IT Workshop

R Programming

Pradipta Sarkar

`R (*.r)` is the actual programming language, **`R Studio`** is a convenient interface in which to use it, and **`R Markdown`** is a specific type of file format designed to produce documents that include both code *and* text.

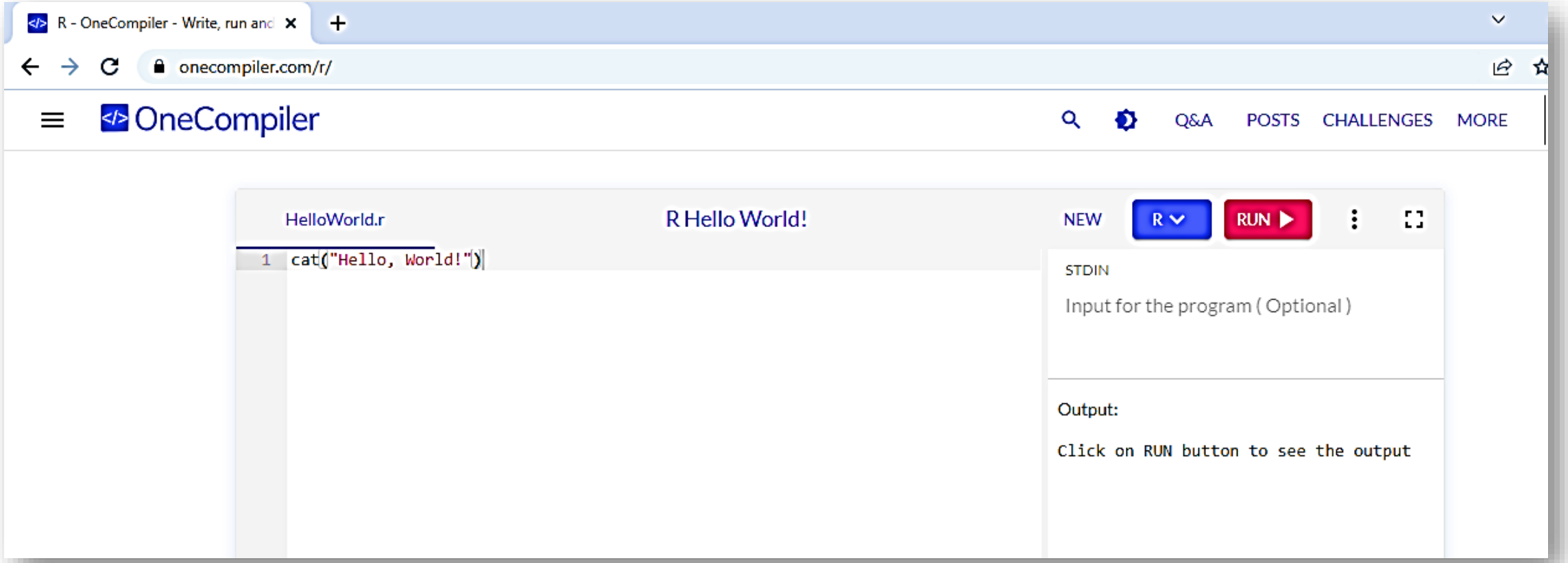
`R Markdown ( *.rmd)` is a tool for converting plain text into formatted text. R Markdown utilizes the markdown syntax in order to combine formatted text with code in a single document.

An **`R script`** is simply a text file containing (almost) the same commands that we would enter on the command line of R

Index

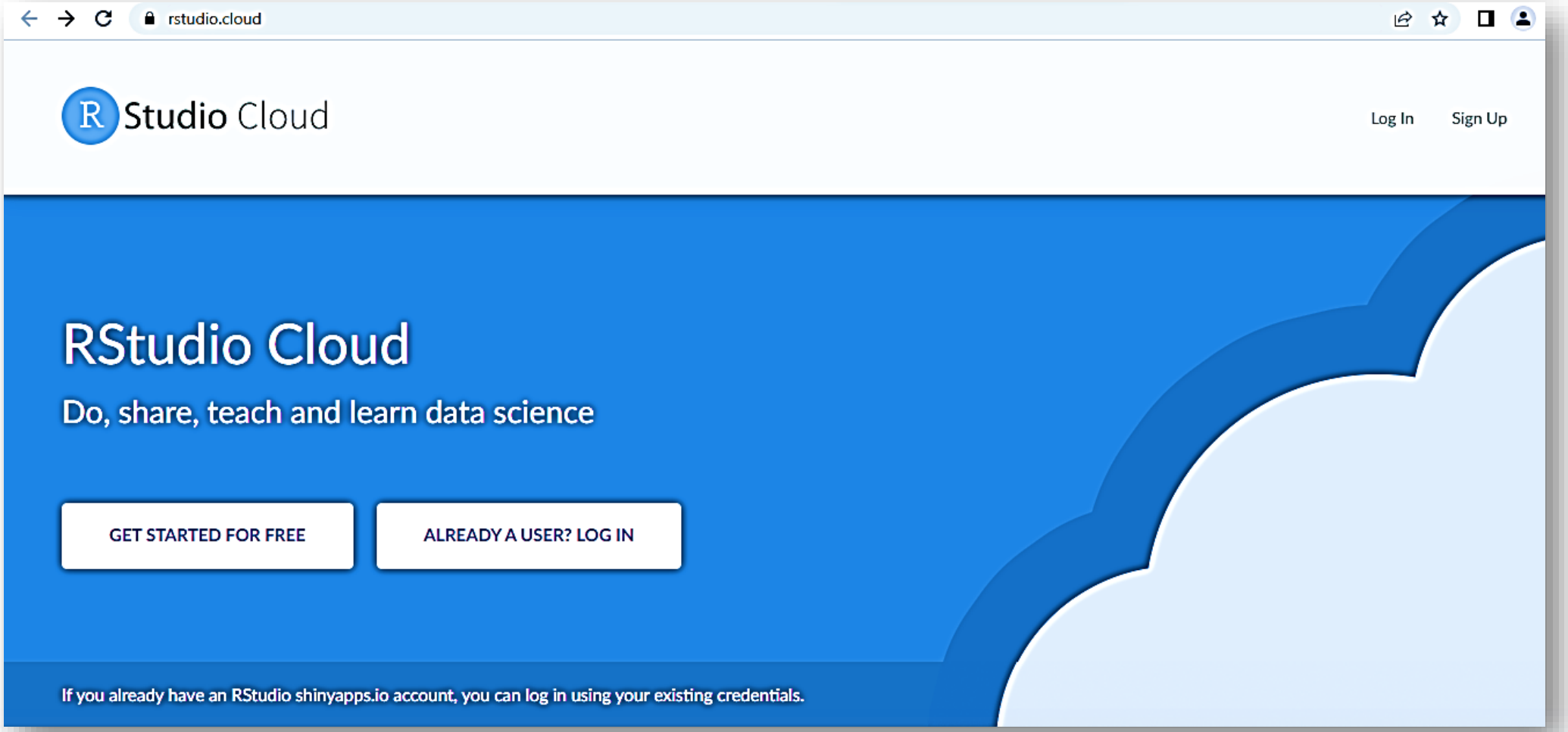
Online IDE of R Language

https://onecompiler.com/r/



Online IDE of R Language

https://rstudio.cloud/



R Studio Desktop

https://www.rstudio.com/products/rstudio/download/

RStudio Desktop 2022.07.0+548 - Release Notes

1. Install R. RStudio requires R 3.3.0+.

2. Download RStudio Desktop. Recommended for your system:

DOWNLOAD RSTUDIO FOR WINDOWS
2022.07.0+548 | 190.14MB

Requires Windows 10/11 (64-bit)

