Assignment(s): Introduction to R

Author: Lianne Ippel

*Edited by C. Utrilla Guerrero and R. de Miranda Azevedo*

November 1, 2020



*Deadline assignment(s): 12:00*

## Introduction

This session introduces the statistical computer language known as R. This first workshop includes how to import data to R, perform various calculations, obtain summary statistics for data and carry out simple visualizations.

Go at your own pace and finish as much as you can. We will give you all the answers later.

## What is R?

R is a programming language and free software environment for statistical computing and graphics supported by the R Foundation for Statistical Computing. The R language is widely used among statisticians and data miners for developing statistical software and data analysis.

Source: <https://en.wikipedia.org/wiki/R_(programming_language)>

## Obtaining and installing R and Rstudio

R can be downloaded directly from CRAN: <https://cran.r-project.org/>

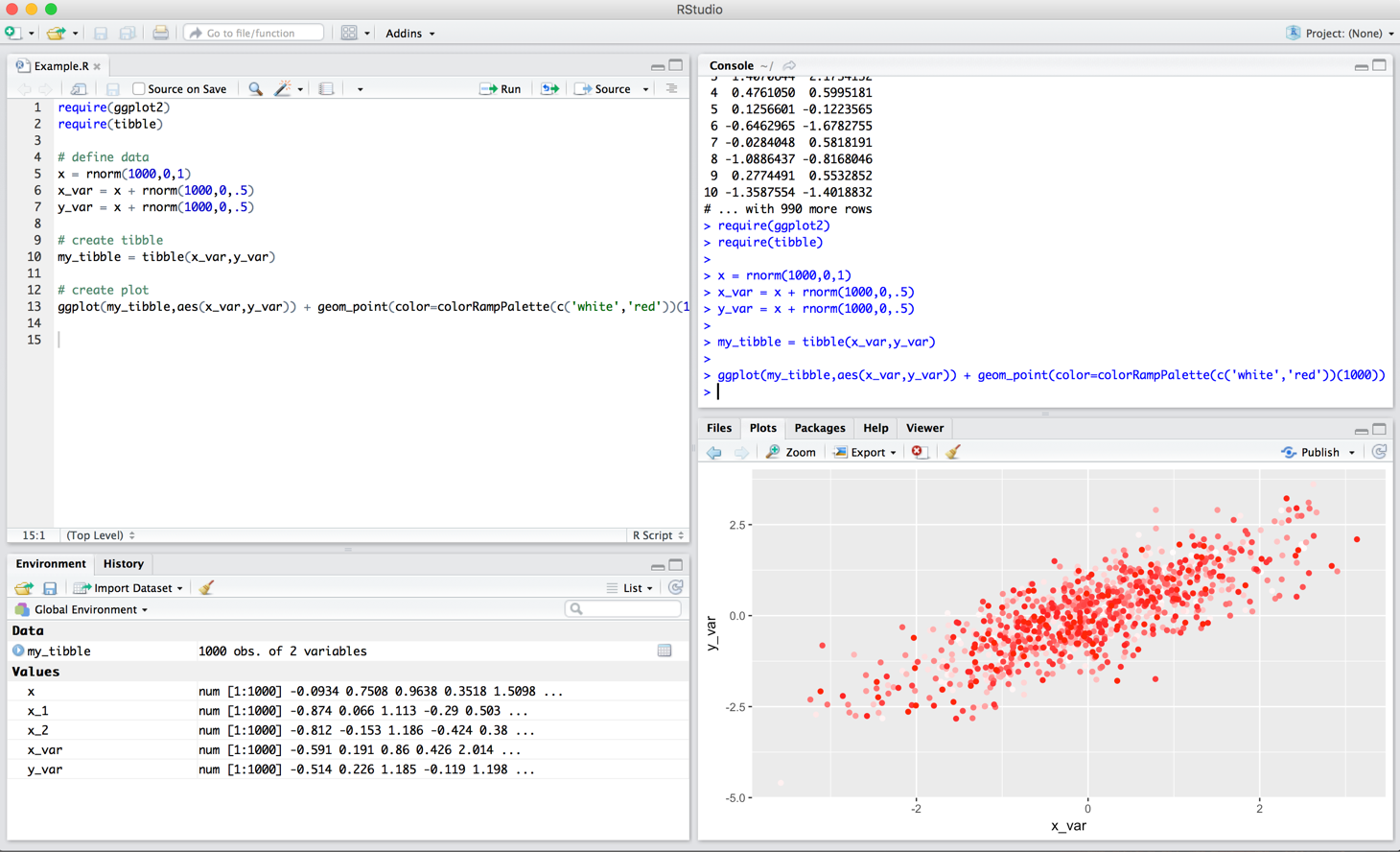
Afterwards, you should download RStudio Desktop available at the link: <https://rstudio.com/products/rstudio/download/> (follow the instructions for installation).

