default extension such as *.txt*, write the file name in double quotation marks, for example, “example.net”. This ensures that the file name has the *.net* extension.

Exporting partitions and vectors is less complicated but more sensitive to errors. Partitions and vectors are simply sorted lists of numbers, one number for each vertex on a separate line, preceded by a **Vertices* statement. A simple report listing the sorted contents of a field in the table of vertices is all you need to create a partition or vector data file; the reports *Export a partition* and *Export a vector* do this. They can be exported to plain text files in the manner just described. Use the *.clu* and *.vec* extensions to enhance recognition by Pajek.

However, the user should be careful. The report can export nonnumeric fields, but Pajek cannot handle these. This will be apparent when you try to open a nonnumeric partition or vector in Pajek. There is an even more important limitation. The report lists just the values sorted by vertex number. The value attached to the lowest vertex number is stored in the first line after the **Vertices* statement. Reading the partition or vector, Pajek will assume that this value belongs to vertex 1. Therefore, vertex numbers in the database must start with 1 for a partition or vector to match the network. For the same reason, no vertex numbers may be missing in the exported network, so vertex numbers must range from 1 to the number of vertices.

This is required for a partition or vector exported from the database to match a network exported from the same database. Otherwise, the partition or vector will contain fewer vertices than the network, so you cannot combine them in Pajek. If you want to export only some of the vertices

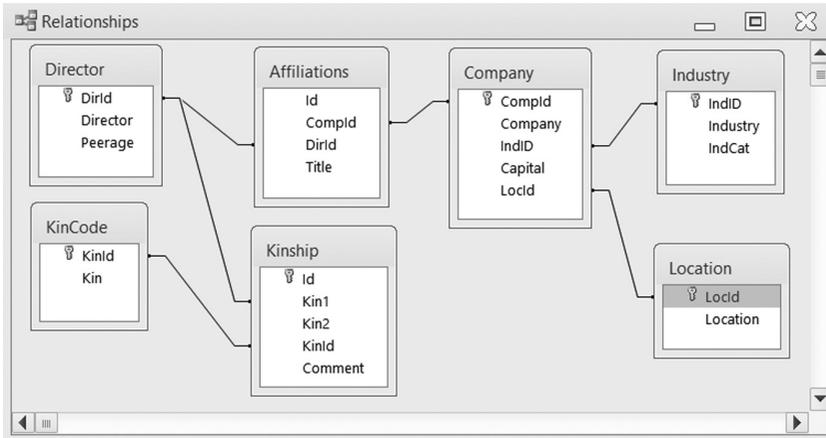


Figure 144. Tables and relations in the database of Scottish companies.

in the database, you should either renumber the remaining vertices in the database (in the vertices table as well as in the relations tables!) or export the entire network, partitions, and vectors and remove the unwanted vertices from the network, partitions, and vectors in Pajek (see Chapter 2).

As a final note on exporting vectors (and coordinates in the list of vertices), we want to draw your attention to the language and regional settings of Windows. If decimal places are not separated by a point, Pajek will not be able to read the numbers. You must either adjust your regional settings or replace your local decimal separator by a point by means of a word processor in your data files.

Exporting a *two-mode network* such as corporate interlocks is more complicated. In a relational database, one would probably store the vertices in two tables: one containing the first mode, for instance, the companies (table *Company* in Figure 144), and the second mode, for instance, the directors, in another table (table *Director* in Figure 144). Ties connect vertices from one table of vertices to the other (table *Affiliations* in Figure 144). In Pajek, the vertex numbers of the two modes should not overlap. One should number the vertices in the first mode (the firms) from 1 to the number of vertices. The numbering of the other mode (the directors) should start with 1 plus the number of vertices in the first mode. If this precaution is taken (in the example database, queries automatically perform this task), a two-mode network can be exported to Pajek using the previously suggested steps, with the additional trick of merging the two tables of vertices into one list.

The example database `Scotland.mdb` contains the data from John Scott's study of interlocking directorates (the example of Chapter 5) and a report for exporting a two-mode network (*Export two-mode network*

to Pajek). In this example, special queries ensure that the vertex numbers of directors and companies do not overlap.

In the Pajek website, we included an empty MS Access (2010) database (`network.mdb`) that you can use as a starting point for your own data collection. The database contains the tables, queries, and reports you need to create Pajek data files for one-mode and two-mode networks. The tables contain a minimum of sample data, which can easily be replaced by your own data. Note that vertex numbers are automatically assigned to vertices in the tables. Do not delete a vertex and its number because that number cannot be used afterward. Furthermore, do not change the names of the reports, tables, and fields within the tables unless you know very well what you are doing. Renaming may affect the reports and the links between tables. To make the most of the relational database, learn more about the database software or find someone who can advise you.

A1.4 Limitations

[Main]
Options> Read
- Write> Max.
vertices to draw

Pajek can handle networks up to 999,999,997 vertices. If your network exceeds this number, use PajekXXL or Pajek3XL instead. In general, drawings of networks should contain no more than a few thousand vertices because for very large networks the drawing procedure is time consuming and the drawing is visually unattractive. By default, Pajek will not draw networks larger than 100,000 vertices, but you can change this limit in the *Options> Read-Write* menu. Available resources on the computer (physical memory) may pose additional limitations. More details are explained in the following section.

A1.5 PajekXXL and Pajek3XL

PajekXXL and Pajek3XL are special editions of program Pajek. Their memory consumption is much lower than the consumption of regular Pajek. For the same network the two programs need at least two to three times less physical memory than regular Pajek. Operations that are memory intensive, such as generating random networks, extracting subnetworks, or shrinking networks, are also faster.

PajekXXL and Pajek3XL are designed to analyze huge networks that cannot be loaded in regular Pajek. While Pajek accepts networks containing up to 1 billion vertices (exactly 999,999,997), PajekXXL can handle networks containing up to 2 billion vertices (exactly 1,999,999,997), and the limit is currently set to 10 billion vertices (exactly 9,999,999,997) in Pajek3XL but this limit can easily be further increased. First use the two programs to extract smaller, interesting parts from huge networks, then

analyze and visualize the extracted parts in regular Pajek, which contains the more sophisticated methods.

PajekXXL and Pajek3XL can also be used to analyze relatively simple properties of networks provided that the identity of vertices is not important because the programs do not store vertex labels. Simulation studies using random networks are a case in point. In random networks, vertices have no identity and interest is focused on simple properties such as degree distributions and triad counts. As we have learned in Chapter 13, we can generate some thousands of large random networks using *Repeat Last Command* and compute mean values, variances, and other interesting statistics. In Chapter 13, we used regular Pajek but we could have used PajekXXL or Pajek3XL to increase speed and allow for much larger networks.

On first sight, the PajekXXL and Pajek3XL main windows look very similar to Pajek's main window with a different color: green in Pajek, blue in PajekXXL, and gray in Pajek3XL. On closer inspection, you will see that the data objects are not exactly the same. The *Permutation* data object and menu do not exist in PajekXXL and Pajek3XL. It is replaced by a new data object and menu called *Vertex ID*, which identifies vertices by retaining their vertex number in the original network.

The internal network data structure in PajekXXL and Pajek3XL is limited to the graph: a list of vertices and lists of edges and/or arcs that represent relations. All additional information such as vertex labels, coordinates, shapes, colors, timestamps, and so on, is eliminated. As a consequence, menu options that require additional information on vertices or lines are not available in PajekXXL and Pajek3XL, for example, the Draw menu that requires vertex coordinates is not available at all.

The internal data structure in PajekXXL and Pajek3XL is highly optimized to use the available memory very efficiently. The space needed to store a network in PajekXXL and Pajek3XL can be precisely calculated, which is not possible in Pajek because labels of vertices are Unicode strings that can be very short or very long. Let n be number of vertices and m the number of lines in a network. The space needed to store a network in PajekXXL and Pajek3XL (64-bit) equals $8n + 64m$. Notice that one line needs the same amount of space as eight vertices (ratio 64: 8). PajekXXL and Pajek3XL have especially good performance when networks contain a huge number of vertices but relatively few lines (sparse). Using this formula we can roughly estimate that we need 16G RAM or more for sparse networks with 100 million vertices and 128G RAM or more for networks with close to a billion vertices.

PajekXXL uses 32-bit (4 bytes) integers for vertices numbers whereas Pajek3XL uses 64-bit (8 bytes) integers. Still both programs need exactly the same amount of memory to save the network ($8n + 64m$). But Pajek3XL needs more memory to store partitions because it uses twice

the number of bytes for each vertex number and partition cluster number (a partition on n vertices requires $8n$ bytes in Pajek3XL and only $4n$ bytes in PajekXXL). To reserve memory for partitions and vectors (the latter require the same amount of memory in all Pajek versions), it is recommended to use PajekXXL if the network contains fewer than 2 billion vertices. Use Pajek3XL only for networks that cannot be loaded to PajekXXL because they contain more than 2 billion vertices.

*Tools> Pajek>
Send Network
to Pajek> +
Add Vertex
Labels from
File(s)*

PajekXXL and Pajek3XL are typically used in the following way: We start with a huge network that cannot be loaded in regular Pajek. First we use PajekXXL or Pajek3XL to find some interesting smaller (cohesive) subnetworks with procedures that are fast enough to handle huge networks. Such procedures are, for example, components, communities, cores, islands, fragments, and main paths. Then we call regular Pajek using *Tools> Pajek> Send Network to Pajek> + Add Vertex Labels from File(s)*. This command asks for the name of the network file containing vertex labels – remember that PajekXXL and Pajek3XL do not store vertex labels. Once we provide this file name, vertex IDs in the subnetwork are replaced by the vertex labels stored in the selected file and the subnetwork is automatically opened in regular Pajek. Now we can use all analyses and visualizations that we have learned in this book.

Files storing huge networks tend to be huge files. If we plan to work with PajekXXL or Pajek3XL, it is a good practice to save the network as two separate files. The first file contains only the *Vertices statement with the number of vertices directly followed by the list of arcs and edges. A list of vertex labels is not needed because all additional vertex information is dropped. The second file contains the list of vertex labels preceded by the *Vertices statement, so the *Arcs and *Edges parts are missing. We need to load only the first file in PajekXXL or Pajek3XL. We use the second file to add vertex labels when we send a subnetwork from PajekXXL or Pajek3XL to Pajek.

*Tools> Pajek>
Locate Pajek*

The first time we export a subnetwork from PajekXXL or Pajek3XL to Pajek, we must locate the directory where ordinary Pajek is installed. Use *Tools> Pajek> Locate Pajek* and select the right folder on your computer.

You can get much more information on using PajekXXL and Pajek3XL on the Pajek website. Some examples of huge networks are also available there.

A1.6 Updates of Pajek

Updates of Pajek, PajekXXL, and Pajek3XL can be downloaded from <http://mrvar.fdv.uni-lj.si/pajek/>. Owing to modifications, newer versions of Pajek may not match the illustrations, command names, and output described in this book exactly. The list of all modifications and additions

in each new Pajek version is available in the history file on the Pajek webpage. The main Pajek webpage contains links to basic introductions to Pajek in different languages. At the time of writing the third edition of this book (February 2018), introductions are available in the following languages: Greek, Polish, Spanish, Portuguese, German, Persian, French, Japanese, Chinese, Italian, Lithuanian, English, and Slovenian. Several video lectures on using Pajek can be found there as well.

Appendix 1

Getting Started with Pajek

A1.1 Installation

Pajek software for network analysis can be installed on all computers operating under Windows[®] 32 and 64. To use computer memory more efficiently, different versions of Pajek for Windows 32 and 64 are available. Pajek runs also under Linux, Linux 64 or Mac OS X[®] via Wine, WineBottler, and similar software. Details on installing and running Pajek on Mac OS X are provided in Appendix 3.

To install Pajek software on the computer's hard disk no administrative user privileges are needed. You can also run Pajek directly from the Pajek webpage as a Web Start application. Thus, you can first test a new version of Pajek to check if it is worth migrating to the newer version.

When you install Pajek on the computer's hard disk, two additional executable files are installed: PajekXXL and Pajek3XL. They are proposed to be used to analyse huge networks that cannot be handled by regular Pajek. While Pajek accepts networks containing up to 1 billion vertices, PajekXXL can handle up to 2 billion vertices, and Pajek3XL can handle networks containing up to 10 billion vertices. We recommend learning regular Pajek very well before using PajekXXL or Pajek3XL. That is why the two programs have not been mentioned so far. At the end of this appendix (Section A1.5), you will find some basic instructions on using both programs and combining them with regular Pajek.

A1.2 Network Data Formats

Pajek can read network data in several plain text formats, that is, files containing unformatted text (ASCII). Labels of vertices, lines, and relations in Unicode UTF8 with BOM (Byte Order Mask) are supported in Pajek as well. This means that labels can be written in any alphabet (see

File>
Network>
Read

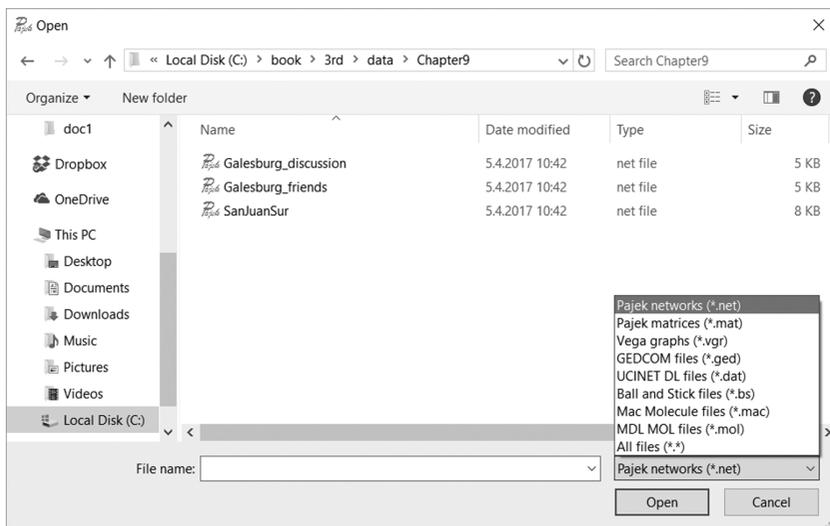


Figure 132. Read Network dialog box.

the Pajek webpage for information on using Unicode). We briefly discuss the network data formats in the order in which they appear on the dialog box issued by the *File > Network > Read* command (Figure 132).

The first two data formats are indigenous Pajek formats: Pajek networks and Pajek matrices. The *Pajek network* format (the filename extension is `.net`) is explained in Chapter 1 (Section 1.3.1), and additional features for longitudinal networks are discussed in Chapter 4 (Section 4.5). Basically, it is a list of vertices followed by a list of arcs and edges. It is the most flexible format because it allows for multiple lines, and many (layout) properties of the vertices and lines can be specified, which are explained in the file `pajekman.pdf` on the Pajek website. The Pajek network format is in line with the logic of relational databases. See Section A1.3.3 for more information on organizing your network data as a relational database and exporting from database software to Pajek. There are three network formats: Arcs/Edges, ArcsList/EdgesList, and ArcsList/EdgesList (min). In Section A1.3, we discuss only the first type (Arcs/Edges) because it is most flexible.

The *Pajek matrix* format (with the filename extension `.mat`) is slightly different. The list of vertices is the same as in the Pajek network format, but the list of edges and arcs is replaced by a matrix consisting of integers or real numbers, which are separated by blanks (Figure 133). This is an ordinary adjacency matrix (see Chapter 12, Section 12.2). The Pajek matrix format is useful for importing network data in matrix format.

The third data format is the *Vega* format devised by Pisanski. The fourth format, *GEDCOM*, is the standard data format for genealogical

```

*Vertices      6
  1 "Ada"      0.1646  0.2144  0.5000
  2 "Cora"     0.0481  0.3869  0.5000
  3 "Louise"   0.3472  0.1913  0.5000
  4 "Jean"     0.1063  0.5935  0.5000
  5 "Helen"    0.2892  0.6688  0.5000
  6 "Martha"   0.4630  0.5179  0.5000
*Matrix
0.000 1.000 1.000 0.000 0.000 0.000
1.000 0.000 0.000 1.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 1.000 0.000
0.000 0.000 0.000 1.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000

```

Figure 133. A network in Pajek matrix format.

data, which is discussed in Chapter 11 (Section 11.3; see also “Further Reading”). The fourth format, *UCINET DL files*, is the raw data format of another widely used program for network analysis: UCINET. Both UCINET and Pajek can read networks exported in this format, so this is the format for exchanging data between the two software packages. For more information about UCINET, consult the website www.analytictech.com/.

The last three data formats (*Ball and Stick*, *Mac Molecule*, and *MDL MOL*) were developed for chemistry. These formats are not widely used in social network analysis.

A1.3 Creating Network Files for Pajek

There are several ways to create data files that can be read by Pajek. We distinguish between four methods: creating data files manually within Pajek, using helper software, creating them in a word processor, and exporting them from a relational database. We present the methods in this order, from the simple and inflexible to the complicated and versatile.

A1.3.1 Within Pajek

In Pajek, you can create new networks, partitions, and vectors manually by means of a number of commands and tricks, which have been presented in the chapters of this book. Let us start with creating a new network. We suppose that you have the data on the vertices and lines at hand (e.g., on paper). Start Pajek and use the *Network> Create New Network> Empty Network* command to create a new empty network (see

```

Network>
Create New
Network>
Empty
Network

```

Vertex	Val	Label
1	0	v1
2	0	v2
3	0	v3
4	0	v4

Figure 134. Editing vertex labels.

Chapter 1, Section 1.4). Just enter the desired number of vertices in the dialog box.

Partition> As a first step, add labels to the vertices so they can be identified more easily. Here, the trick is to create a new partition with the *Partition> Create Constant Partition> 0* command (see Chapter 2, Section 2.3) and then open it in an Edit screen (the *File> Partition> View/Edit* command; see Chapter 2, Section 2.3). The result may look like that in Figure 134. In this screen, you can manually edit the labels of the vertices. Now you have defined the labels of the vertices.

File> As a next step, you must add lines to the network. This can be done in the Editing Network screen (Figure 135), which can be opened either from the Main screen with the *File> Network> View/Edit* command specifying a vertex number or label or from the Draw screen by right-clicking a vertex. Double-click the word *Newline* to add an edge (enter the number of the vertex that must be connected to the selected vertex) or an arc (enter the vertex number preceded by a plus sign [arc toward the selected vertex]) or a minus sign [arc from the selected vertex]. This is discussed in more detail in Chapter 1, Section 1.4.

Partition> If you have a discrete characteristic of the vertices (e.g., the sex of persons), you can store this in a new partition. Create a new empty partition, and open it in an Edit screen as discussed previously. Then give each person (vertex) the appropriate sex code in the Edit Partition screen. Just enter the right code in the “Val” column (see Figure 134 and Chapter 2, Section 2.3). Note that Pajek accepts only numerical codes. As an alternative, you can draw the network with a new empty partition by executing the *Draw> Network + Create Null Partition* command (see

Vertex	Val	Label
1:	4.1	val=1.00000 / v4.v1
1:	1.3	val=1.00000 / v1.v3
1:	1-2	val=1.00000 / v1-v2

Newline

Figure 135. Edit Network screen.

Chapter 2, Section 2.3). Now, you can respectively raise and lower the class associated with a vertex by left-clicking the vertex while pressing the *Shift* key, which is equivalent to clicking with the middle mouse button, or left-clicking it while holding down the *Alt* key (see Chapter 2, Section 2.3).

Vectors, which specify continuous characteristics of vertices, may be created in a similar way. Create a new vector with the *Vector> Create Constant Vector* command (enter 1), and edit it manually in the Editing Vector screen, which opens on execution of the *File> Vector> View/Edit* command. Initially, all vertices have 1 as their vector value. You can manually change it to the desired number.

Don't forget to save the data files before you exit from Pajek; otherwise your work will be lost. Pajek does not save data files automatically. Save the data files by means of the diskette button to the left of the drop-down menus or with the *Save* command in the appropriate submenu of the *File* menu: *Network*, *Partition*, or *Vector*. You can save the network in any of Pajek's formats as well as in UCINET's DL format.

*Vector> Create
Constant
Vector*

*File> Vector>
View/Edit*

*File>
Network> Save*

*File>
Partition> Save*

*File> Vector>
Save*

A1.3.2 Helper Software

Jürgen Pfeffer wrote three Windows software programs for creating Pajek network files: *createpajek.exe*, *txt2Pajek.exe* and *txt2Pajek3.exe* (www.pfeffer.at/txt2pajek/index.php). All programs require a list of lines, each line identified by two vertex names. *createpajek.exe* converts from an MS Excel[®] spreadsheet file with one network line per row, one column containing the name or label of the line's tail, and another column with the name or label of the head. This program can output a one-mode (directed and undirected) and two-mode network in Pajek Arcs/Edges format, assigning sequential numbers to vertices. It cannot read and export line values or line properties.

txt2Pajek.exe and *txt2Pajek3.exe* read data from a plain text file with one line in the text file containing one edge or arc, specifying the name or label of the tail and head at a minimum. The labels must be separated by a special character, for example, a Tab character. We suggest you use *txt2Pajek3.exe* which is a newer and more powerful version of the program *txt2Pajek.exe*. The two programs are more versatile and much faster than *createpajek.exe*. They can output one-mode (directed and undirected) and two-mode networks and they can also read and output line values as well as properties of lines, for example, line color and its time stamp. The user must specify the correct Pajek markers (prefix and suffix), e.g., 1 " and " for line label or [and] for time stamps. See Section A2.2 and the file *pajekman.pdf* for details.

```

*Vertices          6
  1 "v1"           0.1000   0.5000   0.5000
  2 "v2"           0.3000   0.1536   0.5000
  3 "v3"           0.7000   0.1536   0.5000
  4 "v4"           0.9000   0.5000   0.5000
  5 "v5"           0.7000   0.8464   0.5000
  6 "v6"           0.3000   0.8464   0.5000
*Arcs

```

Figure 136. An empty network in Pajek Arcs/Edges format.

