**PRACTICAL FILE**

**of**

**DATA MINING**

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF

**BACHELOR OF TECHNOLOGY**

(Computer Science & Engineering)

**SUBMITTED BY: SUBMITTED TO:**

Aqir Ahmad Lone ER.TARUN DHIMAN

(7318107) (HOD CSE)

****

**Department of Computer Science & Engineering**

**Galaxy Global Educational Trust’s Group of Institutions**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **S.no.** | **Topic** | **Page No.** |
| 1 | Introduction to R | 3 |
| 2 | R Data Types | 4-5 |
| 3 | Built-in Functions in R | 5-6 |
| 4 | Vectors in R | 6-9 |
| 5 | Matrix in R | 10-13 |
| 6 | Factors in R | 13-14 |
| 7 | Data Frames in R | 15-18 |
| 8 | Lists in R | 18-21 |
| 9 | Operators in R | 22-27 |
| 10 | Comparison of Matrices and Vectors | 27 |
| 11 | Programs | 28-42 |

**INTRODUCTION TO R**

**R** is a programming language and software environment for statistical analysis, graphics representation and reporting. R was created by Ross Ihaka and Robert Gentleman at the UniversityofAuckland,NewZealand,andiscurrentlydevelopedbytheRDevelopmentCore Team. R is freely available under the GNU General Public License, and pre-compiled binary versions are provided for various operating systems like Linux, Windows and Mac. This programminglanguagewasnamed**R**,basedonthefirstletteroffirstnameofthetwoRauthors (Robert Gentleman and RossIhaka).

In programming languages, we need to use various variables to store various information. Variables are the reserved memory location to store values. As we create a variable in our program, some space is reserved in memory. In R, there are several data types such asinteger, string, etc. The operating system allocates memory based on the data type of the variable and decides what can be stored in the reservedmemory.

**R data types**

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Example** | **Verify** |
| Logical | TRUE, FALSE | v <- TRUE  print(class(v))  it produces the following result −  [1] "logical" |
| Numeric | 12.3, 5, 999 | v <- 23.5  print(class(v))  it produces the following result −  [1] "numeric" |
| Integer | 2L, 34L, 0L | v <- 2L  print(class(v))  it produces the following result −  [1] "integer" |

|  |  |  |
| --- | --- | --- |
| Complex | 3 + 2i | v <- 2+5i  print(class(v))  it produces the following result −  [1] "complex" |
| Character | 'a' , '"good", "TRUE", '23.4' | v <- "TRUE"  print(class(v))  it produces the following result −  [1] "character" |
| Raw | "Hello" is stored as 48 65 6c 6c 6f | v <- charToRaw("Hello") print(class(v))  it produces the following result −  [1] "raw" |

**Built in Functions in R**

A function is a set of statements organized together to perform a specific task. R has a large number of in-built functions and the user can create their own functions.

In R, a function is an object so the R interpreter is able to pass control to the function, along withargumentsthatmaybenecessaryforthefunctiontoaccomplishtheactions.Thefunction in turn performs its task and returns control to the interpreter as well as any result whichmay be stored in otherobjects.

Thefunctionswhicharealreadycreatedordefinedintheprogrammingframeworkareknown as a built-in function. R has a rich set of functions that can be used to perform almost every task for the user. These built-in functions are divided into the following categories based on theirfunctionality.

Simple examples of in-built functions are seq(), mean(), max(), sum(x) and paste(...) etc. They are directly called by user written programs.

Programs based on built in functions in R:-

*x = c(21, 20, 30)*

*print("Sum:")*

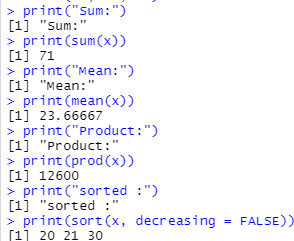
*print(sum(x))*

*print("Mean:")*

*print(mean(x)) print("Product:") print(prod(x)) print("sorted :")*

*print(sort(x, decreasing = FALSE))*

**Output:**



*#seq() is used for a range print(seq(32,44))*

*x<- 4*

*#abs()= absolute value of input print(abs(x))*

*#sqrt()= square root of input value print(sqrt(x))*

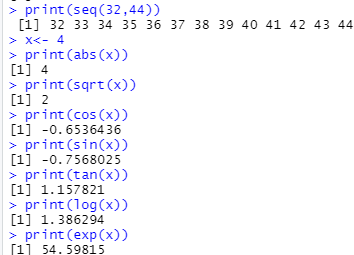
*#cos(x), sin(x), tan(X) value of input x. print(cos(x))*

*print(sin(x))*

*print(tan(x))*

*#log (x)= logarithmic value of x print(log(x)) #exp(x)=exponential value of x print(exp(x))*

**Output:**



# Vectors in R

A vector is a basic data structure which plays an important role in R programming.

In R, a sequence of elements which share the same data type is known as vector. A vector supportslogical,integer,double,character,complex,orrawdatatype.Theelementswhichare contained in vector known as components of the vector. We can check the type of vector with the help of the typeof()function.

The length is an important property of a vector. A vector length is basically the number of elements in the vector, and it is calculated with the help of the length() function.

Creation of vectors

* Single ElementVector

Even when you write just one value in R, it becomes a vector of length 1.

*# Atomic vector of type character. print("abc");*

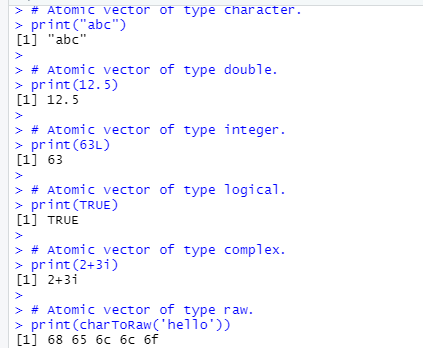
*# Atomic vector of type double. print(12.5)*

*# Atomic vector of type integer. print(63L)*

*# Atomic vector of type logical. print(TRUE)*

*# Atomic vector of type complex. print(2+3i)*

**Output:**



* Multiple ElementsVector
  + Using colon operator with numericdata
  + Using sequence (Seq.) operator
  + Using the c()function

