Aman Raj-200103020

(DSA Individual Assignment)

12/19/2020

Submitted To:

Prof. (Dr.) Sridhar Vaithianathan,

Associate Professor (Analytics).

IMT - Hyderabad.

CONTENTS

1	Introduction
2	Functions
3	Data Types-R
4	Vectors
5	Dataframes
6	Factors
7	Missing Data
8	Matrices
9	Arrays
10	List
11	Read and Write Files
12	Heatmap
13	Hypothesis Testing
14	Linear Regression

INTRODUCTION

```
#Creating Simple Objects and Doing Mathematical Calculation

a = 5                #Value 5 is assigned to Variable a
a

## [1] 5

b = 10
b

## [1] 10

class(a)             # DataType of variable a

## [1] "numeric"

a = "Hello"
class(a)

## [1] "character"

a = TRUE
class(a)

## [1] "logical"

a = FALSE
class (a)

## [1] "logical"

#Object Assignments and Simple Calculations
x = 10
y = 15
x+y                  #Sum of x and y

## [1] 25

x-y

## [1] -5

x*y

## [1] 150

x/y

## [1] 0.6666667
```

```r
sqrt(x)

## [1] 3.162278

x^y

## [1] 1e+15

exp(x)

## [1] 22026.47

log(x, base=exp(1))

## [1] 2.302585

log10(x)

## [1] 1

help("log")      #Utilizing R Help

## starting httpd help server ... done

factorial(x)

## [1] 3628800

cos(x)

## [1] -0.8390715

abs(x)

## [1] 10
```

- We do not need to define the data type in R

FUNCTIONS

```
getwd()       #Get Working Directory

## [1] "C:/Users/Pankaj/Documents/IMT-G/TERM 2/R PROGRAMMING"

# Functions in R
divider = function(x,y) {
  result = x/y
  print(result)
}
divider(50,25)

## [1] 2

divider (100,25)

## [1] 4

# Multiplication
multiply = function(a,b){
  return (a * b)       #Directly returns the value
}
multiply(23,25)

## [1] 575

multiply (19,20)

## [1] 380

#Variables Names are CASE SENSITIVE
A=10
a=24

A       #Prints value for A

## [1] 10

#CONCATENATION AND ARRAYS

f <- c(1,2,3,4,5)
f = c(1,2,3,4,5)
f

## [1] 1 2 3 4 5

f+4             #Adds 4 to each element

## [1] 5 6 7 8 9
```

```
d = f / 4
d

## [1] 0.25 0.50 0.75 1.00 1.25

f+d

## [1] 1.25 2.50 3.75 5.00 6.25
```

R is case sensitive

Return keyword is used to return the value

Rm(object) is used to remove the object created

Not explicitly declaring variables.

DATA TYPES-R

```
x = 10
class(x)

## [1] "numeric"

# Numeric - Integer and Decimal - (R)- Integer (Whole Number) and Numeric
(Float - Decimal)
i = 5L # L - Integer
class(i)

## [1] "integer"

is.integer(i)

## [1] TRUE

is.numeric(x)

## [1] TRUE

# Logical - TRUE (1) and FALSE (0)
#R understands value of TRUE as 1 and FALSE as 0
TRUE * 5

## [1] 5
```

```
FALSE * 5

## [1] 0

K = TRUE
class(K)

## [1] "logical"

is.logical(K)

## [1] TRUE

# Date - Starting Date (1970) - Numeric Value.
# In R - 1 Jan 1970
# Date - mm/dd/yyyy
# POSIXct - Date plus Time.

date1 = as.Date("2012-06-28")
# as.Date()# Auto complete # How to enter
# ? as.Date # help
date1

## [1] "2012-06-28"

class (date1)

## [1] "Date"

as.numeric(date1)

## [1] 15519

#POSIXct - Date and Time
date2 = as.POSIXct("2012-06-28 17:42")
date2

## [1] "2012-06-28 17:42:00 IST"

class(date2)

## [1] "POSIXct" "POSIXt"

as.numeric(date2)

## [1] 1340885520
```

VECTORS

```
v = c(1,2,3,4,5)
s = v*2
s

## [1]  2  4  6  8 10

#Vector Operation
d = v-2
d

## [1] -1  0  1  2  3

f = v /2
f

## [1] 0.5 1.0 1.5 2.0 2.5

sqrt(f)

## [1] 0.7071068 1.0000000 1.2247449 1.4142136 1.5811388

numb = c(7,8,9,3,4)
numb

## [1] 7 8 9 3 4

length(numb)   #Size of the vector

## [1] 5

numb[c(1,3)]   #Access 1st and 3rd element

## [1] 7 9

numb = c(7,8,9,3,4)

numb[5]=10    #Assigns 10 to 5th position

numb

## [1]  7  8  9  3 10

sort(numb)  #Arranges in ascending order

## [1]  3  7  8  9 10

#Give Names to Vector!
c(One = "a", Two = "y", Last = "r") # Name-Value pair
```

```
##   One  Two Last
##   "a"  "y"  "r"

w = 1:3    #numbers 1 to 3 is assigned to w
names(w) = c("a","b","c")
w

## a b c
## 1 2 3
```

Vector - R is called as Vectorized language. It is used to store multiple values.

A vector is collection of elements, all of same type.

A vector cannot be of mixed type.