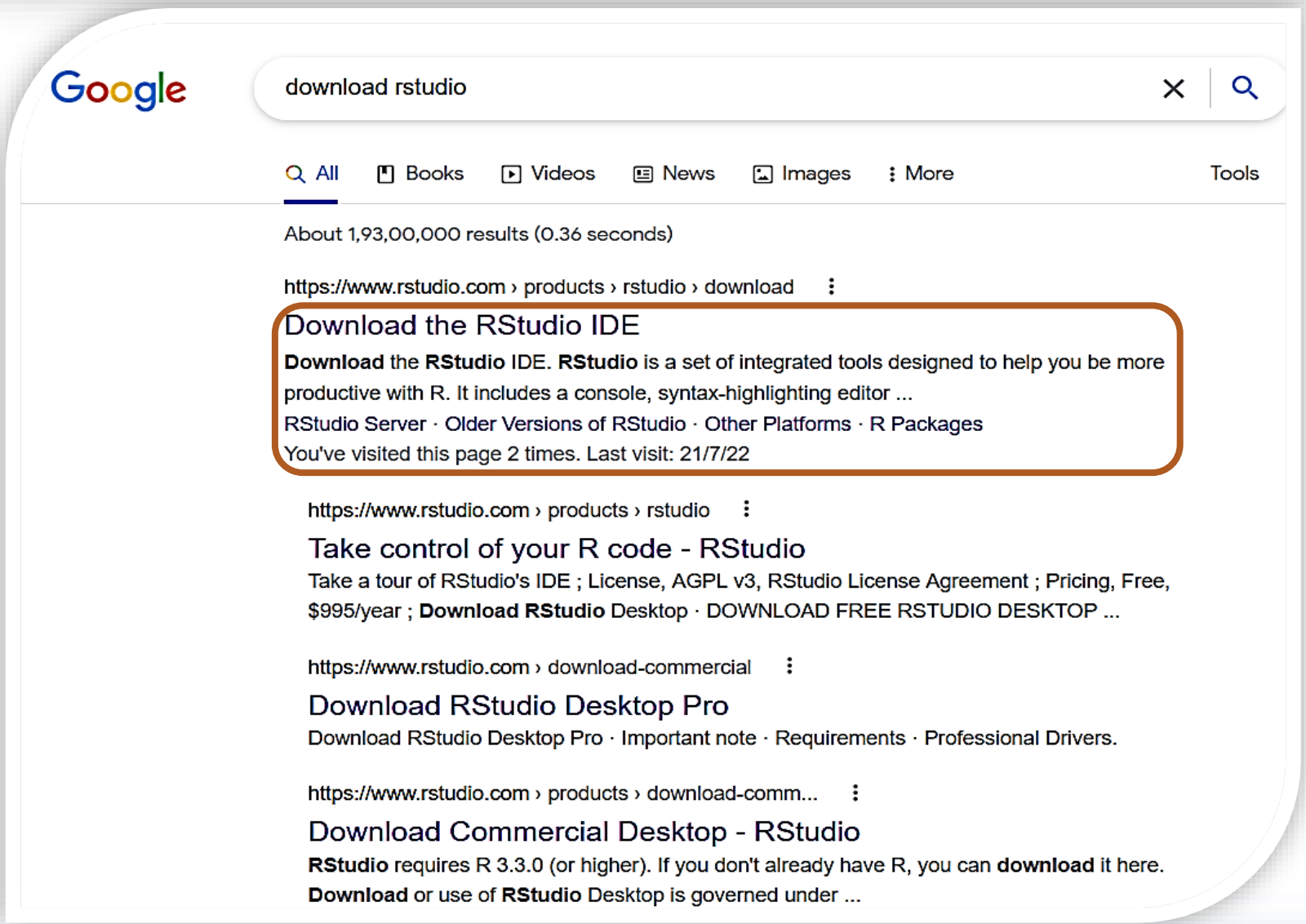
All Installers

Linux users may need to import RStudio's public code-signing key prior to installation, depending on the operating system's security policy.

RStudio requires a 64-bit operating system. If you are on a 32 bit system, you can use an older version of RStudio.

OS	Download	Size	SHA-256
Windows 10/11	RStudio-2022.07.0-548.exe	190.14 MB	7bfd6825
macOS 10.15+	RStudio-2022.07.0-548.dmg	222.07 MB	52cf91ff

R Studio IDE



RStudio is an **integrated development environment (IDE)** for R, a programming language for statistical computing and graphics.

It is available in two formats: **RStudio Desktop** is a regular desktop application while

RStudio Server runs on a remote server and allows accessing RStudio using a web browser.

R Installation

https://cran.rstudio.com

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 (Debian, Fedora/Redhat, Ubuntu)
- Download R for macOS
- 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 (2022-06-23, Funny-Looking Kid) R-4.2.1.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

Questions About R

- 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.

What are R and CRAN?

R is 'GNU S', a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques: linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, etc. Please consult the R project homepage for further information.

R Installation

https://cran.rstudio.com/bin/windows/

R for Windows

Subdirectories:

base	Binaries for base distribution. This is what you want to **install R for the first time**.
contrib	Binaries of contributed CRAN packages (for R >= 3.4.x).
old contrib	Binaries of contributed CRAN packages for outdated versions of R (for R < 3.4.x).
Rtools	Tools to build R and R packages. This is what you want to build your own packages on Windows, or to b R itself.

Please do not submit binaries to CRAN. Package developers might want to contact Uwe Ligges directly in case of questions / suggestions related to W ws binaries.

You may also want to read the R FAQ and R for Windows FAQ.

Note: CRAN does some checks on these binaries for viruses, but cannot give guarantees. Use the normal precautions with downloaded executables.

https://cran.rstudio.com/bin/windows/base/

R-4.2.1 for Windows

Download R-4.2.1 for Windows (79 megabytes, 64 bit)