Also find a more extensive R introduction at the following link:

<https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf>

RStudio is an interface of R and these exercises will teach you how to program the basics of R language. RStudio has four panels: *Script, Console, Environment*, and *Help/Plot/Packages*.

The *Script* panel is a log that allows write and annotate codes for running analyses. The *Console* panel displays the output for the codes written at the Script. The *Environment* panel displays an overview of content stored in R memory (e.g. datasets, vectors). The *Help* panel displays documentation for *Packages*, and also contains a tab for displaying *Plots*.



## 

## Learning outcomes

This assignment(s) covers some basic principles of R programming and descriptive statistics. By the end of this session, you should be able to:

1. Write your own scripts
2. Read code from internet
3. Conduct descriptive statistical analyses

If you are ready, let’s get started!

## 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 that describes the following code. |
| print(‘the thing') | This is a command. Blue means function, and green means object. |
| ## [1] "the output” | ## [1] This is the output of the command ran in R. |
| # insert your code here # | This is the field to type your answers. |

## Starting playing with R

First things first: create a new script file by clicking the  icon located at the left superior panel.

Furthermore, we need to remember the purpose of this script and when it was created. Add this information by including a “#” symbol before the text (see below).

# Fill in with your name, date and file description  
# author:   
# date: 2nd November, 2020  
# Description:

Now that we created and annotated the script, let’s say hello to everybody! To execute this code, select/highlight the piece of code you want to run and either press Ctrl Enter (Command Enter for Apple users) or click Run in the upper right corner of the script panel.

### 1. Open a new script and print ‘Hello World’.

print('Hello World')

## [1] "Hello World

#### *Exercise 1*: Use the print() function to write *“I love you R, maybe”*?

# insert your code here #

Let’s practice the basics. We can start by using R as calculator.

### 

### 2. Use R as a calculator

# Examples of how R can be used as a calculator - THE BASICS  
1+1

## [1] 2

1+2^2

## [1] 5

exp(2)

## [1] 7.389056

log(2)

## [1] 0.6931472

sqrt(16)

## [1] 4

Some examples are provided but you can try other operations yourself. Simply write the equation, select the equation and run the code as before.

#### *Exercise 2*: Ask R to compute ?

# insert your code here #

Well, the calculator is nice, but R can do more things. Let’s go ahead!

### 3. Create an object - Variables

Now, we start with the creation of ‘objects’. Objects in R are containers that can hold numbers, words, large complex model results, any digital thing you can think of. You tell R to create an object with the assignment sign ‘<-’ or “=”.

# An object which holds a word  
wordObject <- 'word'  
# An object which holds a number  
numberObject <- 10 # variable "numberObject" allocate the value 10

Objects that contain numerical values, can also be used for calculation. But operation of objects should be only between numerical.

#### *Exercise 3*: Multiply numberObject x 2, can you do it?

# insert your code here #

#### *Exercise 4*: (1) make an object called ‘one’ with the value 1

#### (2) print the object

#### (3) add 100 to the object and

#### s(4) print again the object.

# insert your code here #

We already discussed different data types during the lecture, so now let’s try some of these data types.

### 4. Data types

#### 4.1. Vectors

There are many data types used in R: Vector, *Matrix, Array, Data frames, and List*. Each of these data types can be stored as an object. Actually, the objects we just created are single vectors. Vectors can contain either number or strings (text), but if you combine them, R will treat everything as a string. That means it can no longer be used for calculations.

When you want to select an entry from a vector, you do that as follows.

# creating our first vector. Create a variable with 3 elements  
firstVector <- c(1,2,3)

How would you access the second *element* of *firstvector*?

To access an element of a vector use square brackets, *e.g* elements number 2:

firstVector[2]

## [1] 2

To access a series of elements, say from 1 to 3:

firstVector[c(1:3)]

## [1] 1 2 3

Say you want to change one particular entry, you do it as follows:

# Change the first element to value 7   
  
firstVector[1] <- 7  
  
# or if you want to include more numbers  
firstVector <- c(firstVector, 8) # adding 8 to last element  
  
#maybe include a missing value   
  
firstVector <- c(firstVector , NA) #NA means Not Available/applicable

##### 

##### Exercise 5: Compute: (1) a vector ‘mynumbers’ with all prime numbers below 10, and (2) another vector ‘alphabet’ with the last five letters of the Latin alphabet?

