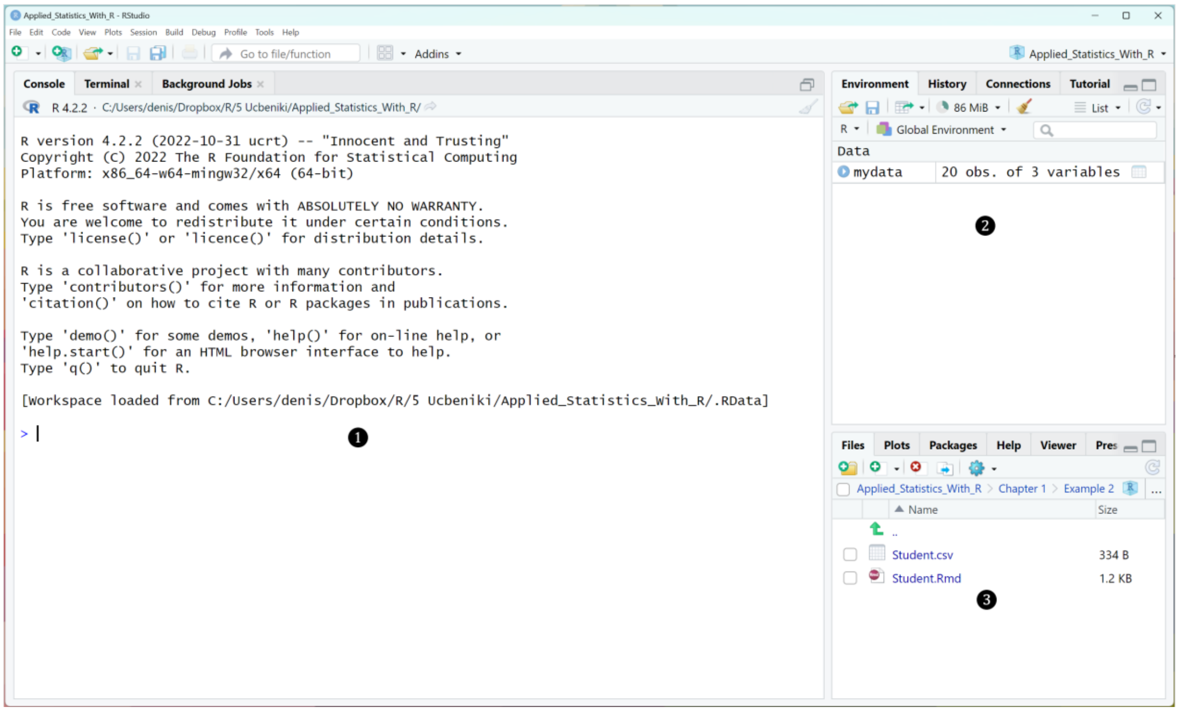
1 BRIEF INTRODUCTION TO THE PROGRAM R

In the textbook we use the widely used free statistical program R. It works on the principle of written commands that execute functions contained in various libraries and used in the preparation and statistical processing of data. Similar statistical processing of data can be done with different functions, so in the examples we will show different ways to increase the reader's knowledge of the set of important functions that allow statistical processing of data in the field of economics and social sciences.

In addition to the R program[1], we will also use an interface program to facilitate its use. In this textbook, we will use the easy-to-use, free interface called RStudio[2], which can be used in a Windows, macOS, or Linux environment. Alternatively, RStudio can be used in the cloud (RStudio Cloud[3]), where the program does not need to be installed on a computer, but all statistical analysis is performed through a web browser and a user account on the RStudio Cloud website. The disadvantage of this approach is that the user is limited by the number of free monthly usage hours. Regardless of the usage approach chosen, the RStudio interface is identical (Figure 1).

Figure 1: RStudio interface



Applied_Statistics_With_R - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Applied_Statistics_With_R
Console
Terminal
Background Jobs
R 4.2.2 · C:/Users/denis/Dropbox/R/5 Ucbeniki/Applied_Statistics_With_R/
Environment
History
Connections
Tutorial
86 MiB
List
R
Global Environment
Files
Plots
Packages
Help
Viewer
Pres
Applied_Statistics_With_R > Chapter 1 > Example 2
Name
Size
1
2
3

```
R version 4.2.2 (2022-10-31 ucrt) -- "Innocent and Trusting"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from C:/Users/denis/Dropbox/R/5 Ucbeniki/Applied_Statistics_With_R/.RData]

> |
```

```
Data
mydata     20 obs. of 3 variables
```

	Name	Size
	..	
☐	Student.csv	334 B
☐	Student.Rmd	1.2 KB

RStudio consists of three windows. On the left is the Console window ❶, where we write the R commands that we use to statistically process the data. On the top right is the Environment window ❷, which lists the currently available data tables that RStudio can access directly. On the bottom right ❸, files, saved in computer, are shown (if we use Cloud version, files need to be uploaded to the server).