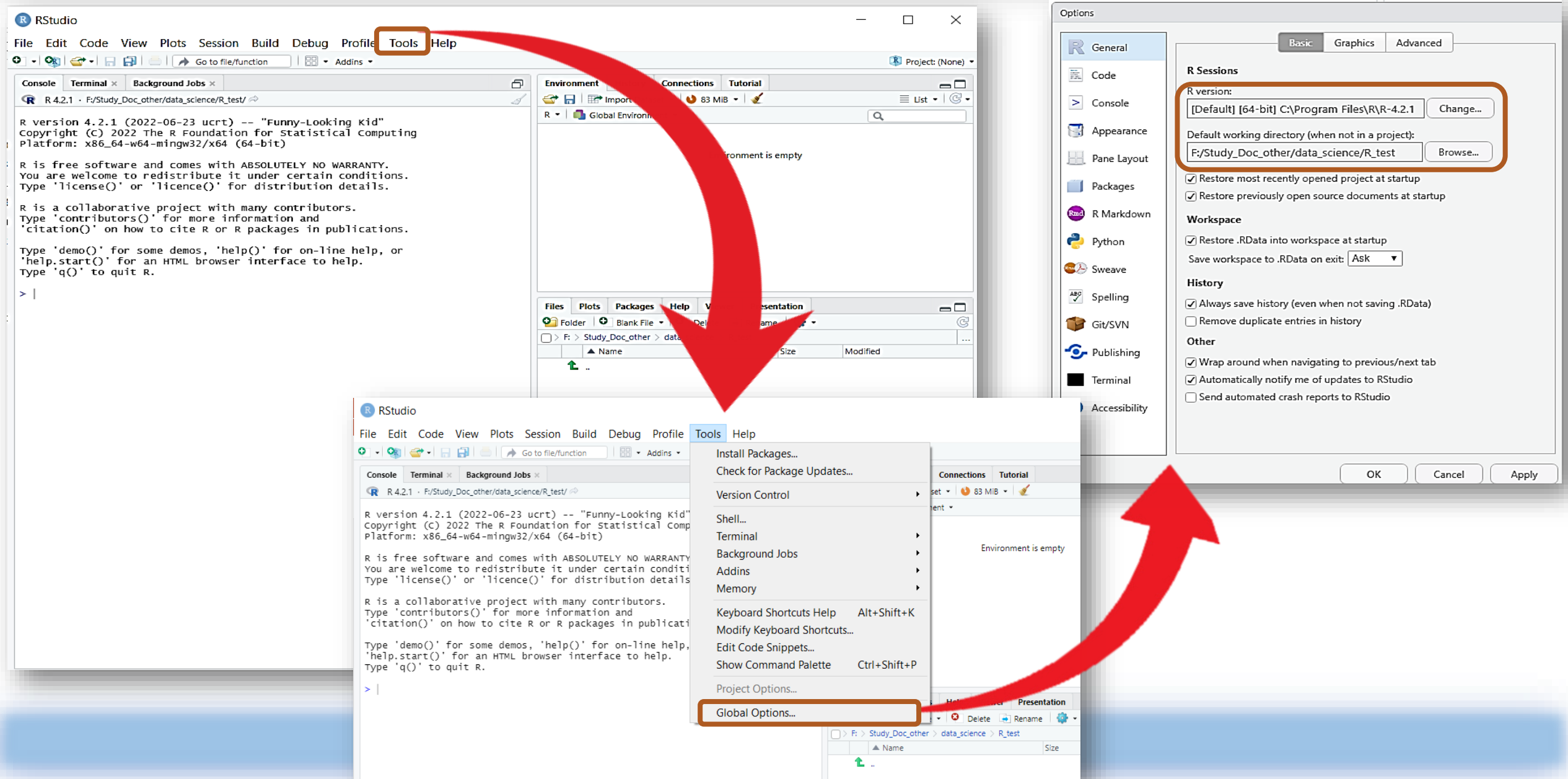
README on the Windows binary distribution

New features in this version

This build requires UCRT, which is part of Windows since Windows 10 and Windows Server 2016. On older systems, UCRT has to be installed manually from here.

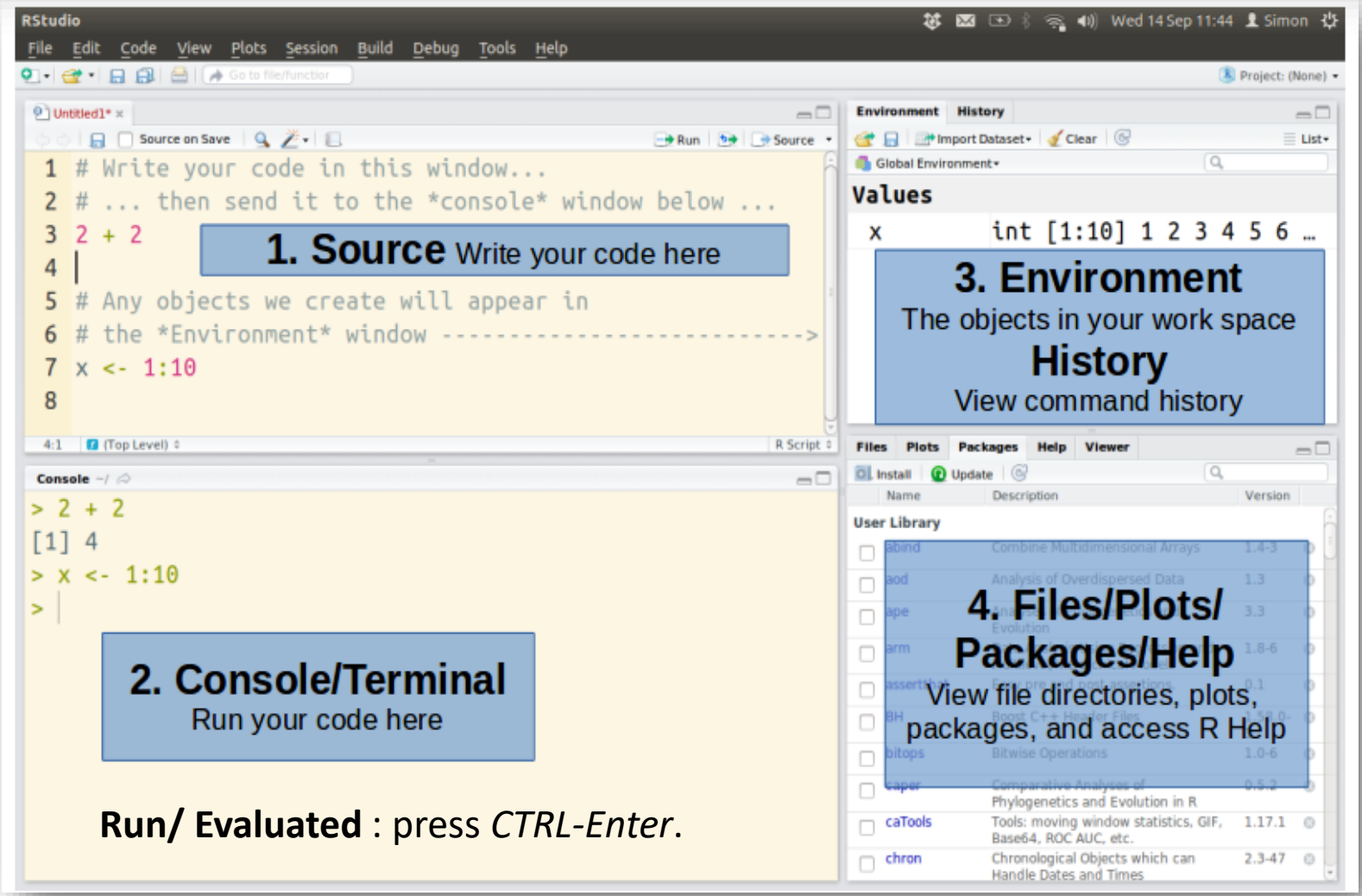
If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the md5sum of the .exe to the fingerprint on the master server.

R Studio Installation



R Studio

The **Source** is the notepad or editor where we write our code, or R scripts.



The **Environment** tab shows us all the objects (e.g., X) that are currently in our R session.

The **History** tab shows all the code that has been entered into the Console in this session.

The **Console** is actually R, where commands are run (or *evaluated*). The arrow symbol (“>”) tells us that R is ready for new code.

The **Files** tab is the equivalent of Explorer or Finder. It shows the file directory structure on our computer’s hard drive. We can use it to navigate through to specific directories to read and write files.

If we type “2 + 2” directly into the Console and press Enter, R will evaluate that code immediately and return the output.

The **Plots** tab displays all the plots created in that session.

R Language (variable)

R - Code

```
var1<- "Hello Data Science" # Creating Variables
var2<- 40                   # Creating Variables
print(var1)                 # Print Variables
var2                        # Output Variables
```

Output

```
> var1<- "Hello Data Science" # Creating Variables
> var2<- 40                   # Creating Variables
> print(var1)                 # Print Variables
[1] "Hello Data Science"
> var2                        # Output Variables
[1] 40
```

R - Code

```
> print(1:50)
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
[37] 37 38 39 40 41 42 43 44 45 46 47 48 49 50
```

We'll notice that a [1] appears next to our result.
R is just letting us know that this line begins with the **first value** in our result. Some commands return more than one value, and their results may fill up multiple lines.

