Assignment(s): Introduction to R

May 13, 2020

# Course: VSK1004 Applied Researcher



***Deadline assignment:*** Wednesday 13th of May 2020 @ 18h00

***How to hand in assignment:*** Hand in an R script. For more information watch

brief video on Student Portal (Folder: Course Materials

🡪 R Workshops)

# WELCOME

Welcome to the workshop number 2: Introduction to stats with R.

## Learning outcomes

By the end of this assignment(s), you should be able to:

* Read in your own data with different methods.
* Clean and manipulate your data with R functions.
* Explore your data with summary statistics.
* Create graphs for exploring your data.
* Apply what you have learned to your/other data

## Essential R assignment(s) document guidelines:

In the current document you will find the following color(s) highlight(s) and format(s). Please refer to this table for legend description.

|  |  |
| --- | --- |
| # this comment: | This is a comment writing by you to describe what you intent to do. |
| print(‘the thing') | This is the thing that you want to run. |
| ## [1] "the output” | ## [1] This is the output of the thing that you run in R. |
| # insert your code here # | This is the expected answer of each question throughout the document. |

## Dataset

In this assignment(s), we will use the following dataset:

1. Workshop Statistics\_ descriptives .xlsx.

## 

## Download the dataset

For this session, make sure to download and save the datasets in .xlsx. Please save the Excel file in a folder that you will use for this practical (e.g. workshop/data/)/ save your R scripts in.

### Setup your working directory

Once you have saved an Excel file containing your dataset, you still need to set your working directory in R.

To do this, try to find out first where your working directory is set at this moment. Ideally, the output of this code is the folder/location in which you saved the Excel file:

getwd()

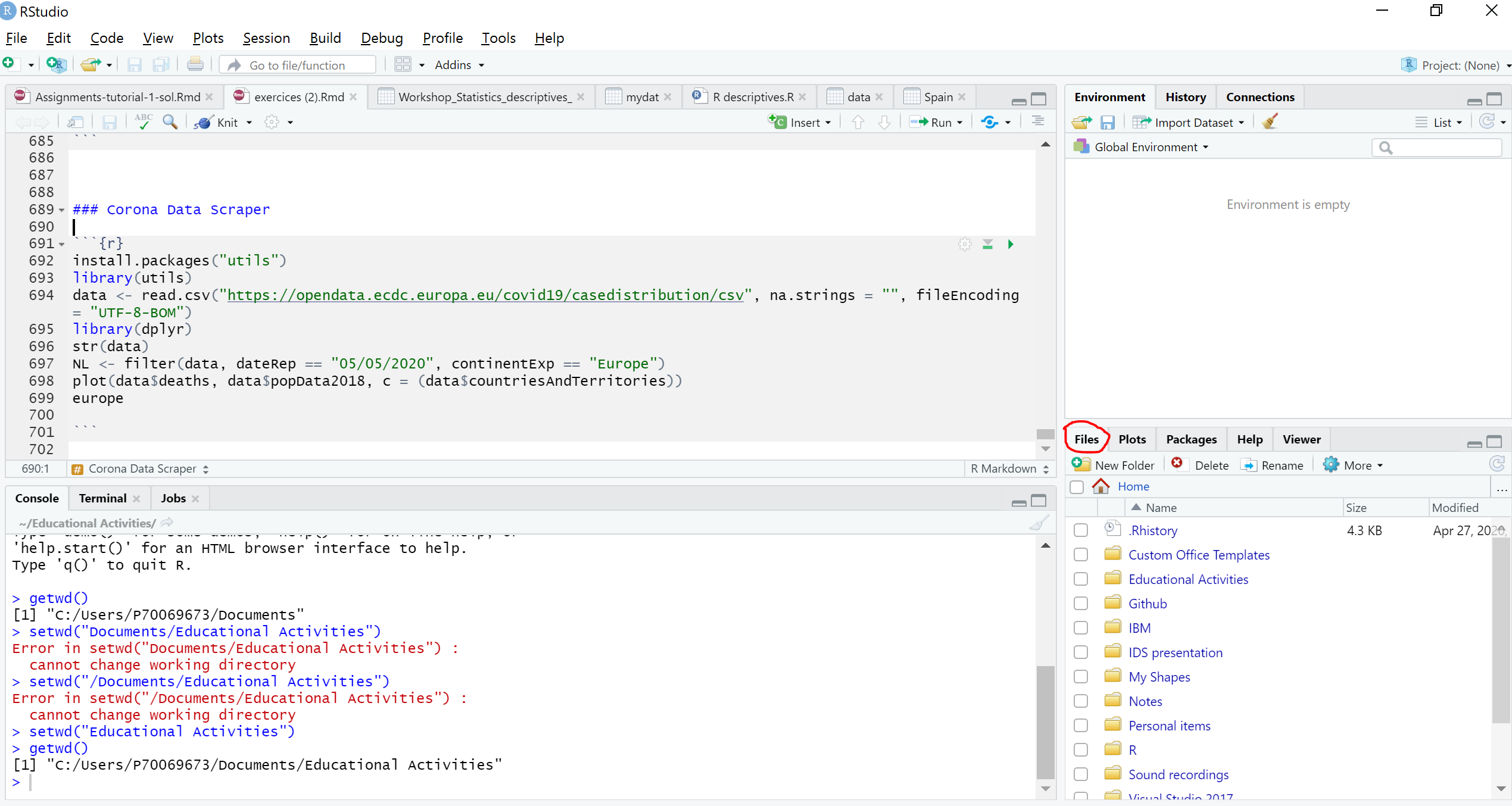
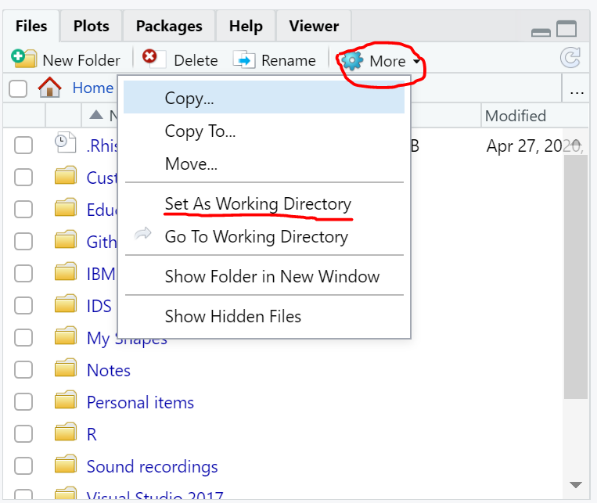
## [1] "C: /Users/workshop/data"

By executing this command, R now knows exactly in which folder you’re working.

If you need to change the current working directory, then use the setwd() function. You can use this function to define “a specific path” of the folder where you want to store your R scripts and datasets:

setwd(“path to folder”)

Another option is to check “files” view and select the path directory you want. Once you find your specific path, click on “More” and “Set as Working Directory”:



### 

### Importing data into R

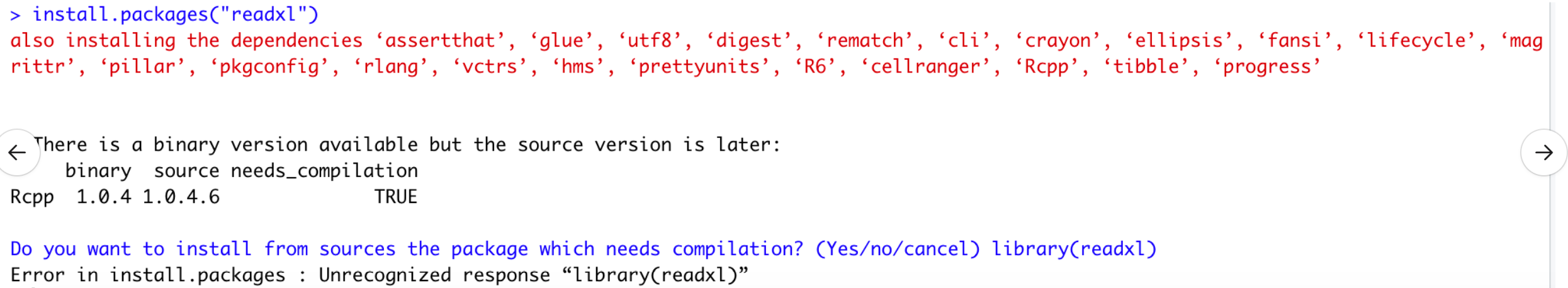
We can import the excel datafile in three different ways:

* 1. Basic R commands using readxl package

This is a package that you can use to load in Excel files in R. You will need to install the package as following:

install.packages("readxl")  
library(“readxl”)

You might encounter a warning message in the R console such this:



By answering ‘Yes’ in the console, you should now be able to complete the installation of readxl package.

