
Data Programming Course Exercises

May 3, 2016

Andrea Spanò
andrea.spano@quantide.com¹

¹<mailto:andrea.spano@quantide.com>

Contents

1	Introduction	5
2	Data Object	7
2.1	Vectors	7
2.1.1	Exercise 1	7
2.1.2	Exercise 2	8
2.1.3	Exercise 3	8
2.2	Matrices	9
2.2.1	Exercise 1	9
2.2.2	Exercise 2	9
2.2.3	Exercise 3	9
2.2.4	Exercise 4	10
2.3	Lists	11
2.3.1	Exercise 1	11
2.3.2	Exercise 2	12
2.4	Factors	13
2.4.1	Exercise 1	13
2.4.2	Exercise 2	14
2.5	Data Frames	14
2.5.1	Exercise 1	14
3	Data Import	19
3.1	Text Files	19
3.1.1	Exercise 1	19
3.1.2	Exercise 2	20

3.1.3	Exercise 3	20
3.2	Excel Files	21
3.2.1	Exercise 1	21
3.2.2	Exercise 2	21
3.3	Databases	22
3.3.1	Exercise 1	22
3.4	R Data Files	23
3.4.1	Exercise 1	23
3.4.2	Exercise 2	23

Chapter 1

Introduction

In this document you will find some exercises about R data objects and about data import and export.

Chapter 2

Data Object

2.1 Vectors

2.1.1 Exercise 1

- a. Create a vector, named `vec1`, containing the following values:

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90

```
vec1 <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90)
```

- b. Select the 5-th element of `vec1`.

```
vec1[5]
```

```
## [1] 5
```

- c. Select the first 10 elements of `vec1`.

```
vec1[1:10]
```

```
## [1] 1 2 3 4 5 6 7 8 9 10
```

- d. Select all the elements of `vec1` apart from the 2nd and the 6th element.

```
vec1[-c(2,6)]
```

```
## [1] 1 3 4 5 7 8 9 10 15 20 25 30 35 40 45 50 60 70 80 90
```

2.1.2 Exercise 2

- a. Generate a vector, named `vec2`, containing the numbers from 1 to 10 and of length 8, using the function `seq()`.

```
vec2 <- seq(from=1, to=10, length.out = 8)
```

- b. Select the values of `vec2` which are greater than 4.

```
vec2[vec2>4] # or y > 4; b[y]
```

```
## [1] 4.857143 6.142857 7.428571 8.714286 10.000000
```

- c. Select the values of `vec2` which are equal or less than 2 or which are equal or greater than 6.

```
vec2[vec2<=2 | vec2>=6]
```

```
## [1] 1.000000 6.142857 7.428571 8.714286 10.000000
```

2.1.3 Exercise 3

- a. Generate the following vector using the function `rep()`:

```
vec3 <- c("one", "two", "one", "two", "one", "two")
```

```
vec3 <- rep(c("one", "two"), times=3)
```

- b. Generate a new vector, named `vec5`, combining the previous vector, `vec3`, with the following one:

```
vec4 <- c("three", "four")
```

```
vec5 <- c(vec3, vec4)
vec5
```

```
## [1] "one" "two" "one" "two" "one" "two" "three" "four"
```


2.2 Matrices

2.2.1 Exercise 1

Generate a matrix, named `mat1`, with 5 rows and 3 columns, using `matrix()` function:

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
## [4,]   10   11   12
## [5,]   13   14   15
```

```
mat1 <- matrix(1:15, nrow = 5, ncol = 3, byrow = TRUE)
```

2.2.2 Exercise 2

Starting from the following vector:

```
mat2 <- 1:8
```

Generate a matrix with 2 rows and 4 columns using `dim()` function.

```
dim(mat2) <- c(2,4)
mat2
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    3    5    7
## [2,]    2    4    6    8
```

2.2.3 Exercise 3

- a. Generate a matrix, named `mat3`, combining the following columns:

```
a <- 1:3
b <- 7:9
c <- 8:6
```

```
mat3 <- cbind(a,b,c)
mat3
```

```
##      a b c
## [1,] 1 7 8
## [2,] 2 8 7
## [3,] 3 9 6
```

b. Add the following row to `mat3`:

```
d <- 4:6

mat3 <- rbind(mat3, d)
mat3

##      a b c
##      1 7 8
##      2 8 7
##      3 9 6
## d 4 5 6
```

2.2.4 Exercise 4

Considering the following matrix, named `mat4`:

```
mat4 <- matrix(1:24, nrow = 6, ncol = 4, byrow = TRUE)
mat4

##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    5    6    7    8
## [3,]    9   10   11   12
## [4,]   13   14   15   16
## [5,]   17   18   19   20
## [6,]   21   22   23   24
```

a. Select the third and the fifth row of `mat4`.

