**Introduction**

This document is part of our “First Steps in R” resources. It is assumed that the reader knows how to define a vector in ***R***. Defining vectors is dealt with in the previous document in this series which can also be found on the MASH website.

**Doing calculations with functions**

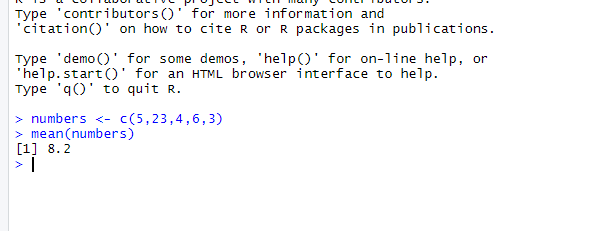
When we type a function into the console, we say we are **calling** the function. For our first example we will call the function “mean”. This function calculates the mean of the numbers in a numerical vector. First, we will define a numerical vector, for example:

numbers <- c(5,23,4,6,3)

We can now call the function “mean”:

mean(numbers)

If we type these two lines of code into the console, R will calculate the mean of the numbers in the vector:



Alternatively, we can define the vector within the brackets of the function. For example:

mean(c(5,23,4,6,3))

would produce the same result as:

numbers <- c(5,23,4,6,3)

mean(numbers)

A few more simple functions are summarised below

|  |  |
| --- | --- |
| sd() | Finds the standard deviation of the numbers in the input vector |
| var() | Finds the variance of the numbers in the input vector |
| median() | Finds the median of the numbers in the input vector |
| max() | Finds the maximum of the numbers in the input vector |
| min() | Finds the minimum of the numbers in the input vector |
| summary() | Finds the five number summary and the mean of the numbers in the input vector |

In ***R***, a function always has brackets after its name. Usually, we type arguments (inputs) into the brackets. The above functions all require a vector as an argument. How many arguments a function requires and what the arguments are depend on the function. The important principle is that we type the name of the function, then open brackets, then the arguments, then close brackets. If there is more than one argument, we separate the arguments using commas.

**Functions with other uses**

All the functions described above carry out calculations. Functions can also be used to tell us about properties of an object. Perhaps the simplest example is the “length” function. This function tells us how many elements there are in a vector. For example,

length(numbers)

Would return the value 5.

The “class” function can be used to tell us what kind of object we are dealing with. For example

class(numbers)

Returns the word “numeric” because numbers is a numeric vector.

**Using Functions to Manipulate the R Environment**

We are used to manipulating a computer using menus, windows and a mouse. This is called a GUI (Graphic User Interface). Before the development of GUIs, computers were given instructions by the user typing into a console. In ***RStudio***, we can use the mouse and menus or type instructions into the console. You may prefer a combination of both approaches.

We will look at three examples of functions we can use to manipulate the ***R*** environment.

First, “remove” or “rm” can be used to remove an object from the environment.

remove(numbers)

or

rm(numbers)

would remove the object “numbers” from the ***R*** environment. “remove” and “rm” can be used interchangeably.

We can see a list of all the objects we have defined so far using the function

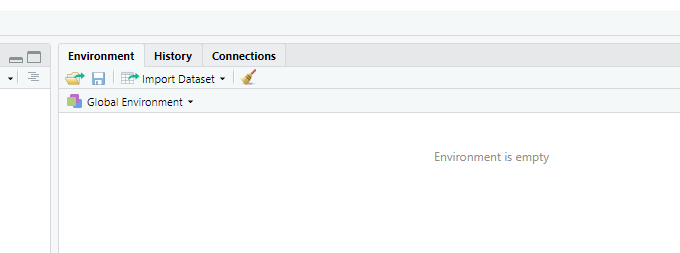
ls()

“ls” is an abbreviation of “list”. This is an example of a function which doesn’t require any arguments (ie. we don’t type anything in the brackets.

The remove function can also be used to delete all objects in the environment like so:

rm(list=ls())

It is a good idea to use this command before starting a new project so that it doesn’t get difficult to keep track of the objects which you have defined. Here the argument of the “rm” function is “list=ls()”. This tells ***R*** to remove a list of objects and that the list to be removed is the list given by the function “ls()”. Clicking on the broom symbol in the top right window in ***RStudio*** has the same effect as typing rm(list=ls()) in the console.



Finally, we can use

q()

to close ***RStudio***. Q stands for quit.