*# Creating a sequence from 5 to 13. v <- 5:13*

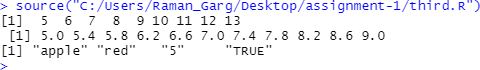
*print(v)*

*# Create vector with elements from 5 to 9 incrementing by 0.4. print(seq(5, 9, by = 0.4))*

*# Using the c(), The logical and numeric values are converted to characters. s <- c('apple','red',5,TRUE)*

*print(s)*

**Output:**



## Accessing vector elements

*# Accessing vector elements using position.*

*t <- c("Sun","Mon","Tue","Wed","Thurs","Fri","Sat") u <- t[c(2,3,6)]*

*print(u)*

*# Accessing vector elements using logical indexing.*

*v <- t[c(TRUE,FALSE,FALSE,FALSE,FALSE,TRUE,FALSE)]*

*print(v)*

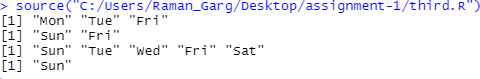
*# Accessing vector elements using negative indexing. x <- t[c(-2,-5)]*

*print(x)*

*# Accessing vector elements using 0/1 indexing. y <- t[c(0,0,0,0,0,0,1)]*

*print(y)*

**Output:**



Vector Manipulation

*v1 <-c(3,8,4,5,0,11)*

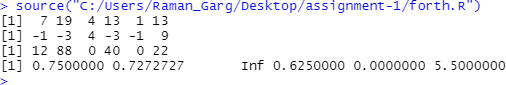
*v2 <-c(4,11,0,8,1,2)*

*# Vector addition. add.result<- v1+v2 print(add.result)*

*# Vector subtraction. sub.result<- v1-v2 print(sub.result)*

*# Vector multiplication. multi.result<- v1\*v2 print(multi.result)*

*# Vector division. divi.result<- v1/v2 print(divi.result)* **Output:**



# Matrix in R

MatricesaretheRobjectsinwhichtheelementsarearrangedinatwo-dimensionalrectangular layout. They contain elements of the same atomic types. Though we can create a matrix containing only characters or only logical values, they are not of much use. We use matrices containing numeric elements to be used in mathematicalcalculations.

A Matrix is created using the **matrix()** function. Syntax

The basic syntax for creating a matrix in R is −

matrix(data, nrow, ncol, byrow, dimnames) Following is the description of the parameters used −

* **data** is the input vector which becomes the data elements of thematrix.
* **nrow**is the number of rows to becreated.
* **ncol**is the number of columns to becreated.
* **byrow**is a logical clue. If TRUE then the input vector elements are arranged byrow.
* **dimname**is the names assigned to the rows andcolumns.

Elements of a matrix can be accessed by using the column and row index of the element.

*# Elements are arranged sequentially by row.* M <- matrix(c(3:14), nrow = 4, byrow = TRUE) print(M)

*# Elements are arranged sequentially by column.*

N <- matrix(c(3:14), nrow = 4, byrow = FALSE) print(N)

*# Define the column and row names.*

rownames = c("row1", "row2", "row3", "row4")

colnames = c("col1", "col2", "col3")

P <- matrix(c(3:14), nrow = 4, byrow = TRUE, dimnames = list(rownames, colnames)) print(P)

*# Access the element at 3rd column and 1st row.*

print(P[1,3])

*# Access the element at 2nd column and 4th row.*

print(P[4,2])

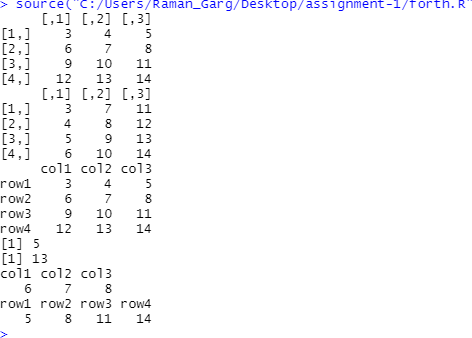
*# Access only the 2nd row.*

print(P[2,])

*# Access only the 3rd column.*

print(P[,3])

**Output:**



Matrix Manipulation

Various mathematical operations are performed on the matrices using the R operators. The result of the operation is also a matrix.

matrix1 <- matrix(c(3, 9, -1, 4, 2, 6), nrow = 2) print(matrix1)

matrix2 <- matrix(c(5, 2, 0, 9, 3, 4), nrow = 2) print(matrix2)

*# Add the matrices.*

result <- matrix1 + matrix2 cat("Result of addition","\n") print(result)

*# Subtract the matrices*

result <- matrix1 - matrix2 cat("Result of subtraction","\n") print(result)

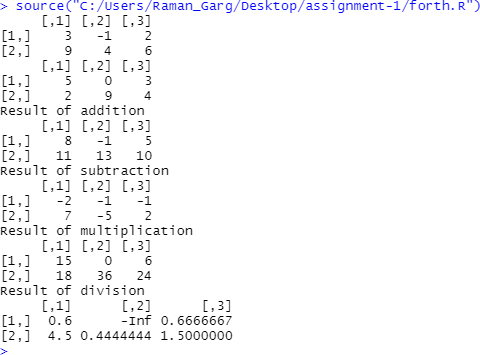
*# Multiply the matrices.*

result <- matrix1 \* matrix2 cat("Result of multiplication","\n") print(result)

*# Divide the matrices*

result <- matrix1 / matrix2 cat("Result of division","\n") print(result)

**Output:**



# Factors in R

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. They are useful in data analysis for statistical modelling.

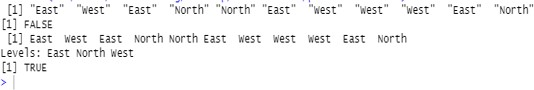
Factors are created using the **factor ()** function by taking a vector as input.

*# Create a vector as input.*

Data<-c("East","West","East","North","North","East","West","West","West","East"

,"North") print(data) print(is.factor(data))

*# Apply the factor function.*

factor\_data<- factor(data) print(factor\_data) print(is.factor(factor\_data)) **Output:**

### Operation on Factors

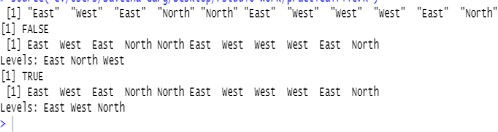
* Changing the Order ofLevels

The order of the levels in a factor can be changed by applying the factor function again with new order of the levels.