A1.3.3 Word Processor

Network>
Create New
Network>
Empty
Network

It is possible to create a network data file in Pajek, but adding lines by means of the Editing Network screen (Figure 135) is quite tedious. Usually, it is more efficient to create a network with the right number of vertices but without lines in Pajek with the *Network> Create New Network> Empty Network* command and to change vertex labels and add lines in a word processor.

Figure 136 shows an empty network consisting of six vertices, which was created with the *Network> Create New Network> Empty Network* command and saved from Pajek as a Pajek Arcs/Edges network file. It is a plain text file, so it can be opened in any word processor including those that support Unicode UTF-8 with BOM format (e.g., NotePad, Textpad, WordPad, NotePad++, BabelPad). Its first line specifies the number of vertices. This number may never be lower than the highest vertex number in the network.

The first line is followed by the list of vertices: Each vertex has one line containing the vertex number (here: 1 to 6); the vertex label, which is the letter *v* followed by a number by default; and the *x*, *y*, and *z* coordinates of the vertex in a layout. In a word processor, you can easily change these properties of the vertices, but it is wise not to meddle with the vertex numbers even though Pajek allows vertices to be out of order or some vertex numbers to be absent in the list. Empty lines, however, are not allowed in the list of vertices.

In Figure 136, the last line reads **Arcs*, which signals the start of the list of arcs. In this example, the list is still empty. You may add arcs manually, using one line for each arc. First, specify the number of the sending vertex. Next, add one or more spaces and the number of the receiving vertex. Optionally, you can add the line value after one or more spaces. If you do not specify the line value, it is considered to be 1 (e.g., the arc from vertex 1 to vertex 2 in Figure 137). For undirected lines, first add the line **Edges* and then use a line for each edge just like the arcs. The only difference is that it does not matter which vertex is specified first (when saving, Pajek always writes the vertex with the smaller number first). There is no limit

```

*Vertices      6
  1 "v1"      0.1000  0.5000  0.5000
  2 "v2"      0.3000  0.1536  0.5000
  3 "v3"      0.7000  0.1536  0.5000
  4 "v4"      0.9000  0.5000  0.5000
  5 "v5"      0.7000  0.8464  0.5000
  6 "v6"      0.3000  0.8464  0.5000

*Arcs
  1 2
  2 5 2.5
  4 5
*Edges
  3 4 -1
  5 6

```

Figure 137. A network in the Pajek Arcs/Edges format.

to the number of arcs or edges, and multiple arcs and edges are allowed. The order of the lines does not matter.

If you prefer to enter the data as a matrix, as in Figure 133, replace the line `*Arcs` with `*Matrix` and type in a square matrix, that is, a matrix with as many rows as columns. Use one line for each row and at least one space between the cells in different columns. You may use any number of spaces between two numbers within a row. Now, the network data file may look like that in Figure 138: There is an arc from vertex 1 to vertex 2 (second number or cell in the first row of the matrix) with a line value of 1. Line values may have decimal places and negative signs.

There are some disadvantages to the matrix data format. First of all, Pajek cannot distinguish between an edge and a bidirectional arc. Therefore, a matrix read by Pajek contains only arcs. Second, Pajek cannot distinguish between a line with a value of 0 and an absent line, so Pajek replaces all zeros with arcs with a line value of 0 unless it is instructed

[Main]
Options> Read
- Write>
Threshold

```

*Vertices      6
  1 "v1"      0.1000  0.5000  0.5000
  2 "v2"      0.3000  0.1536  0.5000
  3 "v3"      0.7000  0.1536  0.5000
  4 "v4"      0.9000  0.5000  0.5000
  5 "v5"      0.7000  0.8464  0.5000
  6 "v6"      0.3000  0.8464  0.5000

*Matrix
  0  1  0  0  0  0
  0  0  0  0  2.5 0
  0  0  0 -1  0  0
  0  0 -1  0  1  0
  0  0  0  0  0  1
  0  0  0  0  1  0

```

Figure 138. A network in the Pajek matrix format.

```

*Vertices      6 2
  1 "org1"      0.1000    0.5000    0.5000
  2 "org2"      0.3000    0.1536    0.5000
  3 "person1"   0.7000    0.1536    0.5000
  4 "person2"   0.9000    0.5000    0.5000
  5 "person3"   0.7000    0.8464    0.5000
  6 "person4"   0.3000    0.8464    0.5000

*Arcs
  1 3
  2 5 2.5
  1 4
*Edges
  2 4 -1
  1 6

```

Figure 139. A two-mode network in the Pajek Arcs/Edges format.

to ignore lines with a value of 0 in the *Threshold* field of the *Options>Read-Write* dialog box. The threshold must be set to 0 before a network is opened. Note that the threshold applies to absolute values: Negative line values (e.g., -1 in Figure 138) are not eliminated if the threshold is 0. Finally, it is impossible to have multiple lines in matrix format.

Until now, we have considered only one-mode networks. Section 5.3 in Chapter 5, however, introduced *two-mode networks*: networks consisting of two sets of vertices such that all lines are found between the sets (e.g., affiliations of people to organizations). Pajek network data files have a special arrangement for two-mode networks. You can split the list of vertices into two sets or modes in the **Vertices* statement by specifying the total number of vertices followed by one or more spaces and the number of vertices in the first mode. This requires that the list of vertices is sorted such that the vertices in the first mode have the lowest vertex numbers and the vertices in the second mode have the highest numbers. Figure 139 shows the data file of a two-mode network consisting of two organizations and four persons. Note that Pajek will issue a warning if it encounters lines within a mode, which should not occur in a two-mode network.

A1.3.4 Relational Database

Data on large networks are optimally stored as relational databases. In a relational database, two tables can represent a *one-mode network* (Figure 140): One table contains the vertices of the network (e.g., countries), and another table stores the lines between the vertices (e.g., trade relations such as imports). This matches the basic structure of the Pajek Arcs/Edges network format: a list of vertices and a list of lines.

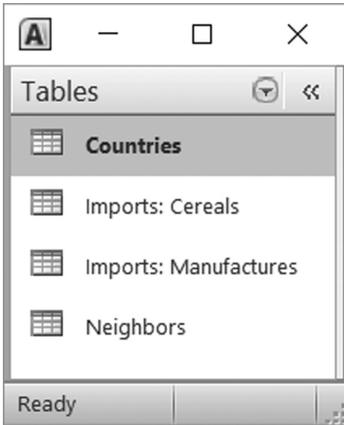


Figure 140. Four tables in the world trade database (MS Access[®] 2010).

This book's website contains a Microsoft Access[®] database with the world trade data of Chapter 2 (see Section 2.10 for the data sources): `world_trade_94.mdb`. The database is made with MS Access version 2010, so you have to convert the database if you open it in a newer version of this software. The *Countries* table is the heart of the database, containing one record (row) for each country (Figure 141). Each country has a numerical code, which is its vertex number in the network. In addition to the country's name [never use double quote characters (") in labels], several properties are listed, including its gross domestic product (GDP),

 A screenshot of the Microsoft Access 'Countries' table. The table has the following columns: vertexn, vertexlabel, x, y, continent, trade bloc, and position. The data is as follows:

vertexn	vertexlabel	x	y	continent	trade bloc	position
1	Algeria	0,4380682	0,4957265	1	0	999998
2	Argentina	0,2159091	0,7749288	6	3	2
3	Australia	0,8636364	0,7008547	5	0	2
4	Austria	0,46875	0,411396	3	0	2
5	Barbados	0,2670454	0,5766382	6	4	999998
6	Bangladesh	0,7301137	0,5071225	2	8	999998
7	Belgium, Lux.	0,45	0,3931624	3	1	1
8	Belize	0,1352273	0,5259259	4	4	999998
9	Bolivia	0,2215909	0,6951567	6	3	999998
10	Brazil	0,2897727	0,6780627	6	3	2
11	Canada	0,1363636	0,3304843	4	2	1
12	Chile	0,1931818	0,7635328	6	3	3

 The status bar at the bottom shows 'Record: 62 of 80', 'No Filter', and a search field.

Figure 141. Contents of the *Countries* table (partial).

tail	head	value	valperc
Belgium, Lux.	Southern Africa	9931	2,136122
France	Southern Africa	12892	2,773022
Germany	Spain	29006	6,239084
United Kingdom	Sri Lanka	20081	4,319349
Australia	Sweden	7514	1,616234
Canada	Switzerland	25318	22,98711
United States	Thailand	70774	64,25822
Chile	Trinidad Tobago	1902	1,726893
Colombia	Tunisia	4836	4,390775
Germany	Ecuador	5062	4,595969

Figure 142. A Lookup to the *Countries* table.

geographic location, and population growth. The table provides the data for the list of vertices in a network data file and the data for partitions and vectors (e.g., GDP and population size of the countries). In principle, any relevant characteristic of the countries can be added to this table. For instance, we added the coordinates of the country in a map (fields x and y), which we used to create the geographical layout of the trade network in Figure 26 (see Chapter 2).

The network lines are stored in separate tables. We collected the information on two kinds of trade: imports of miscellaneous manufactures of metal (table *Imports: Manufactures*) and imports of cereals (table *Imports: Cereals*). Both tables contain one record for each trade tie with the code of the exporting country (field *tail*) and the code of the importing country (field *head*). The value of imports (in thousands of dollars US) and the share in a country's total imports of this commodity have been registered. These tables are very similar to the arcs list in a Pajek network file. In addition, we created a table of undirected ties (viz., countries that share a border [table *Neighbors*]). This table contains a record for each pair of neighboring countries, and it is the source of a list of edges in a network data file.

The *Imports: Cereals* table exemplifies a feature of relational database software that is very useful for efficient data entry. Once the table of vertices has been created and filled with data, references to this table can be made from another table. In a table containing lines between vertices, you can look up the (vertex) number of a country directly. In MS Access, for example, the countries can be displayed as a list, from which you can easily pick the right country: Start typing the country's name and the software will lead you to the country you are looking for (Figure 142). When you pick a country from this list – press *Enter* when the country is

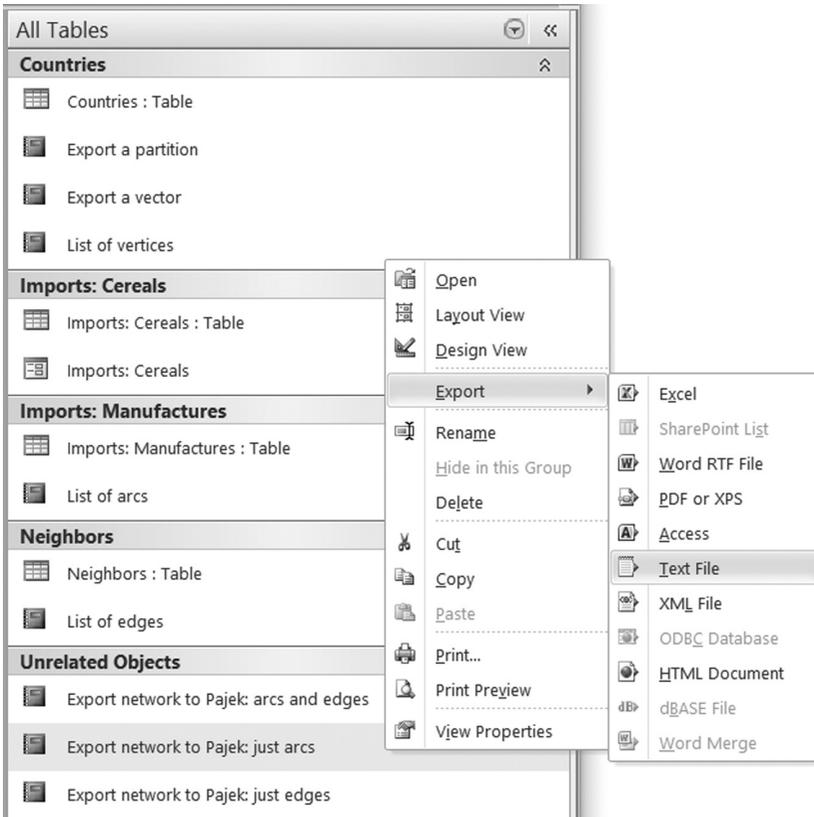


Figure 143. Export a report to plain text.

found – the country's (vertex) number is added to the table of ties although its name is displayed. This helps avoiding data entry errors.

Once all data have been entered into the database, Pajek network data files can be created. This boils down to combining the table of the vertices (*Countries*) with one or more tables of ties (e.g., *Imports: Manufactures* and *Neighbors*) in the right layout. In Microsoft Access, the Report utility is most suited for that. If you open the list of reports in the example database, you will see several reports (Figure 143). For exporting a one-mode network, two reports are needed. The first report (*List of vertices*) lists the vertices in the right layout and adds the *Vertices line. The second report (*List of arcs*) contains the formatted list of arcs with the *Arcs line. In the case of an undirected relation, a report is needed starting with *Edges instead of *Arcs (report *List of edges*). The two reports (or three reports in case there are arcs and edges simultaneously) are combined in a master report (*Export network to Pajek: just arcs*, *Export network to*

Pajek: just edges, and *Export network to Pajek: arcs and edges*), which merely concatenates the partial reports.

The master report must be exported to plain text: text without special layout characters. Unfortunately, Microsoft Access is a little uncooperative at this point. It is possible to export the report in plain text format. Right-click the report in the report list (Figure 143) select *Export*, and finally save it by selecting *Text Files* and providing a name for the file. Sometimes, however, the saved file has an empty line every four lines, which renders it unreadable to Pajek.

As an alternative, the report can be exported to Microsoft Word[®] (in Rich Text format) with the *Export> Word RTF File* command. Access adds page breaks, *which must be deleted* before the file is saved in plain text format. The Rich Text format file is automatically opened in MS Word, where you can delete the page breaks with the command *Replace* (or *Ctrl-h*). Enter \hat{m} , which denotes a page break, in the *Find what:* line and replace it with nothing (leave the *Replace with:* line empty). Finally, use the command *File> Save As* and select the *Plain text (.txt)* type to save the network data file. Give it the *.net* extension so Pajek can recognize it as a network data file. To avoid Windows adding a default extension such as *.txt*, write the file name in double quotation marks, for example, “example.net”. This ensures that the file name has the *.net* extension.

Exporting partitions and vectors is less complicated but more sensitive to errors. Partitions and vectors are simply sorted lists of numbers, one number for each vertex on a separate line, preceded by a **Vertices* statement. A simple report listing the sorted contents of a field in the table of vertices is all you need to create a partition or vector data file; the reports *Export a partition* and *Export a vector* do this. They can be exported to plain text files in the manner just described. Use the *.clu* and *.vec* extensions to enhance recognition by Pajek.

However, the user should be careful. The report can export nonnumeric fields, but Pajek cannot handle these. This will be apparent when you try to open a nonnumeric partition or vector in Pajek. There is an even more important limitation. The report lists just the values sorted by vertex number. The value attached to the lowest vertex number is stored in the first line after the **Vertices* statement. Reading the partition or vector, Pajek will assume that this value belongs to vertex 1. Therefore, vertex numbers in the database must start with 1 for a partition or vector to match the network. For the same reason, no vertex numbers may be missing in the exported network, so vertex numbers must range from 1 to the number of vertices.

This is required for a partition or vector exported from the database to match a network exported from the same database. Otherwise, the partition or vector will contain fewer vertices than the network, so you cannot combine them in Pajek. If you want to export only some of the vertices

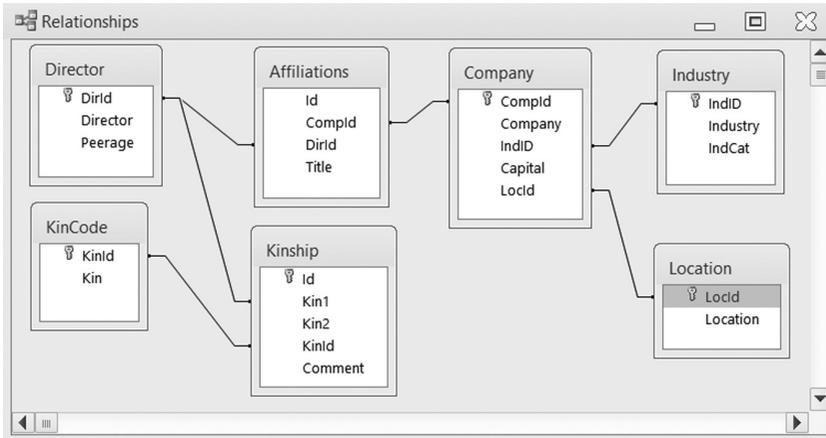


Figure 144. Tables and relations in the database of Scottish companies.

in the database, you should either renumber the remaining vertices in the database (in the vertices table as well as in the relations tables!) or export the entire network, partitions, and vectors and remove the unwanted vertices from the network, partitions, and vectors in Pajek (see Chapter 2).

As a final note on exporting vectors (and coordinates in the list of vertices), we want to draw your attention to the language and regional settings of Windows. If decimal places are not separated by a point, Pajek will not be able to read the numbers. You must either adjust your regional settings or replace your local decimal separator by a point by means of a word processor in your data files.

Exporting a *two-mode network* such as corporate interlocks is more complicated. In a relational database, one would probably store the vertices in two tables: one containing the first mode, for instance, the companies (table *Company* in Figure 144), and the second mode, for instance, the directors, in another table (table *Director* in Figure 144). Ties connect vertices from one table of vertices to the other (table *Affiliations* in Figure 144). In Pajek, the vertex numbers of the two modes should not overlap. One should number the vertices in the first mode (the firms) from 1 to the number of vertices. The numbering of the other mode (the directors) should start with 1 plus the number of vertices in the first mode. If this precaution is taken (in the example database, queries automatically perform this task), a two-mode network can be exported to Pajek using the previously suggested steps, with the additional trick of merging the two tables of vertices into one list.

The example database `Scotland.mdb` contains the data from John Scott's study of interlocking directorates (the example of Chapter 5) and a report for exporting a two-mode network (*Export two-mode network*

to Pajek). In this example, special queries ensure that the vertex numbers of directors and companies do not overlap.

In the Pajek website, we included an empty MS Access (2010) database (`network.mdb`) that you can use as a starting point for your own data collection. The database contains the tables, queries, and reports you need to create Pajek data files for one-mode and two-mode networks. The tables contain a minimum of sample data, which can easily be replaced by your own data. Note that vertex numbers are automatically assigned to vertices in the tables. Do not delete a vertex and its number because that number cannot be used afterward. Furthermore, do not change the names of the reports, tables, and fields within the tables unless you know very well what you are doing. Renaming may affect the reports and the links between tables. To make the most of the relational database, learn more about the database software or find someone who can advise you.

A1.4 Limitations

[Main]
Options> Read
- Write> Max.
vertices to draw

Pajek can handle networks up to 999,999,997 vertices. If your network exceeds this number, use PajekXXL or Pajek3XL instead. In general, drawings of networks should contain no more than a few thousand vertices because for very large networks the drawing procedure is time consuming and the drawing is visually unattractive. By default, Pajek will not draw networks larger than 100,000 vertices, but you can change this limit in the *Options> Read-Write* menu. Available resources on the computer (physical memory) may pose additional limitations. More details are explained in the following section.

A1.5 PajekXXL and Pajek3XL

PajekXXL and Pajek3XL are special editions of program Pajek. Their memory consumption is much lower than the consumption of regular Pajek. For the same network the two programs need at least two to three times less physical memory than regular Pajek. Operations that are memory intensive, such as generating random networks, extracting subnetworks, or shrinking networks, are also faster.

PajekXXL and Pajek3XL are designed to analyze huge networks that cannot be loaded in regular Pajek. While Pajek accepts networks containing up to 1 billion vertices (exactly 999,999,997), PajekXXL can handle networks containing up to 2 billion vertices (exactly 1,999,999,997), and the limit is currently set to 10 billion vertices (exactly 9,999,999,997) in Pajek3XL but this limit can easily be further increased. First use the two programs to extract smaller, interesting parts from huge networks, then

analyze and visualize the extracted parts in regular Pajek, which contains the more sophisticated methods.

PajekXXL and Pajek3XL can also be used to analyze relatively simple properties of networks provided that the identity of vertices is not important because the programs do not store vertex labels. Simulation studies using random networks are a case in point. In random networks, vertices have no identity and interest is focused on simple properties such as degree distributions and triad counts. As we have learned in Chapter 13, we can generate some thousands of large random networks using *Repeat Last Command* and compute mean values, variances, and other interesting statistics. In Chapter 13, we used regular Pajek but we could have used PajekXXL or Pajek3XL to increase speed and allow for much larger networks.

