Applied Statistics: R
Francisco Rosales Marticorena **Semester 2018-I**

1 R Basics

Exercise 1 (Data structures and subsetting)

(a) Create a `list` named `list1` with the following components

`x` : a numeric vector $35, 34, 33, \ldots, 7, 6, 5, 6, 7, \ldots, 33, 34, 35$ (don't type the single numbers!)

`y` : a factor with the first five letters of the Latin alphabet as levels, each repeated 12 times, except the last one which should be repeated 13 times

`mat` : a numeric 4×4 matrix whose entries are randomly drawn from an exponential distribution with $\lambda = 5$. Use the `set.seed()` function to make your result reproducible.

`list2` : a list with the components

`t` : a numeric variable with the value 35

`d` : a data.frame with the variables

`gender` : a factor with elements male, male, male, female, female, female, male, male, male, female, female, female

`age` : a numeric vector with elements $23, 48, 37, 37, 19, 54, 21, 20, 41, 26, 35, 32$

(b) Use subsetting operators to access the following information

(a) the 4th and the 7th element of the vector `x`

(b) the entry on position $(3, 4)$ of the matrix `mat`

(c) the first six elements of the first column of the data frame `d`

(d) the age of the female individuals

What is the difference between the usage of `[]`, `[[]]` and $?

(c) Modify the created objects in the following ways

(a) redefine the levels of the factor y from lowest to highest as 'c', 'd', 'b', 'a', 'e'

(b) eliminate the first row and the third column of `mat`

(c) add a column `age2` to `d` with a factor taking the value 'old', if the individual's age is above the threshold `t`, and the value 'young' otherwise

(d) exclude the individuals that are at over (or exactly) 50 and strictly younger than 21 from the study

Exercise 2 (Loops and data manipulation)

(a) Create a matrix X of dimension 2500×12 containing random numbers $x_{ij} \sim \mathcal{N}(\mu = 0, \sigma^2 = 3)$.

(b) Create a new matrix $(y_{ij})_{ij}$ based on the one defined above, in which each column is standardized, i.e. $y_{ij} = \frac{x_{ij} - \bar{x}_j}{\sigma_j}$, where $\bar{x}_j$ is the sample mean of the j-th column, and σ_j its sample standard deviation.

(c) From X keep only the rows for which at least 6 out of the 12 values are greater than 0. Hint: Remember that a logical vector can be converted to a numeric vector where `FALSE` takes the value 0, and `TRUE` the value 1.

(d) Check, if the elements of the 3rd column of mat from Exercise 1 are within the range defined by the elements in the first two columns. The output should be a logical vector.

(e) In each row of X replace the maximum value with the minimum value.

(f) Write a for-loop to do the following calculation for $i = 1, \ldots, 2500$:

- if the i-th element of the first column X is positive, then add to the subsequent element $x_{i+1,j}$ a random number $u \sim \mathcal{U}(-1, 1)$
- otherwise, if x_{ij} is negative, add to the subsequent element $x_{i+1,j}$ a random number $u \sim \mathcal{U}(-2, 2)$.

(g) Consider the data.frame resulting from

```
set.seed(2015)
w <- runif(10)
x <- runif(10)
y <- runif(10)
z <- runif(10)
dat <- data.frame(a = w, b = x, c = y, d = z).
> dat
            a          b         c          d
1  0.06111892 0.70285557 0.6919100 0.75185674
2  0.83915986 0.39172125 0.4067718 0.25684487
3  0.29861322 0.03306277 0.2109400 0.38967137
4  0.03143242 0.40940319 0.6652073 0.88448520
5  0.13857171 0.74234713 0.7377556 0.57390551
6  0.35318471 0.88301877 0.9190050 0.18367673
7  0.49995552 0.26623321 0.8734601 0.11168811
8  0.07707116 0.07427093 0.8012774 0.32047459
9  0.65134483 0.81368426 0.5243978 0.09095567
10 0.51172371 0.38194719 0.1272213 0.47959896
```

Now imagine in each row the entries define two intervals $[a, b]$ and $[c, d]$. Add a column to `DF` with entry `TRUE`, if the intervals overlap and `FALSE`, if they don't.
Find ways to avoid for-loops in (a)–(c). Useful functions might be

`rnorm(), runif(), apply(), mean(), sd(), scale(), rowSums(), min(), max().`

If you are not familiar with some of these functions, use `?function` to check the documentation. Compare the `system.time()` of the solutions with and without for-loops.

