

R Software

Department of Science and Humanities
K. J. Somaiya College Engineering Mumbai

What is R ?

- * R is an environment for data manipulation, statistical computing, graphics display and data analysis
- * **Effective data handling and storage of outputs is possible.**
- * **Simple as well as complicated calculations are possible**
- * **Graphical display on-screen and hardcopy are possible**

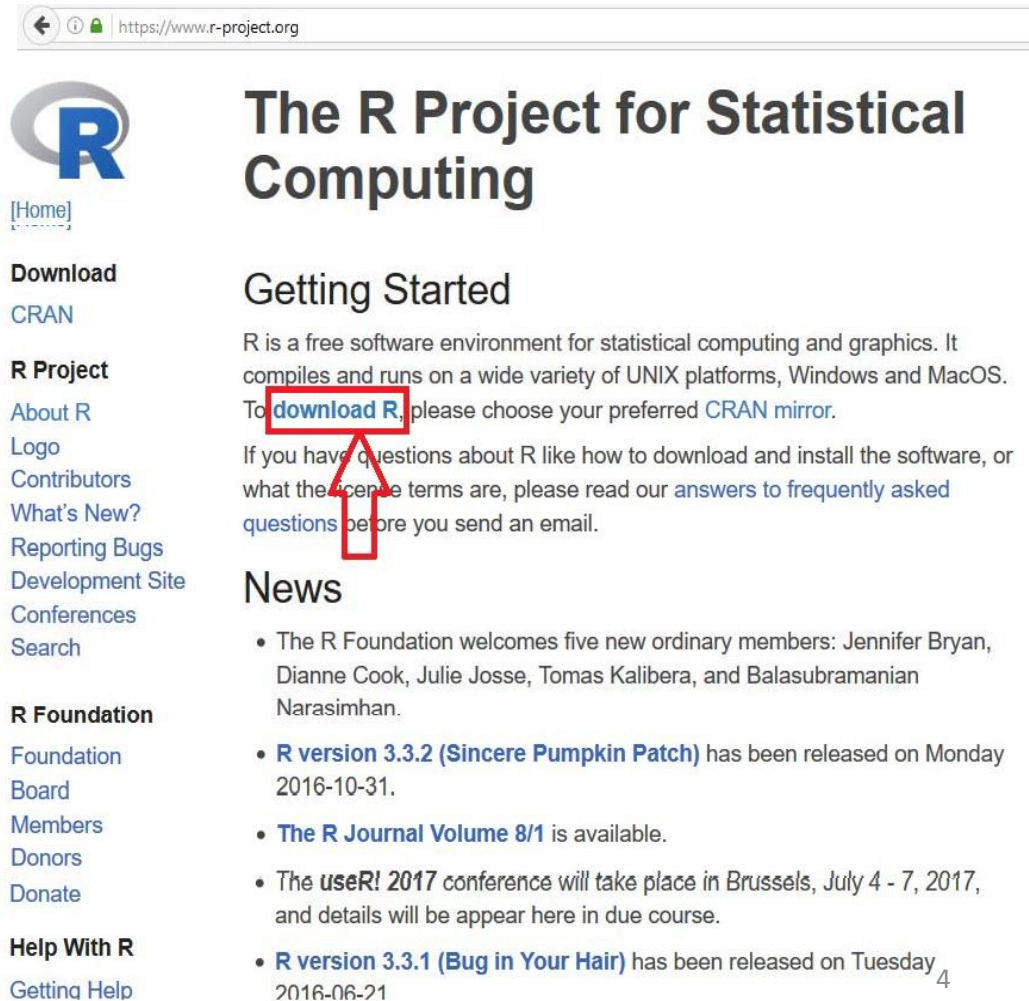
Why R ?

- * R is free.
- * Many statistical packages are freely available.
- * It has a computer language which is convenient to use for statistical and graphical applications.
- * Many people, research and design offices, analytical firms have started using R

Installing R


You may install R in a windows or Apple Computer by downloading from <https://www.r-project.org>

Click on [download R](#)



The screenshot shows the homepage of The R Project for Statistical Computing. The browser address bar at the top displays 'https://www.r-project.org'. The page features a large blue 'R' logo with a '[Home]' link below it. A left-hand navigation menu lists various links: 'Download CRAN', 'R Project' (with sub-links for 'About R', 'Logo', 'Contributors', 'What's New?', 'Reporting Bugs', 'Development Site', 'Conferences', and 'Search'), 'R Foundation' (with sub-links for 'Foundation', 'Board', 'Members', 'Donors', and 'Donate'), and 'Help With R' (with a link for 'Getting Help'). The main content area is titled 'The R Project for Statistical Computing' and 'Getting Started'. It describes R as a free software environment for statistical computing and graphics, available on various platforms. A red rectangle highlights the 'download R' link, with a red arrow pointing to it from below. The text continues: 'To [download R](#), please choose your preferred CRAN mirror.' It also provides information for those with questions, directing them to 'answers to frequently asked questions' before sending an email. Below this is a 'News' section with several bullet points: 'The R Foundation welcomes five new ordinary members: Jennifer Bryan, Dianne Cook, Julie Josse, Tomas Kalibera, and Balasubramanian Narasimhan.', 'R version 3.3.2 (Sincere Pumpkin Patch) has been released on Monday 2016-10-31.', 'The R Journal Volume 8/1 is available.', 'The useR! 2017 conference will take place in Brussels, July 4 - 7, 2017, and details will be appear here in due course.', and 'R version 3.3.1 (Bug in Your Hair) has been released on Tuesday 2016-06-21'.

https://www.r-project.org

 [Home]

Download
CRAN

R Project
[About R](#)
[Logo](#)
[Contributors](#)
[What's New?](#)
[Reporting Bugs](#)
[Development Site](#)
[Conferences](#)
[Search](#)

R Foundation
[Foundation](#)
[Board](#)
[Members](#)
[Donors](#)
[Donate](#)

Help With R
[Getting Help](#)

The R Project for Statistical Computing

Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To [download R](#), please choose your preferred [CRAN mirror](#).

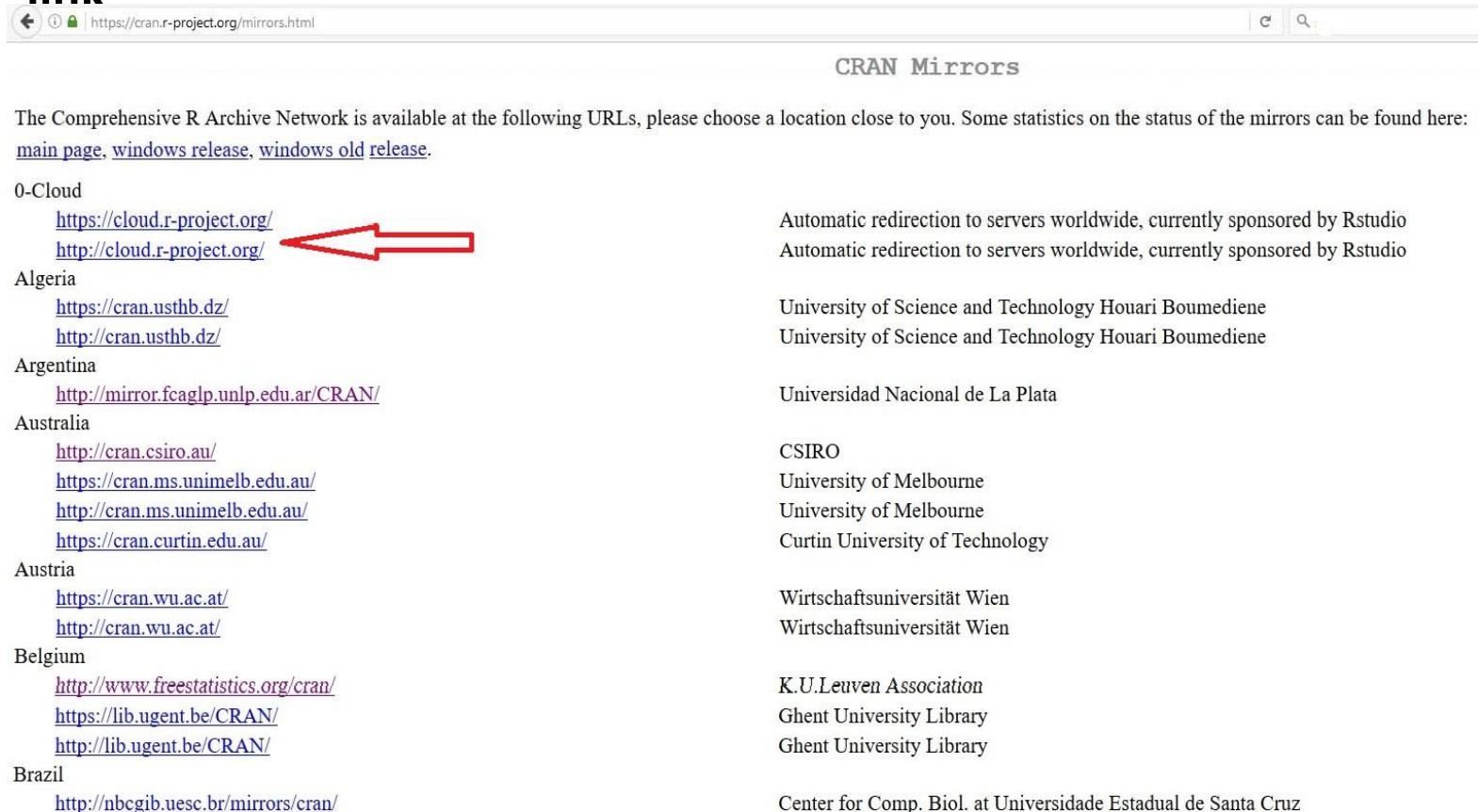
If you have questions about R like how to download and install the software, or what the license terms are, please read our [answers to frequently asked questions](#) before you send an email.

