R and R Studio

The simplest example is to ask R to carry out some calculation:

> **5+4**

> **5*7**

How to run the script (ctrl+Enter) or click the button “Run”. That the mouse pointer should be on the same row or we can highlight that part we want to run.

The rather cryptic “[1]” in front of the output – a place of the element. This is particularly useful for results containing many values, as these may be spread over several lines of output.

Tell about packages, it’s uploaded from R to R studio and such packages carry different functions inside.

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

Some of the packages are already uploaded.

Once we uploaded it, we don’t have to do it every time. We need to SOURCE our package using the function library():

> **library (dplyr)** notice, without quotes this time. We upload it into our session.

and comments.

#about packages
#install.packages("dplyr")

If we run it, R will just copy-paste it into the console but it doesn’t consider it as a command.

> **library(dplyr) #source the library**

Everything in R is stored as **an object**. An object is most of the time associated with a variable name that allows us to refer to its content. We can think of a variable as referring to some storage location in the computer memory that holds some content (an object) that can range from a simple number to a complex model. They all are stored **in the environmen**t, we’ll see in a minute.

R objects may store diverse types of information. The simplest content is some value of one of R basic data types: **numeric, character, or logical values**. Please be aware that R is case-sensitive so true and false must be in capital letters! Other more complex data types may also be stored in objects.

Let’s create a numeric object with a variable name “numeric”. Content (i.e. objects) may be stored in a variable using the assignment operator. This operator is denoted by an angle bracket followed by a minus sign (<-):

> **numeric<-1** The effect of the previous instruction is thus to store the number 1 on a variable named numeric. Now we have our variable in the Environment.

Character values in R are **strings of characters** enclosed by either **single or double quotes** (e.g. "hello" or 'today'):
> **character<-"mark"**

While the logical values are either true or false.
> **logical<-TRUE**

By simply entering the name of a variable we can see its contents:

> **numeric**
> **character**
> **logical**

All the three are of different types. To check the types (or classes) of the objects we use the function class():

> **class(numeric)**
> **class(character)**
> **class(logical)**

By assigning some new content to an existing variable, you lose its previous content:

> **y<-25**
> **y**
> **y<-100**
> **y**

You can also assign **numerical expressions** to a variable. In this case the variable will store the result of the evaluation of the expression, not the expression itself:

> **z <- 5**
> **w <- z+2**
> **w**

This means that we can think of the assignment operation as “evaluate whatever is given on the right side of the operator, and assign (store) the result (an object of some type) of this evaluation in the variable whose name is given on the left side”.

However, if we change the variable **z** to **a**:

> **w <- a+2** it will give us an error because the variable **a** isn’t defined, there is no such variable in the environment.
Once again we remind that names in R are case sensitive, meaning that Color and color are two distinct variables with potentially very different content. This is in effect a frequent cause of frustration for

beginners who keep getting “object not found” errors. If you face this type of error, start by checking the correctness of the name of the object causing the error.

< w
< W

A few words about **vectors.**
The most basic data object in R is a vector. Even when you assign a single number to a variable (like in **x <- 45**),
x <- 45
you are creating a vector containing a single element. A vector is an object that can store a set of values **of the same data type**. Thus, you may have for instance vectors of **character strings, logical values, or numbers.**

You can create a vector in R, using the **c() function.** Some (Crawley - 2007; Zuur et al. - 2009; Matloff - 2011; and Cotton - 2013) say it means **"concatenate."** What means to link, join.
Others (Lander - 2014; Kabacoff - 2015; and Wickham - 2015) say it means **"combine."**

> vector<-c(4, 5, 6, 12, 11, 5, 6, 3)
> class (vector)

All elements of a vector must belong to **the same data type.** If that is not true, R will force it by type **coercion.** The following is an example of this:

> vector<-c("4", 5, 6, 12, 11, 5, 6, 3)
> vector
> class (vector)
> vector<-c("hi", 5, 6, 12, 11, 5, 6, 3)
> class (vector)
All elements of the vector have been converted to the character type, i.e. strings.

numeric -> character

All vectors may contain a special value called NA. This represents a missing value:
> u <- c(4, 6, NA, 2)
> k <- c(TRUE, FALSE, FALSE, NA, TRUE)
> k

Indexing. Here we come to such an important concept as indexing.
You can access a particular element of a vector through an index between square brackets:

> u[2] #The example above gives you the second element of the vector u.
Later we will explore more powerful indexing schemes.

You can also change the value of one particular vector element by using the same indexing strategies and assignment:

> k[4] <- TRUE
> k

We can index 2 elements from the vector and here we use our c() function, because we ask not 1 element but already 2. And the way to combine it is to use the c() function again.

> k[c(2,3)] #so, we use the vector function to index 2 elements from this vector

We can create a new vector using elements of another vector, using the indexing concept.

> v<-c(45, 243, 78, 343, 445, 44, 56, 77)
> v<-c(v[5], v[7])

TASKS I

1. Create a numeric vector with a variable name "num" that contains 10 elements. Check the data type of the variable.
2. Create a character vector with a variable name "char" that contains names of 5 countries. Check the data type of the variable.
3. Create a logical vector with a variable name "log" that contains 5 elements where one is a missing value. Check the data type of the variable.
4. Change the missing values of the "log" variable to TRUE.
5. Change the second element of the "num" variable to a string "number". Check the data type of the variable. Did it change? Why?
6. Create a new vector with a variable name "char_2" that contains names of the 1st and 5th countries of the "char" variable.

TASKS I

1. Create a numeric vector with a variable name "num" that contains 10 elements. Check the data type of the variable.
   num<-c(1,2,3,4,5,6,7,8,9,10)
   class(num)
2. Create a character vector with a variable name "char" that contains names of 5 countries. Check the data type of the variable.
   char<-c("Spain", "Germany", "USA", "UK", "France")
   class(char)
3. Create a logical vector with a variable name "log" that contains 5 elements where one is a missing value. Check the data type of the variable.
   log<- c(TRUE, FALSE, TRUE, NA, FALSE)
   class(log)
4. Change the missing values of the "log" variable to TRUE.

log[4]<-TRUE

5. Change the second element of the “num” variable to a string “number”. Check the data type of the variable. Did it change? Why?
   num[2]<-“number”
   class(num)
   Because a vector must contain only elements of the same data type, R coerced the numeric type to the character type.
6. Create a new vector with a variable name “char_2” that contains names of the 1st and 5th countries of the “char” variable.
   char_2<-c(char[1], char[5])