QUEEN'S UNIVERSITY BELFAST
SCHOOL OF ELECTRONICS, ELECTRICAL ENGNIEERING AND COMPUTER SCIENCE

Data Analysis & Visualisation

CSC3062

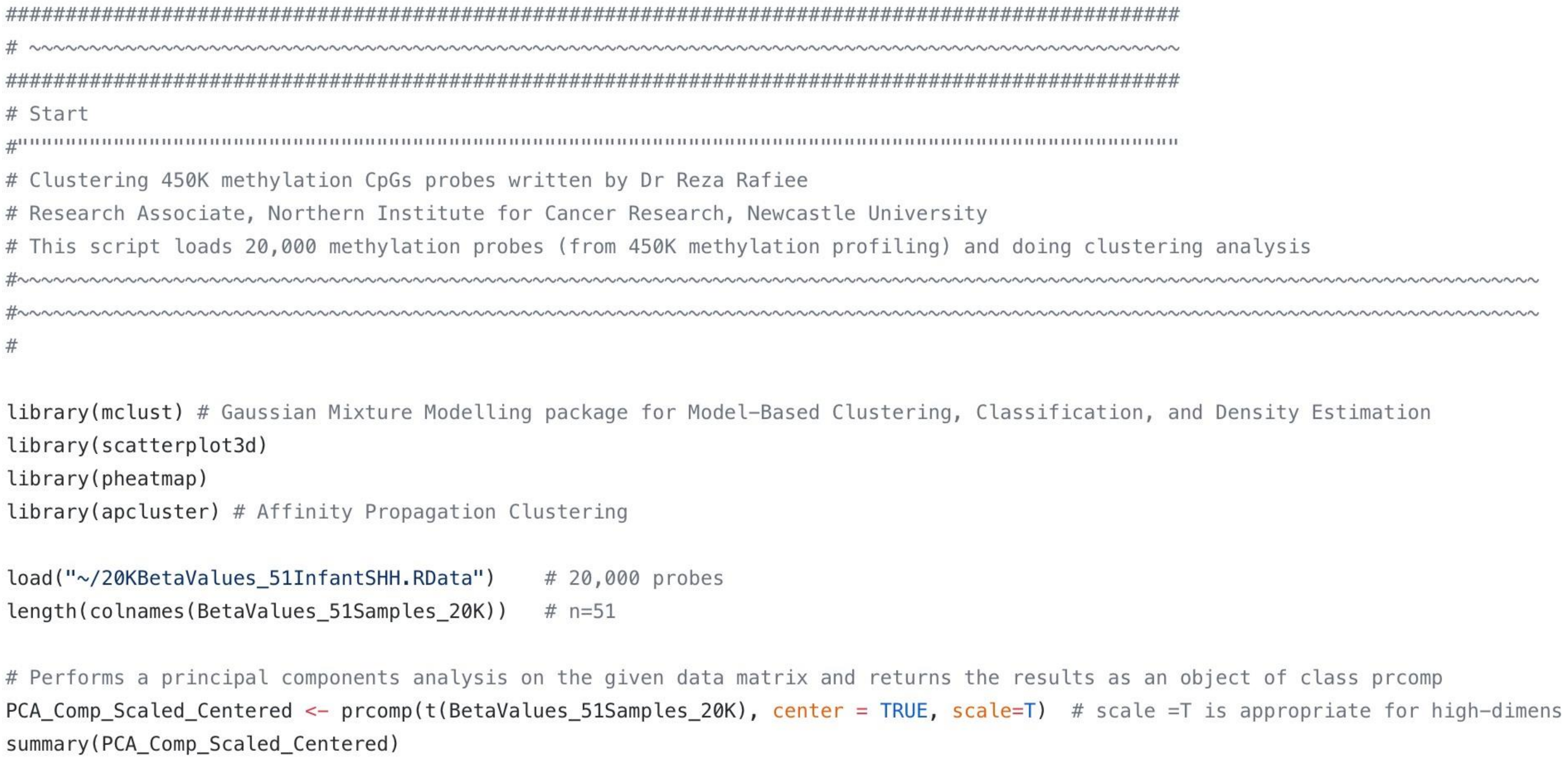
BEng (CS & SE), MEng (CS & SE), BIT & CIT

Dr Reza Rafiee

Semester 1 – 2019/2020

This is R

This is R

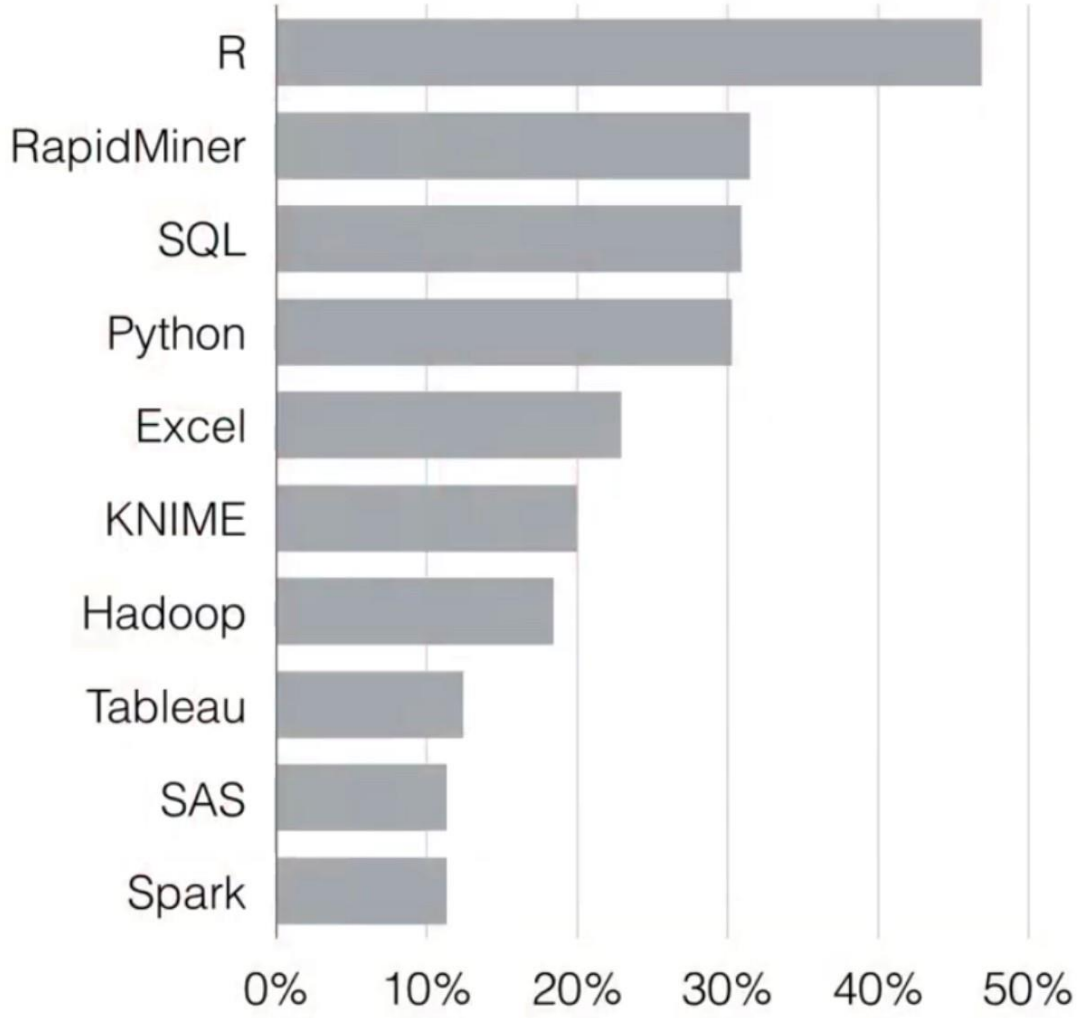


```r
##################################################################################################
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
##################################################################################################
# Start
#"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
# Clustering 450K methylation CpGs probes written by Dr Reza Rafiee
# Research Associate, Northern Institute for Cancer Research, Newcastle University
# This script loads 20,000 methylation probes (from 450K methylation profiling) and doing clustering analysis
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#

library(mclust) # Gaussian Mixture Modelling package for Model-Based Clustering, Classification, and Density Estimation
library(scatterplot3d)
library(pheatmap)
library(apcluster) # Affinity Propagation Clustering

load("~/20KBetaValues_51InfantSHH.RData")    # 20,000 probes
length(colnames(BetaValues_51Samples_20K))    # n=51

# Performs a principal components analysis on the given data matrix and returns the results as an object of class prcomp
PCA_Comp_Scaled_Centered <- prcomp(t(BetaValues_51Samples_20K), center = TRUE, scale=T)  # scale =T is appropriate for high-dimens
summary(PCA_Comp_Scaled_Centered)
```