On first sight, the PajekXXL and Pajek3XL main windows look very similar to Pajek's main window with a different color: green in Pajek, blue in PajekXXL, and gray in Pajek3XL. On closer inspection, you will see that the data objects are not exactly the same. The *Permutation* data object and menu do not exist in PajekXXL and Pajek3XL. It is replaced by a new data object and menu called *Vertex ID*, which identifies vertices by retaining their vertex number in the original network.

The internal network data structure in PajekXXL and Pajek3XL is limited to the graph: a list of vertices and lists of edges and/or arcs that represent relations. All additional information such as vertex labels, coordinates, shapes, colors, timestamps, and so on, is eliminated. As a consequence, menu options that require additional information on vertices or lines are not available in PajekXXL and Pajek3XL, for example, the Draw menu that requires vertex coordinates is not available at all.

The internal data structure in PajekXXL and Pajek3XL is highly optimized to use the available memory very efficiently. The space needed to store a network in PajekXXL and Pajek3XL can be precisely calculated, which is not possible in Pajek because labels of vertices are Unicode strings that can be very short or very long. Let n be number of vertices and m the number of lines in a network. The space needed to store a network in PajekXXL and Pajek3XL (64-bit) equals $8n + 64m$. Notice that one line needs the same amount of space as eight vertices (ratio 64: 8). PajekXXL and Pajek3XL have especially good performance when networks contain a huge number of vertices but relatively few lines (sparse). Using this formula we can roughly estimate that we need 16G RAM or more for sparse networks with 100 million vertices and 128G RAM or more for networks with close to a billion vertices.

PajekXXL uses 32-bit (4 bytes) integers for vertices numbers whereas Pajek3XL uses 64-bit (8 bytes) integers. Still both programs need exactly the same amount of memory to save the network ($8n + 64m$). But Pajek3XL needs more memory to store partitions because it uses twice

the number of bytes for each vertex number and partition cluster number (a partition on n vertices requires $8n$ bytes in Pajek3XL and only $4n$ bytes in PajekXXL). To reserve memory for partitions and vectors (the latter require the same amount of memory in all Pajek versions), it is recommended to use PajekXXL if the network contains fewer than 2 billion vertices. Use Pajek3XL only for networks that cannot be loaded to PajekXXL because they contain more than 2 billion vertices.

*Tools> Pajek>
Send Network
to Pajek> +
Add Vertex
Labels from
File(s)*

PajekXXL and Pajek3XL are typically used in the following way: We start with a huge network that cannot be loaded in regular Pajek. First we use PajekXXL or Pajek3XL to find some interesting smaller (cohesive) subnetworks with procedures that are fast enough to handle huge networks. Such procedures are, for example, components, communities, cores, islands, fragments, and main paths. Then we call regular Pajek using *Tools> Pajek> Send Network to Pajek> + Add Vertex Labels from File(s)*. This command asks for the name of the network file containing vertex labels – remember that PajekXXL and Pajek3XL do not store vertex labels. Once we provide this file name, vertex IDs in the subnetwork are replaced by the vertex labels stored in the selected file and the subnetwork is automatically opened in regular Pajek. Now we can use all analyses and visualizations that we have learned in this book.

Files storing huge networks tend to be huge files. If we plan to work with PajekXXL or Pajek3XL, it is a good practice to save the network as two separate files. The first file contains only the *Vertices statement with the number of vertices directly followed by the list of arcs and edges. A list of vertex labels is not needed because all additional vertex information is dropped. The second file contains the list of vertex labels preceded by the *Vertices statement, so the *Arcs and *Edges parts are missing. We need to load only the first file in PajekXXL or Pajek3XL. We use the second file to add vertex labels when we send a subnetwork from PajekXXL or Pajek3XL to Pajek.

*Tools> Pajek>
Locate Pajek*

The first time we export a subnetwork from PajekXXL or Pajek3XL to Pajek, we must locate the directory where ordinary Pajek is installed. Use *Tools> Pajek> Locate Pajek* and select the right folder on your computer.

You can get much more information on using PajekXXL and Pajek3XL on the Pajek website. Some examples of huge networks are also available there.

A1.6 Updates of Pajek

Updates of Pajek, PajekXXL, and Pajek3XL can be downloaded from <http://mrvar.fdv.uni-lj.si/pajek/>. Owing to modifications, newer versions of Pajek may not match the illustrations, command names, and output described in this book exactly. The list of all modifications and additions

in each new Pajek version is available in the history file on the Pajek webpage. The main Pajek webpage contains links to basic introductions to Pajek in different languages. At the time of writing the third edition of this book (February 2018), introductions are available in the following languages: Greek, Polish, Spanish, Portuguese, German, Persian, French, Japanese, Chinese, Italian, Lithuanian, English, and Slovenian. Several video lectures on using Pajek can be found there as well.

Appendix 2

Exporting Visualizations

In Chapter 1, several options for exporting graphical output were briefly discussed. We provide more details in this appendix. In Section A2.1, we discuss the graphical formats that Pajek exports to. Next we discuss the options to adjust the layout of the exported image (Section A2.2).

A2.1 Export Formats

[Draw] Export Pajek can save a sociogram in nine different graphical formats (five two-dimensional and four three-dimensional). In most cases, viewers and plug-ins must be installed on your computer before you can display the exported layout. We now discuss each graphic format, how it is exported from Pajek, and how it can be viewed and edited. Note that all references to software and websites are accurate at the time of writing, which is February 2018; software updates may have changed and websites may have disappeared since then.

A2.1.1 Bitmap and JPEG

Export> 2D> Bitmap A bitmap and jpeg exported from Pajek with the *Export> 2D> Bitmap* and *Export> 2D> JPEG* commands are merely screen shots of the Draw screen in its current state, which is saved to a file. Because the bitmap and JPEG are very general graphical formats, they can be viewed and embedded in most Windows software. A bitmap can be edited in any Windows paint program, but editing is cumbersome because you have to edit each pixel (screen point) separately. It is not possible to move entire vertices, lines, or labels. The bitmap format is useful mainly for a quick presentation of a sociogram. In contrast to bitmaps which can be very large in size (the larger the Draw window the larger the size of bitmap), JPEG pictures are much smaller in size due to data compression. JPEG is

therefore the preferred format for presentations containing many screen shots. For quality images, use Encapsulated PostScript or Scalable Vector Graphics.

A2.1.2 Encapsulated PostScript

Encapsulated PostScript (and ordinary PostScript) produces two-dimensional vector drawings: Each vertex, line, and label is defined as a separate shape with a particular location in the plane. These formats have been developed for quality printing. The drawing or any part of it can be enlarged or reduced without loss of quality. All figures in this book, except for the screenshots, are based on Encapsulated PostScript from Pajek.

If you export a network layout in Encapsulated PostScript or PostScript formats (the *Export> 2D> EPS/PS* command), you can choose among several file types in the *Save As* dialog box. On the one hand, you can choose between Encapsulated PostScript (EPS) and ordinary PostScript (PS). We advise using Encapsulated PostScript always; ordinary PostScript cannot be directly viewed or printed (the *drawnet.pro* header is needed).

*Export> 2D>
EPS/PS*

On the other hand, you can choose between *What You See Is What You Get* (WYSIWYG), *Clip*, and *NoClip*. This choice affects the size and boundary of the exported drawing. In the WYSIWYG export, a vertex or label located outside the boundaries of the Draw screen will not be visible in the drawing (although it is defined in the drawing and it will be visible if the bounding box is adjusted), and the drawing may not fit on the standard A4 paper size. The *NoClip* export adjusts the size of the drawing so it can be printed on A4 paper (portrait), and the *Clip* file adjusts the boundaries of the drawing so all vertices and labels are included and there is more or less the same white margin everywhere. We recommend the *Clip* format.

Encapsulated PostScript is a supported graphics format in Word, LaTeX, and several other standard programs. If your application cannot show Encapsulated PostScript figures, you can use a special viewer or translate the drawing into another format. GhostScript and GhostView software can be downloaded for free from <http://pages.cs.wisc.edu/~ghost/>. Install both programs on your computer, and you can view (Encapsulated) PostScript drawings by opening the *.eps* or *.ps* file (*File> Open* command) in GhostView. From GhostView, you can print the drawing on any printer and export it in several graphical formats (including Windows MetaFile if you also install *pstoedit.exe*, which is freely available at www.pstoedit.net/).

With Acrobat Distiller, which is part of the Adobe® Acrobat® software package (not free; check www.adobe.com/products/acrobat/main.html), you can translate Encapsulated PostScript to a Portable Document

Format (PDF), which can be viewed with the free Adobe Acrobat Reader software (<http://get.adobe.com/reader/>).

Pajek offers many options for customizing its Encapsulated PostScript export, which we present in Section A2.2. For onscreen editing Encapsulated PostScript drawings, however, you need dedicated drawing software such as CorelDRAW[®]. In CorelDRAW, you can import Encapsulated PostScript files (*File*> *Import* command in CorelDRAW) and translate it to the vector format of CorelDRAW (select the file type “PostScript Interpreted PS, PRN, EPS”). If you ungroup the imported drawing, you can change each element individually. Thus, you can completely customize the sociogram. In Figure 26 in Chapter 2, for instance, we combined the sociogram of world trade in miscellaneous manufactures of metal with a line drawing of the continents (available as `outline.wmf`). In addition, drawing software such as CorelDRAW can export the drawing to a format that can be embedded and viewed in most Windows text processors. We advise using the Windows MetaFile (WMF) format, which is also vector based.

Inkscape (www.inkscape.org/) is another powerful program for editing EPS pictures. This program is free and will be introduced as an editor (also) for SVG in the next subsection.

A2.1.3 Scalable Vector Graphics

Export> *2D*> *SVG* What PostScript is to printing, Scalable Vector Graphics (SVG) is to publishing on the web. Just as with PostScript, SVG defines each distinct shape and assembles the drawing at the moment it is viewed, so it produces optimal quality in every resolution. SVG is XML-based. In addition, SVG pictures can be changed interactively on the web. As a consequence, there are several commands for exporting SVG in Pajek’s *Export*> *2D*> *SVG* submenu, each of which adds a specific type of interaction to the drawing.

Export> *2D*> *SVG*> *General* The most basic way of exporting a network as SVG is provided by the *Export*> *SVG*> *General* command, which produces an image of the network as it is shown in the Draw screen, without interactivity. In the *Save As* dialog box, enter a filename, which will be used for both the SVG file and the HTML file (webpage) within which the drawing will be displayed. Note that you need both files (in the same directory) to view the drawing. The second SVG export command, *Labels/Arcs/Edges*, exports the drawing with checkboxes, which enable the user to toggle the display of arcs, edges, arc labels, edge labels, and vertex labels.

Export> *2D*> *SVG*> *Partition*> *Classes* When the network is drawn with a partition, the *Export*> *2D*> *SVG*> *Partition* submenu contains options for interactively manipulating the classes of the partition. The *Classes* command produces checkboxes with which vertices belonging to a class can be hidden (or displayed), including

the lines among them. The lines between different classes can be hidden separately. The *Classes with semi-lines* command produces similar output, but lines incident with vertices that are hidden are changed to semi-lines (half lines) and semi-lines are hidden on removal of the vertices with which they are incident. The *Nested Classes* command adds a checkbox for each class in the partition, but toggling one checkbox automatically toggles the value of all lower classes. This is very useful for displaying nested classes, such as *k*-cores (Chapter 3). Note that it is possible to display two partitions in the SVG export. If two partitions are selected in the Main screen, the first partition defines the classes that can be selected by means of the checkboxes, whereas the second partition determines the colors of the vertices in the SVG drawing.

If the network contains lines with different values, the SVG export can create checkboxes for showing or hiding lines with particular values. The *Export> 2D> SVG> Line Values* submenu contains commands for exporting ordinary classes (*Classes*) and nested classes (*Nested Classes*). Line values must be grouped into classes, so a dialog box appears asking for the number of classes or the class boundaries. In addition, the submenu contains options for the appearance of lines (*Export> 2D> SVG> Line Values> Options*), which can receive different colors (*Different Colors* option) or grays (*GrayScale* option), or the line width can be made to represent the class of line values (*Different Widths* option).

Instead of controlling parts of the picture in HTML, you can also export a standalone SVG file with embedded controls. To this end, select the type *SVG with Controls* in the *Save as Type* list box of the standard *Save As* dialog box. In this case, you do not need an HTML file because the commands to show or hide parts of the picture are included in the SVG. Note that this kind of SVG export is not available with the *Nested Classes* options.

It is also possible to create a series of SVG images, all linked by previous and next buttons, for a list of networks, partitions, or vectors. First, make sure that the right data objects are selected in the *Options> Previous/Next> Apply to* submenu of the Draw screen. If you have only a sequence of networks to display, only the *Network* option should be selected, but if you have a sequence of networks and a sequence of partitions, *Network + Partition* should be selected. Then select the first network, partition, or vector that you want to export in the drop-down menus of the Main screen, draw the network, and, if applicable, partition and vector. Finally, select the *Export> 2D> SVG> Current and all Subsequent* option and execute one of the *Export> 2D> SVG* commands discussed previously.

Finally, a Pajek helper program generates SVG animations of changes in temporal networks over time. The helper program is *PajekToSvgAnim.exe*; it is available from the Pajek website. The input to this

*Export> 2D>
SVG>
Partition>
Classes with
semi-lines*

*Export> 2D>
SVG>
Partition>
Nested Classes*

*Export>
SVG> 2D>
Line Values>
Classes*

*Export>
SVG> 2D>
Line Values>
Nested Classes*

*Export>
SVG> 2D>
Line Values>
Options*

*[Draw]
Options>
Previous/Next>
Apply to*

*Export> 2D>
SVG> Current
and all
Subsequent*

*Export>
Append to
Pajek Project
File> Select File*

*Export>
Append to
Pajek Project
File> Append*

*[Main]
Network>
Temporal
Network>
Generate in
Time*

program is a Pajek project file formatted in a special way. Let us illustrate the construction of this Pajek project file with the monastery network of Chapter 4: `Sampson.paj`. In Chapter 4, we generated the network at times T_2 , T_3 , and T_4 with the command `Network> Temporal Network> Generate in Time` applied to the temporal network `Sampson.net`. We recommend you first optimize the layout of the original temporal network in the Draw screen – in this way initial positions of vertices in all time slices are generated. The original temporal network must be included as the first network in the Pajek project file for `PajekToSvgAnim`. In the Draw screen, give a name to the project file with `Export> Append to Pajek Project File> Select File` and save the original temporal network to the project file with `Export> Append to Pajek Project File> Append` or press `F3`. Next, use this command to add the networks for each time slice that you are interested in to the Pajek project file. `PajekToSvgAnim` uses the coordinates of the time slice networks in the animation. In our example, select the network at T_2 in the Draw screen, adjust vertex positions if necessary, and add the network to the Pajek project file. Repeat this for the networks at T_3 and T_4 . Now the project file is ready to be loaded by `PajekToSvgAnim.exe`, which generates an animated SVG. For details on using `PajekToSvgAnim.exe` look at its documentation available with the program. Several animations obtained in this way can be found on Pajek webpage.

SVG drawings are viewed in an Internet browser. As mentioned, the SVG drawing can be inserted in an HTML page, which is read by a web browser. Most browsers can display SVG images directly.

Onscreen editing of SVG images can be done with several software packages, for example, Adobe Illustrator (www.adobe.com/products/illustrator/main.html) or Inkscape (www.inkscape.org/). Inkscape is a free, open-source SVG editor. It enables several useful improvements to the picture, such as drawing contours and shaded areas with transparent colors around selected vertices, adding legends, and adjusting labels. It can also transform an SVG picture to other formats, such as EPS and PDF.

A2.1.4 VOSviewer

*Export> 2D>
VOSviewer*

VOSviewer (www.vosviewer.com/) is another software for analyzing and visualizing networks. It includes some powerful visualization methods such as *Item Density View* and *Cluster Density View* that are not available in Pajek. If VOSviewer is installed on your computer, the command `Export> 2D> VOSviewer> Send to VOSviewer` sends the active network directly from Pajek to VOSviewer. If the active partition or vector, or both, are compatible with the active network, they will be exported to and used by VOSviewer as well. For example, a network exported with a

partition is depicted with *Cluster Density Visualization* in VOSviewer. If Pajek cannot find VOSviewer, locate it on your computer with the command *Export> 2D> VOSviewer> Locate VOSviewer*.

A2.1.5 Virtual Reality Modeling Language and X3D

Virtual Reality Modeling Language (VRML) and Extensible 3D (X3D) are standard languages for defining three-dimensional objects that can be published on the web. In a web browser, VRML and X3D models, for example, a three-dimensional model of a social network, can be rotated, walked through, and so on. You may lay out a network in three dimensions with commands like *Layout> Energy> Fruchterman-Reingold> 3D*, *Layout> Pivot MDS> 3D* or *Layout> VOS Mapping> 3D* before you export a VRML or X3D model from Pajek.

With the *Export> 3D> VRML* command, Pajek exports the network to VRML 1.0. A dialog box prompts for a file name in which to store the model. We advise using the default extension `.wrl`, which stands for world, so Internet browsers can correctly identify the file type. If the network was drawn with a partition, the spheres in the model, which represent the vertices, have the right colors. Vertex labels will be transformed to anchor names.

For displaying a VRML world in a web browser, you need a plug in, for instance, Cortona VRML Client (www.parallelgraphics.com/products/downloads). Also a minimalistic and portable standalone viewer for VRML 1.0 and 2.0 – Qiew is available for free (www.qiew.org/). However, some plug-ins can read only newer VRML models (VRML 2.0 or 97). If so, you need a converter such as the Cortona VRML 1.0 converter or the MS-DOS program `vrm11to2.exe` (available from the book's website), which must be run from the MS-DOS prompt with the command `vrm11to2` followed by the names of the VRML 1.0 file and the new VRML 2.0 file separated by spaces. The original vertex names, which are anchor names in the VRML model, may cause problems; if this happens, open the VRML file in a plain text editor and adjust or remove the anchor names. Do not forget to save the changed file to plain text.

Some layout properties of VRML and X3D models can be set in Pajek (see Section A2.2). If you want to edit a VRML model onscreen, you need special software. Consult www.web3d.org/x3d/vrml/tools/authoring/ for VRML design and editing software.

If you want to include an image of a three-dimensional model in a document, it is a good idea to ray-trace it. Ray-tracing is a technique that renders very realistic images from three-dimensional models by calculating (tracing) the trajectory of light from the virtual light sources and the reflection of this light on the surfaces in the model. In Chapter 1, Figure 14 provides an example. POVRay is a good and free software

Layout>
Energy>
Fruchterman-
Reingold> 3D

Layout> Pivot
MDS> 3D

Layout> VOS
Mapping> 3D

Export> 3D>
VRML

package for ray-tracing (www.povray.org/). However, it does not understand the Virtual Reality Modeling language directly, so you must first transform the VRML 2.0 model to the POVray language, which can be done with the MS-DOS program `vrm12pov.exe` (available from the book's website). In the MS-DOS prompt, issue the command `vrm12pov` followed by the names of the VRML 2.0 file and the new POVray file, separated by spaces. For the POVray file, use the extension `.pov`. You can open the newly created file in POVray and render it. The rendered image is automatically saved in the format specified in POVray's INI file. Enjoy, but realize that it may take quite some time before you master the software and achieve satisfactory results.

Export> 3D>
X3D Unfortunately VRML is not supported anymore, but export to VRML is still available in Pajek for backward compatibility. In addition to VRML, Pajek exports 3D images also to X3D. X3D is an XML-based 3D computer graphics, the successor of VRML. There are several viewers for X3D available, for example, Instant Player (www.instantreality.org/home/) and Flux Player (<http://flux-player.software.informer.com/>). Instead of VRML we recommend to use X3D format for 3D models because 3D printing services such as shapeways (www.shapeways.com/) support this format. Isn't it nice to get a 3D model of your network that you can touch with your fingers?

A2.1.6 MDL MOL and Kinemages

Export> 3D>
MDL MOLfile Molecular structures can be modeled as networks, so the software developed by scientists to visualize molecules can be used for the display of social networks. Pajek can export to two molecule model formats: MDL MOL and Kinemages. MDL MOL models are three-dimensional, and they are viewed in a web browser using a special plugin (Chime, which can be downloaded from www.umass.edu/microbio/chime/getchime.htm). This format does not offer real advantages over X3D or VRML, so we do not discuss it in detail.

In contrast, the other format – Kinemages – does have special features that need to be mentioned. This format is designed to present sequences of three-dimensional images. In principle, this can also be done in X3D and VRML, but in Kinemages, it is very easy to add text to each image and to include questions. This software is excellent for teaching purposes and for electronic publishing. For signed networks, Kinemage is less useful because it does not display the sign of lines.