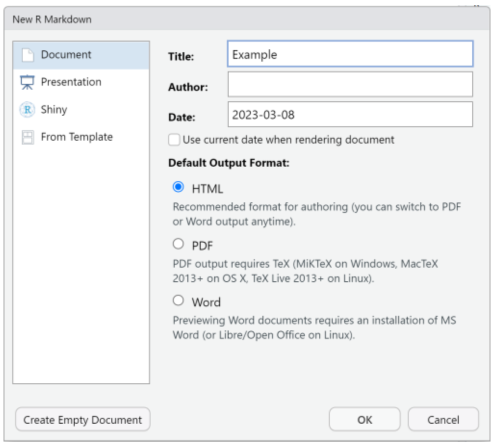
Although we can write R commands directly to the console, this is not practical approach because it is easy to make mistakes. A better choice is to use an R Markdown file (extension .Rmd), which

[1] https://cran.r-project.org/
[2] https://posit.co/download/rstudio-desktop/
[3] https://rstudio.cloud/

makes it easy to write and edit commands and then generate a statistical analysis report. Important advantages are reproducibility of statistical analysis and sharing the R code. Such R Markdown file is created using the File - New File - R Markdown function within the RStudio interface.

When we use it for the first time, we need to install certain libraries, which RStudio does automatically (it asks us if we want to install them - when RStudio tells us that some libraries are missing, we agree to install them). We name the R Markdown and choose OK.

Figure 2: Creating new R Markdown document in RStudio



New R Markdown

Document
Presentation
Shiny
From Template

Title: Example
Author:
Date: 2023-03-08

Use current date when rendering document

Default Output Format:

● HTML
Recommended format for authoring (you can switch to PDF or Word output anytime).

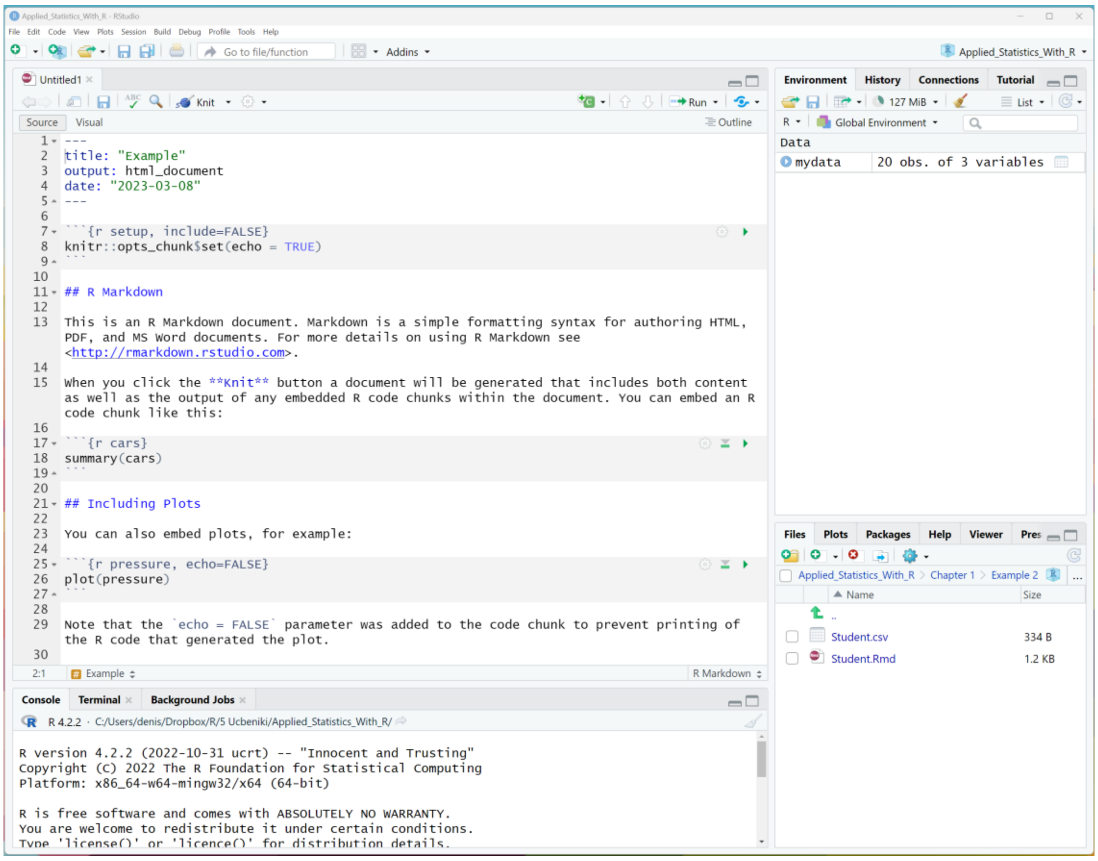
○ PDF
PDF output requires TeX (MiKTeX on Windows, MacTeX 2013+ on OS X, TeX Live 2013+ on Linux).