Exercise 3 (Basic graphical tools)

(a) Consider again the matrix X from Exercise 2 (a) and create the following plots:

- A scaled histogram of the first column. Also add a dashed red line representing the estimated density.
- A quantile-quantile plot comparing the sample quantiles of the first and second column. Also add a line with intercept 0 and slope 1 to improve comparability. Try different `pch`-values and colours.
- Boxplots of the first, fourth and seventh row in one plotting window. Rename the group labels to `blue, green and yellow.`

2 Advanced Graphics

Exercise 1 (From basic plots to complex graphics)
Read the data set `school_math.raw`. It describes the mathematical achievements (`mathach`) of male and female (`Sex`) students at 4 different schools (`school`). Additionally there is information about the socio-economic status (`ses`) of the students family and the sector (`Sector`) of the school (public or catholic).

(a) Get familiar with the data set, e.g. via the functions `summary()` or `head()`.

(b) Create the following four plots in only one plotting window:

a scatterplot of the achievement vs. the socio-economic status. Add a line representing the estimated linear relationship between those variables.

a histogram of `mathach`.

a scaled histogram of `ses`. Add a line with the estimated density.

boxplots of the mathematical achievement for each school separately.
First, don't modify any options. In a second step include the following variations:

- The color of the numbers on the axes should be red.
- Data points should be represented by filled squares (■) instead of circles (○).
- Use squared plotting regions for each plot.
- All occurring lines should be dashed.

Modify your plot by changing the settings within the single plot functions and in the overall graphics options using `par()`.
Hint: Make a backup of the default par() settings, such that you can reset the settings to default afterwards.
Hint: The documentations of plot, points, lines, abline, par, hist, boxplot might be helpful.

(c) Reset the graphic settings to the default values.

Exercise 2 (The layout function)
Add additional information to the scatterplot from Exercise 1 in form of boxplots of the data at the axes. For this purpose, get familiar with the function `layout()` and its options. Also use `layout.show()` to visualize different settings.

Exercise 3 (Lattice plots and grouped data)

(a) Estimate regression lines of the form

$$\texttt{mathach} = \beta_0 + \beta_1 \cdot \texttt{ses} + \epsilon$$

for the different schools separately. Plot your point estimates against the school. Use one plotting window for the intercepts and another one for the slopes. The y-axes should be labeled β_0 and β_1, respectively.
Hint: Look at the function `lmList()` from the package `nlme`. Use `expression()` for the labeling.

(b) Plot the mathematical achievement (in dependence of the `ses`) of boys as green triangles and of girls as red squares. Add a legend to the plot indicating this grouping.

(c) Load the packages lattice and `nlme`. For the school data set, we want to compare the data in the different groups. For that purpose create a groupedData object with response `mathach`, covariate `ses` and school as grouping variable. Now

- Plot the new object. What is the difference between the results from plot(`data`) and plot(`data_new`), where data is the original data set without grouping structure and `data_new` is the `groupedData` object?
- Create box-whiskers plots (`bwplot()`) of the residuals from an overall linear model according to `mathach` $\sim$ `ses` grouped by the different schools.
- Introduce a random intercept for each school and estimate a mixed model using the function `lme()`. Compare the results with your findings from (a). Try `compareFits()` and `comparePred()`.
- Plot the confidence intervals of the estimated intercepts and slopes for the different schools.

3 Data Management

Exercise 1 (Spreadsheet-like data)

(a) Load the file `knee.txt` using the command `read.table()`. Compare what happens when using `header = TRUE` and `header = FALSE`.

(b) What class do the single variables have in R? Are the assigned classes reasonable in all cases? Define the classes of `th`, `gen` and `pain` as factor directly when reading in the data.

(c) Additionally rename the variables to *treatment*, *age*, *sex* and *agony*.

(d) Open the data set `knee2.txt` first in a text editor. What do you observe? Load the data into R such that missing values are automatically identified as those.

(e) Do the same with `knee3.txt`. What is the difference in contrast to `knee2.txt`?

(f) Add a column to the data frame `knee2` with the value *old*, if `age` is larger than 40 and *young* otherwise. Save the resulting data set in the following formats:

- `*.Rdata`,
- `*.raw` without quotes and with : as a separator,
- `*.dat` without column or row names,
- `*.csv`.

For the latter compare the results from `write.table`, `write.csv` and `write.csv2`.

Exercise 2 (Handling date objects and ordering data in R)
The dataset `movies.csv` contains information on the 10 most successful movies of each for the years 2012-2015.