DATAFRAMES

```
x = 10:1
y = -4:5
q = c("Hockey","Football","Baseball","Curlin","Rugby","Lacrosse",
      "Basketball","Tennis","Cricket","Soccer")
theDF = data.frame(x,y,q) # this would create a 10x3 data.frame with x, y and
q as variable names
theDF

##     x  y          q
## 1  10 -4     Hockey
## 2   9 -3   Football
## 3   8 -2   Baseball
## 4   7 -1     Curlin
## 5   6  0      Rugby
## 6   5  1   Lacrosse
## 7   4  2 Basketball
## 8   3  3     Tennis
## 9   2  4    Cricket
## 10  1  5     Soccer

# Checking the dimensions of the DF.
nrow(theDF)

## [1] 10
```

```
ncol(theDF)

## [1] 3

dim(theDF)

## [1] 10  3

names (theDF)

## [1] "x" "y" "q"

names(theDF)[3]

## [1] "q"

rownames(theDF)

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

# Head and Tail
head(theDF)

##    x  y        q
## 1 10 -4   Hockey
## 2  9 -3 Football
## 3  8 -2 Baseball
## 4  7 -1   Curlin
## 5  6  0    Rugby
## 6  5  1 Lacrosse

head(theDF, n=7)

##    x  y          q
## 1 10 -4     Hockey
## 2  9 -3   Football
## 3  8 -2   Baseball
## 4  7 -1     Curlin
## 5  6  0      Rugby
## 6  5  1   Lacrosse
## 7  4  2 Basketball

tail(theDF)

##    x y          q
## 5  6 0      Rugby
## 6  5 1   Lacrosse
## 7  4 2 Basketball
## 8  3 3     Tennis
## 9  2 4    Cricket
## 10 1 5     Soccer

class(theDF)
```

```
## [1] "data.frame"

# Assigning Names
theDF = data.frame (First=x, Second =y, Sport = q)
theDF

##    First Second      Sport
## 1     10     -4     Hockey
## 2      9     -3   Football
## 3      8     -2   Baseball
## 4      7     -1     Curlin
## 5      6      0      Rugby
## 6      5      1   Lacrosse
## 7      4      2 Basketball
## 8      3      3     Tennis
## 9      2      4    Cricket
## 10     1      5     Soccer

# Accessing Individual Column using $

theDF$Sport # gives the third column named Sport

##  [1] "Hockey"     "Football"   "Baseball"   "Curlin"     "Rugby"
##  [6] "Lacrosse"   "Basketball" "Tennis"     "Cricket"    "Soccer"

# Accessing Specific row and column
theDF[3,2] # 3rd row and 2nd Column

## [1] -2

theDF[3,2:3] # 3rd Row and column 2 thru 3

##   Second    Sport
## 3     -2 Baseball

theDF[c(3,5), 2]# Row 3&5 from Column 2;

## [1] -2  0

# since only one column was selected, it was returned as vector and hence no
column names in output.

# Rows 3&5 and Columns 2 through 3
theDF[c(3,5), 2:3]

##   Second    Sport
## 3     -2 Baseball
## 5      0    Rugby

theDF[ ,3] # Access all Rows for column 3

##  [1] "Hockey"     "Football"   "Baseball"   "Curlin"     "Rugby"
##  [6] "Lacrosse"   "Basketball" "Tennis"     "Cricket"    "Soccer"
```

```
theDF[ , 2:3]

##    Second      Sport
## 1      -4     Hockey
## 2      -3   Football
## 3      -2   Baseball
## 4      -1     Curlin
## 5       0      Rugby
## 6       1   Lacrosse
## 7       2 Basketball
## 8       3     Tennis
## 9       4    Cricket
## 10      5     Soccer

theDF[2,]# Access all columns for Row 2

##   First Second    Sport
## 2     9     -3 Football

theDF[2:4,]

##   First Second    Sport
## 2     9     -3 Football
## 3     8     -2 Baseball
## 4     7     -1   Curlin

#Another way to create data frames
custData = data.frame(name=c("Tom", "Sally", "Sue"),
                      age=c(43, 28, 42),
                      stringsAsFactors=T)

custData

##    name age
## 1   Tom  43
## 2 Sally  28
## 3   Sue  42
```

Data Frames(DF) - Most useful features of R & also cited reason for R's ease of use.

In dataframe, each column is actually a vector, each of which has same length.

Each column can hold different type of data.

Also within each column, each element must be of same type, like vectors.

FACTORS

```
#Create a factor vector
q2 = c(q,"Hockey","Lacrosse","Hockey","Water Polo","Hockey","Lacrosse")
q2

##  [1] "Hockey"     "Football"   "Baseball"   "Curlin"     "Rugby"
##  [6] "Lacrosse"   "Basketball" "Tennis"     "Cricket"    "Soccer"
## [11] "Hockey"     "Lacrosse"   "Hockey"     "Water Polo" "Hockey"
## [16] "Lacrosse"

class(q2)

## [1] "character"

as.numeric(q2)

## Warning: NAs introduced by coercion

##  [1] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA

class(q2)

## [1] "character"

#Creating another factor vector
direction = c("Up", "Down", "Left", "Right", "Left", "Up")
factorDir = factor(direction)
factorDir

## [1] Up    Down  Left  Right Left  Up
## Levels: Down Left Right Up

is.factor(factorDir)  #Checks if it is a factor

## [1] TRUE

is.factor(direction)

## [1] FALSE

factorDir   #Only unique values

## [1] Up    Down  Left  Right Left  Up
## Levels: Down Left Right Up

# A Factor object contains levels which store all possible
# values
levels(x=factorDir)

## [1] "Down"  "Left"  "Right" "Up"
```

```
# You can define your levels and their orders
dow = c("Monday", "Tuesday", "Wednesday", "Thursday",
        "Friday", "Saturday", "Sunday")

wDays = c("Tuesday", "Thursday", "Monday")

wdFact = factor(x=wDays, levels=dow, ordered=T)

wdFact

## [1] Tuesday  Thursday Monday
## 7 Levels: Monday < Tuesday < Wednesday < Thursday < Friday < ... < Sunday
```