○ Word
Previewing Word documents requires an installation of MS Word (or Libre/Open Office on Linux).

Create Empty Document   OK   Cancel

The window shown in Figure 3 will open. It already contains some examples of R commands that can be executed with the test data included in the R environment, so for the new analysis we delete the records from the file starting from line 11.

Figure 3: An example of an R Markdown document in RStudio



```
 1  ---
 2  title: "Example"
 3  output: html_document
 4  date: "2023-03-08"
 5  ---
 6
 7  ```{r setup, include=FALSE}
 8  knitr::opts_chunk$set(echo = TRUE)
 9  ```
10
11  ## R Markdown
12
13  This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML,
    PDF, and MS Word documents. For more details on using R Markdown see
    <http://rmarkdown.rstudio.com>.
14
15  When you click the **Knit** button a document will be generated that includes both content
    as well as the output of any embedded R code chunks within the document. You can embed an R
    code chunk like this:
16
17  ```{r cars}
18  summary(cars)
19  ```
20
21  ## Including Plots
22
23  You can also embed plots, for example:
24
25  ```{r pressure, echo=FALSE}
26  plot(pressure)
27  ```
28
29  Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of
    the R code that generated the plot.
30
```

2:1 Example   R Markdown

Data
mydata   20 obs. of 3 variables

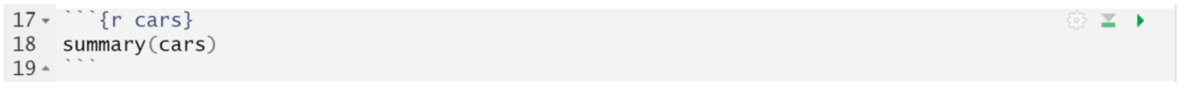
Applied_Statistics_With_R > Chapter 1 > Example 2

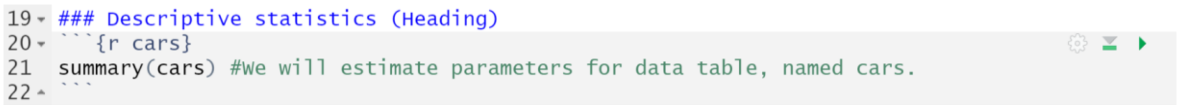
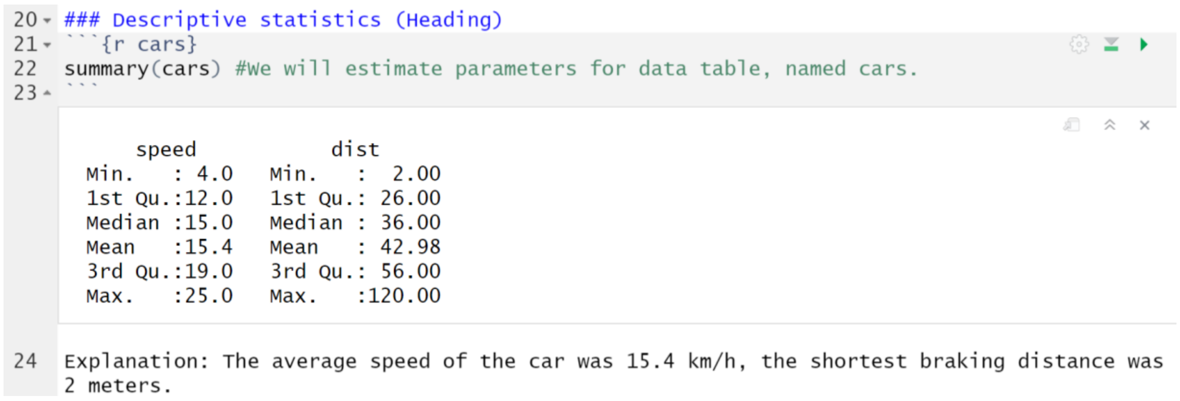
Name	Size
..	
Student.csv	334 B
Student.Rmd	1.2 KB

R 4.2.2 · C:/Users/denis/Dropbox/R/5 Ucbeniki/Applied_Statistics_With_R/

```
R version 4.2.2 (2022-10-31 ucrt) -- "Innocent and Trusting"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
```

An R Markdown file consists of white and gray boxes, the latter being called an R chunk. In the gray box we write R commands, and by default the results are displayed below the gray box (chunk output inline), but in the preferences we can also change that the results are displayed in the console (chunk output in console). Each R chunk starts with the tag ```{r} and ends with the tag ```, and between the tags we write the R command. Let us give an example.