(Hint: 2,3,5, [..])

# insert your code here #

##### Exercise 6: Create a vector with the ten first natural numbers and find out:

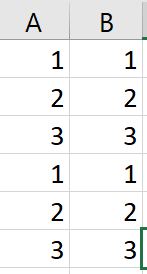
* 6th element *(Hint: To extract the sixth element of a vector we use ‘[]’)*.
* Convert last element into number 4.
* Calculate sum, mean, median and mode of these numbers (ask google!!).

**Note:** An element refers to the position of each number of a vector. For instance you can start coding: natural\_nums <- c(1,2,3,4,5,6,7,8,9,10)

# insert your code here #

#### 4.2. Matrix

The second data type, matrix, is very similar to a vector, with the only difference that a matrix has **rows and columns.** Look at the screenshot for following the logic.

# creating our first matrix   
firstMatrix <- matrix(data = c(1,2,3), nrow = 6, ncol = 2)  
firstMatrix

## [,1] [,2]  
## [1,] 1 1  
## [2,] 2 2  
## [3,] 3 3  
## [4,] 1 1  
## [5,] 2 2  
## [6,] 3 3

#select a row:  
firstMatrix[1, ]

## [1] 1 1

#select a column  
firstMatrix[, 1]

## [1] 1 2 3 1 2 3

#select a cell  
firstMatrix[1,1]

## [1] 1

Another example:

Let us define a variable z

z <- c(1,2,3,4)

using the function c() that combines its argument into a vector. The variable z is still numeric but

z <- as.matrix(z)

Transform z to a matrix object and give it new attributes such as dimensions

dim(z)

## [1] 4 1

We can also reshape z to square matrix

z <- matrix(z, ncol=2, nrow = 2)

which is the same as

z <- matrix(c(1,2,3,4), ncol = 2, nrow = 2)

To select the cell in row 1 and column 2 type

z[1,2]

## [1] 3

As you see right now, many vectors put in rows or column can make a matrix. The combination of the vector ‘mynumber’ and ‘alphabet’ that we previously created (exercise 5) will generate a matrix. Use the following functions:

cbind(mynumbers,alphabet)

## b d   
## [1,] "2" "V"  
## [2,] "3" "W"  
## [3,] "5" "X"  
## [4,] "7" "Y"  
## [5,] "11" "Z"  
## [6,] "13" "V"  
## [7,] "17" "W"  
## [8,] "19" "X"  
## [9,] "23" "Y"  
## [10,] "29" "Z"

rbind(mynumbers,alphabet)

## [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]  
## b "2" "3" "5" "7" "11" "13" "17" "19" "23" "29"   
## d "V" "W" "X" "Y" "Z" "V" "W" "X" "Y" "Z"

##### 

##### Exercise 7: What is the difference between cbind() and rbind()? What has changed in the previous combination of matrixes? Explore in google what do these functions do/ are they about.

# insert your code here #

##### Exercise 8: Create a matrix 2x3 (2 rows and 3 columns) in which the rows are the first six odd numbers consecutively, we ask:

* Convert to zero the element (2,3)
* Convert the matrix into a new one with 1x6 format
* Multiply the matrix x 6

# insert your code here #

#### 

#### 4.3. Data frames

For your current research project, the data frame will be most important. Data frame looks very similar to a matrix, however, data frames can have both numerical and string/text data, without R converting everything to strings. Just to speed things up a little, I am throwing in some additional function of R, which will be explained below:

# creating our first data frame   
help("data.frame")

firstDF <- data.frame(id = 1:10, gender = rep(c('male','female'), times = 5),  
 income = rnorm(10, mean = 1500, sd = 50 ))  
firstDF

## id gender income  
## 1 1 male 1469.202  
## 2 2 female 1505.009  
## 3 3 male 1576.877  
## 4 4 female 1448.782  
## 5 5 male 1476.981  
## 6 6 female 1461.230  
## 7 7 male 1467.756  
## 8 8 female 1480.589  
## 9 9 male 1469.193  
## 10 10 female 1516.956

#select a row:  
firstDF[1, ]