Export> 3D>
Kinemages>
Current and all
Subsequent Pajek can produce these sequences with the *Export> 3D> Kinemages>* *Current and all Subsequent* command. Just like SVG export, you must select the data objects (any combination of network, partition, and vector) that you want to export in the *Options> Previous/Next> Apply to* submenu of the Draw screen, and you must select the first network

[Draw]
Options>
Previous/Next>
Apply to

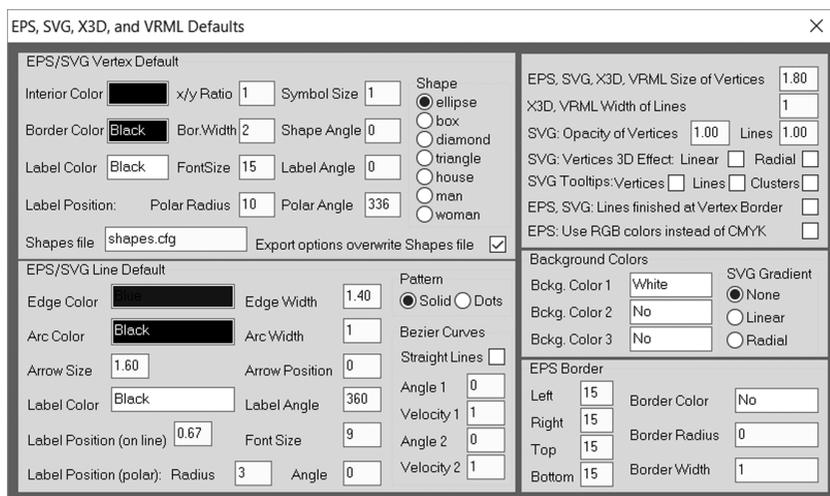


Figure 145. The *Options* screen.

(partition, vector) that you want to export in the Main screen. With the *Export> 3D> Kinemages> Current and all Subsequent* command, Pajek creates one Kinemage file (default extension is `.kin`) containing the series of networks. In the *Save as Type* selection box of the *Save As* dialog box, you can choose whether you want the vertices represented by spheres, vertex labels, or balls in the Kinemage.

You need special software for viewing Kinemages, which is called Mage (<http://kinemage.biochem.duke.edu/software/mage.php>). Open the Kinemage created by Pajek in Mage, and step through the sequence of images with the *KINEMAGE/Next* (or *Ctrl-n*) command in Mage. An even more powerful software for viewing Kinemages is KiNG, which can be downloaded from the same page (<http://kinemage.biochem.duke.edu/software/king.php>).

A2.2 Layout Options

PostScript, SVG, X3D, and VRML visualizations need additional information about the size, placement, and colors of vertices, lines, labels, and background. This information can be supplied either with the *Options* command in the *Export* menu of the Draw screen or in the network data file. The *Export> Options* (Figure 145) define layout properties of all vertices and lines, whereas the parameters in the network data file define the layout of each individual vertex or line. Layout parameters specified in the network data file override the settings in the *Export>*

[Draw]
Export>
Options

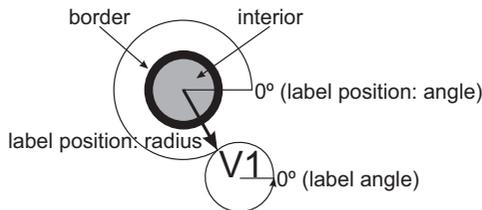


Figure 146. Layout of a vertex and its label.

Options dialog screen. If, for instance, the color of vertices is specified in the network data file, entering a color in the *Options* window has no effect.

The *Export> Options* command opens a window that is divided into five frames; two on the left and three on the right (Figure 145). We discuss each frame separately; and, where appropriate, we mention the layout parameter that defines the layout feature in the network data file. Note that you should type the layout parameters at the end of the line defining a vertex, edge, or arc in the network data file, and you must separate the parameters and their values by one or more blanks. Consult the document `pajekman.pdf` (on the Pajek website) for a description of all layout parameters in the network data file.

[Main] File>
Ini File> Save
[Main] File>
Ini File> Load

If you want to save the settings of a particular layout, use the *File> Ini File> Save* command in the Main screen. This command saves all present settings to a file, which can be loaded (command *File> Ini File> Load*) to restore the settings. In this way, you can prepare several INI files for different types of networks.

A2.2.1 Top Frame on the Left – EPS/SVG Vertex Default

This frame defines how vertices are drawn when you export 2D layouts to (Encapsulated) PostScript and Scalable Vector Graphics. Figure 146 shows some important properties of the layout of a vertex and its label.

[Draw]
Export>
Options>
Interior Color

In the top frame on the left, the color of the interior (*Interior Color*; parameter `ic` followed by a color) and border (*Border Color*; parameter `bc` followed by a color) of the vertices can be specified. Click a color and select one of the colors that is listed in Table 27 (the colors are shown in `pajekman.pdf`, which is included in the documentation accompanying Pajek). In Pajek input files you can also specify any color using RGB (Red–Green–Blue) or CMYK (Cyan–Magenta–Yellow–black) formats. In both formats a color is defined by a sequence of three (RGB) or four (CMYK) decimal numbers within the interval 0.00 to 1.00, or hexadecimal numbers in interval 00 to FF. For example, a *goldenrod* interior color can be described in four different

[Draw]
Export>
Options>
Border Color
[Draw]
Export>
Options> Bor.
Width

Table 27. Names of colors in Pajek

Apricot	Gray60	Periwinkle
Aquamarine	Gray65	PineGreen
Bittersweet	Gray70	Pink
Black	Gray75	Plum
Blue	Gray80	ProcessBlue
BlueGreen	Gray85	Purple
BlueViolet	Gray90	RawSienna
BrickRed	Gray95	Red
Brown	Green	RedOrange
BurntOrange	GreenYellow	RedViolet
CadetBlue	JungleGreen	Rhodamine
Canary	Lavender	RoyalBlue
CarnationPink	LFadedGreen	RoyalPurple
Cerulean	LightCyan	RubineRed
CornflowerBlue	LightGreen	Salmon
Cyan	LightMagenta	SeaGreen
Dandelion	LightOrange	Sepia
DarkOrchid	LightPurple	SkyBlue
Emerald	LightYellow	SpringGreen
Fuchsia	LimeGreen	Tan
Goldenrod	LSkyBlue	TealBlue
Gray	Magenta	Thistle
Gray05	Mahogany	Turquoise
Gray10	Maroon	Violet
Gray15	Melon	VioletRed
Gray20	MidnightBlue	White
Gray25	Mulberry	WildStrawberry
Gray30	NavyBlue	Yellow
Gray35	OliveGreen	YellowGreen
Gray40	Orange	YellowOrange
Gray45	OrangeRed	ForestGreen
Gray50	Orchid	
Gray55	Peach	

ways: `ic RGBFFCC00`, `ic RGB(1,0.8,0)`; `ic CMYK0033FF00`, `ic CMYK(0,0.2,1,0)`. Note that the string that defines a color should not contain any spaces. Finally, the width of the border (*Bor. Width*) can be specified by a single number following the parameter `bw` in the network data file (e.g., `bw 1.5`).

The user may choose among the following predefined shapes of vertices: `ellipse`, `box`, `diamond`, `triangle`, `house`, `man`, and `woman`. To change the shape of a vertex, type `ellipse`, `box`, `diamond`, `triangle`, `cross`, `house`, `man`, `woman`, or `empty` after the vertex's coordinates but before any other layout parameters in the network data file. Figure 147 presents an example visualization using `house`, `man`, and `woman` shapes. Note that the shapes `house`, `man`, and `woman` are displayed only

[Draw]
Export>
Options>
Shape

[Draw]
Export>
Options>
Shape Angle

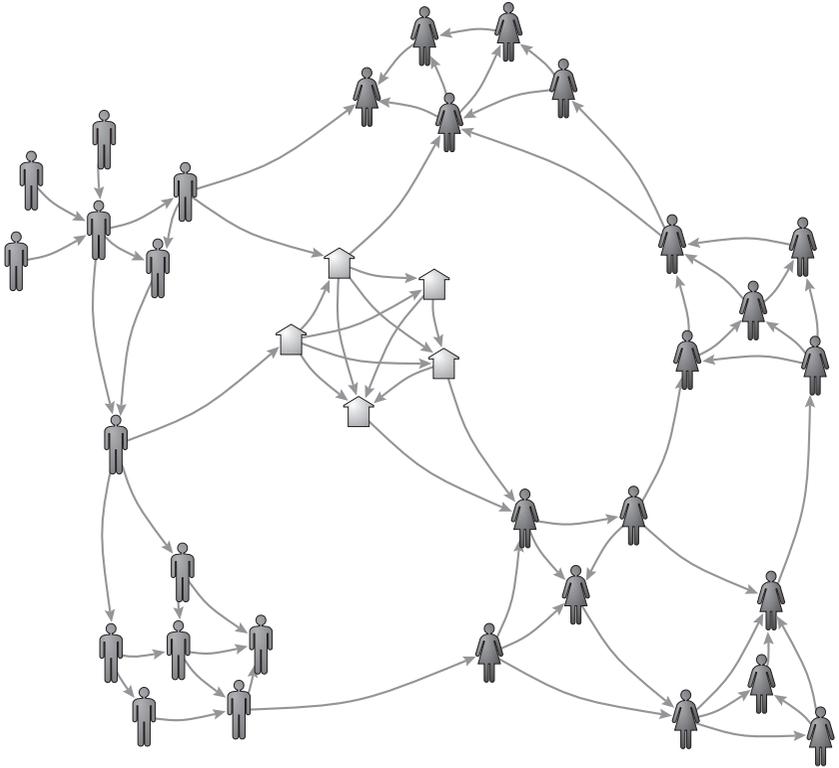


Figure 147. Bezier curves and different vertex shapes (man, woman, and house).

when visualization is exported to EPS or SVG. In the Draw window, house is replaced by a triangle, man by a box, and woman by an ellipse to achieve faster rendering of large networks. Shapes can also be rotated, just provide the angle in degrees (*Shape Angle*, parameter ϕ followed by a number indicating the degree).

[Draw]
Export>
Options> x/y
Ratio

The user may squeeze or stretch the shape horizontally by adjusting the x/y ratio (x/y Ratio; parameters x_fact and y_fact followed by a number). If this ratio is smaller than 1, the shape of the vertex is squeezed; if it is larger than 1, it is stretched. In Figure 148, the horizontal diameter of the vertex on the left is half the vertical diameter, so its x/y ratio is 0.5. The horizontal diameter of the other vertex is 50 percent longer than the vertical diameter.


x/y ratio: 0.5 1.5

Figure 148. The x/y ratio of a vertex.

The color (*Label Color*; parameter `lc` followed by a color name or RGB/CMYK color sequence) and font size (*FontSize*; parameter `fos` followed by a number) of vertex labels can also be adjusted in this frame. Again, use names of colors as listed in Table 27. In addition, the orientation of the vertex label (*Label Angle*; `la` followed by a number indicating the degree) can be changed relative to a horizontal line. Zero degrees is usually the best choice because this will display the labels horizontally. Label angles from 360 to 720 degrees and position the label relative to the center of the layout, which can be useful when all vertices are drawn in concentric circles.

The most complicated part, however, is the position of the label with respect to the vertex. The position of the label is defined by its distance to the center of the vertex (the radius) and by the angle from the horizontal line starting at this center, ranging from 0 to 360 degrees (*Label Position: Polar Radius / Polar Angle*; parameters `lr` followed by a number and `lphi` followed by a degree, respectively). If the angle of the label position is 0 degrees, the label is resting on this horizontal line at the right of the vertex. If the radius is 0, the label is placed in the center of the vertex. With short labels and large vertices (see Section A2.2.3) or a high x/y ratio, the label may fit inside the vertex. To place labels outside but near their vertices, enter a positive number in the *Label Position: Polar Radius* field. It is difficult to give a general rule for a good size of the radius, because it depends on the size of the vertex and the size of the label's type font. Most figures of this book used the sizes shown in Figure 145.

Finally, this window contains a field in which a file can be selected with custom shapes for vertices. The default is the file `shapes.cfg`. You can prepare custom shapes files but this requires knowledge of the PostScript language. We recommend beginners to always select *Export options overwrite Shapes file*. In this case, the options that are set in the *Export> Options* window will be the ones that Pajek uses when exporting visualizations (the shapes file will not be used at all).

There is a more flexible way to use different shapes for vertices. In Chapter 2, we learned that we can represent the classes of the second partition by Unicode symbols with the command *Options> Mark Vertices Using> Cluster Symbols of Second Partition* in the Draw screen. The size of symbols in the exported EPS or SVG (not in the Draw screen) can be adjusted in two ways with the command *Export> Options> Symbol Size*. If symbol size is set to a positive value, for example, 0.75, symbol size is relative to vertex size. In the example, symbol size would be 75 percent of vertex size. Vertices with different sizes also have different symbol sizes. In contrast, a negative value for symbol size, for example, -20, is interpreted as an absolute size in pixels for all symbols. Vertices with different sizes have the symbols of the same size, in the example, a size of 20 pixels. A size of 20 pixels is recommended for

[Draw]
Export>
Options>
Label Color

[Draw]
Export>
Options>
FontSize

[Draw]
Export>
Options>
Label Angle

[Draw]
Export>
Options>
Label Position:
Polar Radius /
Polar Angle

[Draw]
Export>
Options>
Shapes file

[Draw]
Export>
Options>
Export options
overwrite
Shapes file

[Draw]
Options>
Mark Vertices
Using> Cluster
Symbols of
Second
Partition

[Draw]
Export>
Options>
Symbol Size

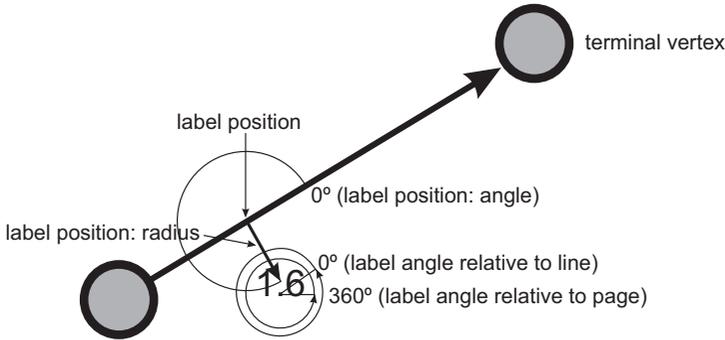


Figure 150. The position and orientation of a line label.

The colors of edges and arcs (*Edge Color* and *Arc Color*; parameter *c* followed by a color name, or RGB/CMYK color sequence) and labels (*Label Color*; *lc* followed by a color name, or RGB/CMYK color sequence) can be changed by selecting one of colors as they appear in Table 27. Note that arcs and edges may receive different colors. You may draw all lines as dotted lines by selecting *Dots* in the *Pattern* field (parameter *p* followed by *Solid* or *Dots*). The width of edges and arcs (*Edge Width* and *Arc Width*; *w* followed by a number) may also differ; it is good practice to draw edges a bit wider than arcs. The size of the arrowhead (*Arrow Size*; *s* followed by a number) is specified independently of the arc's width. The size of the labels is defined by their font size (*Font Size*; *fos* followed by a number), which should be smaller than the font size of the vertex labels for pleasant results.

[Draw]
 Export>
 Options> Edge
 Color, Arc
 Color, Label
 Color, Pattern,
 Edge Width,
 Arc Width,
 Arrow Size,
 Font Size,
 Arrow Position

You may decide where to place the arrowhead on the line (*Arrow Position*; *ap* followed by a number). A number between 0 and 1 is interpreted as a proportion expressing the relative distance from the end to the beginning of the arc; for example, 0 will place the arrowhead at the end of the arc near the terminal vertex and 0.5 will situate the arrowhead in the middle of the arc. A distance larger than 1 is interpreted as an absolute distance from the terminal vertex. This is useful if you want to have all arcs on the same distance from the terminal vertex, regardless of the arc's length. You have to experiment a little to find a nice distance.

The remaining fields concern the orientation and position of the line label (e.g., its value) with respect to the line. Figure 150 illustrates the relevant parameters. First, choose the position on the line (edge or arc) to which the line label is attached (*Label Position (on line)*; *lp* followed by a number). This is similar to the position of the arrowhead: Numbers between 0 and 1 are proportions, that is, relative distances from the end to the beginning of the line, and numbers above 1 are absolute distances from the end of the line. In Figure 150, the label position is 0.67, so it is

[Draw]
 Export>
 Options>
 Label Position
 (on line)

located at two-thirds of the arc, measured from its end. Note that edges do not have a start and an end, so it is most appropriate to position the edge labels halfway.

```
[Draw]
Export>
Options>
Label Position
(polar): Radius
```

Next, the location of the center of the label with respect to the position on the line is defined by two properties that are similar to the location of vertex labels: radius and angle. The radius (*Label Position (polar): Radius*; `1r` followed by a number) is the distance between the position on the line and the center of the line label that is measured at the specified angle (*Label Position (polar): Angle*; `1phi` followed by a number of degrees) from the line. In Figure 148, the angle is 270 degrees.

```
[Draw]
Export>
Options>
Label Position
(polar): Angle
```

Finally, the orientation of the line label is defined in the *Label Angle* field (parameter `1a` followed by a number of degrees). An angle smaller than 360 degrees is measured relative to the direction of the line, where 0 degrees is parallel to the line. Angles of 360 degrees and more are relative to a horizontal line. For easy reading, an angle of 360 degrees is optimal because it displays line labels horizontally.

```
[Draw]
Export>
Options>
Label Angle
```

If two vertices are linked by two or more lines, Pajek automatically draws lines as curves instead of straight lines in its output to EPS and SVG. You can avoid this by checking *Straight Lines* in this frame, which draws multiple lines as straight lines one on top of the other. Some people prefer to have curved instead of straight lines always. All lines are curved in EPS and SVG output if you select suitable Bezier parameters: *Angle1* and *Velocity1* to define the angle (in degrees) and velocity at the initial vertex and *Angle2* and *Velocity2* to define angle and velocity at the terminal vertex. In Figure 147, *Angle1* and *Angle2* were set to 10, *Velocity1* and *Velocity2* were set to 1. Check the Pajek website for more examples of drawing lines as Bezier curves.

```
[Draw]
Export>
Options>
Bezier Curves>
Straight Lines
```

```
[Draw]
Export>
Options>
Bezier Curves>
Angle1,
Velocity1,
Angle2,
Velocity2
```

A2.2.3 Top Frame on the Right

```
[Draw]
Export>
Options> EPS,
SVG, X3D,
VRML Size of
Vertices
```

This frame defines additional defaults when we export layouts to EPS, SVG, X3D, and VRML. The field *EPS*, *SVG*, *VRML Size of Vertices* specifies the default size of vertices when exporting to EPS, SVG, X3D, and VRML. The diameter of lines in X3D and VRML can be changed in the *X3D*, *VRML Width of Lines* (parameter `w` followed by a number) field. It is difficult to give general rules about optimal settings for these fields; you have to experiment.

```
[Draw]
Export>
Options>
X3D, VRML
Width of Lines
```

When exporting to SVG, vertices and lines can be drawn (partially) transparent. Transparency (opacity) is a number between 0.00 (vertices or lines drawn fully transparent) and 1.00 (vertices or lines drawn fully opaque – default). Visit the Pajek website for examples of using transparencies in visualizations.

```
[Draw] Export>
Options> SVG: Opacity
of Vertices, Lines
```

```
[Draw] Export>
Options> SVG:
Vertices 3D Effect
Linear, Radial
```

SVG also enables shading of vertices using gradient colors, which gives them a 3D effect. To enable shading, *Vertices 3D Effect Linear* or *Vertices 3D Effect Radial* (used in Figure 147) should be checked. To get a more

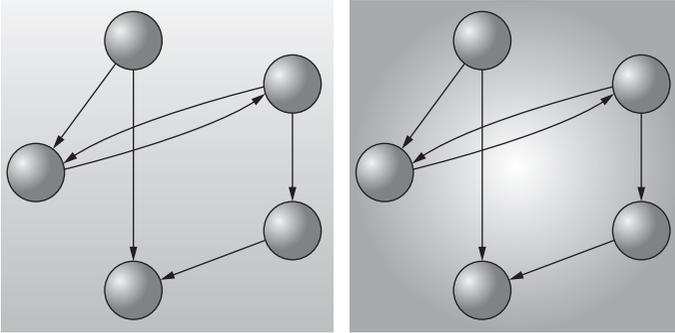


Figure 151. Gradients in SVG export: linear (left) and radial (right).

realistic 3D effect, we recommend to set border width of vertices to 0. Simply try both options to see the difference. More details on gradients are provided in the text that follows.

SVG can display labels of vertices, lines, and clusters only if the mouse pointer hovers over the vertex, line, or cluster. This is called a tooltip. The checkboxes for SVG tooltips in the *Export > Options* dialog allow the user to display labels as tooltips. This is particularly useful when networks are dense and labels overlap.

By default all lines start at the center of the vertex. When vertices are (partly) transparent, lines are visible inside vertices and this is usually not what we want. With transparent vertices, it is better for the lines to start at the vertex border instead of the center of the vertex.

If this option is checked, the RGB color model will be used instead of the default CMYK color model in EPS export.

[Draw] Export>
Options> SVG
Tooltips:
Vertices, Lines,
Clusters

[Draw] Export>
Options> EPS,
SVG: Lines
finished at
Vertex Border

[Draw] Export>
Options> EPS:
Use RGB
colors instead
of CMYK

A2.2.4 Middle Frame on the Right – Background Colors

In this frame, we can select the color that will be used as background color for EPS, SVG, X3D, and VRML export. In the case of exporting to SVG and X3D, we can also select additional parameters that define gradients: continuously smooth color transitions from one color to another.

