- 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!

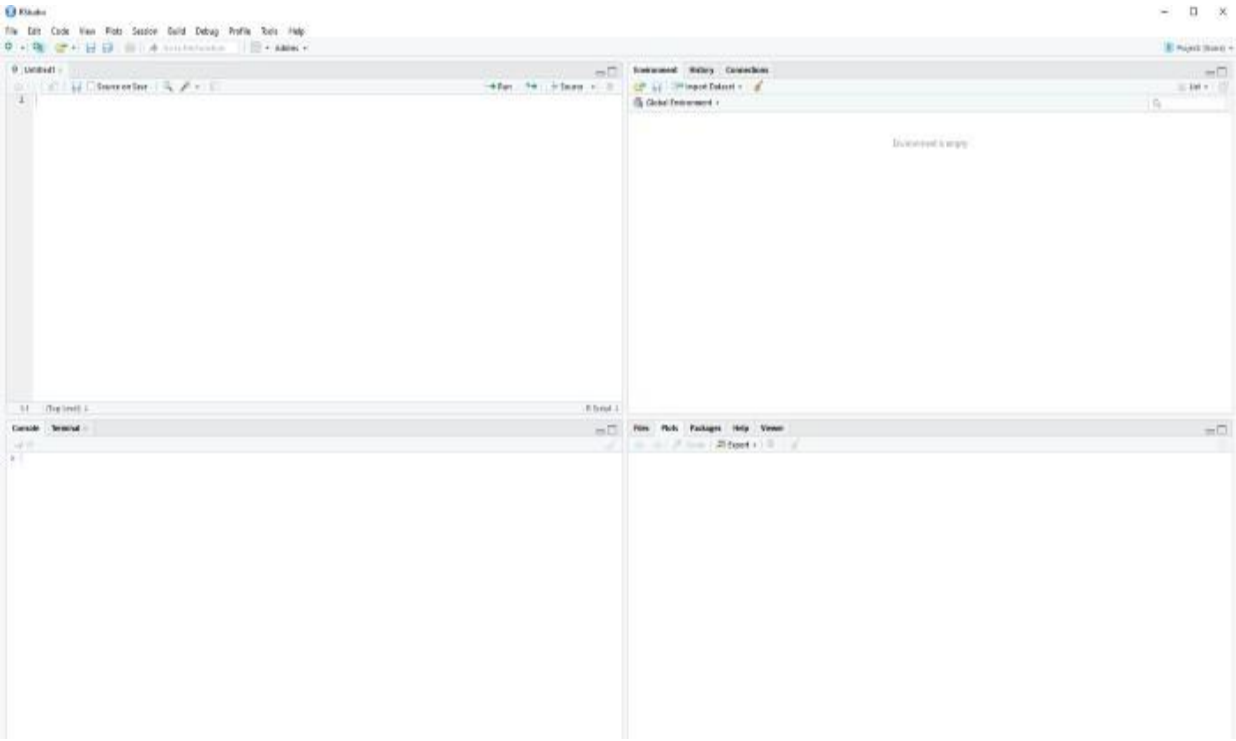
Installing R.

- Home page of The R project for Statistical Computing
- r-project.org
- Rstudio is lot easier and lot more organised.
- Install using the website of Rstudio rstudio.com.

S

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 R Studio:

- **Clear console:** Ctrl + L
- **Quit RStudio:** Ctrl + Qor 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

Variables are nothing but reserved memory locations to store values.

This means when you create a variable you reserve some space in memory.

One of the most basic use cases of R is basic arithmetic. Moreover, you

s

can also assign values to variables in order to make your calculations more flexible and robust.

Assignment operators

x=5

y<-5

20-> z

Relational operators a==b, a!=b(not equal to),a>b,a<b,a>=b,a<=b

Logical operators -and(a&b),or(a|b) , not (!a)

- **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

A data type in programming is a classification that specifies which type of value a variable has what type of mathematical, relational or logical operations can be applied to it without causing an error.

Data type helps to classify the value stored in a variable.

For eg: numeric value , we can apply various arithmetic operations but you cannot apply this operation to character value.

There are 5 main data types in R:

- **Numeric: (real or decimal),** 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
- **Complex:** 1+4i (complex numbers with real and imaginary parts)
- **Integer**: 2L (the L tells R to store this as an integer)

R provides many functions to examine features of vectors and other objects, for example

- class() - what kind of object is it (high-level)?
- typeof() - what is the object's data type (low-level)?
- length() - how long is it? What about two dimensional objects?
- attributes() - does it have any metadata?

R has many **data structures**. These include

- atomic vector
- list
- matrix
- data frame
- factors

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.

```
x1 = c(TRUE, FALSE)
x2 = c(1,2,3)
class(x2)
x3 = c(1L,2L,3L)
class(x3)
x4 = c(12 ,23 ,45,2,2222222222)
x5 = c("HELLO","HI")
x6 = c(TRUE, FALSE,12L)
x7 = c("hello", 12L,1.22, TRUE) #converts to character type vector if it has character
x8 =c(1:12)
class(x7)
```