This is R

The language of data science

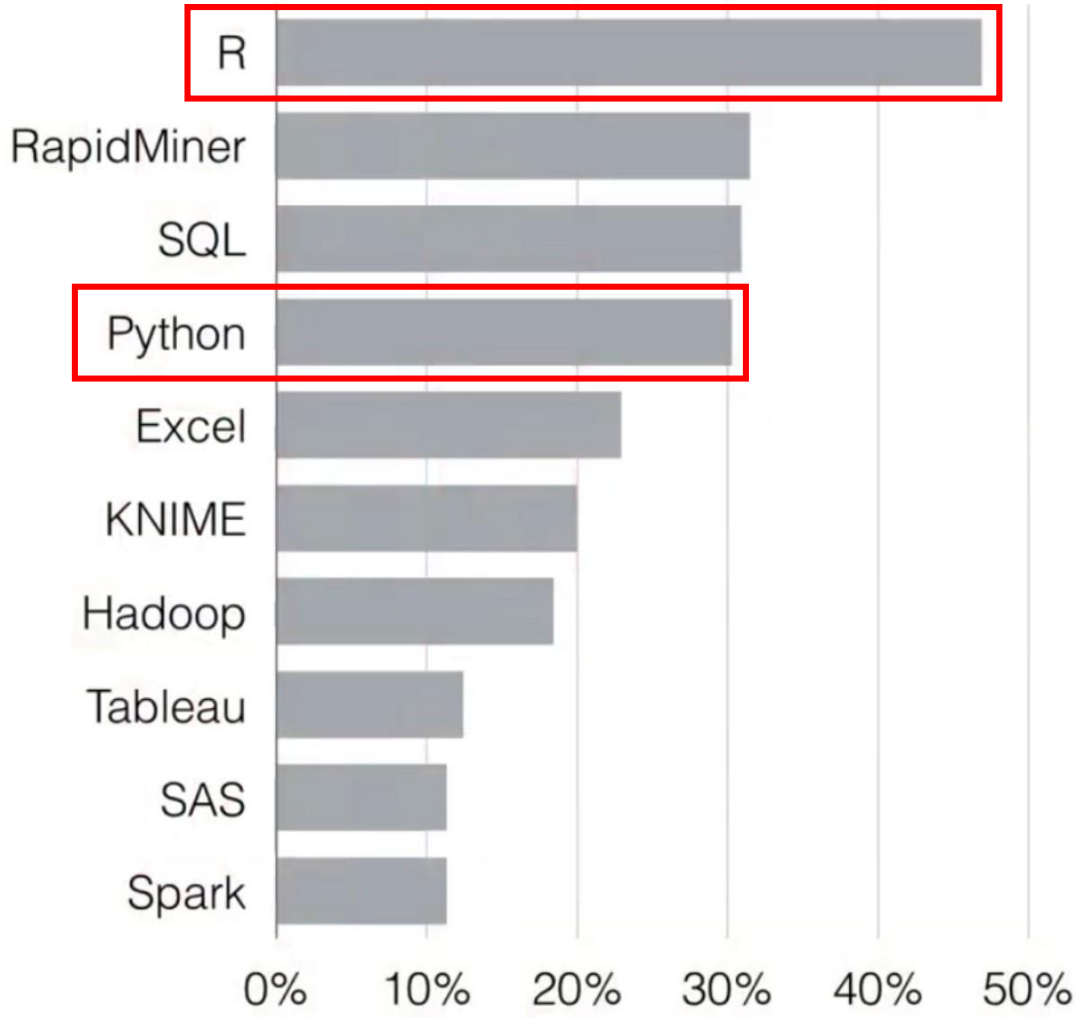
R ranking



Ranking

- Survey of data mining experts
- R is first
- 50% more use than Python

R ranking



Ranking

- Survey of data mining experts
- R is first
- 50% more use than Python

R

Free & open source

Vector operations

Great community

9000+ packages

R Programming – general properties

- An **open source** programming language freely available under the GNU public license (written in primarily C and Fortran). It's an **interpreted** language.
- A **powerful tool** to work with data (**statistical computing** and **data analysis**).
- **Highly extensible** through the use of user-submitted **packages** for specific functions or specific areas of study.
- Including **thousands of packages**, designed, maintained, and widely used by **data scientist** and **statisticians**.
- Advanced users can write C, C++, Java, .NET or Python code to manipulate R objects directly.
- **Portable** and works equally well on Windows, Linux, Mac OS.
- R has stronger **object-oriented programming facilities** than most statistical computing languages.
- Writing and running R scripts could be either in
  - **The command line interface** or a Graphical User Interface called **RStudio**.

R Programming; R packages

- A great advantage to the open source nature of R is that users have contributed a huge number of packages for solving a vast majority of data analysis problems.
- For example, there are packages specifically directed to **visualise data**, **non-parametric statistics**, **signal processing**, **bioinformatics** and so on.