*# Apply the factor function with required order of the level.*

new\_order\_data<- factor(factor\_data,levels = c("East","West","North")) print(new\_order\_data)

**Output:**



* Generating FactorLevels

We can generate factor levels by using the gl() function. It takes two integers as input which indicates how many levels and how many times each level.

Syntax:-

gl(n, k, labels)

Following is the description of the parameters used −

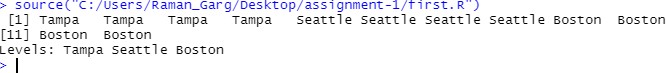
* n is a integer giving the number oflevels.
* k is a integer giving the number ofreplications.
* labels is a vector of labels for the resulting factorlevels.

Example—

v <- gl(3, 4, labels = c("Tampa", "Seattle","Boston"))

print(v)

**Output:**



# Data Frames in R

A data frame is a table or a two-dimensional array-like structure in which each column contains values of one variable and each row contains one set of values from each column.

Following are the characteristics of a data frame.

* The column names should benon-empty.
* The row names should beunique.
* The data stored in a data frame can be of numeric, factor or charactertype.
* Each column should contain same number of data items.Creating Data Frames

In R, the data frames are created with the help of frame() function of data. This function contains the vectors of any type such as numeric, character, or integer. For example:

*# Creating Data frame*

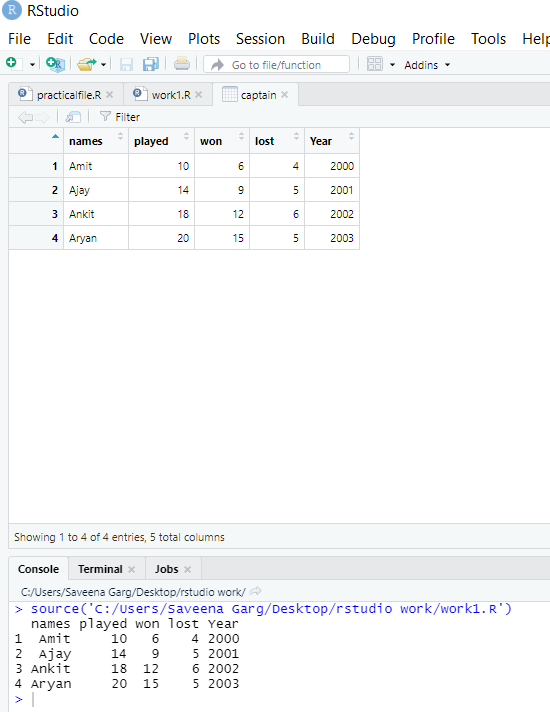
names <- c("Amit","Ajay","Ankit","Aryan") played <- c(10,14,18,20)

won <- c(6,9,12,15)

lost <- c(4,5,6,5)

Year <- c(2000,2001,2002,2003)

captain <- data.frame(names,played,won,lost,Year)

*# Printing Data data*print(captain) View(captain) **Output:**

Operations on Data Frames

* Getting the structure of R DataFrame
* Extracting data from DataFrame
* Modification in Data Frame – adding or deleting rows andcolumns
* Plotting data of a data frame in agraph

*#Getting the structure of R data frame*

str(captain)

*#Printing the summary*

print(summary(captain))

*#extracting data from a data frame*

final <- captain[1,] print(final)

ratio <- captain $won/ captain $played captain $victory <- ratio

print(ratio)

*#plotting data of data frame*

plot(captain $Year,ratio)

*#Adding row in the data frame* x <- list("Ajinkya",5,4,1,2003) rbind(captain,x)