For example, the command 1:50 returns 50 values; It creates a sequence of integers from 1 to 50. Notice that new bracketed numbers appear at the start of the second lines of output. These numbers just mean that the second line begins with the **37th value** in the result.

R Language (Data Type)

class() function to check the data type of a variable

Basic Data Types

Basic data types in R can be divided into the following types:

- **Numeric** - (10.5, 55, 787)
- **Integer** - (1L, 55L, 100L, where the letter "L" declares this as an integer)
- **Complex** - (9 + 3i, where "i" is the imaginary part)
- **Character (a.k.a. string)** - ("k", "R is exciting", "FALSE", "11.5")
- **Logical (a.k.a. boolean)** - (TRUE or FALSE)

R - Code

```
> # numeric
> x <- 10.5
> class(x)
[1] "numeric"
>
> # integer
> x <- 1000L
> class(x)
[1] "integer"
>
> # complex
> x <- 9i + 3
> class(x)
[1] "complex"
>
> # character/string
> x <- "R is exciting"
> class(x)
[1] "character"
>
> # logical/boolean
> x <- TRUE
> class(x)
[1] "logical"
```

R Language (Data Structure)

❖ Vector :

1. **Atomic Vectors** : A sequence of elements which share the same data type is known as vector.
2. **List** : The elements are of different data types.

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

"Red", "Green", c(21,32,11), TRUE, 51.23, 119.1

What is R lists? and How to create list?

```
A list is a data structure which has components of mixed data types.

vec <- c(3,4,5,6)
char_vec<-c("shubham","nishka","gunjan","sumit")
logic_vec<-c(TRUE,FALSE,FALSE,TRUE)
out_list<-list(vec,char_vec,logic_vec)
class(out_list)
```

❖ Array : Data objects which allow us to store data in more than two dimensions. A vector is a uni-dimensional array, which is specified by a single dimension, length. "An array is a collection of a similar data type with contiguous memory allocation."

R Language (Data Structure)

❖ **Matrix** : A two-dimensional rectangular data set is known as a matrix.

```
     [,1] [,2] [,3]
[1,]    3    4    5
[2,]    6    7    8
[3,]    9   10   11
[4,]   12   13   14
```

❖ **Data Frame** : it is a list of equal length vectors.

	emp_id	emp_name	salary	start_date
1	1	Rick	623.30	2012-01-01
2	2	Dan	515.20	2013-09-23
3	3	Michelle	611.00	2014-11-15
4	4	Ryan	729.00	2014-05-11
5	5	Gary	843.25	2015-03-27

❖ **Factor** : Factors are the data objects which are used to categorize the data and store it as levels. They can store both strings and integers. They are useful
in the columns which have a limited number of unique values. Like "Male, "Female" and True, False etc.

East, West, East, North, North, East, West, West -------factor-----→ *East, North, West*

A matrix can contain one type of data, but a data frame can contain different data types such as numeric, character, factor, etc.

R Language (Data Structure)

Vector vs Array

Arrays are still a vector in R but they have added extra options to them. Essentially call them “vector structure”. With **a vector** a list of objects in one dimension. With **an array** we can have any number of dimensions to our data. In the Future 2-dimensional array called a matrix.

Array vs Matrix

An array is a collection of similar data types. So standard array looks like this. [**1,2,3,4,5,6,7,8,9]**

And **A matrix** could look like a collection of array, it means that an array which holds other arrays. From a programming point of view, a matrix is an array of 2 dimensions. Matrix does look something like this. **[[1, 2, 3], [4, 5, 6], [7, 8, 9]]**

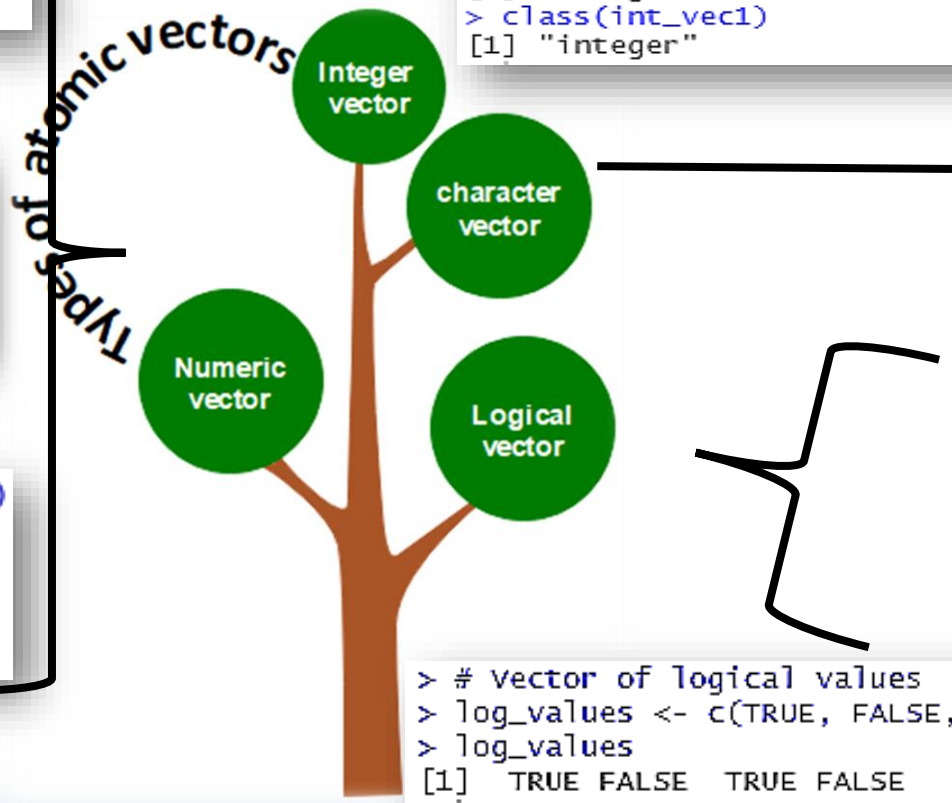
Data Structure (Vector)

To combine the list of items to a vector, use the c( ) function and separate the items by a comma.

```
> seq_vec1<-seq(1,4,by=0.5)
> seq_vec1
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0
> class(seq_vec1)
[1] "numeric"
```

```
> num_vec<-c(10.1, 10.2, 33.2)
> num_vec
[1] 10.1 10.2 33.2
> class(num_vec)
[1] "numeric"
```

```
> seq_vec<-seq(1,4,length.out=5)
> seq_vec
[1] 1.00 1.75 2.50 3.25 4.00
> class(seq_vec)
[1] "numeric"
```




```
> int_vec<-c(1,2,3,4,5)
> int_vec<-as.integer(int_vec)
> int_vec1<-c(1L,2L,3L,4L,5L)
>
> int_vec
[1] 1 2 3 4 5
> int_vec1
[1] 1 2 3 4 5
>
> class(int_vec)
[1] "integer"
> class(int_vec1)
[1] "integer"
```

```
> a<-4:-5
> a
 [1]  4  3  2  1  0 -1 -2 -3 -4 -5
> class(a)
[1] "integer"
```

```
> # Vector of strings
> fruits <- c("banana", "apple", "orange")
> fruits
[1] "banana" "apple"  "orange"
```

```
> # Vector of logical values
> log_values <- c(TRUE, FALSE, TRUE, FALSE)
> log_values
[1]  TRUE FALSE  TRUE FALSE
```

```
> d<-as.integer(5)
> e<-as.integer(6)
> f<-as.integer(7)
> g<-d>e
> h<-e<f
> g
[1] FALSE
> h
[1] TRUE
> log_vec<-c(d<e, d<f, e<d,e<f,f<d,f<e)
> log_vec
[1]  TRUE  TRUE FALSE  TRUE FALSE FALSE
> class(g)
[1] "logical"
> class(h)
[1] "logical"
> class(log_vec)
[1] "logical"
```