Factor Vectors - Ordinal data [Ordered Categorical]

Factors are important concept in R, esp. when building models

---

MISSING DATA

```
# NA - Missing data - Missing Value
#NA is the value which is mission in the vector
z = c(1,2,NA,8,3,NA,3)
#z = c(1,2,na,8,3,na,3) -> R does not understand 'na'
z

## [1]  1  2 NA  8  3 NA  3

# "is.na" tests each element of a vector for missingness
is.na(z)

## [1] FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE

# ...If used inside a vector, it simply disappears! Let's see...
z= c(1,NULL,3)
z

## [1] 1 3

x = c(1,NA,3)
x

## [1]  1 NA  3
```

```
# Notice, here the "NULL" didnot get stored in "z", infact "z" has only
length of 2!
length(z) #NULL is not treated as NA and is not stored

## [1] 2

length(x)

## [1] 3

# Assigning NULL and checking!
d = NULL
is.null(d)

## [1] TRUE
```

```
Missing data plays a crucial role in computing and Statistics

R has two types of missing data - NA and NULL

While they are similar, but they behave differently and hence needs
attention!
```

MATRICES

```
# Create a Matrix with a single column
matrix1 = matrix(data=c(1,2,3,4))
matrix1

##      [,1]
## [1,]    1
## [2,]    2
## [3,]    3
## [4,]    4

# Create a matrix with defined rows and columns
matrix2 = matrix(data=c(1,2,3,4), nrow=2, ncol=2)
matrix2

##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
```

```r
# You can also fill by row (You can use T or TRUE)
matrix3 = matrix(data=c(1,2,3,4), nrow=2, ncol=2, byrow=T)
matrix3

##      [,1] [,2]
## [1,]    1    2
## [2,]    3    4

# Get a Matrix dimension
dim(matrix3)

## [1] 2 2

# A value at row, column
matrix3[1,2]

## [1] 2

# Get a whole row
matrix3[1,]

## [1] 1 2

# Get a whole column
matrix3[,2]

## [1] 2 4

# Combine vectors to make a Matrix
matrix4 = rbind(1:3, 4:6, 7:9)
matrix4

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

# Get 2nd and 3rd row
matrix4[2:3,]

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

# Get 2nd and 3rd row by ommitting the 1st
matrix4[-1,]

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

```
# Change the first value
matrix4[1,1] = 0
matrix4

##      [,1] [,2] [,3]
## [1,]    0    2    3
## [2,]    4    5    6
## [3,]    7    8    9

# Change the 1st row
matrix4[1,] = c(10,11,12)
matrix4

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

#creating few more matrices
A = matrix(1:10, nrow=5)# Create a 5x2 matrix
B = matrix(21:30, nrow=5)#Create another 5x2 matrix
C = matrix (21:40, nrow=2)#Create another 2x10 matrix
D = matrix(41:45, ncol = 5) #Creates 1*5 matrix
D

##      [,1] [,2] [,3] [,4] [,5]
## [1,]   41   42   43   44   45

A

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

B

##      [,1] [,2]
## [1,]   21   26
## [2,]   22   27
## [3,]   23   28
## [4,]   24   29
## [5,]   25   30

C

##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]   21   23   25   27   29   31   33   35   37    39
## [2,]   22   24   26   28   30   32   34   36   38    40
```

```
nrow(A)     #Gives the no. of rows of A

## [1] 5

ncol(A)

## [1] 2

dim(A)      #Gives the dimensions of A

## [1] 5 2

# Add the values of the matrices A and B
A+B

##      [,1] [,2]
## [1,]   22   32
## [2,]   24   34
## [3,]   26   36
## [4,]   28   38
## [5,]   30   40

# Multiply Them (Vector Multiplication!)
A

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

B

##      [,1] [,2]
## [1,]   21   26
## [2,]   22   27
## [3,]   23   28
## [4,]   24   29
## [5,]   25   30

A*B  # A = 5x2 and B = 5x2

##      [,1] [,2]
## [1,]   21  156
## [2,]   44  189
## [3,]   69  224
## [4,]   96  261
## [5,]  125  300

#See if the elements are equal
A == B
```

```
##       [,1]  [,2]
## [1,] FALSE FALSE
## [2,] FALSE FALSE
## [3,] FALSE FALSE
## [4,] FALSE FALSE
## [5,] FALSE FALSE

# Matrix Multiplication(MM. A is 5x2. B is 5x2. B-transpose is 2x5
A %*% t(B)

##      [,1] [,2] [,3] [,4] [,5]
## [1,]  177  184  191  198  205
## [2,]  224  233  242  251  260
## [3,]  271  282  293  304  315
## [4,]  318  331  344  357  370
## [5,]  365  380  395  410  425

# Naming the Columns and Rows
colnames(A)

## NULL

rownames(A)

## NULL

colnames(A)= c("Left","Right")
rownames(A)= c("1st","2nd","3rd","4th","5th")
colnames(B)

## NULL

rownames(B)

## NULL

colnames(B)= c("First","Second")
rownames(B)= c("One","Two","Three","Four","Five")
colnames(C)

## NULL

rownames(C)

## NULL

colnames(C) = LETTERS [1:10]
rownames(C) = c("Top", "Bottom")

# Matrix Multiplication. A is 5x2 and C is 2x10
dim(A)

## [1] 5 2
```

```
dim(C)

## [1]  2 10

t(A)