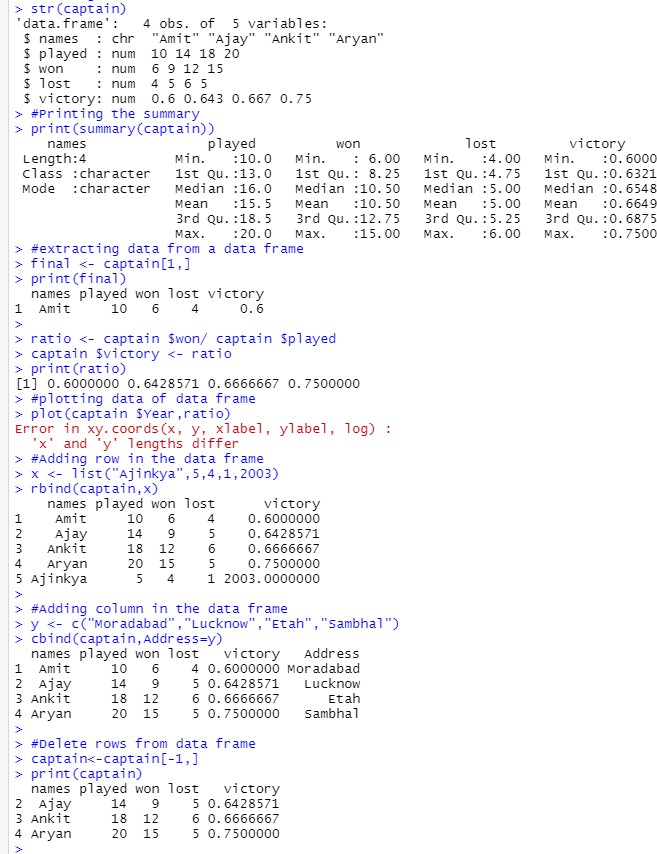
*#Adding column in the data frame*

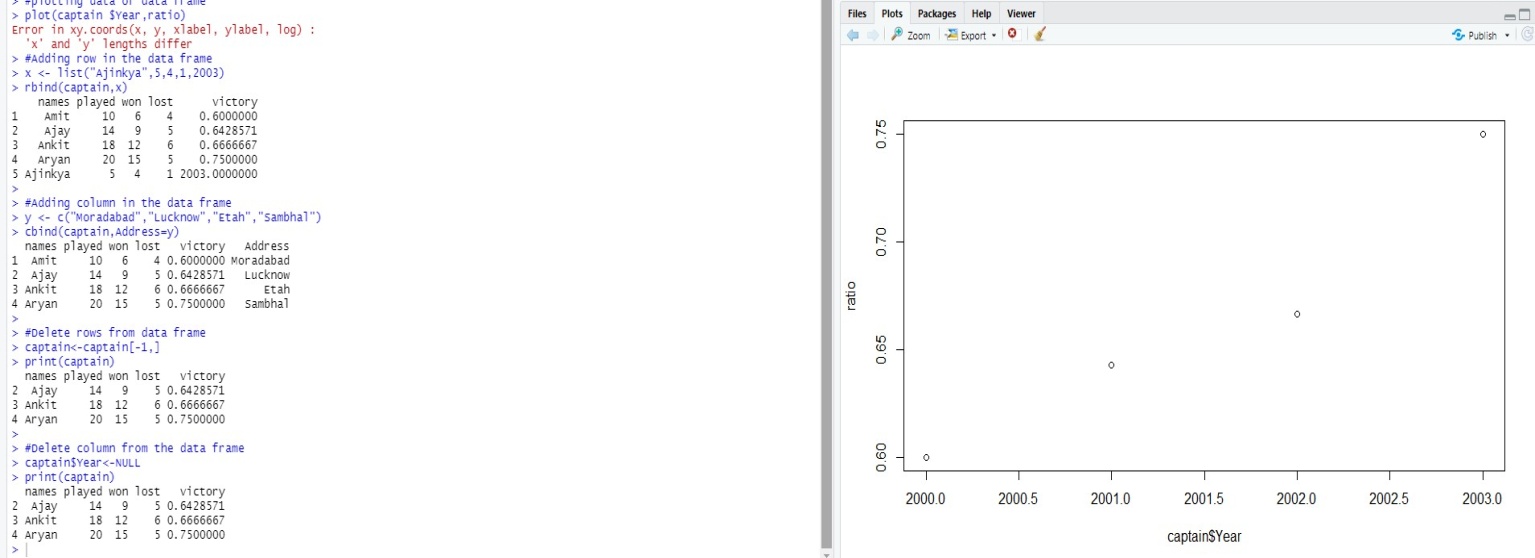
y <- c("Moradabad","Lucknow","Etah","Sambhal") cbind(captain,Address=y)

*#Delete rows from data frame* captain<-captain[-1,] print(captain)

*#Delete column from the data frame* captain$Year<-NULL print(captain)

**Output:**





# Lists in R

In R, lists are the second type of vector. Lists are the objects of R which contain elements of different types such as number, vectors, string and another list inside it. It can also contain a

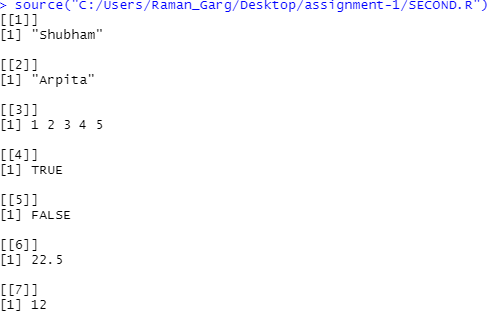
function or a matrix as its elements. A list is a data structure which has components of mixed data types. We can say, a list is a generic vector which contains other objects.

Lists creation

The process of creating a list is the same as a vector. In R, the vector is created with the help of c() function. Like c() function, there is another function, i.e., list() which is used to create a list in R. A list avoid the drawback of the vector which is data type. We can add the elements in the list of different data types. For example:-

list\_data<-list("Shubham","Arpita",c(1,2,3,4,5),TRUE,FALSE,22.5,12L) print(list\_data)

**Output:**



Operations on Lists

* Giving a name to listelements
* Accessing ListElements
* Manipulation of listelements

*# Creating a list containing a vector, a matrix and a list.*

list\_data<- list(c("Shubham","Nishka","Gunjan"), matrix(c(40,80,60,70,90,80), nrow = 2), list("BCA","MCA","B.tech"))

*# Giving names to the elements in thelist.*

names(list\_data) <- c("Students", "Marks", "Course")

*# Accessing the third element. The third element is also a list, so all its elements will be printed.*

print(list\_data[3])

*#Accessing elements using names*

print(list\_data$Marks)

*# Adding element at the end of the list.* list\_data[4] <- "Moradabad" print(list\_data[4])

*# Removing the last element.*

list\_data[4] <- NULL

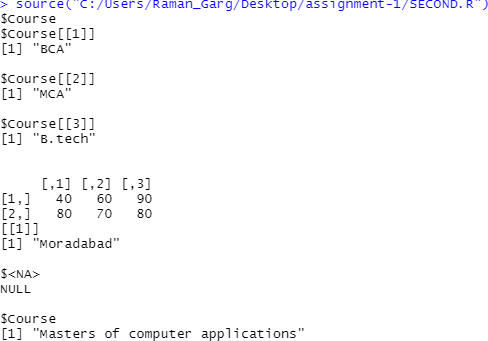
*# Printing the 4th Element.*

print(list\_data[4])

*# Updating the 3rd Element.*

list\_data[3] <- "Masters of computer applications" print(list\_data[3])

**Output:**



# Converting list to vector

* **MergingLists**

*# Creating two lists.* Even\_list<- list(2,4,6) Odd\_list<- list(1,3)

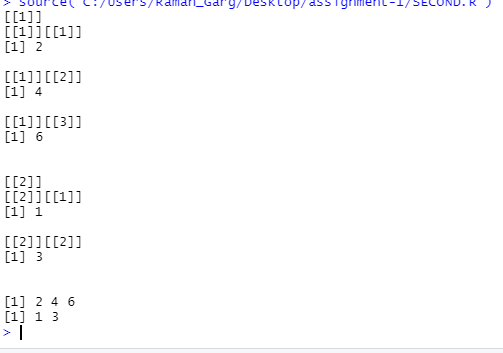
*# Merging the two lists*.

merged.list<- list(Even\_list,Odd\_list) *# Printing the merged list.* print(merged.list)

*# Converting the lists to vectors.*

v1 <- unlist(Even\_list) v2 <- unlist(Odd\_list)

print(v1) print(v2) **Output:**



# Operators in R

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. R language is rich in built-in operators and provides following types of operators.

