D1–Introduction to Computers, Programming and R

Revised on March 18, 2016

1

1. Start **R**.
2. You are now at the prompt of **R**. Read the welcome information and find out how to quit **R** properly.
3. Start **R** again and treat it as a calculator to do the following calculations

```
3 + 4
5 * 6
12/7
2^3
45-2*3
(45-2)*3
```

4. Do the above in another way like this

```
a <- 3
b <- 4
c <- a + b
```

   Now check what `a,b,c` are. You can just type e.g. `a` and RETURN or use the command `print(a)`.
5. Type `quit()` and answer `no`.
6. Start **R** and check if `a,b,c` are still there.
7. Redo **6.** and then quit **R** by answering `yes`.
8. Start **R** and check if `a,b,c` are still there. What did you see?
9. What do `yes` and `no` mean when we use the `quit()` command? You may check the help of `quit()`. Type `?quit` at the **R** prompt and use the **PgUp** and **PgDn** keys on the keyboard to navigate. To quit the current **R** help in the terminal, type `q`.
10. ♡ Discussion: for a computer to complete a task like procedure **1.9**–**1.10** what major role do CPU, memory, hard drive, keyboard and screen play?

2

1. Check what variables you have in the current **R** session, type `ls()`.
2. Remove all of those variables. [Hint: type `??remove` to search possible matches. Read the help for individual functions before you issue any command.]
3. ♡ What's the difference between "`?`" and "`??`"? [Hint: try `help("?")` and `help("??")`]
4. What other ways of help are available in **R**? How to launch the html version of **R** help documentation? [Hint: read **2.3** carefully.]

3

1. ♡ How many ways can you find to assign `5` to variable `myVar`? Show with examples.
2. ♡ What's the difference between "`<-`" and "`=`"? Show with examples.
3. Try the following commands line by line and discuss their implications. [Hint: Be careful of the space you are typing.]

```
a <- 5
print(a)
a < - 7
print(a)
rm(a)
a < - 7
```

4. Read the help of the function `print()` and try

```
x <- "The value of pi is"
print(x)
print(pi)
```

5. Repeat the above commands but replace `print` with `cat`. Use the `cat()` function, `x` and `pi` to print the following sentence to the screen:

```
The value of pi is: 3.141593.
```

6. Repeat the above task but with the function `message()`. What are the differences between `print`, `cat`, and `message` ?

4

1. Create the following vectors:

```
myVar1 <- c(1,2,3,4,5)
myVar2 <- c("Mon","Tue","Wed","Thu","Fri", "Sat", "Sun")
myVar3 <- c(TRUE, TRUE, FALSE, FALSE, FALSE)
myVar4 <- c(0.2,0.4,0.6,0.8,1.0)
myVar5 <- c("Jim","Apple","Linda","School","Math")
myVar6 <- c(2,3,6,7,8,9)
```

2. ♡ How many elements do you have in each vector? [Hint `?length`]
3. What types of vectors are they? [Hint `?mode`]
4. Do some basic calculations e.g.

```
myVar1 + myVar4
myVar1 - myVar4
myVar1 * myVar4
myVar1 / myVar4
(myVar1 - myVar4)^2
myVar1^2 - exp(myVar4)
log(myVar1)
log(myVar1) + 1
```

and speculate on why obtained those results.

5

1. Assume you have the variable `myVar6` and you want to replace the fourth element `7` with `7000`. How can you manage that? [Hint: Use the `fix()` or `edit()` functions].
2. Print your new `myVar6` to check if the variable has been updated.
3. Use the `getwd()` to check the location of your current directory, and assign the current working directory path to a variable `myOldDir`.
4. Use `save.image()` to save all your available variables to a file named "`myOldFullVars.RData`" under current directory.
5. Use `dir()` function to check if you have saved the file successfully.
6. Switch your working directory to a temporary directory "/tmp/" via the function `setwd()`.
7. Use `getwd()` again to check if your working directory have been switched successfully.
8. Again use `ls()` to list all the available variables.
9. Use `save()` to only save some variables to a file "`myNewVars.RData`" under current directory.
10. Quit **R** and start **R** again.
11. Clear all the variables with the command `rm(list=ls())`.
12. Load the file "`myOldFullVars.RData`" using the function `load()` and use `ls()` and `print()` to check if your old variables are all right.
13. ♡ Clear all the variables again and load the file "`myNewVars.RData`". Do you still have the variables that you saved? If not, why?

6

1. ♡ Open the R editor (File → New) and save it as `D1-FirstName.LastName.R` Write your R code to the following questions (**7.3**-**7.8**).
2. Use `read.table()` to read dataset "`Apple.txt`" and name it as "`Apple`". The data file is included in the lab assignment. Place the file in your current working directory so that **R** can find it. [Hint: Use the arguments `header` and `sep`. Look it up in the help function for read.table]
3. Print the variable and check the data type.
4. Export/Save the variable "`Apple`" into the **R** data format as "`Apple.RData`".
5. Repeat the above 1.–4. for "`NASDAQ100.csv`". [Hint: try `read.csv()`].
6. Use an external program (e.g. office suit) to open the file "`Google.xls`" and save it as `Google.csv`. Repeat the loading and exporting procedure.
7. Load the **R** datasets you have converted and export them to `txt` and `csv` formats. [Hint: try `dput()` and `write.csv()` functions]. Attach the dataset file "`NASDAQ100.txt`" in your submitted report.

7 Extra

Installing **R** on your own computer is simple and free.

1. If you use Microsoft Windows, visit `http://www.r-project.org/`. It shipped with a simple graphical user interface (GUI).
2. R is available for Mac OS X users with a nice build-in graphical user interface, visit `http://www.r-project.org`

3. If you use Linux, search the phrase "r-cran" in your system's software repository. Any text editor can be used to edit R source code. But if you want a graphical user interface, RStudio (with is easy to install and also available for other platforms) is a good start. Please visit `http://www.rstudio.org/download/desktop`.