After that, we need to run the following command in the R script:

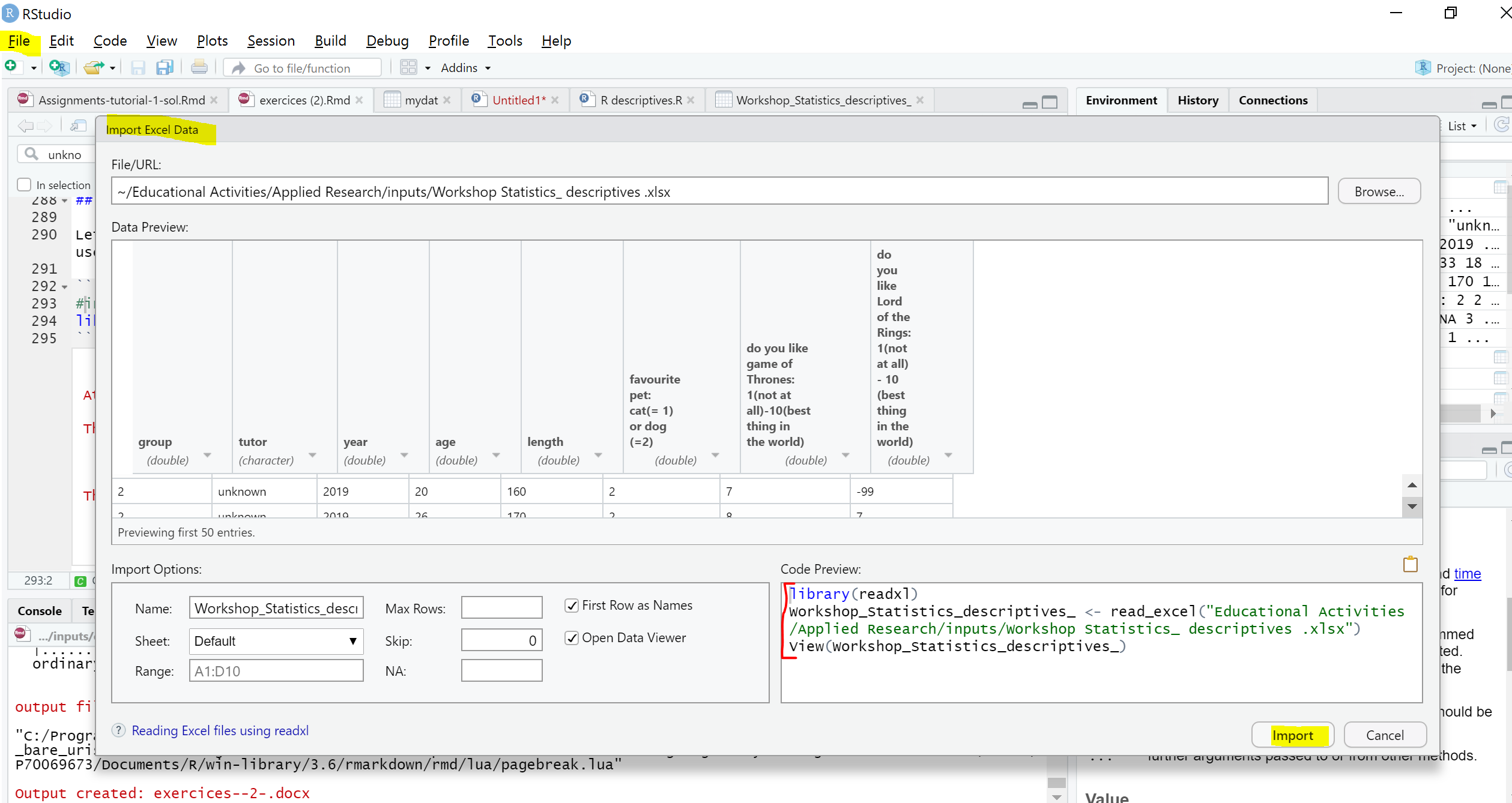
mydat <- read\_xlsx('Workshop Statistics\_ descriptives .xlsx')

At this point, we have inserted our data into R.

* 1. Using graphical user interface

It might be also an option for you to get help of R studio, with no code at all. For this purpose, you can go to:

file -> import Dataset -> excel -> \_select your file\_ -> import



1. Read the file directly from google sheet.

The file doesn’t need to be downloaded since it imports into R automatically.

Install gsheet package:

**install.packages**("gsheet")  
library(gsheet)

Copy and paste the url of the google docs. Then create a object url to store the information

url <- 'https://docs.google.com/spreadsheets/d/1JchKI5u-I5IcWbnM\_JpMkMJIZX1012oJVoIEZz2ra3c/edit#gid=0'

mydat3 <- gsheet2tbl(url)

You still have questions? Check this tutorial to learn more about importing data to R:

<https://www.datacamp.com/community/tutorials/r-data-import-tutorial>

## Understanding your dataset

First things first, we need to make sure that we correctly imported the dataset. Use the View() to look the entire dataframe in new windows.

View(mydat)

Use the code print() to print the entire dataframe in the console

print(mydat)

## # **A tibble: 69 x 8**  
## group tutor year age length `favourite pet:~ `do you like ga~  
## <dbl> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1 unkn~ 2019 19 171 2 -99  
## 2 1 unkn~ 2019 23 173 2 8  
## 3 1 unkn~ 2019 24 165 2 -99  
## 4 2 unkn~ 2019 18 170 2 -99  
## 5 2 unkn~ 2019 20 160 2 7  
## 6 2 unkn~ 2019 26 170 2 8  
## 7 2 unkn~ 2019 27 180 2 -99  
## 8 2 unkn~ 2019 33 182 2 -99  
## 9 3 unkn~ 2019 18 157 1 -99  
## 10 3 unkn~ 2019 19 153 1 3  
## # ... with 59 more rows, and 1 more variable: `do you like Lord of the Rings:  
## # 1(not at all) - 10 (best thing in the world)` <dbl>

Use the code head() and tail() to print the top and bottom dataframe respectively.

head(mydat)

## # A tibble: 6 x 8  
## group tutor year age length `favourite pet:~ `do you like ga~  
## <dbl> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1 unkn~ 2019 19 171 2 -99  
## 2 1 unkn~ 2019 23 173 2 8  
## 3 1 unkn~ 2019 24 165 2 -99  
## 4 2 unkn~ 2019 18 170 2 -99  
## 5 2 unkn~ 2019 20 160 2 7  
## 6 2 unkn~ 2019 26 170 2 8  
## # ... with 1 more variable: `do you like Lord of the Rings: 1(not at all) - 10  
## # (best thing in the world)` <dbl>

tail(mydat,5) # last five observations

## # A tibble: 5 x 8  
## group tutor year age length `favourite pet:~ `do you like ga~  
## <dbl> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 7 Hidde 2020 20 165 2 -99  
## 2 7 Hidde 2020 28 157 2 7  
## 3 0 PEERS 2020 23 169 2 6  
## 4 0 PEERS 2020 20 159 1 7  
## 5 0 PEERS 2020 22 169 2 -99  
## # ... with 1 more variable: `do you like Lord of the Rings: 1(not at all) - 10  
## # (best thing in the world)` <dbl>

It is time to make a first diagnostic of the dataset. We can receive a reasonably set of information per variable running the following code:

str(mydat)