Types of Operators

We have the following types of operators in R programming −

* ArithmeticOperators
* RelationalOperators
* LogicalOperators
* AssignmentOperators
* MiscellaneousOperators

### Arithmetic Operators

Following table shows the arithmetic operators supported by R language. The operators act on each element of the vector.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + | Adds two vectors | v <- c( 2,5.5,6)  t <- c(8, 3, 4)  print(v+t)  it produces the following result − [1] 10.0 8.510.0 |
| − | Subtracts second vector from the first | v <- c( 2,5.5,6)  t <- c(8, 3, 4)  print(v-t)  it produces the following result − [1] -6.0 2.52.0 |
| \* | Multiplies both vectors | v <- c( 2,5.5,6)  t <- c(8, 3, 4)  print(v\*t)  it produces the following result − [1] 16.0 16.5 24.0 |

|  |  |  |
| --- | --- | --- |
| / | Divide the first vector with the second | v <- c( 2,5.5,6)  t <- c(8, 3, 4)  print(v/t)  When we execute the above code, it produces the following result −  [1] 0.250000 1.833333 1.500000 |
| %% | Give the remainder of the first vector with the second | v <- c( 2,5.5,6)  t <- c(8, 3, 4)  print(v%%t)  it produces the following result − [1] 2.0 2.5 2.0 |
| %/% | The result of division of first vector with second (quotient) | v <- c( 2,5.5,6)  t <- c(8, 3,4)  print(v%/%t)  it produces the following result − [1] 0 1 1 |
| ^ | The first vector raised to the exponent of second vector | v <- c( 2,5.5,6)  t <- c(8, 3, 4)  print(v^t)  it produces the following result − [1] 256.000 166.375 1296.000 |

### Relational Operators

Following table shows the relational operators supported by R language. Each element of the first vector is compared with the corresponding element of the second vector. The result of comparison is a Boolean value.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| > | Checks if each element of the first vector is greater than the corresponding element of the second vector. | v <- c(2,5.5,6,9)  t <- c(8,2.5,14,9)  print(v>t)  it produces the following result −  [1] FALSE TRUE FALSE FALSE |

|  |  |  |
| --- | --- | --- |
| < | Checks if each element of the first vector is less than the corresponding element of the second vector. | v <- c(2,5.5,6,9)  t <- c(8,2.5,14,9)  print(v <t)  it produces the following result −  [1] TRUE FALSE TRUE FALSE |
| == | Checks if each element of the first vector is equal to the corresponding element of the second vector. | v <- c(2,5.5,6,9)  t <- c(8,2.5,14,9)  print(v == t)  it produces the following result −  [1] FALSE FALSEFALSETRUE |
| <= | Checks if each element of the first vector is less than or equal to the corresponding element of the second vector. | v <- c(2,5.5,6,9)  t <- c(8,2.5,14,9)  print(v<=t)  it produces the following result −  [1] TRUE FALSE TRUETRUE |
| >= | Checks if each element of the first vector is greater than or equal to the corresponding element of the second vector. | v <- c(2,5.5,6,9)  t <- c(8,2.5,14,9)  print(v>=t)  it produces the following result −  [1] FALSE TRUE FALSETRUE |
| != | Checks if each element of the first vector is unequal to the corresponding element of the second vector. | v <- c(2,5.5,6,9)  t <- c(8,2.5,14,9)  print(v!=t)  it produces the following result −  [1] TRUE TRUETRUE FALSE |

### Logical Operators

Following table shows the logical operators supported by R language. It is applicable only to vectors of type logical, numeric or complex. All numbers greater than 1 are considered as logical value TRUE.

Each element of the first vector is compared with the corresponding element of the second vector. The result of comparison is a Boolean value.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| & | It is called Element-wise Logical AND operator. It combines each element of the first vector with the corresponding element of the second vector and gives a output TRUE if both the elements are TRUE. | v <- c(3,1,TRUE,2+3i) t <- c(4,1,FALSE,2+3i)  print(v&t)  it produces the following result −  [1] TRUE TRUE FALSETRUE |
| | | It is called Element-wise Logical OR operator. It combines each element of the first vector with the corresponding element of the second vector and gives a output TRUE if one the elements is TRUE. | v <- c(3,0,TRUE,2+2i) t <- c(4,0,FALSE,2+3i)  print(v|t)  it produces the following result −  [1] TRUE FALSE TRUETRUE |
| ! | It is called Logical NOT operator. Takes each element of the vector and gives the opposite logical value. | v <- c(3,0,TRUE,2+2i)  print(!v)  it produces the following result −  [1] FALSE TRUE FALSE FALSE |

Thelogicaloperator&&and||considersonlythefirstelementofthevectorsandgiveavector of single element asoutput.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| && | Called Logical AND operator. Takes first element of both the vectors and gives the TRUE only if both are TRUE. | v <- c(3,0,TRUE,2+2i)  t <- c(1,3,TRUE,2+3i)  print(v&&t)  it produces the following result −  [1] TRUE |

|  |  |  |
| --- | --- | --- |
| || | Called Logical OR operator. Takes first element of both the vectors and gives the TRUE if one of them is TRUE. | v <- c(0,0,TRUE,2+2i)  t <- c(0,3,TRUE,2+3i)  print(v||t)  it produces the following result −  [1] FALSE |

### Assignment Operators

These operators are used to assign values to vectors.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| <−  or  =  or  <<− | Called Left Assignment | v1 <- c(3,1,TRUE,2+3i) v2 <<- c(3,1,TRUE,2+3i) v3 = c(3,1,TRUE,2+3i)  print(v1) print(v2) print(v3)  it produces the following result − [1] 3+0i 1+0i 1+0i2+3i  [1] 3+0i 1+0i 1+0i2+3i [1] 3+0i 1+0i 1+0i2+3i |
| ->  or  ->> | Called Right Assignment | c(3,1,TRUE,2+3i) ->v1 c(3,1,TRUE,2+3i) ->>v2  print(v1) print(v2)  it produces the following result − [1] 3+0i 1+0i 1+0i2+3i  [1] 3+0i 1+0i 1+0i2+3i |