## id gender income  
## 1 1 male 1469.202

#select a column  
firstDF[, 1]

## [1] 1 2 3 4 5 6 7 8 9 10

#or   
firstDF$gender

## [1] male female male female male female male female male female  
## Levels: female male

#select a cell  
firstDF[1, 1]

## [1] 1

#or  
firstDF$id[1]

## [1] 1

#want to add a variable?  
firstDF$discipline <- rep(1:5, each = 2)

#need value labels?  
firstDF$discipline <- factor(firstDF$discipline, levels = 1:5,  
 labels = c("Medicine", "Agtech",  
 "Food", "Data science", NA))

##### 

##### Exercise 9: Delete the last column of the firstDF data frame. Check out how would you do this? Consult google!

# insert your code here #

### 5. R functions

Quick introduction to rep(). R has a lot of basic functions implemented. For instance if you want to repeat the same serie of numbers or strings. Not sure on what you have to fill out after a command? type in your console: ?rep This will activate the help window and give you information about how the rep command works. This works the same for every command in R.

# repeat a set with multiple entries, numerical or string using rep()   
# check out the difference between these two options  
  
rep(c(1,2), times = 5) # repeat the whole vector each ==5 times

## [1] 1 2 1 2 1 2 1 2 1 2

rep(c(1,2), each = 5) # each element is repeated each == 5 times

## [1] 1 1 1 1 1 2 2 2 2 2

#### *Exercise 10*: What is the difference between ‘each’ and ‘times’ parameters in ‘rep’?

# insert your code here #

### 6. Data Simulation

In need of practice data? You can create data yourself using rnorm()this command creates random data that follows a normal distribution with a mean and a stddev. Now because this is random data, you will not get the exact same result.

myData <- rnorm(n = 1000, mean = 5, sd = 1)

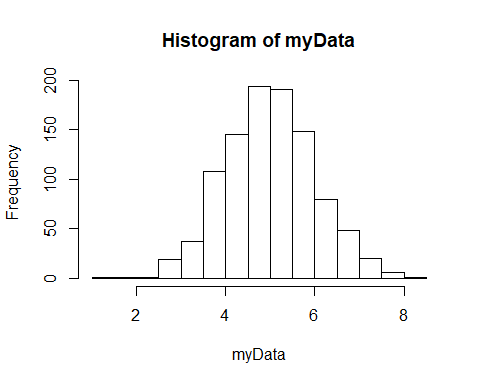
One way to force R to always give the same random sequence of numbers is by setting the seed:

set.seed(98743) # choose a nice funny number  
rnorm(n=5, mean = 0, sd = 1)

## [1] -0.04786583 -0.59425060 -0.09509987 0.54836361 0.77825508

We can also use R to visualize our data, for instance by using a histogram:

hist(myData)



Write a function to compute the mean of 2 numbers, and add 1 to the mean. Below you see an example with default values, this is good practice but you have to be careful with defaults!

meanPlus1 <- function(x1, x2, add = 1){   
 average <- (x1 + x2) / 2  
 result <- average + add  
 return(result)}   
  
meanPlus1(1,2)

## [1] 2.5

#### 

#### *Bonus Exercise*: Challenge code: ‘Opposite number’

Very simple, given a number, find it’s opposite. Examples:

|  |  |
| --- | --- |
| Input | Output |
| 1 | -1 |
| 24 | -24 |
| 33 | 33 |

*Source*: <https://www.codewars.com/collections/learning-r>

opposite <- function(number){

# insert your code here #

return()  
}

### 

### 8. R Packages

Sometimes, the integrated functions of R are not exactly what you are looking for. In that case, you can check whether someone else has written it for you and made the code available. To include such a chunk of code, we have to install a ‘package’. A google search will usually tell you which package you need. Say you want to read in Excel file, you can use the readxl package. If you want to read in an SPSS data file, use the foreign package.

# usually we install packages via the console and attach them using the script,   
# can you imagine why?  
# install.packages('readxl')  
library(readxl)  
# to read in a file use the following command   
# (but remove the #, and insert correct name of the file)  
# myExcelData <- read\_excel('name\_of\_the\_file.xls')

## 

## Working with data – It is time to apply what you have learned!

Lastly, R is only learned by doing it! Google is your friend –The most efficient avenue to help is to simply google whatever it is you want to do. I will finish this exercise by pointing you to some data that you can play with for the remaining time with some functions to learn the data a little better. You can select a dataset from the list that you get once you run the command: data(). For now let’s look at mtcars.