These layouts can handle gradient colors: an area in which one color gradually blends into another color. You can specify three colors. The first color is the background color defined in *Bckg. Color 1* field. The second color (used in SVG and X3D only) can be selected in the *Bckg. Color 2* field. The list of possible colors is the same as before (use *No* if you do not want to use a second color and hence no gradient). If you like, you can select a third color in the *Bckg. Color 3* field. The type of blend can be selected in the *SVG Gradient* field: *Linear* or *Radial* (see Figure 151). In a linear gradient, the original background color is at the top of the area and blends into the second and third colors on its way down. In

[Draw] Export>
Options>
Bckg. Color 1

[Draw] Export>
Options> Bckg.
Color 2

[Draw] Export>
Options> Bckg.
Color 3

[Draw] Export>
Options> SVG
Gradient

a radial gradient, the middle of the drawing is colored with the original background color. Figure 151 also shows the effects of the SVG: *Vertices 3D Effect* option set to Radial.

A2.2.5 Bottom Frame on the Right – EPS Border

- [Draw] Export>
Options> EPS
Border
- This frame defines additional properties of the network layout when we export layouts to Encapsulated PostScript (there are no layout parameters for the network data file). Most fields relate to a border around the layout. With the four fields *Left*, *Right*, *Top*, and *Bottom*, you can add additional space to the left, right and so on of the picture. This is effective for for EPS exports when the EPS Clip format is selected.
- [Draw] Export>
Options>
Border Color
- The color of the border (*Border Color*) can be picked from the color names listed in Table 27. Enter the word *No* in these fields if you do not want a border or background color. The shape of the borderline is characterized by the fields *Border Radius* and *Border Width*. If the radius of the border is 0, the border is a rectangle. Higher values (e.g., 10, 50, or 100) round off the corners. A border width of 1 unit produces a rather thin borderline and higher values yield thicker borderlines.
- [Draw] Export>
Options>
Border Radius
- [Draw] Export>
Options>
Border Width

Appendix 3

Installing Pajek on Mac OS X

As noted earlier, Pajek is a native Windows application. However, it is quite easy to set up Pajek to be used on other platforms as well, e.g., Mac, Linux, Linux64. In this appendix, we provide detailed instructions on how to install and run Pajek on Mac OS X using the installation guide prepared by Miha Gornik. Compared to a Windows installation, some more steps are needed. These additional steps will take you only a few minutes. Check the book webpage for instructions how to install Pajek on Linux and Linux64.

First you should download and install the latest version of *XQuartz* on your computer. *XQuartz* is required for running *WineHQ*, which offers the environment to run Pajek on Mac OS X. You can download *XQuartz* from www.xquartz.org/ (Figure 152).

Next download and install *WineHQ*. You can download *WineHQ* from <https://dl.winehq.org/winebuilds/macosx/download.html>. We recommend selecting *Installer for “Wine Stable”* (Figure 153).

Now we are ready to install Pajek on Mac OS X. You can download the latest version of Pajek from <http://mrvar.fdv.uni-lj.si/pajek/>.

Download the archive labeled *Install-Zip* from the Pajek webpage (see Figure 154) to a folder on your computer, e.g., *Downloads*, and extract all files from the zip archive. Use the default archive application *Archive Utility* on your Mac, which will create a folder named *Pajek32* or *Pajek64* depending on the version of Pajek that you have downloaded (32- or 64-bit). Install Pajek by copying the contents from this folder to any folder for which you have write permissions; the folder *Applications* is usually fine.

You can start using a Pajek application (*Pajek.exe*, *PajekXXL.exe*, or *Pajek3XL.exe*) by double-clicking the corresponding file or icon. The first time you try to launch Pajek, you may get the error message *Wine did not find Gecko package*. The message offers an option to install the *Gecko* package: Simply click *Install*. If the download of the *Gecko*

The XQuartz project is an open-source effort to develop a version of the [X.Org X Window System](#) that runs on OS X. Together with supporting libraries and applications, it forms the X11.app that Apple shipped with OS X versions 10.5 through 10.7.

Quick Download

Download	Version	Released	Info
 XQuartz-2.7.11.dmg	2.7.11	2016-10-29	For OS X 10.6.3 or later

License Info

An XQuartz installation consists of many individual pieces of software which have various licenses. The X.Org software components' licenses are discussed on the [X.Org Foundation Licenses page](#). The [quartz-wm](#) window manager included with the XQuartz distribution uses the [Apple Public Source License Version 2](#).

Web site based on a design by Kyle J. McKay for the XQuartz project.
 Web site content distribution services provided by [CloudFlare](#).
 Distributed by

Figure 152. XQuartz webpage.

WINEHQ
 Run Windows applications on Linux, BSD, Solaris and Mac OS X.

Wine Download Server

Name	Latest version	Size
 Installer for "Wine Stable"	2.0.1	109M
 Installer for "Wine Development"	2.12	111M
 Installer for "Wine Staging"	2.12	120M

Figure 153. WineHQ webpage.

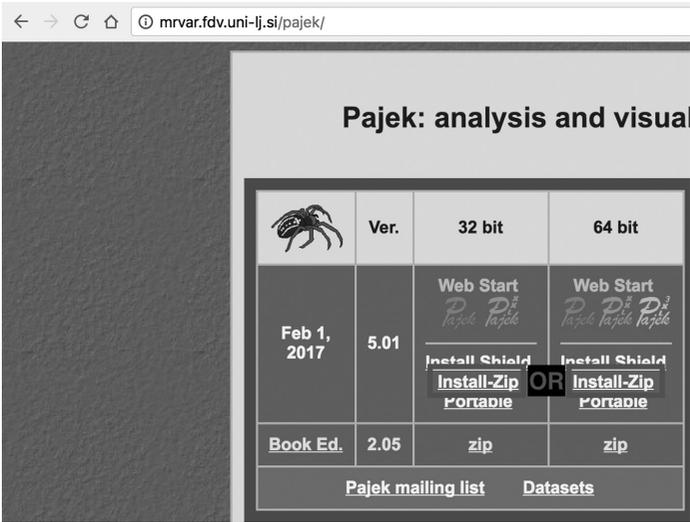


Figure 154. Pajek webpage.

package fails, it can be downloaded manually from <http://dl.winehq.org/wine/wine-gecko/>. We recommend selecting the latest non-beta version. You can check which Gecko package is compatible with your Wine version here: <https://wiki.winehq.org/Gecko>.

Now everything is ready to start working with Pajek (Figure 155).

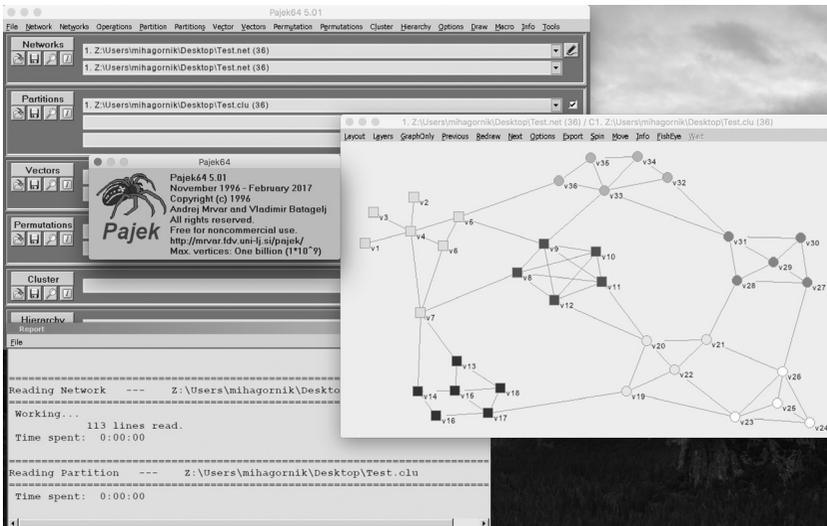


Figure 155. Pajek running on Mac OS X.

Appendix 4

Shortcut Key Combinations

The following tables list the shortcut key combinations and the commands they replace. Some shortcuts are accessible from one screen, whereas others are accessible from several screens.

A4.1 Main Screen

Shortcut	Description and Command
<i>Ctrl-a</i>	Draw the network with all vertices in class 0 of a new empty partition (<i>Draw</i> > <i>Network</i> + <i>Create Null Partition</i>).
<i>Ctrl-e</i>	Extract SubNetwork (<i>Operations</i> > <i>Network</i> + <i>Partition</i> > <i>Extract</i> > <i>SubNetwork Induced by Union of Selected Clusters</i>). The result always consists of only one extracted subnetwork induced by all selected clusters.
<i>Ctrl-f</i>	Extract SubNetworks (<i>Operations</i> > <i>Network</i> + <i>Partition</i> > <i>Extract</i> > <i>SubNetworks Induced by Each Selected Cluster Separately</i>). The result consists of as many extracted subnetworks as the number of selected clusters.
<i>Ctrl-g</i>	Draw the network (<i>Draw</i> > <i>Network</i>).
<i>Ctrl-i</i>	Save the temporal network using time events format (<i>File</i> > <i>Network</i> > <i>Save as Time Events</i>).
<i>Ctrl-m</i>	Play the macro (<i>Macro</i> > <i>Play</i>).
<i>Ctrl-n</i>	Compute all k-neighbors of the selected vertex (<i>Network</i> > <i>Create Partition</i> > <i>k-Neighbours</i> > <i>All</i>)
<i>Ctrl-o</i>	Compute all core partitions (<i>Network</i> > <i>Create Partition</i> > <i>k-Core</i> > <i>All</i>).
<i>Ctrl-p</i>	Draw the network and the active partition (<i>Draw</i> > <i>Network</i> + <i>First Partition</i>).
<i>Ctrl-q</i>	Draw the network, the active partition, and the active vector (<i>Draw</i> > <i>Network</i> + <i>First Partition</i> + <i>First Vector</i>).
<i>Ctrl-s</i>	Repeat the session (<i>Macro</i> > <i>Repeat Session</i>).
<i>Ctrl-t</i>	Read the temporal network, which is stored using time events (<i>File</i> > <i>Network</i> > <i>Read Time Events</i>).
<i>Ctrl-u</i>	Draw the network and the active vector (<i>Draw</i> > <i>Network</i> + <i>First Vector</i>).
<i>Ctrl-w</i>	Compute weakly connected components (<i>Network</i> > <i>Create Partition</i> > <i>Components</i> > <i>Weak</i>).
<i>Ctrl-x</i>	Create a new network with the <i>x</i> coordinates of the vertices from the active vector (<i>Operations</i> > <i>Network</i> + <i>Vector</i> > <i>Transform</i> > <i>Put Coordinate</i> > <i>x</i>).

(continued)

Shortcut	Description and Command
<i>Ctrl-y</i>	Create a new network with the y coordinates of the vertices from the active vector (<i>Operations> Network + Vector> Transform> Put Coordinate> y</i>).
<i>Ctrl-z</i>	Create a new network with the z coordinates of the vertices from the active vector (<i>Operations> Network + Vector> Transform> Put Coordinate> z</i>).
<i>F1</i>	Read the Pajek project file (<i>File> Pajek Project File> Read</i>).
<i>F2</i>	Save all objects available in the main window as the Pajek project file (<i>File> Pajek Project File> Save</i>).
<i>F3</i>	Make a permutation from the active partition (<i>Partition> Make Permutation</i>).
<i>F4</i>	Export the matrix to EPS using the selected permutation and partition (<i>File> Network> Export as Matrix to EPS> Using Permutation + Partition</i>). Lines dividing clusters defined by a partition are also drawn.
<i>F5</i>	Create a new vector holding the values of the active partition (<i>Partition> Copy to Vector</i>).
<i>F6</i>	Create a new partition holding the truncating and absolute values of active vector (<i>Vector> Make Partition> Copy to Partition by Truncating (Abs)</i>)
<i>F7</i>	Basic information about the active network (<i>Network> Info> General</i>).
<i>F8</i>	Basic information about the active partition (<i>Partition> Info</i>).
<i>F9</i>	Basic information about the active vector (<i>Vector> Info</i>).
<i>F10</i>	Repeat the last executed command in Pajek several times with the same or different data objects (<i>Macro> Repeat Last Command</i>).
<i>F11</i>	Information about available computer memory (<i>Info> Memory</i>).
<i>F12</i>	Run one of the already executed commands (with new parameters): First select one of the executed commands from the list, change its parameters, and finally execute the command with updated parameters (<i>Macro> Run Command</i>).

A4.2 Hierarchy Edit Screen

Shortcut	Description and Command
<i>Ctrl-t</i>	Change the type of a cluster from unlabeled to Close to Cut to Border in the hierarchy (<i>Edit> Change Type</i>).
<i>Ctrl-n</i>	Change the label of a cluster in the hierarchy (<i>Edit> Change Name</i>).
<i>Ctrl-s</i>	Toggle the option to show the members of the cluster or all members of the cluster and its subclusters when right-clicking a cluster in the hierarchy (<i>Edit> Show Subtree</i>).
<i>Ctrl-u</i>	Push selected node of hierarchy one level higher (<i>Edit> Push Up</i>).
<i>Ctrl-d</i>	Push selected node of hierarchy one level lower (<i>Edit> Push Down</i>).

A4.3 Draw Screen

Shortcut	Description and Command
<i>Ctrl-a</i>	Draw the network with all vertices in class 0 of a new empty partition (<i>[Main] Draw> Network + Create Null Partition</i>).
<i>Ctrl-g</i>	Draw the network (<i>[Main] Draw> Network</i>).

(continued)

(continued)

Shortcut	Description and Command
<i>Ctrl-p</i>	Draw the network and the active partition (<i>[Main] Draw> Network + First Partition</i>).
<i>Ctrl-q</i>	Draw the network, the active partition, and the active vector (<i>[Main] Draw> Network + First Partition + First Vector</i>).
<i>Ctrl-u</i>	Draw the network and the active vector (<i>[Main] Draw> Network + First Vector</i>).
<i>Ctrl-b</i>	Show the labels of the lines (<i>Options> Lines> Mark Lines> with Labels</i>).
<i>Ctrl-o</i>	Do not show (omit) the values or labels of the lines (<i>Options> Lines> Mark Lines> No</i>).
<i>Ctrl-v</i>	Show the values of the lines (<i>Options> Lines> Mark Lines> with Values</i>).
<i>Ctrl-l</i>	Show the labels of the vertices (<i>Options> Mark Vertices Using> Labels</i>).
<i>Ctrl-n</i>	Show the numbers of the vertices (<i>Options> Mark Vertices Using> Numbers</i>).
<i>Ctrl-d</i>	Do not show labels or numbers of the vertices (<i>Options> Mark Vertices Using> No</i>).
<i>Ctrl-k</i>	Draw layout using the Kamada–Kawai algorithm and tile weakly connected components (<i>Layout> Energy> Kamada-Kawai> Separate Components</i>).
<i>Ctrl-i</i>	Optimize layout of each cluster using the Kamada–Kawai algorithm (<i>Layout> Energy> Kamada-Kawai> Optimize inside Clusters only</i>).
<i>Ctrl-t</i>	Tile components in plane (<i>Layout> Tile Components</i>).
<i>S</i>	Spin around the normal in one direction.
<i>s</i>	Spin around the normal in the opposite direction.
<i>X</i>	Spin around the <i>x</i> -axis, the bottom approaching the viewer.
<i>x</i>	Spin around the <i>x</i> -axis, the top approaching the viewer.
<i>Y</i>	Spin around the <i>y</i> -axis, the left approaching the viewer.
<i>y</i>	Spin around the <i>y</i> -axis, the right approaching the viewer.
<i>Z</i>	Spin around the <i>z</i> -axis clockwise.
<i>z</i>	Spin around the <i>z</i> -axis counterclockwise.
<i>F3</i>	Append the active network to the end of the selected Pajek Project File (<i>Export> Append to Pajek Project File</i>).
<i>Shift-F2</i>	Draw layout using Fruchterman–Reingold algorithm in 2D (<i>Layout> Energy> Fruchterman–Reingold> 2D</i>).
<i>Shift-F3</i>	Draw layout using Fruchterman–Reingold algorithm in 3D (<i>Layout> Energy> Fruchterman–Reingold> 3D</i>).
<i>Ctrl-F2</i>	Draw layout using Pivot MDS algorithm in 2D (<i>Layout> Pivot MDS> Random Pivots> 2D</i>).
<i>Ctrl-F3</i>	Draw layout using Pivot MDS algorithm in 3D (<i>Layout> Pivot MDS> Random Pivots> 3D</i>).

Glossary

The book's important structural concepts are listed and explained in alphabetical order. Numbers refer to the sections presenting the concepts.

Concept	Description	Section
Actor	Actor refers to a person, organization, or nation that is involved in a social relation. Hence, an actor is a vertex in a social network.	1.3
Acyclic network	An acyclic network does not contain cycles.	10.4
Adjacency matrix	An adjacency matrix is a square matrix with one row and one column for each vertex in a network. The content of a cell in the matrix indicates the presence and possibly the sign or value of a tie from the vertex represented by the row to the vertex represented by the column.	12.2
Adjacent	Two vertices are adjacent if they are connected by a line.	3.3
Adoption category	Adoption categories classify people according to their adoption time relative to all other adopters.	8.3
Adoption rate	The adoption rate is the number or percentage of new adopters at a particular moment.	8.2
Affiliation matrix	An affiliation matrix is a rectangular matrix with one row for each vertex from one mode (subset) and one column for each vertex from the other mode (subset) in a two-mode network. The content of a cell in the matrix indicates the presence and possibly the sign or value of a tie from the vertex represented by the row to the vertex represented by the column.	12.2
Aggregate constraint	The aggregate constraint on a vertex is the sum of the dyadic constraint on all of its ties.	7.4
Arc	An arc is a directed line. Formally, an arc is an ordered pair of vertices.	1.3.1
Articulation point	<i>See</i> Cut-vertex	7.3

(continued)

Concept	Description	Section
Assortativity	Assortativity is a preference of vertices to attach to other vertices that are similar to them according to a numeric property (<i>see</i> Homophily).	6.6
Assortativity coefficient	The assortativity coefficient is the correlation between one or two numeric properties of vertices that are directly connected.	6.6
Asymmetric dyad	An asymmetric dyad is a pair of vertices connected by unilateral arc(s).	10.3
Attribute	An attribute is a characteristic of a vertex measured independently of the network.	2.3
Balance model	The balance model applies to an unsigned directed network if it consists of two cliques that are not interrelated.	10.3
Balanced (semi-)cycle	In a signed graph, a (semi-)cycle is balanced if it does not contain an uneven number of negative arcs.	4.2
Balanced signed graph	A signed graph is balanced if all of its (semi-)cycles are balanced. A signed graph is balanced if it can be partitioned into two clusters such that all positive ties are contained within the clusters and all negative ties are situated between the clusters.	4.2
Betweenness centrality	The betweenness centrality of a vertex is the proportion of all geodesics between pairs of other vertices that include this vertex.	6.4
Betweenness centralization	Betweenness centralization is the variation in the betweenness centrality of vertices divided by the maximum variation in betweenness centrality scores possible in a network of the same size.	6.4
Bi-component	A bi-component is a component of a minimum size of 3 that does not contain a cut-vertex.	7.3
Bipartite network	<i>See</i> Two-mode network	5.3
Block	A block contains the cells of an adjacency matrix that belong to the cross section of one or two classes.	12.4
Blockmodel	A blockmodel assigns the vertices of a network to classes, and it specifies the permitted type(s) of relation within and between classes.	12.4.1
Blockmodeling	The technique to obtain a blockmodel is called blockmodeling.	12.4.2
Blood marriage	A blood marriage is the marriage of people with a close common ancestor.	11.4
Bridge	A bridge is a line whose removal increases the number of components in the network.	7.3
Brokerage role	A brokerage role of a vertex is a particular pattern of ties and group affiliations.	7.5
Cell	A cell of a matrix is the intersection of a row and a column.	12.2
Clique	A clique is a maximal complete subnetwork containing three vertices or more.	3.6
Closeness centrality	The closeness centrality of a vertex is the number of other vertices divided by the sum of all distances between the vertex and all others.	6.3

Concept	Description	Section
Closeness centralization	Closeness centralization is the variation in the closeness centrality of vertices divided by the maximum variation in closeness centrality scores possible in a network of the same size.	6.3
Clusterability model	The clusterability model applies to an unsigned directed network if it consists of two or more cliques that are not interrelated.	10.3
Clusterable (semi-)cycle	A cycle or semicycle is clusterable if it does not contain exactly one negative arc.	4.2
Clusterable signed graph	A signed graph is clusterable if it can be partitioned into clusters such that all positive ties are contained within clusters and all negative ties are situated between clusters.	4.2
Clustering coefficient	The clustering coefficient or transitivity of a network is the proportion of all two-paths in the network that are closed.	13.3.1
Complete network	A complete network is a network with maximum density: All possible lines occur.	3.3
Community	Community is a group (or cluster) of vertices for which density (and values) of lines inside clusters is larger than density (and values) of lines between clusters.	5.5
Community detection	Community detection is an approach for identifying dense clusters (communities) in networks.	5.5
Component	A (weak) component is a maximal (weakly) connected subnetwork.	3.4
Contextual view	In a contextual view of a network, vertices in all but one class are shrunk.	2.4.3
Coordinator role	A vertex is a coordinator if it is situated on a path between two vertices within its own class (group) that are not directly connected.	7.5
Critical mass	The critical mass of a diffusion process is the minimum number of adopters needed to sustain a diffusion process.	8.4
Cut-vertex	A cut-vertex is a vertex whose deletion increases the number of components in the network.	7.3
Cycle	A cycle is a closed path.	4.2
Degree	The degree of a vertex is the number of lines incident with it.	3.3
Degree centrality	The degree centrality of a vertex is its degree.	6.3
Degree centralization	Degree centralization of a network is the variation in the degrees of vertices divided by the maximum degree variation that is possible in a network of the same size.	6.3
Degree sequence	The degree sequence of a network is a list of the degree of all vertices sorted by the vertices' sequential numbers.	13.3.1
Delete a vertex	Deleting a vertex from a network means that the vertex and all lines incident with this vertex are removed from the network.	7.3