## tibble [69 x 8] (S3: spec\_tbl\_df/tbl\_df/tbl/data.frame)  
## $ group : num [1:69] 1 1 1 2 2 2 2 2 3 3 ...  
## $ tutor : chr [1:69] "unknown" "unknown" "unknown" "unknown" ...  
## $ year : num [1:69] 2019 2019 2019 2019 2019 ...  
## $ age : num [1:69] 19 23 24 18 20 26 27 33 18 19 ...  
## $ length : num [1:69] 171 173 165 170 160 170 180 182 157 153 ...  
## $ favourite pet: cat(= 1) or dog (=2) : num [1:69] 2 2 2 2 2 2 2 2 1 1 ...  
## $ do you like game of Thrones: 1(not at all)-10(best thing in the world) : num [1:69] -99 8 -99 -99 7 8 -99 -99 -99 3 ...  
## $ do you like Lord of the Rings: 1(not at all) - 10 (best thing in the world): num [1:69] 6 7 1 -99 -99 7 8 9 -99 1 ...  
## - attr(\*, "spec")=  
## .. cols(  
## .. group = col\_double(),  
## .. tutor = col\_character(),  
## .. year = col\_double(),  
## .. age = col\_double(),  
## .. length = col\_double(),  
## .. `favourite pet: cat(= 1) or dog (=2)` = col\_double(),  
## .. `do you like game of Thrones: 1(not at all)-10(best thing in the world)` = col\_double(),  
## .. `do you like Lord of the Rings: 1(not at all) - 10 (best thing in the world)` = col\_double()  
## .. )

Alrighty, the data contains 69 observations and 8 different variables. We can also observe that all variables are considered as `numeric (num)` except tutor variable which is character or text `char`. Before starting to make further data exploration, we need to highlight some remarkable data quality inconsistencies.

* Firstly, the name of the variables are very large and it requires to be re-named if we want to make this dataset ready and more convenient for further data explorations.
* We can notice that the amount of empty responses and/or -99 are considerably high. We need to consistently assign missing values to empty responses and -99 in a way that R can read it.
* Some variables are not in the right format, for instance `favourite pet` as considering `numeric` instead of `categorical`. Indeed, it might be worth to convert this variable into a dummy one.

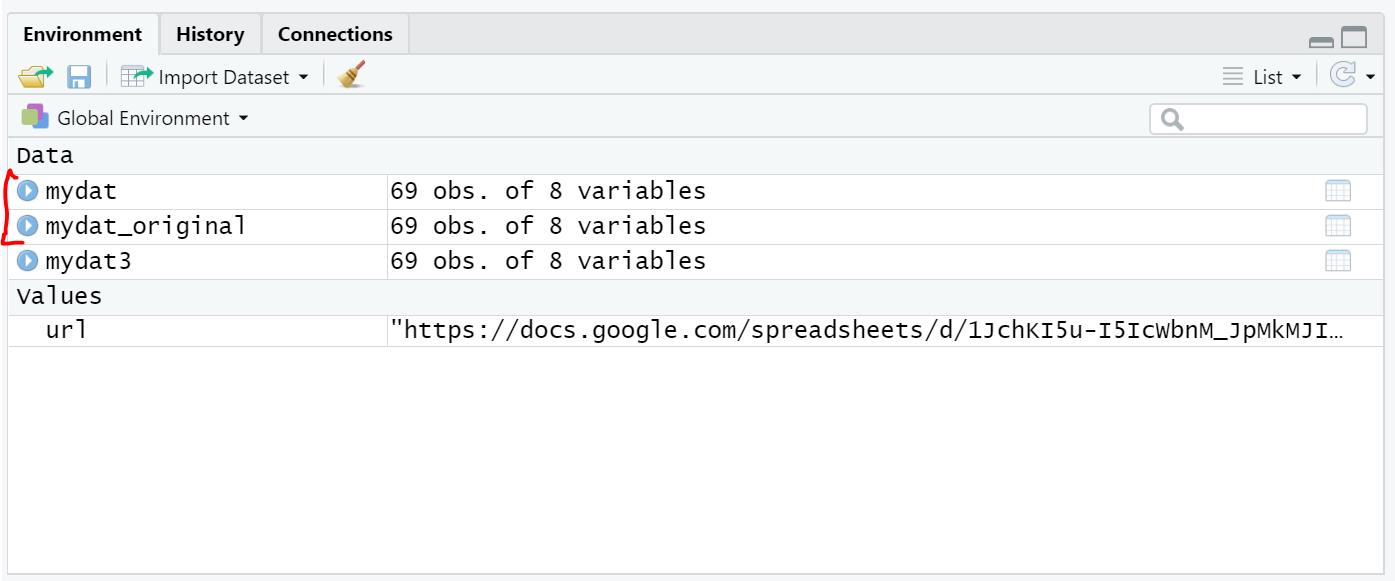
It is therefore necessary to wrangle the data, and clean it a little bit! We will do this in the next section ‘Data cleaning’, which consists of three sub-sections.

Do you want to learn more about how to know your data? Have a look at this tutorial: <https://www.datacamp.com/community/blog/an-introduction-to-cleaning-data-in-r>

## Data cleaning

*N.B.*: It is a good practice to always save the original dataset before starting to make further steps. The old and new datasets are saved at the ‘Global Environment view’. Please type the following:

mydat\_original <- mydat



### Change column names

Firstly, display the column names with names() function:

names(mydat)

## [1] "group"   
## [2] "tutor"   
## [3] "year"   
## [4] "age"   
## [5] "length"   
## [6] "favourite pet: cat(= 1) or dog (=2)"   
## [7] "do you like game of Thrones: 1(not at all)-10(best thing in the world)"   
## [8] "do you like Lord of the Rings: 1(not at all) - 10 (best thing in the world)"

Last three columns are inconveniently named. Using names() with indicating the specific position will change the name of these columns. Be sure to select properly the elements of the vector.

For instance, rename column 6 to favPet. The command will be as following:

# change the name col 6  
names(mydat)[6] <- "favPet"

Do you want to change the name of more than one column at the same time? Use the following command to replace the last two column names to GoT and LotR, respectively.

names(mydat)[7:8] <- c('GoT','LotR')

#### Exercise 1: Check/listlist the variable names in our data set. Have the variable names of column 6 through 8 been successfully changed?

names(mydat) # yes

## [1] "group" "tutor" "year" "age" "length" "favPet" "GoT" "LotR"

### 

### Missing data

Dataframes/ datasets are not always complete. Consequently, we have to deal with missing values. You have to identify values as missing if your dataframe isn’t complete because you do not want them to be included in your analysis. In the current dataframe (mydat), missings have been coded as -99 (which is usually the code given to a “missing value”). In R, we need to convert all these records to NA.

mydat$GoT[mydat$GoT == -99] <- NA  
mydat$LotR[mydat$LotR == -99] <- NA

Check one of the variable if it represents a missing value

is.na(mydat$GoT)

## [1] TRUE FALSE TRUE TRUE FALSE FALSE TRUE TRUE TRUE FALSE TRUE FALSE  
## [13] FALSE FALSE FALSE TRUE TRUE FALSE FALSE FALSE TRUE FALSE TRUE TRUE  
## [25] TRUE TRUE TRUE TRUE FALSE TRUE FALSE TRUE FALSE FALSE TRUE FALSE  
## [37] TRUE FALSE TRUE TRUE FALSE TRUE FALSE TRUE FALSE FALSE TRUE TRUE  
## [49] FALSE TRUE FALSE FALSE FALSE FALSE TRUE TRUE FALSE TRUE FALSE FALSE  
## [61] TRUE TRUE FALSE FALSE TRUE FALSE FALSE FALSE TRUE TRUE TRUE TRUE  
## [73] TRUE TRUE

Check again, first 6 rows of the data set

head(mydat,6)

## # A tibble: 6 x 8  
## group tutor year age length favPet GoT LotR  
## <dbl> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1 unknown 2019 19 171 2 NA 6  
## 2 1 unknown 2019 23 173 2 8 7  
## 3 1 unknown 2019 24 165 2 NA 1  
## 4 2 unknown 2019 18 170 2 NA NA  
## 5 2 unknown 2019 20 160 2 7 NA  
## 6 2 unknown 2019 26 170 2 8 7

It looks like we still have missing values within the variables age, length and favPet. They still have to be identified as such, let’s do that right now!

#### Exercise 2: The following variables also contain missing data (-99): age, length and favPet. Could you convert all these records into NA?

