

R 2012

ASSIGNMENT 1

1. Prescribed manual

Venables WN, Smith DM and the R Development Core Team. 2010. *An Introduction to R* (AIR). Available for R Help: Manuals.

2. General

- (a) Website and downloading R: <http://www.R-project.org> or <http://cran.ru.ac.za> . To download R to your own computer: Navigate tobin/windows/base and obtain R-2.14.x.-win.exe.
- (b) S-PLUS (<http://www.insightful.com>): Statistical package and programming language.
- (c) S, S-PLUS and R.
- (d) Fundamental goal of S: ***Conversion of an idea into useful software***
- (e) The Mission of R (Chambers, John M, 2008. *Software for Data Analysis*. Springer: New York):
... to enable the best and most thorough exploration of data possible
- (f) The Prime Directive (Chambers, 2008):

... places and obligation on all creators of software to program in such a way that the computations can be understood and trusted

3. A quick sample R session

- (a) Click on R icon to open the *Commands window* or *Console*
- (b) At the R prompt `>` enter `5 - 8`
- (c) Repeat (b) but enter only `5 -` and see what happens.
- (d) Enter `xx <- 1:10` at the R prompt.
- (e) Enter `yy <- rnorm(10)` at the R prompt.
- (f) Enter `xx` at the R prompt.
- (g) Enter `yy <-` at the R prompt.
- (h) Enter `objects()` or `ls()` at the R prompt.
- (i) To remove objects: `rm(name1, name2, . . .)`
- (j) Enter `objects` at the R prompt.
- (k) Enter `plot(xx, yy)` at the R prompt.
- (l) End the R session by closing the window or entering `q()` at the R prompt.

4. S: an interpretive computer language

- What is the difference between a *compiler* computer language and an *interpretive* computer language?
- Advantages and disadvantages of an interpretive computer language.
- The R evaluator.

5. S /R: more basics

(a) S / R as an *interactive* language: Fast acquisition of results.

(b) S as a *functional* language.

(c) S as an *object-oriented* language.

- Objects: What? , Storage? , Permanence?
- Creating objects by *assignment*: `name <- object`

(d) The difference between `q()` and `q`.

(e) The help facility:

```
> ?name
> ??name
> help.search( )
> help.start( )
```

(f) Electronic manuals.

(g) Comments: using the symbol `#`.

6. From single instructions to sets of instruction: introducing R functions

Consider the following problem: the R data set `sleep` contains the extra hours of sleep of 20 patients after a drug treatment. Suppose this data set can be considered a sample from a normal population. Required is a 95% confidence interval for the mean extra hours of sleep. It is known that the confidence interval is given by $[\bar{x} - (s/\sqrt{n})t_{n-1; 0.025}; \bar{x} + (s/\sqrt{n})t_{n-1; 0.025}]$. This problem can be solved by entering the following instructions one by one:

