Base R exercises

by courtesy of
Kerstin Pierick

August 28, 2019

1 Exercise 1: Make a new project in RStudio and save a script in it

(a) Click on the blue Symbol in the upper left-hand corner in Rstudio

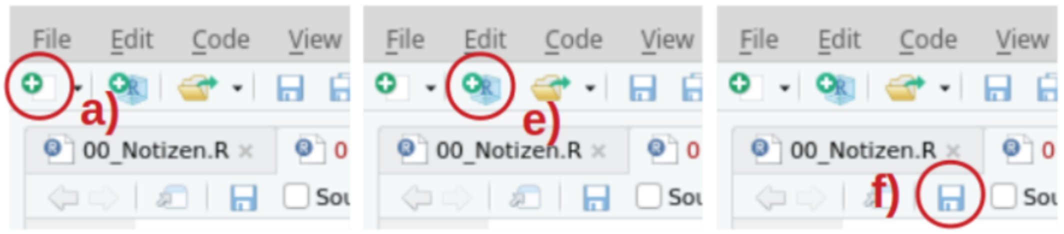
(b) Select "New Directory"

(c) Select "New Project"

(d) Give the project a name ("Directory name") and choose a location for it on your PC (you can use the "Browse.." – button and choose the location manually). Click on "Create Project" and Rstudio should automatically switch to the new directory.

(e) To make a new script, click on the white symbol in the upper left-hand corner and select "Rscript"

(f) To save the script in the new project directory, click on the floppy disk symbol at the top of the script-window. Solve the exercises below and save them in this script, so you can come back to them in the future.



2 Vectors

Solve these exercises by making use of the Base R cheat-sheet. There are often multiple ways of solving them.

(a) Create the following vectors. Give them the names a-e (for exercises further down the line)

- a: 1 2 3 4 5 6 7
- b 2 5 3 9 12
- c: 1 1 1 2 2 2 3 3 3
- d: 5 6 7 5 6 7
- e: 2 4 6 8 10

(b) Sort vector b

(c) Create a vector which contains the each of c's different values exactly once

(d) Display the 2nd and 3rd value of vector e

(e) Display all values of vector d, except for the 3rd and 5th

(f) Display all values of vector a greater than 4

3 Calculations on vectors

Look for the ”Math Functions” - area on the Base R cheat-sheet to solve these exercises.

(a) Which one’s the largest value of vector e?

(b) What’s the median of vector a?

(c) Calculate the sum of the natural logarithms of all elements of vector b

(d) Calculate the logarithm of the second element of c and round the value to the second decimal place

4 Data types

You’ve seen only one data type so far: numeric data. Numeric data are real numbers and you can use them in mathematical calculations. Subtypes include integer (numbers without fractional components) or double (numbers with decimals). There’s an array of other types and subtypes, but the most common are ’character’ and ’logical’.

Data type	Example	Description
character	”a”, ”23”, ”Hello world”	Any letters or numbers (also called ’strings’). Always in quotes.
logical	TRUE, FALSE	Usually needed in conditions. 1 >3 is FALSE, 3 >1 is TRUE
numeric	2, -4, 0.34	Any numbers you can do calculations with (i.e. integer or double)

A vector can only have elements of the same type. R automatically ”decides” how it interprets the data type of a vector (”coercion”).

(a) Create the following vectors:

- f: TRUE, FALSE, TRUE
- g: ”Apple”, ”Pear”, ”Mango”
- h: 5, 6, ”Fruit salad”
- i: 3, 5, FALSE, TRUE
- j: ”A”, FALSE, TRUE

(b) The functions typeof(), is. character(), is.logical(), is.numeric() can tell you the data type of an object. How does R decide how it interprets the datatype of these vectors?

(c) Data types of an object can be changed with the functions as.numeric(), as.logical(), or as.character(). Change...

