Lecture #2

The R language and environment

Outline

- What is R?
- Basic syntax rules
- Math operators
- Vectors, Matrices, and Data frames
- Logical operators
- Casting data types
- Read/Write
- Iteration
- Packages
- Libraries/Functions
- Plotting commands
- Par command
- Data sets in R

What is R?

- R is both a language and environment for statistical and computational analysis

- Where does the name come from?
  - Authors: Robert Gentleman and Ross Ihaka and S language play-on words

- The language is similar to that of Splus, which gets its roots from S
  - Developed by Bell Labs

- R was developed as a collaborative project from multiple contributors (CRAN)
  - You can readily download and contribute “packages” to R

- R software is available free, however there is no support
  - Splus (on the other hand) is not free, so support is provided

Basic syntax rules

- Procedures in R are implemented as functions
  - “()” syntax
  - i.e. c(4,5,6) means to concatenate the three numbers into a vector
- R processes are best handled as objects
  - If you run any function, assign it to a variable (object) so you can manipulate the attributes in the object
- Numerical objects are interpreted best in formats of matrices, vectors, data frames, or lists
  - Statistical language vs. computer language
- **!To get help with any function in R, use the help function!**
  - **help(mean)**
  - **?mean**
- To quit out of R, use the quit function
  - q()
- Variable assignment is “=“ or “<-”

Basic syntax rules (cont.)

- Annotation syntax is “#” (R will not read comments that start with this symbol)
- To view the objects in the database
  - ls()
- Remove an object in the database
  - rm()
- Concatenate/combine multiple numbers/objects together
  - c()
- Underscores (“_”) are not handled well in variable names
  - The underscore is synonymous with the “=“ and “<-” assignment operators
  - Try to avoid underscores in scripts
- Calculate a mean and variance
  - mean()
  - var()
- View current memory size and set memory limits
  - memory.size(T)
  - memory.limit

Mathematical Calculations

- General math operators
  - > 5 + 5
  - > 10 – 2
  - > 10 * 10
  - > 25 / 5
  - > 3 ^ 2
  - > exp(2)
  - > log(10)
  - > logb(10,2)

Vectors

- Vector syntax
  - Create a vector
    - > x <- c(2,4,6,3,4,6)
    - > y <- c(5,6,7,8,8,0)
  - Specify certain elements of the vector
    - > x[1:3]

      [1]   2 4 6
  - Remove an element from the vector
    - > x[-2]

      [1]   2 6 3 4 6
  - Bind 2 vectors together (must be same length)
    - > x.y.bound <- cbind(x,y)   # cbind means column bind

Misc. vector operations

- Intersect elements of 2 vectors

  -
    ```
    > intersect(x,y)
    [1]  6
    ```
- Diff 2 vectors

  -
    ```
    > setdiff(x,y)        # what is in x that is not in y?
    [1]  2  4  3
    ```
- Find length of x vector

  -
    ```
    > length(x)
    [1]  6
    ```

Matrices

- Matrix syntax
  - Create a matrix
    - > x.matrix <- matrix(data=x,nrow=3,ncol=2)
    - > x.matrix

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

  - Look at matrix dimensions
    - > dim(x.matrix)

```
[1]  3  2
```

  - Specify elements in the matrix
    - > x.matrix[1:2,2]   # rows 1 and 2 of column 2

```
[1]  3  4
```

    - > x.matrix[1,]   # rows 1, all columns

```
[1]  2  3
```

  - Transpose the matrix
    - > x.matrix.t <- t(x.matrix)
    - > x.matrix.t

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

Data frames

- Similar syntax to matrices, but have row and column names
  - Create a data frame
    - > x.df <- data.frame(x,y)
    - > x.df

```
  x y
1 2 5
2 4 6
3 6 7
4 3 8
5 4 8
6 6 0                    # has row names (1-6) and column names (x and y)
```

  - Look at row names
    - > dimnames(x.df)[[1]]

      [1] "1" "2" "3" "4" "5" "6"
  - Look at column names
    - > dimnames(x.df)[[2]]

      [1] “x” “y”

Logical Operators

- R works very well with boolean logic
  - Look for values greater than 4 in x.matrix
    - > x.matrix.g4 <- x.matrix>4
    - > x.matrix.g4

```
      [,1]  [,2]
[1,] FALSE FALSE
[2,] FALSE FALSE
[3,]  TRUE  TRUE
```

  - Print only those values out
    - > x.matrix[x.matrix.g4]