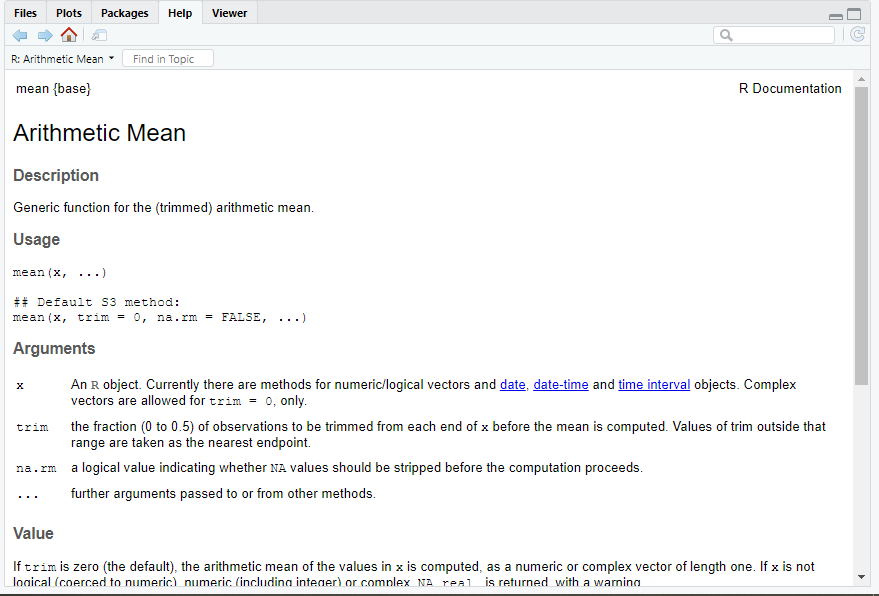
**Looking up functions**

***R*** is a programming language. We can think of all the available functions as the vocabulary of the language. We are unlikely ever to know the details of every function in ***R*** (just as we are unlikely ever to learn every word in the dictionary). A wide “vocabulary” will allow us to do more with our language but even a few words will get us somewhere. A reasonable goal is to memorise the details of some useful functions which we use regularly and then make sure that we know how to look up functions which we use less often or which we have forgotten.

Typing “?” followed by the name of a function will bring up the details of the function in the help tab of the bottom right window in ***RStudio***. If we type

?mean

into the console, the help tab shows the following



We see that we can input more than one argument into the mean function. We can, for example, use the mean function like so:

mean(vector, rm.na = TRUE)

The argument “rm.na=TRUE” must be separated from the first argument by a comma. “rm.na” is short for “Remove entries of NA from the vector before calculating the mean.” “NA” is short for “Not Available” and it is the standard label for a missing piece of data in ***R***. “rm.na” is set to FALSE by default. In the example above we have set it to TRUE (so values of NA will be removed).

It often happens that when we use the help tab to look up a function we have been using, we discover that we can use the function to do more than we thought. We may well find that we don’t understand all the information given in the help tab. Usually, we can find the information we need and not worry about other details.

If we suspect a function exists in ***R*** but we don’t know what it is, a google search will often help. The MASH website has materials to help you get to grips with many functions for carrying out the most common statistical processes.

If we can’t find a function which works the way we would like, we can define a new function using R. The MASH website contains other resources to help you learn how to do this.

**Exercise**

* Define a numerical vector with at least ten of your favourite numbers.
* Use ***R*** to find the mean, median and standard deviation of the numbers in your vector.
* Remove your vector from the environment using a command in the console.
* Check your vector has been removed by checking the list of all objects in the environment (do this by typing the appropriate command in the console).
* Use the help tab to look up the details of the “median” function.
* Close ***RStudio*** without touching your mouse.

**Solution to Exercise**

Code used:

> new\_vect <- sample(1:100,20, replace=TRUE)

> mean(new\_vect)

[1] 48.95

> median(new\_vect)

[1] 48.5

> sd(new\_vect)

[1] 29.63191

> rm(new\_vect)

> ?median

Notes:

The code

sample(1:100,20, replace=TRUE)

Tells R to sample 20 numbers at random from the vector 1:100. ie. pick 20 numbers from 1, 2, 3,...100 at random.

replace=TRUE

Means that each number can be picked more than once.

You could also answer this question by picking the numbers yourself. eg:

new\_vect<- c(...)

To close RStudio without using your mouse, use the command q() or type: **Ctrl+Q**