- Vector f to a numeric and a character vector
- Vector i to a character and a logical vector
- Vector h to a numeric and a logical vector

(d) Missing values are displayed by R as NA (not available). Try out the following code:

```
x <- NA
# is.na() can tell you if a value is NA
is.na(x)
# a vector can have any number of NAs
y <- c(5,6,NA,7,NA)
is.na(y)
# if you try to do maths on NAs, the result will also be NA
y + 5
mean(y)
# One possibility to circumvent the problem is to filter out all NAs
y[!is.na(y)] + 5 # the ! means "not"
# many functions, have an option to ignore NAs: na.rm (rm is short for remove)
mean(y, na.rm = TRUE)
```

5 Data Frames

The most common way of storing, modifying or analyzing data in R are data frames. Dataframes are two dimensional tables, where every row represents an observation and every column a variable. Each row and each column are made up of vectors. The name of a vector is the name of a column in a dataframe. A dataframe can contain multiple data types, but the data type has to be consistent within a single column. All columns have to have the same length. Try out the following code:

```
# Have a look at Base R cheatsheet - Data Frames
# Data frames are created with the data.frame() function
df1 <- data.frame(colour = c("yellow", "red", "blue"), count = 1:3)
df1
# You can also insert already existing vectors
df2 <- data.frame(f,g)
df2
# Important: Vectors have to be the same length
df3 <- data.frame(e,f)
# cbind() can append two dataframes by column
df4 <- cbind(df1,df2)
df4
# How to select specific elements of a dataframe
# selecting a whole row
df[2,]
# selecting a whole column
df[,2]
# selecting a specific observation
df[2,2]
# selecting a whole column by its name
df$farbe
```

(a) Load the dataset "iris" with data("iris"). Now you should be able to access the data with the name iris. Use the Base R cheat-sheet to solve the following exercises.

(b) How many rows and columns does iris have

(c) What's in column 3, row 30?

(d) What's the mean of Sepal.Length?

(e) What's the datatype of Sepal.Length?

6 Reading in your own data

Most of the time, you don't create your own raw data in R itself, but you get them from external resources or programs and bring them into R for further analyses. The best file format for this is .csv. Watch out for regional differences in decimal and field separators! You can read in a .csv by using the read.csv() function (check out the cheat sheet for more information).

7 Exporting data

Create a small dataset of your choice in R and export it with the function write.csv() (check out the cheat sheet for more information).

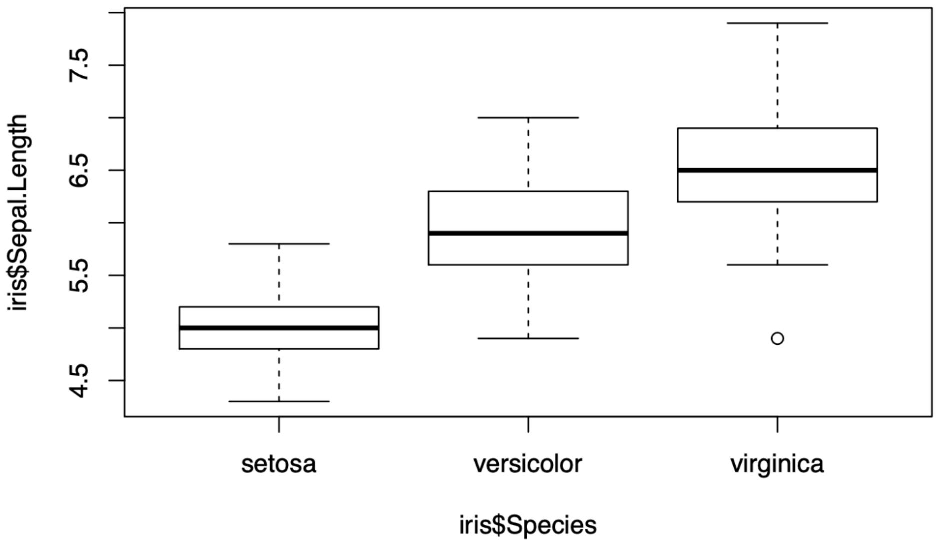
8 Plotting

Create plots on the iris dataset:

(a) Use plot() to create a scatterplot of Petal.Length (x-axis) and Petal.Width(y-axis)

(b) Use histogram() to make a histogram of Sepal.Length

(c) Try to reproduce the boxplot below. The relevant function is not on the cheat sheet, so try to find the solution using a search engine on the internet.


iris$Sepal.Length
setosa
versicolor
virginica
iris$Species