- **Two steps** to use any packages.
  - First, they **must be installed on your system**, and this is an **one-time** step.
  - Second, once a package is installed, it **must be loaded** by calling the ***library()*** function with the name of the library as an argument.

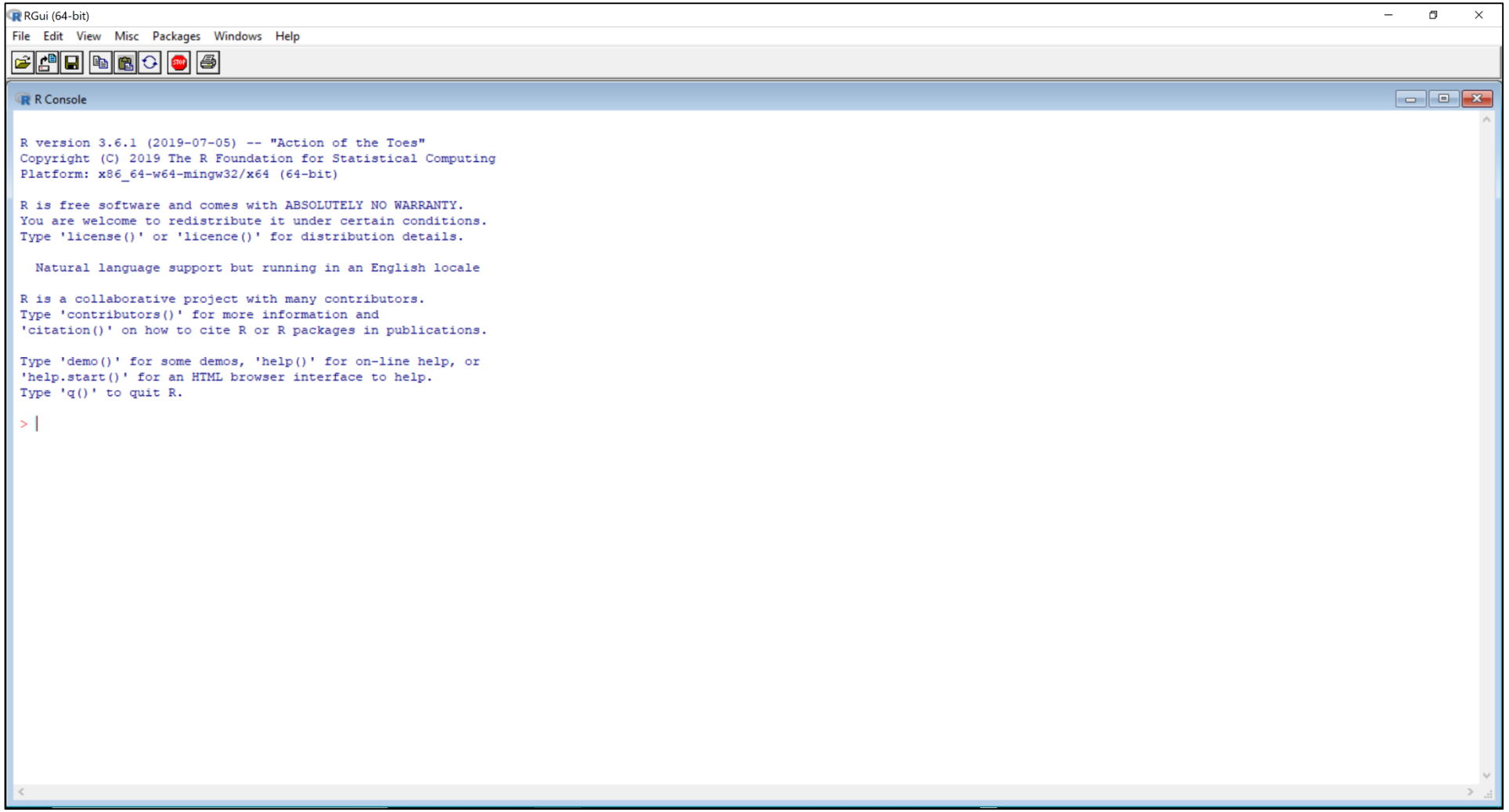
- Some examples of using installed packages in R:
  - library(NMF) # Nonnegative Matrix Factorization (Algorithms and Framework) package
  - library(mclust) # Gaussian Mixture Modelling package for Model-Based Clustering, Classification, and Density Estimation
  - library(shiny) # Web Application Framework for R

- **CRAN** (The **C**omprehensive **R** **A**rchive **N**etwork)
  - https://cran.r-project.org/