##       1st 2nd 3rd 4th 5th
## Left    1   2   3   4   5
## Right   6   7   8   9  10

A %*% C

##       A   B   C   D   E   F   G   H   I   J
## 1st 153 167 181 195 209 223 237 251 265 279
## 2nd 196 214 232 250 268 286 304 322 340 358
## 3rd 239 261 283 305 327 349 371 393 415 437
## 4th 282 308 334 360 386 412 438 464 490 516
## 5th 325 355 385 415 445 475 505 535 565 595
```

A matrix (plural matrices) is a rectangular array or table of numbers, symbols, or expressions...

Arranged in rows and columns.(i.e.) 2-Dimensional Array

Similar to data.frame(RxC) and also similar to Vector

Matrix - Element by element operations are possible

ARRAYS

```
theArray = array(1:12, dim=c(2,3,2))# Total Elements = R x C x OD
theArray

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

```
theArray [1, ,]# Accessing all elements from Row 1, all columns, all outer
dimensions & build C x OD (R x C)

##      [,1] [,2]
## [1,]    1    7
## [2,]    3    9
## [3,]    5   11

theArray[1, ,1]# Accessing all elements from Row 1, all columns, first outer
dimension

## [1] 1 3 5

theArray[, ,1]# Accessing all rows, all columns, first outer dimension

##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6

# Array with Four Outer Dimensions (OD)
theArray_4D = array(1:32, dim=c(2,4,4))
theArray_4D

## , , 1
##
##      [,1] [,2] [,3] [,4]
## [1,]    1    3    5    7
## [2,]    2    4    6    8
##
## , , 2
##
##      [,1] [,2] [,3] [,4]
## [1,]    9   11   13   15
## [2,]   10   12   14   16
##
## , , 3
##
##      [,1] [,2] [,3] [,4]
## [1,]   17   19   21   23
## [2,]   18   20   22   24
##
## , , 4
##
##      [,1] [,2] [,3] [,4]
## [1,]   25   27   29   31
## [2,]   26   28   30   32

theArray_4D [1, ,]

##      [,1] [,2] [,3] [,4]
## [1,]    1    9   17   25
## [2,]    3   11   19   27
## [3,]    5   13   21   29
## [4,]    7   15   23   31
```

```
theArray_4D[1, ,1]
```

```
## [1] 1 3 5 7
```

```
theArray[, ,1]
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

Arrays - An array is essentially a multidimensional vector.

It must all be of the same type and

individual elements are accessed using Square Brackets.

First element is Row(R) Index, Second Element is Column(C) Index and

the remaining elements are for Outer Dimensions (OD).

LIST

```
list(1,2,3)# creates a three element list
```

```
## [[1]]
## [1] 1
## 
## [[2]]
## [1] 2
## 
## [[3]]
## [1] 3
```

```
list(c(1,2,3))# creates a single element(vector with three elements)
```

```
## [[1]]
## [1] 1 2 3
```

```
list3 = list(c(1,2,3), 3:7)# create two element list
# first is three elements vector, next is five element vector.
list3
```

```
## [[1]]
## [1] 1 2 3
##
## [[2]]
## [1] 3 4 5 6 7

# The same can be written as
(list3 = list(c(1,2,3), 3:7))

## [[1]]
## [1] 1 2 3
##
## [[2]]
## [1] 3 4 5 6 7

# Two Element list
# First element is data.frame and next is 10 element vector
list(theDF, 1:10)# theDF is already created in previous exercise!

## [[1]]
##    First Second      Sport
## 1     10     -4     Hockey
## 2      9     -3   Football
## 3      8     -2   Baseball
## 4      7     -1     Curlin
## 5      6      0      Rugby
## 6      5      1   Lacrosse
## 7      4      2 Basketball
## 8      3      3     Tennis
## 9      2      4    Cricket
## 10     1      5     Soccer
##
## [[2]]
##  [1]  1  2  3  4  5  6  7  8  9 10