(a) What is the class of the variables `release` and `end`? Obviously those two variables are the dates of the release and the last day of the movies in cinemas. Use `as.Date()` to transform them into dates.

(b) Add a column to the data frame with the number of days the movies have been shown in cinemas. Add another column with the number of complete weeks.

(c) Add a column with the year of release.

(d) Rank the movies according to the number of weeks the movies have been shown in cinemas.

(e) (optional) Write a piece of code which has as output the name of each year?s most successful movie (according to its gross income).

(f) (optional) Find out on which `weekday()` Carl Friedrich Gauss was born.

Exercise 3 (Converting objects and *.RData-objects)
Load the data stored in `data.Rdata`. Use `ls()` to find out which objects are stored in that file. Convert the list into a vector and the data frame into a matrix. What do you observe? Try to explain what happens.

Exercise 4 (Foreign data structures)

(a) The file `blutdruck.sav` is a SPSS data file. Use the according function from the package foreign to load it into R. What do you need to do to directly create a data frame in R?

(b) Use the function `read.sas7bdat()` from the package `sas7bdat` to read in the SAS file `s05_01.sas7bdat`.

(c) Read in the Stata file `golf.dta` with `read.dta()`.

4 Functions, Debugging & Condition Handling

Exercise 1 (Simple functions and restrictions)
In this exercise, we want to build a function to calculate the binomial coefficient of two integers n, k

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{1}$$

Here, we will treat the binomial coefficient as not defined for negative numbers and non-integers[1] Follow the next steps carefully:

(a) What are the definition and basic properties of the factorial of an integer? Also recall the properties of (1) for two integers.

(b) Write a simple function `my_factorial()` which calculates the factorial of an integer n. Don't add any restrictions on n yet. Also, evaluate `my_factorial` at numbers from $\mathbb{R} \setminus \mathbb{N}$. What do you observe?

(c) Let your function return an error message, if n is not an integer. The built-in function `is.integer()` will not help you in this case (why?). Hence, write your own function which returns `TRUE` for integers and `FALSE` for non-integers. Check if your function fulfills the properties from (a). If not, add necessary exceptions.

(d) Write a function `my_binomial()` to calculate the expression in (1). Make use of your results from (b) and (c). Evaluate the expression `my_binomial(2, 3)`. Is the resulting error message reasonable? Use traceback() to locate where the error occurred.

(e) Finally, add an exception for the case $k > n$ to `my_binomial()`. The function should then return the value 0. All other restrictions should be kept as before.

Exercise 2 (Scope and environments)
For a given vector $x = (x_1, \ldots, x_n)$, the function

[1]Note that there are extensions based on the Γ–function for these cases.

```
running_mean = function(k) {
  1 / k * sum(x[1:k])
}
```

calculates the running mean of order $k \leq n$, i.e.

$$\bar{x}_k = \frac{1}{k} \sum_{j=1}^{k} x_k$$

Guess the output of the following two scenarios. Then, check if your prediction was correct.

(a)
```
x = 1:5
running_mean = function(k) {
1 / k * sum(x[1:k])
}
running_mean(2)
```

(b)
```
x = 1:5
rm(list = ls())
running_mean = function(k) {
1 / k * sum(x[1:k])
}
running_mean(2)
```

(c) Explain what went wrong in (b). Modify the function `running_mean` to make it independent from the environment/workspace.

(d) Add reasonable restrictions on x and k.

Exercise 3 (Closures and environments)
In contrast to problems caused by different environments as in the previous exercise, one can explicitly use these environments.

1. Write a function n.root that allows for generating the family of n-th root functions

$$\sqrt[n]{x} = x^{\frac{1}{n}}$$

   For that purpose write a function within a function, such that x is used in the inner environment and n in the outer environment.

2. Can you use n.root to explicitly calculate the roots of a number?

3. Based on `n.root` write functions which calculate the square and cubic root, respectively.

4. Use `n.root` to calculate the first 10 roots of $x = 500$.

5 Profiling, Performance & Parallelization

Exercise 1 (Loops and if statements) Recall Exercise 2 (c) from the first section.

(a) Use `system.time()` or `proc.time()` to compare the solutions resulting from

   (i) `if ... else` statements within a `for` loop,

   (ii) `ifelse()` within a `for` loop,

   (iii) when calculating the row sums within a `for` loop,

   (iv) `rowSums` as presented in the solution of exercise sheet 1.