R Programming; The command line interface

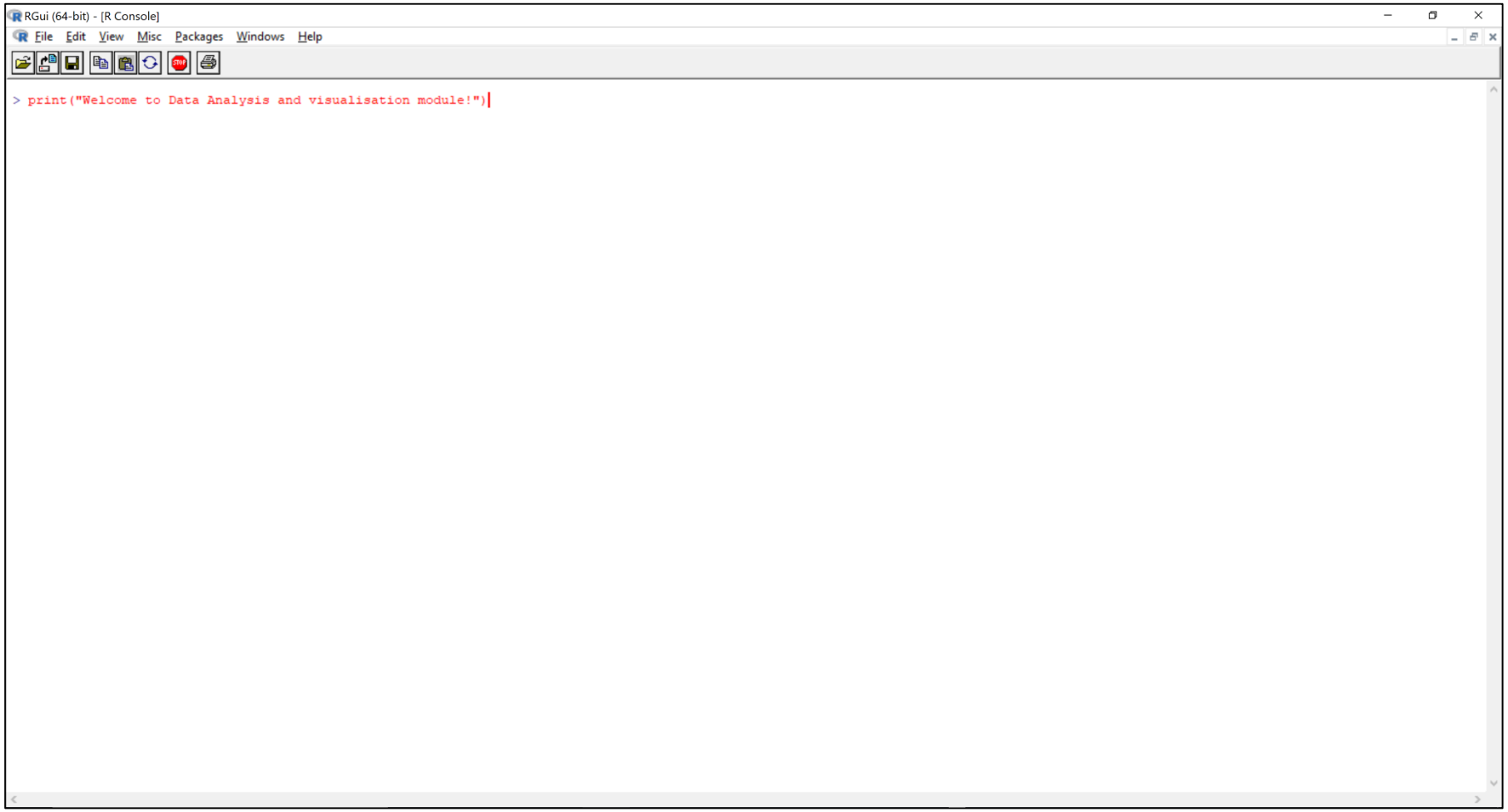
- **The command line interface**

R Console



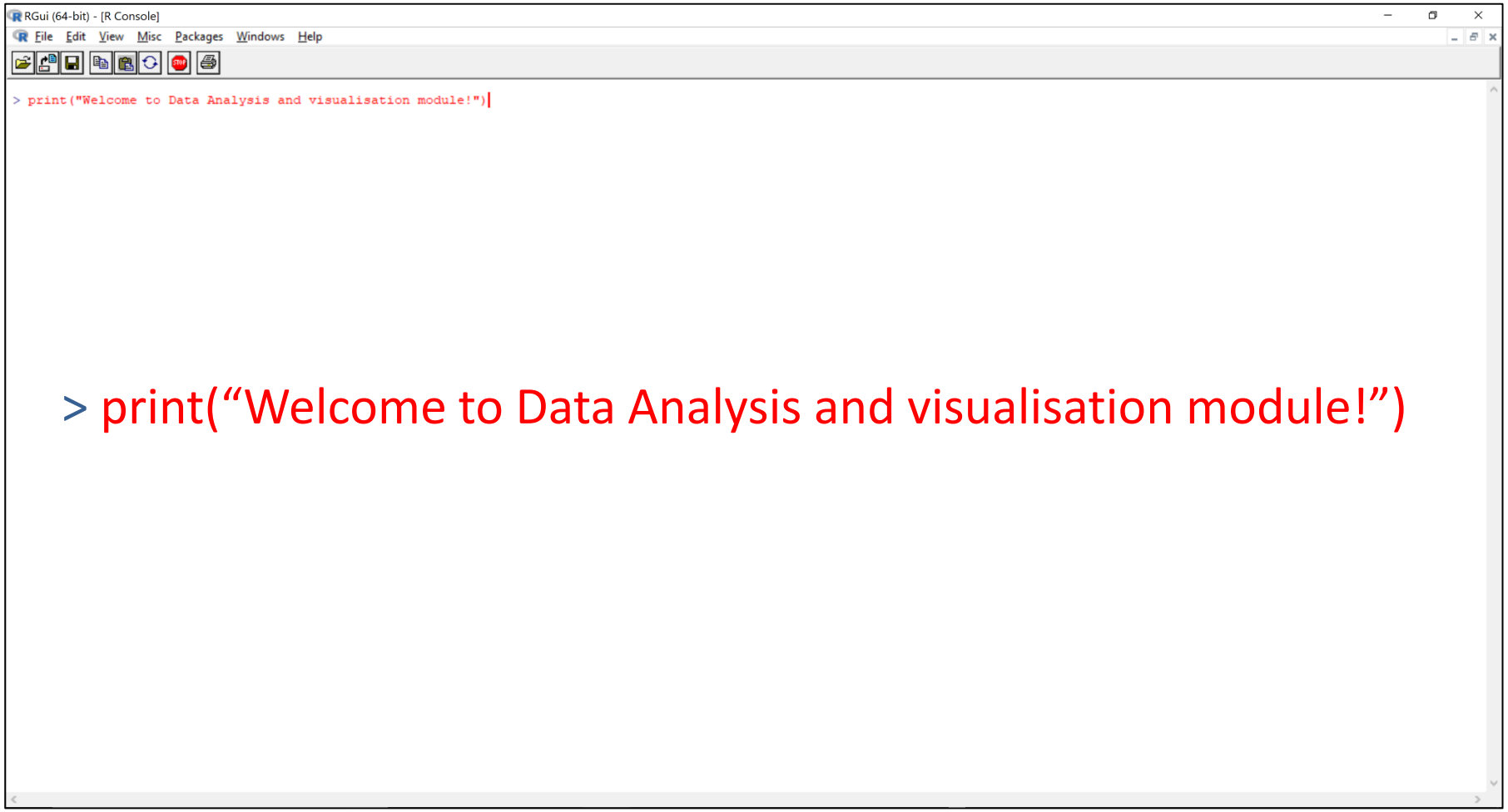
R Programming; The command line interface