```
[1]   6  6
```

Casting

- Casting is basically changing the data-type from one type to another
  - i.e. data is in matrix format and you wish to convert it to a vector:
  - use the “as.” syntax followed by the desired data-type

```
> x.vector <- as.vector(x.matrix)          # change to vector
> x.vector
[1] 2 4 6 3 4 6
> x.df <- as.data.frame(x.matrix)      # change to dataframe
> x.df
   V1 V2
1   2  3
2   4  4
3   6  6
> x.char <- as.character(x.vector)         # change to character
> x.char
[1] "2" "4" "6" "3" "4" "6"
```

- Can also check type of data object with “mode” and “class” functions

Read in/Write out

- Multiple ways to read a data file in
  - > read.table(file="C:\\Class\\data.txt",header=T)
    - must use "\\" instead of "\" for path
  - > scan(file="C:\\Class\\data.txt")
    - must use "\\" instead of "\" for path
    - best for vectors or lists (not best for 2D data)

- Writing data out to a file
  - > write.table(x.matrix,file="dataFile.txt",sep="\t")
  - > write(x.matrix,file="dataFile.txt",ncolumns=2)

Iteration & If Statements

- Loops are not optimized in R
  - They are possible, but not memory favorable

- Loop syntax
  - > for(i in 1:6) {}   # for loop
  - > while(i < 7) {}   # while loop

- apply statements
  - Inherent looping mechanism
  - Much more memory efficient
  - > apply(x.matrix,1,mean)
    - Will calculate a mean on each row in x.matrix
  - > apply(x.matrix,2,mean)
    - Will calculate a mean on each column in x.matrix

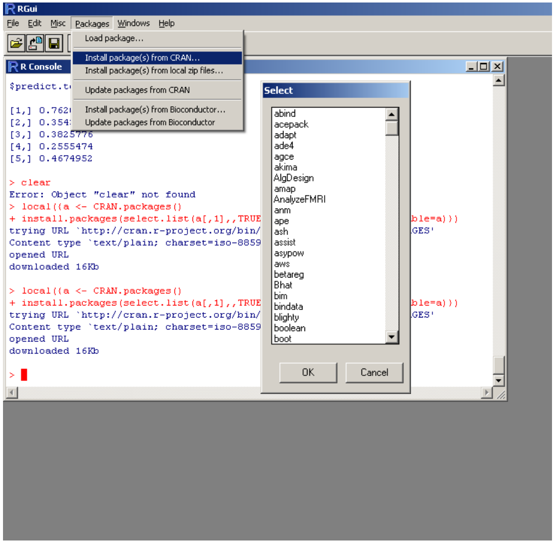
- if statements are fairly well implemented
  - if(x[1]>5) {x[1]=6}   # simple if command
  - ifelse(x[1]>5,x[1]=6,x[1]=9)   # combine if and else into one statement

Packages

- Binary files that build library access structure
  - Allow specific functions to be accessible
  - If you know the package name, you can simply type: `install.packages("tree")`
    - `“tree”` is the example package in this case

Can install packages from CRAN or Bioconductor

Can install packages from local drive (if saved zip file is on local drive)

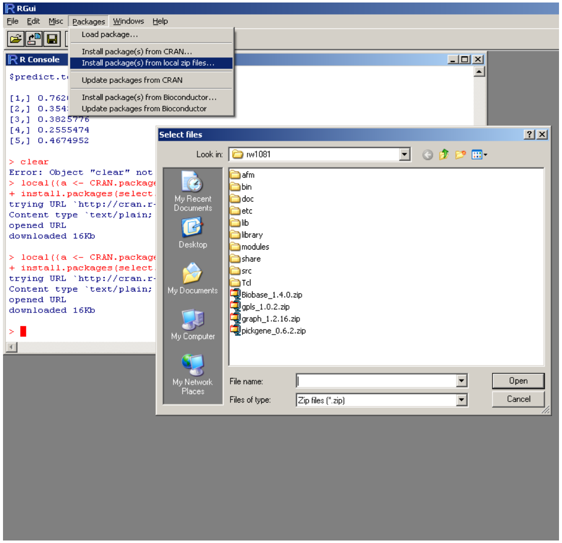

RGui
File Edit Misc Packages Windows Help
Load package...
Install package(s) from CRAN...
Install package(s) from local zip files...
Update packages from CRAN
Install package(s) from Bioconductor...
Update packages from Bioconductor
R Console
Select
abind
acepack
adapt
ade4
agce
akima
AlgDesign
amap
AnalyzeFMRI
anm
ape
ash
assist
asypow
aws
betareg
Bhat
bim
bindata
blighty
boolean
boot
OK
Cancel

