- Introduction to R and RStudio
- R Studio interface and basics
- Basic arithmetic and variable assignment
- Comparison and logical operators
- Data types
- Vectors
- Functions
- Loops and conditionals
- Probability and statistics
- Data frames
- Data visualisation

Introduction to R and R Studio

R is a popular programming language for statistical computing and graphics. It is widely used among statisticians and data miners for developing statistical software and data analysis.

R Studio, on the other hand, is an Integrated Environment Environment (IDE) for R that is available in two formats: RStudio Desktop, which is a regular desktop application, and RStudio Server,

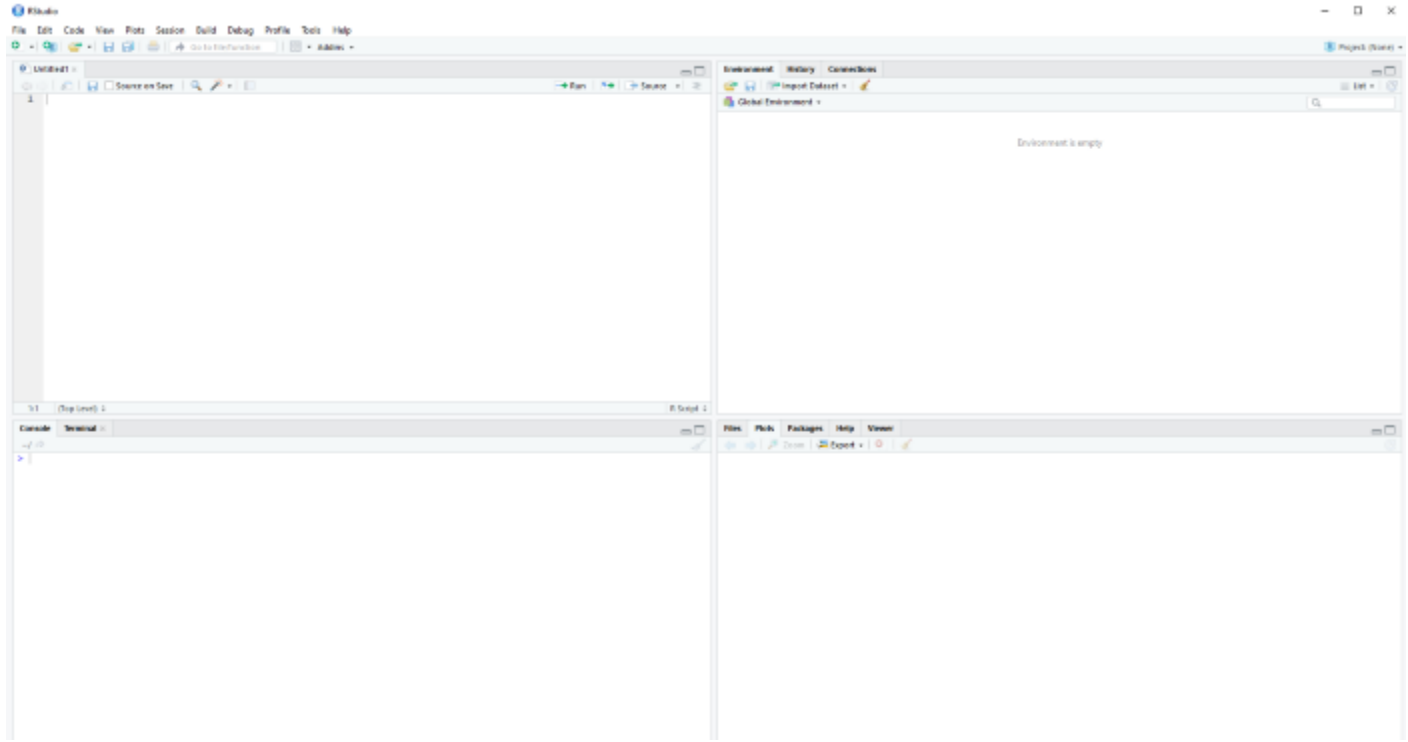
which runs on a remote server and allows access to RStudio via a web browser.

R Studio Deskstop which you can find here and install on your local computer.

So, why should you learn R?

- R is open-source, which means that it is constantly being updated and improved by other collaborative developers around the world
- It has many external packages that are suited for different purposes e.g. data manipulation, text cleaning, data visualisation, and more
- It is relatively easy and straightforward to pick up once you are familiar with the basic syntax
- Almost everyone who works with data understands and knows how to use R, so you should know it too!