- **How can we find out the maximum & minimum value from a given vector**

```
Maximum - max();Minimum – min()
```

- **Using R program how to create a vector which contains 10 random integer values between -50 and +50.**

```
v <- sample(-50:50, 10, replace=TRUE)
print("Content of the vector:")
print("10 random integer values between -50 and +50:")
print(v)
```

- **Write a R program to add two vectors of integers type and length 3.**

```
x = c(10, 20, 30)
y = c(20, 10, 40)
print("Original Vectors:")
print(x)
print(y)
print("After adding two Vectors:")
z = x + y
print(z)
```

- **Write a R program to test whether a given vector contains a specified element**.

```
x = c(10, 20, 30, 25, 9, 26)
print("Original Vectors:")
print(x)
print("Test whether above vector contains 25:")
print(is.element(25, x))
print("Test whether above vector contains 56:")
print(is.element(56, x))
```

- **Is there any function in R program to access the last value in a given vector. If yes explain with example**

```
x = c(10, 20, 30, 25, 9, 26)
print("Original Vectors:")
print(x)
tail(x,1)   # Last element in the list
```

- **Write a R program to reverse the order of given vector.**

```
x = c(10, 20, 30, 25, 9, 26)
print("Original Vectors:")
print(x)
rev(x)
```

- **How to fetch 2nd to 5th element from list of 10 elements using R program**.

```
x = c(10, 20, 30, 25, 9, 26)
print("Original Vectors:")
print(x)
x[2:5]
```

- **Is there any method to concatenate two list in R language? explain with example**

```
x <- c(10, 20, 30, 25, 9, 26)
y <- c(1,2,3,4,5,6)
print("Original Vectors:")
print(x)
c(x,y)                    # concatenate two list
append(x,y)               # concatenate two list
```

- **Write a R program which count the specific value and in between a range in a given vector**

```
A = c(20, 30, 30, 10, 30, 25, 30, 27)
print("Original Vectors:")
print(A)
print("Count specific value(30) in above vector:")
print(sum(A==30))

A= c(0, 10, 20, 30, 40, 50, 60, 70, 80, 90)
print("Original vector:")
print(A)
count =  sum(A> 20 & A < 70)
print("Number of vector values between 20 and 70:")
print(count)
```

- **How to find common elements from multiple vectors in the R program**

```
A <- c(15, 10, 40, 20, 14, 25, 29, 15);
B <- c(35, 20, 30, 40, 18, 35, 25, 14);
C <- c(25, 30, 30, 20, 13, 25, 14, 17);
print("Original Vectors:")
print("A: ")
print(A)
print("B: ")
print(B)
print("C: ")
print(C)
print("Common elements from above vectors:")
inter= intersect(intersect(A,B),C)
print(inter)
```

Data Structure (Array)

```
thisarray <- c(1:24)
```

```
# An array with more than one dimension
multiarray <- array(thisarray, dim=c(3,4,2))
```

```
# Accessing Elements from an array
multiarray[1,3,1]                # answer 7
```

```
 [1]  1  2  3  4  5  6  7  8  9 10 11 ......
, , 1

     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12

, , 2

     [,1] [,2] [,3] [,4]
[1,]   13   16   19   22
[2,]   14   17   20   23
[3,]   15   18   21   24
```

Data Structure (Matrix)

```
#Arranging elements sequentially by row.
P <- matrix(c(5:16), nrow = 4, byrow = TRUE)
print(P)
```

```
     [,1] [,2] [,3]
[1,]    5    6    7
[2,]    8    9   10
[3,]   11   12   13
[4,]   14   15   16
```

```
# Arranging elements sequentially by column.
Q <- matrix(c(3:14), nrow = 4, byrow = FALSE)
print(Q)
```

```
     [,1] [,2] [,3]
[1,]    3    7   11
[2,]    4    8   12
[3,]    5    9   13
[4,]    6   10   14
```

```
# Defining the column and row names.
row_names = c("row1", "row2", "row3", "row4")
ccol_names = c("col1", "col2", "col3")

R <- matrix(c(3:14), nrow = 4, byrow = TRUE, dimnames = list(row_names, col_names))
print(R)
```

```
     col1 col2 col3
row1    3    4    5
row2    6    7    8
row3    9   10   11
row4   12   13   14
```

Write a R program to convert a matrix to a vector

```
my_matrix <- matrix(1:20, nrow = 5)
my_matrix

#convert matrix to vector (sorted by columns) using c()
new_vector <- c(my_matrix)

#convert matrix to vector (sorted by rows) using c()
new_vector <- c(t(my_matrix))

#convert matrix to vector (sorted by columns) using as.vector()
new_vector <- as.vector(my_matrix)

#convert matrix to vector (sorted by rows) using as.vector()
new_vector <- as.vector(t(my_matrix))
```

Data Structure (Data Frame)

```
# Create a data frame

Data_Frame <- data.frame (
Training = c("Strength", "Stamina", "Other"),
Pulse = c(100, 150, 120),
Duration = c(60, 30, 45)
)
```

```
# Print the data frame

print(Data_Frame)

Data_Frame[2,3]

Data_Frame[['Pulse']]

Data_Frame$Duration
```

```
  Training Pulse Duration
1 Strength   100       60
2  Stamina   150       30
3    Other   120       45
```

```
[1] 30
```

```
[1] 100 150 120
```

```
[1] 60 30 45
```

What is the function which is used for merging of data frames vertically in R?

```
a1 <-c(6.2,6.1,5.9,5.7,5.8,5.6)
print(a1)
a2 <-c(0.1,-0.1,0.1,0.1,-0.3,-0.3)
print(a2)
a3 <- cbind(a1,a2)
print(a3)
```

What is the function which is used for merging of data frames horizontally in R?

```
a1 <-c(6.2,6.1,5.9,5.7,5.8,5.6)
print(a1)
a2 <-c(0.1,-0.1,0.1,0.1,-0.3,-0.3)
print(a2)
a3 <- rbind(a1,a2)
print(a3)
```

Data Structure (Factor)

```
music_genre <- factor(c("Jazz", "Rock", "Classic", "Classic", "Pop", "Jazz", "Rock"))
```

```
print(music_genre)
```

Output:

```
[1] Jazz    Rock    Classic Classic Pop     Ja
Levels: Classic Jazz Pop Rock
```

```
levels(music_genre)
```

```
[1] "Classic" "Jazz"    "Pop"     "Rock"
```

Operator

Arithmetic

```r
vec1 <- c(1, 4)
vec2 <- c(2, 3)

# Performing operations on Operands
cat ("Addition of vectors :", vec1 + vec2, "\n")
cat ("Subtraction of vectors :", vec1 - vec2, "\n")
cat ("Multiplication of vectors :", vec1 * vec2, "\n")
cat ("Division of vectors :", vec1 / vec2, "\n")
cat ("Modulo of vectors :", vec1 %% vec2, "\n")
cat ("Power operator :", vec1 ^ vec2)