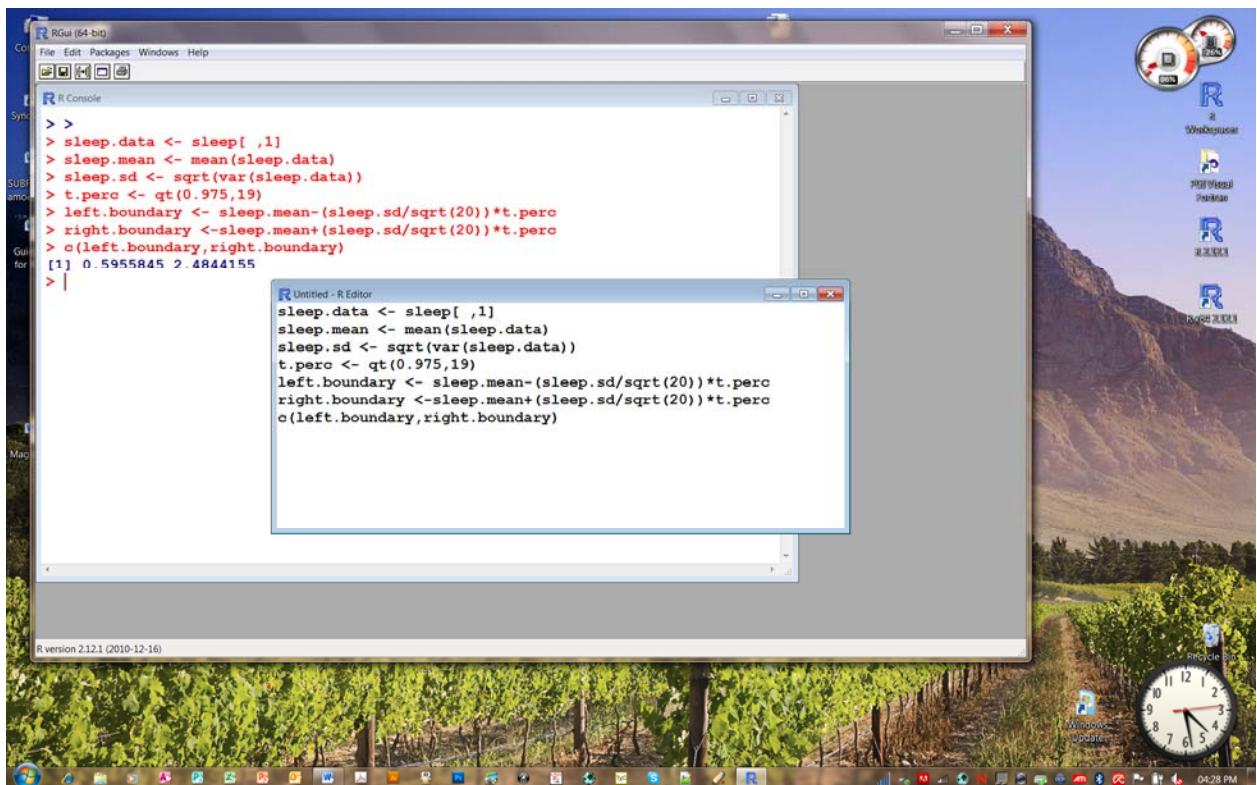
```
> sleep.data <- sleep[,1]
> sleep.mean <- mean(sleep.data)
> sleep.sd <- sqrt(var(sleep.data))
> t.perc <- qt(0.975,19)
> left.boundary <- sleep.mean-(sleep.sd/sqrt(20))*t.perc
> right.boundary <- sleep.mean+(sleep.sd/sqrt(20))*t.perc
```

In situations like the above, the problem can be addressed using a script file or writing a function. There are two methods for writing functions in R: (i) using a script file and (ii) using the function `fix()`. Both methods make use of a *text editor*. The built-in R *text editor* can be used when using script files but in the windows environment *notepad* or better still *notepad++* (available for free from www.notepad-plus-plus.org/download/) is preferred. The following

instruction is necessary for changing the default editor to be used with `fix()`: `> options(editor = "notepad")` or `> options(editor = "full path to exe file")`