```
$predict.t
[1,] 0.762
[2,] 0.354
[3,] 0.3825776
[4,] 0.2555474
[5,] 0.4674952

> clear
Error: Object "clear" not found
> local({a <- CRAN.packages()
+ install.packages(select.list(a[,1],,TRUE                 ble=a))
trying URL `http://cran.r-project.org/bin/                GES'
Content type `text/plain; charset=iso-8859
opened URL
downloaded 16Kb

> local({a <- CRAN.packages()
+ install.packages(select.list(a[,1],,TRUE                 ble=a))
trying URL `http://cran.r-project.org/bin/                GES'
Content type `text/plain; charset=iso-8859
opened URL
downloaded 16Kb

>
```



```
$predict.t
[1,] 0.762
[2,] 0.354
[3,] 0.3825776
[4,] 0.2555474
[5,] 0.4674952

> clear
Error: Object "clear" not
> local({a <- CRAN.package
+ install.packages(select
trying URL `http://cran.r-
Content type `text/plain;
opened URL
downloaded 16Kb

> local({a <- CRAN.package
+ install.packages(select
trying URL `http://cran.r-
Content type `text/plain;
opened URL
downloaded 16Kb

>
```

Libraries & Functions

- Once package has been installed, library can be accessed

- Libraries include many functions
  - > library(base)   # implements base library
  - Allows access to functions inside the library

- Data sets internal to R can also be accessed and utilized
  - > data(iris)   # allows access to famous iris flower data set
  - MORE EXPANATION ON THIS FURTHER IN LECTURE

- Functions
  - Handled well in R
  - Similar to a subroutine in other languages (packaged operation that is called)
  -
```
> square <- function(x=num1,y=num2) {
                        r1 <- x^2
                        r2 <- y^2
                        output <- c(r1,r2)
                        return(output)
              }
```
  -
```
> square(x=4,y=2)                    # call square function
   [1] 16  4
```

Packages & Libraries