# Three element list
list5 = list(theDF, 1:10, list3)
list5

## [[1]]
##    First Second      Sport
## 1     10     -4     Hockey
## 2      9     -3   Football
## 3      8     -2   Baseball
## 4      7     -1     Curlin
## 5      6      0      Rugby
## 6      5      1   Lacrosse
## 7      4      2 Basketball
## 8      3      3     Tennis
## 9      2      4    Cricket
## 10     1      5     Soccer
```

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

#Naming List (similar to column name in data.frame)
names(list5)= c("data.frame", "vector","list")
names(list5)

## [1] "data.frame" "vector"     "list"

list5

## $data.frame
##    First Second      Sport
## 1     10     -4     Hockey
## 2      9     -3   Football
## 3      8     -2   Baseball
## 4      7     -1     Curlin
## 5      6      0      Rugby
## 6      5      1   Lacrosse
## 7      4      2 Basketball
## 8      3      3     Tennis
## 9      2      4    Cricket
## 10     1      5     Soccer
##
## $vector
##  [1]  1  2  3  4  5  6  7  8  9 10
##
## $list
## $list[[1]]
## [1] 1 2 3
##
## $list[[2]]
## [1] 3 4 5 6 7

#Naming using "Name-Value" pair
list6 = list(TheDataFrame = theDF, TheVector = 1:10, TheList = list3)
names(list6)

## [1] "TheDataFrame" "TheVector"    "TheList"

list6

## $TheDataFrame
##    First Second      Sport
## 1     10     -4     Hockey
## 2      9     -3   Football
## 3      8     -2   Baseball
## 4      7     -1     Curlin
## 5      6      0      Rugby
## 6      5      1   Lacrosse
## 7      4      2 Basketball
## 8      3      3     Tennis
## 9      2      4    Cricket
## 10     1      5     Soccer
##
## $TheVector
##  [1]  1  2  3  4  5  6  7  8  9 10
##
## $TheList
## $TheList[[1]]
## [1] 1 2 3
##
## $TheList[[2]]
## [1] 3 4 5 6 7
```

```
# Creating an empty list
(emptylist = vector(mode="list", length =4))
```

```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
```

```
# Accessing individual element of a list - Double Square Brackets
# specify either element number or name
list5[[1]]
```

```
##    First Second      Sport
## 1     10     -4     Hockey
## 2      9     -3   Football
## 3      8     -2   Baseball
## 4      7     -1     Curlin
## 5      6      0      Rugby
## 6      5      1   Lacrosse
## 7      4      2 Basketball
## 8      3      3     Tennis
## 9      2      4    Cricket
## 10     1      5     Soccer
```

```
list5[["data.frame"]]

##    First Second      Sport
## 1     10     -4     Hockey
## 2      9     -3   Football
## 3      8     -2   Baseball
## 4      7     -1     Curlin
## 5      6      0      Rugby
## 6      5      1   Lacrosse
## 7      4      2 Basketball
## 8      3      3     Tennis
## 9      2      4    Cricket
## 10     1      5     Soccer

list5[[1]]$Sport

##  [1] "Hockey"     "Football"   "Baseball"   "Curlin"     "Rugby"
##  [6] "Lacrosse"   "Basketball" "Tennis"     "Cricket"    "Soccer"

list5[[1]][,"Second"]

##  [1] -4 -3 -2 -1  0  1  2  3  4  5

list5[[1]][,"Second", drop = FALSE]

##    Second
## 1      -4
## 2      -3
## 3      -2
## 4      -1
## 5       0
## 6       1
## 7       2
## 8       3
## 9       4
## 10      5

# LENGTH OF LIST
length(list5)

## [1] 3

names(list5)

## [1] "data.frame" "vector"     "list"

list5

## $data.frame
##    First Second      Sport
## 1     10     -4     Hockey
## 2      9     -3   Football
## 3      8     -2   Baseball
## 4       7      -1      Curlin
## 5       6       0       Rugby
## 6       5       1    Lacrosse
## 7       4       2  Basketball
## 8       3       3      Tennis
## 9       2       4     Cricket
## 10      1       5      Soccer
##
## $vector
##  [1]  1  2  3  4  5  6  7  8  9 10
##
## $list
## $list[[1]]
## [1] 1 2 3
##
## $list[[2]]
## [1] 3 4 5 6 7
```

READ AND WRITE FILES

```
myPeople = read.table("People.txt.txt",
                      header=T, sep=" ",
                      na.strings="`",
                      stringsAsFactors=F)
myPeople

##       fname  lname sex
## 1      aman    raj   M
## 2        ms  dhoni   M
## 3   Anushka Sharma   F
## 4    aviral saxena   M
## 5     Ayush  Smith   M
## 6   Anshika   Jain   F
## 7     Rahul Dravid   M
## 8     Rahul Dravid   M
## 9     Rahul Dravid   M
## 10    Rahul Dravid   M

#Reading the files

# Add another person
addname = data.frame(fname="Rahul",
                         lname="Dravid",
                         sex="M")
```

```
myPeople = rbind(myPeople, addname)
myPeople

##       fname  lname sex
## 1      aman    raj   M
## 2        ms  dhoni   M
## 3   Anushka Sharma   F
## 4    aviral saxena   M
## 5     Ayush  Smith   M
## 6   Anshika   Jain   F
## 7     Rahul Dravid   M
## 8     Rahul Dravid   M
## 9     Rahul Dravid   M
## 10    Rahul Dravid   M
## 11    Rahul Dravid   M

# Update a record
myPeople[5,2] = "Smith"
myPeople

##       fname  lname sex
## 1      aman    raj   M
## 2        ms  dhoni   M
## 3   Anushka Sharma   F
## 4    aviral saxena   M
## 5     Ayush  Smith   M
## 6   Anshika   Jain   F
## 7     Rahul Dravid   M
## 8     Rahul Dravid   M
## 9     Rahul Dravid   M
## 10    Rahul Dravid   M
## 11    Rahul Dravid   M

# Update the file by supplying the data.frame,
# the file to write, seperator, na, whether to
# quote strings, whether to include row numbers
write.table(x=myPeople, "People.txt.txt",
            sep=" ", na="`",
            quote=F, row.names=F)

# Get 1st 3 records
head(myPeople, 3)

##      fname  lname sex
## 1     aman    raj   M
## 2       ms  dhoni   M
## 3 Anushka Sharma   F

# Get remaining records
tail(myPeople, 3)
```

```
##    fname  lname sex
## 9  Rahul Dravid   M
## 10 Rahul Dravid   M
## 11 Rahul Dravid   M
```