News

- The R Foundation welcomes five new ordinary members: Jennifer Bryan, Dianne Cook, Julie Josse, Tomas Kalibera, and Balasubramanian Narasimhan.
- **R version 3.3.2 (Sincere Pumpkin Patch)** has been released on Monday 2016-10-31.
- **The R Journal Volume 8/1** is available.
- The *useR!* 2017 conference will take place in Brussels, July 4 - 7, 2017, and details will be appear here in due course.
- **R version 3.3.1 (Bug in Your Hair)** has been released on Tuesday 2016-06-21

Installing R

Choose any mirror and click on the link



The screenshot shows the CRAN Mirrors website. The browser address bar displays <https://cran.r-project.org/mirrors.html>. The page title is "CRAN Mirrors". Below the title, a paragraph states: "The Comprehensive R Archive Network is available at the following URLs, please choose a location close to you. Some statistics on the status of the mirrors can be found here: [main page](#), [windows release](#), [windows old release](#)." Below this, a list of mirrors is presented, each with a country name and a list of URLs. A red arrow points to the first mirror link, <https://cloud.r-project.org/>.

Country	URLs	Description
0-Cloud	https://cloud.r-project.org/ http://cloud.r-project.org/	Automatic redirection to servers worldwide, currently sponsored by Rstudio Automatic redirection to servers worldwide, currently sponsored by Rstudio
Algeria	https://cran.usthb.dz/ http://cran.usthb.dz/	University of Science and Technology Houari Boumediene University of Science and Technology Houari Boumediene
Argentina	http://mirror.fcaglp.unlp.edu.ar/CRAN/	Universidad Nacional de La Plata
Australia	http://cran.csiro.au/ https://cran.ms.unimelb.edu.au/ http://cran.ms.unimelb.edu.au/ https://cran.curtin.edu.au/	CSIRO University of Melbourne University of Melbourne Curtin University of Technology
Austria	https://cran.wu.ac.at/ http://cran.wu.ac.at/	Wirtschaftsuniversität Wien Wirtschaftsuniversität Wien
Belgium	http://www.freeststatistics.org/cran/ https://lib.ugent.be/CRAN/ http://lib.ugent.be/CRAN/	K.U.Leuven Association Ghent University Library Ghent University Library
Brazil	http://nbcgib.uesc.br/mirrors/cran/	Center for Comp. Biol. at Universidade Estadual de Santa Cruz

Installing R

Else, download it from the Comprehensive R Archive Network (CRAN) website: <http://cran.r-project.org/>

CRAN

[Mirrors](#)

[What's new?](#)

[Task Views](#)

[Search](#)

About R

[R Homepage](#)

[The R Journal](#)

Software

[R Sources](#)

[R Binaries](#)

[Packages](#)

[Other](#)

Documentation

[Manuals](#)

[FAQs](#)

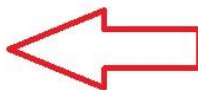
[Contributed](#)

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- [Download R for Linux](#)
- [Download R for \(Mac\) OS X](#)
- [Download R for Windows](#)



R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (Monday 2016-10-31, Sincere Pumpkin Patch) [R-3.3.2.tar.gz](#), read [what's new](#) in the latest version.
- Sources of [R alpha and beta releases](#) (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are [available here](#). Please read about [new features and bug fixes](#) before filing corresponding feature requests or bug reports.
- Source code of older versions of R is [available here](#).
- Contributed extension [packages](#)

Installing R Studio

R studio is a software which helps in running the R software.

Several such editors are available, e.g. Tinn R (<https://sourceforge.net/projects/tinn-r>)

R studio is written in C++ programming language.

R studio is a free and open-source integrate development environment (IDE) for R.

Download R-Studio software from website <https://www.rstudio.com/>

Installing R Studio

https://www.rstudio.com

RStudio

rstudio::conf Products Resources Pricing About Us Blogs

RStudio

Open source and enterprise-ready professional software for R

Announcing RStudio v1.0!

Download RStudio

Discover Shiny

shinyapps.io Login

RStudio

RStudio makes R easier to use. It includes a code editor, debugging & visualization tools.

[Download](#) [Learn More](#)

Shiny

Shiny helps you make interactive web applications for visualizing data. Bring R data analysis to life.

[Learn More](#)

R Packages

Our developers create popular packages to expand the features of R. Includes ggplot2, dplyr, R Markdown & more.

[Learn More](#)

Download and double click on the downloaded file.

Command Line versus Scripts

R Console

```
R version 3.2.3 (2015-12-10) -- "Wooden Christmas-Tree"  
Copyright (C) 2015 The R Foundation for Statistical Computing  
Platform: x86_64-w64-mingw32/x64 (64-bit)
```

```
R is free software and comes with ABSOLUTELY NO WARRANTY.  
You are welcome to redistribute it under certain conditions.  
Type 'license()' or 'licence()' for distribution details.
```

```
Natural language support but running in an English locale
```

```
R is a collaborative project with many contributors.  
Type 'contributors()' for more information and  
'citation()' on how to cite R or R packages in publications.
```

```
Type 'demo()' for some demos, 'help()' for on-line help, or  
'help.start()' for an HTML browser interface to help.  
Type 'q()' to quit R.
```

```
[Previously saved workspace restored]
```

```
> |
```

Type the commands here

This is command line

Command Line versus Scripts

Execution of commands in R is not menu driven. (Not like Clicking over buttons to get outcome)

We need to type the commands.

Single line and multi line commands are possible to write.

When writing multi-line programs, it is useful to use a text editor rather than execute everything directly at the command line.

Command Line versus Scripts

Option 1:

One may use R's own built-in editor.

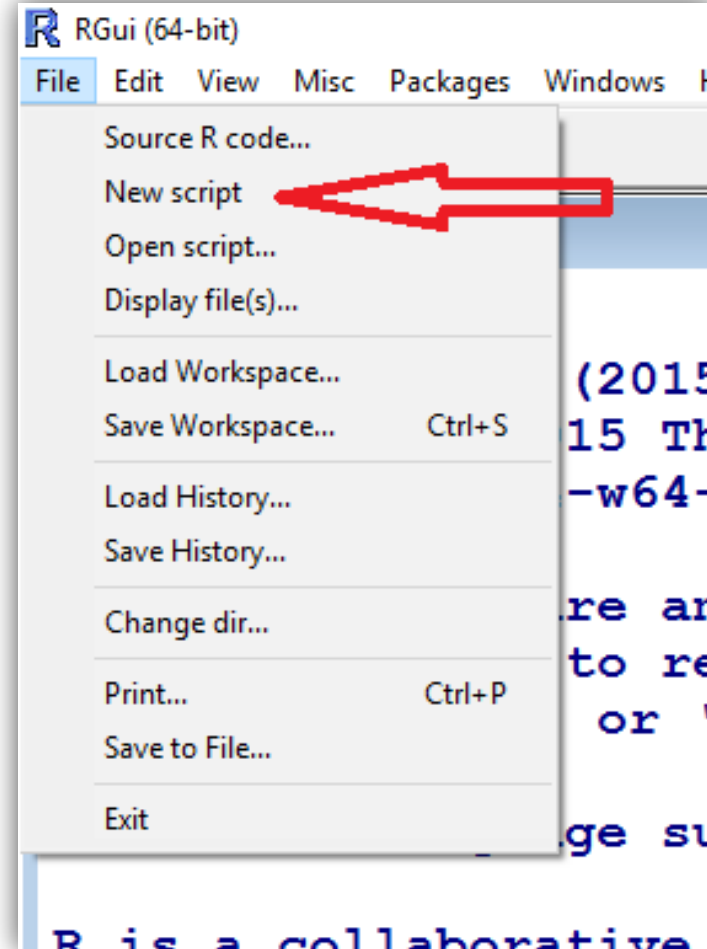
It is accessible from the **RGui** menu bar.