*Intructions*: [mydat$variable == -99] should be now [mydat$variable == NA]

mydat$age[mydat$age == '99'] <- NA

mydat$length[mydat$length == '99'] <- NA

mydat$favPet[mydat$favPet == '99'] <- NA

### 

Next to changing -99 to NA per variable, it can be changed for all variables in an entire data set. The pre-condition is that -99 represents in all variables a missing data point! This was the case in the current data set, so we could have also chosen to change it for all variables at once by giving the command

Mydat[mydat == -99] <- NA

Check more information in relation to NA here: <https://uc-r.github.io/missing_values>

### Dummy variables:

Dummy variables (or binary variables) are a specific type of categorical variable. They have only two values (0 = an event hasn’t occurred; 1 = an event occurred). Dummy variables are commonly used in statistical analyses and in more simple descriptive statistics. A dummy column is one which has a value of one when a categorical event occurs and a zero when it doesn’t occur. Indeed, we want to modify the variable favPet to a dummy variable. Specifically, we want the variable favPet to say whether we have seen a dog (=1) or not (=0; thus a cat). In this case, value 0 takes when is ‘cat’. Otherwise value 1.

Here you can see how to convert the value 1 for cat to value 0 when it is a cat.

# convert into dummy variable. O when cat  
mydat$favPet[mydat$favPet==1] <- 0

Check out your data! You will notice that values are 0 and 2 in favPet column. So we need one more step to create a true dummy variable!

#### Exercise 3: Can you convert favPet outcome to 1 when is a dog (=2)?

mydat$favPet[mydat$favPet==2] <- 1

Please, always check the new modifications out in the console.

#### Exercise 4: Can you please return the last 10 records of mydat?

Forgot how? Check out exploration of data again.

# show an example 10 recods  
tail(mydat,10)

## # A tibble: 10 x 8  
## group tutor year age length favPet GoT LotR  
## <dbl> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 7 Hidde 2020 19 180 1 8 9  
## 2 7 Hidde 2020 19 167 1 NA 7  
## 3 7 Hidde 2020 25 168 1 NA NA  
## 4 7 Hidde 2020 19 157 1 5 5  
## 5 7 Hidde 2020 20 170 0 7 NA  
## 6 7 Hidde 2020 20 165 1 NA 4  
## 7 7 Hidde 2020 28 157 1 7 NA  
## 8 0 PEERS 2020 23 169 1 6 4  
## 9 0 PEERS 2020 20 159 0 7 4  
## 10 0 PEERS 2020 22 169 1 NA NA

#### Exercise 5: Can you please return the first 15 records of the column favPet from mydat? You should see a set of 1 and 0 values.

head(mydat$favPet,15)

## [1] 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1

### Factor Variable

Factors are used to store categorical data. They are important for statistical modeling (analysis), since categorical variables are treated differently in statistical models than continous variables. This ensures categorical data is treated accordingly in our statistical analysis and data visualisation. Factors are the data objects which are used to categorize the data and store it as labels. It seems like we need also favPet to be treated as a factor since we might be interested in making some graphs at a later stage.

Now we can check if the variable favPet is a factor variable.

# check if factor  
is.factor(mydat$favPet)

## [1] FALSE

Well, it returns FALSE. Indeed, we have seen before that favPet is a numeric one. You can confirm it by using class():

class(mydat$favPet) # high level information

## [1] "numeric"

By the way, the most common numeric types in R are integers and doubles. Check here for further explanation: <https://swcarpentry.github.io/r-novice-inflammation/13-supp-data-structures/>

Now it is time to convert it into a factor variable. The first label ‘cat’, will correspond to favPet=0 and the second label ‘dog’, will correspond to favPet=1 because the order of the labels will follow the numeric order of the data (which in this case is 0, 1).

mydat$favPet <- factor(mydat$favPet, labels = c('cat','dog'))

We can check our new modifications now and check which R object is our variable:

class(mydat$favPet)

## [1] “factor”

#### 

#### Exercise 6: Can you please return the record number 12 of the column favPet from mydat? You should see whether she/he answered pet or dog.

mydat$favPet[12]

## [1] dog  
## Levels: cat dog

To illustrate the usefulness of the factor variables, we can create a table plotting favPet against year like this

table(mydat$year, mydat$favPet)

##   
## cat dog  
## 2019 8 19  
## 2020 13 27

Tables are much easier to interpret when using factor variables because they add useful labels to the table and they arrange the factors in a more understandable order.

#### 

#### Exercise 7: Could you please briefly interpret the results of the above table(mydat$year, mydat$favPet)? What are your conclusions based on the above table? Write down in 2-3 sentences.

#### 

#### Exercise 8: Could you create another table that shows the distribution of favPet responses among tutor’s name?

table(mydat$tutor, mydat$favPet)

##   
## cat dog  
## Anouk 4 2  
## Britt 1 5  
## Daniëlle 1 3  
## Hidde 3 7  
## Khrystyna 0 5  
## Koen 3 3  
## PEERS 1 2

## unknown 8 19

Learn more about how to convert and use factors here:

<https://swcarpentry.github.io/r-novice-inflammation/12-supp-factors/index.html>

## Data Manipulation

Let’s assume we are seeking to explore responses for only the year 2020, how would you do that? We can use filter() function from dpyyr library. We therefore need to install it:

**install.packages**("dplyr")  
library(dplyr)

Now, we can create a subset of mydat dataset. Lets filter the data for the year 2020 as following:

mydat2020 <- filter(mydat, year == '2020')

So, mydat2020 will contain just data from 2020.

You can also use the filter() function to set two conditions, which could retrieve a single observation.

- Creating a new object resulting from filtering for one favourite pet and one year:

mydat2019cats <- filter(mydat, year == '2019',favPet == 'cat')

#### Exercise 9: Filter mydat dataset for only records within tutor name equal(==) to `Koen`.

mydatkoen <- filter(mydat,tutor == 'Koen')

Learn more about aggregating and manipulating data using dplyr library here: <https://rpubs.com/williamsurles/292547>

At this point, we have successfully completed our goal of handling data quality issues such as missing values, transforming data types and dummy variables. Now, it is time to explore the data.

## Data Exploration

We first need to compute some statistics for the variables in our dataset. In this section, we will go deeply into summary statistics such as mean, variance and standard deviation.

Like Workshop1, lets try to calculate the mean of one variable. Let’s take the mean of variable age as following:

mean(mydat$age)

## [1] NA

It returns NA.

#### 

#### Exercise 10: Can you calculate the mean of length? What do you observe? Why is this the case?.

Check this Rdocumentation tutorial to get more information (or type ?mean() in the R source): <https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/mean>

mean(mydat$length) # It also returns NA.

## [1] NA

An important takehome is that mean() function will return NA if you attempt to compute the mean, or other computation, with NA values and without any noticeable input to R. Make sense right? Indeed, using mean() with no specification dealing with NA will not return the mean.

Thus, in order to calculate the mean, we need to add na.rm = TRUE on mean() function. It indicates to drop the NA values from the calculations.

mean(mydat$age, na.rm = TRUE)

## [1] 21.64179

#### Exercise 11: Can you please calculate the mean() of variable length?

Remember to tell mean function to deal with NA values.

mean(mydat$length, na.rm = TRUE)

## [1] 169.8955

A good point of using statistical programs like R is that instead of using separate functions, you can use the summary() function to print descriptive statistics from each column in mydat.

For instance, the summary statistics for all our variables:

summary(mydat)

## group tutor year age   
## Min. :0.000 Length:69 Min. :2019 Min. :18.00   
## 1st Qu.:2.000 Class :character 1st Qu.:2019 1st Qu.:19.50   
## Median :4.000 Mode :character Median :2020 Median :21.00   
## Mean :3.841 Mean :2020 Mean :21.64   
## 3rd Qu.:6.000 3rd Qu.:2020 3rd Qu.:23.00   
## Max. :7.000 Max. :2020 Max. :33.00   
## NA's :2   
## length favPet GoT LotR   
## Min. :153.0 cat :21 Min. :1.000 Min. : 1.000   
## 1st Qu.:164.0 dog :46 1st Qu.:4.000 1st Qu.: 4.000   
## Median :170.0 NA's: 2 Median :7.000 Median : 6.000   
## Mean :169.9 Mean :5.946 Mean : 5.902   
## 3rd Qu.:176.0 3rd Qu.:8.000 3rd Qu.: 8.000   
## Max. :185.0 Max. :9.000 Max. :10.000   
## NA's :2 NA's :32 NA's :28

Or just one variable, e.g. age:

summary(mydat$age)

## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's   
## 18.00 19.50 21.00 21.64 23.00 33.00 2

Nonetheless, if you look at results, neither the variance nor standard deviation appears. We need to have this information because, in short, it tells us something about the spread of observations around the mean. We will have to calculate the variance and standard deviation separately and will do so for the age and length variables.

