LGN 5822 - Biometrical Genetics

L02 – Introduction to R

Michele Jorge Silva Siqueira

2023

Introduction to R

Purpose

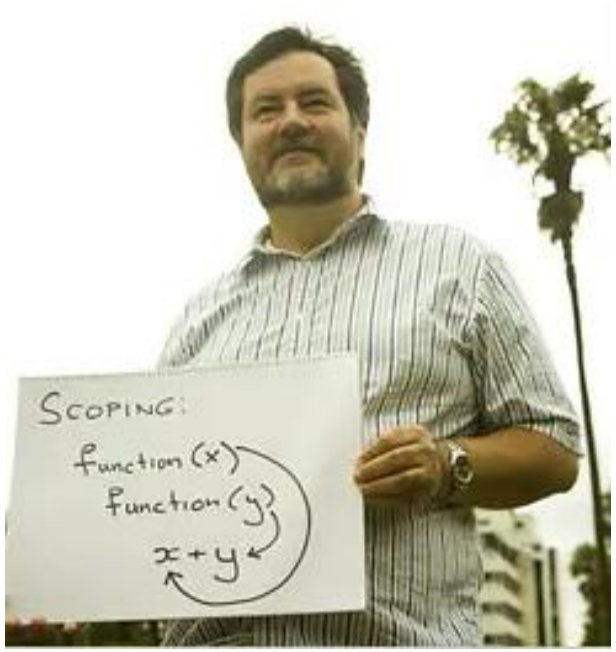
The idea of this class is to present the basic commands and concepts about the R language for genetic and bioinformatic students

Introduction to R

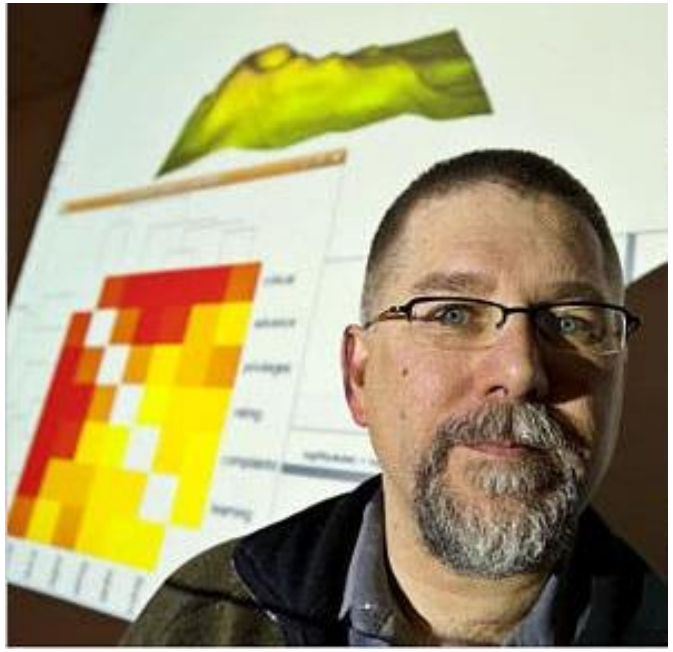
Introduction to R

Did you know?

- Created by



Ross Ihaka



Robert Gentleman

Introduction to R

R: A Language for Data Analysis and Graphics

Ross IHAKA and Robert GENTLEMAN

In this article we discuss our experience designing and implementing a statistical computing language. In developing this new language, we sought to combine what we felt were useful features from two existing computer languages. We feel that the new language provides advantages in the areas of portability, computational efficiency, memory management, and scoping.

Key Words: Computer language; Statistical computing.

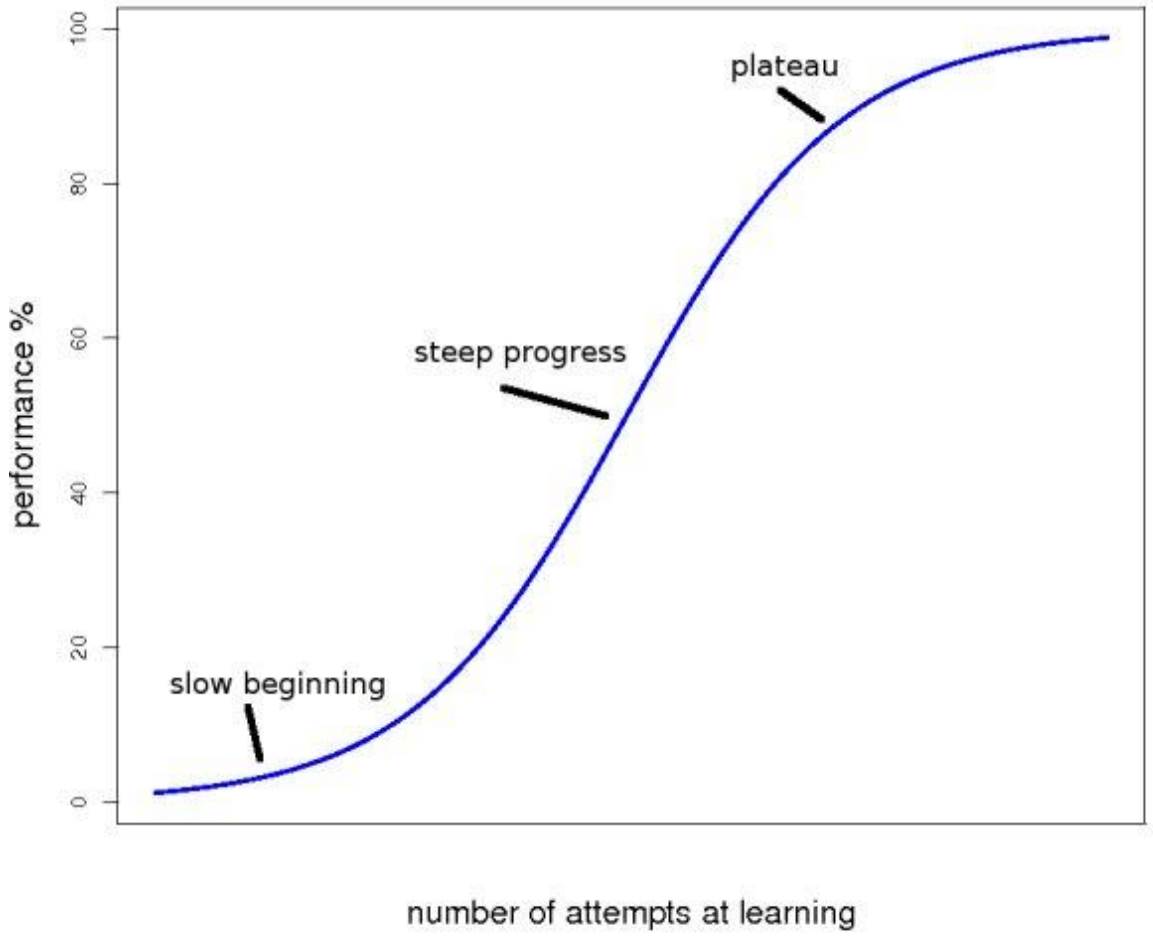
Ihaka, Ross, and Robert Gentleman. "R: A Language for Data Analysis and Graphics." *Journal of Computational and Graphical Statistics*, vol. 5, no. 3, 1996, pp. 299–314.

Introduction to R

"R changed my opinion of humanity to some extent, to see how people are really willing to freely give of themselves and produce something larger than themselves without any thought of personal glory."

Ross Ihaka

Learning - Patience!



Introduction to R

Learning - Patience!

Constant practice is essential to becoming proficient in any programming language

Introduction to R

Important Steps to Learn in R

- Basics of Language
- Data Structures in R
- Functions in R
- Packages in R
- Data Manipulation
- Data Visualization
- Practice and Learn with Projects
- Use the Learning Resources
- Join the Community

Introduction to R

Errors are common! Learn through experience!

Introduction to R

Why use R?

- Programming language
- Free software
- Open source
- Extensive
- Large community of users
- Large variety of “packages”
- Comprehensive environment
- Compatible with other Programming Languages
- Excellent for Statistical Analysis
  - Modeling
  - Data Manipulation
  - Powerful Graphics
  - Data Visualization

Introduction to R

The R Studio

- IDE: Integrated Development Environment
- Its interface has of the tools required to work with R in one place (console, source, plots, workspace, help, history, etc.)
- Free software

Introduction to R

The R Studio

RStudio is becoming Posit

After install R, I strongly advise you to use RStudio for your firsts R contact

Introduction to R

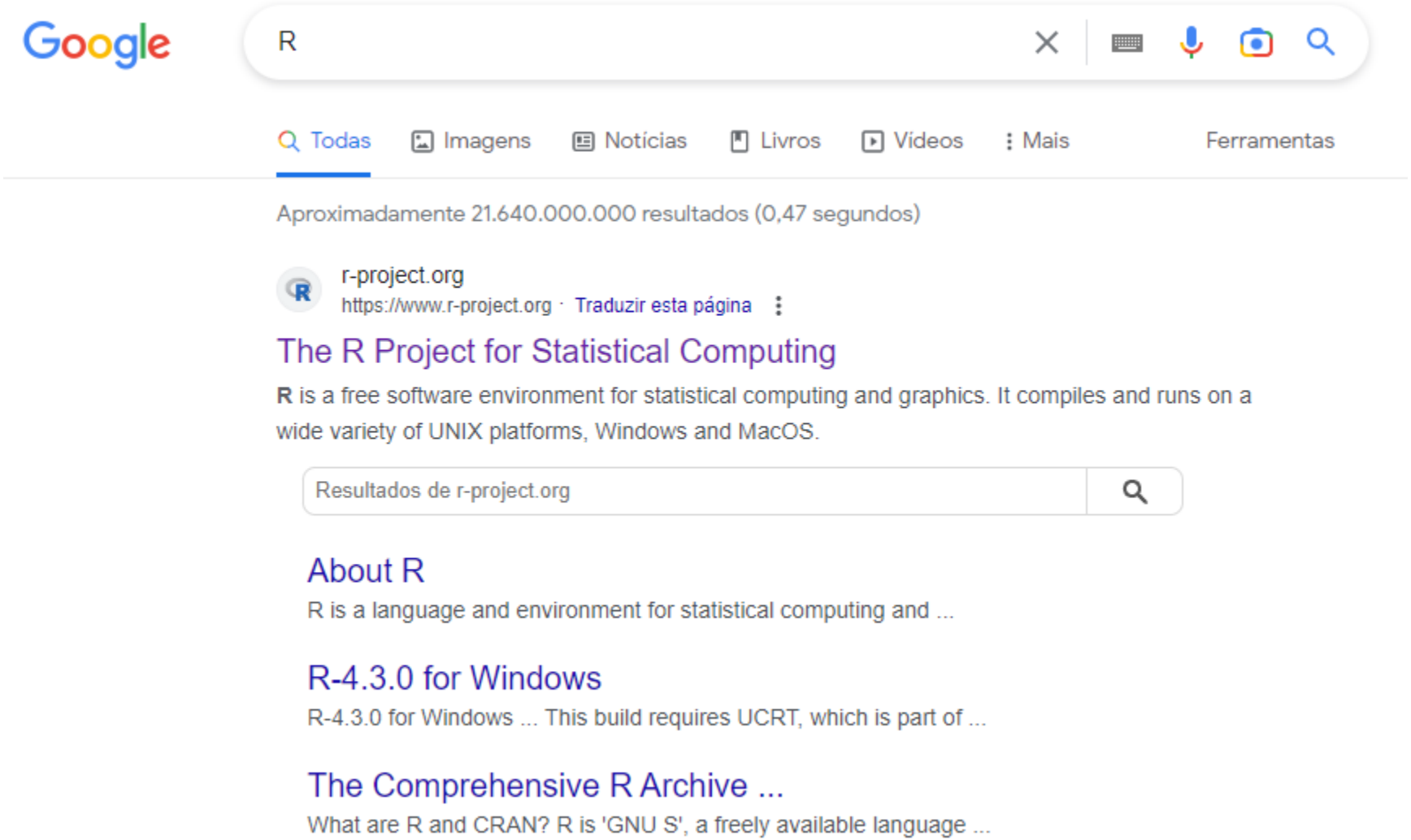
The R Studio

R: Engine	RStudio: Dashboard
	

http://moderndive.com/

Installation

Install R



Installation

Install R

[Home]

Download

CRAN

R Project

About R
Logo
Contributors
What's New?
Reporting Bugs
Conferences
Search
Get Involved: Mailing Lists
Get Involved: Contributing
Developer Pages
R Blog

R Foundation

The R Project for Statistical Computing

Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To **download R**, please choose your preferred CRAN mirror.

If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

News

- **R version 4.3.0 (Already Tomorrow)** has been released on 2023-04-21.
- **R version 4.2.3 (Shortstop Beagle)** has been released on 2023-03-15.
- Thanks to the organisers of useR! 2020 for a successful online conference. Recorded tutorials and talks from the conference are available on the R Consortium YouTube channel.
- You can support the R Foundation with a renewable subscription as a supporting member

News via Twitter

The R Foundation Retweeted

https://www.r-project.org

Installation

Install R

CRAN Mirrors

The Comprehensive R Archive Network is available at the following URLs, please choose a location close to you. Some statistics on the status of the mirrors can be found here: main page, windows release, windows old release.