```
mat4[c(3,5),]

##      [,1] [,2] [,3] [,4]
## [1,]    9   10   11   12
## [2,]   17   18   19   20
```

b. Select all columns of `mat4` apart from the first.

```
mat4[, -1]

##      [,1] [,2] [,3]
## [1,]    2    3    4
## [2,]    6    7    8
## [3,]   10   11   12
## [4,]   14   15   16
## [5,]   18   19   20
## [6,]   22   23   24
```

c. Select second and third rows and second and third columns of `mat4`.

```
mat4[2:3, 2:3] # or mat4[c(2,3) , c(2,3)]

##      [,1] [,2]
## [1,]    6    7
## [2,]   10   11
```

2.3 Lists

2.3.1 Exercise 1

a. Generate a list, named `list1` that contains the following R elements:

```
vec <- 1:10
mat <- matrix(1:9, ncol = 3)
name <- "Oscar"

list1 <- list(vec = 1:10, mat = matrix(1:9, ncol = 3), name = "Oscar")
list1

## $vec
## [1] 1 2 3 4 5 6 7 8 9 10
##
## $mat
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
##
## $name
## [1] "Oscar"
```

b. Add to `list1` the following element:

```
letters <- c("a", "b", "c", "d")

list1$letters <- letters
list1

## $vec
## [1] 1 2 3 4 5 6 7 8 9 10
##
## $mat
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
##
## $name
## [1] "Oscar"
##
## $letters
## [1] "a" "b" "c" "d"
```

2.3.2 Exercise 2

Given the following list, named `list2`:

```
list2 <- list(vec = c(1,3,5,7,8), mat = matrix(1:12, ncol = 4),
             sub_list = list(names = c("Veronica", "Enrico", "Andrea", "Anna"),
                             numbers = 1:4))

list2

## $vec
## [1] 1 3 5 7 8
##
## $mat
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
##
## $sub_list
## $sub_list$names
## [1] "Veronica" "Enrico"  "Andrea"  "Anna"
##
## $sub_list$numbers
## [1] 1 2 3 4
```

- a. Extract the first element of `list2`.

```
list2[1]

## $vec
## [1] 1 3 5 7 8
```

- b. Extract the objects contained in the first element of `list2`.

```
list2[[1]]

## [1] 1 3 5 7 8
```

- c. Extract the element named `sub_list` of `list2`.

```
list2$sub_list

## $names
## [1] "Veronica" "Enrico"   "Andrea"   "Anna"
##
## $numbers
## [1] 1 2 3 4
```

- d. Extract the second rows of the matrix included in the second element of `list2`.

```
list2[[2]][2,] # or list2$mat[2,]

## [1] 2 5 8 11
```

2.4 Factors

2.4.1 Exercise 1

Starting from the vector:

```
fac1 <- c("F", "F", "M", "M", "F")
```

Generate the corresponding factor with two levels: “F” and “M”

```
fac1 <- factor(fac1, levels = c("F", "M"))
fac1

## [1] F F M M F
## Levels: F M
```

2.4.2 Exercise 2

Starting from the vector:

```
fac2 <- c(1, 1, 1, 2, 2, 2)
```

- a. Generate the corresponding factor considering that 1 = “Female”, 2 = “Male” e 3 = “Trans”.

```
fac2 <- factor(fac2, levels = c(1,2,3), labels = c("Female", "Male", "Trans"))
fac2
```

```
## [1] Female Female Female Male   Male   Male
## Levels: Female Male Trans
```

- b. Select the all elements of `fac2` apart from “Male”.

```
fac2[fac2!= "Male"]

## [1] Female Female Female
## Levels: Female Male Trans
```

2.5 Data Frames

2.5.1 Exercise 1

- a. Generate a data frame, named `df1`, corresponding to:

```
##      id      name class mean
## 1     1      Luca   5A  6.0
## 2     2   Chiara   5A  7.0
## 3     3      Lisa   5A  5.0
## 4     4   Matteo   5A  6.5
## 5     5    Alice   5A  7.5
## 6     6    Marco   5B  4.5
## 7     7  Veronica   5B  9.0
## 8     8    Nicola   5B  8.0
## 9     9     Elena   5B  8.5
## 10    10  Daniele   5B  7.0
```

Remember to maintain character vectors as they are, specifying `stringsAsFactors = FALSE`.