Read.table is used to read the txt file

Read.csv is used to read the csv file

```
x = sample(x=1:100, size = 20, replace = TRUE)
x  # the output of "x" is a vector of data

##  [1] 85 98 10 86 23 96 75 11 55 12 74 98 44 47 14 16 83 53 36 34

#Also, the 20 numbers are selected randomly
# Simple Arithmetic Mean
mean(x)

## [1] 52.5

# Calculate Mean when Missing Data is found
y = x # copy x to y
y[sample(x=1:100, size = 20, replace = FALSE )] = NA # Null Values
y

##  [1] 85 98 NA 86 23 NA NA NA 55 NA NA NA 44 47 14 16 NA 53 36 34 NA NA NA
NA NA
## [26] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
NA NA
## [51] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
NA NA
## [76] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA

y = sample(x=1:100, size = 20, replace = FALSE)
y

##  [1] 91 51 98 36 24 45 26 41 54  7 22 35 81 76 67 12 40 65 33 19

mean(y)# Will give NA!

## [1] 46.15

# Remove missing value(s)and calculate mean
mean(y, na.rm=TRUE) # Now, it will give the mean value
```

```
## [1] 46.15

# Weighted Mean
Grades = c(95,72,87,66)
Weights = c(1/2, 1/4, 1/8, 1/8)
mean(Grades)# Simple Arithmetic mean

## [1] 80

weighted.mean(x = Grades, w = Weights)# Weighted Mean

## [1] 84.625

#Variance
var(x)

## [1] 1036.158

#Calculating Variance using formula!
sum((x-mean(x))^2)/ (length(x)-1)

## [1] 1036.158

# Standard Deviation
sqrt(var(x))

## [1] 32.18941

sd(x)

## [1] 32.18941

sd(y)

## [1] 26.22027

sd(y, na.rm=TRUE)

## [1] 26.22027

# Other Commonly Used Functions
min(x)

## [1] 10

max(x)

## [1] 98

median(x)

## [1] 50

min(y)
```

```
## [1] 7

min(y, na.rm=TRUE)

## [1] 7

# Summary Statistics
summary(x)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.00   21.25   50.00   52.50   83.50   98.00

summary(y)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    7.00   25.50   40.50   46.15   65.50   98.00

# Quantiles
quantile(x, probs = c(0.25, 0.75)) # Calculate 25th and 75th Quantile

##   25%   75%
## 21.25 83.50

quantile(x, probs = c(0.1,0.25,0.5, 0.75,0.99))

##   10%   25%   50%   75%   99%
## 11.90 21.25 50.00 83.50 98.00

quantile(y, probs = c(0.25, 0.75)) # Calculate 25th and 75th Quantile

##  25%  75%
## 25.5 65.5

quantile(y, probs = c(0.25, 0.75), na.rm = TRUE)

##  25%  75%
## 25.5 65.5
```

HEATMAP

```
# Correlation

# Prepare the Data
mydata <- mtcars[, c(1,3,4,5,6,7)]
head(mydata)

##                    mpg disp  hp drat    wt  qsec
## Mazda RX4         21.0  160 110 3.90 2.620 16.46
## Mazda RX4 Wag     21.0  160 110 3.90 2.875 17.02
## Datsun 710        22.8  108  93 3.85 2.320 18.61
## Hornet 4 Drive    21.4  258 110 3.08 3.215 19.44
## Hornet Sportabout 18.7  360 175 3.15 3.440 17.02
## Valiant           18.1  225 105 2.76 3.460 20.22
```

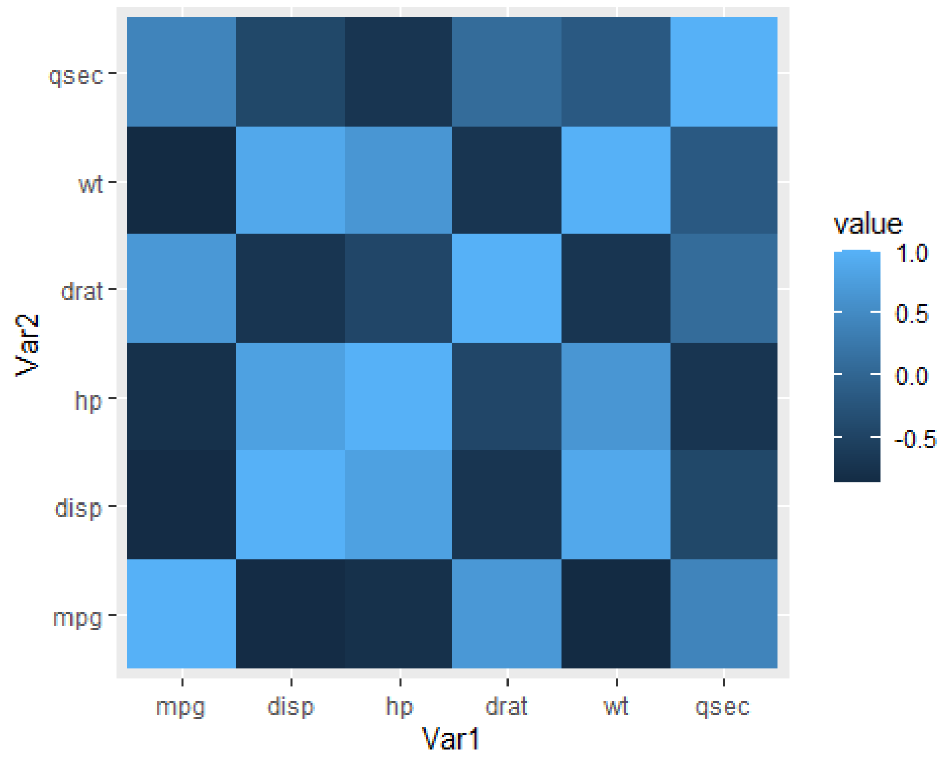
```
# Compute the correlation matrix - cor()
cormat <- round(cor(mydata),2)
head(cormat)
```

```
##        mpg  disp    hp  drat    wt  qsec
## mpg   1.00 -0.85 -0.78  0.68 -0.87  0.42
## disp -0.85  1.00  0.79 -0.71  0.89 -0.43
## hp   -0.78  0.79  1.00 -0.45  0.66 -0.71
## drat  0.68 -0.71 -0.45  1.00 -0.71  0.09
## wt   -0.87  0.89  0.66 -0.71  1.00 -0.17
## qsec  0.42 -0.43 -0.71  0.09 -0.17  1.00
```