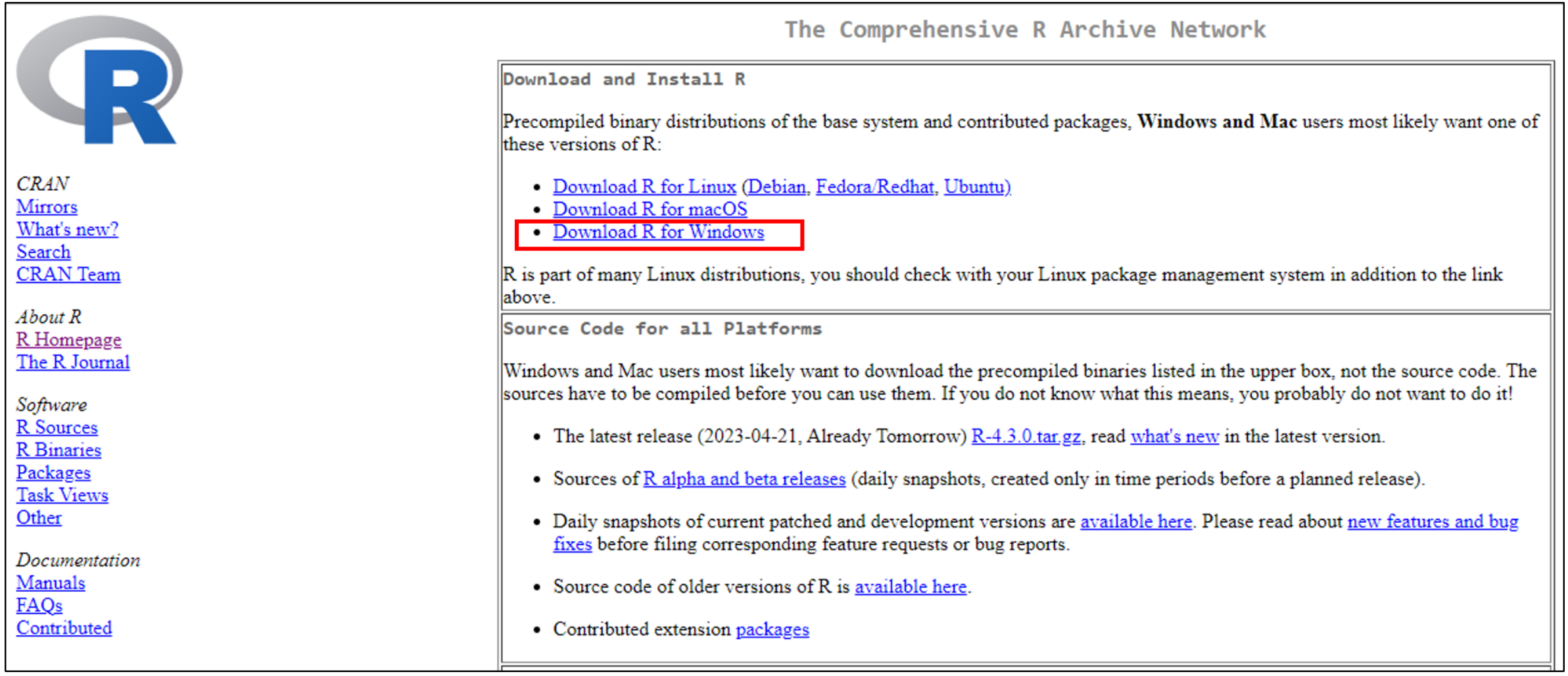
If you want to host a new mirror at your institution, please have a look at the CRAN Mirror HOWTO.

0-Cloud	
https://cloud.r-project.org/	Automatic redirection to servers worldwide, currently sponsored by Rstudio
Argentina	
http://mirror.fcaglp.unlp.edu.ar/CRAN/	Universidad Nacional de La Plata
Australia	
https://cran.csiro.au/	CSIRO
https://mirror.aarnet.edu.au/pub/CRAN/	AARNET
https://cran.ms.unimelb.edu.au/	School of Mathematics and Statistics, University of Melbourne
https://cran.curtin.edu.au/	Curtin University
Austria	
https://cran.wu.ac.at/	Wirtschaftsuniversität Wien
Belgium	
https://www.freestatistics.org/cran/	Patrick Wessa
https://ftp.belnet.be/mirror/CRAN/	Belnet, the Belgian research and education network
Brazil	
https://cran-r.c3sl.ufpr.br/	Universidade Federal do Parana
https://cran.fiocruz.br/	Oswaldo Cruz Foundation, Rio de Janeiro
https://vps.fmvz.usp.br/CRAN/	University of Sao Paulo, Sao Paulo
https://brieger.esalq.usp.br/CRAN/	University of Sao Paulo, Piracicaba

BRieger
https://brieger.esalq.usp.br/CRAN/
Since 2003

Installation

Install R



CRAN
Mirrors
What's new?
Search
CRAN Team

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Task Views
Other

Documentation
Manuals
FAQs
Contributed

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux (Debian, Fedora/Redhat, Ubuntu)
- Download R for macOS
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

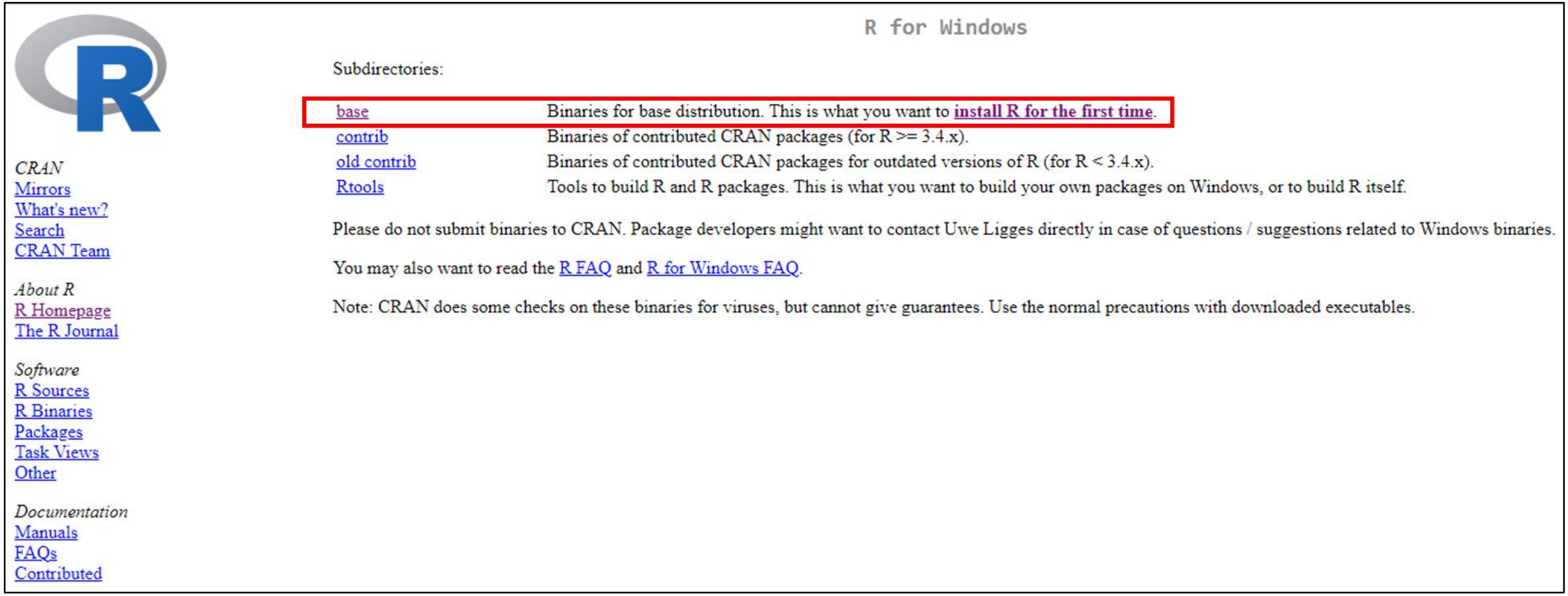
Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2023-04-21, Already Tomorrow) R-4.3.0.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

Installation

Install R



R for Windows

Subdirectories:

base	Binaries for base distribution. This is what you want to **install R for the first time**.
contrib	Binaries of contributed CRAN packages (for R >= 3.4.x).
old contrib	Binaries of contributed CRAN packages for outdated versions of R (for R < 3.4.x).
Rtools	Tools to build R and R packages. This is what you want to build your own packages on Windows, or to build R itself.

Please do not submit binaries to CRAN. Package developers might want to contact Uwe Ligges directly in case of questions / suggestions related to Windows binaries.

You may also want to read the R FAQ and R for Windows FAQ.

Note: CRAN does some checks on these binaries for viruses, but cannot give guarantees. Use the normal precautions with downloaded executables.

CRAN
Mirrors
What's new?
Search
CRAN Team

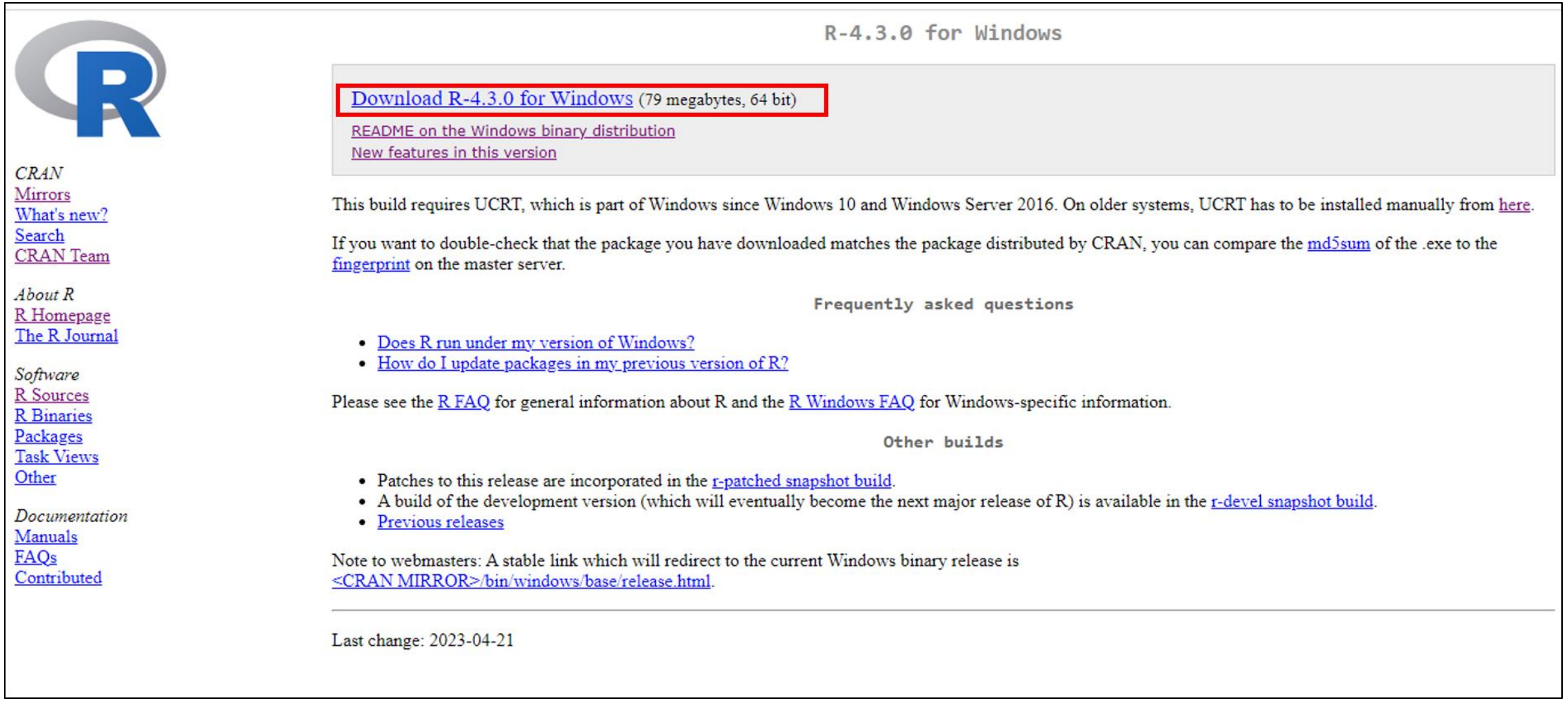
About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Task Views
Other

Documentation
Manuals
FAQs
Contributed

Installation

Install R



CRAN
Mirrors
What's new?
Search
CRAN Team

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Task Views
Other

Documentation
Manuals
FAQs
Contributed

R-4.3.0 for Windows

Download R-4.3.0 for Windows (79 megabytes, 64 bit)

README on the Windows binary distribution
New features in this version

This build requires UCRT, which is part of Windows since Windows 10 and Windows Server 2016. On older systems, UCRT has to be installed manually from here.

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the md5sum of the .exe to the fingerprint on the master server.

Frequently asked questions

- Does R run under my version of Windows?
- How do I update packages in my previous version of R?

Please see the R FAQ for general information about R and the R Windows FAQ for Windows-specific information.