Click **File** and then click on **New script**.

Command Line versus Scripts

At this point R will
open a window entitled
Untitled-R Editor.

We may type and edit in this.



If we want to execute a line or a group of lines, just highlight them and press **Ctrl+R**.

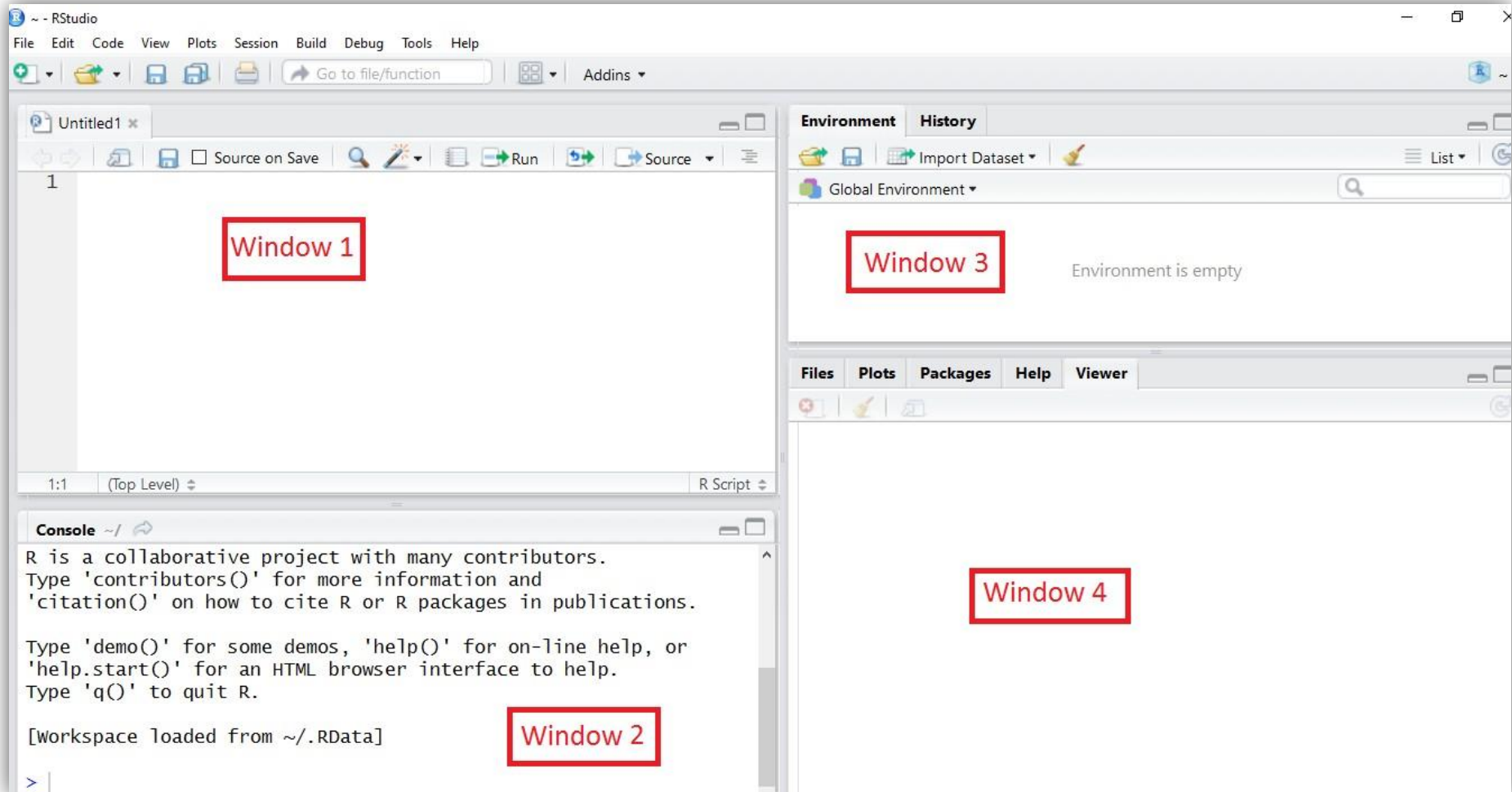
Introduction to R Studio

Option 2:

- * It is an interface between R and us.
- * More useful for beginners.
- * It makes coding easier.
- * When we start R studio, we see 4 windows

Introduction to R Studio

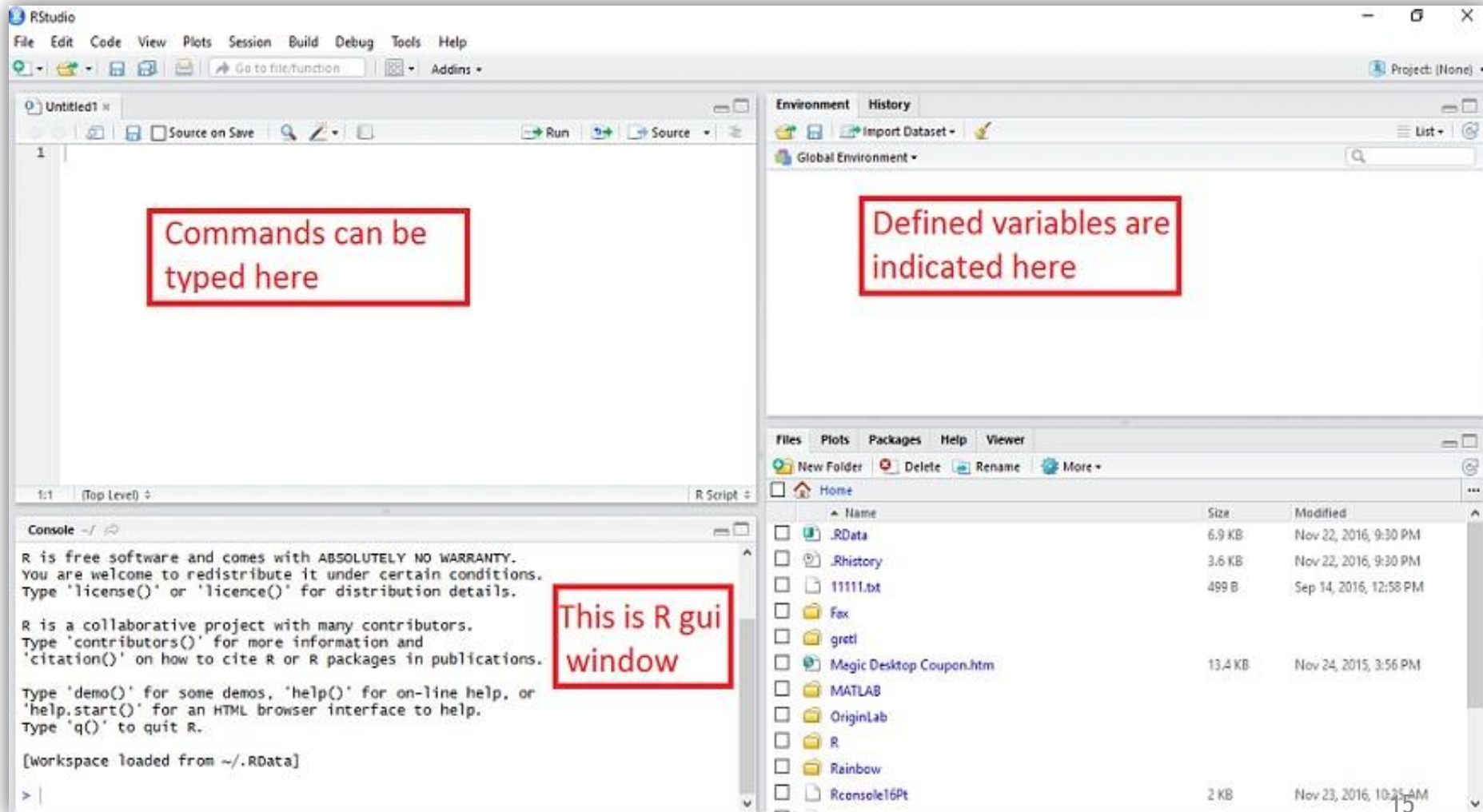
First opening window of Rstudio is as follows having four windows.