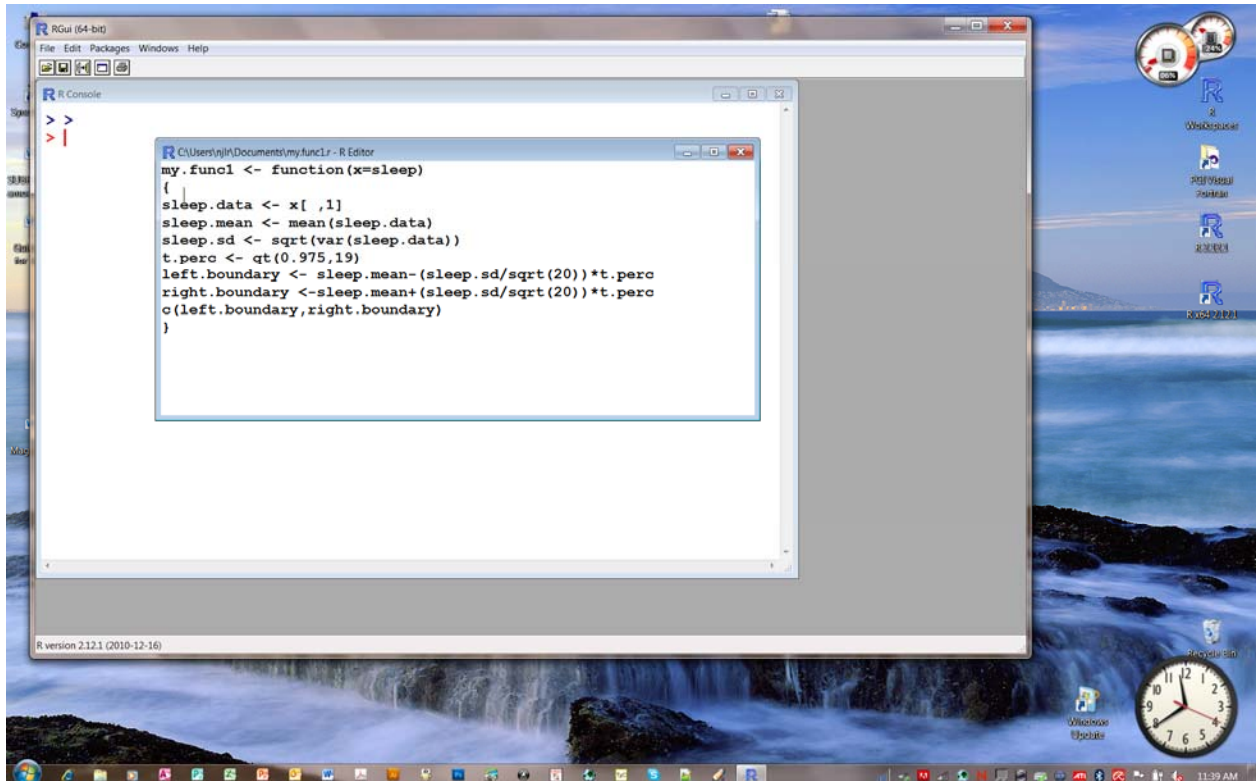
(a) Script file

- From the R top menu select File/New script. A script window will open (see below) with a simultaneous change in the menu bar. Note the name of the editor appearing in the script window.
- Type the instructions in the script window as shown in the figure below.
- Select all the typed text and run the script by clicking the run icon (or *Contr+R*).
- Note what is shown in the R console window.



- Script files are ordinary text files. They can be saved, edited and opened using any text editor.
- By convention R script files have the extension `xx.r`.
- Next, change the spelling of the last `right.boundary` to `Right.boundary`. Select all the text and run the script. Check the output on the console. Explain.
- Script windows can also be used for creating an R function – see the figure on the next page.
- Create an R function by changing the text in the script as shown in the figure on the next page.
- Select the text and notice what happens in the R commands window (the console).
- Give the instruction `objects()` at the R prompt. What has happened?