Check whether all solutions lead to the same results (see `?identical`). What are your conclusions (if any)?

(b) Install and load the package `microbenchmarking` and read the documentation of the likewise named function.

   (i) Compare again the four approaches from (a).

   (ii) Load the package `ggplot2` and use `autoplot()` to visualize the findings.

Exercise 2 (R specific tricks - implemented functions)

(a) Consider the least squares estimator

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y}$$

from the linear model

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

Load the data from `linear.RData` and compare the computation times of

- using `%*%` and `solve` and
- using `crossprod` whenever possible and inverting $\boldsymbol{X}^\top \boldsymbol{X}$ via a QR decomposition.

(b) Use the function `microbenchmarking` again to compare the computation times of `mean(y)` and `sum(y)/length(y)`.

6 Numerics and Simulation

Exercise 1 (Cancellation)
Use `curve()` to visualize the function

$$f(x) = \frac{1 - \cos(x)}{x^2}$$

(a) between -15 and 15 as well as

(b) between -4×10^{-8} and 4×10^{-8}.

Try to explain what you observe.

Exercise 2 (A Monte-Carlo approximation of π) Consider a circle of radius 1 around $(0, 0)$. The circle consists of the points $(x, y) \in \mathbb{R}$ with the restriction

$$x^2 + y^2 \leq 1 \tag{2}$$

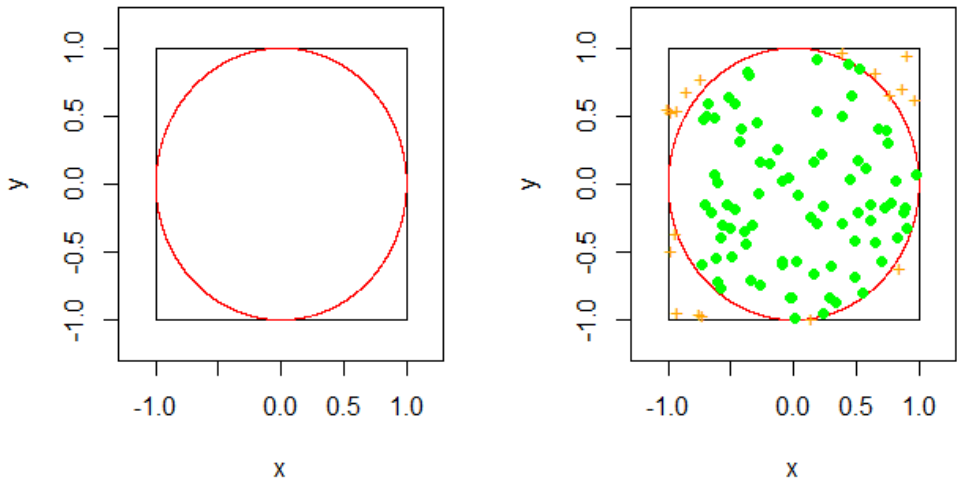
such that its area is exactly π. We use this property to find a suitable approximation of π. Translate the following ideas into R-Code:

- Embed the mentioned circle into a square with vertices $(-1, -1)$, $(1, -1)$, $(1, 1)$ and $(-1, 1)$ (see Figure 1), the ratio of the areas of the circle and the square is

$$R = \frac{\pi}{4} \tag{3}$$

- Approximate R to find an approximation of π.

Figure 1: *Left:* A circle within a square. *Right:* Random points.



- Proceed as follows:
  1. Draw random points from within the square (use `runif()`).
  2. Identify those points that lie in the circle (Equation (1) might be helpful).
  3. Use the proportion of the points from 2. as approximation for R.
  4. Use Equation (2) to approximate π.
  5. Start with $n = 10$ random points and increase the sample size as long as the relative error of the approximation is larger than 10^{-5} (make use of `while`).

How large is the required sample size in your case? Compare your results with those from other course participants.

Exercise 3 (Propagation of round off errors - the Vancouver stock exchange bug)
The Vancouver stock market was established in January 1982 with a starting index of 1000. After 22 month of trade, it decreased to a value of about 525, whereas the expected value was above 1000. What happened?

- After each transaction the index was recalculated
- The value of each transaction was available up to four decimal places
- For the calculation of the new index, only three where used (the fourth was cut off and <u>not</u> rounded)

Approximate the overall error under the following assumptions:

- 2900 transaction per day
- 20 business days per month
- 22 month of trade
- an average error of 0.00045 per transaction

Based on the approximation, what is the true index after those 22 month of trade?