Here is our variance for age:

var(mydat$age)

## [1] NA

Again NA result. Please bear in mind the NA for future calculations:

var(mydat$age, na.rm= TRUE)

## [1] 9.293985

Here is our standard deviation for age. Now we know what exactly we need to add on it, NA!:

sd(mydat$age, na.rm = TRUE)

## [1] 3.048604

#### Exercise 12: Can you please calculate the variance and standard deviation of the variable length?

var(mydat$length, na.rm = TRUE)

## [1] 61.06468

sd(mydat$length, na.rm = TRUE)

## [1] 7.814389

The summarize() function (not the summary () function) can be used to print together different descriptive statistics that you want, for one specific variable, for instance, age. In this example we want to know the mean, standard deviation and variance of the variable age.

So, what are the average, variance and standard deviation for age variable?

summarize(mydat, mean\_age = mean(mydat$age, na.rm = TRUE),  
 var\_age = var(mydat$age, na.rm = TRUE),  
 stdv\_age = sd(mydat$age, na.rm = TRUE))

## # A tibble: 1 x 3  
## mean\_age var\_age stdv\_age  
## <dbl> <dbl> <dbl>  
## 1 21.6 9.29 3.05

In the output, we see the answer to our question: so roughly, the mean age is about 21.64 year. If you think about this, it does make more sense if we can ask questions about averages in a particular year within a specific group.

To answer this, you can combine summarize() and filter() function. Here is the example for the variable age.

summarize(filter(mydat, year == 2019),  
 mean2019 = mean(age, na.rm = TRUE))

## # A tibble: 1 x 1  
## mean2019  
## <dbl>  
## 1 22.2

That output shows you the average age in the year 2019 was about 22,22 year old.

But you can of course summarize into multiple columns. Let’s suppose that along with finding the average age, you want to find the median in 2019:

summarize(filter(mydat, year == 2019),  
 mean2019 = mean(age, na.rm = TRUE),  
 median2019 = median(age, na.rm = TRUE))

## # A tibble: 1 x 2  
## mean2019 median2019  
## <dbl> <dbl>  
## 1 22.2 21

You’ve seen how to find the mean and median age across a set of observations.

#### 

#### Exercise 13: Can you find the mean and median of length variable across 2020?

summarize(filter(mydat, year == 2020),  
 mean2020 = mean(length, na.rm = TRUE),  
 median220 = median(length, na.rm = TRUE))

## # A tibble: 1 x 2  
## mean2020 median220  
## <dbl> <dbl>  
## 1 170. 170.

You’ve seen how to find the mean and median age and length across a set of observations, that is impressive!

To sum up, using summary as before, R will return as output the statistics of all the dataset, like 2019 and 2020. Instead, filter and summarize compute a specific measurement in a specific subset of your data such as 2019.

Now it its time to make graphs!

However, for challengers who want to experiment with making their own functions, instead of using pre-developed ones, please go to the bonus section and enjoy it!

## Data Visualisation

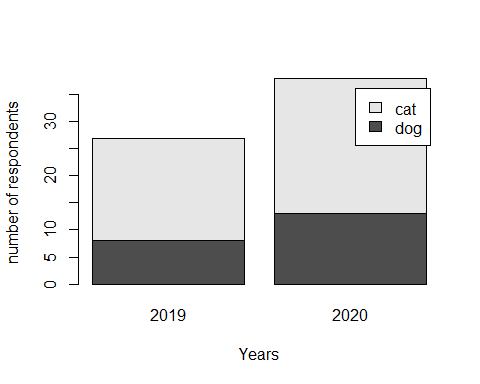
We are just about to finish. Please make last effort and lets make great graphs!

### Barplot

If you recall from the factor section, we created two tables (exercise 7 and 8) in which rows indicate year or tutor, and columns indicating FavPet labels (dog/cat).

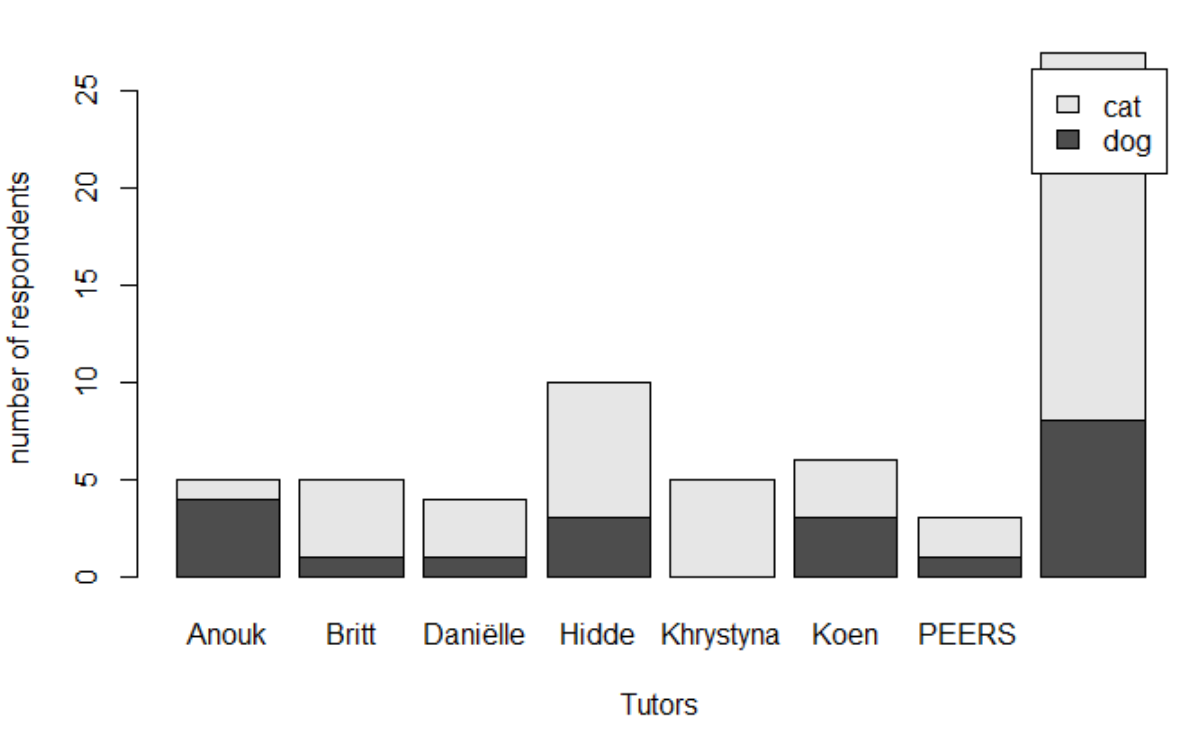
Instead of just priniting the table, you might want to represent the outcome visually. For that, we can use barplot. In the following illustration, the plot represents the data using one bar for each year (x-axis/lab), with the height and colour of the bar representing the favPet variable, how many cat and dogs (y-axis/lab).

barplot(table(mydat$favPet, mydat$year), ylab = 'number of respondents', xlab = 'Years',legend = c('dog','cat'))



#### Exercise 14: Can you create a barplot representing the tutor names on the bars?

barplot(table(mydat$favPet, mydat$tutor), ylab = 'number of respondents', xlab = 'Tutors', legend = c('dog','cat'))



### Boxplot

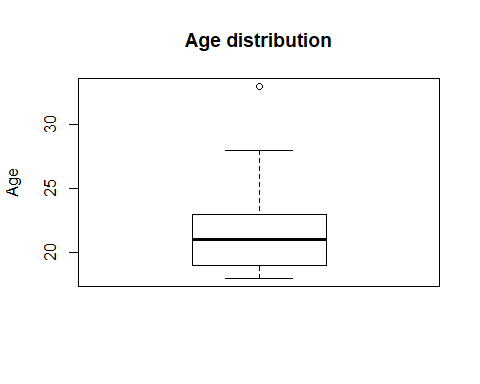
Let’s assume you want to get a visual representation of your dataset, focusing particularly on the distribution/spread of age and length. boxplots fit your aim perfectly.

