R programming language
It is used to analyze and visualize data for statistical computing and graphical presentation.

R Features	Details
Used	data analysis, data visualization, data science and machine learning
Draw graphs	pie charts, histograms, box plot, scatter plot, etc.
Platforms	Windows, Mac, Linux
Open source	Yes

R programming Syntax

Program Name	**Program**	**Output**
Hello World	"Hello World!"	Hello World!
R Print Output	print("Hello World!")	Hello World!
Display Numbers	1 3 9	1 3 9
Addition of 2 numbers	5 + 5	10

For program	for (x in 5:9)
	{ print(x) }
		5 6 7 8 9

R Comments used to skip execution and is written after "#"	# In R program "Hello World!"	Hello World!"

R programming Variables

Variable Definition	It is a name used to store data and is declared for its data type.
R Variable declaration	Variable-name = Value
R Variable print	Variable-name

#Example Program
company = "wisdom materials"
age = 99
company
age
	#output
	[1] "wisdom materials"
	[1] 99

R programming Concatenate Elements
Join, two or more elements, by using the paste () function.

Program	**Output**

Age = 99
paste("wisdom materials ", Age)
	[1] "wisdom materials 99"

N1 = 1
N2 = 2
N1 + N2
	3

n1 = 5
companyname = "wisdom materials"
n1 + text
	Error in num + text: non-numeric argument to binary operator. Execution halted

v1 = v2 = "wisdom materials" v1 v2	[1] "wisdom materials" [1] "wisdom materials" [1] "wisdom materials"

R programming Data Types

Definition	Variable has to be declared for its data type.
Basic Data Types	Numeric, Integer, Complex, character, logical(boolean), class() function.

Sample Program

Program	Output

v1 = 5.9
class(v1)
v2 = 999L
class(v2)
v3 = 3i + 6
class(v3)
v4 = "Wisdom materials"
class(v4)
v5 = FALSE
class(v5)
	[1] "numeric"
	[1] "integer"
	[1] "complex"
	[1] "character"
	[1] "logical"

R programming Numbers

Example	**R Numbers**
Numeric	a = 3.5
Integer	b = 6L
Complex	c = 9i

R programming Math

Program	**Output**
Simple Math	5+ 5
Built-in Math Functions	
max(3, 9, 6)	[1] 9
min(6, 3, 9)	[1] 3
sqrt(25)	[1] 5
abs(-2.5)	[1] 2.5
ceiling(1.2) – Upwards	[1] 2
floor(2.4) - Downwards	[1] 2

R programming Escape Characters
These are used to handle illegal characters in a string.

Program	**Output**
var1 = "wisdom “materials", company." var1	Error: unexpected symbol in "var1 = "wisdom "materials" Execution halted
var1 = "wisdom \"materials\", company." var1	"wisdom \"materials\", company."

R programming Escape characters ---- Purpose

\\ **----** Backslash	\t **----** Tab	\r **----** Carriage Return
\n **----** New Line	\b **----** Backspace	

R programming Arithmetic Operators

Operator	**Operation**	**Example**	**Operator**	**Operation**	**Example**
-	Subtraction	x - y	^	Exponent	x ^ y
%%	Modulus (Remainder from division)	x %% y	+	Addition	x + y
*	Multiplication	x * y	%/%	Integer Division	x%/%y
/	Division	x / y			

Example

Program	**Operation Name**	**Example**
10-5	Subtraction	5
10%%5	Modulus (Remainder from division)	0
2*3	Multiplication	6
10/5	Division	2
2^3	Exponent	8
2+3	Addition	5
10%/%3	Integer Division	3

R programming Comparison Operators

Operator	**Name**	**Example**
!=	Not equal	x != y
<	Less than	x < y
<=	Less than or equal to	x <= y
==	Equal	x == y
>	Greater than	x > y
>=	Greater than or equal to	x >= y

R programming Logical Operators: These are used in conditional statements.

Operator	**Details**	**Returns**
!	Logical NOT	False if statement is True
&	Logical AND operator	True if both statements are True
&&	Logical AND operator	True if both statements are True
|	Logical OR operator	True if one of the statement is True
||	Logical OR operator	True if one of the statement is True

R programming If Else statement

If statement Type	**Syntax**	**Details**

Simple If	if (condition)
	{ Block Of Statements }
		If condition is true it executes Block Of Statements.

