Appendix D

R exercises

Content *Page*

Beginner R exercises

Creating and removing objects in the environment

B 1) Run the following code in R:

```
a = b = 1
a = 2
```

What happened? How do you know (which command)? How many objects did you create in your environment? Try assigning the values using the `<-` operator. Does the result differ from the result when you are using `=` to create the variables?

B 2) Remove the variables `a` and `b` from your environment using the `rm()` function.

B 3) Run the following code in R:

```
apples <- 5; pears <- 3; pineapples <- 6
```

Then try the following code:

```
apples + pineapples
apples + Pears
```

Why can't you add `apples` and `Pears` ?

B 4) Use R to compute the square root of 81 and store the result in `t1` .

B 5) Use R to compute 81 to the power a half and store the result in `t2` .

B 6) Use the `==` operator to check whether the contents of `t1` and `t2` are the same.

Working directory and help functionality

B 7) With the command `getwd()` you can find the current working directory of the R process. Explain what the working directory means and why it is important.

B 8) Change the working directory of the R session to a folder of your preference. What function do you use? How can you check whether your change has worked?

B 9) Run the following code in R:

```
demo()
```

What does this code do? How do you run the demo `persp` (in package `graphics`)?

B 10) Run the following code in R:

```
a <- c(1, 6, 7, 8, 9, NA)
mean(a)
```

This gives `NA`, why? Check `?mean` to find out what arguments the `mean()` function takes as input. Can you find a way to compute the mean with the missing value removed?

B 11) According to Google there exists an R-function called `mvrnorm()`. Typing `mvrnorm` or `?mvrnorm`, however, gives an error. Why? Which library do you have to load first? How?

B 12) What does `%%` do? Can you ask help with `?%%`, like in `?mean`? If not, how? What is `%%` doing?

B 13) The `cor()` function computes correlations. As an example, run the following code in R:

```
cor(c(1, 2, 3, 4), c(1, 4, 7, 15))
```

How can you find out (which function) whether this correlation is statistically significant? How can you find such a function? What is the confidence interval of the correlation? How can you find all functions that have 'cor' in their name?

Data types of objects

B 14) Run the following code in R:

```
a1 <- '1'; a2 <- 1; a3 <- TRUE
```

What are the data types of `a1` , `a2` , and `a3` ? Now run the following code in R:

```
b1 <- c(a1, a2); b2 <- c(a1, a3); b3 <- c(a2, a3); b4 <- c(a1, a2, a3)
```

What are the modes of `b1` , `b2` , `b3` , and `b4` ? Can you explain?

B 15) Why does `TRUE/TRUE` yield 1, `FALSE/TRUE` yield 0, `FALSE/FALSE` yield `NaN` , and `TRUE/FALSE` yields `Inf` ?

B 16) Convert the logical `TRUE` to a numerical. Convert this numerical to a character. Next, convert the logical `TRUE` to a character. Why do these results differ? (Hint: look at as.numeric())

B 17) How can you check whether a vector is numeric? Is the vector `c(1,0)` a numeric vector? And the vector `c(TRUE, FALSE)` ? And the vector `c(TRUE, FALSE, 1, 0)` ? Why? (Hint: look at is.numeric())

Object types: Vectors

B 18) Use the `c()` function (or `:`) to create the following vector:

`-2 -1 0 1 2 3 4 5 6 7 8`

What is the length of this vector?

B 19) Make a vector that starts at -5 and goes to 5 in steps of 0.5. Can you find out more ways to construct this vector than by using the `c()` function?

B 20) Create the following vector:

`13 15 17 19 21 23 25 27 29 31 33`

B 21) Run the following code in R:

```
a <- 1:5
b <- -a
```

Combine the vectors `a` and `b` into one vector.

B 22) What is the result of `rep(2, 10)`? What function argument receives the value 10?

B 23) Use the `rep()` function to create the following vector:

`3 3 3 4 4 4 5 5 5 6 6 6 7 7 7`

B 24) Suppose you want to make the vector `"a" "a" "b" "b" "c" "c" "d" "d" "e" "e"` with the `rep()` function but `rep(letters[1:5], 2)` does not work. What do you have to change in this command?

B 25) The following code gives the first 10 uneven numbers. What is the sum of these numbers?

```
(1:10) * 2 - 1
```

B 26) Make a vector with the first 10 even numbers. Next, make a vector with the first 10 numbers divided by 5. Finally, create the vector `5 8 11 14 17 20 23 26 29 32`.

B 27) `logical(5)` makes a logical vector of length 5. How can you make the same vector with the `vector()` function?

B 28) Run the following code in R:

```
paste(rep(c('a', 'b'), each = 5), 1:5, sep = '.')
```

Next, create the following vector:

`"x1m" "x1f" "x2m" "x2f" "y1m" "y1f" "y2m" "y2f"`

B 29) Run the following code in R:

```
s <- c(5, 7, 2, 8)
```

First sort `s` from highest to lowest. What option in the `sort()` function do you use? Next, sort `s` from lowest to highest. Why don't you have to use the same option here?

