Introduction to R and RStudio

This tutorial is an attempt to explain beginners how to install, run, and use RStudio. Unlike many of the beginners' assumption, Both R and RStudio are two different applications/software. One need to install R first. RStudio is a separate piece of software that works with R to make R much more user friendly with some helpful features that make your R programming easier and more efficient.

RStudio runs in windows mac and Linux and even over the web using RStudio Server. It is advisable to learn to use R, especially if you will be dealing with statistical operations, where you may be expected to know how to use it.

How to install R Studio

Installing R

Go to R Project website and download R for your operating system. Link https://cran.r-project.org/

Installing R Studio

Go to RStudio website and click on "Download RStudio" and follow the directions for your operating system. Link https://cran.r-project.org/

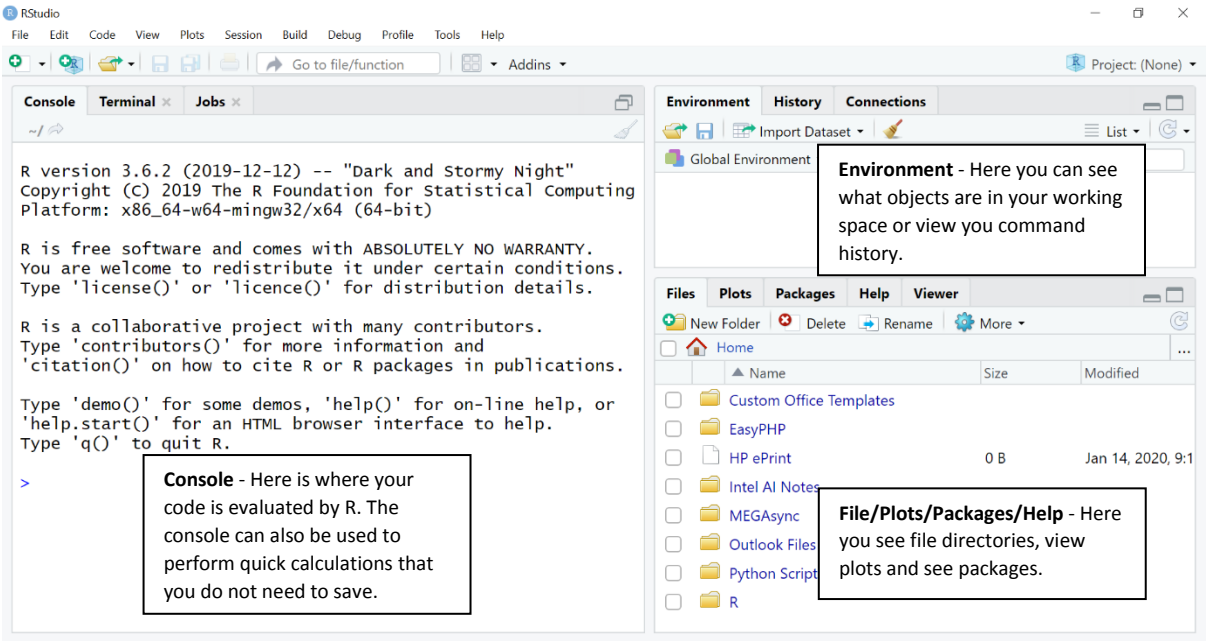
Launching RStudio

Open RStudio by clicking on the RStudio desktop icon. You're ready to go!

R Studio is an integrated development environment (IDE) for R. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management.

When you launch RStudio, you will see the following four windows or panes.

- Files/Plots/Packages/Help
- Environment/History
- Console



Files/Plots/Packages/Help

By default, This is located at the bottom right of the window and it shows you lots of helpful information. Here you can open files, view plots, install and load packages, read main pages, and view markdown and other documents in the viewer tab.

Environment/History

The Environment tab of this panel shows you the names of all the data objects (like vectors, matrices, and dataframes) that you have defined in your current R session. You can also see information like the number of observations and rows in data objects. As you get more comfortable with R, you might find the Environment / History panel useful. you can also declutter the panes in the screen, or just minimize the window by clicking the minimize button on the top right of the panel.

The history window shows all commands that were executed in the console.

Console

The console is the heart of R. By default, It is present at the bottom left of the window. It is also called a command window. Here is where R actually evaluates code. At the beginning of the console you will see the character. This is a prompt that tells you that R is ready for new code. You can type code directly into the console after the prompt and get an immediate response, just like any REPL.