```

Output:

```
Addition of vectors : 3 7
Subtraction of vectors : -1 1
Multiplication of vectors : 2 12
Division of vectors : 0.5 1.333333
Modulo of vectors : 1 1
Power operator : 1 64
```

Logical

```r
vec1 <- c(0,2)
vec2 <- c(TRUE,FALSE)

# Performing operations on Operands
cat ("Element wise AND :", vec1 & vec2, "\n")
cat ("Element wise OR :", vec1 | vec2, "\n")
cat ("Logical AND (if both element TRUE then true) :", vec1 && vec2, "\n")
cat ("Logical OR (if any element TRUE then true):", vec1 || vec2, "\n")
cat ("Negation :", !vec1)

```

Output:

```
Element wise AND : FALSE FALSE
Element wise OR : TRUE TRUE
Logical AND (if both element TRUE then true) : FALSE
Logical OR (if any element TRUE then true): TRUE
Negation : TRUE FALSE
```

Operator

Relational

```r
vec1 <- c(0, 2)
vec2 <- c(2, 3)

# Performing operations on Operands
cat ("Vector1 less than Vector2 :", vec1 < vec2, "\n")
cat ("Vector1 less than equal to Vector2 :", vec1 <= vec2, "\n")
cat ("Vector1 greater than Vector2 :", vec1 > vec2, "\n")
cat ("Vector1 greater than equal to Vector2 :", vec1 >= vec2, "\n")
cat ("Vector1 not equal to Vector2 :", vec1 != vec2, "\n")

```

Output:

```
Vector1 less than Vector2 : TRUE TRUE
Vector1 less than equal to Vector2 : TRUE TRUE
Vector1 greater than Vector2 : FALSE FALSE
Vector1 greater than equal to Vector2 : FALSE FALSE
Vector1 not equal to Vector2 : TRUE TRUE
```

Assignment

```r
vec1 <- c(2:5)
c(2:5) ->> vec2
vec3 <<- c(2:5)
vec4 = c(2:5)
c(2:5) -> vec5

# Performing operations on Operands
cat ("vector 1 :", vec1, "\n")
cat("vector 2 :", vec2, "\n")
cat ("vector 3 :", vec3, "\n")
cat("vector 4 :", vec4, "\n")
cat("vector 5 :", vec5)
```

Output:

```
vector 1 : 2 3 4 5
vector 2 : 2 3 4 5
vector 3 : 2 3 4 5
vector 4 : 2 3 4 5
vector 5 : 2 3 4 5
```

Input / Output in R

Input :

```
# Taking input using readLine() this command will prompt you to input a desired value
var <- readLines("stdin",n=1)

# convert the inputted value to integer
var1 <- as.integer(var)

# print the value
print(var1)
```

Output :

```
x = "GeeksforGeeks"
cat(x, "is best\n")

# print normal string
cat("This is R language")
```

```
x <- "GeeksforGeeks"

# using paste inside print()
paste(x, "is best (paste inside print())")

# using paste0 inside print()
paste0(x, "is best (paste0 inside print())")
```

```
# print string
print("GFG")

# print variable
# it will print 'GeeksforGeeks' on the console
x <- "GeeksforGeeks"
print(x)
```

FOR – IF/ELSE in R

IF–ELSE

VAT has different rate according to the product purchased. Imagine we have three different kind of products with different VAT applied:

Categories	Products	VAT
A	Book, magazine, newspaper, etc..	8%
B	Vegetable, meat, beverage, etc..	10%
C	Tee-shirt, jean, pant, etc..	20%

```
> category <- 'A'
> price <- 10
> if (category =='A'){
+   cat('A vat rate of 8% is applied.','The total price is',price *1.08)
+ } else if (category =='B'){
+   cat('A vat rate of 10% is applied.','The total price is',price *1.10)
+ } else {
+   cat('A vat rate of 20% is applied.','The total price is',price *1.20)
+ }
A vat rate of 8% is applied. The total price is 10.8
```

FOR Loop

```
> # Create fruit vector
> fruit <- c('Apple', 'Orange', 'Passion fruit', 'Banana')
> # Create the for statement
> for ( i in fruit){
+   print(i)
+ }
[1] "Apple"
[1] "Orange"
[1] "Passion fruit"
[1] "Banana"
```

```
> # Create an empty list
> list <- c()
> # Create a for statement to populate the list
> for (i in seq(1, 4, by=1)) {
+   list[[i]] <- i*i
+ }
> print(list)
[[1]]
[1] 1

[[2]]
[1] 4

[[3]]
[1] 9

[[4]]
[1] 16
```

Assignment - 1

1) Write a R program to check whether a number is even or odd

2) Write a R program to calculate the factorial of a number

3) Write a R program to calculate the sum of n natural number.

4) Write a R program to find out an element from array of number using linear search.

5) Write a R program which count the specific value and in between a range in a given vector

User Defined Function in R

```
# Create a function to print squares of numbers in sequence.

nf <- function(a) {
  for(i in 1:a) {
    b <- i^2
    print(b)
  }
}

# Call the function nf supplying 6 as an argument.
nf(6)
```

```
my_function <- function(x) {
  return (5 * x)
}

print(my_function(3))
print(my_function(5))
print(my_function(9))
```

```
my_function <- function(country = "Norway") {
  paste("I am from", country)
}

my_function("Sweden")
my_function("India")
my_function() # will get the default value, which is Norway
my_function("USA")
```

Output:

```
[1] "I am from Sweden"
[1] "I am from India"
[1] "I am from Norway"
[1] "I am from USA"
```

Some Library Function in R

apply() used in R?

The apply() function lets us apply a function to the rows or columns of a matrix or data frame. This function takes matrix or data frame as an argument along with function and whether it has to be applied by row or column and returns the result in the form of a vector or array or list of values obtained.

Syntax: apply(x, margin, function) **margin = 1 -> Row ; margin = 2 -> Col**

```
# create sample data
sample_matrix <- matrix(C<-(1:10),nrow=3, ncol=10)
print( "sample matrix:")
sample_matrix

# Use apply() function across row to find sum
print("sum across rows:")
apply( sample_matrix, 1, sum)

# use apply() function across column to find mean
print("mean across columns:")
apply( sample_matrix, 2, mean)
```

Write a R program to print only current system date and current system date with time.

```
print(Sys.time())
print(Sys.Date())
```

Assignment -2

1) Write a R program to find the factors of a number (using function)

2) Write a R program to print below mentioned figure (using function)

```
1
12
123
1234
12345
```

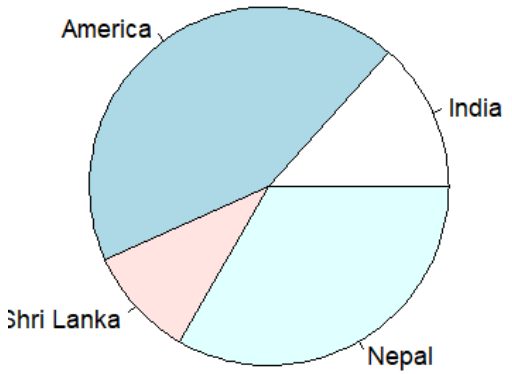
3) Write a R program to print below mentioned figure (using function)

```
    1
   123
  12345
 1234567
123456789
12345678901
```