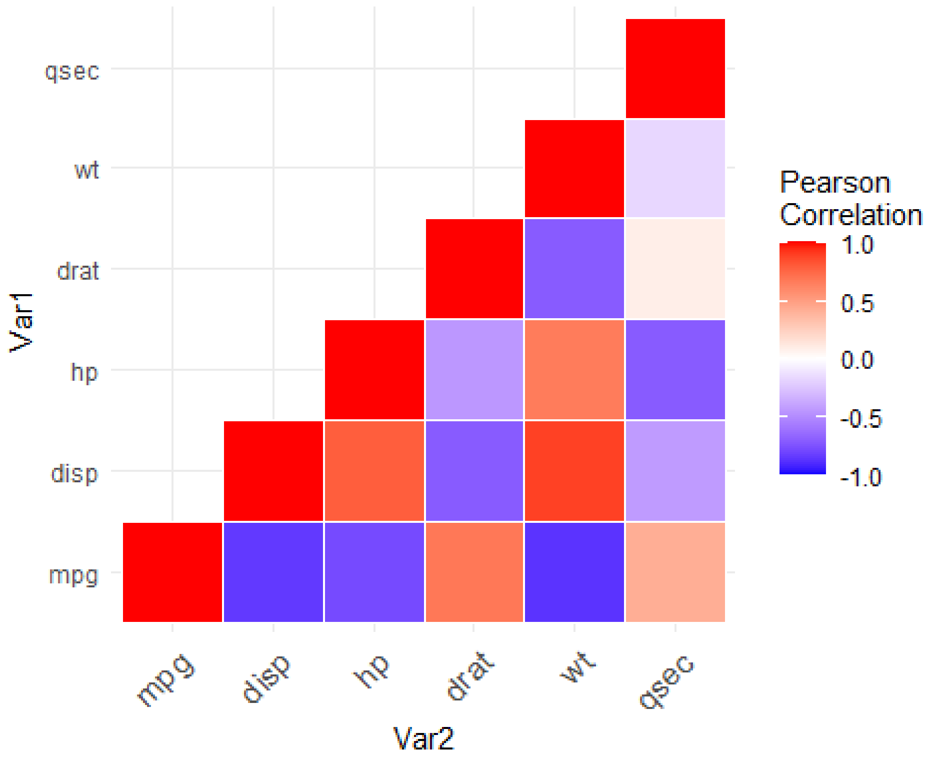
```
# Create the correlation heatmap with ggplot2
# The package reshape is required to melt the correlation matrix.
library(reshape2)
melted_cormat <- melt(cormat)
head(melted_cormat)
```

```
##   Var1 Var2 value
## 1  mpg  mpg  1.00
## 2 disp  mpg -0.85
## 3   hp  mpg -0.78
## 4 drat  mpg  0.68
## 5   wt  mpg -0.87
## 6 qsec  mpg  0.42
```

```
#The function geom_tile()[ggplot2 package] is used to visualize the
correlation matrix :
library(ggplot2)
ggplot(data = melted_cormat, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile()
```

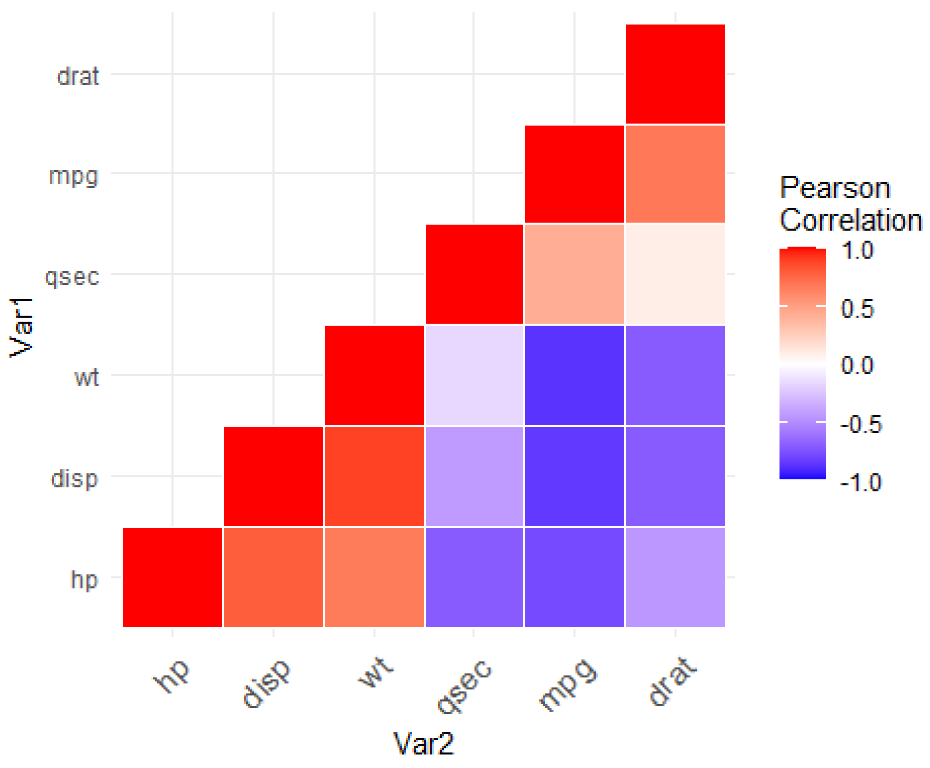


```
#Doesnot Look Great.. Let's Enhance the viz!

#Get the lower and upper triangles of the correlation matrix
## a correlation matrix has redundant information. We'll use the functions
below to set half of it to NA.

# Get lower triangle of the correlation matrix
get_lower_tri<-function(cormat){
  cormat[upper.tri(cormat)] <- NA
  return(cormat)
}
# Get upper triangle of the correlation matrix
get_upper_tri <- function(cormat){
  cormat[lower.tri(cormat)]<- NA
  return(cormat)
}

upper_tri <- get_upper_tri(cormat)
upper_tri

##      mpg  disp    hp  drat    wt  qsec
## mpg    1 -0.85 -0.78  0.68 -0.87  0.42
## disp  NA  1.00  0.79 -0.71  0.89 -0.43
## hp    NA    NA  1.00 -0.45  0.66 -0.71
## drat  NA    NA    NA  1.00 -0.71  0.09
## wt     NA    NA    NA    NA  1.00 -0.17
## qsec   NA    NA    NA    NA    NA  1.00

#  Finished correlation matrix heatmap
## Melt the correlation data and drop the rows with NA values
# Melt the correlation matrix
library(reshape2)
melted_cormat <- melt(upper_tri, na.rm = TRUE)
# Heatmap
library(ggplot2)
ggplot(data = melted_cormat, aes(Var2, Var1, fill = value))+
  geom_tile(color = "white")+
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name="Pearson\nCorrelation") +
  theme_minimal()+
  theme(axis.text.x = element_text(angle = 45, vjust = 1,
                                   size = 12, hjust = 1))+
  coord_fixed()
```