- The command line interface
- To clear the console: **CTRL + Enter**



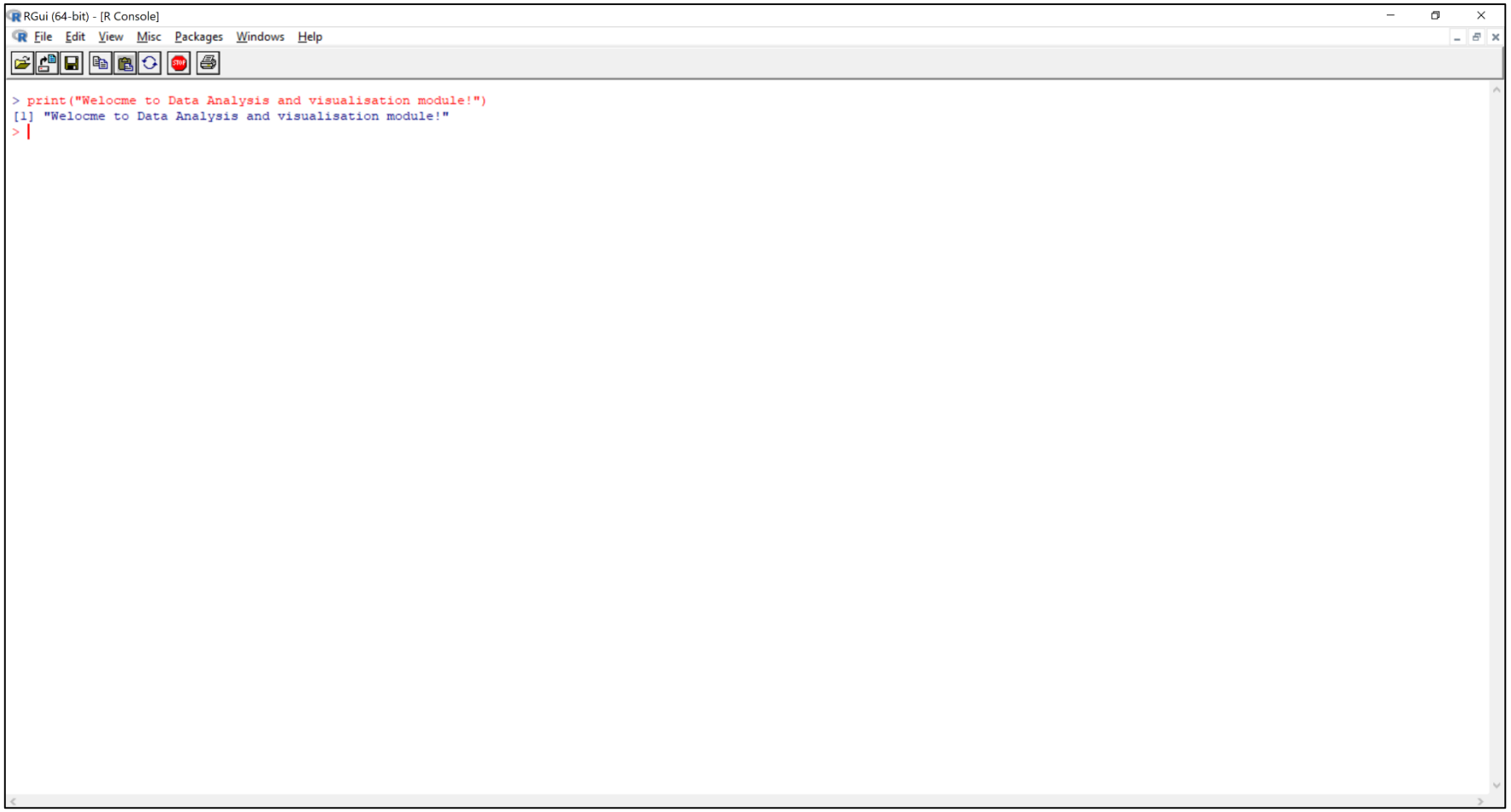
R Programming; The command line interface

- The command line interface
- To clear the console: **CTRL + Enter**



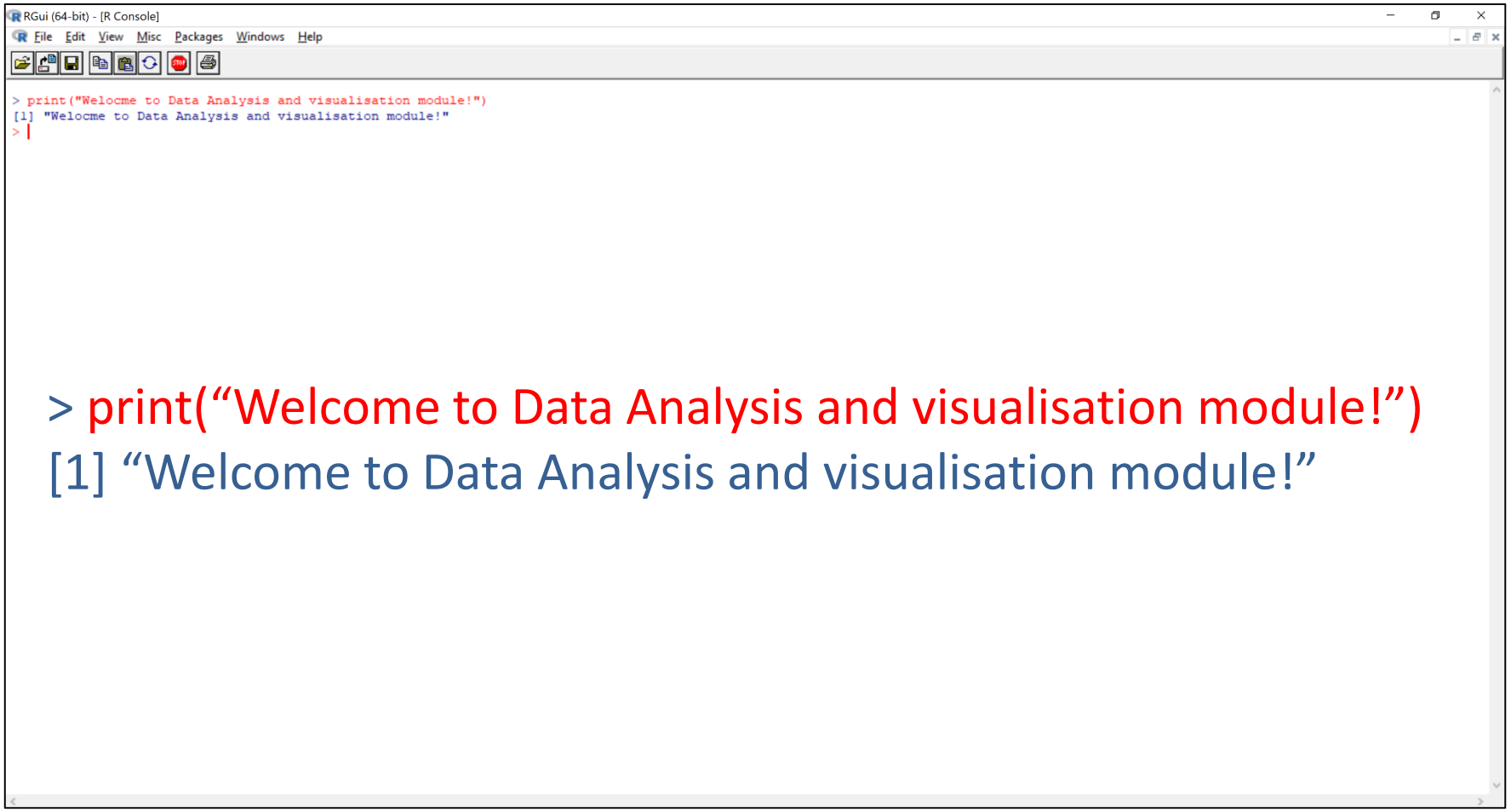
R Programming; The command line interface

- **The command line interface**
- To clear the console: **CTRL + Enter**



R Programming; The command line interface

- **The command line interface**
- To clear the console: **CTRL + Enter**



R Programming; The command line interface

- The command line interface
- R is a case-sensitive language (similar to Unix/Linux)

```
> Print("Welcome to Data Analysis and visualisation module!")
Error in Print("Welcome to Data Analysis and visualisation module!") :
  could not find function "Print"
>
```