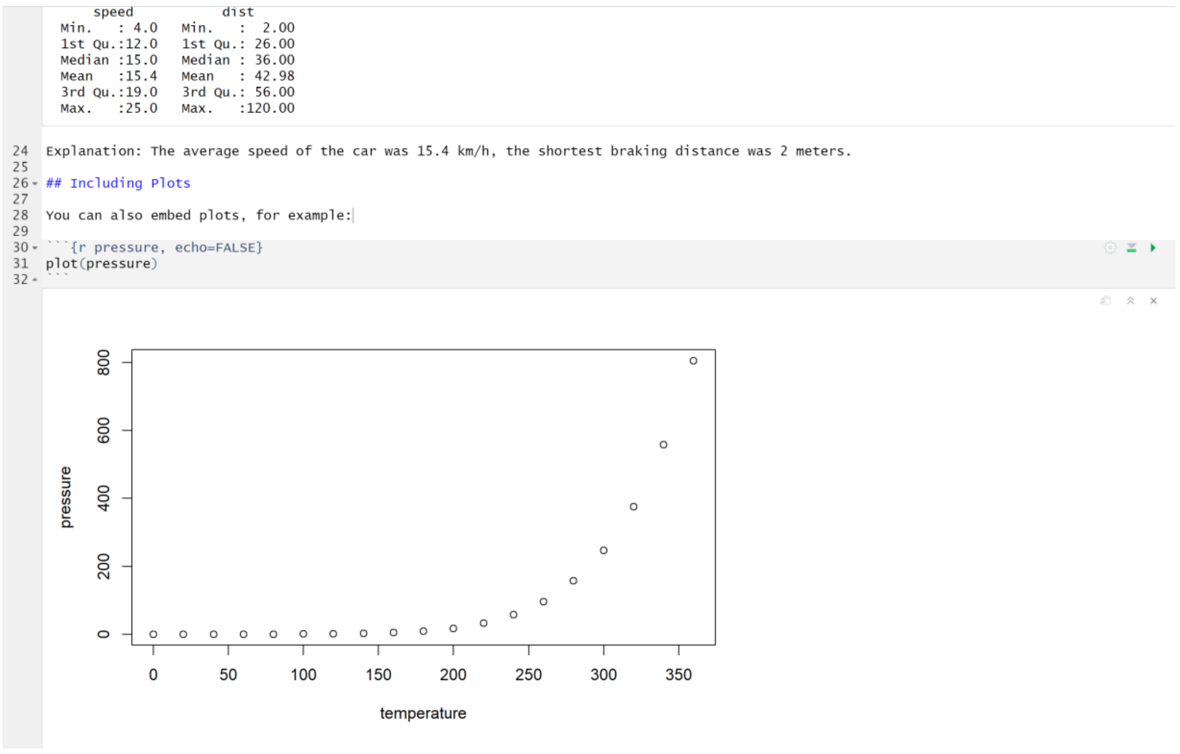


~~~
```{r cars}
summary(cars)
~~~
```

If we want to enter a comment in the gray field, we must prefix it with #, otherwise the program will treat it as an R command. In the white fields you can write title, content text, explanations of the results, etc. Let us give an example.



```
### Descriptive statistics (Heading)
```{r cars}
summary(cars) #We will estimate parameters for data table, named cars.
```
```