Other builds

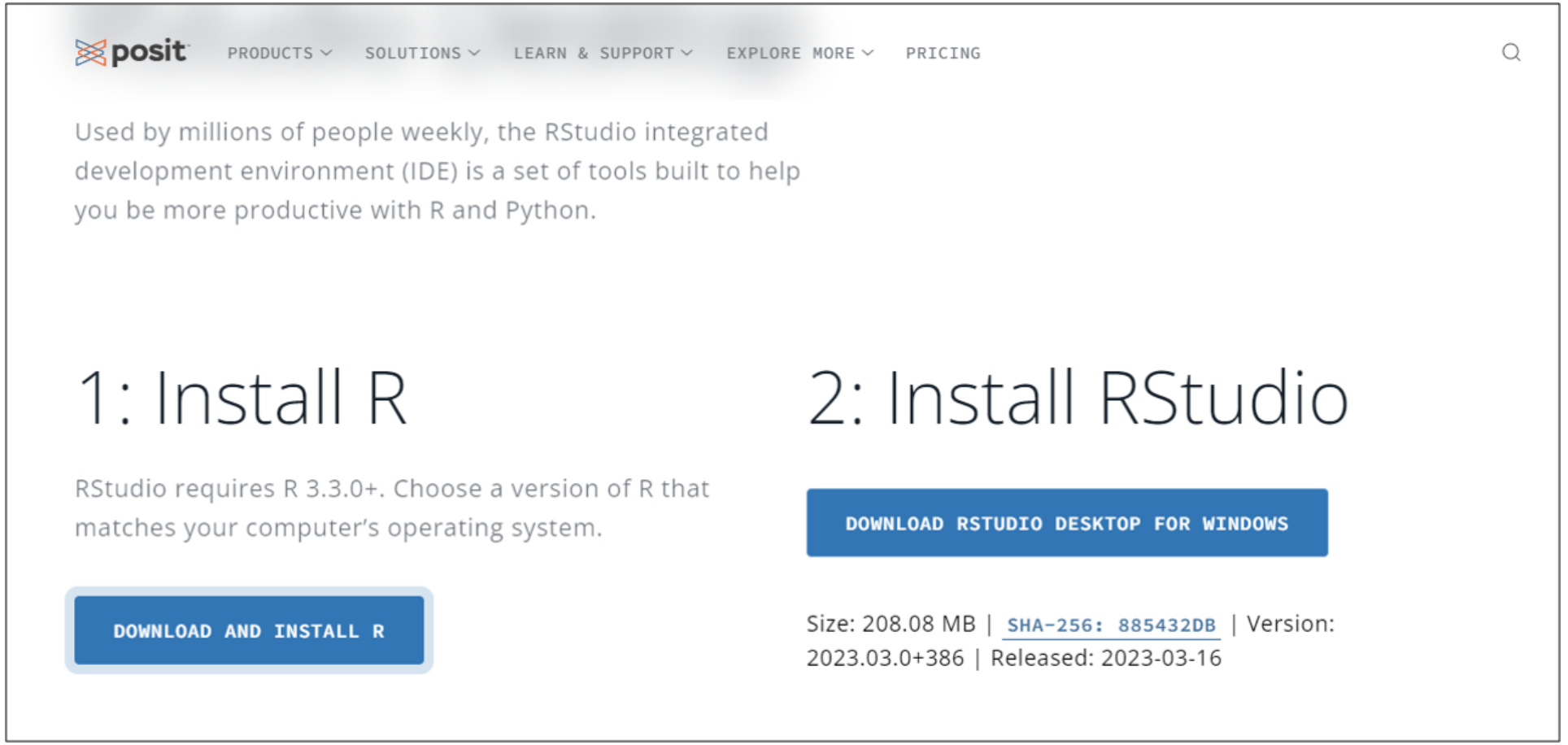
- Patches to this release are incorporated in the r-patched snapshot build.
- A build of the development version (which will eventually become the next major release of R) is available in the r-devel snapshot build.
- Previous releases

Note to webmasters: A stable link which will redirect to the current Windows binary release is <CRAN MIRROR>/bin/windows/base/release.html.

Last change: 2023-04-21

Installation

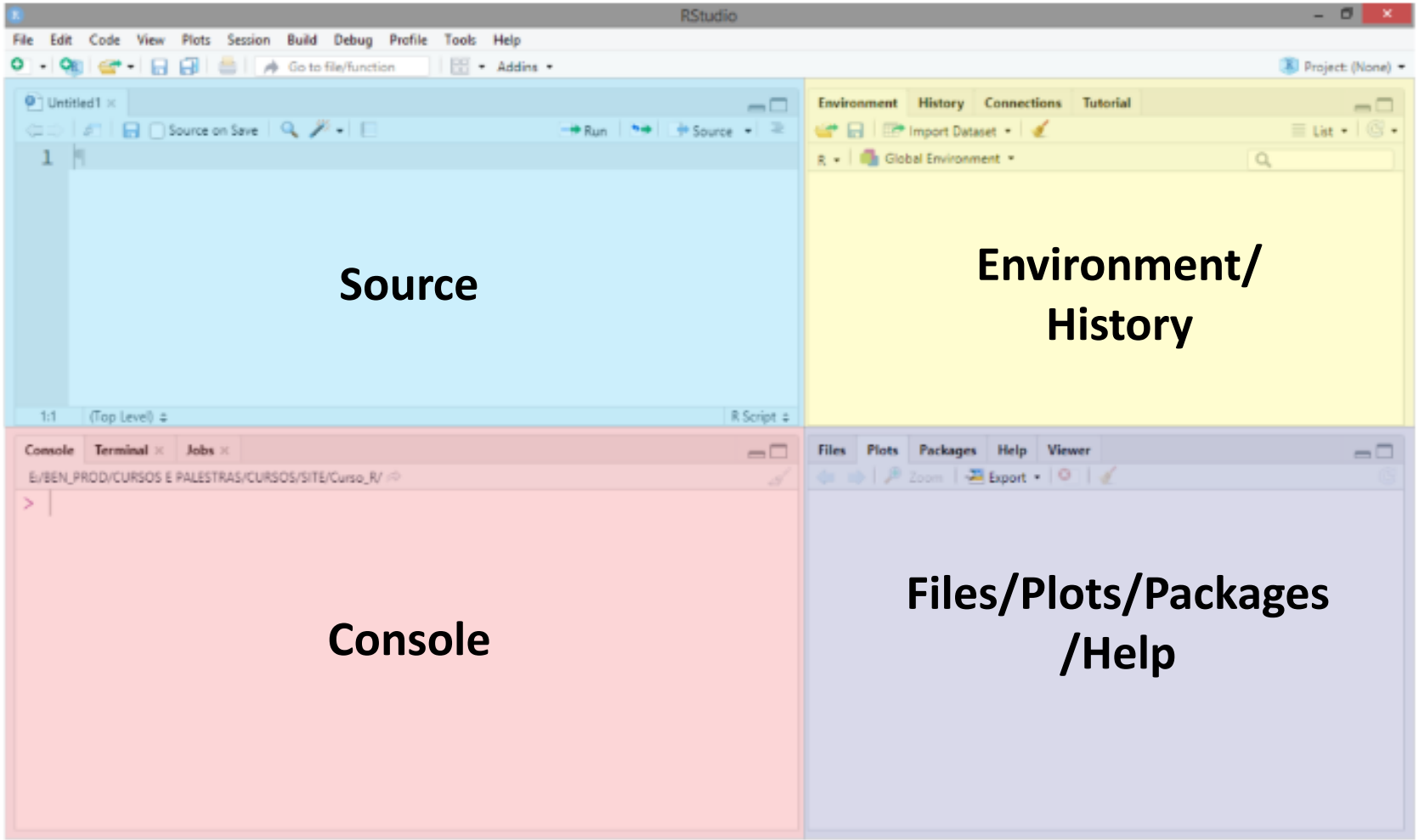
Install R Studio



https://www.rstudio.com

Installation

R Studio



Interface do RStudio (Versão 1.4.1103).

Installation

R Studio: Console

- Looking at the console, you will see the > symbol.
- This symbol indicates the line where you must enter the commands. Click this symbol to position the cursor on the command line and type your first command in R: 2 * 3 and **enter**

Installation

R Studio: Console

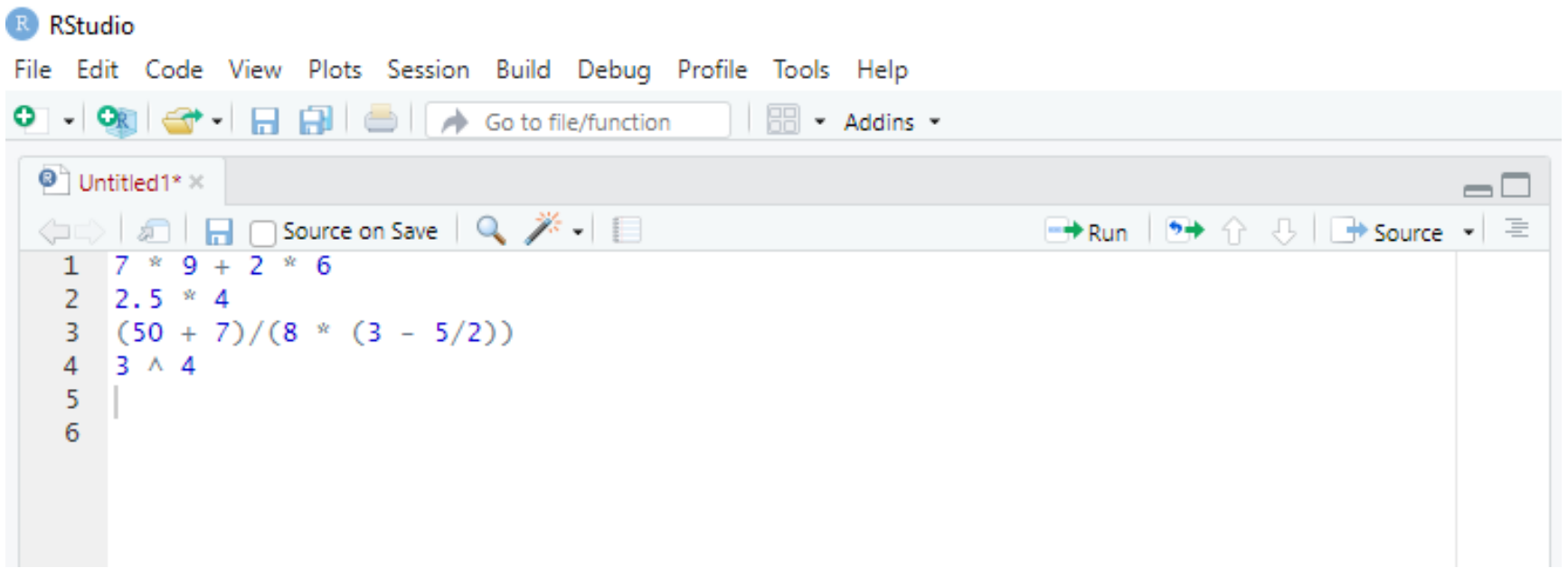
- Clear Console: To clear all messages and console output, you can use keyboard shortcut "Ctrl + L"

Installation

R Studio: Source

- To run the script click on Source or *ctrl + shift + enter*.

- To run the command click on Run or *ctrl + enter*.



Getting Help with R

Help is available!

- The help( ) function or ? is used to access help information about functions, packages, datasets, and other objects available in the R environment.

```
help(function_name)
# or
?function_name
```

Getting Help with R

Help is available!

```
> help(seq)
```

seq {base} R Documentation

Sequence Generation

Description

Generate regular sequences. `seq` is a standard generic with a default method. `seq.int` is a primitive which can be much faster but has a few restrictions. `seq_along` and `seq_len` are very fast primitives for two common cases.

Usage

```
seq(...)

## Default S3 method:
seq(from = 1, to = 1, by = ((to - from)/(length.out - 1)),
    length.out = NULL, along.with = NULL, ...)
```

Getting Help with R

Help is available!

```
> ?mean
```

mean {base} R Documentation

Arithmetic Mean

Description

Generic function for the (trimmed) arithmetic mean.

Usage

```
mean(x, ...)

## Default S3 method:
mean(x, trim = 0, na.rm = FALSE, ...)
```

Arguments

Getting Help with R

Help is available!

```
> help(package="agricolae")
```

Statistical Procedures for Agricultural Research

Documentation for package ‘agricolae’ version 1.3-5

- DESCRIPTION file.
- User guides, package vignettes and other documentation.
- Package NEWS.

Getting Help with R

Help is available!

```
> help(package="ExpDes")
```

Experimental Designs Package

Documentation for package ‘ExpDes’ version 1.2.2

- DESCRIPTION file.

Help Pages

Getting Help with R

Help is available!

```
> help(package="ggplot2")
```

Create Elegant Data Visualisations Using the Grammar of Graphics

Documentation for package ‘ggplot2’ version 3.4.3

- DESCRIPTION file.
- User guides, package vignettes and other documentation.
- Package NEWS.

Help Pages

Getting Help with R

Help is available!

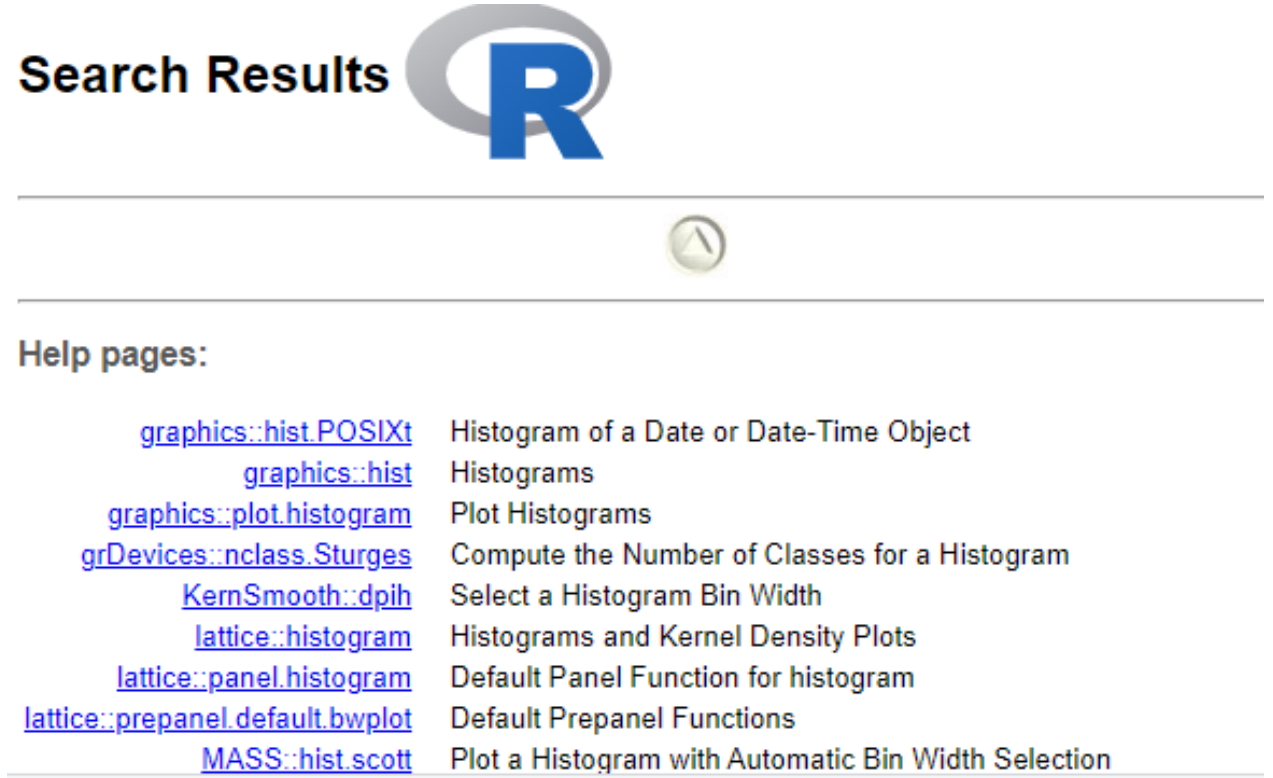
- The help.search( ) function allows you to find information related to a specific topic, functions, packages or other keywords that you supply as arguments

help.search("search word")