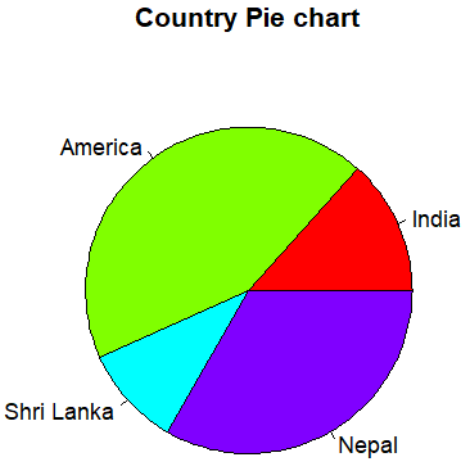
Data Visualization in R

Pie Chart

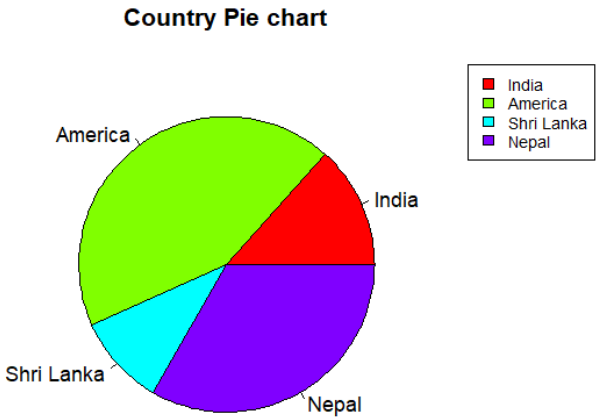
```
> # Creating data for the graph.
> x <- c(20, 65, 15, 50)
> labels <- c("India", "America", "Shri Lanka", "Nepal")
> # Plotting the chart.
> pie(x,labels)
```




```
> pie(x,labels,main="Country Pie chart",col=rainbow(length(x)))
```



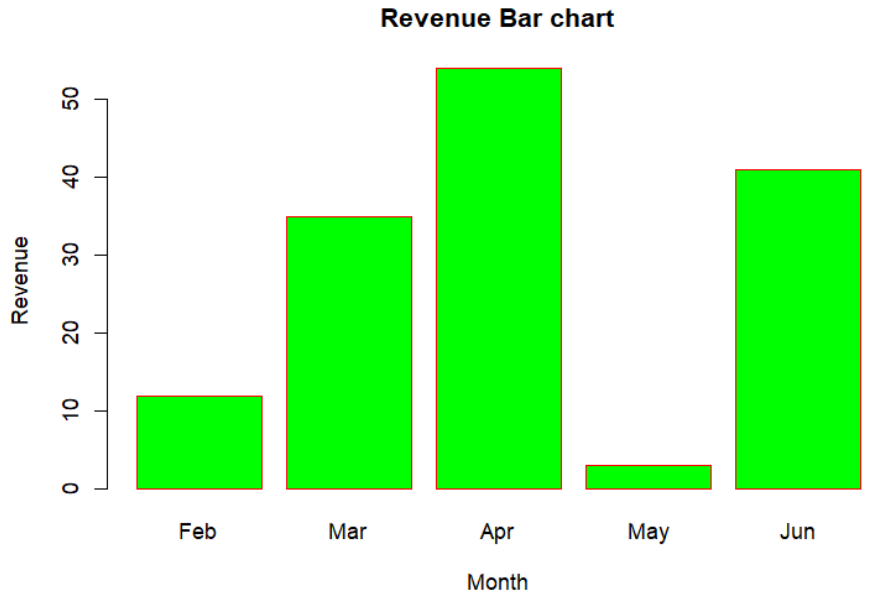

```
> legend("topright", c("India", "America", "Shri Lanka", "Nepal"), cex = 0.8,
+        fill = rainbow(length(x)))
```



Data Visualization in R

Bar Chart

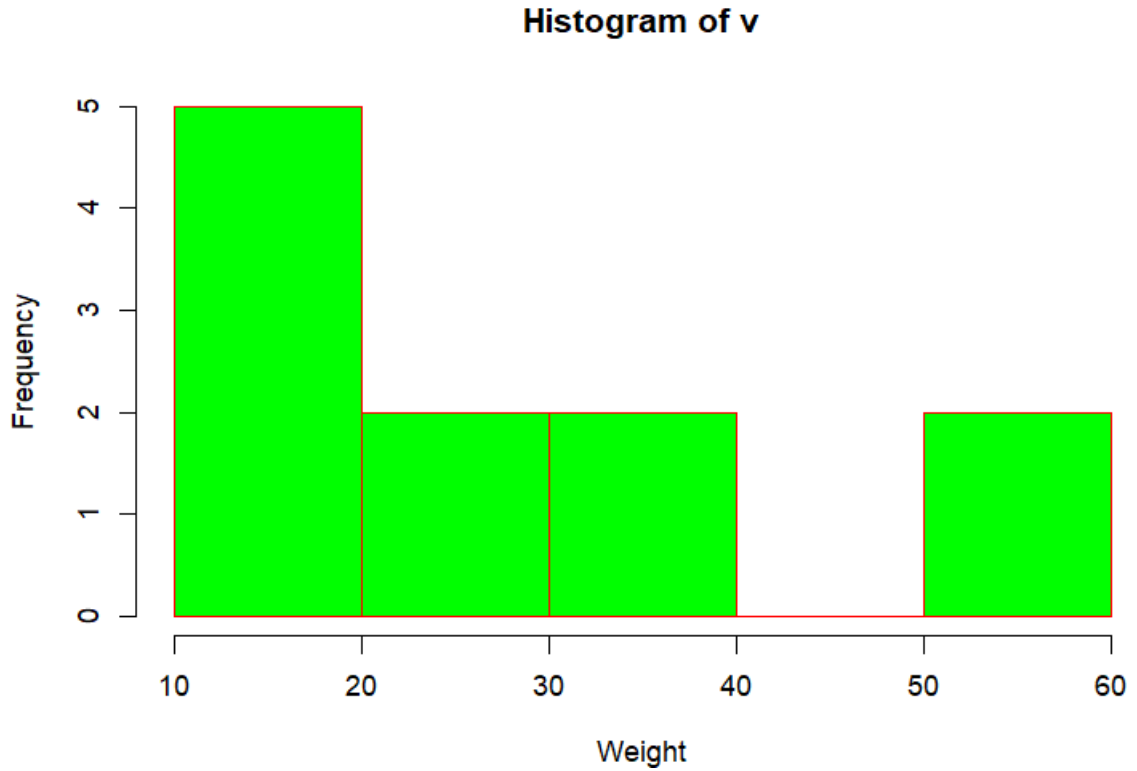
```
> # Creating the data for Bar chart
> H <- c(12,35,54,3,41)
> M<- c("Feb","Mar","Apr","May","Jun")
>
> # Plotting the bar chart
> barplot(H,names.arg=M,xlab="Month",ylab="Revenue",col="Green",
+         main="Revenue Bar chart",border="red")
```