We create a new gray box by standing in the white box where we want to write a new R command, and then selecting Insert → R (keyboard shortcut CTRL + ALT + i). A new R chunk appears, into which we write the command. We execute the single R chunk by clicking the green arrow on the right side of the gray box, or by selecting the corresponding lines with the mouse and pressing the keys CTRL + Enter on the keyboard. Example: If we run the gray box with the `summary(cars)` command, the white box will display the statistical parameter estimates for the testing dataset called cars contained in the R environment. We will also explain the results in white box.



```
### Descriptive statistics (Heading)
```{r cars}
summary(cars) #We will estimate parameters for data table, named cars.
```

     speed   dist
 Min.   : 4.0   Min.   : 2.00
 1st Qu.:12.0   1st Qu.: 26.00
 Median :15.0   Median : 36.00
 Mean   :15.4   Mean   : 42.98
 3rd Qu.:19.0   3rd Qu.: 56.00
 Max.   :25.0   Max.   :120.00

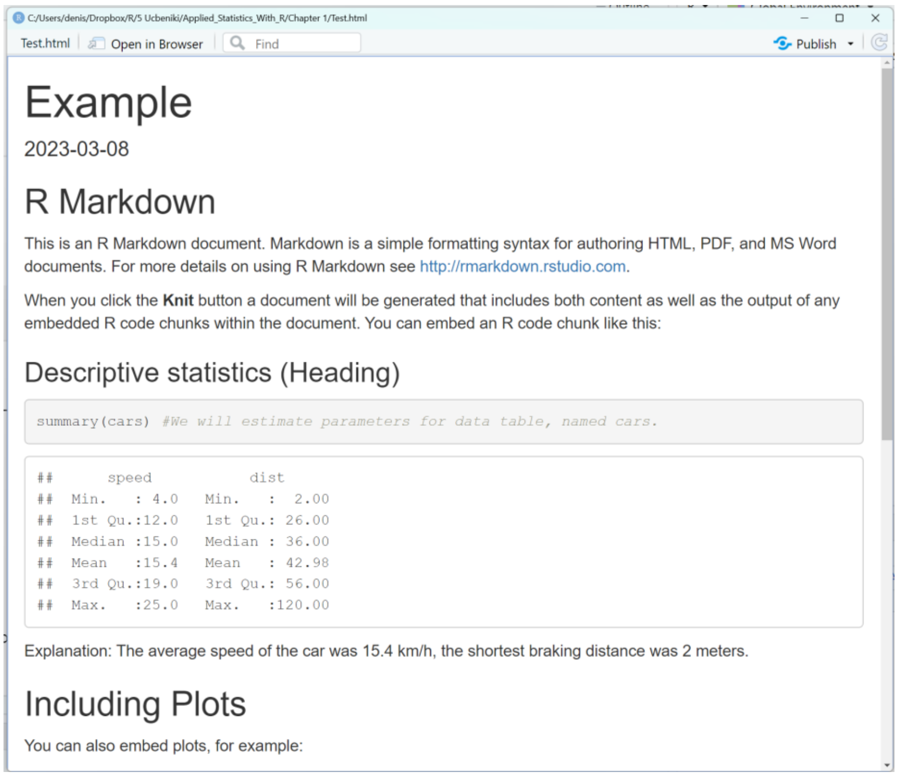
Explanation: The average speed of the car was 15.4 km/h, the shortest braking distance was
2 meters.
```

When we run the command `plot(pressure)`, the test plot is drawn (see Figure 4). When we prepare the final R Markdown document, we select the Knit command at the top of the R Markdown document. The entire document will be translated and executed, and if everything is written correctly, an HTML window will open with the results (it is also possible to get the results in a PDF and Word format). The program will ask us to save the R markdown document and then it will start translating it. If an error occurs in any part of the R command set, RStudio informs us about it and writes down in which line the error is. If the commands are executed correctly, an HTML window opens with the results of the statistical analysis. The content can be open in Web Browser and freely copied or even published online using the function Publish (RPubs) (Figure 5).
```

*Figure 4:* An example of R markdown



~~~
     speed           dist       
 Min.   : 4.0   Min.   :  2.00  
 1st Qu.:12.0   1st Qu.: 26.00  
 Median :15.0   Median : 36.00  
 Mean   :15.4   Mean   : 42.98  
 3rd Qu.:19.0   3rd Qu.: 56.00  
 Max.   :25.0   Max.   :120.00  

Explanation: The average speed of the car was 15.4 km/h, the shortest braking distance was 2 meters.

## Including Plots

You can also embed plots, for example:

```{r pressure, echo=FALSE}
plot(pressure)
```
~~~

Figure 5: An HTML window with the results obtained with R Markdown and the Knit command



Example

2023-03-08

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

Descriptive statistics (Heading)

```
summary(cars)  #We will estimate parameters for data table, named cars.
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

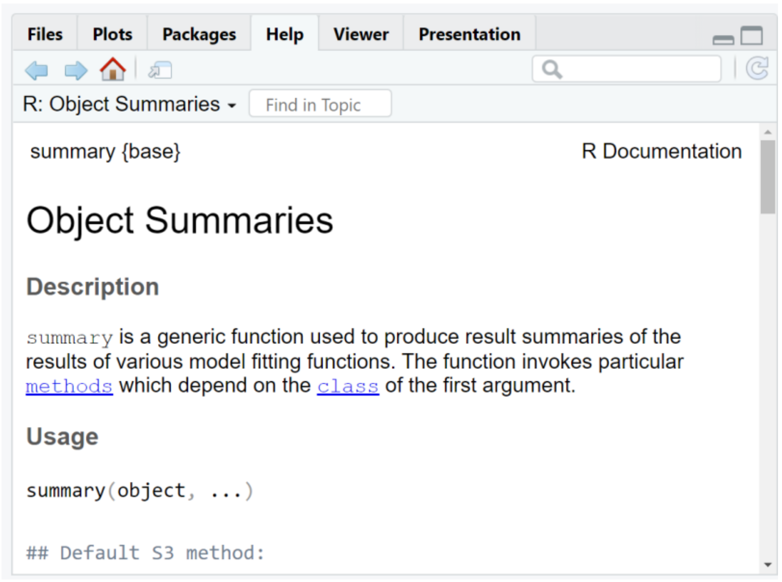
Explanation: The average speed of the car was 15.4 km/h, the shortest braking distance was 2 meters.