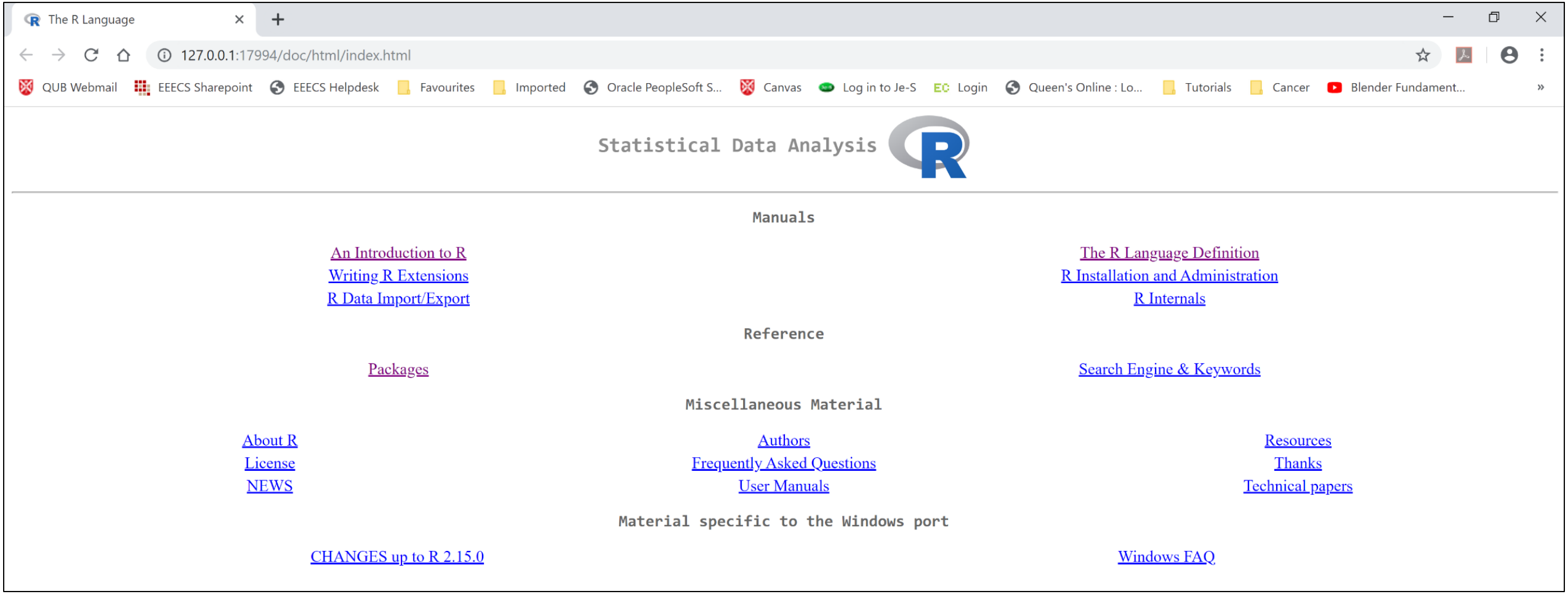
R Programming; help.start() command

- **The command line interface**
- R is a case-sensitive language (similar to Unix/Linux)
- print() is an R function

R Programming; help.start() command

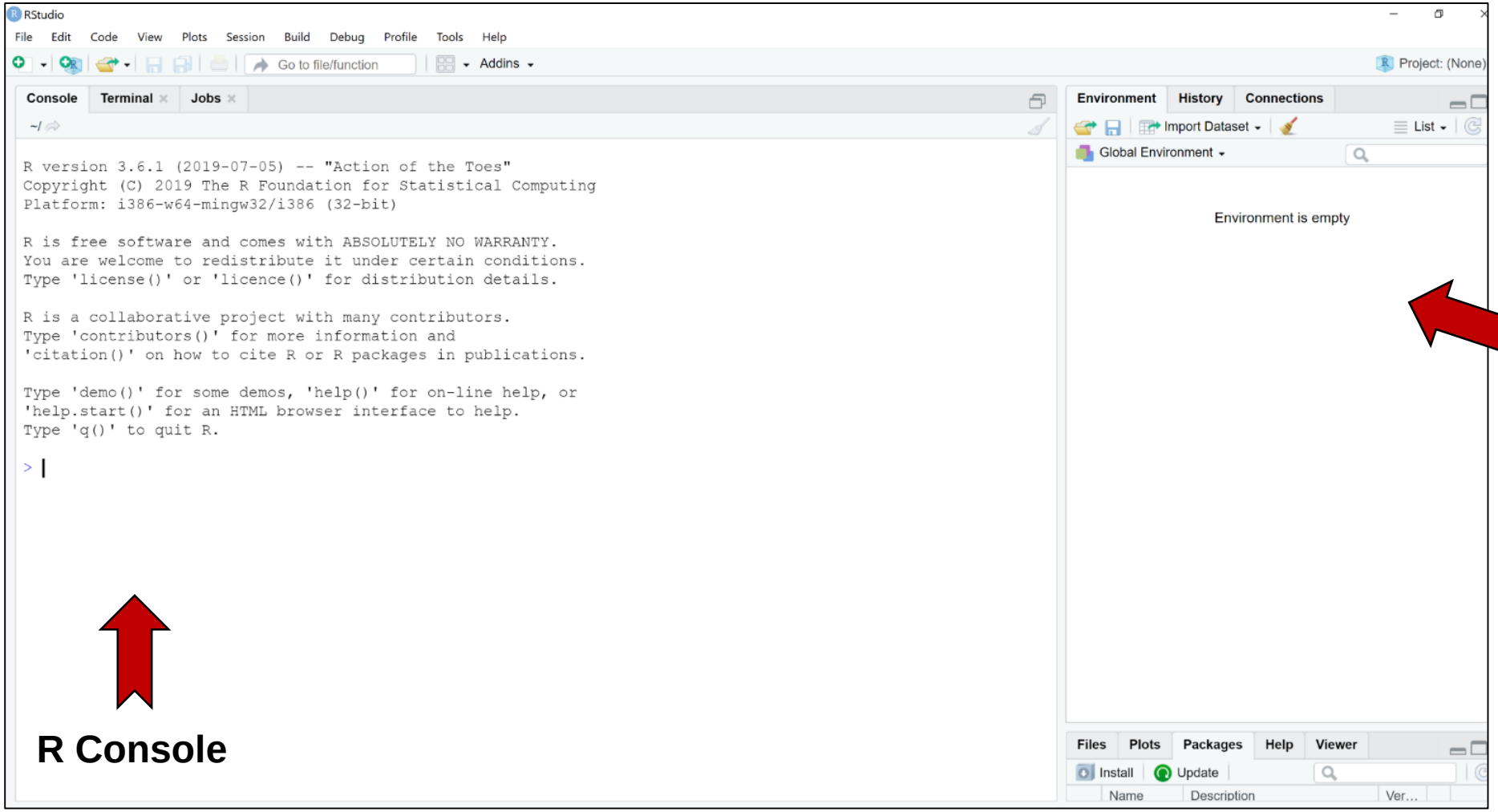
- **The command line interface**
- R is a case-sensitive language (similar to Unix/Linux)