Command Line versus Scripts

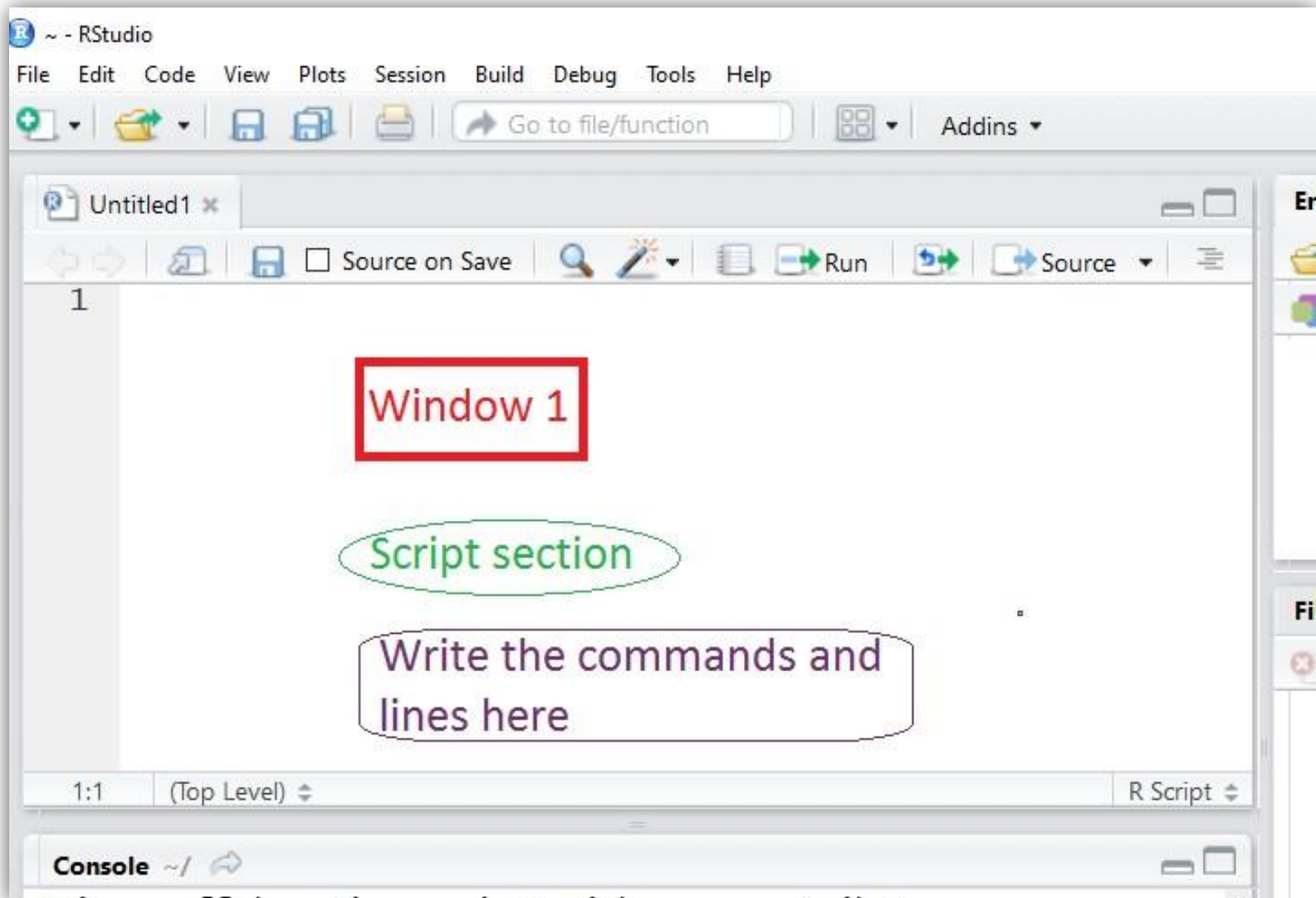
Option 2:

Use R studio software.