```
df1 <- data.frame(id=1:10,
                  name=c("Luca", "Chiara", "Lisa", "Matteo", "Alice", "Marco",
                        "Veronica", "Nicola", "Elena", "Daniele"),
                  class=c(rep("5A", times=5), rep("5B", times=5)),
                  mean= c(6,7,5,6.5,7.5,4.5, 9, 8, 8.5, 7), stringsAsFactors = FALSE)
```

```
df1
```

```
##      id      name class mean
## 1     1      Luca   5A  6.0
## 2     2  Chiara   5A  7.0
## 3     3      Lisa   5A  5.0
## 4     4  Matteo   5A  6.5
## 5     5   Alice   5A  7.5
## 6     6   Marco   5B  4.5
## 7     7 Veronica   5B  9.0
## 8     8   Nicola   5B  8.0
## 9     9    Elena   5B  8.5
## 10    10 Daniele   5B  7.0
```

```
# Other solution
```

```
id <- 1:10
name <- c("Luca", "Chiara", "Lisa", "Matteo", "Alice", "Marco",
          "Veronica", "Nicola", "Elena", "Daniele")
class <- c(rep("5A", times=5), rep("5B", times=5))
mean <- c(6,7,5,6.5,7.5,4.5, 9, 8, 8.5, 7)
df1 <- data.frame(id, name, class, mean, stringsAsFactors = FALSE)
```

```
df1
```

```
##      id      name class mean
## 1     1      Luca   5A  6.0
## 2     2  Chiara   5A  7.0
## 3     3      Lisa   5A  5.0
## 4     4  Matteo   5A  6.5
## 5     5   Alice   5A  7.5
## 6     6   Marco   5B  4.5
## 7     7 Veronica   5B  9.0
## 8     8   Nicola   5B  8.0
## 9     9    Elena   5B  8.5
## 10    10 Daniele   5B  7.0
```

b. Select the first 3 rows of df1.

```
df1[1:3,]
```

```
##      id      name class mean
## 1     1      Luca   5A     6
```

```
## 2 2 Chiara 5A 7
## 3 3 Lisa 5A 5
```

- c. Select the last 6 rows and the first 3 columns of `df1`.

```
df1[5:10, 1:3]

##      id      name class
## 5     5    Alice   5A
## 6     6    Marco   5B
## 7     7  Veronica   5B
## 8     8    Nicola   5B
## 9     9     Elena   5B
## 10    10  Daniele   5B
```

- d. Select the column `class` of `df1`.

```
df1$class

## [1] "5A" "5A" "5A" "5A" "5A" "5B" "5B" "5B" "5B" "5B"
```

- e. Convert the column `class` of `df1` in a factor with levels: “5A” and “5B”

```
df1$class <- factor(df1$class, levels = c("5A", "5B"))
df1$class

## [1] 5A 5A 5A 5A 5A 5B 5B 5B 5B 5B
## Levels: 5A 5B
```

- f. How many columns and rows `df1` has?

```
dim(df1) # or ncol(df1) and nrow(df1)

## [1] 10 4
```

- g. Generate another dataframe, named `df2` composed by the columns `name` and `mean` of `df1`, specifying the argument `stringsAsFactors = FALSE`.

```
df2 <- data.frame(name = df1$name, mean=df1$mean, stringsAsFactors = FALSE)
df2
```



```
##      name mean
## 1    Luca  6.0
## 2  Chiara  7.0
## 3    Lisa  5.0
## 4  Matteo  6.5
## 5    Alice  7.5
## 6    Marco  4.5
## 7  Veronica  9.0
## 8    Nicola  8.0
## 9     Elena  8.5
## 10 Daniele  7.0
```

h. Show the first rows and the structure of `df2`.

```
head(df2)
```

```
##      name mean
## 1    Luca  6.0
## 2  Chiara  7.0
## 3    Lisa  5.0
## 4  Matteo  6.5
## 5    Alice  7.5
## 6    Marco  4.5
```

```
str(df2)
```

```
## 'data.frame':   10 obs. of  2 variables:
## $ name: chr  "Luca" "Chiara" "Lisa" "Matteo" ...
## $ mean: num  6 7 5 6.5 7.5 4.5 9 8 8.5 7
```


Chapter 3

Data Import

First of all, set your working directory in the *data* folder, using `setwd()` function, like in this example

```
setwd("C:/Users/Veronica/Documents/rbase/data")
```

We will work inside this folder.

3.1 Text Files

3.1.1 Exercise 1

- a. Import text file named *"tuscany.txt"* and save it in an R object named `tuscany_df`.
Open the text file before importing it to control if the first row contains column names and to control the field and the decimal separator characters. Remember to not import the character columns as factors.