(continued)

Concept	Description	Section
Dendrogram	A dendrogram is a chart visualizing the results of hierarchical clustering.	12.3
Density	Density is the number of lines in a simple network, expressed as a proportion of the maximum possible number of lines.	3.3
Diffusion curve	A diffusion curve displays the prevalence of an innovation in the course of time.	8.2
Digraph	A digraph or directed graph is a graph containing one or more arcs.	1.3.1
Dimension of a partition	The number of entries (vertices) of a partition.	4.4
Distance	The distance from vertex u to vertex v is the length of the geodesic from u to v .	6.3
Domain	In a directed network, the (input, output) domain of a vertex is the number or percentage of all other vertices that are connected by a path to this vertex.	9.5
Dyad	A dyad is a pair of vertices and the lines between them.	10.3
Dyadic constraint	The dyadic constraint on vertex u exercised by a tie between vertices u and v is the extent to which u has more and stronger ties with neighbors that are strongly connected with vertex v .	7.4
E-I index	The External-Internal Index measures how well clusters divide a network into cohesive groups. It compares number (and values) of lines between groups to number (and values) of lines inside groups.	5.5
Edge	An edge is an undirected line. Formally, an edge is an unordered pair of vertices.	1.3.1
Ego-centered approach	An ego-centered approach to a network considers the structural characteristics of individual vertices.	6.1
Egocentric density	Egocentric density is the density of the ego-network without ego.	7.4
Ego-network	The ego-network of a vertex contains this vertex, its neighbors, and all lines among the selected vertices.	7.4
Eigenvector centrality	The eigenvector centrality of a vertex is the extent to which a vertex is linked to vertices with high eigenvector centrality.	6.5
Eigenvector centralization	<i>Eigenvector centralization</i> is the variation in the eigenvector centrality of vertices divided by the maximum variation in eigenvector centrality scores possible in a network of the same size.	6.5
Endogamy	Endogamy or intermarriage means that families are linked by several kinship ties.	11.4
Equivalent (class, position)	Actors with similar patterns of ties are said to be relationally equivalent, to constitute an equivalence class, or to occupy equivalent positions in the network.	12.3
Event	An event is a happening, context, or organization where actors may gather.	5.3

Concept	Description	Section
Exposure	The exposure of a vertex in a network at a particular moment is the proportion of its neighbors that have adopted before that moment.	8.3
Extract a subnetwork	To extract a subnetwork from a network, select a subset of its vertices and all lines that are incident only with the selected vertices.	2.4.1
Family of child or orientation (FAMC)	A person's family of child or orientation (FAMC) is the family in which this person is a child.	11.3
Family of spouse or procreation (FAMS)	A person's family of spouse or procreation (FAMS) is the family in which this person is a parent.	11.3
FishEye	FishEye is a general technique in data visualization that magnifies nearby objects (e.g. objects close to mouse pointer) while shrinking distant objects.	2.5
Forest	A forest is a graph consisting of two or more distinct (unconnected) trees.	11.4
Gatekeeper	A vertex is a gatekeeper if it is situated on a path from a vertex of another class (group) toward a vertex of its own class, provided that these vertices are not directly connected.	7.5
Genealogical generation	A genealogical generation is a set of people connected to a (close) common ancestor at the same remove.	11.3
Generalized blockmodeling	In generalized blockmodeling, the permitted block types are specified for each individual block.	12.4.3
Generation jump	In a genealogy, a generation jump occurs when people marry who are connected to a common ancestor at different removes.	11.3
Geodesic	A geodesic is the shortest path between two vertices.	6.3
Global main path	In an acyclic network, a global main path is a main path with the overall highest sum of traversal weights.	11.6
Global view	A global view is a sociogram of a network in which all classes are shrunk.	2.4.2
Graph	A graph is a set of vertices and a set of lines between pairs of vertices.	1.3.1
Hierarchical clustering	Hierarchical clustering is a statistical technique for subdividing units into increasingly more homogeneous subsets.	12.3
Hierarchical clusters model	The hierarchical clusters model applies to an unsigned directed network if it consists of connected clusters such that clusters within ranks are not related and clusters between ranks are related by null dyads or asymmetric dyads pointing toward the higher rank with the additional provision that a cluster contains no cycles of asymmetric dyads.	10.3
Hierarchy	A hierarchy is a data object for classifying vertices if a vertex may belong to several classes. It is especially suited for a hierarchical clustering of vertices in which units are subdivided into more and more homogeneous subsets.	3.6

(continued)

Concept	Description	Section
Homophily	Homophily is a preference of vertices to attach to other vertices which are similar to them according to a categorical property (see: Assortativity).	6.6
Image matrix	An image matrix is a simplification of an adjacency matrix by shrinking each block to one new cell indicating the block type.	12.4.1
Immediacy index	The immediacy index is the average number of citations of the articles in a journal during the year of its publication.	11.6
Impact factor	The impact factor of a journal is the average number of citations to articles in this journal.	11.6
Incident	A line is defined by its two endpoints, which are the two vertices that are incident with the line.	1.3.1
Indegree	The indegree of a vertex is the number of arcs it receives.	3.3
Induced subnetwork	A subset of vertices from a network and all lines that are incident only with these vertices is called an induced subnetwork.	2.4.1
Island	An island is a maximal subnetwork of vertices connected directly or indirectly by lines with a value greater than the lines to vertices outside the subnetwork.	5.4
Isomorphic	Two networks are isomorphic, that is, they have the same structure, if a permutation of the vertices in one network produces the other network.	12.2
Itinerant broker	A vertex is an itinerant broker if it is situated on a path between two vertices from the same class to which the brokering vertex does not belong, provided that the two vertices are not connected directly.	7.5
k -Connected component	A k -connected component is a maximal subnetwork in which each pair of vertices is connected by at least k distinct (noncrossing) semipaths or paths.	3.4
k -Core	A k -core is a maximal subnetwork in which each vertex has at least degree k within the subnetwork.	3.5
Key-Routes	Key-Routes are selected arcs (citations) in an acyclic network (usually arcs with high traversal weights) that start the search for main paths.	11.6
Liaison	A vertex is a liaison if it is situated on a path between two vertices that are not connected directly and all three vertices are in different classes (groups).	7.5
Line	A line is a tie between two vertices in a network: either an arc or an edge.	1.3.1
Local main path	In an acyclic network, a local main path is a main path where we repeatedly choose the arc with the highest traversal weight emanating from the current arc. The result is not necessarily the main path with the overall highest sum of traversal weights.	11.6
Local view	A sociogram of an induced subnetwork offers a local view.	2.4.1
Loop	A loop is a line that connects a vertex to itself.	1.3.1

Concept	Description	Section
Main path	In an acyclic network, a main path is a path from a source vertex to a sink vertex with the highest traversal weights on its arcs.	11.6
Main path component	In an acyclic network, a main path component is a component in the network after removal of all arcs with traversal weights below a particular value (usually the lowest traversal weight on the network's main paths).	11.6
Matrix	A matrix is a two-way table containing rows and columns.	12.2
Modularity	Modularity measures how well a selected partition divides a network into communities.	5.5
Multiple lines	If a particular arc or edge, that is, a particular ordered or unordered pair of vertices, occurs more than once, there are multiple lines.	1.3.1
Multiple relations network	A network containing different relations on the same set of vertices.	1.3.1
Multiplex network	See Multiple relations network	1.3.1
Multiplicity	Line multiplicity is the number of times a specific line (ordered or unordered pair of vertices) occurs in a network.	5.3
Neighbor	A vertex that is adjacent to another vertex is its neighbor.	3.3
Nested	Subnetworks are said to be nested if one subnetwork is a subset of the other.	3.5
Network	A network consists of a graph and additional information on the vertices or the lines of the graph.	1.3.1
Network motif	Network motifs are small network configurations that occur with significantly high frequency.	13.4
Nonblood relinking	Nonblood relinking refers to multiple marriages between families in which no couple has a close common ancestor.	11.4
Null dyad	A null dyad is a pair of vertices that are not connected by lines.	10.3
One-mode network	In a one-mode network, each vertex can be related to every other vertex.	5.3
Optimization technique	An analytic technique searching for the best solution according to a criterion function by repetition is called an optimization technique.	4.4
Ore graph	The Ore graph is a sociogram of kinship ties in which men are represented by triangles, women by ellipses, marriages by (double) lines, and parent-child ties by arcs pointing from parent to child.	11.3
Outdegree	The outdegree of a vertex is the number of arcs it sends.	3.3
Partial order	In a partial order, some but not all pairs of units (e.g., vertices) are ordered.	10.5
Partition	A partition of a network is a classification or clustering of the vertices in the network such that each vertex is assigned to exactly one class or cluster.	2.3

(continued)

Concept	Description	Section
Path	A path is a walk in which no vertex in between the first and last vertex of the walk occurs more than once.	3.4
Pedigree	The pedigree of a person is the set of his or her ancestors.	11.3
Permutation	A permutation of a network is a renumbering of its vertices.	12.2
P-graph	The P-graph, or parentage graph, is a genealogical network in which couples and unmarried individuals are the vertices and arcs, which represent individuals, point from children to parents.	11.4
Popularity	The popularity or indegree of a vertex is the number of arcs it receives in a directed network.	9.3
Position	A position is a particular pattern of ties.	12.3
Preprint transformation	The preprint transformation is an approach for transforming an almost acyclic network (acyclic network containing few short cycles) to an acyclic network.	11.6
Prevalence	The prevalence of an innovation is the cumulative percentage of adopters at a particular time.	8.2
Proportional strength	The proportional strength of a tie with respect to all ties of a person is the value of the line(s) representing a tie, divided by the sum of the values of all lines incident with a person.	7.4
Proximity prestige	The proximity prestige of a vertex is the proportion of all vertices (except itself) in its input domain divided by the mean distance from all vertices in its input domain.	9.6
Ranked clusters model	The ranked clusters model applies to an unsigned directed network if it consists of cliques and ranks such that cliques within ranks are not related and cliques between ranks are related by asymmetric dyads pointing toward the higher rank.	10.3
Rate of participation	In a two-mode network, the degree of a vertex is called the rate of participation of an actor if the vertex refers to an actor.	5.3
Reachable	We say that a vertex is reachable from another vertex if there is a path from the latter to the former.	6.3
Regular block	A regular block is a block containing at least one arc in each row and in each column.	12.4.3
Regular equivalence	Vertices that are regular equivalent do not have to be connected to the same vertices, but they have to be connected to vertices in the same classes.	12.4.3
Relinking index	The relinking index measures the amount of relinking in a P-graph.	11.4
Representative	A vertex is a representative if it is situated on a path from a vertex of its own class (group) toward a vertex of another class, provided that these vertices are not directly connected.	7.5

Concept	Description	Section
Restricted domain	The restricted (input, output) domain of a vertex in a directed network is the number or percentage of all other vertices that are connected by a path of a selected maximum length to or from this vertex.	9.5
Secondary structural hole	In the ego-network of a vertex, a secondary structural hole exists if there is a vertex outside the ego-network that is at least as central as a vertex in the ego-network but not directly linked to this vertex. In this case, ego can replace the tie with the neighbor with a tie to the vertex outside the ego-network.	7.5
Semicycle	A semicycle is a closed semipath.	4.2
Semipath	A semipath is a semiwalk in which no vertex in between the first and last vertex of the semiwalk occurs more than once.	3.4
Semiwalk	A semiwalk from vertex u to vertex v is a sequence of lines such that the end vertex of one line is the starting vertex of the next line, and the sequence starts at vertex u and ends at vertex v .	3.4
Shrink a network	To shrink a network, replace a subset (class) of its vertices with one new vertex that is incident to all lines that were incident with the vertices of the subset in the original network.	2.4.2
Signed graph	A signed graph is a graph in which each line carries either a positive or a negative sign.	4.2
Simple graph	A simple undirected graph contains neither multiple edges nor loops. A simple directed graph does not contain multiple arcs.	1.3.1
Sink vertex	In an acyclic network, a sink vertex is a vertex with a zero outdegree.	11.6
Size of an event	In a two-mode network, the degree of a vertex is known as the size of an event if the vertex refers to an event.	5.3
Social capital	The number and intensity of a person's social ties is called his or her social capital or sociability.	6.3
Social contagion	Social contagion is the diffusion of behavior or information via social ties.	8.2
Social generation	A social generation is the set of people who are born in the same period: a birth cohort.	11.3
Sociocentered approach	A sociocentered approach to a network considers the structural characteristics of the entire network.	6.1
Source vertex	In an acyclic network, a source vertex is a vertex with zero indegree.	11.6
Star-network	A star-network is a network in which one vertex is connected to all other vertices but these vertices are not connected among themselves.	6.3
Statistical network model	A statistical network model is a mathematical description of a collection of possible networks and a probability distribution on this set.	13.1
Strong component	A strong component is a maximal strongly connected subnetwork.	3.4

(continued)

Concept	Description	Section
Strongly connected	A network is strongly connected if each pair of vertices is connected by a path.	3.4
Structural equivalence	Two vertices are structural equivalent if they have identical ties with themselves, each other, and all other vertices.	12.3
Structural hole	There is a structural hole in the ego-network of a vertex if two of its neighbors are not directly connected.	7.4
Structural property	A structural property is a characteristic (value) of a vertex that is a result of network analysis.	2.3
Structural relinking	Structural relinking occurs when families intermarry more than once in the course of time.	11.4
Symmetric-acyclic model	The symmetric-acyclic model applies to a directed network if it consists of clusters of vertices that are linked by symmetric ties directly or indirectly and if the ties among the clusters produce an acyclic network (when the clusters are shrunk).	10.5
Symmetrize	To symmetrize a directed network is to replace unilateral and bidirectional arcs with edges.	3.3
Threshold	The threshold of a vertex is its exposure at the time of adoption. It is equal to the proportion of its neighbors that have adopted earlier than this vertex.	8.3
Threshold category	A threshold category is a set of vertices with similar thresholds.	8.3
Threshold lag	A threshold lag is a period in which an actor does not adopt although he or she is exposed at the level at which he or she will adopt later.	8.4
Transitive triad	In a transitive triad, each path of length 2 is closed by an arc from the starting vertex to the end vertex of the path.	10.3
Transitivity model	The transitivity model applies to an unsigned directed network if it consists of cliques such that cliques within ranks are not related and cliques between ranks are related by null dyads or asymmetric dyads pointing toward the higher rank.	10.3
Transitivity of a network	<i>See</i> Clustering coefficient.	9.3
Transposed network	A transposed network is a network in which the direction of all arcs is reversed.	
Traversal weight	The traversal weight of an arc or vertex is the proportion of all paths between source and sink vertices that contain this arc or vertex.	11.6
Tree	A tree is a connected graph that contains no semicycles.	11.4
Triad	A triad is a (sub-)network consisting of three vertices.	3.6
Triad census	The triad census of a directed network is the frequency distribution of the sixteen types of triads in this network.	10.3
Two-mode network	In a two-mode network, vertices are divided into two sets and vertices can be related only to vertices in the other set.	5.3

Concept	Description	Section
Undirected graph	An undirected graph contains no arcs: All of its lines are edges.	1.3.1
Valued network	A valued network is a network in which lines have (variable) values.	5.3
Vector	A vector is a data object assigning a numerical value to each vertex in a network.	2.5
Vertex (vertices)	A vertex (singular of vertices) is the smallest unit in a network.	1.3.1
Walk	A walk is a semiwalk with the additional condition that none of its lines is an arc of which the end vertex is the arc's tail. One might say that you always follow the direction of arcs in a walk.	3.4
Weakly connected	A network is weakly connected if each pair of vertices is connected by a semipath.	3.4

Index of Pajek and R Commands

Draw screen

- [Draw] Change the class number of
 - vertices, 44, 390–1
- [Draw] Draw screen, 18–20
- [Draw] Export, 25, 404–11
- [Draw] Export > 2D >
 - > Bitmap, 26, 404–5
 - > EPS/PS, 26–7, 405
 - > JPEG, 26, 404–5
 - > SVG, 26–7, 406–8
 - > Current and all Subsequent, 407
 - > General, 406
 - > Labels/Arcs/Edges, 406
 - > Line Values
 - > Classes, 407
 - > Nested Classes, 131–2, 407
 - > Options, 407
 - > Partition, 406–7
 - > Classes, 406–7
 - > Classes with semi-lines, 406–7
 - > Nested Classes, 406–7
 - > VOSviewer, 408–9
- [Draw] Export > 3D
 - > Kinemages
 - > Current and all Subsequent, 410–11
 - > MDL MOLfile, 410
 - > VRML, 409
 - > X3D, 138–9
- [Draw] Export > Append to Pajek Project File
 - > Append, 407–8
 - > Select File, 407–8
- [Draw] Export > Options
 - > Arc Color, 417
 - > Arc Width, 417
 - > Arrow Position, 417
 - > Arrow Size, 417
 - > Bckg. Color 1, 419–20
 - > Bckg. Color 2, 419–20
 - > Bckg. Color 3, 419–20
 - > Bezier Curves
 - > Angle1, 418
 - > Angle2, 418
 - > Straight Lines, 418
 - > Velocity1, 418
 - > Velocity2, 418
 - > Border Color, 412–14, 420
 - > Border Radius, 420
 - > Border Width, 412–14, 420
 - > Bottom frame on the right, 420
 - > Edge Color, 417
 - > Edge Width, 417
 - > EPS, SVG: Lines finished at Vertex Border, 418–19
 - > EPS, SVG, X3D, VRML Size of Vertices, 418–19
 - > EPS Border, 420
 - > EPS: Use RGB colors instead of CMYK, 418–19
 - > Export options overwrite Shapes file, 415
 - > Font Size, 417
 - > Interior Color, 412–14
 - > Label Color, 417
 - > Label Position (on line), 417–18
 - > Label Position (polar)
 - > Angle, 418
 - > Radius, 418
 - > Middle frame on the right, 419–20
 - > Pattern, 417

- [Draw] Export > Options (*cont.*)
 - > Shape, 412–14
 - > Shape Angle, 412–14
 - > Shapes file, 415
 - > SVG
 - > Opacity of Vertices, Lines, 418–19
 - > Tooltips: Vertices, Lines, Clusters, 418–19
 - > Vertices 3D Effect Linear, Radial, 418–19
 - > SVG Gradient, 419–20
 - > Symbol Size, 415–16
 - > Top frame on the left, 412–16
 - > Top frame on the right, 418–19
 - > X3D, VRML Width of Lines, 418–19
 - > x/y Ratio, 412–14
- [Draw] FishEye
 - > Cartesian, 58–9, 302
 - > Exit, 58–9
 - > Factor, 58–9
 - > Polar, 58–9, 302
- [Draw] GraphOnly, 23
- [Draw] Info
 - > All Properties, 24–5
 - > Closest Vertices, 24–5
 - > Correlation (Layout, Geodesics)*, 25
- [Draw] Layers
 - > Averaging x Coordinate, 273–4, 302
 - > In y Direction, 198, 273–4
 - > In z Direction, 137
 - > Optimize Layers in x Direction, 198–9, 273–4
 - > Type of Layout, 137
 - > 3D, 137
- [Draw] Layout > Circular > using Partition, 45
- [Draw] Layout > Energy, 20–1
 - > Fruchterman–Reingold, 21–2
 - > 2D, 21–2
 - > 3D, 21–2, 138–9, 409
 - > Factor, 21
 - > Kamada–Kawai
 - > Fix First and Last Vertex, 21
 - > Fix One Vertex in the Middle, 21
 - > Fix Selected Vertices, 45
 - > Free, 21, 42
 - > Separate Components, 21, 131
 - > Starting Positions, 20–1
 - > Circular, 20–1
 - > Given xy, 20–1
 - > Given z, 20–1
 - > Random, 20–1
- [Draw] Layout > Pivot MDS > Random Pivots > 2D, 3D, 22, 409
- [Draw] Layout > VOS Mapping > 2D, 3D, 22, 409
- [Draw] Move
 - > Circles, 23, 42
 - > Fix, 23
 - > Radius, 23
 - > y, 199
 - > Grid, 23
- [Draw] Next, 23
- [Draw] Options, 23–4
 - > Colors
 - > Arcs > Relation Number, 19–20
 - > Edges > Relation Number, 19–20
 - > Partition Colors > for Vertices, 42
 - > Relation Colors, 19–20
 - > Use Third Partition for Symbol Color, 44
 - > Layout, 23–4
 - > Real xy Proportions, 60
 - > Lines > Mark Lines
 - > No, 20
 - > with Labels, 20
 - > with Values, 20, 51
 - > Mark Vertices Using, 23–4
 - > Clusters of Second Partition, 43–4
 - > Cluster Symbols of Second Partition, 44, 415–16
 - > Labels Centered, 23–4
 - > Labels as Tooltips, 23–4
 - > Mark Cluster Only, 284
 - > Vector Values, 58
 - > Previous/Next
 - > Apply to, 23, 111–12, 407, 410–11
 - > Optimize Layouts, 111–12
 - > Scrollbar On/Off, 137
 - > Size
 - > of Symbols, 44
 - > of Vertices, 57
 - > Size > of Vertices, 57–8, 183
 - > Symbols for Partition Clusters
 - > Change, 44
 - > Transform, 45
 - > Fit Area, 60
 - > Fit Area > max(x), max(y), max(z), 45
 - > Rotate 2D, 199, 260
 - > Values of Lines > Similarities, 106, 182–3
- [Draw] Previous, 23
- [Draw] Redraw, 23