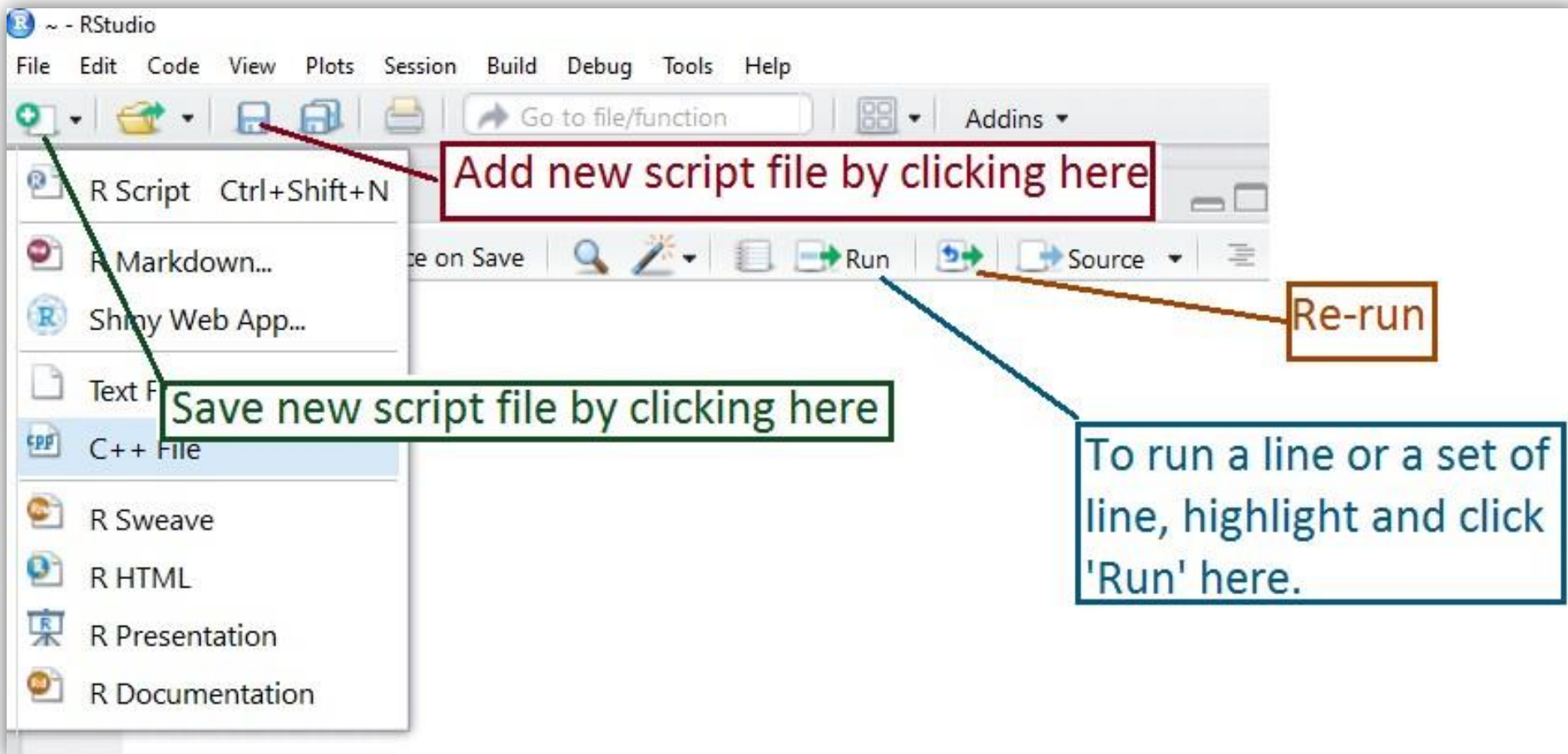
Introduction to R Studio

Description of Window 1



Introduction to R Studio

Description of Window 1



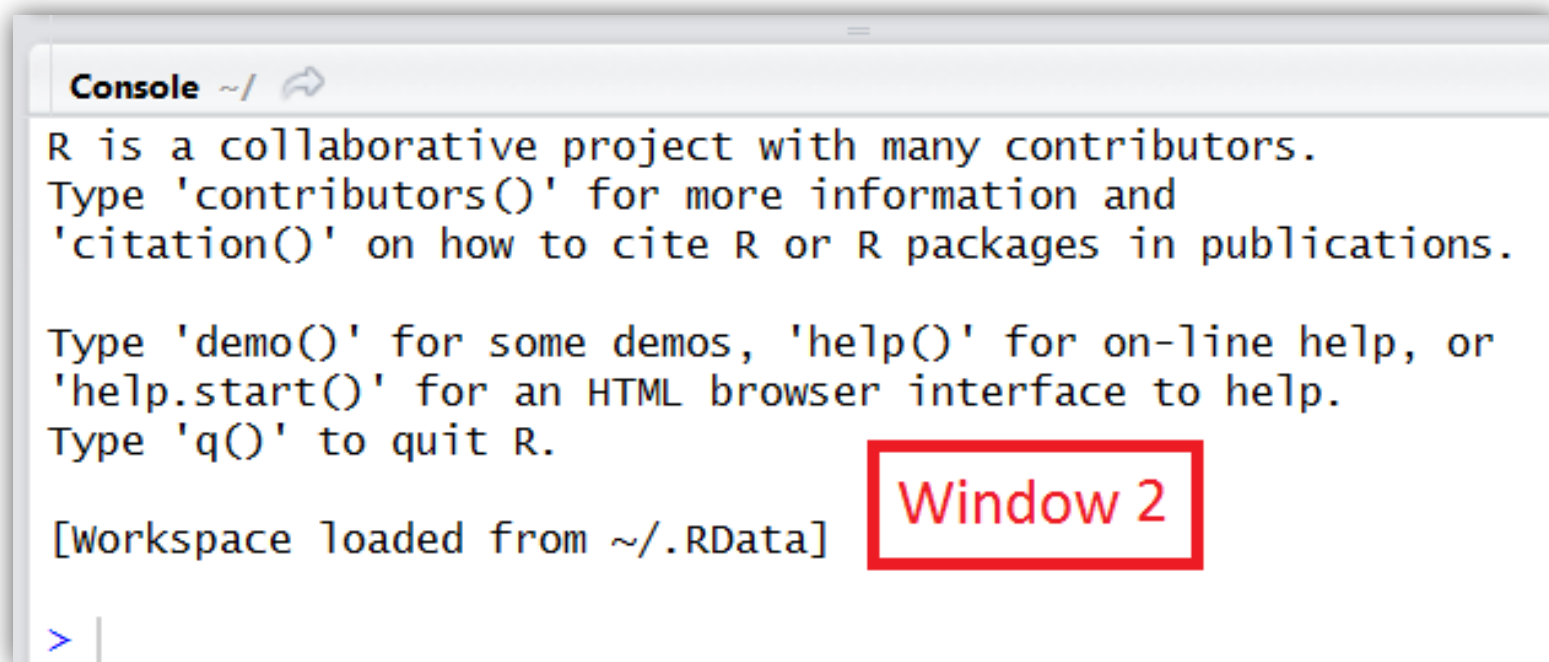
Introduction to R Studio


Description of Window 2 : Console

R program window appears here.

Calculations take place in console window.

One can write programmes in console also but it is hard to make corrections and experiments with the coding.



```
Console ~/   
R is a collaborative project with many contributors.  
Type 'contributors()' for more information and  
'citation()' on how to cite R or R packages in publications.  
  
Type 'demo()' for some demos, 'help()' for on-line help, or  
'help.start()' for an HTML browser interface to help.  
Type 'q()' to quit R.  
  
[Workspace loaded from ~/.RData]  
> |
```

Window 2

Introduction to R Studio

Description of Window 3 : Environment window

All the variables and objects used in the programme appear here. The nature and values of variables and objects also appear here.

The screenshot shows the R Studio Environment window. At the top, there are two tabs: 'Environment' and 'History'. Below the tabs, there is a toolbar with icons for file operations and a button labeled 'Import Dataset'. The main area of the window is divided into two sections: 'Global Environment' and 'Values'. The 'Values' section displays a table of variables and their values. In this case, the variable 'x' has the value '1'. Several callout boxes provide additional information:

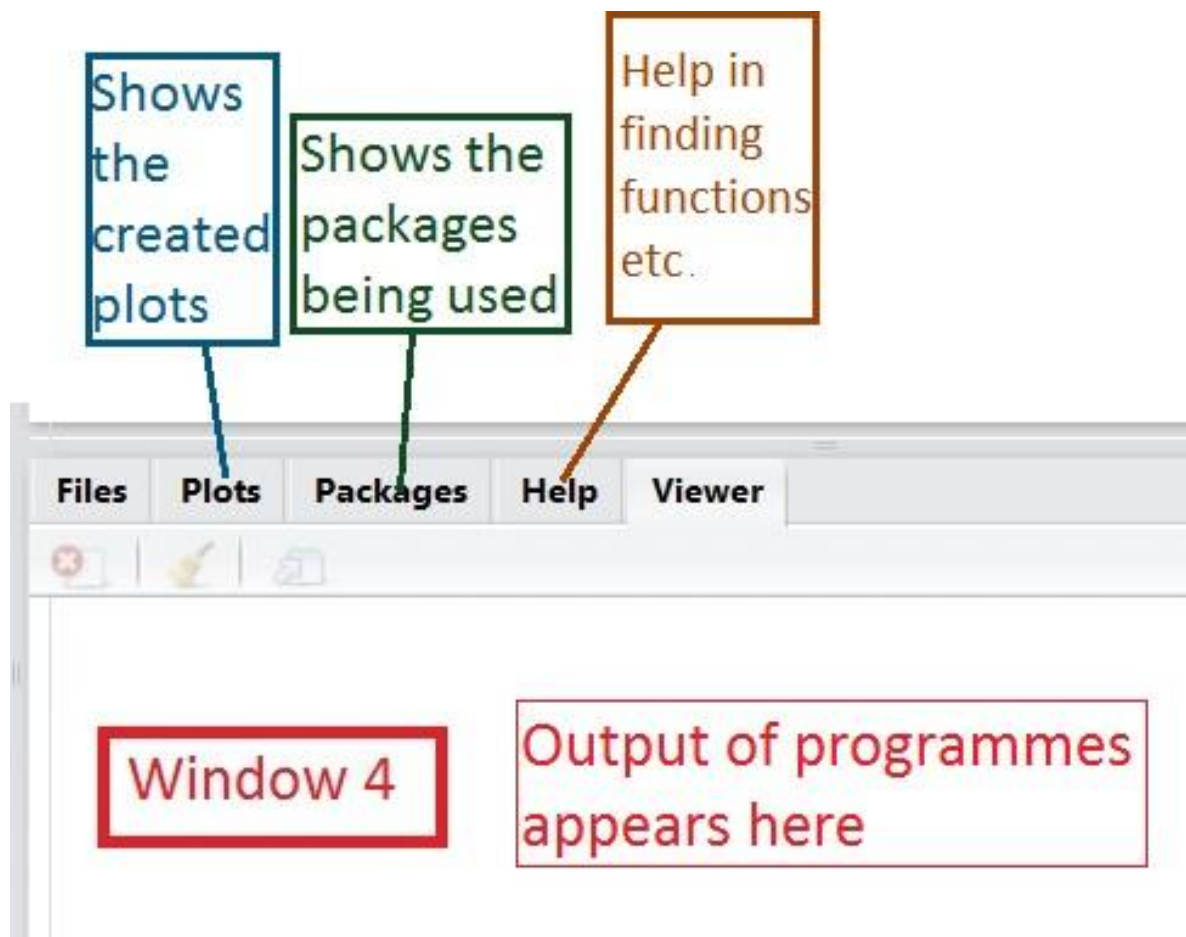
- A blue box with the text 'History tells about the codes used earlier.' points to the 'History' tab.
- A blue box with the text 'Stored value can be erased from here' points to the 'x' variable in the 'Values' table.
- A green box with the text 'The stored value x = 1 appears here' points to the value '1' in the 'Values' table.
- A blue box with the text 'Data can be imported from other files by clicking here' points to the 'Import Dataset' button.

Window 3

Introduction to R Studio

Description of Window 4 : Output window

The output of programmes appears in this window.