### Miscellaneous Operators

These operators are used to for specific purpose and not general mathematical or logical computation.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| : | Colon operator. It creates the series of numbers in | v <-2:8  print(v)  it produces the following result − [1] 2 3 4 5 6 7 8 |

|  |  |  |
| --- | --- | --- |
|  | sequence for a vector. |  |
| %in% | This operator is used to identify if an element belongs to a vector. | v1 <- 8  v2 <-12  t <-1:10  print(v1 %in% t) print(v2 %in% t)  it produces the following result −  [1] TRUE  [1] FALSE |
| %\*% |  | M = matrix( c(2,6,5,1,10,4), nrow= 2,ncol = 3,byrow = |
|  |  | TRUE) |
|  | This operator is | t = M %\*% t(M) |
|  | used to multiply a | print(t) |
|  | matrix with its transpose. | it produces the following result − [,1] [,2] |
|  |  | [1,] 6582 |
|  |  | [2,] 82117 |

# Comparison of matrices and vectors in R

A **vector** is a basic data structure which plays an important role in R programming.

In R, a sequence of elements which share the same data type is known as vector. A vector supports logical, integer, double, character, complex, or raw data type,

A vector is created using the **c()** function.

Whereas

A **matrix** is a two-dimensional vector (fixed size, all cell types the same). Matrices are the R objects in which the elements are arranged in a two-dimensional rectangular layout. They contain elements of the same atomic types. Though we can create a matrix containing only charactersoronlylogicalvalues,theyarenotofmuchuse.Weusematricescontainingnumeric elements to be used in mathematicalcalculations.

A Matrix is created using the **matrix()** function.

# Program on If-else statement in R

Anif-elsestatementistheifstatementfollowedbyanelsestatement.Anif-elsestatement,else statement will be executed when the boolean expression will false. In simple words, If a Boolean expression will have true value, then the if block gets executed otherwise, the else block will getexecuted.

R programming treats any non-zero and non-null values as true, and if the value is either zero or null, then it treats them as false. The basic syntax of If-else statement is as follows:

*if(boolean\_expression) {*

*// statement(s) will be executed if the boolean expression is true.*

*} else {*

*// statement(s) will be executed if the boolean expression is false.*

*}*

*#Example 1*

x <- c("Hardwork","is","the","key","of","success") if("key" %in% x) {

print("key is found")

} else {

print("key is not found")

}

*#Example 2*

a<- 100

#checking boolean condition if(a<20){

# if the condition is true then print the following cat("a is less than 20\n")

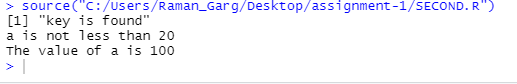
}else{

# if the condition is false then print the following cat("a is not less than 20\n")

}

cat("The value of a is", a)

# OUTPUT



**Program on For loops in R**

InR,aforloopisawaytorepeatasequenceofinstructionsundercertainconditions.Itallows us to automate parts of our code which need repetition. In simple words, a for loop is a repetition control structure. It allows us to efficiently write the loop that needs to execute a certain number oftime.

There is a following syntax of for loop in R:

for (value in vector) { statements

}

Example1

# Creating an empty list list<- c()

# Creating a for statement to populate the list for (i in seq(1, 5, by=1)) {

list[[i]] <- i\*i

}

print(list)

Example 2

# Create a list with three vectors

fruit <- list(Basket = c('Apple', 'Orange',"Guava", 'Pinapple', 'Banana','Grapes'), Money = c(10, 12, 15), purchase = TRUE)

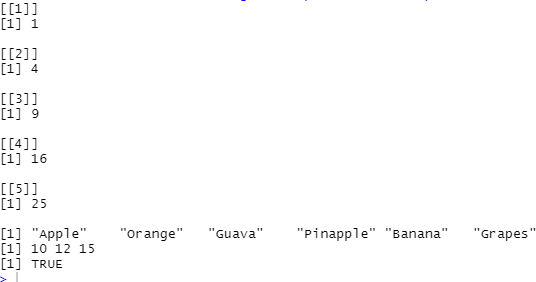
for (p in fruit)

{

print(p)

}

# OUTPUT:



**Program on while loops in R**

Awhileloopisatypeofcontrolflowstatementswhichisusedtoiterateablockofcodeseveral numbersoftimes.ThewhileloopterminateswhenthevalueoftheBooleanexpressionwillbe false.

Inwhileloop,firstlytheconditionwillbecheckedandthenafterthebodyofthestatementwill execute. In this statement, the condition will be checked n+1 time, rather than ntimes.

The basic syntax of while loop is as follows:

while (test\_expression) { statement

}

Example 1

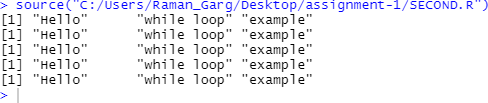
v <- c("Hello","whileloop","example") cnt<- 2

while (cnt< 7) { print(v)

cnt = cnt + 1

}

**Output :-**



# Program to print "Hello world" using R.

mystring<- "Hello World!" print (mystring)

**Output:**



# R program to find the Factorial of a Number by taking input from user.

myfun<- function()

{

z <- TRUE

while( z )

{

x <- readline( prompt="Enter an integer: " ) if ( x < 1 ) {

z <- FALSE

} else {

y <- factorial( as.numeric( x ) ) print( y )

}

}

}

myfun()

**Output:**



# R program to Check whether a number is prime or not.

isprime<- function(n) { lim<- n/2

prime <- T

for( i in 2:lim) { if(n %% i == 0) prime <- FALSE

}

if(n==2) prime <- T

if(prime) print(paste(n," is a Prime Number")) else print(paste(n," is a Composite Number"))

}

print(isprime(10))

**Output:**



# R program to perform Sum, Mean, Product ,Min,Max and sorting of Vector.

x = c(21, 20, 30)

print("Sum:")