What is a boxplot? A boxplot is a standardized way of displaying the distribution of data based on a five number summary (“minimum”, first quartile (Q1), median, third quartile (Q3), and “maximum”). It can tell you about your outliers and what their values are. It can also tell you if your data is symmetrical, how tightly your data is grouped, and if and how your data is skewed.

Need additional explanation? Consult in following source:

<https://www.dummies.com/education/math/statistics/what-a-boxplot-can-tell-you-about-a-statistical-data-set/>

boxplot(mydat$age, main = 'Age distribution', ylab = 'Age')

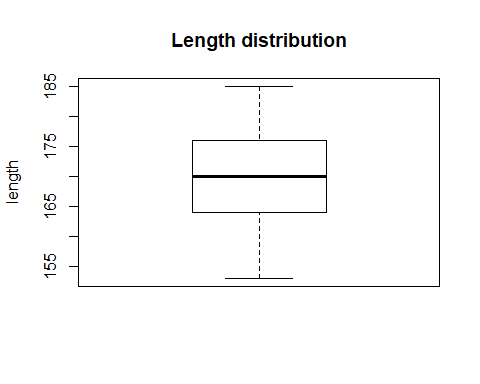


#### Exercise 15: There is a point lying outside of the box (see top quadrant). What does this indicate?

#### 

#### Exercise 16: Can you make a boxplot for the length variable?

boxplot(mydat$length, main = 'Length distribution', ylab = 'length')

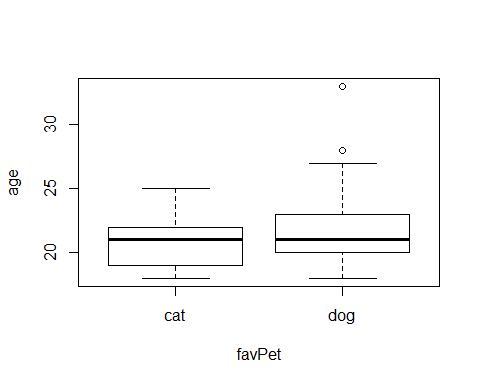


#### 

#### Exercise 17: Now that you have explored both variables, age and length, how do they differ in terms of data distribution?

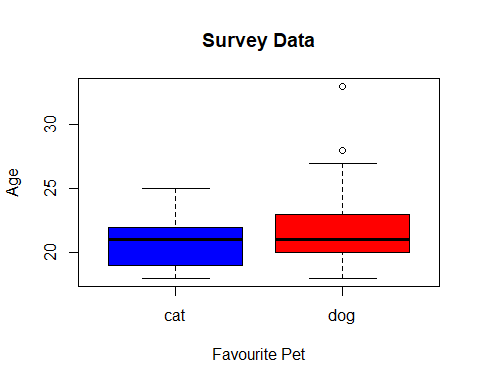
So far we have only learnt about the distribution of one variable. What happens if we portrait the age variable with the favPet variable in a boxplot? *Note: in case you experience challenges in inserting the symbol ~ correctly in the R script, please copy-paste the code from this or a word document!*

boxplot(age ~ favPet, data = mydat)



Let’s make it a little bit more fancy. It is just matter of colors!:

boxplot(age ~ favPet, data = mydat, main = "Survey Data", xlab = "Favourite Pet", ylab="Age", col = (c('blue','red')))

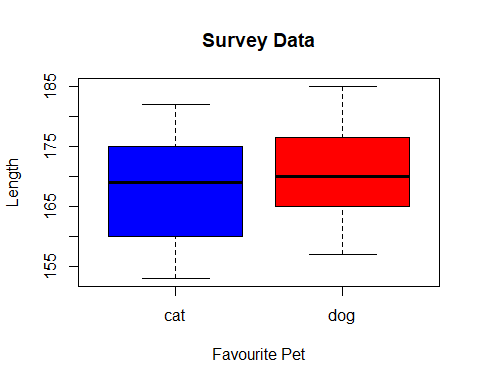


#### 

#### Exercise 18: Which favourite pet did the oldest person choose in the dataset?

#### Exercise 19: As in the last graph, could you create another boxplot that shows the distribution of the two variables favPet responses and lenght?

boxplot(length ~ favPet, data = mydat, main = "Survey Data", xlab = "Favourite Pet", ylab="Length", col = (c('blue','red')))

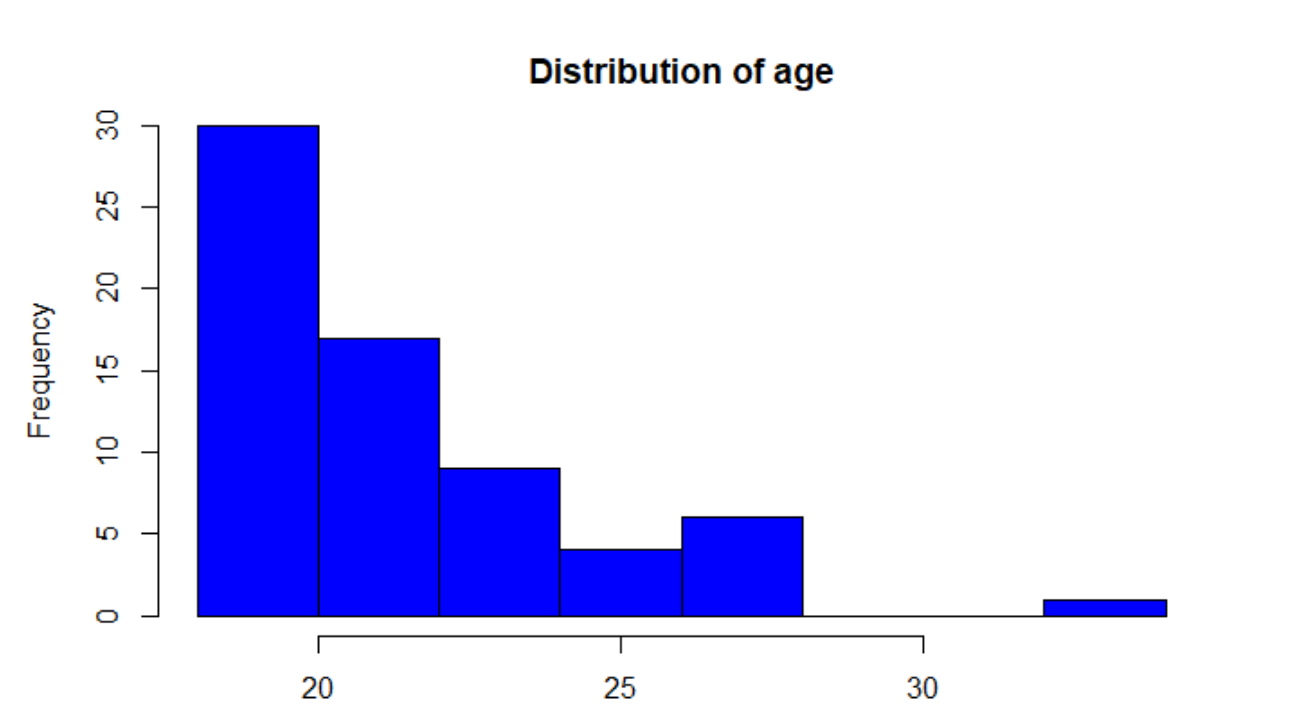


### Histogram

Another way to get a visual representation of the distribution of your dataset is creating a histogram. It allows you to easily see where the middle is in your data distribution, how close the data lie around the middle and where possible outliers are to be found.