If Else	if (condition1)
	{ Block1 Of Statements }
	else if (condition2)
	{ Block2 Of Statements }
		If condition1is true it executes Block1 Of Statements otherwise it will check condition2. If condition2 is true it executes Block2 Of Statements.

Nest if else	if (condition1)
	{ Block1 Of Statements }
	else if (condition2)
	{ Block2 Of Statements }
	else
	{ Block3 Of Statements }
		If condition1is true it executes Block1 Of Statements otherwise it will check condition2. If condition2 is true it executes Block2 Of Statements. Otherwise it will execute Block3 Of Statements.

Examples

If statement Type	**Example**	**Output**

Big of two numbers	if (9 > 6)
	{ print("9 is bigger") }
		"9 is bigger"

If Else	if (9 > 9)
	{ print("9 is greater than 9")}
	else if (9 == 9)
	{ print ("9 and 9 are equal") }
		"9 and 9 are equal"

Nest if else	if (6 > 9)
	{ print("6 is big") }
	else if (6 == 9)
	{ print("Both are equal") }
	else { print("9 is big") }
		"9 is big"

R programming for Loop

Program	for (var1 in 5:9)
	{ print(var1) }
		EvenNos = list(2, 4,6)
		for (var1 in EvenNos)
		{ print(var1) }

Output	5 6 7 8 9	2 4 6

R programming Break

Program	EvenNos = list(2, 4,6, 8)
	for (var1 in EvenNos) {
	if (var1 == 6)
	{ break }
	print(var1)
	}
		Note: R supports next if and nested loops concepts.

Output	2 4	

R programming Functions

Definition	**Program**	**Output**

It is a set of lines of code used to perform a particular task.	add = function(n1, n2)
	{ return (n1 + n2) }
	sum = add(2,3)
	sum
		5

R programming Vectors: It is a collection of items which is created using c() function.

Operation on vectors	**Program**	**Output**

Create Vectors	Even_Nos = c(1, 2, 3)
	Even_Nos
	Alphabets = c('A', 'B', 'C')
	Alphabets
		1 2 3
		"A" "B" "C"

Sort Vectors (sort items list.)	Even_Nos = c(3, 2,1)
	Even_Nos
	sort(Even_Nos)
	Alphabets = c("C", "B","A")
	Alphabets
	sort(Alphabets)
		3 2 1
		1 2 3
		"C" "B" "A"
		"A" "B" "C"

Access Vectors	Even_Nos = c(3, 2,1)
	Even_Nos[1]
	Alphabets = c("C", "B","A")
	Alphabets[1]
		3
		"C"

Change an Item	Even_Nos = c(3, 2,1)
	Even_Nos[1] =9
	Even_Nos
		9 2 1

R programming list: It is a collection of items created using list () function.

Operation on list	**Program**	**Output**

Create List	Even_Nos = list (1, 2, 3)
	Even_Nos
	Alphabets = list ('A', 'B', 'C')
	Alphabets
		1 2 3 A B C

Access Lists	Even_Nos = list (3, 2,1) Even_Nos[1]	3

Change an Item	Even_Nos = list (3, 2,1)
	Even_Nos[1] =9
	Even_Nos
		9 2 1

Add List Items	Even_Nos = list (3, 2,1)
	append(Even_Nos, 4)
	Even_Nos
		3 2 1 4
		3 2 1

Remove List Items	Even_Nos = list(3, 6, 9)
	Enos = Even_Nos[-2]
	Enos
		3
		9

R programming Matrices
It consists of rows and columns where the data is stored, by using index we address the elements of a matrix.

Operation on list	**Program**	**Output**
Create matrix	mymat = matrix(c(1,2,3,4), nrow = 2, ncol = 2) mymat	[,1] [,2] [1,] 1 3 [2,] 2 4
Access Matrix Items	mymat = matrix(c(1,2,3,4), nrow = 2, ncol = 2) mymat[1, 2]	[1] 3

Add Columns using Cbind() function	mymat = matrix(c(1,2,3,4), nrow = 2, ncol = 2)
	mymat
	mymat1 = cbind(mymat, c(5,6))
	mymat
		[,1] [,2]
		[1,] 1 3
		[2,] 2 4
		    [,1] [,2] [,3]
		[1,] 1 3 5
		[2,] 2 4 6