Getting Help with R

Help is available!

```
> help.search("histogram")
```



Getting Help with R

Help is available!

```
> help.search("dispersão")
```

Search Results

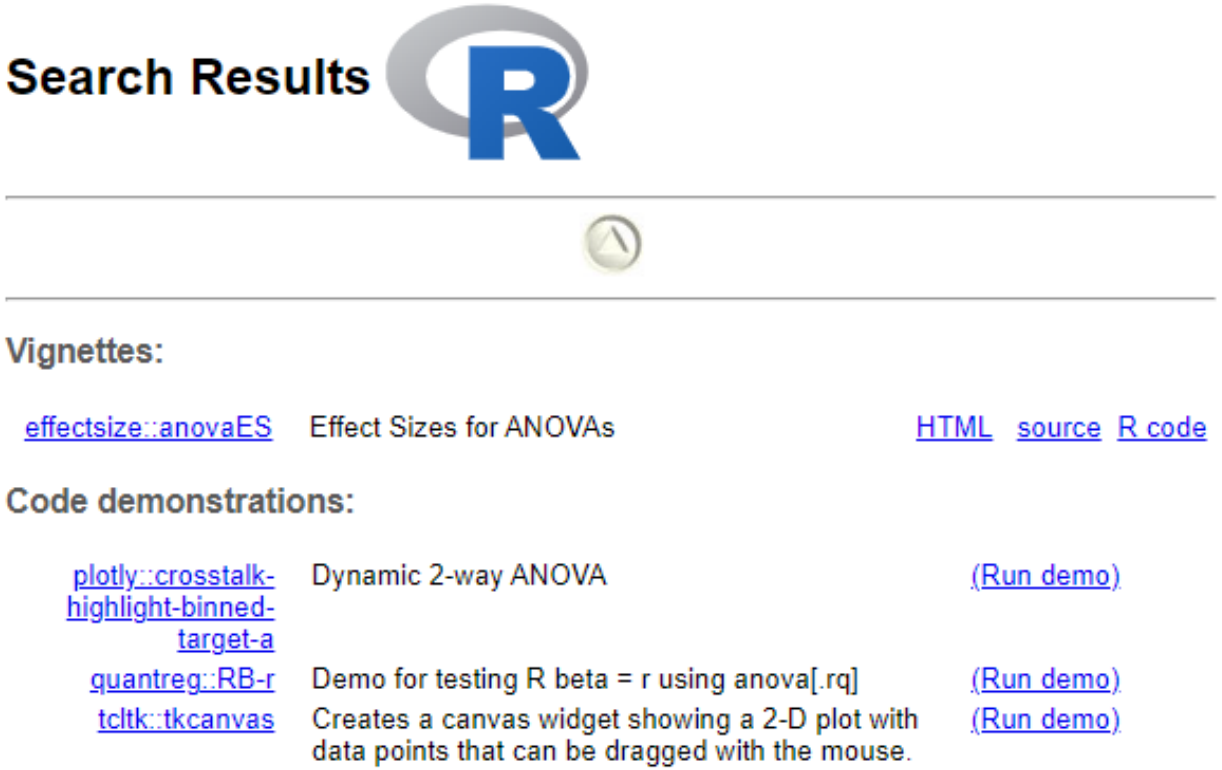
Help pages:

insight::get_variance	Get variance components from random effects models
MASS::gamma.dispersion	Calculate the MLE of the Gamma Dispersion Parameter in a GLM Fit
performance::check_overdispersion	Check overdispersion of GL(M)M's
statmod::fitNBP	Negative Binomial Model for SAGE Libraries with Pearson Estimation of Dispersion
statmod::remlscoregamma	Approximate REML for Gamma Regression with Structured Dispersion

Getting Help with R

Help is available!

```
> help.search("t.test")
```



Search Results

Vignettes:

effectsize::anovaES	Effect Sizes for ANOVAs	HTML source R code

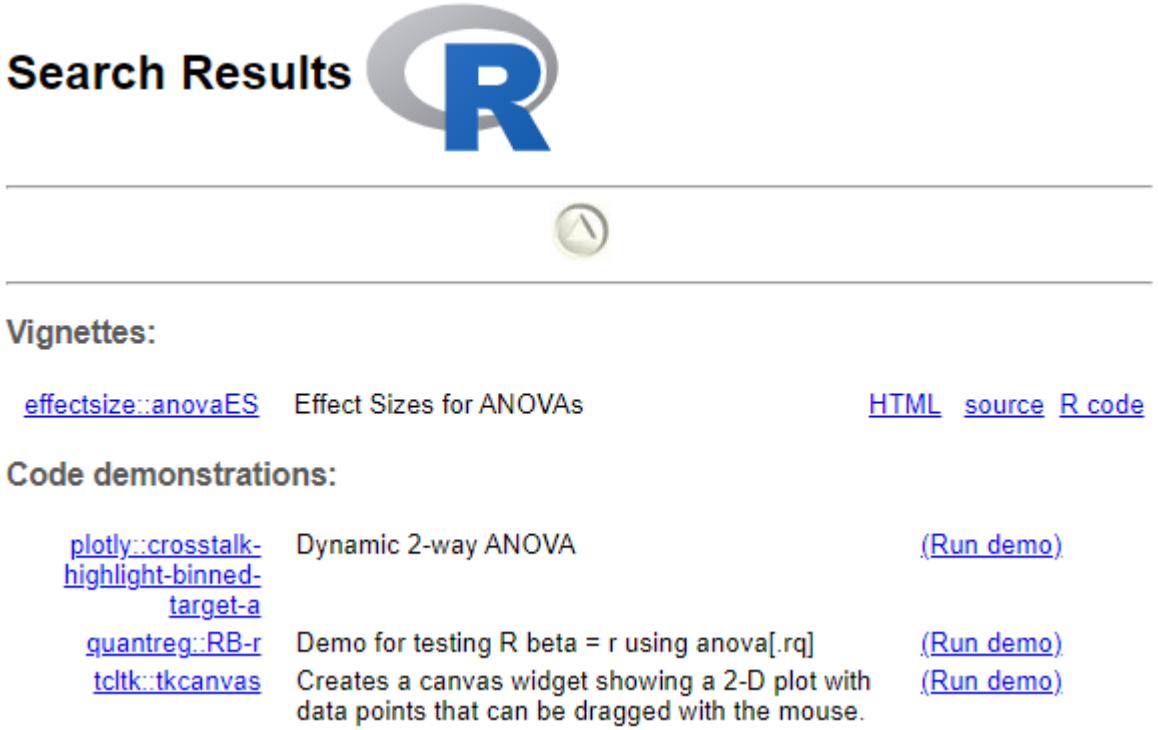
Code demonstrations:

plotly::crosstalk-highlight-binned-target-a	Dynamic 2-way ANOVA	(Run demo)
quantreg::RB-r	Demo for testing R beta = r using anova[.rq]	(Run demo)
tcltk::tkcanvas	Creates a canvas widget showing a 2-D plot with data points that can be dragged with the mouse.	(Run demo)

Getting Help with R

Help is available!

```
> help.search("anova")
```



Search Results

Vignettes:

effectsize::anovaES	Effect Sizes for ANOVAs	HTML source R code

Code demonstrations:

plotly::crosstalk-highlight-binned-target-a	Dynamic 2-way ANOVA	(Run demo)
quantreg::RB-r	Demo for testing R beta = r using anova[.rq]	(Run demo)
tcltk::tkcanvas	Creates a canvas widget showing a 2-D plot with data points that can be dragged with the mouse.	(Run demo)

Getting Help with R

Install package

- To install packages you can use the install.packages( ) function.
- This function allows you to install packages directly from the CRAN (Comprehensive R Archive Network) package repository

```
install.packages("name_of_package")
```

Getting Help with R

Install package

```
#install packages

install.packages("BGLR")
install.packages("MASS")
install.packages("lme4")

install.packages(c("ggplot2", "dplyr", "tidyr"))
```

Getting Help with R

Library function

- Remember that to use a package after installation you need to load it into your R session using the library( ) function

#library("name_of_package")
#require("name_of_package")

```
library("ggplot2")
library("lme4")
require("devtools")
```

Getting Help with R

Swirl package

Learn R, in R.

swirl teaches you R programming and data science interactively, at your own pace, and right in the R console!

Getting Help with R

Swirl package

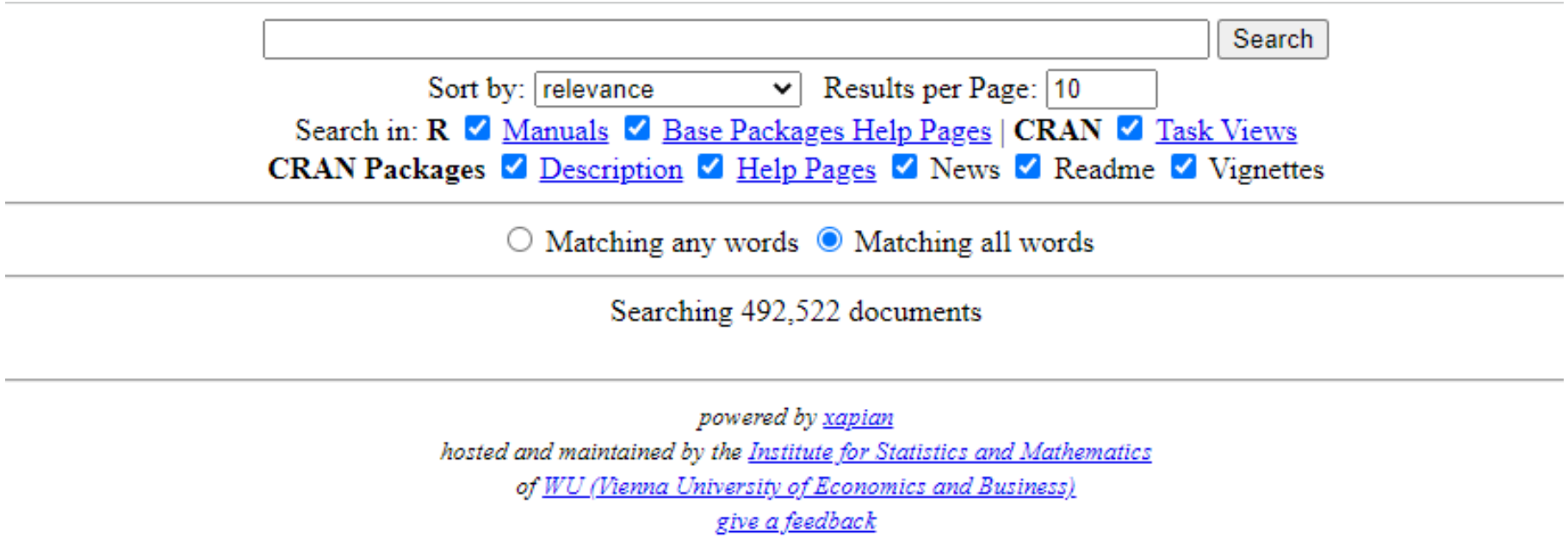
```
What shall I call you? require("devtools")

| Please don't use any quotes or other punctuation in your name.

What shall I call you? Michele

| Thanks, Michele. Let's cover a couple of quick housekeeping items before we begin
| our first lesson. First of all, you should know that when you see '...', that
| means you should press Enter when you are done reading and ready to continue.

...  <-- That's your cue to press Enter to continue

| Also, when you see 'ANSWER:', the R prompt (>), or when you are asked to select
| from a list, that means it's your turn to enter a response, then press Enter to
| continue.

Select 1, 2, or 3 and press Enter

1: Continue.
2: Proceed.
3: Let's get going!

Selection: 1
```