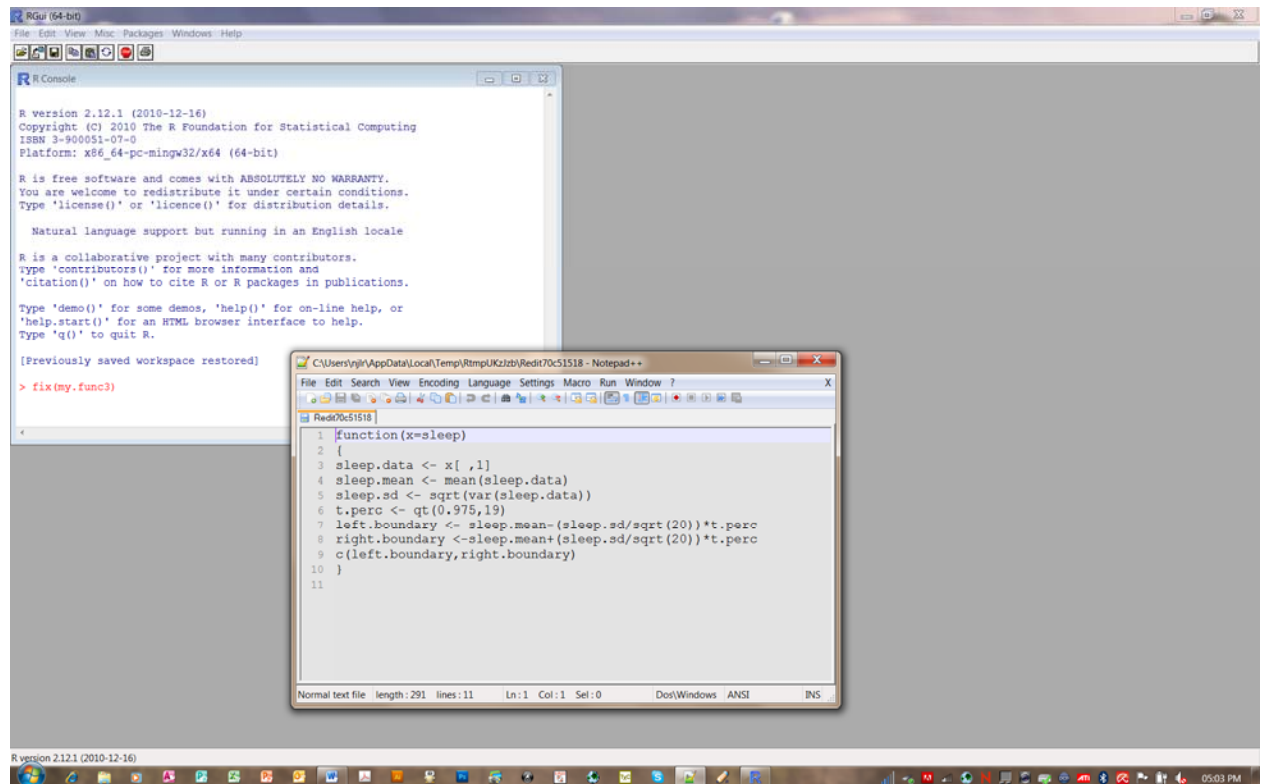
- If you want to create and run the function `my.func1` in a script window then add the instruction `my.func1(x=sleep)` as the last line in the script window. Now, select only this line and run it. Check the R console.



- What will happen if a syntax error is made in the script window? Change `my.func1` to `my.func2` and deliberately delete the last closing parenthesis. Select, run and check the R console. Discuss.

(b) Constructing a function using fix

- Make sure the default text editor is notepad or notepad++ (or any other text editor of your choice). This can be done using the following instruction from the R prompt:
`> options(editor="notepad")`
 or
`> options(editor="c:\\Program files (x86) \\ Notepad++ \\ Notepad++")`
- Enter `fix(my.func3)` at the R prompt. A text editor window will open. See figure below. Type the instructions as shown in the figure below. Close the window. Check what happens in the R console.



- What will happen if a syntax mistake is made when using fix? First copy `my.func3` to `my.func4`. Then at the R prompt type `fix(my.func4)`. Make a deliberate syntax error e.g. delete the last closing brace. Close the text editor window. What happens in the console? What is to be done to correct the mistake?
- Carefully study the message in the R console when a syntax error occurred in a function created by fix:

```
Error in edit(name, file, title, editor) :
  unexpected 'yyy' occurred on line xx
  use a command like
  x <- edit()
  to recover.
```

WARNING

Before writing a function for solving any problem: make sure the problem is understood exactly; make 100% sure the relevant statistical theory is understood correctly. Failure to do so is careless and dangerous!