Introduction to R Studio

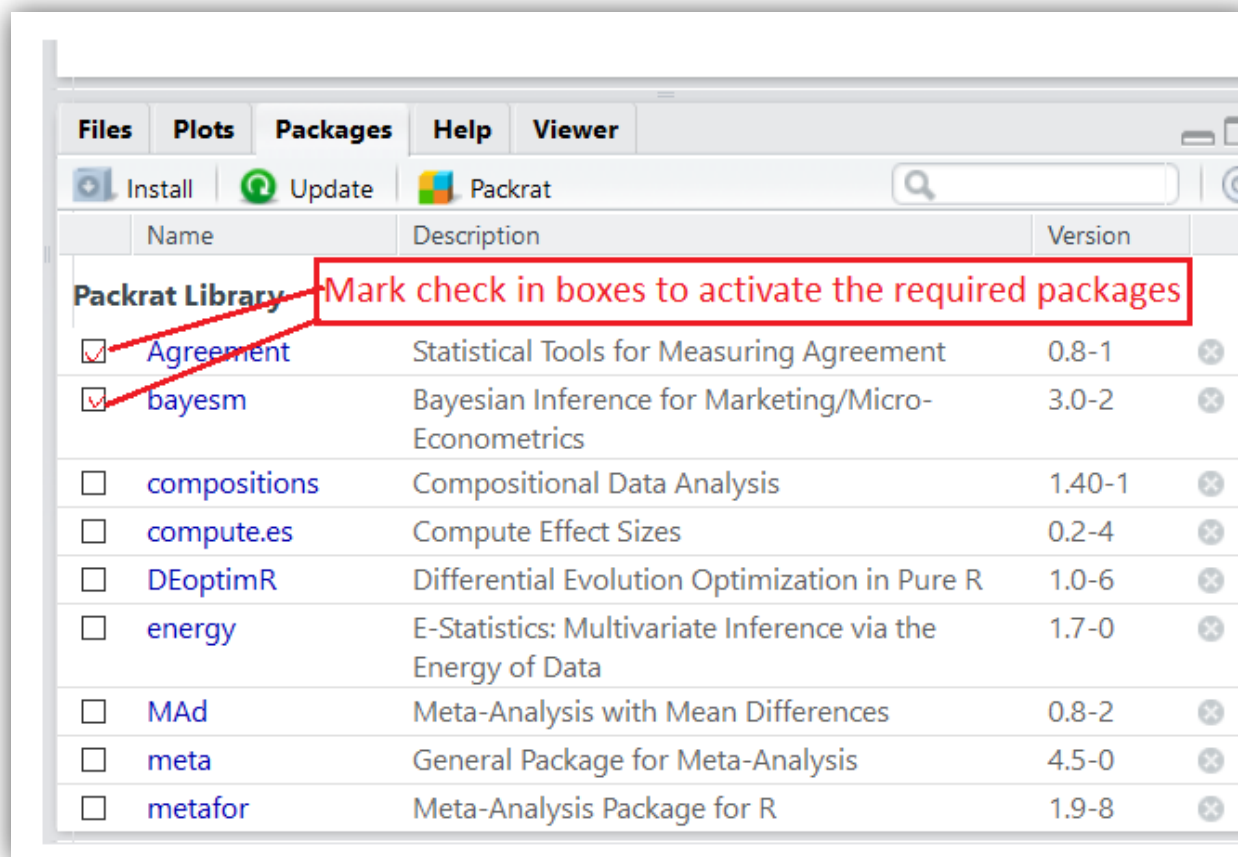
Description of Window 4 : Output window

Packages:

All the packages being installed appear here.

Packages are not active.

Check mark in the boxes to activate them.



Introduction to R Studio

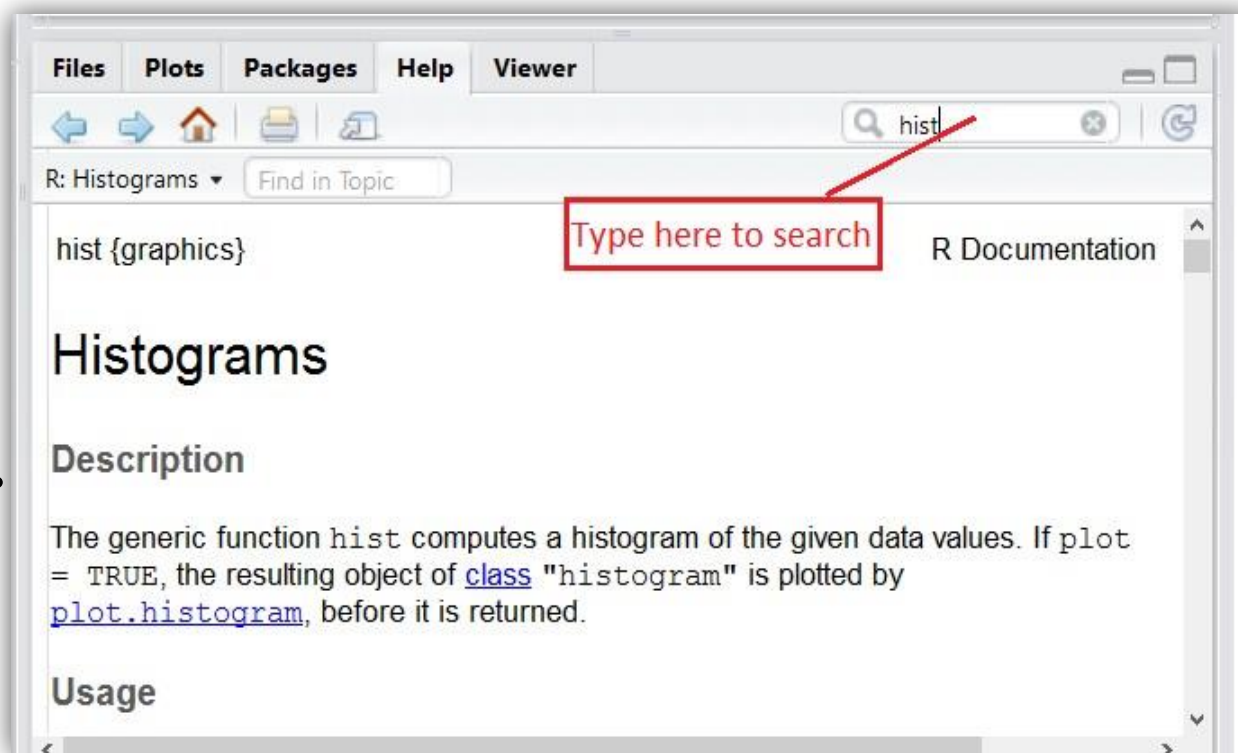
Window 4 : Output window

Help:

Various types of help can be asked.

E.g., to know about histogram,
type **hist**.

Information appears.



Introduction to R Studio

Example:

Histogram of values 1,2,1,1,2,3,1,2,3,1,2,2,3

R studio has following operation and output:

```
Untitled1* x
1 x=c(1,2,1,1,2,3,1,2,3,1,2,2,3)
2 hist(x)
3 |
```

Step 1: Type commands here

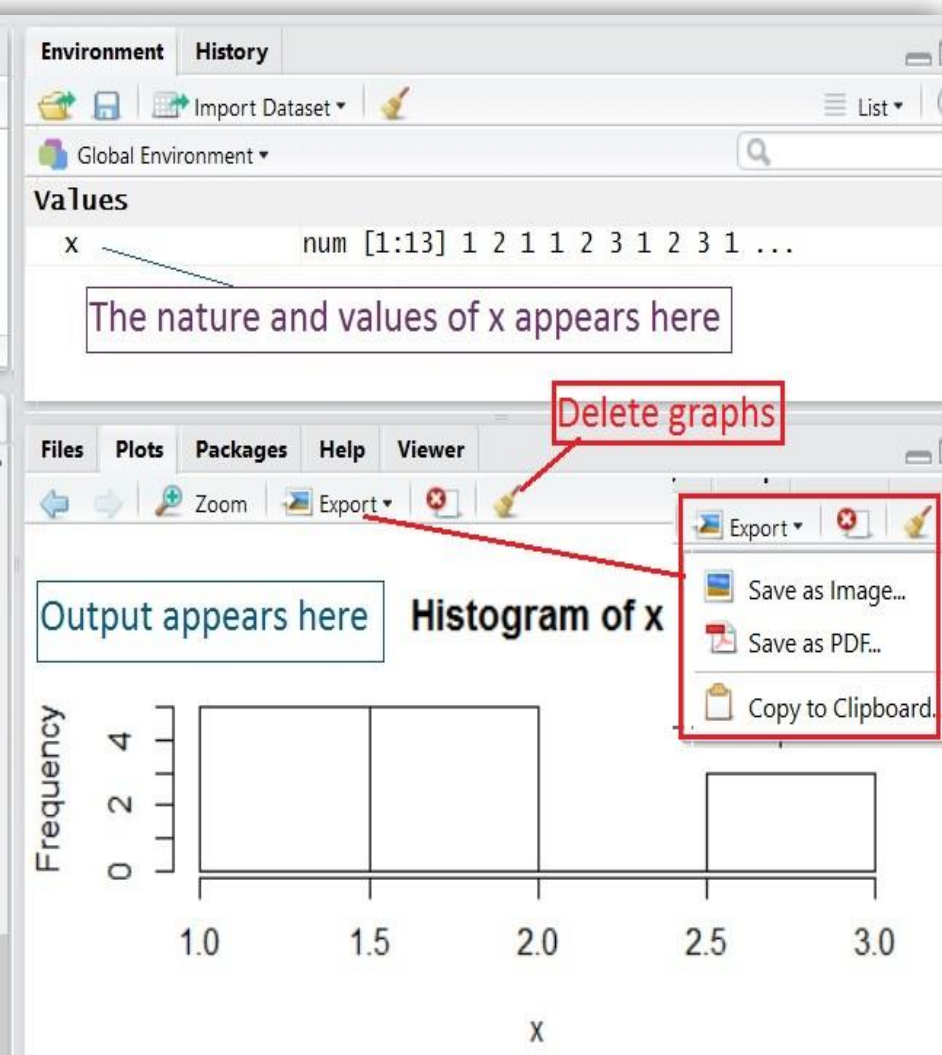
Step 2 :Click on 'Run'

3:1 (Top Level) R Script

Console ~/

```
> hist(x)
```

Step 3: The executed command appear here



Getting Help in R

- * Search for help in Google www.google.com
- * If you need help with a function, then type question mark followed by the name of the function.
- * For example, `?read.table` to get help for function `read.table`.
- * Sometimes, you want to search by the subject on which you want help (e.g. data input). In such a case, type
- * `help.search("data input")`
- * `'help()'` for on-line help, or `'help.start()'` for an HTML browser interface to help.