- print() is an R function
- help.start()



R Programming; RStudio

- **RStudio**



R Console

Environment

How to install R & RStudio

- R refers to a software environment that comes with a GUI (Graphical User Interface). R GUI looks more similar to the old DOS console than to SPSS or Stata.
  - For different OS (Linux, Win or Mac, etc. users): https://cran.r-project.org/
- RStudio is an IDE (Integrated Development Environment) that makes R easier to use. It includes a code editor, debugging and visualization tools.
  - For different platforms : https://rstudio.com/products/rstudio/download/
- To download and install R and RStudio (Win or Mac)
  - Step by step: ttps://courses.edx.org/courses/UTAustinX/UT.7.01x/3T2014/56c5437b88fa43cf828bff5371c6a924/

R Programming; R-object

- In contrast to other programming languages like C and java, the variables in R are **not declared** as some data type. The **variables** are assigned with **R-Objects** and the data type of the R-object becomes the data type of the variable. There are many types of R-objects.

- The frequently used **R-Objects**:
  - Vectors
  - Lists
  - Matrices
  - Arrays
  - Factors
  - Dataframes

R Programming; Vectors

- The frequently used **R-Objects**:
  - Vectors
  - Lists
  - Matrices
  - Arrays
  - Factors
  - Dataframes

R Programming; Working directory & workspace

The working directory is the default place where R looks for files that are read from disk, or written to disk. The current working directory is obtained with:

```
> getwd()
[1] "C:/Users/1234567/Documents/Rwork"
```

We can also set the working directory using the function setwd()

```
> setwd("C:/Users/1234567/Documents")
```

Saving current session of R (workspace including all objects in memory) by save.image()

```
> save.image("myWspace1.RData")
```

Alternatively, you could use function load() to load your already saved workspace

```
> load("C:/Users/1234567/Documents/myWspace1.RData")
```

In the line below, we avoid R asking again whether it should save the workspace when using quit.

```
> q(save = "no")
```

R Programming; Vectors

Definition:

A string or numbers, sequential numbers, random numbers and so on

```
#-----------------------------------------------------------------
# Some examples (running in the console)
#-----------------------------------------------------------------
> VecNum1 <- vector(length=10, mode= "double")
[1] 0 0 0 0 0 0 0 0 0 0
#-----------------------------------------------------------------
> VecLog <- vector(length= 5)
[1] FALSE FALSE FALSE FALSE FALSE
#-----------------------------------------------------------------
> VecNum2 <- c(0,0,0,0,0,0,0,0,0,0)      # the easiest way to create a vector using c()
> VecNum2 <- c(rep(10,x=0))  # replicates elements of vectors and lists
[1] 0 0 0 0 0 0 0 0 0 0
#-----------------------------------------------------------------
> SeqVec <- 1:100      # creates a sequence of numbers (from 1 to 100) – consecutive numbers
[1] 1 2 3 … 100
```

R Programming; Vectors & access to the elements

```
> x_vector <- seq(8,20,length.out=6)
> x_vector
[1] 8.0 10.4 12.8 15.2 17.6 20.0
#-----------------------------------------------------------------
# Access to 3rd element of x_vector
> x_vector[3]
[1] 12.8
#-----------------------------------------------------------------
# How to access to 2nd and 4th element of x_vector?

> x_vector[2,4]    # is it a correct call?
```

R Programming; Vectors & access to the elements

```
> x_vector <- seq(8,20,length.out=6)
> x_vector
[1] 8.0 10.4 12.8 15.2 17.6 20.0
#-----------------------------------------------------------------
# Access to 3rd element of x_vector
> x_vector[3]
[1] 12.8
#-----------------------------------------------------------------
# How to access to 2nd and 4th element of x_vector?

> x_vector[2,4]    # is it a correct call? No

Error in x_vector[2,4] : incorrect number of dimensions

> x_vector[c(2,4)]  # this is the correct one
#-----------------------------------------------------------------
# How to access to all elements but 1st element?
```

R Programming; Vectors & access to the elements

```
> x_vector <- seq(8,20,length.out=6)
> x_vector
[1] 8.0 10.4 12.8 15.2 17.6 20.0
#-----------------------------------------------------------------
# Access to 3rd element of x_vector
> x_vector[3]
[1] 12.8
#-----------------------------------------------------------------
# How to access to 2nd and 4th element of x_vector?

> x_vector[2,4]    # is it a correct call? No

Error in x_vector[2,4] : incorrect number of dimensions

> x_vector[c(2,4)]  # this is the correct one
#-----------------------------------------------------------------
# How to access to all elements but 1st element?
> x_vector[-1]    # [1] 10.4 12.8 15.2 17.6 20.0
```

R Programming; Vectors & access to the elements

```
> x_vector <- seq(8,20,length.out=6)
> x_vector
[1] 8.0 10.4 12.8 15.2 17.6 20.0
> typeof(x_vector)

> length(x_vector)

#-----------------------------------------------------------------
> x_vector[c(2.1,4.5)]      # real numbers are truncated to integers

#-----------------------------------------------------------------
> x_vector_char <- c(11, 50, TRUE, 'hello')     # what if we use this: c(11, 50, TRUE, "hello") ?

> typeof(x_vector_char)

#-----------------------------------------------------------------
> x_vector <- seq(1,3,by=0.2)  # specify step size
```

R Programming; Vectors & access to the elements

```
> x_vector <- seq(8,20,length.out=6)
> x_vector
[1] 8.0 10.4 12.8 15.2 17.6 20.0
> typeof(x_vector)
[1] "double"
> length(x_vector)
[1] 6
#-----------------------------------------------------------------
> x_vector[c(2.1,4.5)]      # real numbers are truncated to integers
[1] 10.4 15.2
#-----------------------------------------------------------------
> x_vector_char <- c(11, 50, TRUE, 'hello')     # what if we use this: c(11, 50, TRUE, "hello") ?
[1] "11"   "50"   "TRUE"   "hello"
> typeof(x_vector_char)
[1] "character"
#-----------------------------------------------------------------
> x_vector <- seq(1,3,by=0.2)  # specify step size
[1] 1.0  1.2  1.4  1.6  1.8  2.0  2.2  2.4  2.6  2.8 3.0
```

R Programming; Vectors & access to the elements

```
> x_vec_logic <- c(TRUE,FALSE,TRUE,FALSE,FALSE)
> x_vec_logic
[1] TRUE   FALSE   TRUE   FALSE   FALSE
> typeof(x_vec_logic)
[1] "logical"
> length(x_vec_logic)
[1]  5
#-------------------------------------------------------------
```