- (i) Right-click on any open space on the Desktop and select New\Folder.
 - (ii) Right-click on the “New folder” and select Properties\Customize\Change icon\Browse.
 - (iii) Navigate to ...\\Program files\\R\\R-2.14.x\\bin\\Rgui.exe\\OK to assign the characteristic **R** icon of R to the group of directories.
 - (iv) Change the name of “New folder” to, for example, R-projects.
 - (v) Left-click on the R-projects folder (open the directory group). Now right-click on any open space of the opened folder. Now select:
 - (vi) New\\Shortcut\\Browse and navigate as above to ...\\Program files\\R\\R-2.14.x\\bin\\Rgui.exe. Choose “Finish”. We have now created within the R-projects directory group a “shortcut” to the R user’s directory and if this “shortcut” is activated, an R-session is initialised. The “shortcut” can be represented as an icon with an appropriate name.
 - (vii) The “shortcut” in (vi) can now be duplicated in R-projects. Right-click this duplicate and select Properties\\Start in. Provide in the given space the path to, for example, ...\\PROJECT01. Rename the “shortcut” concerned to, for example, PROJECT01
 - (viii) Repeat step (vii) for every separate project (each with own path and name) that you want to put in the R-projects folder. Select any of these icons in order to begin a session with the respective project.
- (d) Study how the following instructions work in R:
- File\\Load workspace
 - File\\Save workspace
 - File\\Change Dir
- (e) An alternative method for allocating each R project its own working directory
- (i) Create a folder for the project e.g. C:\\MyProject
 - (ii) Click on any existing R icon to open an R session
 - (iii) In R, select from the top menu: File\\Save Workspace
 - (iv) Navigate to C:\\MyProject and select: Save
 - (v) Close R session
 - (vi) In Windows go to C:\\MyProject and select *.RData*
 - (vii) It may be necessary to prepare the *.RData* file by (a) creating a suitable *.First* object and (b) by deleting all unnecessary objects e.g. by the instruction
`> rm(list = objects())` (BE CAREFUL, Why?)

8. Chapter 1: An Introduction to R

- Work through §1.1 to §1.5.
- Do §1.6.
- Work through §1.7 to §1.11.

9. Built-in data sets in R

- (a) R contains several built-in data sets collected in the package *datasets*. This package is automatically attached to the search path. Type `?datasets` at the R prompt for details. Apart from these data sets several other data sets are also used in this module. These data sets are available as R-objects in the Fharga pool under the names *auto.stats.spd*, *saving.x*, *rain.nyc1.spd* and *rain.nyc2.spd* respectively. These data sets can be imported into R as R-objects with the help of the following instruction:

```
> source("Full path ... \\ . . . \\ . . . \\ Datasets \\ name").
```

Note the use of the double `\\`.

- (b) The use of **.First** and **.Last**.

The function `.First()` is executed at the beginning of every R session. Therefore `.First <- function() options(editor = "notepad")` ensures that “Notepad” is the text editor during the current session.

10. Security

An example of the usage of `.First`

The `.First` facility can be used to gain access to a R working-directory that is password-protected. This can be done as follows:

- Create the following R function:

```
password <- function()
# Note the structure of a function
{cat("Password? \n")
password <- readline()
if (password != "PASSWORD") q(save="no"
)
else (cat("You can proceed \n"))
}
```

- Now create the function:

```
.First <- function(){# What must you be careful of?
password()
}
```

Execute the above function.

Discuss the construction and usage of the above function.

11. Options

- (a) Study the result of the instruction `> options()` in R.
- (b) Study the R main menu choice *Edit / GUI preferences*.

12. R output (text and graphics) to MS Word

R (and Splus) can be used together with MS Word (and Powerpoint) to round off statistical analyses with a report. The basic principles are illustrated with the following example:

- (a) Initialise an R session and open an x.doc file.
- (b) Activate the *auto.stats* data set in R.
- (c) Execute the following R instructions: (i) `apply(auto.stats[, 1:3], 2, mean)` and (ii) `hist(auto.stats[, 2])`.
- (d) Select the applicable R text output and copy it to the x.doc file.
- (e) Activate the x.doc file and change the font of the copied R text output to (i) Times New Roman and (ii) Courier New. What very important difference is noticed? What is the reason for this?
- (f) Now activate the R graph-window. Copy the graph by either of the following
 - (i) Ctrl+W shortcut
 - (ii) File/Copy to clipboard/ as a Metafile.
- (g) Activate x.doc and execute the following: Edit/Paste special/Picture.
- (h) Select the graph in x.doc. Right-click and select Format Picture. Study the different options to change the size, layout, colours and other characteristics of the graph.
- (i) **Important:** what is the correct way of resizing in MS Word a graph constructed with R?
- (j) Remark: Powerpoint provides various additional facilities to edit graphs with. (E.g., changing the colours and plotting characters.)

13. Command line editing

- Use of the up and down arrows.
- Use of backspace, delete, home, end, copy and paste.