- [Draw] Spin, 137
- [Draw] ZoomOut, 23
- Edit Hierarchy screen**
- [Editing Hierarchy] Editing Hierarchy Screen, 329
 - > Change Type, 330
 - > Show Subtree, 329–30
- Edit Network screen**
- [Editing Network] Editing Network Screen, 28–9
- [Editing Network] Newline, 28–9
- Edit Partition screen**
- [Editing Partition] Editing Partition Screen, 39–41
- Main screen**
- [Main] Cluster
 - > Create Complete Cluster, 326
 - > Create Empty Cluster, 210, 284
- [Main] Draw > Network, 11
- [Main] Draw > Network + Create Null Partition, 44
- [Main] Draw > Network + First Partition, 42
- [Main] Draw > Network + First Partition + First Vector, 57, 131
- [Main] Draw > Network + First Vector + Second Vector, 57–8
- [Main] File > Hierarchy > View/Edit, 89–90, 177, 284–5
- [Main] File > Ini File
 - > Load, 412
 - > Save, 412
- [Main] File > Network
 - > Dispose, 376–7
 - > Export as Matrix to EPS
 - > Options, 320–1
 - > Original, 320
 - > Using Permutation + Partition, 320–1, 330
 - > Read, 11–12, 387–9
 - > Save, 25, 30, 77–8, 391
 - > View/Edit, 28–9, 390
- [Main] File > Pajek Project File, 39–41
- [Main] File > Partition
 - > Dispose, 376–7
 - > Read, 39
 - > Save, 39, 391
 - > View/Edit, 39–41, 228–9, 284, 390–1
- [Main] File > Vector, 55
 - > Dispose, 376–7
 - > Read, 11–12, 59–60
 - > Save, 391
 - > View/Edit, 157, 237, 391, 398
- [Main] Hierarchy drop-down menu, 89–90
- [Main] Hierarchy
 - > Extract Cluster, 285
 - > Make Partition, 330
- [Main] Info > Memory, 376–7
- [Main] Info > Show Report Window, 16–17
- [Main] Macro
 - > Add Message, 208
 - > Play, 208, 278, 288
 - > Record, 208
 - > Repeat Last Command, 375, 376
 - > Fix (First) Partition, 376
 - > Fix (Second) Partition, 376
- [Main] Main Screen, 11–12
- [Main] Network drop-down menu, 11, 319
- [Main] Network > 2-Mode Network
 - > 2-Mode to 1-Mode, 125–6
 - > Columns, 125–6
 - > Include Loops, 126
 - > Multiple Lines, 126
 - > Rows, 125–6
 - > Partition into 2 Modes, 121
- [Main] Network > Acyclic Network
 - > Create (Sub)Network > Main Paths
 - > Global Search > Key-Route, 295
 - > Global Search > Standard, 295, 298–300
 - > Global Search > Through Vertices in Cluster, 298–300
 - > Local Search > Backward, 300
 - > Local Search > Forward, 298
 - > Local Search > Key-Route, 300
 - > Local Search > Through Vertices in Cluster, 298
 - > Create Partition > Depth Partition
 - > Acyclic, 259
 - > Genealogical, 273–4
 - > Create Weighted Network + Vector
 - > Traversal Weights, 296, 298
 - > Normalization of Weights, 297
 - > Info, 284, 285
 - > Transform > Preprint Transformation, 303
- [Main] Network > Create Hierarchy
 - > Clustering*, 329
 - > Symmetric-Acyclic, 259

- [Main] Network > Create New Network
 - > Empty Network, 285, 389
 - > SubNetwork with Paths
 - > All Shortest Paths between Two Vertices, 156–7, 275
 - > Info on Diameter, 361
 - > Transform
 - > 1-Mode to 2-Mode, 278
 - > Arcs → Edges > All, 15, 77–8, 88, 207, 215–16, 275
 - > Arcs → Edges > Bidirected only, 260
 - > Edges → Arcs, 14, 155
 - > Line Values, 274–5
 - > Remove
 - > all Edges, 275, 277
 - > Lines with Value > lower than, 50, 52, 301–2
 - > Lines with Value > within interval, 112
 - > Loops, 155, 259, 302–3
 - > Multiple Lines, 155, 275, 372
 - > Transpose > 1-Mode, 228–9
 - > with Bi-Connected Components stored as Relation Numbers, 176–7, 284–5
 - [Main] Network > Create Partition
 - > Blockmodeling*
 - > Optimize Partition, 335
 - > Random Start, 337–8, 341–2
 - > Restricted Options, 335
 - > Short Report, 335
 - > Communities
 - > Louvain Method, 133
 - > VOS Clustering, 133
 - > Components
 - > Strong, 83, 86, 254, 260, 302–3
 - > Weak, 83, 301–2, 360
 - > Degree, 155
 - > All, 78, 361
 - > Input, 78, 127, 228
 - > Output, 78, 277
 - > Islands
 - > Generate Network with Islands, 131
 - > Line Weights, 129–31
 - > *k*-Core
 - > All, 86, 296
 - > Input, 86
 - > Output, 86
 - > *k*-Neighbours, 155–6, 276
 - > All, 183–4
 - > Input, 233–4
 - > Vertex Labels Matching Regular Expression, 53
 - [Main] Network > Create Random Network
 - > Bernoulli/Poisson, 362
 - > Extended Model, 372
 - > Scale Free, 371
 - > Adding > Free, 372
 - > Directed, 371
 - > Small World, 365–6
 - > Total No. of Arcs, 362
 - > Vertices Output Degree, 203, 362
 - [Main] Network > Create Vector
 - > Centrality
 - > Betweenness, 159, 215–16
 - > Closeness, 157–8
 - > Degree, 155
 - > All, 370
 - > Input, 210
 - > Hubs-Authorities, 161
 - > Proximity Prestige > Input, 234, 237
 - > Clustering Coefficients > CC1, 184, 361, 365, 376
 - > Distribution of Distances*, 157, 365
 - > Get Coordinate, 60, 137–8
 - > Structural Holes, 182
 - [Main] Network > Info
 - > Degree Assortativity, 163
 - > General, 16, 41–2, 77, 78, 183–4, 187, 260–1, 300
 - > Line → Rank of its Value, 300
 - > Line Values, 126, 297–8
 - > Triadic Census, 252
 - [Main] Network > Multiple Relations Network
 - > Change Relation Number – Label, 278
 - > Extract Relation(s) into Separate Network(s), 12–14, 278
 - > Info, 17
 - [Main] Network > Multiple Relations Network > Extract Relation(s) into Separate Network(s), 12–14
 - [Main] Network > Signed Network > Create Partition > Doreian–Mrvar Method*, 106–9
 - [Main] Network > Temporal Network > Generate in Time, 111, 407–8
 - [Main] Networks drop-down menu, 12

- [Main] Networks > Cross-Intersection > First, 14
- [Main] Networks > Fragment (First in Second), 88–9, 90
 - > Check relation numbers, 286
 - > Check values of lines, 286
 - > Find, 285
 - > Induced, 285
- [Main] Networks > Multiply Networks, 278
- [Main] Networks > Union of Vertices, 60
- [Main] Operations drop-down menu, Ch2
- [Main] Operations > Network + Cluster
 - > Dissimilarity* > Network-based
 - > d1 > All, 326–7
 - > Options > Report Matrix, 326–7
 - > Extract SubNetwork, 285
- [Main] Operations > Network + Partition
 - > Brokerage Roles, 188–9
 - > Extract > SubNetworks Induced by Each Selected Cluster Separately, 135
 - > Extract > SubNetwork Induced by Union of Selected Clusters, 47, 86, 131, 134, 183–4, 287–8, 302, 325–6, 335
 - > Info > E-I Index, 134
 - > Shrink Network, 50, 52, 255, 302–3
 - > Transform
 - > Direction, 209–10
 - > Lower → Higher, 302
 - > Remove Lines
 - > Between Clusters, 187, 260–1
 - > Inside Clusters, 53
- [Main] Operations > Network + Permutation
 - > Reorder Network, 321
- [Main] Operations > Network + Vector
 - > +Cluster
 - > Diffusion Partition, 210
 - > Info > Assortativity, 163–4
 - > Neighbours > Sum
 - > All, 207–8
 - > Input, 207–8
 - > Output, 207–8
 - > Transform > Put Coordinate, 60
- [Main] Operations > Partition + Permutation
 - > Reorder Partition, 321
- [Main] Operations > Vector + Partition
 - > Extract Subvector, 59, 131, 288
 - > Shrink Vector, 59
- [Main] Options > Blockmodel – Shrink, 50–1
- [Main] Options > Read–Write
 - > 0/0, 208
 - > Bipartite Pgraph, 283–4
 - > GEDCOM – Pgraph, 273, 283–4
 - > Ignore Missing Values in menu Vector and Vectors, 216, 376
 - > Max. vertices to draw, 400
 - > Ore: Different relations for male and female links, 273, 274–5
 - > Pgraph + labels, 283–4
 - > Threshold, 393–4
- [Main] Partition drop-down menu, 39
- [Main] Partition, 45
 - > Binarize Partition, 207, 287–8
 - > Copy to Vector, 56, 231
 - > Count, Min–Max Vector, 361
 - > Create Constant Partition, 44, 390
 - > Create Random Partition > 1-Mode, 106
 - > Info, 41–2, 78, 90, 188–9, 203, 215, 233, 276, 277, 286–7, 361
 - > Make Cluster
 - > Vertices from selected Clusters, 210, 326
 - > Make Network
 - > Random Network, 362
 - > Make Permutation, 320
- [Main] Partitions drop-down menu, 45
- [Main] Partitions, 45
 - > Add (First + Second), 287–8
 - > Expand Partition
 - > First according to Second (Shrink), 260
 - > Extract SubPartition (Second from First), 48, 90–1, 127, 131, 133–4, 277, 302
 - > Info, 62–3
 - > Cramer's V, Rajscki, Adjusted Rand Index, 62–3, 133–4, 287, 337
 - > Spearman Rank, 230, 237–8, 240–1
 - > Make Random Network, 362
- [Main] Permutation drop-down menu, 320
- [Main] Tools > Excel, 138
- [Main] Tools > Export to Delimited File, 138
- [Main] Tools > R
 - > Locate R, 137–8
 - > Send to R

- [Main] Tools > R (*cont.*)
 - > All Vectors, 137–8
 - > Current Vector, 370
- [Main] Vector drop-down menu, 55
- [Main] Vector > Create Constant Vector, 391, 398
- [Main] Vector > Info, 55, 57, 183, 215–16, 237, 273, 288, 376
- [Main] Vector > Make Partition
 - > by Intervals, 56–7
 - > First Threshold and Step, 56–7, 237–8, 240–1
 - > Selected Thresholds, 286
 - > Copy to Partition by Truncating (Abs), 56, 137, 176–7, 188–9
- [Main] Vector > Transform
 - > Multiply by, 183
 - > Normalize > Max, 57–8
- [Main] Vectors drop-down menu, 207–8
- [Main] Vectors
 - > Divide (First/Second), 207–8, 210
 - > Info, 231
- PajekXXL amd Pajek3XL Main screen**
- [Main] Tools > Pajek
 - > Locate Pajek, 402
 - > Send Network to Pajek> + Add Vertex Labels from File(s), 402
- Report screen**
- [Report] File > Empty Report, 377
- R Commands**
- R: {igraph} power.law.fit(), 370–1
- R: Edit
 - > Run all, 370
 - > Run line or selection, 370–1
- R: File > Open Script, 370

Subject Index

Note: Italicized page numbers with f's or t's refer to figures or tables.

- Acrobat Distiller, 405
- active network, 11
- actors, 5
 - defined, 427*t*
 - input domain, 232
 - participation rate, 122
 - threshold of, 206
 - in two-mode networks, 121
- acyclic networks, 253–5, 284, 289, 293, 427*t*
- Adamic, L. A., 355–7, 381
- adjacency matrix, 317, 427*t*
- adjacent vertices, 427*t*
- Adjusted Rand Index (ARI), 63, 133–4, 287, 337
- Adobe Illustrator, 408
- adoption categories, 204, 427*t*
- adoption rate, 202–3, 215, 427*t*
- adoption time, 198
- affective relations, 99
- affiliation matrix, 317, 427*t*
- affiliations, 119–40
 - brokerage roles and, 184–9
 - communities, 132–5
 - islands, 127–32
 - one-mode networks, 121–7
 - overview, 119–20
 - partitions, 127
 - three-dimensional, 135–9
 - two-mode networks, 121–7
- aggregate constraints, 182, 183, 427*t*. *See also* dyadic constraint
- Albert, R., 366, 382
- Aldenderfer, Mark S., 348
- alpha, 366, 369, 371, 381
- Anatomy of Scottish Capital, The* (Scott and Hughes), 121
- ancestors, 276
 - closest common, 273
 - pedigree, 273
- angle, 418
- arc, 7, 9–10, 155, 362
 - bidirectional, 7, 155
 - colors, 417
 - defined, 427*t*
 - head of, 7
 - tail of, 7
 - traversal weight, 293–4
- articulation points, 173, 176–7, 427*t*
- assortative mixing, 162
- assortativity, 73, 162–4, 428
- assortativity coefficient, 162, 163–4, 428
- asymmetric dyads, 247, 428
- Attiro data, 73–5, 76, 78, 83, 88, 226–7
- attribute, 428
- attribution, 101
- automatic drawing, 20–2
- average degree, 76, 78
- BabelPad, 392
- background colors, 419–20
- backward local main path search, 294, 300
- balance model, 248, 428
- balance theory, 99–102. *See also* structural balance
 - clusterability, 103–9

- balance theory (*cont.*)
 development in time, 109–13
 structural balance, 103–9
 balanced (semi-) cycle, 428
 balanced network, 101, 102
 balanced signed graph, 101, 428
 balance-theoretic models, 250*t*
 Barabási, A.L., 366, 382
 Barabási–Albert model, 366, 367, 372
 Batagelj, Vladimir, 265, 307, 348, 382
 Bernoulli random graph, 358–60, 362
 betweenness, 155–6
 betweenness centrality, 159, 428. *See also*
 centrality
 betweenness centralization, 159, 428
 bi-components, 82–3, 172–7, 284–5, 428.
See also components
 defined, 173–4
 bidirectional arc, 7, 14, 155, 186
 bipartite network. *See* two-mode networks
 bipartite parentage graph, 283–4
 birth cohorts, 287*t*
 bitmap, 26, 404–5
 Blashfield, Roger K., 348
 block, 331
 complete, 332, 339–40
 defined, 428
 null, 332, 339–40
 regular, 339–40, 434
 blockmodel, 332–3, 342*f*, 428
 blockmodeling, 50–1, 333–8
 defined, 428
 generalized, 341, 342
 matrices, 316–21
 overview, 315
 permutation, 316–21
 steps in, 333–4
 blogs, 355–7, 373–5
 Blondel, V. D., 142
 blood marriages, 280, 428. *See also*
 nonblood relinking; structural
 relinking
 blood relations, 271
 Bollobás, B., 382
 Bonacich, Phil, 167
 Boorman, S. A., 348
 border color, 420
 Borgatti, Stephen, 142
 Bornschieer, Volker, 66
 boundary specification, 6
 Brandes, U., 33, 382
 Breiger, Ronald L., 141, 348
 bridge
 bi-components and, 172–7
 defined, 173, 428
 ego-networks, 177–84
 finding in hierarchy of bi-components,
 177
 overview, 170
 brokerage role, 184–9. *See also* coordinator
 role; gatekeeper; itinerant broker;
 liaison
 defined, 185, 428
 in strike network, 186–7
 types of, 185–6
 Brothers Keeper, 306
 Burt, Ronald S., 193, 241
 calculation, in social network analysis,
 15–17
 Carlson, R.O., 219
 Carrington, P.J., 220
 Cartwright, Dorwin, 116
 cell (of a matrix), 428
 centrality
 betweenness, 159
 closeness, 154
 ego-centered approach to, 149
 eigenvector, 160–2
 network, 289–90
 number of neighbors and, 151
 centrality literature data, 289–90, 291–2,
 297*t*, 302, 303*f*
 centralization, 149
 betweenness, 159
 closeness, 154
 degree, 152, 155
 eigenvector, 161
 Cerinšek, M., 307
 Chime, 410
 chi-square statistic, 252
 Church of Jesus Christ of Latter-Day
 Saints, 306
 circles, 45
 citation, 291–304
 analysis, 291
 networks, 291–2, 296
 Clifford, Roy A., 96, 241
 Clip format, 405
 cliques, 86–91, 191–2
 defined, 428
 overlapping, 88
 closeness centrality, 154, 157–8, 428. *See*
also centrality

- closeness centralization, 154, 429
 closest common ancestor, 273
 clusterability, 102, 248
 detecting, 103–9
 clusterability model, 429
 clusterable (semi-) cycle, 429
 clusterable signed graph, 429
 clustering, 361
 clustering coefficients, 184, 359, 361, 365, 376, 429
 clusters, 102, 187, 284, 326, 329
 creating, 326
 ranked, 248–9
 Unicode symbols, 415–16
 vertices, 326
 cohesion concept, 292–3
 cohesive subgroups, 73–93
 cliques, 86–91
 components of, 79–83
 cores of, 83–6
 degree of, 75–9
 density of, 75–9
 example of (Attiro neighborhood in Costa Rica), 73–5
 family–friendship groupings, 73–5
 overview, 73
 Coleman, James S., 219, 241
 colors, names of, 413*t*
 column (matrix), 316
Commodity Trade Statistics, 37
 communication network, 150–1
 betweenness in, 158–60
 distance, 151–8
 of striking employees, 171*f*
 communities, 132–5, 429
 community detection, 429
 complete block, 332, 339–40
 complete dyads, 246
 complete network, 429
 components, 79–83, 429
 bi-components, 82–3
 giant, 359, 360
 strong, 81–3
 weak, 81–3
 conditional uniform random graph models, 359–60
 connected network, 21, 22, 80
 connectedness, 80
 constraints
 aggregate, 182, 183, 427*t*
 dyadic, 180–1, 430
 contagion, 200–3
 contextual view, 429
 continuous property, 53–4
 continuous-time Markov process models, 355
 contours, 82–3, 176–7
 coordinates, 53–60
 coordinator role, 185–6, 189, 429. *See also*
 brokerage role; gatekeeper; itinerant broker; liaison
 CoreDRAW, 406
 core-periphery structure, 325–6, 331, 333–4
 cores, 83–6
 correlation, 229–31
 correlation coefficients, 229
 Pearson's, 230
 Spearman, 229–30
 Cortona VRML Client, 409
 Cramer's V, Rajski, Adjusted Rand Index, 63, 133–4, 287, 337
 createpajek.exe, 391
 critical mass, 211–16, 429
 critical path method, 295
 cross-sectional networks, 111
 cut-vertex, 173, 429
 cycle, 101, 429

 data collection techniques, 27
 free recall, 27
 paired comparison, 27
 ranking, 27
 roster, 27
 unrestricted choices, 27
 data objects, 9
 Davis, J. A., 116
 De Nooy, Wouter, 266
 De Solla Price, D., 367, 382
 De Solla Price's model, 367–8, 371
 Degenne, Alain, 167, 193, 348
 degree, 429
 degree assortativity, 162, 163
 degree centrality, 429
 degree centralization, 152, 155, 429
 degree sequence, 361, 429
 degrees of network, 75–9
 dendrogram, 324*f*, 324–5, 429. *See also*
 hierarchical clustering
 density, 429
 descendants, 276
 diffusion
 adoption rate, 202–3
 from central and marginal vertex, 202*f*