Including Plots

You can also embed plots, for example:

When using R functions (e.g. `summary`, `plot`, etc.), you can use the Help function, which explains how the functions must be used. The help command is activated by selecting the name of the function written in R Markdown and pressing the F1 key on the keyboard. In the right bottom part of RStudio, a description of the function, the name of the library in which the function is located (in curly brackets `{}`), and how the function is used are displayed (see Figure 6). Alternatively, we can search the Web for examples how the command for each function is property written.

Figure 6: Help for function `summary`



summary {base} R Documentation

Object Summaries

Description

`summary` is a generic function used to produce result summaries of the results of various model fitting functions. The function invokes particular `methods` which depend on the `class` of the first argument.

Usage

```
summary(object, ...)

## Default S3 method:
```

When installing the R programme, the most important libraries are installed automatically (for example, the `{base}` library), but many useful libraries are unfortunately not included, but must be installed the first time they are used in RStudio. The next time the program is used, the installation is no longer necessary, but library must be activated. The installed libraries can be checked on the Packages tab in the bottom right part of RStudio (currently active libraries are marked with a check mark). The libraries loaded in the User Library are the libraries we installed additionally, and the libraries installed in the System Library are the base libraries we got by installing the R program. For certain libraries, RStudio itself informs us that they need to be installed. The library is installed with the command:

```
install.packages("library_name")
```

The name of the library must be enclosed in quotation marks and it is case sensitive. When we need a specific library that is currently not active, we activate it with the command:

```
library(library_name)
```

The name of the library is given without quotation marks and it is case sensitive. This command activates the library and we can use the functions it contains.

Throughout the book, many data files and R codes will be used. All are available for download on the Textbook website. There will be three types of files:

- file with the extension .Rmd: An R Markdown file containing R commands and comments;
- file with the extension .html: output with results made with the Knit function;
- file with extension .csv: data file in .csv format;
- file with extension .xlsx: data file in .xlsx format.

At the end of the introduction, it should also be mentioned that although the use of the software package R may be difficult at the beginning, the user learns it very quickly and it is quite comparable to other statistical programs in terms of difficulty, and at the same time it is open source and free. Because of the widespread use of R, if we encounter a problem, we can simply use a web browser and find out very quickly what is wrong with our code.

In the following pages, two illustrations are given about basic data manipulation and analysis with R Studio.

1.1 Illustration 1: Manual data entry

We have data of Age and Gender for 6 people. We would like to input the data into the R program.

ID	Age	Gender
1	20	Male
2	25	Female
3	21	Female
4	18	Male
5	24	Male
6	23	Female

We will use the `data.frame` function, which allows data to be entered manually. Non-numeric variables such as gender will be entered with numbers. Let us decide to code Male with number 1 and Female with number 2. The data is stored in a data frame called mydata. With the `<-` tag we assign the right standing command to the left standing object. It is very similar to the tag = but `<-` is usually used.

#Assigning the function data.frame to the object called mydata.[4]

```
mydata <- data.frame("ID" = c(1, 2, 3, 4, 5, 6),
                     "Age" = c(20, 25, 21, 18, 24, 23),
                     "Gender" = c(1, 2, 2, 1, 1, 2))

print(mydata)  #Showing the data set
```

```
##   ID Age Gender
## 1  1  20      1
## 2  2  25      2
## 3  3  21      2
## 4  4  18      1
## 5  5  24      1
## 6  6  23      2
```

We created 3 variables (ID, Age and Gender) and store them in a data frame (name of the table with the data in the program R). The variables are represented with a vector of 6 values entered with the `c()` command. Each time we are entering more than only one value, this command must be used.

[4] Text in gray boxes represents R commands in Courier New font. Text in Times New Roman italic font beginning with # represents the explanation of the executed R command. The gray boxes are followed by a white box with the results of the executed command.

The names of the variables must be written with quotation marks. The function `print()` shows the data frame.

