DISCOVERING STATISTICS USING R

The R-environment

1

DISCOVERING STATISTICS USING R

Aims and Objectives

- Get R and R Studio to work on your computer
- Have a basic understanding of the R-chitecture
- Know how to set the working directory, install and load packages, enter and import data
- Work with, and write your own scripts

2

DISCOVERING STATISTICS USING R

Why use R?

- Open-source, across-platform and free
- Powerful and versatile analytic/graphing abilities
- Large user community
  - Lots of online support
  - Steady flow of innovation
  - Increasingly the analytical software of choice in biology

3

DISCOVERING STATISTICS USING R

The main problems with R

- Steep learning curve
- Many different ways in which to achieve the same goal
- Biologists are often used to working with a Graphical User Interface (GUI) instead of a command line prompt

4

DISCOVERING STATISTICS USING R

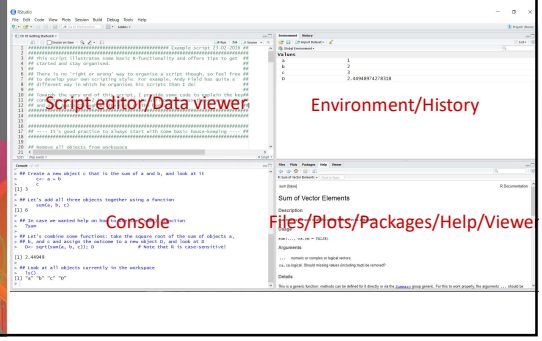
Downloading and installing R

- R
  - Google for 'CRAN'
    - The Comprehensive R Archive Network
    - Repository from where to download R and 'packages'
  - Select your operating system
  - Most recent version currently seems to be: R 3.5.3
- R Studio (highly recommended)
  - Offers *a.o.* a more intuitive and user-friendly interface

5

DISCOVERING STATISTICS USING R

Main windows in R Studio



6

Main windows in R Studio

- Four windows, many panes
  - Script editor/Data viewer
    - Window in which to open/write scripts (sets of commands)
    - Possible to have multiple open in same session
  - Console
    - Window to directly execute commands and see output
  - Environment/History
    - Lists all objects in workspace
    - Lists previously executed commands
  - Files/Plots/Packages/Help/Viewer
    - You will probably mainly use the Files, Plots and Help panes

7

R: The basics

- A list of basic functions available by default, can be found at: http://cran.r-project.org/doc/contrib/Short-refcard.pdf
- Other useful resources are:
  - The documentation within R itself
  - Many books…
  - … and websites

The best way to learn R, however, is to do it yourself!

8

R: The basics

Directly in Console

- An overgrown calculator

```
> 1 + 2
[1] 3
```

- Assign values to symbolic variables (**objects**)

```
> a<- 1
> b<- 2
> c<- a + b
> c
[1] 3
```

- As we will see objects can be far more complex than just numbers

9

R: The basics

- Objects are manipulated with **functions**…

```
> sum(a, b, c)
[1] 6
```

- … which can be combined

```
> D<- sqrt(sum(a, b, c))
> D
[1] 2.44949
```

10

R: The basics

- Objects are stored in "workspace" (*i.e.* memory)
- To view all user-defined objects, use '*ls()*'

```
> ls()
[1] "a" "b" "c" "D"
```

- The contents of your workspace can be saved and loaded to continue at a later time

11

R: Using scripts

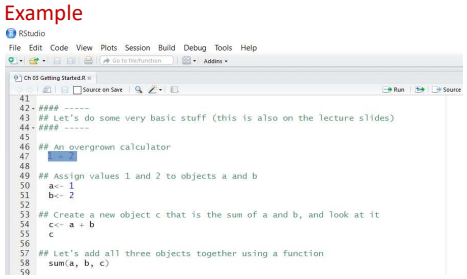
In Editor window

- Good scripting etiquette
  - Start with summarizing what the script is meant to do
  - Annotate commands (using: #)
    - The more you 'talk to yourself', the easier it is to remind yourself of what you're trying to do
  - Use indents
    - Greatly helps to keep the overview
  - Use similar style for all your scripts

12

R: Using scripts

- Example



Ctrl + R *or* Ctrl + Return ⌘ + Return

13

Set a working directory

- Working directory
  - Folder on your computer in which R will by default look for and save files
  - Good idea to define this folder at the beginning of each new script
- Set the working directory using *'setwd()'*, *e.g.*

```
> setwd("C:/BIO209/Data")
```

- We can now access files in this folder directly, *e.g.*

```
> myData<- read.csv("data.csv")
```

14

Installing packages

- Functions are available in "packages" (or "libraries")
- We need to **install** a package **just once**...

```
> install.packages("pastecs", dependencies= T)
```

  - R may ask you to select a 'CRAN mirror'
- ... but **load** it **every time** we restart R to be able to use a function it contains

```
> library(pastecs)
```

15

Getting help

- Within R, *e.g.*

```
> ?sum
```

or

```
> help(sum)
```

  - Information about the function will appear in the "Help" pane of the bottom-right window

- Also, Google is really helpful
  - "CRAN sum"
  - Also for much more complicated problems!