Try finding some information about this data set. (hint: ?how do you ask for help in R). You can look at the spreadsheet of the data by using View(mtcars). Please mind that View is with a capital. Try plotting some variables. You can use hist(), or plot(), even boxplot works or google for some other kind of plot. Select a variable (see previously for the data frame, selecting a column) and compute the mean. What do you see when you do table(mtcars$gear) , and what summary(mtcars$carb) do?

sessionInfo()

traindf <- mtcars # import dataset  
summary(mtcars)

## mpg cyl disp hp   
## Min. :10.40 Min. :4.000 Min. : 71.1 Min. : 52.0   
## 1st Qu.:15.43 1st Qu.:4.000 1st Qu.:120.8 1st Qu.: 96.5   
## Median :19.20 Median :6.000 Median :196.3 Median :123.0   
## Mean :20.09 Mean :6.188 Mean :230.7 Mean :146.7   
## 3rd Qu.:22.80 3rd Qu.:8.000 3rd Qu.:326.0 3rd Qu.:180.0   
## Max. :33.90 Max. :8.000 Max. :472.0 Max. :335.0   
## drat wt qsec vs   
## Min. :2.760 Min. :1.513 Min. :14.50 Min. :0.0000   
## 1st Qu.:3.080 1st Qu.:2.581 1st Qu.:16.89 1st Qu.:0.0000   
## Median :3.695 Median :3.325 Median :17.71 Median :0.0000   
## Mean :3.597 Mean :3.217 Mean :17.85 Mean :0.4375   
## 3rd Qu.:3.920 3rd Qu.:3.610 3rd Qu.:18.90 3rd Qu.:1.0000   
## Max. :4.930 Max. :5.424 Max. :22.90 Max. :1.0000   
## am gear carb   
## Min. :0.0000 Min. :3.000 Min. :1.000   
## 1st Qu.:0.0000 1st Qu.:3.000 1st Qu.:2.000   
## Median :0.0000 Median :4.000 Median :2.000   
## Mean :0.4062 Mean :3.688 Mean :2.812   
## 3rd Qu.:1.0000 3rd Qu.:4.000 3rd Qu.:4.000   
## Max. :1.0000 Max. :5.000 Max. :8.000

How many cases and variables are there?

dim(traindf)

## [1] 32 11

We can also get an editable spread-sheet version of the data using function ‘fix()’:

# fix(traindf)

#### 

#### *Exercise 11*: Can you calculate the absolute numbers per category for the variable ‘gear’? from ‘traindf’?:

# insert your code here #

Add new variable to give information about brand (‘brandA’, ‘brandB’):

traindf$brand <- rep(c('brandA','BrandB'))

To calculate the proportion of brands, we can take the mean of a Boolean that is equal to TRUE if a respondent is brandA

mean(traindf$brand=='brandA')

## [1] 0.5

The variable ‘brand’ is not numeric, so adding, subtracting values for this variable does not make sense. To see what type of values a variable takes type:

mode(traindf$brand)

## [1] "character"

#### 

#### *Exercise 12*: What is the average of ‘wt’? Can you calculate it?

(Hint: We can do it “manually” using the definition of the arithmetic mean or using the build function ‘mean’. First manually, + + . + and divide the result by the number of *n*.)

# insert your code here #

#### *Exercise 13*: What about using the R function ‘mean()’ as before? Do you get the same results?

# insert your code here #

#### *Exercise 14*: Is the median greater or smaller than the mean and does it mean that the distribution is skewed?

# insert your code here with explanation #

Now we continue with some other calculations. To get a sense of the variation we calculate the minimun and maximun values and the variance

range(traindf$wt) # This will give same results as calculating min and max

## [1] 1.513 5.424

As for the mean we are going to calculate the variance manually first. The expression = written is R is:

sum((traindf$wt - mean(traindf$wt))^2)/(N-1)

## [1] 0.957379

and the built function is:

var(traindf$wt)

## [1] 0.957379

#### 

#### *Exercise 15*: To get a visual confirmation of your summary measures of ‘*wt’*, draw a histogram please.

# insert your code here #

#### *Exercise 16*: Visualize the distribution of variable *brand* in the data.

Note that neither ‘plot()’ nor ‘hist()’ works with *traindf$brand* as the values are nonnumerical.