We will do some basic data manipulation. We create a new data frame called mydata2 that contains only the variables ID and Age and excludes the person ID 4. We perform the data manipulation using the `mydata[ , ]` command. The values before the comma refer to the rows of the data frame and the values after the comma refer to the columns of the data frame.

```
mydata2 <- mydata[-4 , c(1, 2)]  #Exclusion of the 4th row and selection of the 1st and the
                                  2nd column.
print(mydata2)
##   ID Age
## 1  1  20
## 2  2  25
## 3  3  21
## 5  5  24
## 6  6  23
```

Let us assume that the student ID 1 is 22 years old (and not 20). We can change individual values in the data frame by specifying the position of the value we want to change.

```
mydata[1, 2] <- 22
print(mydata)
##   ID Age Gender
## 1  1  22      1
## 2  2  25      2
## 3  3  21      2
## 4  4  18      1
## 5  5  24      1
## 6  6  23      2
```

We would like to add to existing data frame `mydata` a new variable Height. Let us assume that heights of 6 people are: 180, 173, 172, 178, 182, 169. We refer to variables in data frame with command `dataframe$variable` (variable, which is saved in dataframe). We can call individual variable by `name_of_dataframe + $ + name_of_variable`.

```
mydata$Height <- c(180, 172, 172, 178, 182, 169)  #Adding a new variable Height to
                                                  exisiting data frame, named mydata.

print(mydata)
##   ID Age Gender Height
## 1  1  22      1    180
## 2  2  25      2    172
## 3  3  21      2    172
## 4  4  18      1    178
## 5  5  24      1    182
## 6  6  23      2    169
```

For categorical variables coded with values, it is advisable to use the `factor` function, which converts the variable into a factor (categorical variable).

```
#Creating new variable, named GenderF, which is factor (categorical variables) with categories 1 and 2
```

```
where 1 means Male and 2 Female.

mydata$GenderF <- factor(mydata$Gender,
                         levels = c(1, 2),
                         labels = c("M", "F"))

print(mydata)
```

```
##   ID Age Gender Height GenderF
## 1  1  22      1    180       M
## 2  2  25      2    172       F
## 3  3  21      2    172       F
## 4  4  18      1    178       M
## 5  5  24      1    182       M
## 6  6  23      2    169       F
```

1.2 Illustration 2: Reading .csv data file

In program R Studio we open the file Rading data.Rmd, which is the R Markdown file. It includes R code for analysis of the dataset named Reading csv data.csv.

```
knitr::opts_chunk$set(echo = TRUE)
options(width = 70)  #Width of the html output
```

First R chunk is setup chunk which allows us to set the form of results output. If we want to copy results to word document, width of 70 is advised to be used. In the next R chunk we will read the .csv datafile. R function `read.table()` is used to read .csv datafiles. If .csv data file is in the same folder as .Rmd file, used to analyze that data file, we just write `./` and name of the data file. Otherwise the full path of the data file must be written.

```
#Data import
mydata <- read.table("./Reading csv data.csv", #The name of the file which the data
                                                  are to be read from
                     header = TRUE, #A logical value indicating whether the file contains
                                      the names of the variables as its first line
                     sep = ";", #The field separator character
                     dec = ",") #The character used in the file for decimal points
```

We will show the first 6 rows of data frame, called mydata.

```
#Showing first 6 rows of dataframe
head(mydata)
```

```
##   ID Height Weight
## 1  1  179.0     70
## 2  2  178.0     68
## 3  3  174.0     64
## 4  4  174.0     63
## 5  5  173.5     61
## 6  6  173.0     60
```

Description of variables:

- ID of a person

- Height in cm
- Weight in kg

In the next step we will order people (units, i.e. rows of data frame) by their height.

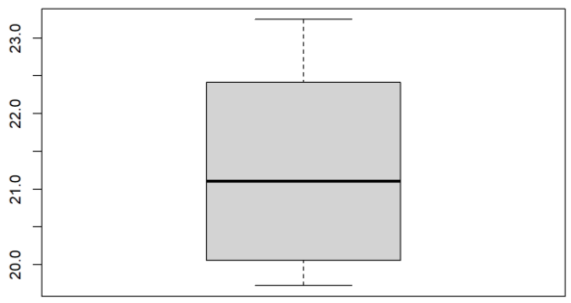
```
#Ordering units by height
head(mydata[order(mydata$Height), ])
##    ID Height Weight
## 20 20    166     55
## 8   8    168     57
## 9   9    168     56
## 7   7    170     57
## 18 18    170     58
## 19 19    170     57
```