16

Getting data into R

- Creating variables

```
> a<- 1
> Age<- c(22, 27, 24)
> Name<- c("Ben", "Martin", "Carina")
```

'*c()*': combine

Numeric

String

- Types of variables
  - Numeric
    - Numbers (*e.g.* 7, -53.168)
  - String (character)
    - Letters (*e.g.* "Statistics", "Awesome")
  - Factor (or coding variable)
    - Represent different groups (*e.g.* "Male", "Female")
  - Date
    - Dates (*e.g.* 20-02-2018)

17

Getting data into R

- Creating a factor or coding variable

```
> Sex<- c(2, 2 , 1)
```

  - Now convert the object 'Sex' to a factor

```
> Sex<- factor(Sex, levels= c(1:2), labels=
                c("Female", "Male"))
```

- Creating a date variable

```
> DoB<- as.Date(c("1991-07-03", "1986-05-24",
                  "1989-06-21"))
```

  - Note how dates are entered as "yyyy-mm-dd"
  - There are many different date formats in R!!

18

DISCOVERING STATISTICS USING R

Getting data into R

- We can now combine these variables into a dataframe

```
> students<- data.frame(Name, Sex, DoB, Age)
> students
    Name    Sex        DoB Age
1    Ben   Male 1991-07-03  22
2 Martin   Male 1986-05-24  27
3 Carina Female 1989-06-21  24
```

- To view the structure of this object use '*str()*'

```
> str(students)
'data.frame':   3 obs. of  4 variables:
 $ Name: Factor w/ 3 levels "Ben","Carina",..: 1 3 2
 $ Sex : Factor w/ 2 levels "Female","Male": 2 2 1
 $ DoB : Date, format: "1991-07-03" "1986-05-24" ...
 $ Age : num  22 27 24
```

19

DISCOVERING STATISTICS USING R

Getting data into R

- Typically though, you will import data from a pre-existing file
  - *e.g.* from Access, Excel, OpenOffice, SPSS, …
  - Easiest to import from a non-proprietary format
    - *.txt*, *.dat*, *.csv* files
- Import files

```
> myData<- read.delim("filename.txt")
> myData<- read.delim("filename.dat")
> myData<- read.csv("filename.csv")
```

20

DISCOVERING STATISTICS USING R

Getting data into R

- After importing data, always check whether it worked using *e.g.* the '*str()*' and '*head()*' functions

```
> str(myData)
> head(myData, 7)
```

  - Often variables of the type Factor and Date are not properly imported and you need to correct this to make sure subsequent functions work

21

Getting data into R

- The book also mentions the use of *R commander*
  - A package (called 'Rcmdr') that adds a GUI to R
  - Useful at first, but soon limited functionality

- We will not cover the use of *R commander* in this course, so you can skip those sections in the book

22

Rest of afternoon…

- First practical
  - Create a working directory on your laptop for this course
  - Work through an example script together
  - Play around with the "Getting started" script
  - Read rest of Chapter 3 (§ 3.9 onwards)
  - While reading, follow along in R Studio with the "*Chapter 3 DSUR The R Environment.R*" script
  - Complete Smart Alex's tasks 2 and 3

23

Errata

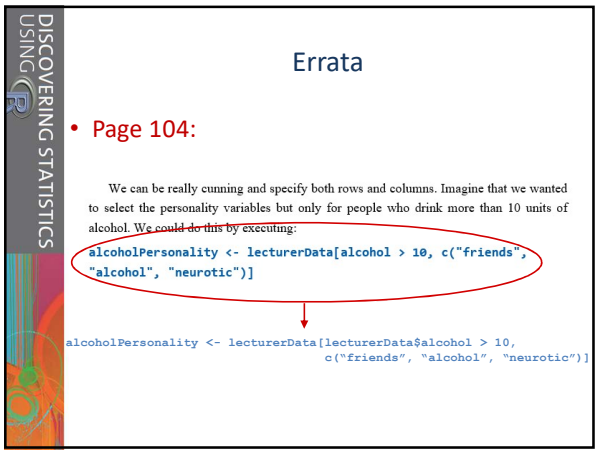
- Page 104:

Similarly, we can select specific cases of data by specifying an instruction for *rows* in the general function. This is done using a logical argument based on one of the operators listed in Table 3.5. For example, let's imagine that we wanted to keep all of the variables, but look only at the lecturers' data. We could do this by creating a new dataframe *(lecturerOnly)* by executing this command:

```
lecturerOnly <- lecturerData[job=="Lecturer",]
```

```
lecturerOnly <- lecturerData[lecturerData$job== "Lecturer", ]
```

24


DISCOVERING STATISTICS USING R
Errata
• Page 104:
We can be really cunning and specify both rows and columns. Imagine that we wanted to select the personality variables but only for people who drink more than 10 units of alcohol. We could do this by executing:
alcoholPersonality <- lecturerData[alcohol > 10, c("friends", "alcohol", "neurotic")]
alcoholPersonality <- lecturerData[lecturerData$alcohol > 10, c("friends", "alcohol", "neurotic")]

25