Getting Help in R

- * Other useful functions are find and apropos.
- * The find function tells us what package something is in.
- * The apropos returns a character vector giving the names of all objects in the search list that match your enquiry.

Basics and R as a Calculator

- * The assignment operators are the left arrow with dash <- and equal sign =
 - > `x = 20` assigns the value 20 to x.
 - > `y = x * 2` assigns the value $2 \times x$ to y.
 - > `z = x + y` assigns the value $x + y$ to z.
- * The character # marks the beginning of a comment. All characters until the end of the line are ignored.
- * Capital and small letters are different.
- * R as a calculator eg. `2+3` command gives the output 5, > `2**3` or `2^3` Command gives the output 8

Basics and R as a Calculator

- * The command `c(1,2,3,4,5)` combines the numbers 1,2,3,4 and 5 to a vector.
- * Multiplication and Division $x * y$, x/y
- * Addition and Subtraction $x + y$, $x - y$
- * Integer Division `%%` (fractional part (remainder) is discarded)
- * Modulo Division $(x \bmod y) \%$ (remainder is the output)

Exercise

* Assign values 2 to variable x, -5 to variable y and find

1. $z1 = x^2 + y^2$

2. $z2 = 2x - 5y$

3. $z3 = y^{**}2$

4. $z4 = y\%/\%x$

5. $z5 = y\%\%x$

Application to a Vector

* Command

output

* $c(2,3,5,7)^2$

[1] 4 9 25 49

* $> c(2,3,5,7)^{c(2,3)}$

[1] 4 27 25 343

* $> c(2,3,5,7)^{c(2,3,4)}$

[1] 4 27 625 49

Warning message : longer object length is not a multiple of shorter object length in:

$c(2,3,5,7)^{c(2,3,4)}$

Application to a Vector

* Command	output
* > c(2,3,5,7) + 10	[1] 12 13 15 17
* > c(2,3,5,7) + c(-2,-3, -5, 8)	[1] 0 0 0 15
* > c(2,3,5,7) + c(8,9,10)	[1] 10 12 15 15 (warning)
* > c(2,3,5,7) %/% 2	[1] 1 1 2 3
* > c(2,3,5,7) %% 2	[1] 0 1 1 1
* > max(1.2, 3.4, -7.8)	[1] 3.4
* > min(1.2, 3.4, -7.8)	[1] -7.8
* > sqrt(c(4,1,9))	[1] 2 1 3

Functions

- * Functions are a bunch of commands grouped together in a sensible unit
- * Functions take input arguments, do calculations (or make some graphics, call other functions) and produce some output and return a result in a variable. The returned variable can be a complex construct, like a list.

Functions

Syntax

```
Name <- function(Argument1, Argument2, ...)  
{  
  expression  
}
```

where expression is a single command or a group of commands

Functions

Syntax

```
Name <- function(Argument1, Argument2, ...)
```

```
{
```

```
  expression
```

```
}
```

where expression is a single command or a group of commands

Examples

- * Functions (Single variable)

```
> abc <- function(x){ x^2 }
```

```
> abc(3)           [1] 9
```

- * Functions (Two variables)

```
> abc <- function(x,y){ x^2+y^2 }
```

```
> abc(2,3)         [1] 13
```

```
> abc <- function(x){ sin(x)^2+cos(x)^2 + x }
```

```
> abc(8)           [1] 9
```

Matrices

- * Matrices are important objects in any calculation. A matrix is a rectangular array with p rows and n columns. An element in the i -th row and j -th column $X[i, j]$, $i = 1, 2, \dots, n$, $j = 1, 2, \dots, p$.
- * We are mostly interested in numerical matrices, whose elements are generally real numbers in R ,

Matrices

- * A 4×2 -matrix X can be created with a following command:

- * `> x <- matrix(nrow=4, ncol=2, data=c(1,2,3,4,5,6,7,8))`

`> x`

	[,1]	[,2]
[1,]	1	5
[2,]	2	6
[3,]	3	7
[4,]	4	8

Matrices

- * The parameter nrow defines the row number of a matrix.
- * The parameter ncol defines the column number of a matrix.
- * The parameter data assigns specified values to the matrix elements. The values from the parameters are written column-wise in matrix.
- * One can access a single element of a matrix with $x[i,j]$:
- * $> x[3,2]$ $[1] 7$

Matrices

- * In case, the data has to be entered row wise, then a 4 × 2-matrix X can be created with `> x <- matrix(nrow=4, ncol=2, data=c(1,2,3,4,5,6,7,8), byrow = TRUE)`

```
> x
```

```
      [,1] [,2]  
[1,]  1    2  
[2,]  3    4  
[3,]  5    6  
[4,]  7    8
```

Matrices

➤ `x1<-matrix(1:9,3,3,byrow=F)`

`> x1`

```
      [,1] [,2] [,3]  
[1,]  1   4   7  
[2,]  2   5   8  
[3,]  3   6   9
```

➤ `> x2 <- diag(3, nrow=2, ncol=2)`

`> x2`

```
      [,1] [,2]  
[1,]  3   0  
[2,]  0   3
```


Matrices

➤ `x3<-matrix(1,3,3)`

`> x3`

```
      [,1] [,2] [,3]  
[1,]  1   1   1  
[2,]  1   1   1  
[3,]  1   1   1
```

➤ `x4<-matrix(1,3,2)`

`> x4`

```
      [,1] [,2]  
[1,]  1   1  
[2,]  1   1  
[3,]  1   1
```

Matrices

➤ `diag(nrow = 3)`

	[,1]	[,2]	[,3]
[1,]	1	0	0
[2,]	0	1	0
[3,]	0	0	1

Matrices

```
x1<-matrix(1:9,3,3,byrow=F)
```

```
> x1
```

```
  [,1] [,2] [,3]  
[1,]  1  4  7  
[2,]  2  5  8  
[3,]  3  6  9
```

Command

Output

```
> dim(x1)
```

```
[1] 3 3
```

```
> x1[2,3]
```

```
[1] 8
```

```
> x1[,2]
```

```
[1] 4 5 6
```

```
> x1[1,]
```

```
[1] 1 4 7
```

Matrices Multiplication

```
➤ x1<-matrix(1:9,3,3,byrow=F)
```

```
> x1
```

```
      [,1] [,2] [,3]  
[1,]    1    4    7  
[2,]    2    5    8  
[3,]    3    6    9
```

```
> x4<-matrix(1,3,2)
```

```
> x4
```

```
      [,1] [,2]  
[1,]    1    1  
[2,]    1    1  
[3,]    1    1
```

`x%*%y` is the correct command to obtain the multiplication of two matrices `x` and `y` if exists.

```
> x1%*%x4
```

```
      [,1] [,2]  
[1,]    3    3  
[2,]    3    3
```

Eigen Values and Vectors of Matrices

```
* > x1<-matrix(1:9,3,3,byrow=F)
> x1
      [,1][,2][,3]
[1,]  1  4  7
[2,]  2  5  8
[3,]  3  6  9
> eigen(x1)
eigen() decomposition $values
[1] 1.611684e+01 -1.116844e+00 -4.054214e-16
$vector
      [,1] [,2] [,3]
[1,] -0.4645473 -0.8829060 0.4082483
[2,] -0.5707955 -0.2395204 -0.8164966
[3,] -0.6770438 0.4038651 0.4082483
```

system of linear equations

- * $x + y + z = 5,$
- * $2x + 3y + z = 10,$
- * $3x - 2y + 2z = 3$

```
> a <- rbind(c(1, 1, 1), c(2, 3, 1), c(3, -2, 2))  
> b <- c(5, 10, 3)  
> solve(a, b)
```

```
[1] 1 2 2
```

Conditional Executions and Loops

- * Conditional execution -- ifelse(test, yes, no)

- * Syntax

if (condition) {executed commands if condition is
TRUE} else { executed commands if condition is FALSE }

```
> x <- 1:10
```

```
>x
```

```
[1] 1 2 3 4 5 6 7 8 9 10
```

```
>ifelse( x<6, x^2, x+1 )
```

```
[1] 1 4 9 16 25 7 8 9 10 11
```

Control structures in R :Loops

- * Repetitive commands are executed by loops for loop, while loop, repeat loop
- * The for loop: If the number of repetitions is known in advance, a for() loop can be used.
- * `for (name in vector) {commands to be executed}`
- * `> for (i in 1:5) { print(i^2) }`

`[1] 1`

`[1] 4`

`[1] 9`

`[1] 16`

`[1] 25`

Control structures in R :Loops

```
> for ( i in c(2,4,6,7) ) { print( i^2 ) }
```

```
[1] 4
```

```
[1] 16
```

```
[1] 36
```

```
[1] 49
```

Control structures in R :Loops

```
> for ( i in c(2,4,6,7) ) { print( i^2 ) }
```

```
[1] 4
```

```
[1] 16
```

```
[1] 36
```

```
[1] 49
```

The while() loop

- * If the number of loops is not known in before, e.g. when an iterative algorithm to maximize a likelihood function is used, one can use a while() loop.