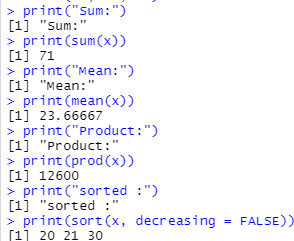
print(sum(x))

print("Mean:")

print(mean(x)) print("Product:") print(prod(x)) print("sorted :")

print(sort(x, decreasing = FALSE))

**Output:**



# R program to make simple calculator.

Add <- function(a,b)

{

c=a+b print(paste("Resultis",c))

}

Subt<- function(a,b)

{

c=a-b

print(paste("Result is",c))

}

Multi <- function(a,b)

{

c=a\*b print(paste("Resultis",c))

}

Div<- function(a,b)

{

c=a/b

print(paste("Result is",c))

}

print("1.Addition") print("2.Subtraction") print("3.MUltiplication") print("4.Division")

choice<-as.integer(readline(prompt="Enter your choice 1/2/3/4 :")) n1<-as.double(readline(prompt="Enter first number :"))

n2<-as.double(readline(prompt="Enter second number :")) if(choice==1)

{

Add(n1,n2)

}else if(choice==2)

{

Subt(n1,n2)

}else if(choice==3)

{

Multi(n1,n2)

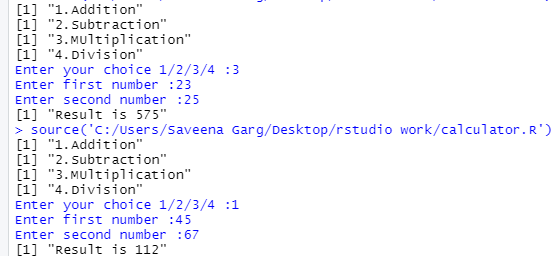
}else if(choice==4)

{

Div(n1,n2)

}

**Output:**



# R program to create histogram, pie chart and bar chart from a csv file

*#Clear R workspace*

rm(list=ls())

*#Declare a variable to read and store moviesData*

movies <- read.csv("moviesData.csv")

*#View the stored data frame*

View(movies)

*# Creating histogram*

hist(movies$runtime, main = "This will be Title of histogram", #adding label to x-axis

xlab = "Runtime of movies ",

*#to set the range of values in x-axis*

xlim = c(0,300),

*#to set the colour of histogram*

col = "blue", breaks = 4,)

*# Creating Pie chart*

*# to count the frequency in column genre of movies dataset*

count <- table(movies$genre) *#to design pie chart* pie(count,

main ="Title of pie chart", border = "blue",

col = "pink")