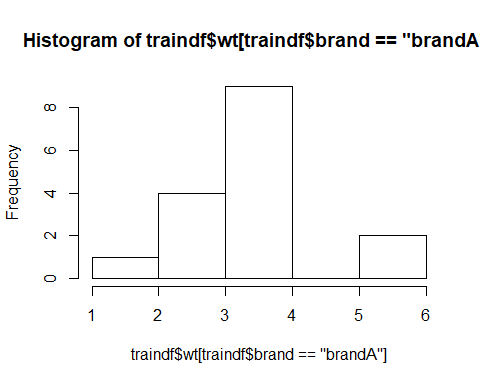
We use ‘table()’ again, to provide the plotting function the frequencies.

# insert your code here #

### 9. Lastly some bivariate summaries

Plotting *wt* against *brand* is not particularly instructive. We can however make a histogram of *wt* only for brandA

hist(traindf$wt[traindf$brand=='brandA'])



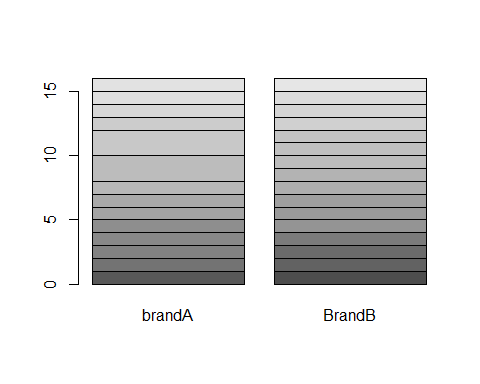
To investigate whether the ‘*wt’* differs between brands, you can calculate group-wise means as:

aggregate(traindf$wt, by = list(traindf$brand), mean)

## Group.1 x  
## 1 brandA 3.319062  
## 2 BrandB 3.115438

We can cross-tabulate choice of brands and wt easily using the function ‘table’ as above.We can get a grouped bar-chart by applying ‘barplot()’ to the table:

barplot(table(traindf$wt, traindf$brand))



### 10. Super Bonus: Descriptive Statistics Report

#### Given two different topics, can you perform a brief summary of descriptive statistics? *(Note: there is no right or wrong answer in the assignment. Limited to 1 page, inclusive graphs and tables)*

##### Option A: How contagious a disease is?

We study a well-known infectious diseases, for which WHO organization have quantified infectiousness (RO), and then project future incidence and estimation of infectiousness, as measured by the reproduction number (R), in the early stages of an outbreak. We would like to know more about basic reproducible rate and differences between infection disease types.

***Please, do the following:***

*Table 1: Values of R0 of well-known infectious diseases [1]*

|  |  |  |
| --- | --- | --- |
| **Id\_dis** | **t\_type** | **R0** |
| Measles | Airbone | 15.6 |
| Varicela | Airbone | 11.4 |
| Pertussis | Airbone dropet | 5.3 |
| Rubella | Airbone dropet | 6.2 |
| Mumps | Airbone dropet | 4.3 |
| Smallpox | Airbone dropet | 3.2 |
| SARS | Airbone dropet | 3.4 |
| Common Cold | Airbone dropet | 3.5 |
| COVID-19 | Airbone dropet | 2.3 |
| Influenza-1918 | Airbone dropet | 0.4 |
| HIV/AIDS | Body fluids | 3.1 |
| Ebola | Body fluids | 1.4 |
| Polio | Others | 6 |
| Diphtheria | Others | 3.4 |

*[1]: History and epidemiology of global smallpox eradication (*[*https://stacks.cdc.gov/view/cdc/27929*](https://stacks.cdc.gov/view/cdc/27929)*)*

|  |  |
| --- | --- |
| variable | description |
| id\_dis | Official contagious infection name. |
| t\_type | Transmission type. |
| RO | Basic Reproducible number (how contagious is a disease). |
|  |  |

***1: Create data frame with the official statistics given in the table 1****. (Preliminary steps: You can make the table manually in R or copy paste the table into excel file, save it and ask google how to read excel file into R)*

***2: Perform brief summary of descriptive statistics per transmission type. What do you find it there?***

##### Option B: Experiment on Plan Growth.

A research group compared yields (as measured by dried weight of plans) obtained under a control and two different treatment conditions.

***Please, do the following:***

***1: Import ‘PlantGrowth’ table.***

***2: Perform brief summary of descriptive statistics per group. What do you find it there?***

*Source: Dobson, A. J. (1983) An Introduction to Statistical Modelling. London: Chapman and Hall.*