```
tuscany_df <- read.table("tuscany.txt", header = TRUE, sep = "|",  
                        dec=".", stringsAsFactors = FALSE)
```

- b. Visualize the first rows of `tuscany_df`

```
head(tuscany_df)
```

##	id	sex	year_of_birth	marital_status	income	house_number
## 1	1	M	1969	married	16101.1	5144.0
## 2	2	M	1962	single	17220.0	6158.0
## 3	3	M	1965	divorcee	28801.9	10078.0
## 4	4	F	1968	single	25964.0	11133.7
## 5	5	M	1975	married	16522.5	5078.0
## 6	6	M	1977	married	18124.0	5115.0

	city_name	province	provincial_acronym
## 1	Riparbella	Pisa	PI
## 2	Capolona	Arezzo	AR
## 3	Pomarance	Pisa	PI
## 4	Cascina	Pisa	PI
## 5	Quarrata	Pistoia	PT
## 6	Castiglion Fiorentino	Arezzo	AR

3.1.2 Exercise 2

Import 7 rows of the text file named “*solar.txt*” skipping the first two rows. Save it in the object `solar_df`.

Open the text file before importing it to control if the first row contains column names and to control the field and the decimal separator characters. Remember to not import the character columns as factors.

```
solar_df <- read.table("solar.txt", header = FALSE, sep = ",",
                      dec=".", stringsAsFactors = FALSE,
                      nrows = 7, skip = 2)

solar_df
```

	V1	V2	V3	V4
## 1	mar	23877	24671	22455
## 2	apr	24377	23677	23670
## 3	mag	24581	25476	24999
## 4	giu	22154	21998	22451
## 5	lug	20924	21645	23871
## 6	ago	23183	22576	23556
## 7	set	27446	27695	28664

3.1.3 Exercise 3

Considering the following data frame, named `df`:

```
df <- data.frame(col1=1:4, col2=4:1, col3=c("one", "two", "three", "four"),
                 stringsAsFactors = FALSE)
```

Save it in a .txt file named “*exercise-3.txt*” in *data* folder.

```
write.table(df, file="exercise-3.txt")
```



```
##   a     b
## 1 1   one
## 2 2   two
## 3 3 three
## 4 4  four
```

Add it to `df` sheet of `ex_2` R workbook object, starting from row 3 and from column 2. Use the function `writeWorksheet()`. Remember to save the changes also in `.xlsx` file (use `saveWorkbook()` function).

```
writeWorksheet(object = ex_2, data = numbers_df, sheet = "df", startRow = 3, startCol = 3)
saveWorkbook(ex_2)
```

3.3 Databases

3.3.1 Exercise 1

- Connect to “*plant.sqlite*” SQLite database, using `dbConnect()` function of `RSQLite` package. Save the connection in an R object, named `con`. Remember to load `RSQLite` package, supposing it is already installed.

```
require(RSQLite)
```

```
con <- dbConnect(RSQLite::SQLite(), "plant.sqlite")
```

- See the list of available tables in “*plant.sqlite*” db, using `dbListTables()` function.

```
dbListTables(con)
```

```
## [1] "PlantGrowth"
```

- See list of fields in “*PlantGrowth*” table of “*plant.sqlite*” db, using `dbListFields()` function.

```
dbListFields(con, name = "PlantGrowth")
```

```
## [1] "weight" "group"
```

- Send query to “*PlantGrowth*” table of “*plant.sqlite*” which select the records with `weight` greater than 5.5.

```
dbGetQuery(con, "SELECT * FROM PlantGrowth WHERE weight >= 5.5")
```

```
##   weight group
## 1   5.58  ctrl
## 2   6.11  ctrl
## 3   5.87  trt1
## 4   6.03  trt1
## 5   6.31  trt2
## 6   5.54  trt2
## 7   5.50  trt2
## 8   6.15  trt2
## 9   5.80  trt2
```

e. Disconnect from the database, using `dbDisconnect()` function.

```
dbDisconnect(con)

## [1] TRUE
```

3.4 R Data Files

3.4.1 Exercise 1

Given the following data frame, named `df_rdata`:

```
df_rdata <- data.frame(a=1:20, b=20:1)
```

Save it in *.Rda* format in the file “*df_rdata.Rda*”, using `save()` function.

```
save(df_rdata, file = "df_rdata.Rda")

## [1] TRUE
```

3.4.2 Exercise 2

Load “*drug.Rda*” file into the environment, using `load()` function.

```
load("drug.Rda")
```