*# Creating bar chart*

moviessub<- movies[1:20,] barplot(moviessub$imdb\_rating,

ylab = "Title at Y-axis", xlab = "Title at X-axis", col = "Red",

*#To set the range of values on Y-axis*

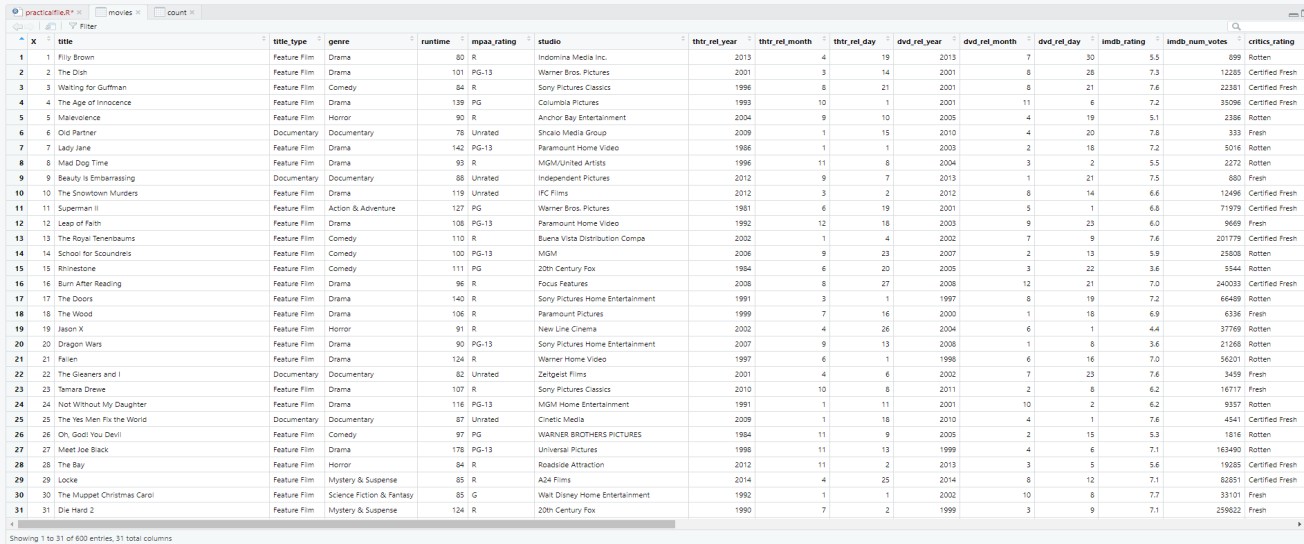
ylim = c(0,10),

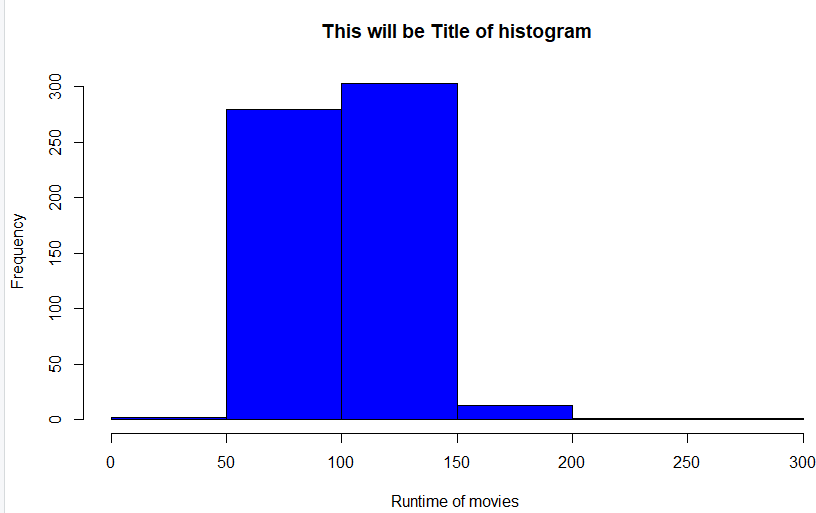
main= "Title of Bar Chart",

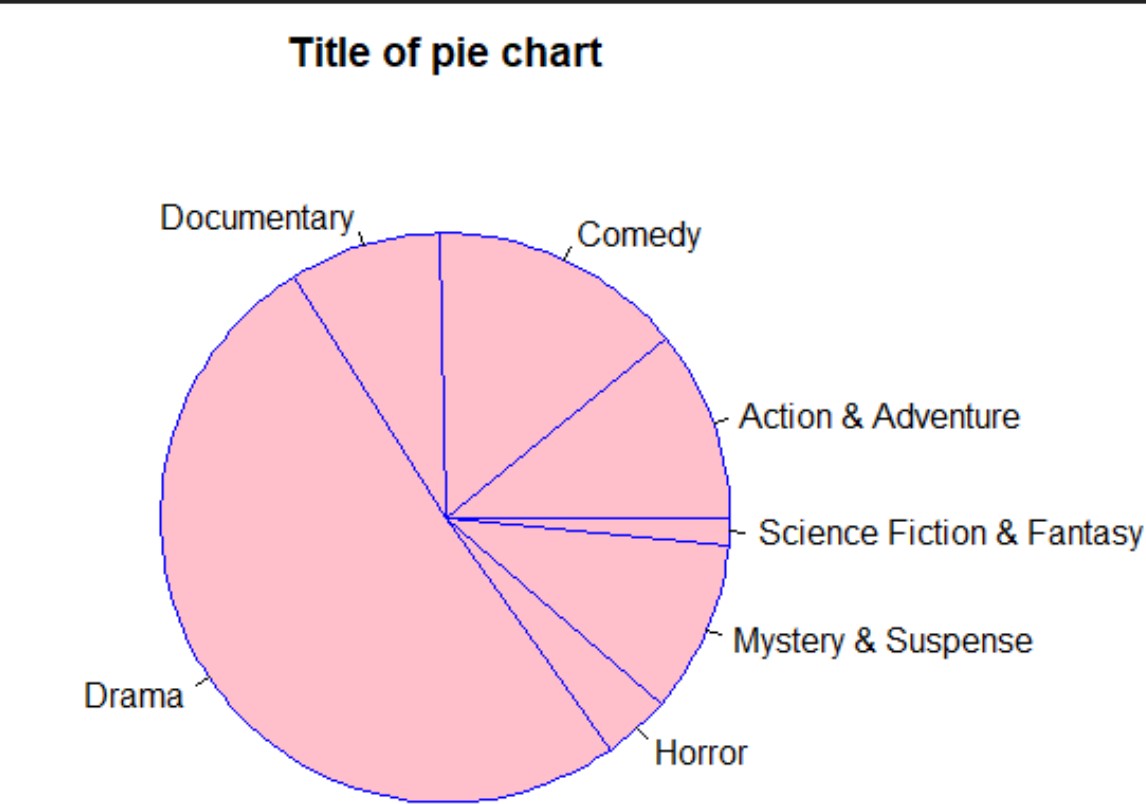
*# To display the name of movies on X-axis*

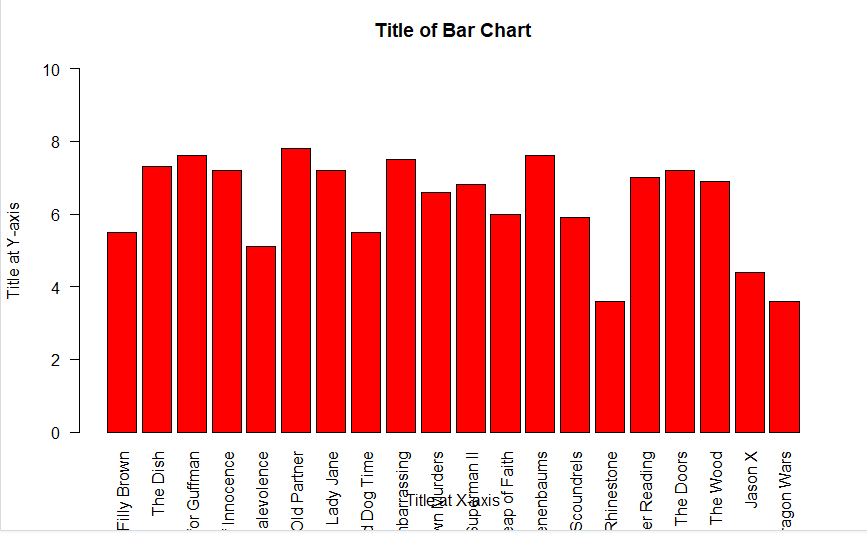
names.arg = moviessub$title, las=2)

**Output:**









# R program to use dplyr package and perform data manipulation functions

library(dplyr) rm(list = ls())

movies <- read.csv("moviesData.csv")

*# View movies data frame*

View(movies)

*# Data manipulation functions: filter, arrange, select, mutate, summarise*moviescomdr<- filter(movies, genre %in% c("Comedy","Drama")) View(moviescomdr)

moviescomim<- filter(movies, genre == "Comedy" &imdb\_rating>= 7.5) View(moviescomim)

*#arrange data in ascending order* moviesina<- arrange(movies, imdb\_rating) View(moviesina)

*#arrange data in descending order*

moviesind<- arrange(movies, desc(imdb\_rating)) View(moviesind)

*#to select specific columns*

moviessel<- select(movies,title, genre, imdb\_rating) View(moviessel)

*#to select column based on starting words* moviesthtr<- select(movies,title, starts\_with("thtr")) View(moviesthtr)

*#to rename a column/variable name*

moviesr<- rename(movies, rel\_year = "thtr\_rel\_year") View(moviesr)

*#to view only few columns of data frame* moviesfew<- select(movies, title:audience\_score) View(moviesfew)

*#to add a new variable in data frame*

moviesmu<- mutate(moviesfew, criaud = critics\_score-audience\_score) View(moviesmu)

**Output:**