R Programming; Operators

Arithmetic operators	+	-	*	/	%%	%/%	^
Relational operators	<	>	==	<=	>=	!=	
Logical operators	&	|	!	&&	||		
Assignment operators	=	<-	->	<<-	->>		
Misc. operators	:	%in%	%*%				

R Programming; Arithmetic operators

Operator	Description	Usage
+	Addition of two operands	a + b
–	Subtraction of second operand from first	a – b
*	Multiplication of two operands	a * b
/	Division of first operand with second	a / b
%%	Remainder from division of first operand with second	a %% b
%/%	Quotient from division of first operand with second	a %/% b
^	First operand raised to the power of second operand	a^b

R Programming; Arithmetic operators

```
# R Arithmetic Operators Example for integers

a <- 7.5
b <- 2

print ( a+b ) #addition
print ( a-b ) #subtraction
print ( a*b ) #multiplication
print ( a/b ) #Division
print ( a%%b ) #Reminder
print ( a%/%b ) #Quotient
print ( a^b ) #Power of
```

Output

```
[1] 9.5
[1] 5.5
[1] 15
[1] 3.75
[1] 1.5
[1] 3
[1] 56.25
```

R Programming; Arithmetic operators

```
# R Operators - R Arithmetic Operators Example for vectors

a <- c(8, 9, 6)
b <- c(2, 4, 5)

print ( a+b ) #addition
print ( a-b ) #subtraction
print ( a*b ) #multiplication
print ( a/b ) #Division
print ( a%%b ) #Reminder
print ( a%/%b ) #Quotient
print ( a^b ) #Power of
```

Output

```
[1] 10 13 11
[1] 6 5 1
[1] 16 36 30
[1] 4.00 2.25 1.20
[1] 0 1 1
[1] 4 2 1
[1] 64 6561 7776
```

R Programming; Relational operators

Operator	Description	Usage
<	Is first operand **less than** second operand	a < b
>	Is first operand **greater than** second operand	a > b
==	Is first operand **equal to** second operand	a == b
<=	Is first operand **less than** or equal to second operand	a <= b
>=	Is first operand **greater than** or equal to second operand	a > = b
!=	Is first operand **not equal** to second operand	a!=b

R Programming; Relational operators

```
# R Operators - R Relational Operators Example for Numbers

a <- 7.5
b <- 2

print ( a<b ) # less than
print ( a>b ) # greater than
print ( a==b ) # equal to
print ( a<=b ) # less than or equal to
print ( a>=b ) # greater than or equal to
print ( a!=b ) # not equal to
```

Output

```
[1] FALSE
[1] TRUE
[1] FALSE
[1] FALSE
[1] TRUE
[1] TRUE
```

R Programming; Relational operators

```
# R Operators - R Relational Operators Example for Numbers

a <- c(7.5, 3, 5)
b <- c(2, 7, 0)

print ( a<b ) # less than
print ( a>b ) # greater than
print ( a==b ) # equal to
print ( a<=b ) # less than or equal to
print ( a>=b ) # greater than or equal to
print ( a!=b ) # not equal to
```

Output

```
[1] FALSE TRUE FALSE
[1] TRUE FALSE TRUE
[1] FALSE FALSE FALSE
[1] FALSE TRUE FALSE
[1] TRUE FALSE TRUE
[1] TRUE TRUE TRUE
```

R Programming; Logical operators

Operator	Description	Usage
&	Element wise logical AND operation.	a & b
|	Element wise logical OR operation.	a | b
!	Element wise logical NOT operation.	!a
&&	Operand wise logical AND operation.	a && b
||	Operand wise logical OR operation.	a || b

R Programming; Logical operators

```
# R Operators - R Logical Operators Example for basic logical
elements

a <- 0 # logical FALSE
b <- 2 # logical TRUE

print ( a & b ) # logical AND element wise
print ( a | b ) # logical OR element wise
print ( !a ) # logical NOT element wise
print ( a && b ) # logical AND consolidated for all elements
print ( a || b ) # logical OR consolidated for all elements
```

Output

```
[1] FALSE
[1] TRUE
[1] TRUE
[1] FALSE
[1] TRUE
```

R Programming; Logical operators

```
# R Operators - R Logical Operators Example for boolean vectors

a <- c(TRUE, TRUE, FALSE, FALSE)
b <- c(TRUE, FALSE, TRUE, FALSE)

print ( a & b ) # logical AND element wise
print ( a | b ) # logical OR element wise
print ( !a ) # logical NOT element wise
print ( a && b ) # logical AND consolidated for all elements
print ( a || b ) # logical OR consolidated for all elements
```

Output

```
[1] TRUE FALSE FALSE FALSE
[1] TRUE TRUE TRUE FALSE
[1] FALSE FALSE TRUE TRUE
[1] TRUE
[1] TRUE
```

R Programming; Logical operators

& vs. &&

```
> ((-2:2) >= 0) & ((-2:2) <= 0)
[1] FALSE  FALSE  TRUE  FALSE  FALSE
```

-2 -1 0 1 2

```
> ((-2:2) >= 0) && ((-2:2) <= 0)
[1] FALSE
```

& is vectorised

R Programming; Assignment operators

Operator	Description	Usage
=	Assigns right side value to left side operand	a = 3
<-	Assigns right side value to left side operand	a <- 5
->	Assigns left side value to right side operand	4 -> a
<<-	Assigns right side value to left side operand	a <<- 3.4
->>	Assigns left side value to right side operand	c(1,2) ->> a

R Programming; Assignment operators

```
# R Operators - R Assignment Operators

a = 2
print ( a )

a <- TRUE
print ( a )

454 -> a
print ( a )

a <<- 2.9
print ( a )

c(6, 8, 9) -> a
print ( a )
```

Output

```
[1] 2
[1] TRUE
[1] 454
[1] 2.9
[1] 6 8 9
```

R Programming; Misc. operators

Operator	Description	Usage
:	Creates series of numbers from left operand to right operand	a:b
%in%	Identifies if an element(a) belongs to a vector(b)	a %in% b
%*%	Performs multiplication of a vector with its transpose	A %*% t(A)

R Programming; Misc. operators

```
# R Operators - R Misc Operators

a = 23:31
print ( a )

a = c(25, 27, 76)
b = 27
print ( b %in% a )
```

```
M = matrix(c(1,2,3,4), 2, 2, TRUE)
print ( M %*% t(M) )
```

Output

```
[1] 23 24 25 26 27 28 29 30 31
[1] TRUE
     [,1] [,2]
[1,]    5   11
[2,]   11   25
```

R Programming; Lists

- The frequently used **R-Objects**:
  - Vectors
  - Lists
  - Matrices
  - Arrays
  - Factors
  - Dataframes