- * Syntax

while(condition){ commands to be executed as long as condition is TRUE }

The while() loop

```
> i <- 1  
> while (i<5) {  
+ print(i^2)  
+ i <- i+2  
+}  
[1] 1  
[1] 9
```

Note: The programmer itself has to be careful that the counting variable *i* within the loop is incremented. Otherwise an infinite loop occurs.

Sequences

- * The default increment is +1 or -1

```
> seq(from=-4, to=4)      [1] -4 -3 -2 -1 0 1 2 3 4
```

- * Sequence with constant increment:

Command

Output

```
seq(from=10, to=20, by=2)  [1] 10 12 14 16 18 20
```

```
seq(from=3, to=-1, by=-0.5) [1] 3.0 2.5 2.0 1.5 1.0 0.5 0.0 -0.5 -1.0
```

```
seq(to=10, length=10)     [1] 1 2 3 4 5 6 7 8 9 10
```

```
seq(from=10, length=10, by=0.1)
```

```
[1] 10.0 10.1 10.2 10.3 10.4 10.5 10.6 10.7 10.8 10.9
```

```
> x<-2
```

```
> seq(1, x, x/10)         [1] 1.0 1.2 1.4 1.6 1.8 2.0
```

Sequences

Command

Output

The regular sequences can be generated in R.

* Code

```
>seq(10)
```

```
>seq(1:10)
```

```
>abs(seq(-2,2))
```

```
>sqrt(abs(seq(-6,6, by = 3)))
```

* Output

```
[1] 1 2 3 4 5 6 7 8 9 10
```

```
* [1] 1 2 3 4 5 6 7 8 9 10
```

```
* [1] 2 1 0 1 2
```

```
* [1] 2.449490 1.732051  
0.000000 1.732051  
2.449490
```

The regular sequences can be generated in R.

* Code

```
>seq(10)
```

```
>seq(1:10)
```

```
>abs(seq(-2,2))
```

```
>sqrt(abs(seq(-6,6, by = 3)))
```

* Output

```
[1] 1 2 3 4 5 6 7 8 9 10
```

```
* [1] 1 2 3 4 5 6 7 8 9 10
```

```
* [1] 2 1 0 1 2
```

```
* [1] 2.449490 1.732051  
0.000000 1.732051  
2.449490
```


The regular sequences can be generated in R.

* Code

```
>seq(to = 100, length = 8)
```

```
>seq(to = 22, length = 5, by  
= 6)
```

```
>seq(from = -10, length = 5,  
by = -0.3)
```

* Output

```
* [1] 93 94 95 96 97 98 99  
100
```

```
* [1] -2 4 10 16 22
```

```
* [1] -10.0 -10.3 -10.6 -10.9 -  
11.2
```

Sorting

- * sort function sorts the values of a vector in ascending order (by default) or descending order.
- * Example
- *

```
> y <- c(8,5,7,6)
```

```
> Y
```

```
[1] 8 5 7 6
```
- *

```
> sort(y)
```

```
[1] 5 6 7 8
```
- *

```
> sort(y, decreasing = TRUE)
```

```
[1] 8 7 6 5
```

Repeats

- * Command rep is used to replicates the values
- * `rep(x, times=n)` # Repeat x as a whole n times a vector.

- * `Rep(x, each=n)` # Repeat each cell n times

```
> rep(3.5, times=4)           [1] 3.5 3.5 3.5 3.5
```

```
> rep(1:4, 2)                 [1] 1 2 3 4 1 2 3 4
```

```
> rep(1:4, each = 2, times = 3)
```

```
[1] 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4
```

Repeats

- * Every object is repeated a different number of times:

```
>rep(1:4, 2:5)
```

```
[1] 1 1 2 2 2 3 3 3 3 4 4 4 4
```

```
>rep(c("a", "b", "c"), 2)
```

```
[1] "a" "b" "c" "a" "b" "c"
```

```
>rep(c("apple", "banana", "cake"), 2)
```

```
[1] "apple" "banana" "cake" "apple" "banana"
"cake"
```

Generating current time and date

- * `Sys.time()` command provides the current time and date from the computer system.

```
Sys.time()
```

```
[1] "2019-01-02 14:24:29 IST"
```

- * `Sys.Date()` command provides the current date from the computer system.

```
> Sys.Date()
```

```
[1] "2019-01-02"
```

Generating current time and date

- * Generating sequences of dates

- * Sequence of years, months or days

```
> seq(as.Date("2017-01-01"), by = "years", length = 5)
```

```
[1] "2017-01-01" "2018-01-01" "2019-01-01" "2020-01-01" "2021-01-01"
```

```
> seq(as.Date("2017-01-01"), by = "months", length = 4)
```

```
[1] "2017-01-01" "2017-02-01" "2017-03-01" "2017-04-01"
```

```
> seq(as.Date("2017-01-01"), by = "days", length = 5)
```

```
[1] "2017-01-01" "2017-01-02" "2017-01-03" "2017-01-04" "2017-01-05"
```

Generating sequences of letters

- * A vector of positive integers (letters and LETTERS return the 26 lowercase and uppercase letters, respectively).
- * `> letters`
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n"
[15] "o" "p" "q" "r" "s" "t" "u" "v" "w" "x" "y" "z"
- * `> letters[21:23]`
[1] "u" "v" "w"
- * `> LETTERS[1:3]`
[1] "A" "B" "C"

Vector indexing A logical vector

* `> x <- 1:10`

* `> x`

* `[1] 1 2 3 4 5 6 7 8 9 10`

* `> x[(x > 5)]`

* `[1] 6 7 8 9 10`

* `> x[(x%%2==0)]`

`##% indicates x mod y`

* `[1] 2 4 6 8 10`

`#values for which x mod 2 is 0`

* `> x[(x%%2==1)]`

* `[1] 1 3 5 7 9`

`# values for which x mod 2 is 1`

logical vector

```
* > x <- 1:10
* > x
* [1] 1 2 3 4 5 6 7 8 9 10
* > x[5] <- NA
* > x
* [1] 1 2 3 4 NA 6 7 8 9 10
  > y <- x[ !is.na(x) ]    #! Means negation
* > y
* [1] 1 2 3 4 6 7 8 9 10 # 5 is missing
  > mean(x)
* [1] NA
* > mean(y)
* [1] 5.555556
```

Partition values

- * **Quartile:** Divides the data into 4 equal parts.
- * **Decile:** Divides the data into 10 equal parts.
- * **Percentile:** Divides the data into 100 equal parts.

Partition values

- * **quantile** function computes quantiles corresponding to the given probabilities.
- * The smallest observation corresponds to a probability of 0 and the largest to a probability of 1.
- * **quantile(x, ...)**
- * **quantile(x, probs = seq(0, 1, 0.25), ...)**
- * **Arguments**
- * **x** numeric vector whose sample quantiles are wanted,
- * **probs** numeric vector of probabilities with values in $[0, 1]$.

Partition values

Example: Marks of 15 students are

```
marks <- c(68, 82, 63, 86, 34, 96, 41, 89, 29, 51, 75, 77, 56, 59, 42)
```

Command

```
quantile(marks)
```

Output

0%	25%	50%	75%	100%
29.0	46.5	63.0	79.5	96.0

Partition values

Example: Marks of 15 students are

```
marks <- c(68, 82, 63, 86, 34, 96, 41, 89, 29, 51, 75, 77, 56, 59, 42)
```

Defining probabilities

Command

```
quantile(marks, probs=c(0,0.20,0.4,0.6,0.8,1))
```

Output

0%	20%	40%	60%	80%	100%
29.0	41.8	57.8	70.8	82.8	96.0

Central tendency of data

Example

```
marks<- c(68, 82, 63, 86, 34, 96, 41, 89, 29, 51, 75, 77, 56, 59, 42)
```

	Command	Output
Mean	<code>mean(marks)</code>	63.2
Geometric mean	<code>prod(marks)^(1/length(marks))</code>	59.61099
Harmonic mean	<code>1/mean(1/marks)</code>	55.78628
Median	<code>median(marks)</code>	63
Variance	<code>var(Marks)</code>	439.3143
standard deviation	<code>sqrt(var(marks))</code>	20.95983

References

- * Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.
- * Hands-On Programming with R. Garrett Grolmund Publisher: O'Reilly Media