R Studio interface and basics



R Studio interface

RStudio is split into 4 quadrants:

- **Script (top left):** where commands are written, executed, and saved
- **Environment (top right):** lists the data, variables, and functions that are currently in the workspace
- **Console (bottom left):** for quickly testing code and where commands and outputs are displayed, except plots
- **Plot (bottom right):** where graphics are displayed

As for basic commands that you need to know when using RStudio:

- **Clear console:** `Ctrl + L`
- **Quit RStudio:** `Ctrl + Q` or `quit()`
- **Run code from the script:** `Ctrl + Enter`
- **Remove saved variable:** `remove()`
- **Clear everything in the workspace:** `remove(list = ls())`
- **Access previous command:** `Arrow up`
- **R awaits the next command:** `>`
- **R is expecting more inputs:** `+`
- **Comment / uncomment code in script:** `Ctrl + Shift + C`
- **Getting help:** `help()` or `?`

Basic arithmetic and variable assignment

One of the most basic use cases of R is basic arithmetic. Moreover, you can also assign values to variables in order to make your calculations more flexible and robust.

- **Add:** `+`
- **Subtract:** `-`
- **Multiply:** `*`
- **Divide:** `/`

- **Power:** `^` or `**`
- **Integer divide:** `%/%`
- **Modulo (remainder after division):** `%%`
- **Variable assignment:** `=` or `<-`

Comparison and logical operators

Comparison operators compare a pair of values and return either a true or a false.

- **Equal to:** `==` (note the difference between `==`, which is used for comparison and `=`, which is used for assignment)
- **Not equal to:** `!=`
- **Greater than:** `>`
- **Less than:** `<`
- **Greater than or equal to:** `>=`
- **Less than or equal to:** `<=`

Logical operators, on the other hand, are used to combine multiple true and false statements.

- **And:** `&`
- **Or:** `|`

Data types

There are 3 main data types in R:

- **Numeric:** numbers e.g. 0, 3.5
- **Character:** can contain letters, numbers, and special characters e.g. “hello, world”
- **Logical:** boolean values i.e. true or false

Vectors

Vector is the most fundamental data structure that is used to store data in R. Vector is a one-dimensional, ordered collection of data of the same data type.

To manually create a vector in R, we use `c( )`. Alternatively, there are also built-in functions in R specifically for the purpose of creating numeric vectors such `rep( )` and `seq( )`.

We can also access, add, remove, and alter the elements in any given vector.

Functions

Programming, much like any problem-solving scenario in general, is about breaking down a big problem into smaller constituent components. This helps to not only better structure and organise our

work but more importantly, it allows for easier code review and debugging when needed.

Functions are a piece of code that performs a certain task that can be readily reused again. A function involves a simple 3-step process: input, process and output.

Inputs are sometimes called parameters or arguments which is what we pass into the function itself. Process is what the function will perform on the inputs that it is being given, which can be any form of data transformation or numerical computation. Last but not least, the function will then return the desired output.

In addition to using the built-in functions in R, we can also construct our own function using `function( )`.

Some built-in functions in R include:

- `abs( )`
- `sqrt( )`
- `round( )`
- `log( )`
- `exp( )`
- `sin( )`

Loops and conditionals

Loops repeatedly run a piece of code for a given number of times or until a condition has been met. To define a condition in loops, we need if-else statements.

There are two types of loops in R which is consistent across many other programming languages:

- **For loop:** run a piece of code many times
- **While loop:** keep running a piece of code until some condition fails

Probability and statistics

R was mainly built for statistical analysis and handling large volumes of data.

It has several built-in functions that enable summary statistics, probability distributions as well as hypothesis testing to be carried out easily:

- Summary statistics are used to summarise a set of observations e.g. `summary( )`
- Probability distributions assign probabilities to different outcomes e.g. `dbinom( )`, `pnorm( )`, `qchisq( )`, and `rexp( )`

- Hypothesis testing offers a way to test the result of an experiment e.g. `t.test( )`, `prop.test( )`, and `chisq.test( )`

Data frames

Similar to a vector, data frame is data structure that is used for storing data in R. It is a list of vectors which are of equal lengths but can be of different data types.

The vectors are presented as columns, each with a name and represent variables. The rows represent observations and can be named or unnamed.

While they can be constructed from scratch, data frames are typically produced from importing data sets e.g. csv or Excel format files.

Data visualisation

A picture is worth a thousand words. A good data scientist is able to communicate findings and persuade stakeholders through effective data visualisations.

Following built-in functions are available in R:

- **Scatter plot or line plot:** `plot( )`
- **Add graph on top of existing plot:** `points( )`

- **Draw straight lines on existing plot:** `abline( )`
- **Box plot:** `boxplot( )`
- **Histogram:** `hist( )`
- **Column graph:** `barplot( )`
- **Pie chart:** `pie( )`