Type the following command (calculation)

```
> 3+5
```

into the console and press enter. See that R immediately gives an output of 8. Type the following commands and see the outputs.

```
> 10 * 10
```

```
> log10(100)
```

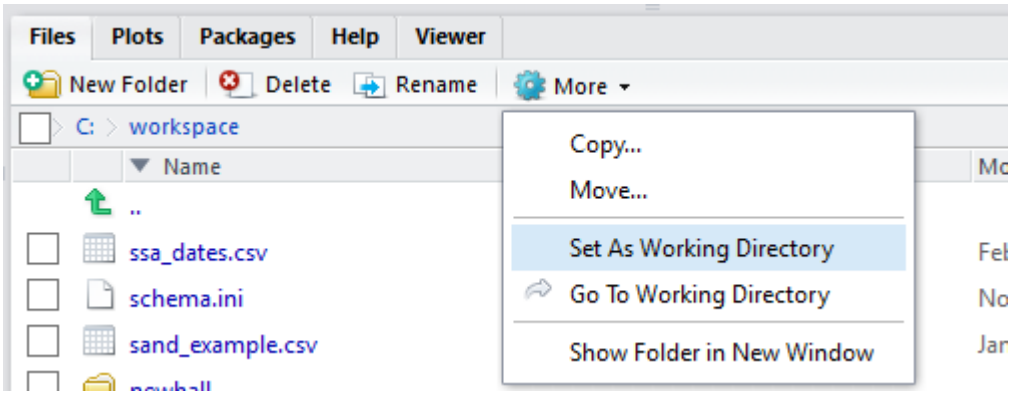
```
> "Hello World"
```

Now click History tab of the Environment/History window. You will the commands that you have entered just now.

Setting the Working Directory

Before you begin working in RStudio, you should set your working directory (a folder to hold all of your project files). For example, "C:\your_username\R_workspace".

To change the working directory in RStudio, select main menu Session → Set Working Directory → Choose Directory ... or, from the Files tab click More → Set As Working Directory to use the current location of the Files tab as your working directory.



The working directory can be set via the Console with the setwd() command. For example,

```
> setwd("C:/your_username/R_workspace")
```

This directory is where all your input data should be stored and also is the default location for plot files and other output.

To check the file path of the current working directory, type

```
> getwd()
```

Variables

A variable is a name for a value, such as x, current_temperature, or subject.id. We can create a new variable by assigning a value to it using <-.

```
> weight_kg <- 55
```

RStudio helpfully shows us the variable in the "Environment" tab. We can also print it by typing the name of the variable and hitting enter. In general, R will print to the console any object returned by a function or operation unless we assign it to a variable. Type the name of the variable as follows.

```
> weight_kg
```

Examples of valid variables names: var, var_1, var.1, value1. Spaces aren't ok inside variable names. Dots (.) are ok, unlike in many other languages.

We can do arithmetic with the variable such as follows.

```
> 2,2 * weight_kg
```

We can also change a variable's value by assigning it a new value.

```
> weight_kg <- 57.5
```

Vectors

A vector of numbers is a collection of numbers. We call the individual numbers elements of the vector. We can make vectors with c( ), for example c(1,2,3). c means "combine". In R, numbers are just vectors of length one. Many things that can be done with a single number can also be done with a vector. For example arithmetic can be done on vectors just like on single numbers. Type the following commands and see the output.

```
> myvec <- c(10,20,30,40,50)
> myvec + 1
> myvec + myvec
```

We can check the size of the vector as follows.

```
> length(myvec) <-
```

Indexing Vectors

Access elements of a vector with [ ], for example myvec[1] to get the first element. You can also assign to a specific element of a vector. Type the following commands and see the output.

```
> myvec[1]
> myvec[2]
> myvec[2] <- 5
> myvec
```

We can slice (retrieve elements) from a vector such as follows.

```
> myvec[3:5]
```

Lists

Vectors contain all the same kind of thing. Lists can contain different kinds of thing. Lists can even contain vectors or other lists as elements. We generally give the things in a list names. Try list(num=42, greeting="Hello, world").

```
> mylist <- list(num=42, greeting="Hello, world")
```

To access named elements we use $.

```
> mylist$greeting
```

Individual elements can be accessed by their index numbers such as follows.

```
> mylist[[2]]
```

Built-in Functions

R has various functions, such as sum( ). Here we have called the function sum with the argument myvec.

```
> sum(myvec)
```

Some functions take more than one argument. Let's look at the function rep, which means "repeat", and which can take a variety of different arguments. In the simplest case, it takes a value and the number of times to repeat that value.

```
> rep(42, 10)
```

Exercises

Type the following commands

```
> phrase <- c("I", "don't", "know", "I", "know")
> phrase[1:3] # first three words
> phrase[3:5] # last three words
```

1. What is `phrase[-2]`? What is `phrase[-5]`? Given those answers, explain what `phrase[-1:-3]` does.
2. Use indexing of phrase to create a new character vector that forms the phrase "I know I don't", i.e. `c("I", "know", "I", "don't")`.
3. Use `sum` to calculate the summation from 1 to 5 (ie 1+2+3+4+5).
4. Use `sum` to calculate the summation from 1 to 10,000.
5. What does `seq` do?