- diffusion (*cont.*)
 by contacts, 201*f*
 contagion, 200–3
 critical mass, 211–16
 curve, 201*f*, 200–1
 exposure, 204–10
 modern math diffusion data, 197–8
 overview, 197
 thresholds, 204–10
- diffusion curve, 430
- dining-table partners data, 4*f*, 4–5, 6, 7–8, 9, 11–12, 15
- directed graph (digraph), 7, 8, 430
- disassortativity, 162
- dissimilarity, 324*t*, 326–7
- distance, 430
- distance distribution, 157
- divide et impera* (divide-and-rule) strategy, 178
- domain, 231–5, 430
 input, 232
 restricted input, 233
- Doreian, Patrick, 115, 116, 265, 307, 348
- Doreian–Mrvar Method, 106, 111–12
- Dutch literary criticism data, 265
- dyad, 245–6, 253, 430
 asymmetric, 247, 428
 complete, 246
 null, 247
 symmetric, 247, 250*t*
- dyadic constraint, 180–1, 430. *See also*
 aggregate constraints
- edge, 7, 155, 417, 430
- ego-centered approach, 178, 430
- egocentric density, 182, 183–4, 430
- ego-networks, 177–84
 defined, 430
 density of, 182
 dyadic constraint, 180
 proportional strength of ties, 180
- eigenvector centrality, 160–2, 430. *See also*
 centrality
- eigenvector centralization, 161, 430
- Encapsulated PostScript, 405–6, 412–18, 420
- endogamy, 279, 430
- equivalence, 322–31
 class, 322
 defined, 430
 regular, 338–43, 434
 structural, 323–4, 435
- Erdős, P., 358, 381
- Erdős–Rényi random graph model, 358, 362
- error matrix, 340*f*
- error scores, 105, 111–12, 112*t*
- events, 121, 430
 size of, 122, 435
- Everett, Martin, 142
- Excel, 138
- exponential random graph models (ERGM), 355
- exposure, 204–10, 430
- Extensible 3D (X3D), 409–10
- External-Internal Index, 132–3, 135, 430
- family of child or orientation (FAMC), 271, 431
- family of spouse or procreation (FAMS), 271, 431
- family trees, 270–8
- family–friendship groupings, 73–5
- Faulkner, Robert R., 142
- Faust, Katherine, 32, 96, 115, 142, 167, 241, 265, 348
- Fennema, Meindert, 142
- Ferligoj, Anuška, 265, 266, 348
- Fernandez, Roberto M., 193
- first-order inflection point, 212
- FishEye* mode, 58–9, 431
- fixed choices, 27
- Flament, Claude, 265, 298–300, 302
- Flux Player, 410
- flying teams data, 115
- forest, 431
- Forsé, Michel, 167, 193, 348
- forward local main path, 294
- fragments, 88–9, 286
- free choices, 27
- free recall method, 27
- Freeman, Linton, 32, 167, 289, 298–300, 302, 307
- Fruchterman–Reingold, 21–2, 138–9
- Galesburg drug study data, 219, 237–8, 240–1
- Garfield, Eugene, 307, 381
- gatekeeper, 185–6, 431. *See also* brokerage
 role; coordinator role; itinerant
 broker; liaison
- GDP per capita, 53–4, 55, 56, 57, 61
- Gecko package, 423

- GEDCOM (genealogical data format), 271, 273, 274–5, 277, 283–4, 306, 388
- genealogical generation, 272, 431
- genealogy, 269–70
- family trees, 270–8
 - social research, 278–88
- generalized blockmodeling, 341, 342, 431
- generalized random graph models, 359–60
- generation jump, 273, 288, 431
- geodesic, 153–4, 157*f*, 156–7, 275–6, 431
- GhostScript, 405
- GhostView, 405
- giant component, 359, 360. *See also* components
- Gilbert, E. N., 382
- Gil-Mendieta, Jorge, 348
- Glance, N., 355–7
- global main path, 431
- key-route, 295
 - standard, 295, 298–300
- global main path methods, 295
- global view, 431
- Gondola family tree, 270–3, 274*f*
- Gornik, Miha, 421
- Gould, Roger V., 193
- Granovetter, Mark, 193, 219
- graph, 7, 9–10, 431
- directed (digraph), 7, 8, 430
 - Ore, 272*f*, 272, 274–5, 276, 280*f*, 433
 - parentage (P-graph), 282*f*, 281–3, 434
 - signed, 100, 101, 102, 428, 429, 435
 - simple, 435
 - undirected, 7, 8, 436
- graph drawing esthetics, 18
- graph theory, 6
- Guillaume, J.-L., 142
- Hage, Per, 115
- Harary, Frank, 115, 116
- head (of an arc), 7
- Heider, Fritz, 99, 101, 116
- hierarchical clustering, 324, 328*f*, 329*f*, 431
- hierarchical clusters model, 249, 431
- hierarchy, 89–90, 431
- hi-tech unionization data, 192–3
- Hlebec, Valentina, 266
- Holland, Paul W., 265, 382
- Hollywood composers data, 141
- homophily, 73, 432
- Hughes, Michael, 121, 142
- Hummon, Norman P., 307
- image matrix, 332, 340*f*, 432
- immediacy index, 291, 432
- impact factor, 291, 432
- incident, 432
- indegree, 227–9, 432
- independence, 76
- induced subnetwork, 46–8, 89, 432
- Inkscape, 406, 408
- innovativeness, 206
- input domain, 231–5
- Instant Player, 410
- Institute for Scientific Information (ISI), 291
- interactive innovation, 213
- interlocking directorates, 120–1, 399–400
- intermarriage, 279
- islands, 130*f*, 127–32
- defined, 432
 - landscape of, 138*f*
 - in three dimensions, 136*f*
- Isle of Man genealogical data, 306
- isomorphic network, 318, 432
- itinerant broker, 185–6, 187, 432. *See also* brokerage role; gatekeeper; liaison
- Johnsen, Eugene C., 265
- joint stock companies, 120–1
- Jorion, Paul, 306
- JPEG, 26, 404–5
- Kadushin, Charles, 141
- Kalish, Y., 382
- Kamada, T., 33
- Kamada-Kawai, 21, 25, 42, 45, 90–1, 106, 131
- Katz, Elihu, 219, 241
- Katz, L., 382
- Kawai, S., 33
- k*-connected component, 432
- k*-core, 83, 296, 432
- key-route global main path, 295
- key-route local main path search, 294–5, 300
- key-routes, 432
- Kick, E., 66
- Kincaid, D. Lawrence, 166, 167
- Kinemages, 410–11
- KiNG, 411
- k*-Neighbors, 155, 276
- Knoke, David, 241
- Kolaczyk, E. D., 381
- Korea family planning data, 166

- Krackhardt, David, 193
 Kramberger, A., 266
- labels, 415
 landscape, 136*f*, 138*f*, 137–8
 LaTeX, 405
 Lefebvre, E., 142
 Leinhardt, Samuel, 265, 382
 Leonard, Olen E., 96, 241
 liaison, 185–6, 187, 432. *See also*
 brokerage role; coordinator role;
 gatekeeper; itinerant broker
- Lin, Nan, 193, 241
 line (network), 7, 45. *See also* arc; edge;
 loop
 defined, 432
 labels, 417–18
 removing, 50
 line multiplicity, 124, 126
 line values, 8, 126
 similarities, 182–3
 line-network, 152*f*, 151–2
 Liu, J. S., 307
 local clustering coefficient, 361. *See also*
 clustering coefficients
 local main path methods, 295, 300, 432
 local view, 432
 longitudinal networks, 109–11
 Loomis, Charles P., 96, 241
 loop, 7, 432
 Louvain method, 132, 133
 Lu, L. Y. Y., 307
 Luce, R. D., 96
 Lusher, D., 382
- macros, 208, 288
 Mage, 410–11
 Mahnken, Irmgard, 306
 main path, 294
 analysis, 292–3
 backward local, 294, 300
 component, 296, 303*f*, 433
 defined, 432
 forward local, 294
 global, 295
 key-route local, 294–5
 local, 295
 as multirelational network, 301
 manipulation, in social network analysis,
 12–15
 manual drawing, 22–5
 marriages, 269–70
 blood, 280
 family trees and, 270–8
 intermarriage, 279
 multiple, 283
 nonblood relinking, 280
 polygamy, 283
 remarriages, 276, 280–1, 283
 structural relinking and, 280–1
- Massey, J. G., 167
 matrilineal lines, 286
 matrix, 316–21, 433
 adjacency, 317, 427*t*
 affiliation, 427*t*
 cell, 428
 column, 316
 error, 340*f*
 image, 332, 340*f*, 432
 row, 316
 matrix format, 388, 393–4
 matrix multiplication, 277
 M-clusters model, 249
 MDL file, 342–3
 MDL MOL, 410–11
 Menzel, Herbert, 219, 241
 Mexican political elite data, 347
 Michael, Judd H., 167, 193
 Microsoft Access, 395, 398
 Microsoft Word, 398, 405
 Milgram, Stanley, 32
 missing values, 376
 modern math diffusion data, 197–8
 adopters, 211–12
 adoption rate and acceleration, 211–12
 early adopters, 204
 exposure of vertices, 204–5
 threshold, 206
 modularity, 132, 433
 Monte Carlo simulation, 373–7
 Morales, Julio O., 96, 241
 Moreno, J. L., 3, 32, 115
 Mrvar, A., 115, 116
 multiple lines, 7–8, 372, 433
 multiple relations network, 9, 433
 multiplex network, 433
 multiplicity, 433
- neighbor, 433
 nested subnetwork, 433
 network, 6–8, 45
 active, 11
 acyclic, 253–5, 284, 289
 centrality, 289–90
 combining, 60
 complete, 429

- creating, 28
- cross-sectional, 111
- defined, 433
- degree centralization of, 152
- degree sequence of, 361
- diameter of, 361
- editing, 28–9
- extracting, 46–7
- extracting subnetwork from, 46–7
- extracting vector values from, 59
- line-network, 151–2
- longitudinal, 109–11
- manipulation of, 12–15
- motifs, 375, 433
- multiple relations, 9, 433
- multiplex, 433
- one-mode, 121–7, 433
- optimal layout of, 18
- parts of, 8
- reduction of, 45–53
 - contextual view, 51–3
 - global view, 48–51
 - local view, 46–8
- relations, 8
- shrinking, 48, 255, 435
- star-network, 151, 152*f*, 154
- strongly connected, 80–1
- symmetrizing, 77–8, 436
- transposed, 436
- two-mode, 121–7, 436
- unconnected directed, 79*f*
- valued, 124, 436
- weakly connected, 80–1
- network data formats, 387–9
- network format, 388
- network growth models, 366
- Newman, M. E., 167, 381, 382
- NoClip* export, 405
- nonblood relinking, 280, 433. *See also*
 - blood marriages; structural,
 - relinking
- Norman, Robert Z., 116
- NotePad++, 392
- null block, 332, 339–40
- null dyads, 247, 433

- one-mode networks, 121–7, 433
- optimization technique, 105, 433
- Ore graph, 272*f*, 272, 274–5, 276, 280*f*, 433
- outdegree, 433
- paired comparison, 27

- Pajek
 - automatic drawing methods, 22
 - blockmodeling commands in, 335–8
 - coordinate system of, 136*f*
 - creating network files for, 389–400
 - helper software, 391
 - within Pajek, 389–91
 - relational database, 394–400
 - word processor, 392–4
 - data objects, 9
 - dialog box, 14*f*
 - Draw screen, 18–20, 25, 54–5, 136–9, 425–6
 - export formats, 404–11
 - bitmap, 404–5
 - Encapsulated PostScript, 405–6
 - Extensible 3D (X3D), 409–10
 - JPEG, 404–5
 - Kinemages, 410–11
 - MDL MOL, 410–11
 - Scalable Vector Graphics, 406–8
 - VOSviewer, 408–9
 - VRML, 409–10
 - installing, 387
 - limitations of, 400
 - Mac OS X, installing on, 421–3
 - Main screen, 11–12, 424
 - menu structure in, 13*f*
 - names of colors in, 413*t*
 - network data formats, 387–9
 - Options screen, 411*f*
 - project file, 39–41
 - Report screen, 16–17, 77
 - scrollbar, 137
 - shortcut key combinations, 424–6
 - updates of, 402–3
- Pajek3XL, 387, 400–2
- PajekToSvgAnim.exe, 407–8
- PajekXXL, 387, 400–2
- parentage graph (P-graph), 282*f*, 281–3, 434
- partial order, 258, 433
- participation rate, 434
- partitions, 38–45, 90–1, 286–7
 - affiliation, 127
 - binarized, 287–8
 - creating, 44
 - cross-tabulation of, 61–3
 - defined, 433
 - definition of, 38
 - dimensions of, 430
 - editing, 39

- partitions (*cont.*)
 - exporting, 398–9
 - optimizing, 335
 - order of class numbers in, 39
 - removing, 376–7
 - saving, 39
 - as storage of discrete characteristics of vertices, 38
 - translating vectors into, 56
- path, 80, 153, 434
- patrilineal genealogy, 270–8
- patrilineal lines, 286
- Pattison, P., 382
- Pearson's correlation coefficient, 230
- pedigree, 276, 434
- permutation, 316–21, 434
- Perry, A., 96
- Personal Ancestral File, 306
- Pfeffer, Jürgen, 391
- Ph.D. students in computer science data, 306
- Pich, Ch., 33
- Pivot MDS, 22
- Poisson distribution, 359
- polarization, 112
- polygamy, 280, 283
- popularity of vertex, 227–9, 434
- position, 434
- PostScript format, 320–1, 327, 405
- POVRay, 409–10
- Powell, J.H., 382
- power-law distributions, 367
- preferential attachment models, 366–72
- preprint transformation, 434
- preprints, 303
- prestige
 - correlation, 229–31
 - domains, 231–5
 - overview, 225–6
 - proximity, 235–8
 - social, 225
- prevalence, 434
- probability distribution, 354–5, 359–60
- proportional strength, 180, 434
- proximity prestige, 235–8, 434
- Qiew, 409
- R (software), 137–8, 370
- radius, 418
- Ragusan nobility data, 269–70, 279, 283
- random graph models
 - Bernoulli, 358–60, 362
 - classic uniform models, 358–62
 - Erdős–Rényi, 358
 - Monte Carlo simulation, 373–7
 - overview, 353–5
 - Poisson, 359
 - preferential attachment models, 366–72
 - small-world models, 362–6
- ranked clusters, 248–9, 434
- ranked structure (blockmodel), 340–1
- ranking, 27
 - acyclic networks, 253–5
 - overview, 244–5
 - triadic analysis, 245–53
- ray-tracing, 409–10
- reachable, 434
- Read, K., 115
- receiver (head of arc), 7
- regular block, 339–40, 434
- regular equivalence, 338–43, 434
- relational database, 394–400
- relaxed balanced, 106
- relinking index, 282, 284–5, 434
- relocation techniques, 20–1
- remarriages, 276, 280–1, 283
- Rényi, A., 358, 381
- representative role, 434
- resolution parameter, 133
- restricted domain, 435
- restricted input domain, 233
- rewiring, 364, 372, 373–5
- Riordan, O., 382
- Roberts, F. S., 116
- Robins, G., 382
- Rogers, Everett M., 167, 219
- rotation, 137
- row (matrix), 316
- Sabidussi, G., 167, 298–300
- Sampson, Samuel F., 103, 115
- Sampson monastery data, 103
 - clusters, 103–9
 - structural balance, 103–9
- San Juan Sur data, 96, 226–7, 237
- sawmill communication data, 150–1, 155, 157, 159, 272
- Scalable Vector Graphics (SVG), 406–8, 412–19
- Schijf, Huibert, 142
- Schmidt, Samuel, 348
- Schweitzer, Thomas, 306
- Scientometrics data, 381

- Scott, John, 32, 96, 121, 142, 167, 220, 348
- Scottish capital data, 120–1
- islands in network of Scottish firms, 130*f*
 - one-mode networks, 121–7
 - two-mode networks, 121–7
- Search Path Count, 296
- Search Path Link Count, 296
- Search Path Node Pair, 296
- secondary structural hole, 184–5, 187–8, 435
- second-order inflection of S-curve, 211
- Seidman, S. B., 96
- semicycle, 101, 435
- semipath, 80, 435
- semiwalk, 79–80, 435
- sender (tail of arc), 7
- shortcut key combinations, 424–6
- sibling, 273
- sibling group, 276, 279
- signed graph, 100, 435
- balanced, 101, 428
 - clusterability, 102
 - clusterable, 429
- signed network, 106
- Simmel, Georg, 119, 141, 178, 193
- simple graph, 435
- sink vertex, 293, 435
- Small World problem, 6
- small-world models, 362–6, 373–5
- Smith, David H. A., 66
- Sneath, Peter, 348
- Snijders, T.A.B., 382
- Snyder, D., 66
- sociability, 147
- social atom, 3
- social capital, 147, 151, 435
- social circles, 119
- social generation, 273, 435
- social network analysis, 5–17, 30
- assembling a social network, 27–30
 - estimation techniques in, 6
 - main goal of, 5
 - manipulation, 12–15
 - network definition, 6–8
 - statistics and, 61–3
 - visualization, 17–27
 - automatic drawing, 20–2
 - manual drawing, 22–5
 - saving a drawing, 25–7
- social prestige, 225
- society, 3
- sociocentered perspective, 149, 435
- sociogram, 4*f*, 3–5
- sociometric choice, 4
- sociometry, 3–5
- Sokal, Robert R., 348
- source vertex, 293, 435
- Spearman's rank correlation, 229–30
- Spencer, J., 382
- spring embedders, 20
- standard global main path, 295, 298–300
- star-network, 151, 152*f*, 154
- betweenness centrality, 159
 - defined, 435
- statistical network model, 355, 435
- statistics, 61–3
- Steglich, C. E. G., 382
- strength-of-weak-ties hypothesis, 176
- strike network data, 171–2, 174–5, 186–7
- binary matrix, 320*f*
 - communication lines, 316*f*
 - coordinator roles in, 188*t*
 - matrix, 318*f*
- Strogatz, S.H., 382
- strong component, 435
- strongly connected, 435
- structural balance, 101
- detecting, 103–9
- structural equivalence, 323–4, 435
- structural hole, 178, 182, 436
- secondary, 184–5, 187–8
- structural prestige, 225
- structural property, 39, 436
- structural relinking, 279, 280–1, 282*f*, 436.
- See also* blood marriages; nonblood relinking
- student government data, 245
- error matrix, 340*f*
 - image matrix, 340*f*
 - matrix, 339*f*
- subnetwork
- complete, 86–91
 - extracting, 431
 - induced, 46, 89, 302, 325–6, 335
 - nested, 433
- subtree, 329–30
- SVG file, 131–2
- symbiosis, 163
- symbols, 415–16
- symmetric clusters, 256–61
- symmetric dyads, 247, 250*t*
- symmetric-acyclic decomposition, 256–61, 436

- tab delimited file, 138
- tertus gaudens* strategy, 178, 182
- Textpad, 392
- threshold, 204–10, 436
- threshold category, 436
- threshold lag, 214, 215–16, 436
- ties
 - asymmetric, 245, 258, 260
 - strong, 175, 325–6
 - weak, 170, 175
- transaction networks, 185–6
- transitive triad, 436
- transitivity model, 249, 436
- transposed network, 436
- traversal weight, 293*f*, 293–4, 297*t*, 436
- tree, 282, 436
- Trezzini, Bruno, 66
- triad, 178*f*, 178, 245–53, 436
- triad census, 251*t*, 252*t*, 250–2, 436
- Tusnady, G., 382
- two-mode networks, 121–7, 394, 399–400, 436
- txt2Pajek3.exe, 391
- txt2Pajek.exe, 391

- UCINET DL files, 389
- undirected graph, 7, 8, 436
- Unicode symbols, 415–16
- Unicode UTF-8 with BOM format, 392

- Valente, Tom W., 220
- valued network, 124, 436
- Van de Bunt, G.G., 382
- vector graphics, 26–7
- vectors, 53–60
 - defined, 437
 - drawing, 57–8
 - exporting, 398–9
 - removing, 376–7
 - shrinking, 59
 - size of vertices and, 57
 - translation into partitions, 56
- velocity, 418
- vertex/vertices, 7, 9, 10
 - adjacent, 76
 - attributes of, 39
 - average degree of, 76
 - class numbers, 43–5, 156
 - closeness centrality of, 154
 - colors, 42–3, 156, 156*f*
 - coordinates, 59–60
 - cut-vertex, 173
 - defined, 437
 - degree centrality of, 152–5, 167
 - degree of, 76
 - deleting, 173, 429
 - discrete characteristics of, 38
 - distance between, 153–4
 - distribution of distances between, 157
 - exposure of, 204
 - geodesics between, 157*f*, 156–7
 - indegree of, 76
 - input domain, 232
 - labels, 58, 415
 - labels, editing, 390
 - layout of, 412*f*
 - movement of, 23
 - outdegree of, 76
 - popularity or indegree of, 227–9
 - proximity prestige of, 236
 - shapes of, 412–14, 415
 - sink, 293, 435
 - source, 293, 435
 - symbols, 44
 - threshold of, 208–9
 - traversal weight, 293–4
 - uniting, 60
- Virtual Reality Modeling Language (VRML), 409–10
- visiting ties, 73–5
- visualization, 17–27
 - automatic drawing, 22–5
 - manual drawing, 22–5
 - saving a drawing, 25–7
- VOS Clustering, 132, 133
- VOS Mapping, 22
- VOSviewer, 408–9
- vrm12pov.exe, 410

- walk, 79–80, 437
- Wallerstein, Immanuel, 36, 66
- Wasserman, Stanley, 32, 96, 115, 142, 167, 220, 241, 265, 348, 382
- Watts, D. W., 382
- Watts–Strogatz clustering coefficient, 361, 364. *See also* clustering coefficients
- weakly connected network, 437
- Web of Science*, 291
- White, Douglas R., 306
- White, Harrison C., 348
- Wolff, Kurt H., 141
- word processor, 392–4
- WordPad, 392
- world system, 36–8

- in South America, *48f*
- world trade of manufactures of metal,
39
- world trade data, 39–41, 325–6, *328f*, 335,
337–8, 395
- WYSIWYG export, 405
- X3D models, 138–9,
409–10
- XQuartz, 421
- z*-axis, 135
- Zeleny, Leslie D., 115