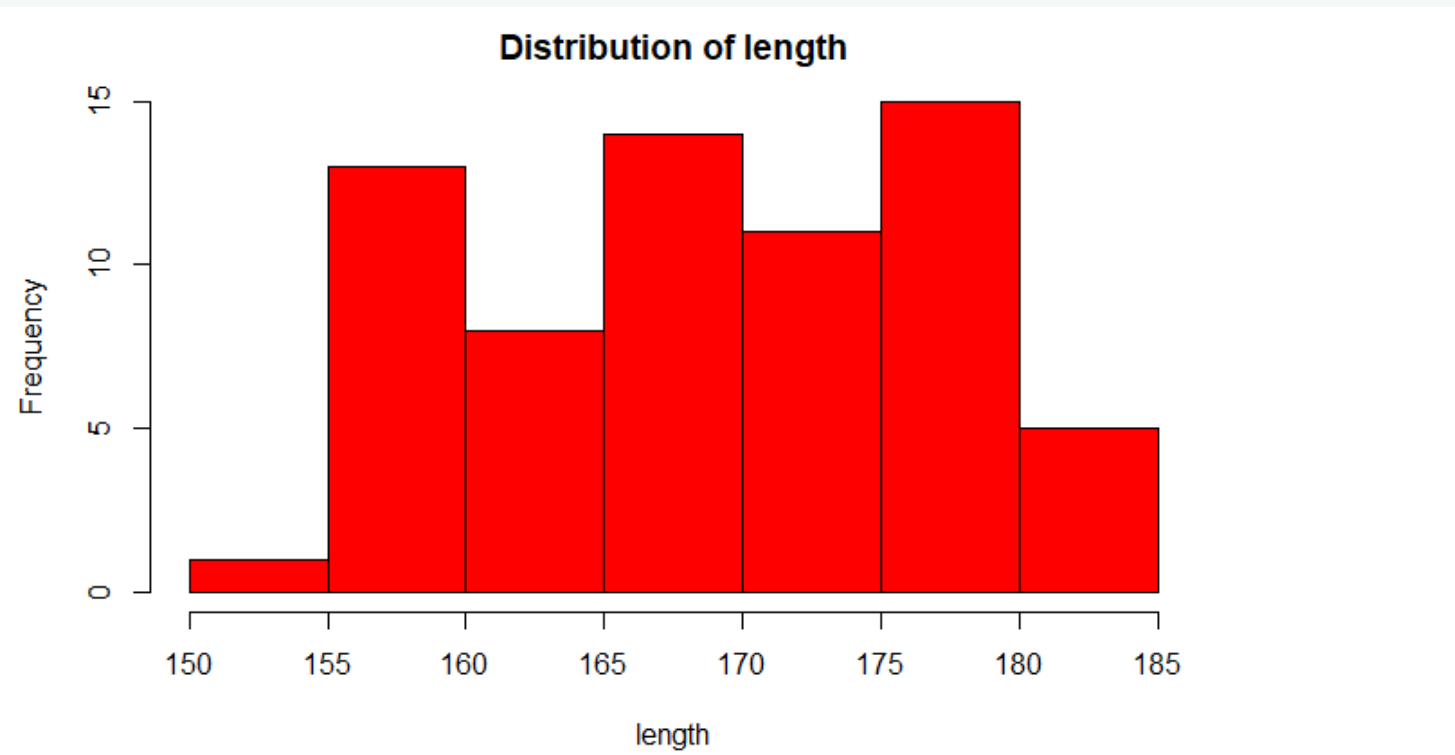
Need additional explanation? Consult in following source: <https://www.datamentor.io/r-programming/histogram/>

hist(mydat$age, main = “Distribution of age”, xlab = “age”, ylab = “Frequency”, col = “Blue”)



#### Exercise 20: Can you create a histogram that shows in red the distribution of length variable in mydat dataset? Give some interpretations about the differences between age and length distributions.

hist(mydat$length, main = 'Distribution of length', xlab = 'length', ylab = 'Frequency', col = 'red')

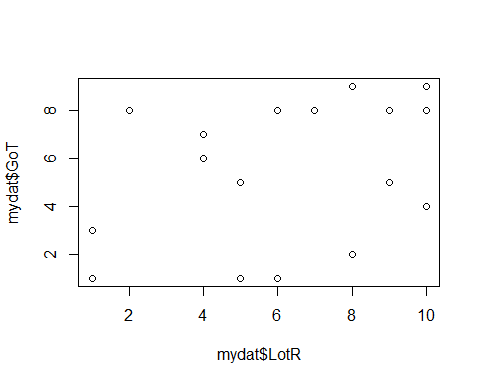


### Scatterplot

Another visualization of data is provided by a scatterplot. We now can go ahead and create scatterplot that allow us to find whether or not there are relationships between variables, which might suggest the need for further investigations.

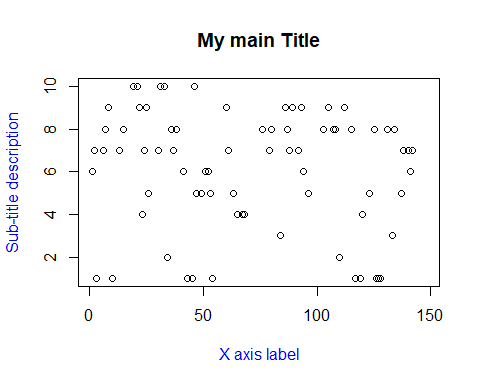
Let’s look at a scatterplot of lotR fanship nd GoT fanship:

plot(x = mydat$LotR, y = mydat$GoT)



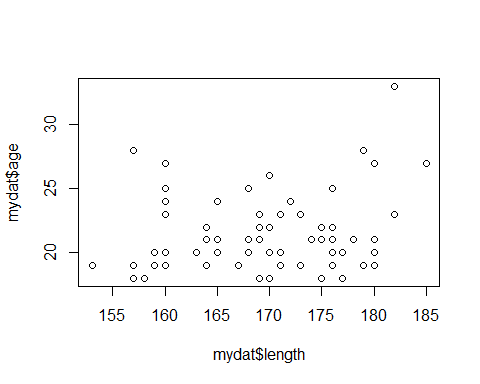
How can you add a title to the plot? Change labels of the x and y axes?

plot(c(mydat$LotR, mydat$GoT),  
 main= 'My main Title',  
 xlab = 'X axis label',  
 ylab = 'Sub-title description',  
 col.lab = 'blue')



#### Exercise 21: Can you create a graph that shows the relationship between the variables age and length in mydat dataset? Give some interpretations about the nature of the relationship between these two variables.

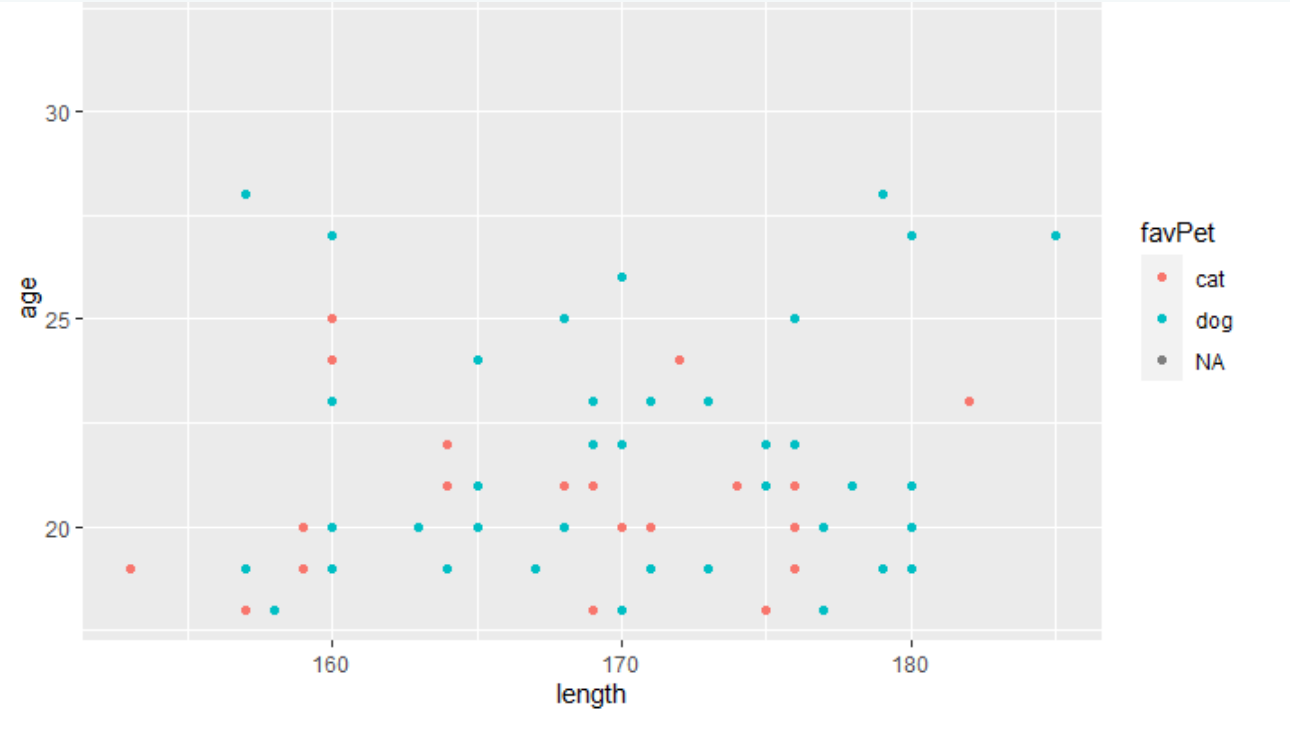
plot(mydat$length, mydat$age)