#bye( ): to exit swirl

R Help on the Internet

R Site Search

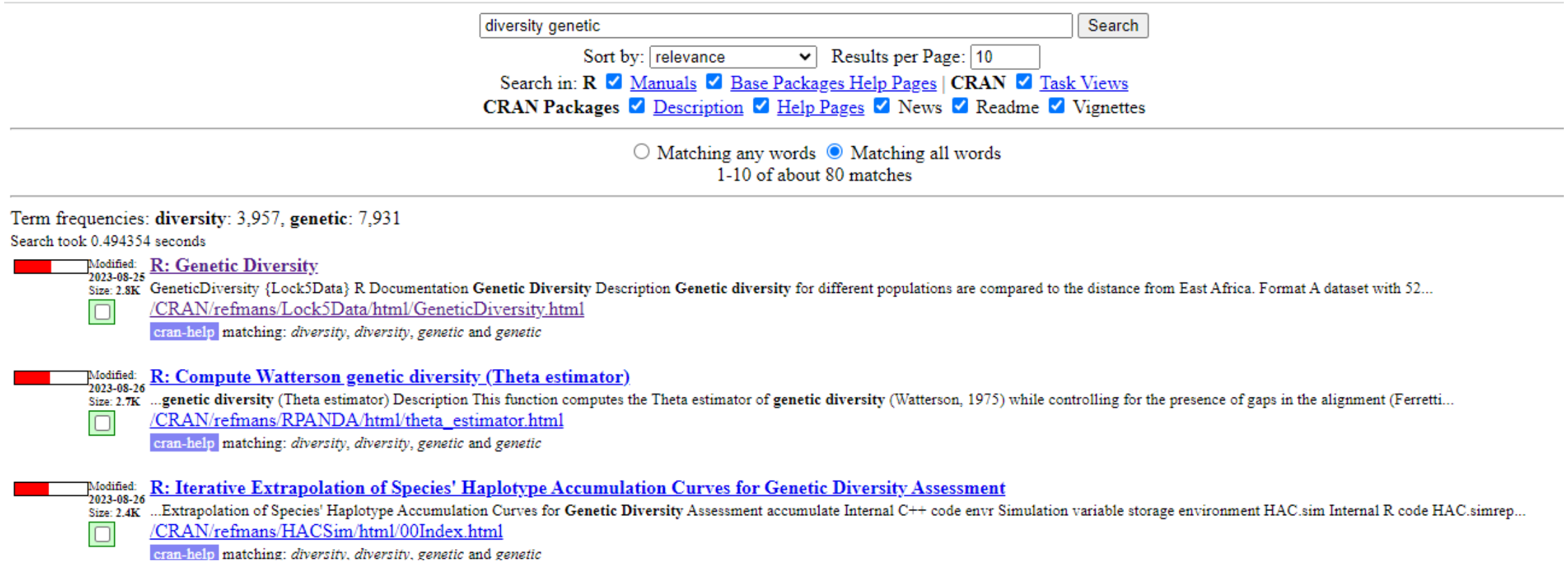
https://search.r-project.org/



R Help on the Internet

R Site Search

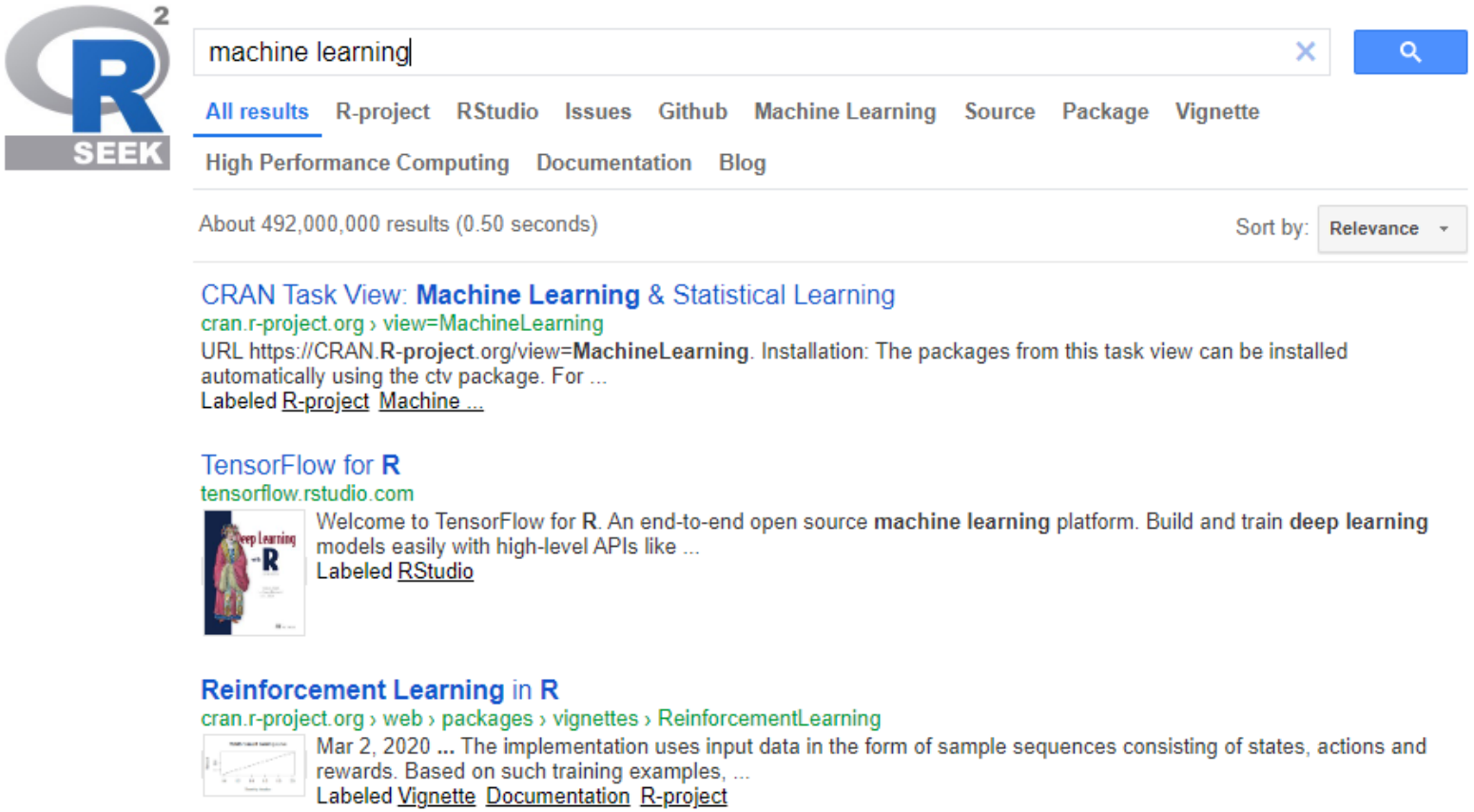
https://search.r-project.org/



R Help on the Internet

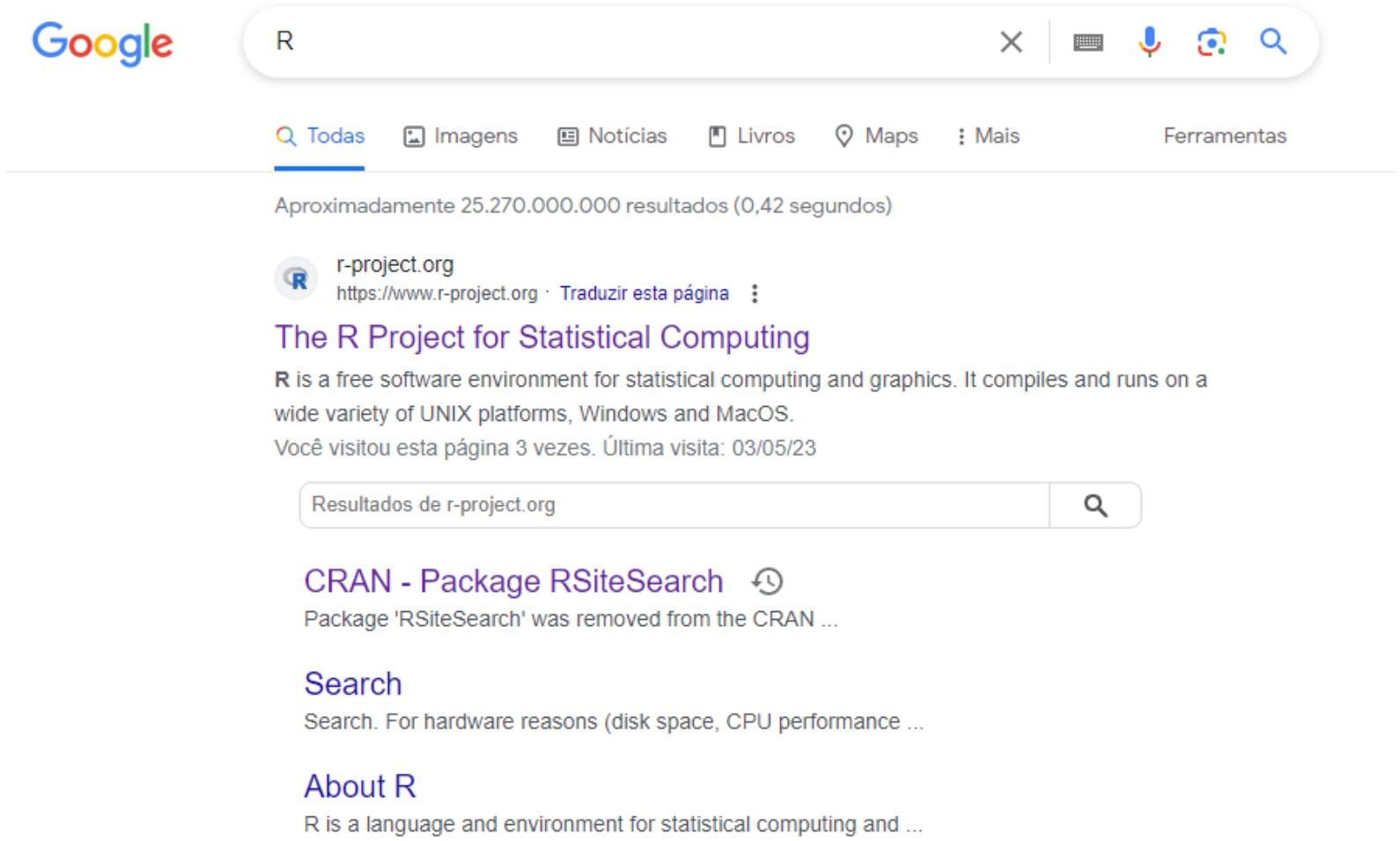
Rseek

http://www.rseek.org



R Help on the Internet

Or just Google R!



R Help on the Internet

Mailing Lists

https://www.r-project.org/mail.html

[Home]

Download

CRAN

R Project

About R
Logo
Contributors
What's New?
Reporting Bugs
Conferences
Search
Get Involved: Mailing Lists
Get Involved: Contributing
Developer Pages
R Blog

Mailing Lists

Please read the instructions below and the posting guide *before* sending anything to any mailing list!

Thanks to Martin Maechler (and ETH Zurich), there are five general mailing lists devoted to *R*.

R-announce

This list is for *major* announcements about the development of *R* and the availability of new code. It has a *low volume* (typically only a few messages a month) and everyone mildly interested should consider subscribing, but note that `R-help` gets everything from `R-announce` as well, so you don't need to subscribe to both of them.

Note that the list is *moderated* to be used for announcements mainly by the R Core Development Team. Use the web interface for information, subscription, archives, etc.

R-help

The 'main' *R* mailing list, for discussion about problems and solutions encountered using *R*, including using *R* packages in the standard *R* distribution and on CRAN; announcements (not covered by `R-announce` or `R-packages`); the availability of new functionality for *R* and documentation of *R*; and for posting nice examples and benchmarks.

Programming

#I can comment or annotate my scripts by writing after the # sign

R as calculator

Arithmetic Operators

Sum

```
> 1+4
[1] 5
> 2+5
[1] 7
```

Subtract

```
> 6-3
[1] 3
> 8-4
[1] 4
```

Divide

```
> 10/2
[1] 5
> 10/3
[1] 3.333333
```

Multiplication

```
> 2*3
[1] 6
> 4*1.2
[1] 4.8
```

Exponential

```
> 2^2
[1] 4
> 2**2
[1] 4
```

Square Root

```
> sqrt(16)
[1] 4
> sqrt(64)
[1] 8
> 9^0.5 #or
[1] 3
> sqrt(9)
[1] 3
```

Log

```
> log(10)
[1] 2.302585
> log(10,base = 10)
[1] 1
> log(10,base = 2)
[1] 3.321928
```

R as calculator

Comparison operators

==: Equal to

!=: Not equal to

<: Less than

>: Greater than

<=: Less than or equal to

>=: Greater than or equal to

R as calculator

Comparison operators

```
> 3*2==5 #equal to (# caution: '==' is different than '=')
[1] FALSE
>
> 4!=1 #not equal to
[1] TRUE
>
> 10>4 #greater than
[1] TRUE
>
> 3<5 #less than
[1] TRUE
>
> 10>=10 #greater than or equal to
[1] TRUE
>
> 12<=10 #less than or equal to
[1] FALSE
>
> 10!=5 #different than
[1] TRUE
```

R as calculator

Let´s Practice!

```
a <- 5
b <- 3

a == b
a != b
a < b
a > b
a <= b
a >= b
```

R as calculator

Parentheses are important!

```
> 4*5+2-3/8
[1] 21.625
>
> 4*(5+2)-(3/8)
[1] 27.625
>
> 4*(5+2-3/8)
[1] 26.5
>
> ((4*5)+2-3)/8
[1] 2.375
```

Simple Functions

Logical operators

- Another important class is the logical whice can be TRUE or FALSE.

```
> 40 & 5 > 30
[1] FALSE
>
> 40 | 5 > 30
[1] TRUE
```

```
> 1<0
[1] FALSE
>
> 1==0
[1] FALSE
>
> 1>0
[1] TRUE
```

#Pay attention here, '==' is different than '='!

Simple Functions

Logical operators

```
> p <- TRUE
>
> q <- FALSE
>
> p & q
[1] FALSE
>
> p | q
[1] TRUE
```

Summary statistics

Summary statistics for your data

- You can compute summary statistics for your data using various built-in functions.

Summary statistics

Summary statistics for your data

- Here are some commonly used functions to calculate summary statistics:

```
# Calculate the mean of a numeric vector
mean(numeric_vector)

# Calculate the median of a numeric vector
median(numeric_vector)

# Calculate the standard deviation of a numeric vector
sd(numeric_vector)

# Calculate the variance of a numeric vector
var(numeric_vector)
```

Summary statistics

Summary statistics for your data

```
> d <- c(10, 8, 12, 7, 6, 9, 8, 13, 10, 7)
>
> mean(d)
[1] 9
>
> median(d)
[1] 8.5
>
> var(d)
[1] 5.111111
>
> sd(d)
[1] 2.260777
```

Simple Functions

- There are a lot of functions in R. A function is represented by a name and can be used calling it names followed by arguments

```
?functionname
```

Simple Functions

- The seq( ) function is used to generate sequences of numbers in R

```
?seq
```

Sequence Generation

Description

Generate regular sequences. `seq` is a standard generic with a default method. `seq.int` is a primitive which can be much faster but has a few restrictions. `seq_along` and `seq_len` are very fast primitives for two common cases.

Usage

```
seq(...)

## Default S3 method:
seq(from = 1, to = 1, by = ((to - from)/(length.out - 1)),
    length.out = NULL, along.with = NULL, ...)

seq.int(from, to, by, length.out, along.with, ...)
```

Simple Functions

```
> seq(from=0, to=100, by=3) #we can write in this way, or
 [1]  0  3  6  9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57 60 63 66 69
[25] 72 75 78 81 84 87 90 93 96 99
>
> seq(0,100,3) #make sequence
 [1]  0  3  6  9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57 60 63 66 69
[25] 72 75 78 81 84 87 90 93 96 99
```

Simple Functions

Let´s Practice!