Add Rows using rbind () function	mymat = matrix(c(1,2,3,4), nrow = 2, ncol = 2)
	mymat
	mymat2 = rbind(mymat, c(5,6))
	mymat2
		[,1] [,2]
		[1,] 1 3
		[2,] 2 4
		    [,1] [,2]
		[1,] 1 3
		[2,] 2 4
		[3,] 5 6

Rows & columns count using dim()	mymat = matrix(c(1,2,3,4), nrow = 2, ncol = 2)
	dim(mymat)
		[1] 2 2

Matrix Length	mymat = matrix(c(1,2,3,4), nrow = 2, ncol = 2)
	length(mymat)
		[1] 4

R programming Arrays
It is a collection of elements all of similar type. All the elements can be accessed by using array index. Dim keyword is used to mention dimensions of the array.

Program Name	**Program**	**Output**

Create	v1 = c(1,2,3)
	v2 = c(4,5,6,7,8,9)
	# vectors v1, v2
	myarray = array(c(v1,v2),dim = c(3,3,2))
	myarray
		, , 1
		    [,1] [,2] [,3]
		[1,] 1 4 7
		[2,] 2 5 8
		[3,] 3 6 9
		, , 2
		    [,1] [,2] [,3]
		[1,] 1 4 7
		[2,] 2 5 8
		[3,] 3 6 9

Naming Columns and Rows	v1 = c(1,2,3)
	v2 = c(4,5,6,7,8,9)
	column.names = c("Col1","Col2","Col3")
	row.names = c("Row1","Row2","Row3")
	matrix.names = c("Matrix1","Matrix2")
	
	myarray = array(c(v1,v2),dim = c(3,3,2),dimnames = list(row.names,column.names,
	  matrix.names))
	myarray
		, , Matrix1
		    Col1 Col2 Col3
		Row1 1 4 7
		Row2 2 5 8
		Row3 3 6 9
		
		, , Matrix2
		       Col1 Col2 Col3
		Row1 1 4 7
		Row2 2 5 8
		Row3 3 6 9

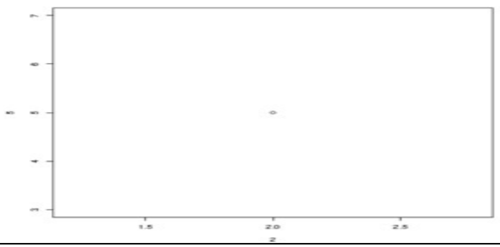
	v1 = c(1,2,3)
	v2 = c(4,5,6,7,8,9)
	column.names = c("Col1","Col2","Col3")
	row.names = c("Row1","Row2","Row3")
	matrix.names = c("Matrix1","Matrix2")
	
	myarray = array(c(v1,v2),dim = (3,3,2),dimnames = list(row.names,column.names,matrix.names))
	
	myarray[3,,2]
	myarray[1,3,1]
	myarray[,,2]
		Col1 Col2 Col3
		  3 6 9
		
		[1] 7
		
		    Col1 Col2 Col3
		Row1 1 4 7
		Row2 2 5 8
		Row3 3 6 9

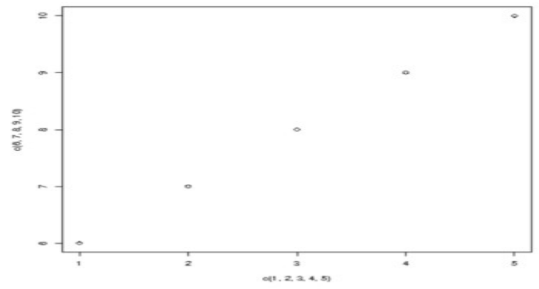
Manipulating Array Elements	v1 = c(1,2,3)
	v2 = c(4,5,6,7,8,9)
	array1 = array(c(v1,v2),dim = c(3,3,2))
	
	v3 = c(9,1,0)
	v4 = c(6,0,11,3,14,1,2,6,9)
	array2 = array(c(v1,v2),dim = c(3,3,2))
	