B 30) Run the following code in R and explain why the second line gives a warning:

```
1:10 + 1:2 - 1
1:10 + 1:3 - 1
```

Object types: Matrices

B 31) Use the `matrix()` function to create the following matrix:

```
25  24  23  22  21
20  19  18  17  16
15  14  13  12  11
10   9   8   7   6
 5   4   3   2   1
```

B 32) Create the following matrix:

```
0  0  0  0
1  1  1  1
0  0  0  0
1  1  1  1
```

B 33) Run the following code in R:

```
m1 <- matrix(1:20, , 4)
```

Why don't you have to give R the number of rows for the matrix?

B 34) With what function can you transpose a matrix?

B 35) The code `mean(m1)` returns one number. How can you get the mean of each column in `m1` without calculating them each in turn?

B 36) Find out the values of the diagonal in matrix `m1`. Can you make a new matrix of 0's with on the diagonal the diagonal of `m1`?

B 37) Add a new column to the matrix `m1` that contains the sum of the values in each row of the matrix. Use the `rowSums()` function.

B 38) Run the following code in R:

```
m2 <- scale(m1)
```

What are the means of the columns in matrix `m2`? And the standard deviations (hint: use the R function apply)? Given these values, can you find out what the `scale()` function does?

Object types: Data frames

B 39) Create the following data frame without typing out the first two columns:

	subject	time	score
1	1	t1	7
2	1	t2	8
3	2	t1	8
4	2	t2	8
5	3	t1	9
6	3	t2	8
7	4	t1	9
8	4	t2	7
9	5	t1	7
10	5	t2	6

B 40) Run the following code in R:

```
a <- matrix(1:10, , 2, TRUE)
```

Convert the matrix `a` to a data frame.

Object types: Lists

B 41) Create a list consisting of the following entries:

- A numeric vector named `subject` containing a sequence of 1 to 5,
- A character vector named `time` containing the characters `t1` , and `t2` ,
- A numeric matrix named `score` with five rows and five columns, containing the numbers 1 to 25.

Object types: Factors

B 42) Run the following code in R:

```
g <- gl(4, 3)
```

Is `g` a vector? (If not, what is it?) How can you check?

B 43) Run the following code in R:

```
v <- rep(c('m', 'f'), 3)
```

Encode the variable `v` as factor levels. How do you check whether you succeeded?

B 44) Run the following code in R:

```
x <- 0:10%%3
```

Convert `x` to a factor. How many factor levels has `x` ?

Indexing

B 45) Suppose you have `x <- 3` . Now type `x[3] <- 5` . What is the value of `x[2]` ?

B 46) Run the following code in R:

```
a <- c(9, 2, 4, 5, 2, 7, 5)
```

Change the first element into an 8. Next, change the 2's in the vector into zero's.

B 47) Run the following code in R:

```
l <- matrix(c(1, 4, 6, 7, 4, 7), ncol = 3)
```

Use the square brackets to find out the second value of the third column of the matrix `l` .

B 48) Run the following code in R:

```
m <- matrix(sample(1:100, 100, replace = T), 10, 10)
```

Select the third and fifth row of the matrix `m` .

B 49) Select those rows from matrix `m` for which the value in the first column is less than 5.

B 50) Select all but the fourth column of matrix `m` .

B 51) Change the matrix `m` to a data frame using `as.data.frame()` function. Give the columns of the data frame the names `'trial.1'`, `'trial.2'`, etc using the `colnames()` function.

B 52) Select the second to the fifth element of column `trial.1`. Select the first 2 elements of column `trial.4`.

B 53) Run the following code in R:

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

Suppose you want to change all ones into twos and all twos to ones. To try out, run the following code in R. It doesn't work. Can you find a method so that this is done correctly?

```
b[b==1] <- 2
b[b==2] <- 1
```

B 54) Run the following code in R:

```
n <- c('bananas', 'apples')
```

Use the `gsub()` function to change all a's in `n` to dots (e.g., bananas becomes b.n.n.s).

B 55) Run the following code in R:

```
set.seed(1)
grades <- data.frame(1:30, matrix(sample(4:10, 60, TRUE), , 2))
names(grades) <- c('student', 'exer', 'exam')
```

Students pass a course when the average grade is at least 5.5 and both grades are larger than 5. Select (index) which students passed the course. Also find out how can you select which students did not pass the course.

Conditions

B 56) What does `(a < 0 | b < 0) & (a * b < 0)` mean (`|` and `&` are the logical 'or' and 'and' operator, respectively)? If this condition is `TRUE` and `a` is negative, what do you know about `b` ?

B 57) Why does `!sum(is.na(c(1, 2, 3, 4, NA, 7))) <= 2` return `FALSE` ?

B 58) Run the following code in R and explain the result:

```
x <- 1:24
which(24%%x == 0)
```

B 59) Run the following code in R:

```
x <- y <- 1:10
sum(x==y) == length(x==y)

x <- y <- 1:10
y[1] <- 0
sum(x==y) == length(x==y)
```

Explain the result of this code.

Sampling and simulating data

B 60) You can throw a die 100 times by typing the following code in R:

```
throws <- sample(1:6, 100, TRUE)
```

How can you see how often each number shows up? How often did each number show up?