- Create a sequence of integers from 1 to 10:

- Create a sequence of integers from 0 to 20, in increments of 2:

- Create a descending sequence from 10 to 1:

Simple Functions

Let´s Practice!

- Create a sequence of integers from 1 to 10:

```
x <- seq(1, 10)
```

- Create a sequence of integers from 0 to 20, in increments of 2:

```
x <- seq(0, 20, by = 2)
```

- Create a descending sequence from 10 to 1:

```
x <- seq(10, 1, by = -1)
```

Simple Functions

- The rep( ) function in R is used to create arrays or sequences by repeating specified elements or values.

```
?rep
```

Replicate Elements of Vectors and Lists

Description

`rep` replicates the values in `x`. It is a generic function, and the (internal) default method is described here.

`rep.int` and `rep_len` are faster simplified versions for two common cases. Internally, they are generic, so methods can be defined for them (see InternalMethods).

Usage

```
rep(x, ...)

rep.int(x, times)

rep_len(x, length.out)
```

Simple Functions

```
> rep(x=1,times=3) #or
[1] 1 1 1
>
> rep(1,3)
[1] 1 1 1
```

Simple Functions

Let´s Practice!

- Repeat value 3 five times:

- Repeat vector [1, 2, 3] twice:

Simple Functions

Let´s Practice!

- Repeat value 3 five times:

```
x <- rep(3, times = 5)
```

- # Repeat vector [1, 2, 3] twice:

```
y <- c(1, 2, 3)
z <- rep(y, times = 2)
```

Programming

#Pay attention, R is case-sensitive (i.e. ‘X’ is different than ‘x’).
When in doubt, write everything of your script in lower case!

R creating objects

- You can create objects to store data, calculation results, functions

- Simple Attribution: Use the <- or = operator to assign a value to an object

```
x <- 10
name <- "João"
```

- Vectors: Vectors are one-dimensional objects. You can create arrays using the c( ) function

```
v <- c(1, 2, 3, 4, 5)
```

Matrix: Matrixs are two-dimensional objects. You can create matrices using the matrix() function.

R creating objects

- Simple Attribution: Use the <- or = operator to assign a value to an object

```
x <- 10
name <- "João"
```

- Vectors: Vectors are one-dimensional objects. You can create vectors using the c( ) function

```
v <- c(1, 2, 3, 4, 5)
```

Matrix: Matrix are two-dimensional objects. You can create matrices using the matrix() function.

```
matriz <- matrix(1:9, nrow = 3, ncol = 3)
```

R creating objects

- To create a object use the following syntax

```
> x=10

> x*2
[1] 20
>
> x^5
[1] 1e+05
>
> x+2
[1] 12
```

```
> x<-5

> x*5
[1] 25
>
> x^5
[1] 3125
>
> x+1
[1] 6
```

Your object name '**x**' followed by '=' and then your variable
In some script you will see this '**<-**' symbol

R creating objects

- You can also have a character object (which can not be used in math operations)

```
> x_name = "name"
> x_name
[1] "name"
```

To see which type your object is

```
> class(x)
[1] "numeric"
> class(x_name)
[1] "character"
```

R Data Types

Vectors

- In R, an vector is a fundamental data structure that stores an ordered sequence of elements of the same type

- Elements can be numbers, characters (strings), logical values (Boolean) or factors

- You can create vectors to store a sequence of values using the c( ) function ("combine" or "concatenate")

R Data Types

Vectors

- Numerical Vector

  x<- c(1, 2, 3, 4, 5)

- Character Vector (Strings)

  y<- c(“apple", "banana", “orange", “grape")

- Logical Vector (Boolean)

  z<- c(TRUE, FALSE, TRUE, FALSE)

R Data Types

Vectors

- Vector with Numeric Sequence

```
x<- seq(1, 10)
```

- Repeating Vector

```
x<- rep(0, times = 5)
```

- Vector with Randomly Generated Values

```
x<- runif(5)
```

R Data Types

- Single Element Vector

```
> print("abc")
[1] "abc"
>
> print(12.5)
[1] 12.5
>
> print(63L)
[1] 63
>
> print(TRUE)
[1] TRUE
>
> print(2+3i)
[1] 2+3i
```

R Data Types

Vector Manipulation

```
> # Create two vector
> v1 <- c(3,8,4,5,0,11)
> v2 <- c(4,11,0,8,1,2)
```

- Vector addition

```
> v <- v1+v2
> v
[1]  7 19  4 13  1 13
```

- Vector multiplication

```
> v <- v1*v2
> v
[1] 12 88  0 40  0 22
```

- Vector subtraction

```
> v <- v1-v2
> v
[1] -1 -3  4 -3 -1  9
```

- Vector division

```
> v <- v1/v2
> v
[1] 0.7500000 0.7272727       Inf 0.6250000 0.0000000 5.5000000
```

R Data Types

- Math on vectors

```
> x <- c(10, 20, 40, 80)
> x + 10
[1] 20 30 50 90
```

```
> x * 2
[1]  20  40  80 160
```

```
> y <- c(0, -10, 5, -20)
> x + y
[1] 10 10 45 60
```

R Data Types

- Vector Element Sorting (Ordination)
  - Sort the elements of the vector

```
> v <- c(3,8,4,5,0,11,-9,304)
>
> # Sort the elements of the vector
> v <- sort(v)
> v
[1]  -9   0   3   4   5   8  11 304
```

  - Sort character vectors

```
> v <- c("Red","Blue","Yellow","Violet")
>
> v <- sort(v)
> v
[1] "Blue"   "Red"    "Violet" "Yellow"
```

R Data Types

- Naming vectors

```
> yield <- c(150, 200, 225)
>
> names(yield) <- c("Piracicaba", "Limeira", "Campinas")
> yield
Piracicaba    Limeira   Campinas
       150        200        225
>
> yield["Piracicaba"]
Piracicaba
       150
```

R Data Types

- Example:

```
> x = c(1.5,2.1,2.5,3.4,4.3,6.1) #A vector with numeric value
> y = c("A","A","B","B","C","C") #A vector with character values
>
> str(x)
 num [1:6] 1.5 2.1 2.5 3.4 4.3 6.1
>
> str(y)
 chr [1:6] "A" "A" "B" "B" "C" "C"
```

The above line will create: the 'x' vector with 6 numeric objects, and the 'y' vector with 6 character objects. To visualize this information you can use the str( ) function

This function is really powerful when working with big data, keep it in mind, it gives you all the structure of the object

R Data Types

- Notice that you replace the old x object by this new one. Be careful when naming your objects! With vector, you can make more complex calculus

```
> x = c(1.5,2.1,2.5,3.4,4.3,6.1) #A vector with numeric value
> y = c("A","A","B","B","C","C") #A vector with character values
>
> str(x)
 num [1:6] 1.5 2.1 2.5 3.4 4.3 6.1
>
> str(y)
 chr [1:6] "A" "A" "B" "B" "C" "C"
```

```
> sum(x) #the sum of the vector objects
[1] 19.9
>
> mean(x) #the mean of the vector objects
[1] 3.316667
>
> var(x) #the variance between the vector objects
[1] 2.833667
>
> sum(y) #returns error, because y is not numeric
Error in sum(y) : invalid 'type' (character) of argument
```

Exercises

- Let´s Practice!

- The runif( ) function in R is used to generate random numbers fror continuous uniform distribution
  - runif(n, min = 0, max = 1)

1. Create ten random numbers between 0 and 1

   a) Calculate the sum of these values
   b) Calculate the mean of these values
   c) Calculate the variance of these values
   d) Check which of the ten values are less than 0.05
   e) Check which are greater than or equal to 0.5

Exercises

- Let´s Practice!

2. a) Creates a vector from 1 to 10 by 0.1 of interval
   b) Sample from this vector 4 numbers.

Exercises

1. a) Calculate the sum of these values

```
> x <- runif(10)
> x
 [1] 0.91257127 0.71665176 0.49689256 0.51331487 0.93875172 0.49787134 0.10398766 0.07125127
 [9] 0.81125375 0.88073752
>
> sum(x)
[1] 5.943284
```

b) Calculate the mean of these values

```
> mean(x)
[1] 0.5943284
```

c) Calculate the variance of these values

```
> var(x)
[1] 0.1003734
```

Exercises

d) Check which values are less than 0.05

```
> x[x<0.05]
[1] 0.04362124 0.03175783
```

e) Check which values are greater than or equal to 0.5

```
> x[x>=0.5]
[1] 0.8001303 0.8687504 0.5385993 0.8652702
```

Exercises

2. a) Creates a vector from 1 to 10 by 0.1 of interval:

```
> seq(1,10,0.1) #Creates a vector from 1 to 10 by 0.1 of interval
 [1]  1.0  1.1  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9  2.0  2.1  2.2  2.3  2.4  2.5  2.6  2.7
[19]  2.8  2.9  3.0  3.1  3.2  3.3  3.4  3.5  3.6  3.7  3.8  3.9  4.0  4.1  4.2  4.3  4.4  4.5
[37]  4.6  4.7  4.8  4.9  5.0  5.1  5.2  5.3  5.4  5.5  5.6  5.7  5.8  5.9  6.0  6.1  6.2  6.3
[55]  6.4  6.5  6.6  6.7  6.8  6.9  7.0  7.1  7.2  7.3  7.4  7.5  7.6  7.7  7.8  7.9  8.0  8.1
[73]  8.2  8.3  8.4  8.5  8.6  8.7  8.8  8.9  9.0  9.1  9.2  9.3  9.4  9.5  9.6  9.7  9.8  9.9
[91] 10.0
```

b) Sample from this vector 4 numbers:

```
> sample(seq(1,10,0.1),4) #Sample from the seq(1,10,0.1) 4 numbers
[1] 1.1 3.6 5.4 1.4
```

R Data Types

Matrices

- You can create matrices using the matrix( ) function.
- Matrices are two-dimensional structures consisting of **rows and columns** of elements

```
matrix(data, nrow, ncol, byrow = FALSE)
```

R Data Types

Matrices

- The basic syntax for creating a matrix in R is

```
matrix(data, nrow, ncol, byrow = FALSE)
```

  - *data* is the input vector which becomes the data elements of the matrix
  - *nrow* is the number of rows to be created
  - *ncol* is the number of columns to be created
  - *byrow* is a logical clue. If TRUE then the input vector elements are arranged by row

R Data Types

Matrices

- Create a 2x3 matrix from a vector

```
x<- c(1, 2, 3, 4, 5, 6)
y<- matrix(x, nrow = 2, ncol = 3)
```

```
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

R Data Types

Matrices

```
> matrix(0, 2, 3)
     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    0
>
> (vals <- 1:6)
[1] 1 2 3 4 5 6
>
> (x <- matrix(vals, 2, 3))
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
>
> dim(x)
[1] 2 3
```

or

```
> (vals <- 1:6)
[1] 1 2 3 4 5 6
>
> (y <- matrix(vals, 2, 3, byrow = TRUE))
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
>
> dim(y)
[1] 2 3
```

R Data Types

Matrices

- Elements are arranged sequentially by row

```
> M <- matrix(c(3:14), nrow = 4, ncol = 3, byrow = TRUE)
> M
     [,1] [,2] [,3]
[1,]    3    4    5
[2,]    6    7    8
[3,]    9   10   11
[4,]   12   13   14
```

- Elements are arranged sequentially by column

```
> N <- matrix(c(3:14), nrow = 4, ncol = 3, byrow = FALSE)
> N
     [,1] [,2] [,3]
[1,]    3    7   11
[2,]    4    8   12
[3,]    5    9   13
[4,]    6   10   14
```

R Data Types

Matrices

- Define the column and row names

```
> rownames = c("row1", "row2", "row3", "row4")
> colnames = c("col1", "col2", "col3")
>
> x <- matrix(c(3:14), nrow = 4, ncol = 3, byrow = TRUE, dimnames = list(rownames, colnames))
> x
     col1 col2 col3
row1    3    4    5
row2    6    7    8
row3    9   10   11
row4   12   13   14
```

R Data Types

Matrices

- Accessing Elements of a Matrix

- Access the element at 3rd column and 1st row

```
> x[1,3]
[1] 5
```

- Access only the 2nd row

```
> x[2, ]
col1 col2 col3
   6    7    8
```

- Access only the 3rd column

```
> x[,3]
row1 row2 row3 row4
   5    8   11   14
```

R Data Types

Mathematical Operations with Matrices

- You can perform various mathematical operations with matrices

R Data Types

Mathematical Operations with Matrices

- Create two 2x3 matrices

```
> matrix1 <- matrix(c(3, 9, -1, 4, 2, 6), nrow = 2)
> matrix1
     [,1] [,2] [,3]
[1,]    3   -1    2
[2,]    9    4    6
>
>
> matrix2 <- matrix(c(5, 2, 0, 9, 3, 4), nrow = 2)
> matrix2
     [,1] [,2] [,3]
[1,]    5    0    3
[2,]    2    9    4
```

- Add the matrices

```
> M <- matrix1 + matrix2
> M
     [,1] [,2] [,3]
[1,]    8   -1    5
[2,]   11   13   10
```

- Subtract the matrices

```
> M <- matrix1 - matrix2
> M
     [,1] [,2] [,3]
[1,]   -2   -1   -1
[2,]    7   -5    2
```

R Data Types

Mathematical Operations with Matrices

- Create two 2x3 matrices

```
> matrix1 <- matrix(c(3, 9, -1, 4, 2, 6), nrow = 2)
> matrix1
     [,1] [,2] [,3]
[1,]    3   -1    2
[2,]    9    4    6
>
>
> matrix2 <- matrix(c(5, 2, 0, 9, 3, 4), nrow = 2)
> matrix2
     [,1] [,2] [,3]
[1,]    5    0    3
[2,]    2    9    4
```

Multiply the matrices

```
> M <- matrix1 * matrix2
> M
     [,1] [,2] [,3]
[1,]   15    0    6
[2,]   18   36   24
```

Divide the matrices

```
> M <- matrix1 / matrix2
> M
     [,1]      [,2]      [,3]
[1,]  0.6      -Inf 0.6666667
[2,]  4.5 0.4444444 1.5000000
```

R Data Types

Matrix Algebra

- Transposed Matrix