	# matrices created from arrays.
	m1 = array1[,,2]
	m2 = array2[,,2]
	
	# matrices Addition
	myarray = m1+m2
	myarray
		[,1] [,2] [,3]
		[1,] 2 8 14
		[2,] 4 10 16
		[3,] 6 12 18

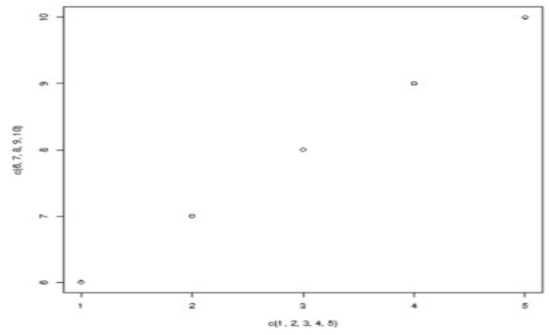
R programming Plotting
The plot() function is used to draw Points, Multiple Points, Point Sequences, Draw Line, Plot Labels and which takes the parameters **x-axis points and y-axis points.**

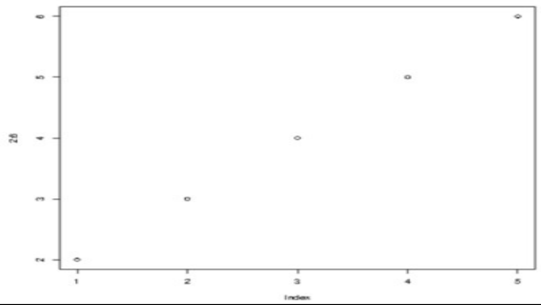
Examples

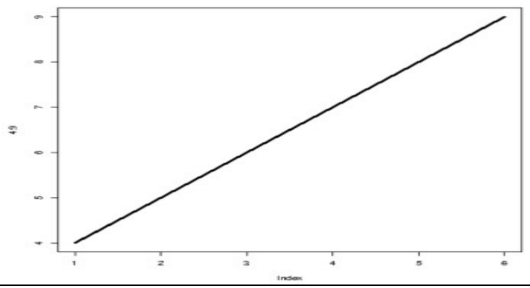
list Operations	**Program**	**Output**

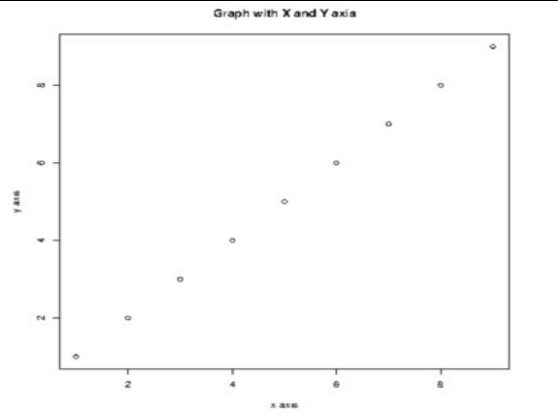
Plot Point	bitmap(file="out.png")
	plot(2, 5)
		

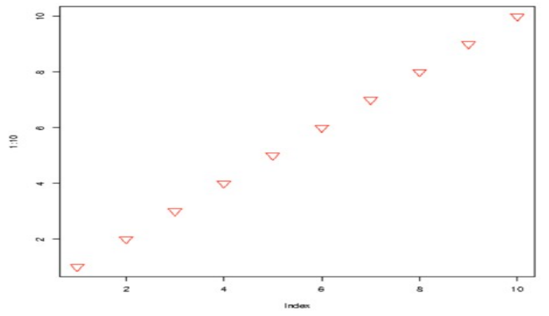
Multiple Points	bitmap(file="out.png")
	plot(c(1, 2, 3, 4, 5), c(6, 7, 8, 9, 10))
		

Multiple Points	bitmap(file="out.png")
	
	p1 = c(1, 2, 3, 4, 5)
	p2 = c(3, 7, 8, 9, 10)
	
	plot(p1, p2)
		

Point Sequences	bitmap(file="out.png")
	plot(2:6)
		

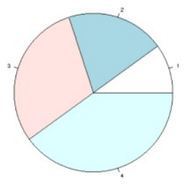
Draw Line	bitmap(file="out.png")
	plot(4:9, type="l", lwd=2)
	#lwd= line width
		