Data Visualization in R

Histogram

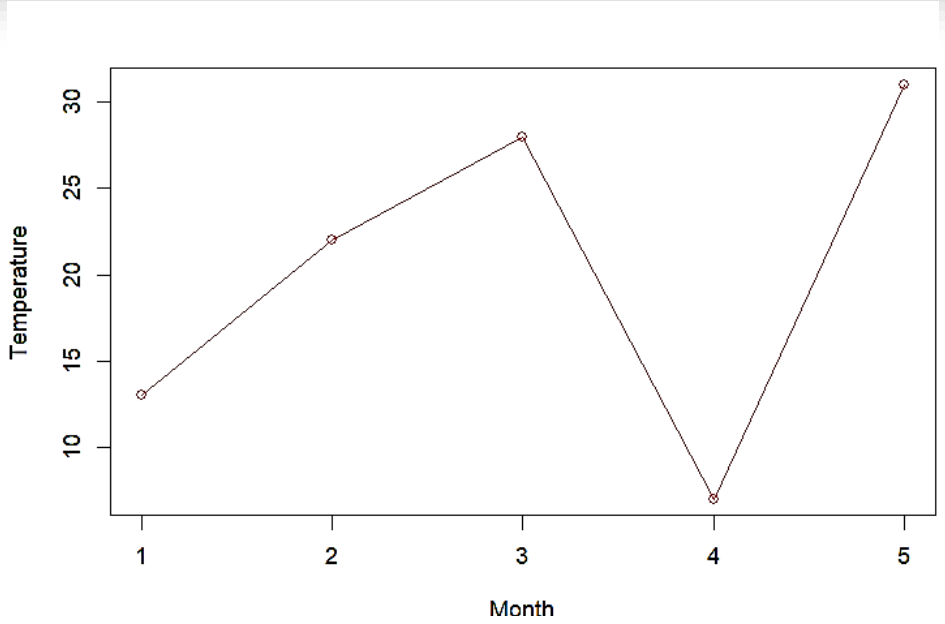
```
> # Creating data for the graph.
> v <-  c(12,24,16,38,21,13,55,17,39,10,60)
>
> # Creating the histogram.
> hist(v,xlab = "Weight",ylab="Frequency",col = "green",border = "red")
```



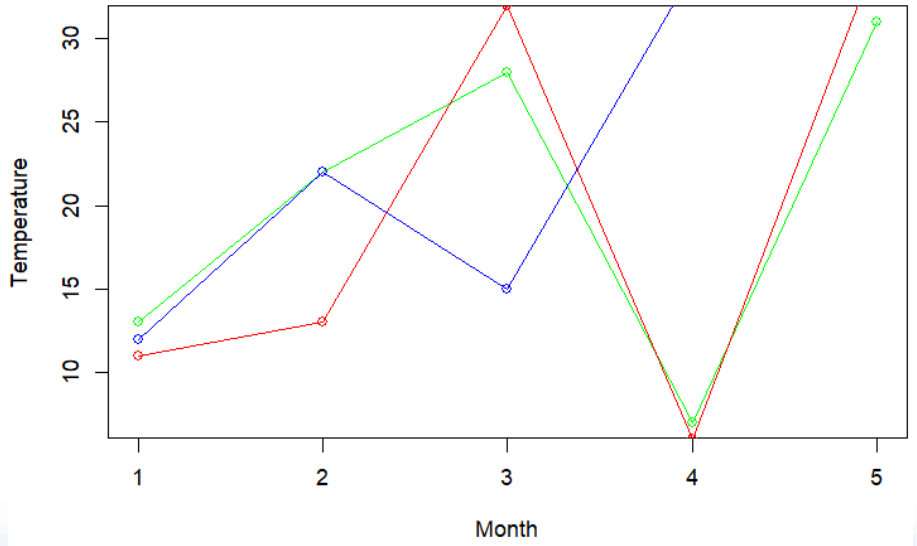
Data Visualization in R

Line Chart

```
> # Creating the data for the chart.
> v <- c(13,22,28,7,31)
>
> # Plotting the bar chart.
> plot(v,type = "o",col="red",xlab="Month",ylab="Temperature")
```



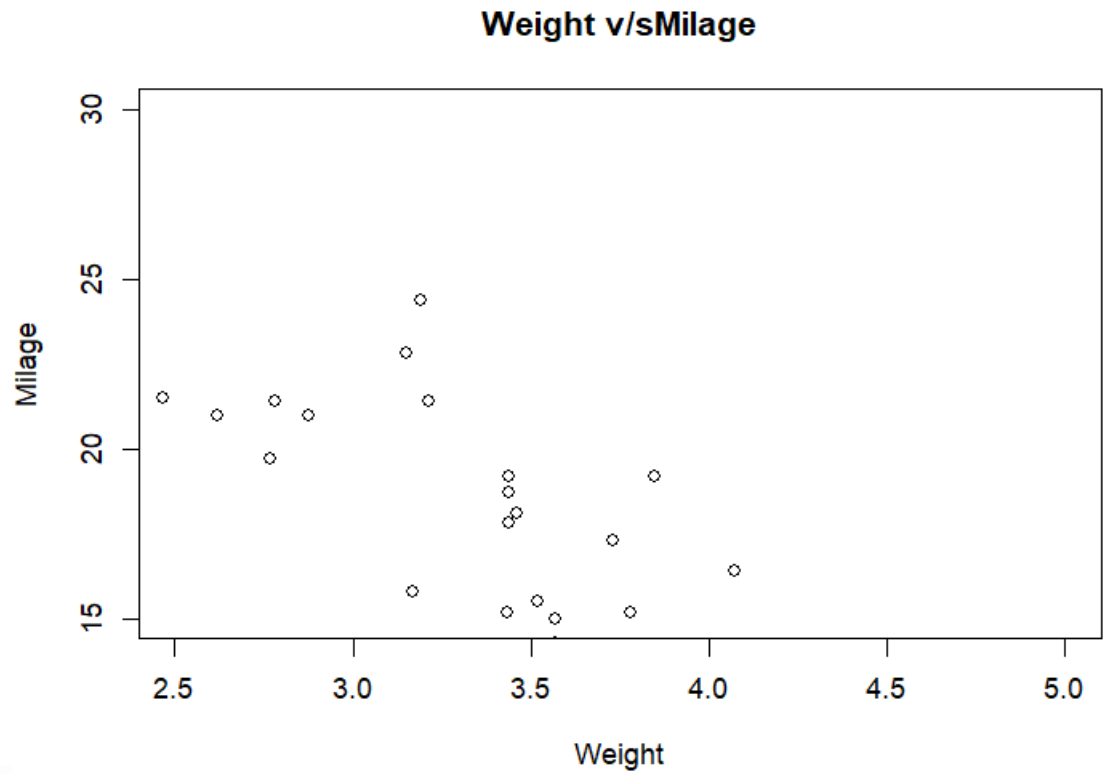

```
# Creating the data for the chart.
v <- c(13,22,28,7,31)
w <- c(11,13,32,6,35)
x <- c(12,22,15,34,35)

# Plotting the bar chart.
plot(v,type = "o",col="green",xlab="Month",ylab="Temperature")
lines(w, type = "o", col = "red")
lines(x, type = "o", col = "blue")
```



Data Visualization in R

Scatterplot

```
> #Fetching two columns from mtcars
> data <-mtcars[,c('wt','mpg')]
>
> # Plotting the chart for cars with weight between 2.5 to 5 and mileage between 15 and 30.
> plot(x = data$wt,y = data$mpg, xlab = "Weight", ylab = "Milage", xlim = c(2.5,5), ylim = c(15,30), main = "Weight v/sMilage")
```



Assignment -3

- Write a R program to draw the pie chart of student marks (Ram - 20, Shyam – 65 ,Paltu – 15 ,Sumit – 50)
- Write a R program to draw the bar chart for month wise revenue(Feb – 12,Mar – 35,Apr – 54,May – 3,Jun – 41)
- Write a R program to draw the month wise temperature using line chart (1st- 13 , 2nd –22 , 3rd–28 , 4th –7 , 5th– 31)
- Write a R program to draw the age wise speed of car using scatterplot `age=(5,7,8,7,2,2);`
  `speed=(99,86,87,88,111,103)`

Thank you