```
y <- t(matrix1)
```

R Data Types

Matrix Algebra

- Matrix Determinant

```
y <- det(matrix1)
```

R Data Types

Matrix Algebra

- Inverse of matrix

```
x<- solve(matrix1)
```

Exercises

- Let´s Practice!

- Let $\boldsymbol{M} = \begin{bmatrix} 5 & 9 \\ -4 & 1 \\ 8 & 0 \end{bmatrix}$ and $\boldsymbol{N} = \begin{bmatrix} -4 & 1 \\ -2 & 0 \\ 0 & 5 \end{bmatrix}$

Next, do the following:

a) Create matrices M and N in R
b) Get the transpose of M and N
c) Add M and N
d) Subtract N from M

Exercises

a) Create matrices M and N in R

```
> v1 <- c(5,9)
> v2 <- c(-4,1)
> v3 <- c(8,0)
>
> M <- rbind(v1, v2, v3)
> M
   [,1] [,2]
v1    5    9
v2   -4    1
v3    8    0
```

```
> v1 <- c(-4,1)
> v2 <- c(-2,0)
> v3 <- c(0,5)
> N <- rbind(v1, v2, v3)
> N
   [,1] [,2]
v1   -4    1
v2   -2    0
v3    0    5
```

#rbind: Combine vectors as rows in a matrix

#cbind: Combine vectors as columns in a matrix

Exercises

b) Get the transpose of M and N

```
> t(M)
     v1 v2 v3
[1,]  5 -4  8
[2,]  9  1  0
```

```
> t(N)
     v1 v2 v3
[1,] -4 -2  0
[2,]  1  0  5
```

Exercises

c) Add M and N:

```
> result <- M+N
> result
   [,1] [,2]
v1    1   10
v2   -6    1
v3    8    5
```

d) Subtract N from M:

```
> result1 <- N-M
> result1
   [,1] [,2]
v1   -9   -8
v2    2   -1
v3   -8    5
```

R Data Types

Data frame

- We can combine vectors of same lenght in a data frame, to combine x, y, and z vectors
- To tabular data, where each column can contain data of different types
- Data frames are one of the most common data structures used for data manipulation and analysis in R

df<- data.frame(coluna1, coluna2, ...)

R Data Types

Data frame

- Criar um data frame com três colunas

```
name<- c("Ana", "João", "Maria")
age <- c(25, 30, 28)
score <- c(85, 90, 88)

df<- data.frame(name, age, score)
```

	name	age	score
1	Ana	25	85
2	João	30	90
3	Maria	28	88

R Data Types

Data frame

```
dados <- data.frame(
  Ind = c(1:10),
  Yield = c(1.80, 1.77, 1.71, 1.65, 1.66, 1.63, 1.77, 2.58, 2.15, 1.56),
  Resis = c(TRUE, FALSE, FALSE, FALSE, TRUE, FALSE, TRUE, TRUE, FALSE, FALSE)
)
```

	Ind	Yield	Resis
1	1	1.80	TRUE
2	2	1.77	FALSE
3	3	1.71	FALSE
4	4	1.65	FALSE
5	5	1.66	TRUE
6	6	1.63	FALSE
7	7	1.77	TRUE
8	8	2.58	TRUE
9	9	2.15	FALSE
10	10	1.56	FALSE

Graphic Plotting

- There are several types of graphics in R
- I will present some of the most used ones

Graphic Plotting

Scatter plots

- A scatter plot is a type of graph used in statistics and data visualization to display the relationship between two quantitative variables

- The simple scatter plot is created using the plot( ) function

Graphic Plotting

Scatter plots

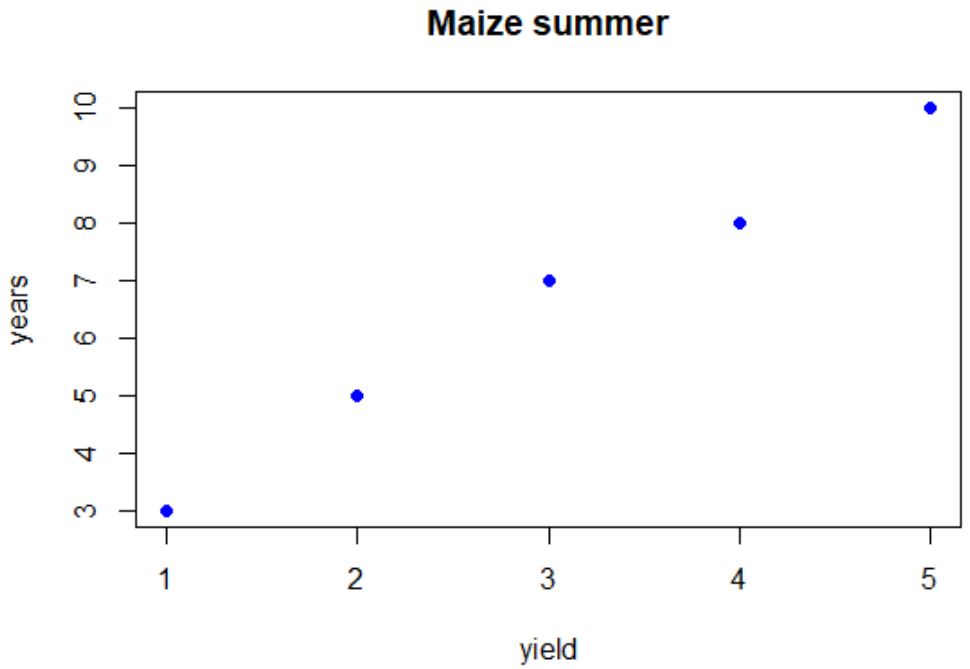
plot(x, y, main, xlab, ylab, col, pch)

- *x, y:* the data vectors for the independent (x-axis) and dependent (y-axis) variables
- *main:* title of the graph
- *xlab:* the label (name) in the horizontal axis
- *ylab:* the label (name) in the vertical axis
- *col:* the color of the points on the scatter plot
- *pch:* the type of point used to represent each point

Graphic Plotting

Scatter plots

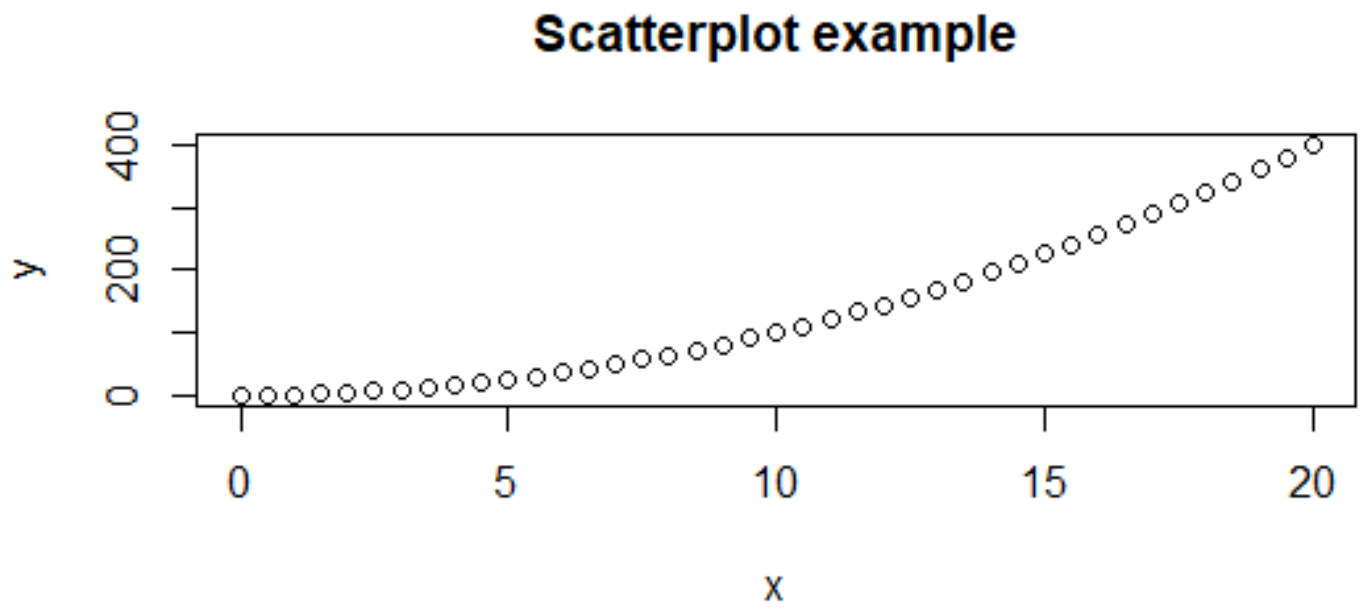
```
x <- c(1, 2, 3, 4, 5)
y <- c(3, 5, 7, 8, 10)

# Create the scatter plot
plot(x, y, main = “Maize summer", xlab = “yield", ylab = “years”, col = "blue", pch = 19)
```



Graphic Plotting

Scatter plots

```
x <- seq(0, 20, 0.5)
y <- x^2
plot(x, y, main = "Scatterplot example")
```



Graphic Plotting

Histograms

- A histogram is a graphical representation of the distribution of a numerical variable
- R creates histogram using hist( ) function

Graphic Plotting

Histograms

hist(data_vector, breaks, main, xlab, ylab, col)

data_vector: the data vector you want to represent in the histogram

breaks: defines how intervals should be defined

main: the title of the histogram

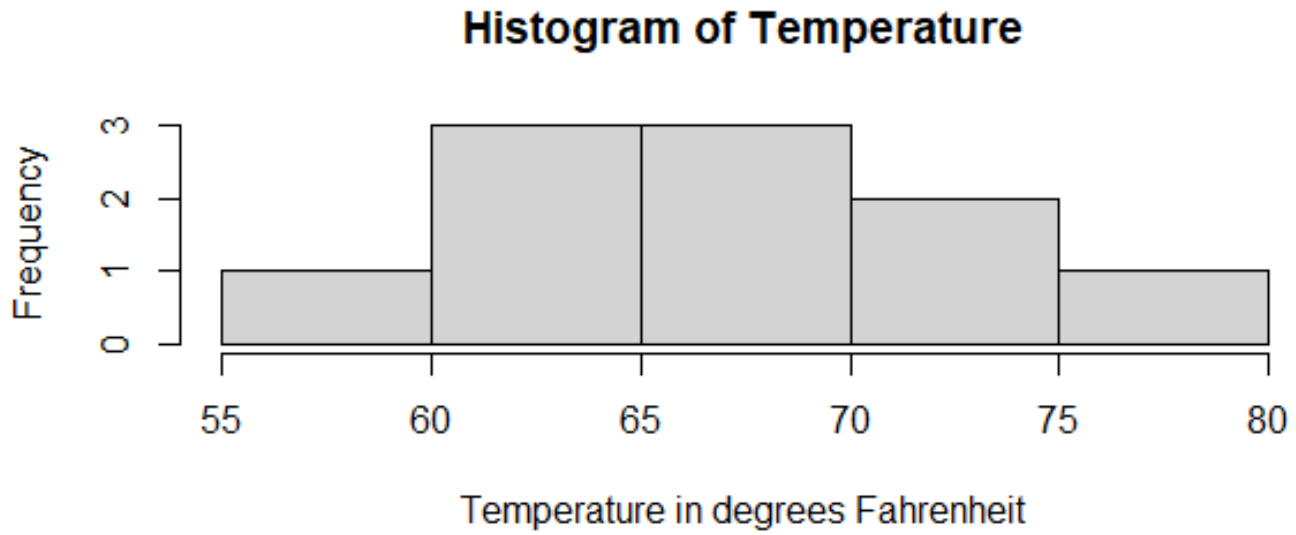
xlab and ylab: the x and y axis labels

col: the color of the histogram bars (by default they are black).

Graphic Plotting

Histograms

```
temperatures <- c(67 ,72 ,74 ,62 ,76 ,66 ,65 ,59 ,61 ,69 )

# histogram of temperatures vector
result <- hist(temperatures,
               main = "Histogram of Temperature",
               xlab = "Temperature in degrees Fahrenheit")
```



Graphic Plotting

Boxplots

- The boxplot, also known as a box plot, is a graphical tool used to visualize the distribution of a data set and identify the presence of outliers

- R creates boxplots using boxplot( ) function

Graphic Plotting

Boxplots

boxplot(data_vector, main, xlab, ylab, col)

data_vector: the data vector you want to represent in the boxplot

main: the title of the boxplot

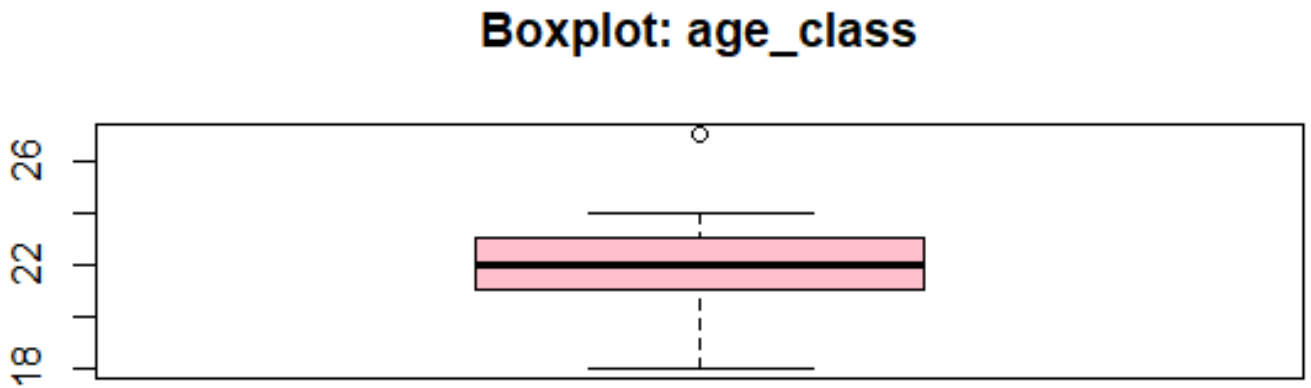
xlab and ylab: the x and y axis labels

col: the color of the boxplot boxes

Graphic Plotting

Boxplots