You want to know how to make it fancier/ better?. Install the ggplot2 package and try it out!:

# install.packages("ggplot2")  
library(ggplot2)  
ggplot(mydat, aes(x = length, y = age, color = favPet)) + geom\_point()

## Warning: Removed 9 rows containing missing values (geom\_point).



This short tutorial has focused on preliminary exploratory analysis. Here you can find a simple guide for taking a step back to understand your dataset: <https://towardsdatascience.com/exploring-exploratory-data-analysis-1aa72908a5df>

## Apply your knowledge section

Now it is time to play with other’s data OR your own (if you have collected numerical and categorical data for your own study). Below you will find datasets with different questions. Select one that you want to work through or work on your own data set for the applied researcher or PEERS.

### Dataset 1: Machine Learning Repository: Iris Dataset

UC Irvine Machine Learning Repository currently maintain 436 data sets as a service to the machine learning community. <https://archive.ics.uci.edu/ml/index.php>.

We are going to use the very well know data set Iris. You can find it here: <https://archive.ics.uci.edu/ml/machine-learning-databases/iris/>.

Iris, introduced by Ronald Fisher in his 1936 paper The use of multiple measurements in taxonomic problems, contains three plant species (setosa, virginica, versicolor) and four features measured for each sample. These quantify the morphologic variation of the iris flower in its three species, all measurements given in centimeters. Check this tutorial out: <https://rpubs.com/moeransm/intro-iris>

iris.uci <- read.csv(url("http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"), header=FALSE, col.names=c("sepal.length","sepal.width","petal.length","petal.width","species"))  
  
library(datasets)  
data(iris)  
summary(iris)

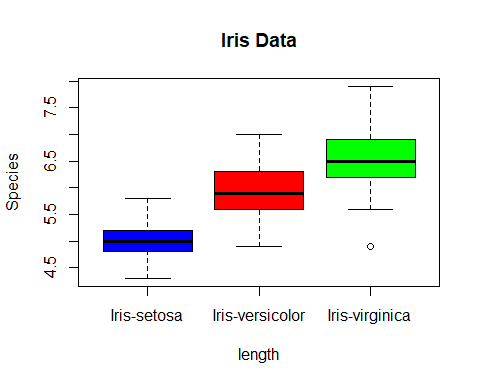
## Sepal.Length Sepal.Width Petal.Length Petal.Width   
## Min. :4.300 Min. :2.000 Min. :1.000 Min. :0.100   
## 1st Qu.:5.100 1st Qu.:2.800 1st Qu.:1.600 1st Qu.:0.300   
## Median :5.800 Median :3.000 Median :4.350 Median :1.300   
## Mean :5.843 Mean :3.057 Mean :3.758 Mean :1.199   
## 3rd Qu.:6.400 3rd Qu.:3.300 3rd Qu.:5.100 3rd Qu.:1.800   
## Max. :7.900 Max. :4.400 Max. :6.900 Max. :2.500   
## Species   
## setosa :50   
## versicolor:50   
## virginica :50   
##   
##   
##

library(dplyr)  
  
hist(iris.uci$sepal.length)

#### Q1: Can you please briefly evaluate the quality of your data? Please, apply data cleaning process and remove the missing values or outliers.

#### Q2: Can you please create a boxplot in which x-axis are species type and y-axis is sepal.length variable? Interpret the graph. If needed, calculate the necessary statistics to support your conclusion.

boxplot(sepal.length ~ species, data = iris.uci, main = "Iris Data", xlab = "length", ylab="Species", col = (c('blue','red','green')))



#### Research Question: Does the association between pental.length and pental.width variables vary between groups?

Hint: create a plot showing the relationship between these variables among groups. Based on your plots, do you expect to find a relationship between the two variables? Does it vary among species?

### Dataset 2: ACTG 175 Clinical trial

ACTG 175 was a randomized clinical trial to compare monotherapy with zidovudine or didanosine with combination therapy with zidovudine and didanosine or zidovudine and zalcitabine in adults infected with the human immunodeficiency virus type I whose CD4 T cell counts were between 200 and 500 per cubic millimeter.

The trial is documented in Hammer SM, et al. (1996), “A trial comparing nucleoside monotherapy with combination therapy in HIV-infected adults with CD4 cell counts from 200 to 500 per cubic millimeter.”, New England Journal of Medicine, 335:1081–1090. The full publication can be found at <https://www.nejm.org/doi/full/10.1056/NEJM199610103351501>

You can download the dataset from here: <https://raw.githubusercontent.com/carlosug/AppliedRR/master/data/trial_ACTG175.csv>

#### Q1: Can you please conduct a very short data quality check? Please remove corrupt or inaccurate records from the data if necessary.

#### Q2: Can you give us a short description of the dataset?

Hint: Give us the number of total patients, gender distribution, and compute age statistics as mean, variance or standard deviation

#### Research Question: Is there a correlation between a patient’s age and their CD4 T cell count at baseline?

Hint: create a plot showing the relationship between age and CD4 T cell count at baseline. Based on this plot, do you expect to find a relationship between the two variables?

### Dataset 3: Your own

## Super bonus section

### Writing our own functions

At some point, you will want to write a function, and it will probably be sooner than you think. Functions are core to the way that R works, and the sooner that you get comfortable writing them, the sooner you’ll be able to leverage R’s power, and start having fun with it.

We will learn how to create functions for mean, variance and standard deviation. This section is particularly tricky since our dataset is full of NA. Then, the code will be a little bit more complex than usual:

Start by writing the mean\_age function into your script:

my\_mean <- function(x) {  
 n = length(x[!is.na(x)])  
 average <- sum((x),na.rm = TRUE)/n  
 return(average)  
}  
my\_mean(mydat$age)

## [1] 21.64179

In order to check whether your own function works properly, compare your results to with the mean() function alike:

my\_mean(mydat$age) == mean(mydat$age, na.rm = TRUE)

## [1] TRUE

If you get TRUE then means we are doing very well.

#### Super Bonus 1: Can you use our function called my\_mean() to compute the length mean and check if you get same results as with mean(mydat$length, na.rm = TRUE)?.

my\_mean(mydat$length)

## [1] 169.8955

my\_mean(mydat$length) == mean(mydat$length, na.rm = TRUE)

## [1] TRUE

Then, we will perform similar work for variance and standard deviation.

For the variance:

my\_var <- function(x){  
 n <-length((x[!is.na(x)]))  
 m <-mean(x, na.rm = TRUE)  
 (1/(n - 1)) \* sum((x - m)^2, na.rm = TRUE)  
}

#### Super Bonus 2: Can you use our function called my\_var() to compute the length variance and check if you get same results as with var(mydat$length, na.rm = TRUE)?.

my\_var(mydat$length)

## [1] 61.06468

my\_var(mydat$length) == var(mydat$length, na.rm = TRUE)

## [1] TRUE

Now that you already master how to write functions, do make similar work for the standard deviation.

#### Super Bonus 3: Write your own function that computes the standard deviation of age in the mydat data set. Remember to add specification for dealing with NA:

Solution with NA:

sd(mydat$age, na.rm = TRUE)

## [1] 2.989967

# standard deviation  
  
my\_stdv <- function(x){  
 n <- length((x[!is.na(x)]))  
 m <- mean(x, na.rm = TRUE)  
 v <- sum((x-m)^2,na.rm = TRUE)/(n-1)  
 sqrt(v)  
}  
my\_stdv(mydat$age)

## [1] 2.989967

Well done! You have done one of the most challenging part of R.