- You can view the complete list of installed packages
  - > .packages(all = TRUE)
- Also view the list of current attached packages
  - > (.packages())
- Can set the path for R to look for the libraries
  - > .libPaths("C:/PROGRA~1/R/RW2000~1/library")
- Remove package from session
  - > detach(“package:stats”)
- View package contents (e.g. functions, author, date, version, description, etc.)
  - > library(help=odesolve)

Plotting commands

- There are many optional commands available in R for plotting data
  - Only a few of the main commands are necessary

- Scatter plot
  - > plot(x,y)   # x and y must be same length and numeric
  - Labels
    - xlab='x vector'# x-axis label
    - ylab='y vector'# y-axis label
    - main='Scatter plot'   # title for plot
  - Points, size, and colors
    - pch='*****'# symbol to plot
    - cex=1.5   # size of symbol
    - col='red'   # color of points

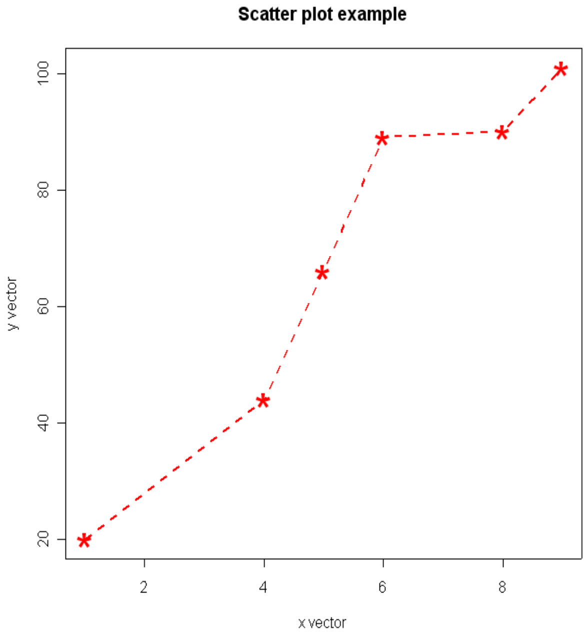
Plotting commands (steps)

- Plotting function can be given in subsequent commands for modifications to the graph

- Scatter plot options
  - Plot without points/lines
    - type='n'   # no points or lines are not drawn
    - type='b'   # lines and points drawn together
    - type='p'   # only points are draw
    - type='l'   # only lines are drawn
  - Points only
    - points(x,y,col='blue')
  - Lines only
    - lines(x,y,col='yellow')
    - lwd(2)   # line width
    - lty(3)   # line pattern

Plot example

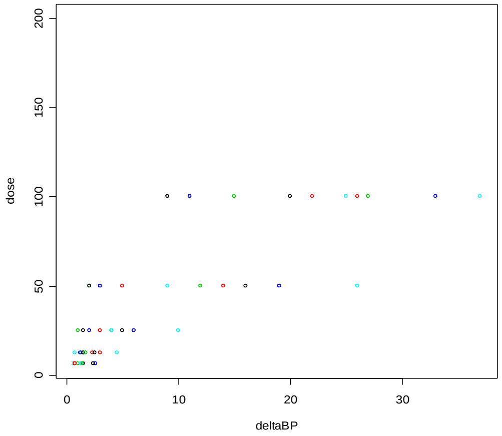
```
> x <- c(1,4,5,6,8,9)
> y <- c(20,44,66,89,90,101)
> plot(x,y,type='n',xlab='x vector',ylab='y vector',main='Scatter plot example')
> points(x,y,col='red',pch='*',cex=3)
> lines(x,y,col='red',lwd=2,lty=2)
```

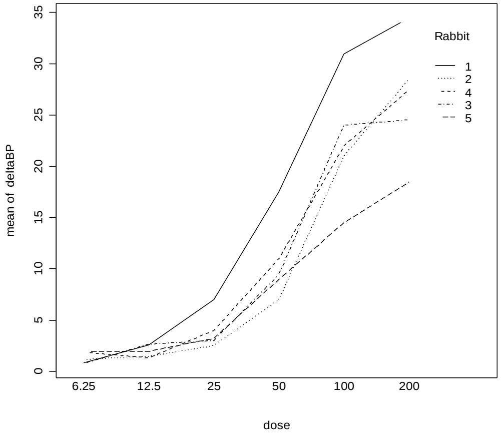


Plotting commands (options)

- Data can be plot using grouping variable commands