```
> age_class <- c(21,22,24,18,19,27,22,22,23,21,23,21)
> summary(age_class)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  18.00   21.00   22.00   21.92   23.00   27.00
>
> boxplot(age_class, main="Boxplot: age_class", col="Pink")
```



Graphic Plotting

Barplot

- Barplot is a graphical representation used to display categorical data or counts in specific categories

- R creates bar chats using barplot( ) function

Graphic Plotting

Barplot

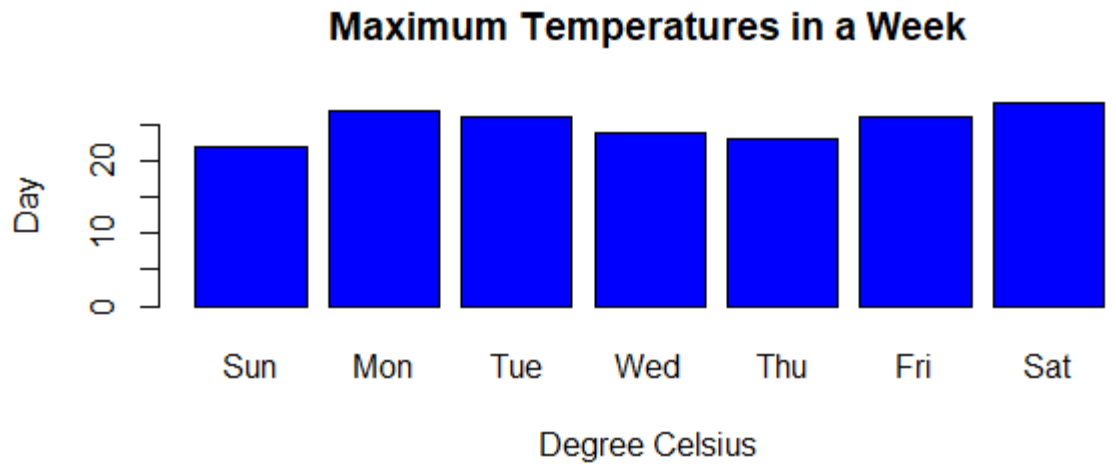
barplot(H, main, xlab, ylab, names.arg, col)

- *H* is a vector or matrix containing numeric values used in bar chart
- *main* is the title of the bar plot
- *xlab* is the label for x axis
- *ylab* is the label for y axis
- *names.arg* is a vector of names appearing under each bar;
- *col* is used to give colors to the bars in the graph

Graphic Plotting

Barplot

```
max.temp <- c(22, 27, 26, 24, 23, 26, 28)

barplot(max.temp,
        main = "Maximum Temperatures in a Week",
        xlab = "Degree Celsius",
        ylab = "Day",
        names.arg = c("Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"),
        col = "Blue")
```



Graphic Plotting

HeatMaps

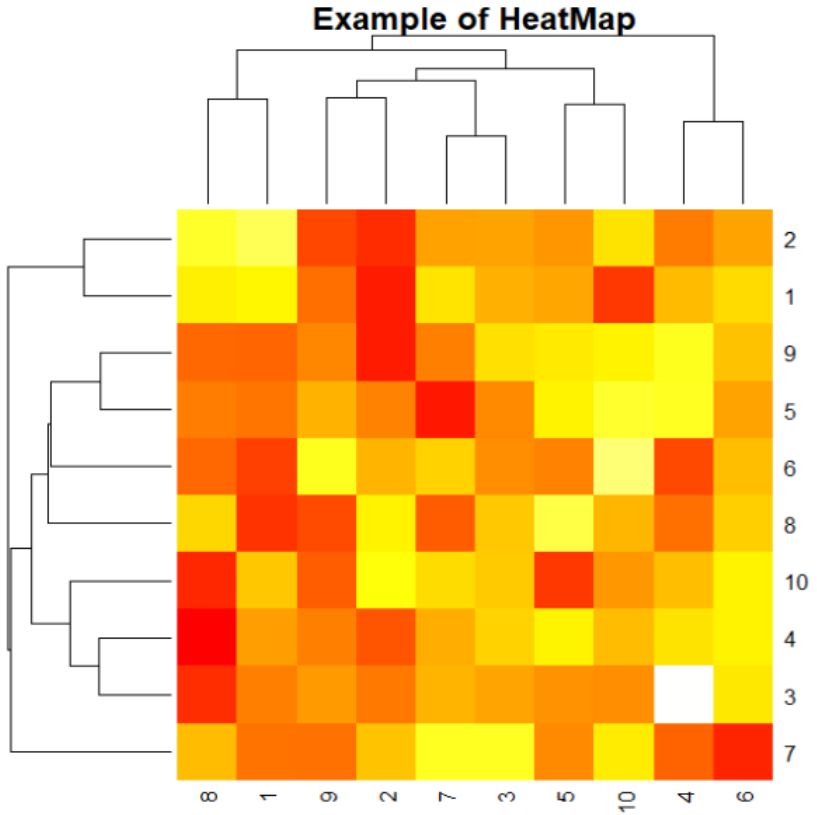
- Heat maps are an effective way to visualize numerical data in a matrix where colors represent values

- R creates using heatmap( ) function

Graphic Plotting

HeatMaps

```
#Create an example matrix
matriz <- matrix(rnorm(100), nrow = 10)

#Create a matrix heatmap
heatmap(matriz, col = heat.colors(256), main = "Example of HeatMap")
```



Working Directory

- The working directory is the default location where R will look for files you want to load and where it will put any files you save.

- The getwd( ) function is used to get the current working directory of the environment

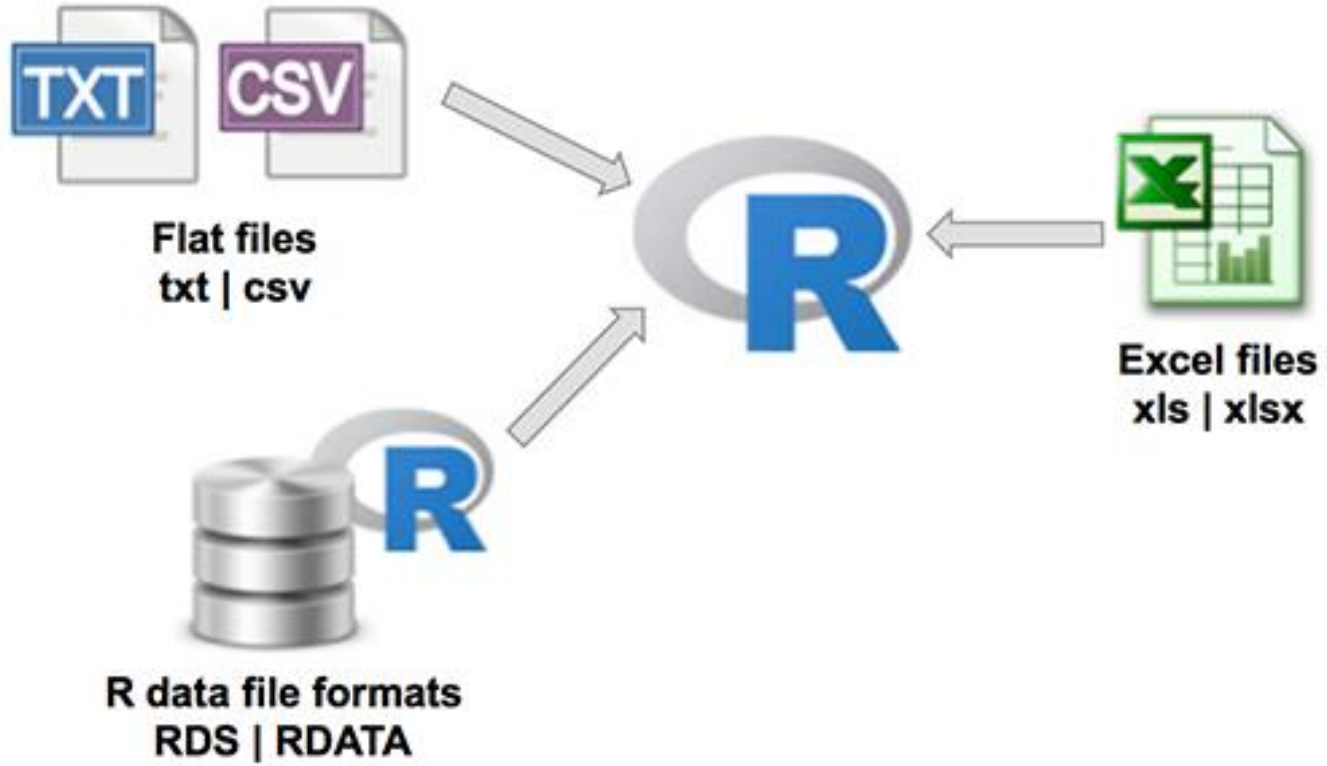
```
Working Directory
current_directory <- getwd()
```

Working Directory

- The setwd( ) function in R is used to set the current Working Directory to the specified directory

```
setwd(dir)
```

Importing data into R



File Formatting

- R is capable of reading text files in a variety of formats

- To import data into R, you can use several functions, depending on the file format you want to import

```
data <- read.csv("data.csv")
head(data)
```

Importing data into R

- To read a table in R, you can use the read.table( ) function
- read.table( ) function reads a file into data frame in table format

Importing data into R

read.table(file, header = FALSE, sep = "", quote = "\"'", dec = ".", ...)

- *file:* the name of the text file to be read
- *header:* a logical value indicating whether the file has a **header** row
- *sep:* the separator used to delimit the fields in the file
- *quote:* the character used to quote fields containing the separator
- *dec:* the character used to represent the decimal point in numbers

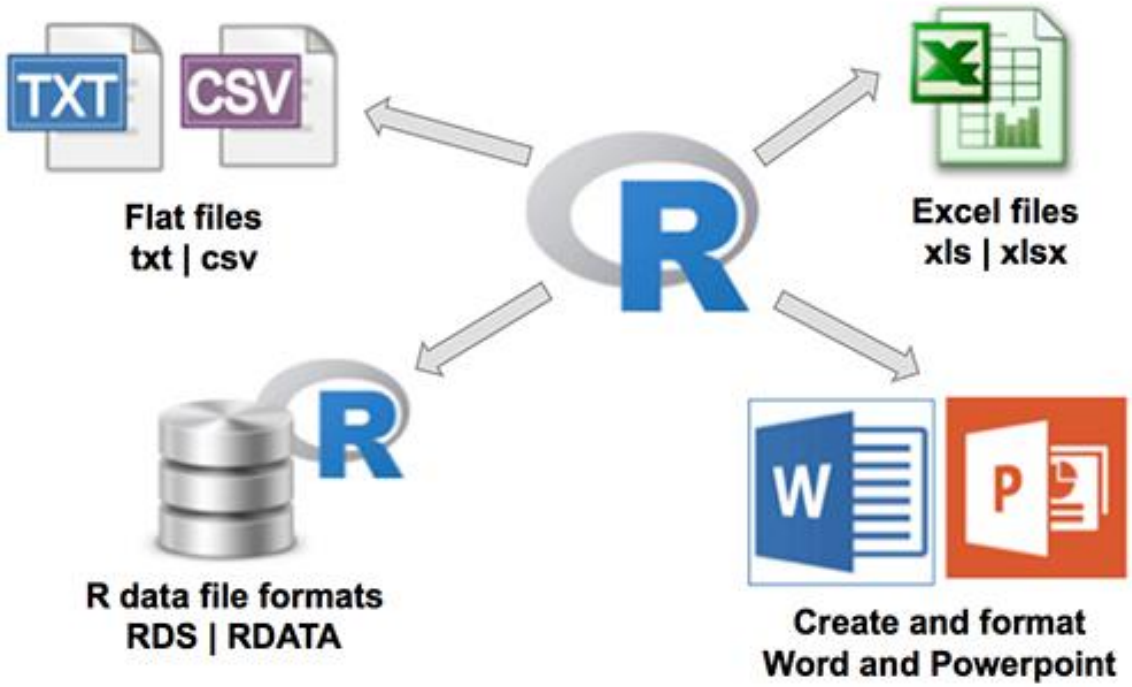
Importing data into R

```
data <- read.table("data.txt", header = TRUE, sep = "\t")

#Shows the first rows of data
head(data)
```

Exporting data from R

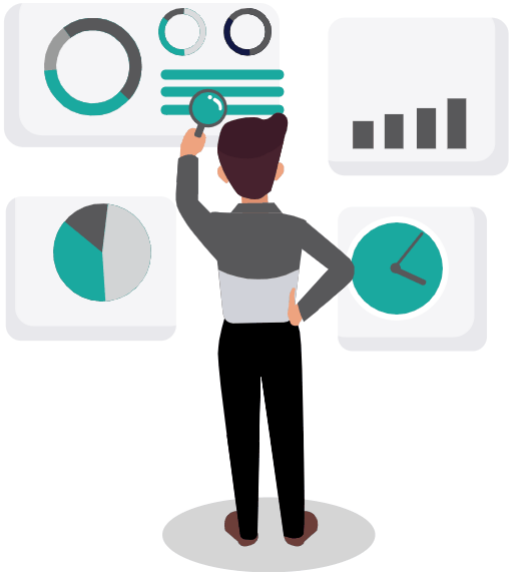
- You can use the write.table( ) function in R to export a data frame

Exporting data from R



Introduction to R

Review your data well before starting your analysis!



Introduction to R

References

- 1. Venables, W., Smith, D. & the R Core Team. **An Introduction to R - Notes on R: A Programming Environment for Data Analysis and Graphics**. (2017)
- 2. Verzani, John. "**simpleR–Using R for Introductory Statistics**." New York: CUNY, 0.4 edition URL http://www. math. csi. cuny. edu/Statistics/R/simpleR/index. html 106 (2002)
- Coursera: https://www.coursera.org/learn/r-programming
- Datacamp: https://www.datacamp.com/courses/free-introduction-to-r
- TryR: https://www.pluralsight.com/search?q=R
- Manuals and tutorials:
  R official documents – particularly the Introduction to R[1]
  R contributed documentation