Exercises for introduction to R, day 1

Exercise 1, Your first R syntax: Using R as a calculator, using some functions and creating objects

a)
Write in the following in a text editor (not a word processor!) you prefer to use for your R programming, like for instance Notepad++ (https://notepad-plus-plus.org/) or RStudio (https://www.rstudio.com/)…

```
2 + 4
```

…and then run the code in R. The output from R should look like this:

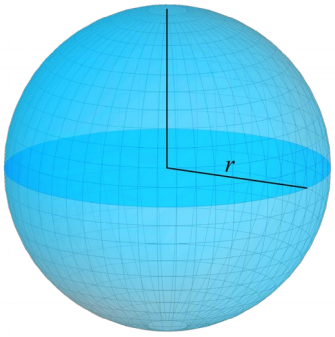
```
[1] 6
```

The "[1]" just means that the first element will follow. This will be more useful later when you work with variables and objects that contains many elements.

Try the same with subtraction:

```
6 - 4
```

b)



According to Wikipedia, a sphere (from Greek σφαῖρα — sphaira, "globe, ball") is a perfectly round geometrical object in three-dimensional space that is the surface of a completely round ball.

The formula for the area of a sphere is: $4\pi r^2$, and the volume is $4/3\pi r^3$

The symbols for multiplication, division and exponentiation in R is `*`, `/`, and `^`, respectively. The symbol for for π. is pi. Thus, writing the area of a sphere in R syntax becomes: `4*pi*r^2`

Use R to calculate the area of a sphere that is 30 cm in diameter.
Use R to calculate the the volume of the same sphere.

c)
To "store" information in objects the assignment operator (`<-`) is used. This will assign the information on the right side of the "arrow" to the left side. It is easier to understand this with some examples:

```
test <- 23
test
[1] 23
```

```
test2 <- test + 3
test2
[1] 26
```

In this example **test** is an object in the first line and is assigned the value 23. In the second line the value is printed. In the third line a new object, **test2**, is created and get assigned the value in test plus 3.

A vector in R is just a collection of numbers (integer or real) or characters (char). Names in parentheses are for users familiar with other programming languages. We define an integer as a whole number and real as number that contains a fractional part. 3 is an integer, 3.0 is a real.

```
a <- c(2,4,6)
b <- c(2.4, 2.7, 3.5)
c <- c("s", "e", "g")
```

The vector **a** is assigned the integers 2, 4 and 6. The function **c** takes a set of numbers or characters and assign them to a vector. Vector **b** is assigned three real numbers and vector **c** is assigned characters (notice the quotation marks).

Print out the contents of the vector a, b and c on your screen.

A command line in R typically contains three parts: The **name** you choose for an object , an **assignment** (**<-**), and a **function**. The arguments for a function commonly contain information on what data to use and often further specifications for the function.

An example would be:

```
tdpi <- round(pi, digits=2)
```

The name of the new object we make is our choice (here: tdpi). The function "round" round off a number to a specified number of digits. The arguments for the function is given in brackets and the two arguments are separated by comma. The first argument is the data that we want rounded, and the second argument tells how many digits we want. If we don't specify the latter argument it will in most cases be a default, which in this case is no digits.

Make two vectors with 7 values in each, for example:

```
x <- c(2,5,8,10,2,6,8)
y <- c(1:7)
```

To see the two vectors, write:

```
x
y
```

Check how many values your vector has:

```
length(x)
```

This counts the number of entries your vector has.

`summary(x)`
gives a useful summary of your vector.

The sum of a vector is found by:
`sum(y)`

Put your two vectors together in a new object called z:
`z<-c(x,y)`

Have a look at the new vector z and make sure you understand what you did!
`z`

Now, calcualte mean, median and variance of z by using the functions mean, median and var on your object z, respectively:
`mean(z)`
`median(z)`
`var(z)`

If you are tired of just seeing syntax and the commands window, type:
`hist(z)`
This will give you a simple historgram of your variable z :-)

Exercise 2, import data and make plot

At <http://www.folk.uib.no/nzlkj/IMRkurs> you will find a spreadsheet file called stopping.distance.xls. This dataset contains the relationship between speed of a car and its stopping distance.

a)
Import the dataset to R via text file. Remember that you must first create a text file of your dataset and then do the R syntax needed to import it. Set the working directory of R to where you put the text file before you perform import via the **`read.table`** function in R. Write all your syntax in a text editor and copy it over to the commands window of R.

Tip! If you have trouble with importing the data from a text file, it may help to have a look at the data by opening them in a simple text editor (not a Word processor!) like Notepad for Windows. By doing so, you will see what is used as separator between variables. Depending on the format of the text file it could be comma (`sep=","`), tabulator (`sep="\t"`), semicolon (`sep=";"`) etc. You may also have to specify what is the decimal symbol, by the `dec=` statement.

b)
Have a look at variable names and variable mode by using the `str` function.

c)
Plot the relationship with stopping distance on the y-axis and speed on the x-axis.

d)
Add a linear regression line to the plot, including a 95% confidence interval for the line.

Exercise 3, import data and make plot

At <http://www.folk.uib.no/nzlkj/IMRkurs> you will find a spreadsheet file called "pinus.xls". This file contains data on age and number of cones of *Pinus sylvestris* trees found in an area of cyberspace at a certain time in the near future. A total of 105 trees are sampled in this study, divided over the three subspecies; *sylv*, *elect*, and *cal*.

a)
Create a text file of the dataset and import this text file data into R. After importing the data, use the `head` function to check that everything looks OK with the data.

b)
Make a plot of number of cones depending on subspecies.

c)
Make a plot of Age depending on subspecies.

d) Make a plot of Cones depending on Age but split the plot into subplots for each subspecies.

e)
Add a regression line to each subplot in d), including 95% confidence intervals for the regression lines.

Exercise 4, import data and make plot

At <http://www.folk.uib.no/nzlkj/IMRkurs> you'll find a spreadsheet file called "TCB.xls". It contains sample results for the amount of TCB (thermotolerant coliform bacteria), measured in CFU (coliform forming units) from two locations in Møllendal river; one above and one below a pipeline with water running out of it.

a)
Create a text file of the dataset and import it into R. After importing the data, use the `str` function to check that everything looks OK with the data.

b)
Make a plot of the data that shows how CFU depends on location.

Exercise 5, , import data and make a map and plot

At <http://www.folk.uib.no/nzlkj/IMRkurs> you will find a spreadsheet file called "Store.lungegaardsvann.xls". It contains sample results for the amount of TCB (thermo-tolerant coliform bacteria), measured in CFU (coliform forming units) from 0.5 m depth at different locations in Lake Store Lungegaardsvann. The latitude and longitude for each sample is also given in the dataset.

a)
Create a text file of the dataset and import it into R. After importing the data, look at the whole dataset by typing the name you gave to it and press enter. Does it look OK?

b)
Use ggmap to make a map of the study area with sampling locations indicated with location number (Location is a variable in the dataset).

c)
Let's assume that the municipality of Bergen follow this standard for water quality:

CFU > 100 => Poor conditions for swimming activity.
100 > CFU >= 50 => Acceptable conditions for swimming activity.
CFU<50 => Good conditions for swimming activity.

Make a plot with location number on the x-axis (by using the variable Location) and CFU on the y-axis. Additionally, make a horizontal red line at CFU=100 to indicate the lower limit for poor conditions, and a horizontal line at CFU=50 for the upper limit of good conditions. You may add the lines by the geom_abline function. As an example, the arguments for this function when adding the green line could be as follows: geom_abline(slope=0, intercept=50, col="green", lwd=2) where lwd defines the thickness of the line (default is 1).