|  |  |  |  |
| --- | --- | --- | --- |
| Date: | **29-07-2020** | Name: | **Varun G Shetty** |
| Course: | **Coursera** | USN: | **4AL17EC093** |
| Topic: | Basic Statestics | Semester & Section: | **6th & ‘B’** |
| GitHub Repository: | **Varunshetty4** |  |  |

**Report-**

R Language:

**Functions:**

Documentation: To consult the R documentation on this function, you can use the following commands:

* help(mean)
* ?mean

There is another way of getting help on a function. For instance, if you want to know which parameters need to be provided, you can use the R function args on the specified function. An example of using args on a function is the following: args(mean)

When getting help on the mean function, you saw that it takes an argument x. X here is just an arbitrary name for the object that you want to find the mean of. Usually this object will be an R vector. We also saw the .... This is called an elipsis and is used to provide a number of optional arguments to the function.

Remember that R can match arguments both by position and by name. Let's say we want to find the mean of a vector called temperature. An example of matching by name is the following:

mean(x = temperature)

An example of matching by position is the following:

mean(temperature)

When we looked at the documentation of mean. The documentation showed us the following method:

mean(x, trim = 0, na.rm = FALSE, ...)

Na.rm can be changed by the user if a given vector contains missing values. For instance, if a the aforementioned vector called temperature would have missing values, calling mean on it would throw an output of NA. If you want the mean function to exclude the NA values when calculating the mean, you can specify na.rm = TRUE

Own Function:  You can define a function using function() code chunk. Fro example;

sum\_a\_b <- function(a, b){

return (a + b)

}

You could call this function and assign its result to the variable result, using the following code: result = sum\_a\_b(4, 5)

Getting data into R:  R contains many functions to read in data from different formats. To name only a few:

* read.table: Reads in tabular data such as txt files
* read.csv: Read in data from a comma-separated file format
* readWorksheetFromFile : Reads in an excel worksheet
* read.spss: Reads in data from .sav SPSS format.

Load in the mtcars dataset using the read.csv function. Your code should look something like: read.csv("dataset\_url"). Store the result into a variable called cars. . All you needed to specify was the "address" where the dataset could be found.

You can specify the separator in your read.csv function using the sep argument. By default, this argument for csv files is a comma. You can however easily change this to a tab by using the following code: sep = '\t'.

When reading in local files, it's good to have an idea what your working directory is. Your working directory is basically the part of your file system that R will look for files. Usually this is something along the lines of C:/Users/Username/documents. Of course this working directory is not static and can be changed by the user.

In R there are two important functions:

* getwd(): This function will retrieve the current working directory for the user
* setwd(): This functions allows the user to set her own working directory

setwd(): This function takes a character string as a name to set the working directory. You can either provide it a relative path, or you provide it an absolute path. An example of an absolute path is the following:

setwd("C:/Users/Username/Documents/datasets")

An example of a relative path is the following:

setwd("./datasets")

If you would use the latter option in your local R session, it uses the string "C:/Users/Username/Documents" through the use of the **.** character. In datacamp, it takes the current working directory and combines it with the datasets folder. As such, it saves the user a lot of typing.

Checking the directory: R has some great convenience functions for checking the files that exist in your current working directory. For instance, list.files() lists all the files that exists in your working directory.