Matrix

- **Matrix are R objects in which the elements are arranged in a two dimensional rectangular layout**
- **Syntax: matrix(data,nrow,ncol,byrow,dimnames)**
- **Data – is the input vector which becomes the data elements of the matrix.**
- **Nrow- is the number of rows to be created.**
- **Ncol- is the number of columns to be created.**
- **Byrow- is a logical value, if TRUE then the input vector elements is aarranged by rows.**
- **Dimnames- is the names assigned to the rows and columns**

```
m = matrix(c(1:25),4)
m1 = matrix(c(1:25),c(5,5))
print(m1)
m2 = matrix(c(1:20,byrow=TRUE))
print(m2)


m3 <- matrix(1:20, nrow = 4)
dimnames(m3) <- list(month.abb[1:4], month.abb[5:9])
print(m3)


# 3 array syntax- array(data,dim,dimnames)
a = array(c(1:9),dim = c(2,2,4,2))
print(a)

x1 = c("Hello")
class(x1)
```

Array

Arrays are R data objects which can store data in more than two dimensions.
Syntax: array(data,dim,dimnames)

List

Lists are the R objects which contain elements of different types like numbers,strings,vectors and another list inside it without actually changing there data type unlike vectors.
Syntax: list(data)

```
x1 = c("HI","HELLO")
x2 = c(12,13,14)
mylist = list(x1,x2)
l = list(1,2,1.23,1L,"HELLO")
mylist[2]
l[3]
```

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

```
#if statement
v1 =2
v2 =1
if ((v1+v2)>20){
  print("numbr is greater than 20")
}
```

```
#else if statement
v1 =8
v2 =12
if ((v1+v2)>20){
  print("numbr is greater than 20")
}else if((v1+v2)<20){
  print("number is less than 20")
}else
  print("number is 20")
```

S

```
#switch
switch( 2,
        '1'= print("MONDAY"),
        '2'= print("TUESDAY"),
        '3'= print("WEDNESDAY"),
        '4'=print("THURSDAY"),
        print("none of these")
)

v1= 1
repeat{
  print(v1)
  v1=v1+2
  if(v1>20){
    break
  }
}


v1=1
while(v1<20){
  print(v1)
  v1=v1+2
}

for(x in 1:20){
  if (x%%2!=0 ){
  print(x)
}
}
```

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( ).

```
fibo <- function(a){
  v1 =0
  v2 = 1
  print(v1)
  print(v2)
  for(x in 1:a){
    v3 = v1+v2
    print(v3)
    v1 = v2
    v2 = v3
  }
}
```

s

fibo(5)
fibo(10)
fibo(13)

Some built-in functions in R include:

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

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.

```
y1 = c(1:5)
y2 =c("NEHA", "DWIJESH","SNEHA","AB","AK")
y3 = c(12,32,43,54,12)
data.frame(y1,y2,y3)
data.frame(airquality)
```

PACKAGES IN R.

- dplyr- for manipulating data frames
- tidyr-for cleaning up information
- Stringr-for working with strings or text information
- Lubridate-for manipulating date information
- Httr- for working with website data
- Ggvis-(stands for grammar of graphics) this is for interactive visualizations
- Ggplot2-for creating graphics or data visualizations in R
- Shiny- for interactive applications which u can install on website
- Rio-(stands for R input and output) its for importing and exporting data
- Rmarkdown-allows to create interactive notebooks or documents for sharing information

One package to load them all

- Packman (stands for package manager)
- Command for installation of packages – install.packages("nameof the package")
- Eg.:run install.packages("pacman")
- This will make it available in the hardware but loading means actually making it accessible for use.
- Command for loading is library("nameof the package")
- Eg.: library("pacman")

Once pacman is installed, other packages can be installed using pacman.

- Or use "pacman::p_load"
- P_load function from pacman without actually loading pacman.
- Command: pacman::p_load (pacman, dplyr,Ggally, ggplot2, ggthemes, ggvis, httr,lubridate, plotly, rio, rmarkdown, shiny, stringr, tidyr)
- So once you have pacman, this command will help you to install ,make it available and load all these packages.

Base packages

s

- Packages which come it with r natively like the datasets packages
- You still have to load and unload packages manually.
- Use the command- library(datasets)

Clear packages

- With pacman, specific packages can be unloaded separately
- Command- p_unload(dplyr, tidyr)
- Or unload all using the command- p_unload(all) , this command unloads all the third party packages.
- For base packages use command- detach("package:datasets",unload=TRUE)
- To clear console, command- cat("\014")

Questions:

1. Create a vector with some of your friend's names

i. Get the length of above vector
ii. Get the first two friends from above vector
iii. Get the 2nd and 3rd friends
iv. Sort your friends by names using 2 methods

2. Compute the difference between 2014 and the year you started at this university and divide this by the difference between 2014 and the year you were born. Multiply this with 100 to get the percentage of your life you have spent at this university.
3. Compute the sum of 4, 5, 8 and 11 by first combining them into a vector and then using the function sum.
4. Create three vectors x,y,z with integers and each vector has 3 elements.Combine the three vectors to become a 3×3 matrix A where each column represents a vector. Change the row names to a,b,c.
5. What is a vector? How to create it? Create a vector A of elements 5, 2, -2, 6,7,10,12,14,15 and from it create a vector Y containing elements of A>6
6. Create a vector containing following mixed elements {1, 'a', 2, 'b'} and find out its class.
7. Write a R program to create three vectors numeric data, character data and logical data. Display the content of the vectors and their type.

8. Write a R program to create a 4 x 5 matrix, 3 x 2 matrix with labels and fill the matrix by rows and 2 × 2 matrix with labels and fill the matrix by columns.
9. Write a R program to compute sum, mean and product of a given vector elements.
10. List all the observations of “airmiles” dataset.
11. Write a R program to compute addition, subtraction and multiplication of two matrices of dimension 4x4.
12. Write a R program to create a list containing a vector, a matrix and a list; and
13. Give names to the elements in the list. Access the second element of the list.