```
> library(nlme)
> data(PBG)
> attach(PBG)
> plot(deltaBP dose)
> plot(deltaBP dose, col=as.integer(Rabbit))
> interaction.plot(dose, Rabbit, deltaBP)
```





Par commands

- Page formatting to add more than 1 plot per page
  - `> par(mfrow=c(2,3))        # 6 total plots (2 rows & 3 columns)`
- Page formatting to add margin sizes
  - `> par(oma=c(2,4,2,4))    #order is bottom, left, top, right`
- Multiple other page format options
  - See help section for “par”

Data sets in R

Using packages in Bioconductor, multiple microarray data sets are available

```
library(Biobase);
library(annotate);
library(golubEsets);
library(multtest);
```

<u>Golub et al. AML/ALL Affy data set (training set and test set are separate)</u>

```
data(golubTrain);
data(golubTest)
dat.train <- exprs(golubTrain)                   (7129 genes x 38 samples)
data(golubTest)
dat.test <- exprs(golubTest)                     (7129 genes x 34 samples)
clas <- pData(phenoData(golubTrain))             (class labels for training set samples)
```

or

```
data(golub)
smallgd<-golub                   (7129 genes x 38 samples)
classlabel<-golub.cl             # class labels for training set samples
```

<u>Alon et al. colon cancer Affy data set</u>

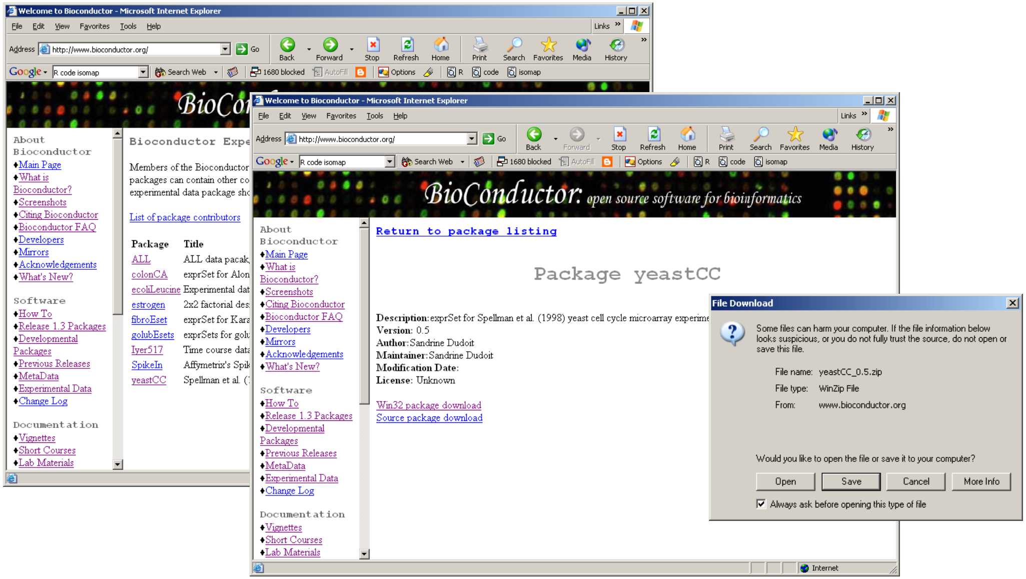
```
library(colonCA)
data(colonCA)
dat <- exprs(colonCA)            (2000 genes x 62 samples)
classlabel <- colonCA$class      # class labels for samples
```

<u>Some unidentified Affy data set</u>

```
data(geneData)
dat <- geneData         (500 genes x 26 samples)
```

Getting data sets from CRAN

Bioconductor (http://www.bioconductor.org/)
Under <Software>
<Experimental Data>
yeastCC: **Spellman et al. yeast cell cycle data - ** cDNA data
golubEsets: **Golub et al. AML/ALL data - ** Affymetrix data
colonCA: **Alon et al. Colon Cancer data** – Affymetrix data


Welcome to Bioconductor - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links
Address http://www.bioconductor.org/
Go
Back Forward Stop Refresh Home Print Search Favorites Media History
Google R code isomap Search Web 1680 blocked AutoFill Options R code isomap
BioConductor: open source software for bioinformatics
Internet

About Bioconductor
♦Main Page
♦What is Bioconductor?
♦Screenshots
♦Citing Bioconductor
♦Bioconductor FAQ
♦Developers
♦Mirrors
♦Acknowledgements
♦What's New?

Software
♦How To
♦Release 1.3 Packages
♦Developmental Packages
♦Previous Releases
♦MetaData
♦Experimental Data
♦Change Log

Documentation
♦Vignettes
♦Short Courses
♦Lab Materials

Bioconductor Expe

Members of the Bioconductor
packages can contain other co
experimental data package sho

List of package contributors

Package	Title
ALL	ALL data pacak
colonCA	exprSet for Alon
ecoliLeucine	Experimental dat
estrogen	2x2 factorial des
fibroEset	exprSet for Kara
golubEsets	exprSets for golu
Iyer517	Time course data
SpikeIn	Affymetrix's Spik
yeastCC	Spellman et al. (

Return to package listing

Package yeastCC

Description: exprSet for Spellman et al. (1998) yeast cell cycle microarray experime
Version: 0.5
Author: Sandrine Dudoit
Maintainer: Sandrine Dudoit
Modification Date:
License: Unknown

Win32 package download
Source package download

File Download

Some files can harm your computer. If the file information below looks suspicious, or you do not fully trust the source, do not open or save this file.

File name: yeastCC_0.5.zip
File type: WinZip File
From: www.bioconductor.org

Would you like to open the file or save it to your computer?

☑ Always ask before opening this type of file

Getting data sets from CRAN (cont.)

Once the zip file has been downloaded to a local directory, go into R and use package instructions to install zip file from local directory

Then,

access data from data object as follows:

```
# load package from local drive
> library(Biobase)
> library(annotate)
> library(colonCA)
> data(colonCA)
> dat <- exprs(colonCA)              # expression data
> ann.dat <- colonCA$class           # annotations for samples
```

References

- R Primer
  - http://www.r-project.org/

    <Documentation>

        <Manuals>

            <An Introduction to R> or

            <The R Reference Index>

- Introductory Statistics with R, Peter Dalgaard