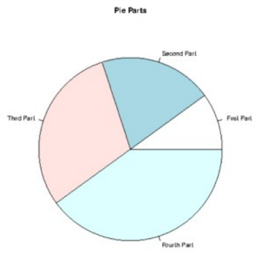
Plot Labels	bitmap(file="out.png")
	plot(1:9, main="Graph with X and Y axis", xlab="x axis", ylab="y axis")
		

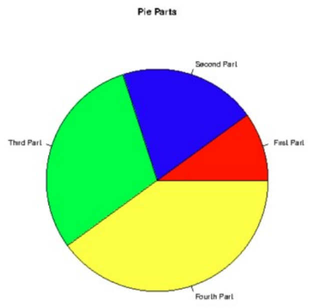
Colors, Size Shape	bitmap(file="out.png") plot(1:10, col="red", pch=25, cex=2) # color=col # size=cex # shape=pch pch values = 0 to 25	

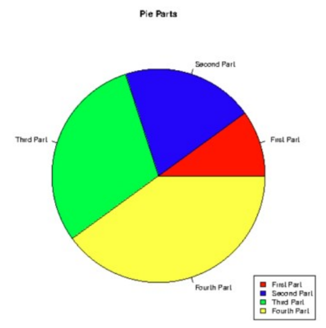
R programming Pie Charts

Pie Charts Operations	**Program**	**Output**

Create	bitmap(file="out.png")
	p1 = c(5,10,15,20)
	pie(p1)
		

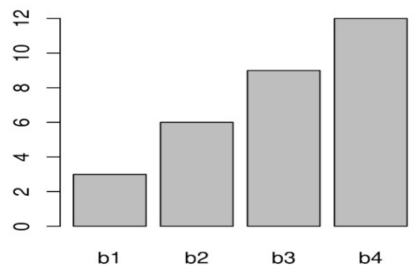
Labels	bitmap(file="out.png")
	p1 = c(5,10,15,20)
	
	mylabels = c("First Part", "Second Part", "Third Part", "Fourth Part")
	
	pie(p1, label = mylabels, main = "Pie Parts")
		

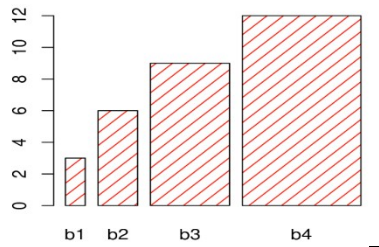
Color	bitmap(file="out.png")
	colors = c("red", "blue", "green", "yellow")
	p1 = c(5,10,15,20)
	
	mylabels = c("First Part", "Second Part", "Third Part", "Fourth Part")
	
	pie(p1, label = mylabels, main = "Pie Parts", , col = colors)
		

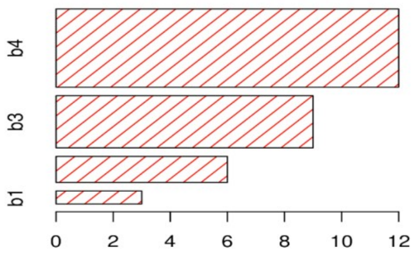
Legend (explanatio n)	bitmap(file="out.png")
	colors = c("red", "blue", "green", "yellow")
	
	p1 = c(5,10,15,20)
	mylabels = c("First Part", "Second Part", "Third Part", "Fourth Part")
	
	pie(p1, label = mylabels, main = "Pie Parts", , col = colors)
	legend("bottomright", mylabels, fill = colors)
		

R programming Bar Charts

list Operations	**Program**	**Output**

create	x = c("b1", "b2", "b3", "b4")
	y = c(3, 6, 9, 12)
	
	barplot(y, names.arg = x)
		

colour, Texture, Bar Width	x = c("b1", "b2", "b3", "b4")
	y = c(3, 6, 9, 12)
	barplot(y, names.arg = x,col = "red", density = 10, c(2,4,8,12))
	#col= colour, Texture=density = 10, Bar Width = width = c(1,2,3,4)
		

Horizontal Bars	x = c("b1", "b2", "b3", "b4")
	y = c(3, 6, 9, 12)
	
	barplot(y, names.arg = x,col = "red", density = 10, c(2,4,8,12), horiz = TRUE)
	
	#col= colour, Texture=density = 10, Bar Width = width = c(1,2,3,4)
		 b4
		b3
		b1
		0 2 4 6 8 10 12