B 61) Sample, with replacement, 4 cards from the set `'jack'`, `'queen'`, `'king'` and `'ace'` with equal probabilities.

B 62) Create 20 uniform random numbers between 0 and 100. Why is `sample(1:100, 20)` not correct?

B 63) Put 21 uniform random numbers in a vector. What is the median of this vector? Sort the vector from highest to lowest. What is the 11th element of the sorted vector?

B 64) Create 100 normally distributed numbers with `mean = 100` and `sd = 15`. What is the variance of these numbers? Run the code that you used for this again. Why is the variance not exactly the same the second time? What can you use so that, if you run your code again, the results are the same?

B 65) Run the following code in R:

```
x <- matrix(rnorm(100), 25, 4)
```

Find out the covariance matrix of matrix `x`. Also find out the correlation matrix of matrix `x`.

Reading and writing data

B 66) What does the `read.csv()` function do? How would you read in Excel files? What package contains the function `read.spss()`? Can you find other file types that R has read functions for?

B 67) Read in the file `example.csv` from the online resources using the `read.csv()` function. How many observations does this data contain?

B 68) Run the following code in R:

```
d <- data.frame(sex = c('m', 'f'),
                Age = c(6.7, 6.5, 5.6, 5.4),
                var1 = c(9, 5, 4, 4),
                var2 = c(10, 5, 8, 4))
```

Write the data frame `d` to a file named `example.xlsx` so that Excel is able to open it. What options did you use?

Advanced R exercises

A 1) *Let's sort*

Difficulty: Easy

Sorting is easy in R. You can just use `sort(x)` . Let's try to build your own sort function. One straightforward algorithm is called bubble-sort, check the 'Bubblesort' entry on Wikipedia for more information (https://en.wikipedia.org/wiki/Bubble_sort).

☐ Create a function that can sort a vector using the bubble-sort approach.

A 2) *T-test*

Difficulty: Easy

R had a built-in function `t.test()` that performs a t-test. Read the Wikipedia entry for the T-test (https://en.wikipedia.org/wiki/Student%27s_t-test) to find out exactly what is going on in this function.

☐ Program a function that can be used to perform a t-test. Think about what results the user would like to get and how the user should use the function.

A 3) *Mann-Whitney U test*

Difficulty: Easy

Read the Wikipedia entry for the Mann-Whitney U Test for some first information (https://en.wikipedia.org/wiki/Mann%E2%80%93Whitney_U_test). The Mann-Whitney U test is the non-parametric equivalent of the two sample t-test and is used when the assumptions of the parametric t-test are violated.

☐ Program your own Mann-Whitney U function and compare it to the built-in Mann-Whitney U test in R to see whether you get the same results.

A 4) *Caesar cipher*

Difficulty: Medium

For some initial information, read the Wikipedia entry on the Caesar cipher encryption/decryption method (https://en.wikipedia.org/wiki/Caesar_cipher).

☐ Program a function that can be used to encrypt a character string using the Caesar cipher and program another function that can be used to decrypt a character string using the Caesar cipher.

☐ Use the decrypt function to decrypt the following string that is encrypted using a Caesar cipher and a key of 13:

```
Nznmvat jbex! Lbh qrpelcgrq guvf zrffntr pbeerpgyl.
```

A 5) *Infinite monkeys, infinite typewriters*

Difficulty: Medium

The infinite monkey theorem says that a monkey hitting keys at random on keyboard for an infinite amount of time will almost surely type any given text, such as the entire contents of this workbook. In fact, the monkey would almost surely type every possible finite text an infinite number of times. For further information, read the Wikipedia entry for the Infinite Monkey Theorem (https://en.wikipedia.org/wiki/Infinite_monkey_theorem).

- ☐ Simulate one monkey, typing random letters on a typewriter in sequences of 5 letters (so, typing 5 letters, then a space, and so on). Turn this into a function that returns the number of letters typed before making a coherent 5-letter word. Use a file that contains all English 5 letter words to check whether a word is valid (e.g., http://www-cs-faculty.stanford.edu/~knuth/sgb-words.txt).

- ☐ Run the function 500 times and make a nice plot with the results (WARNING: a 500 times will take approximately 30 minutes, so start with a much smaller number for testing your code).

A 6) *Hangman*

Difficulty: Hard

Hangman is a paper and pencil guessing game for two or more players. One player thinks of a word, phrase or sentence and the other(s) tries to guess it by suggesting letters within a certain number of guesses. Read the Wikipedia entry on "Hangman" for some information about the rules of the game (https://en.wikipedia.org/wiki/Hangman_%28game%29).

- ☐ Play Hangman against the computer by programming the game in R. For an additional challenge, try to include graphics for wrong answers.

A 7) *Blackjack*

Difficulty: Hard

Blackjack is a casino card game between one or more players and a dealer, where each player in turn competes against the dealer. Read the Wikipedia entry for the "Blackjack" casino game for some more information about the rules of the game (https://en.wikipedia.org/wiki/Blackjack).

- ☐ Program the Blackjack game. For an additional challenge, try to include a betting system.