In the next step we would like to calculate body mass index, which is defined as weights of a person in kg, divided by squared height in meters.

```
mydata$Height_m <- mydata$Height / 100  #Calculating new variable Height in meters and
                                          saving it to the existing data frame

mydata$BMI <- mydata$Weight / mydata$Height_m^2  #Calculating new variable Body
                                                   mass index

head(mydata)
##   ID Height Weight Height_m      BMI
## 1  1  179.0     70    1.790 21.84701
## 2  2  178.0     68    1.780 21.46194
## 3  3  174.0     64    1.740 21.13886
## 4  4  174.0     63    1.740 20.80856
## 5  5  173.5     61    1.735 20.26427
## 6  6  173.0     60    1.730 20.04745
```

Lastly, we will show descriptive statistics and boxplot for newly created variable BMI.

```
summary(mydata$BMI)  #Descriptive statistics of variable BMI
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   19.72   20.06   21.10   21.20   22.35   23.25

boxplot(mydata$BMI)  #Showing boxplot for variable BMI
```



1.3 Illustration 3: Reading .xlsx data file

In this illustration we will assume that randomly assigned to 3 groups, each group receiving one type of medicine (A, B or C), which have a negative side effect of severe headaches. We would like to analyze if there are difference in average number of headaches between receiving medicine A, B or C.

We have dataset in .xlsx format, which can be read with function `read_xlsx`. This function demands specific library, called `readxl`. Since this library is not installed in R package by default, we need to install it with function `install.packages`. Once library is installed on computer, it only needs to be activated with function `library`. To prevent running the function `install.packages` every time, we used `#` in front of the function, which prevents running it. When importing .xlsx data, it is advised to convert it to typical data frame format, which is done with function `as.data.frame`.

```
#install.packages("readxl")  #Installing library. Once it is installed, we use # to prevent
                              reinstalling

library(readxl)  #Activating library

mydata <- read_xlsx("./Reading xlsx data.xlsx")  #Reading data set in xlsx format

mydata <- as.data.frame(mydata)  #Creating data frame format

head(mydata)
##   ID Headache Medicine
## 1  1       59        A
## 2  2       50        A
## 3  3       45        A
## 4  4       55        A
## 5  5       35        A
## 6  6       28        A
```

Description:

- ID of a person
- Headache: number of headaches over 1 year period
- Medicine: Type of medicine (A, B or C)

Since variable Medicine is nonnumeric, we will convert it to factor.

```
mydata$MedicineF <- factor(mydata$Medicine,
                           levels = c("A", "B", "C"),
                           labels = c("A", "B", "C"))
```

With function str we can check the type of the variable used in data frame.

```
str(mydata)
## 'data.frame':    30 obs. of  4 variables:
##  $ ID       : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ Headache : num  59 50 45 55 35 28 29 40 39 35 ...
##  $ Medicine : chr  "A" "A" "A" "A" ...
##  $ MedicineF: Factor w/ 3 levels "A","B","C": 1 1 1 1 1 1 1 1 1 1 ...
```

Some of most used types of variables are:

- `num`: numeric data with or without decimal values,
- `int`: special type of numeric data without decimals (discrete variable), `integer`,
- `chr`: text data, `character`,
- `factor`: categorical variable,
- `logical`: a variable with only two values: `TRUE` or `FALSE`.

In the last step we will show descriptive statistics with function `summary` and descriptive statistics by type of medicine, which will be shown with function `describeBy` (library `psych`).

```
summary(mydata[(c("Headache", "MedicineF"))])
##     Headache     MedicineF
##  Min.   :15.00   A:10
##  1st Qu.:25.75   B:10
##  Median :36.00   C:10
##  Mean   :35.10
##  3rd Qu.:41.75
##  Max.   :59.00

#Showing descriptive statistics by groups

library(psych)
describeBy(x = mydata$Headache,
           group = mydata$MedicineF)
##
##  Descriptive statistics by group
## group: A
##    vars  n mean    sd median trimmed   mad min max range skew
## X1    1 10 41.5 10.56   39.5      41 11.86  28  59    31  0.3
##    kurtosis   se
## X1    -1.45 3.34
## ---------------------------------------------------------
## group: B
##    vars  n mean    sd median trimmed  mad min max range  skew
## X1    1 10   37 10.37   40.5   37.75 5.93  18  50    32 -0.66
##    kurtosis   se
## X1     -1.1 3.28
## ----------------------------------------------------
## group: C
##    vars  n mean   sd median trimmed   mad min max range skew kurtosis
## X1    1 10 26.8 8.84     24   26.75 11.12  15  39    24 0.14    -1.84
##     se
## X1 2.8
```

We can see that people using medicine A had on average 41.5 severe headaches in 1 year period, people taking medicine B 37, while people taking medicine C only 26.8.