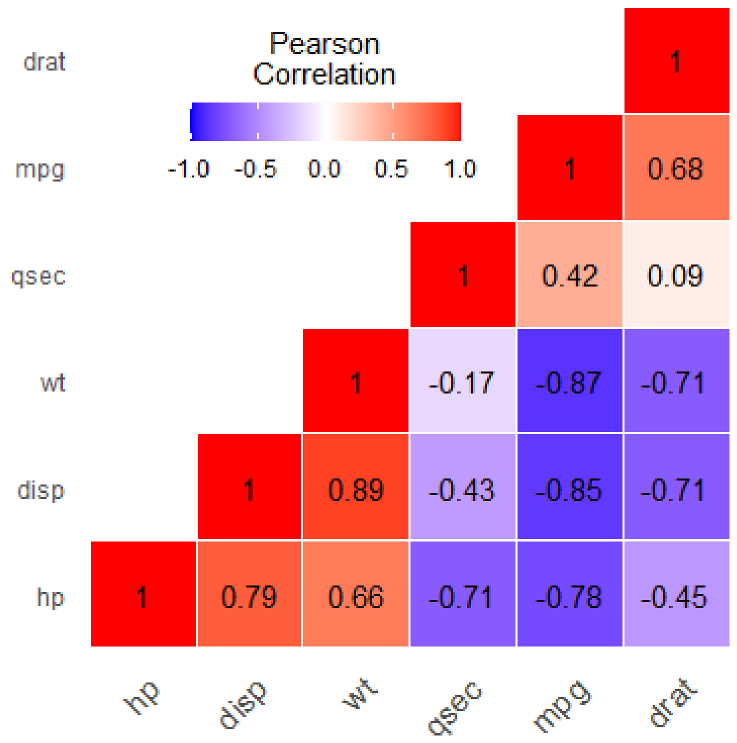


```
# negative correlations are in blue color and positive correlations in red.
# The function scale_fill_gradient2 is used with the argument limit = c(-1,1)
as correlation coefficients range from -1 to 1.
# coord_fixed() : this function ensures that one unit on the x-axis is the
same length as one unit on the y-axis.

# Reorder the correlation matrix
```

```r
# This section describes how to reorder the correlation matrix according to
the correlation coefficient.
# This is useful to identify the hidden pattern in the matrix.
# hclust for hierarchical clustering order is used in the example below.

reorder_cormat <- function(cormat){
  # Use correlation between variables as distance
  dd <- as.dist((1-cormat)/2)
  hc <- hclust(dd)
  cormat <-cormat[hc$order, hc$order]
}

# Reorder the correlation matrix
cormat <- reorder_cormat(cormat)
upper_tri <- get_upper_tri(cormat)
# Melt the correlation matrix
melted_cormat <- melt(upper_tri, na.rm = TRUE)
# Create a ggheatmap
ggheatmap <- ggplot(melted_cormat, aes(Var2, Var1, fill = value))+
  geom_tile(color = "white")+
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name="Pearson\nCorrelation") +
  theme_minimal()+ # minimal theme
  theme(axis.text.x = element_text(angle = 45, vjust = 1,
                                   size = 12, hjust = 1))+
  coord_fixed()
# Print the heatmap
print(ggheatmap)
```



```
#Add correlation coefficients on the heatmap

## Use geom_text() to add the correlation coefficients on the graph
## Use a blank theme (remove axis labels, panel grids and background, and
axis ticks)
## Use guides() to change the position of the legend title

ggheatmap +
  geom_text(aes(Var2, Var1, label = value), color = "black", size = 4) +
  theme(
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    panel.grid.major = element_blank(),
    panel.border = element_blank(),
    panel.background = element_blank(),
    axis.ticks = element_blank(),
    legend.justification = c(1, 0),
    legend.position = c(0.6, 0.7),
    legend.direction = "horizontal")+
  guides(fill = guide_colorbar(barwidth = 7, barheight = 1,
                               title.position = "top", title.hjust = 0.5))
```



HYPOTHESIS TESTING

```
# T-tests
# Dataset: Tips dependents on...
data(tips, package = "reshape2")
head(tips)

##   total_bill  tip    sex smoker day   time size
## 1      16.99 1.01 Female     No Sun Dinner    2
## 2      10.34 1.66   Male     No Sun Dinner    3
## 3      21.01 3.50   Male     No Sun Dinner    3
## 4      23.68 3.31   Male     No Sun Dinner    2
## 5      24.59 3.61 Female     No Sun Dinner    4
## 6      25.29 4.71   Male     No Sun Dinner    4

str(tips)

## 'data.frame':    244 obs. of  7 variables:
##  $ total_bill: num  17 10.3 21 23.7 24.6 ...
##  $ tip       : num  1.01 1.66 3.5 3.31 3.61 4.71 2 3.12 1.96 3.23 ...
##  $ sex       : Factor w/ 2 levels "Female","Male": 1 2 2 2 1 2 2 2 2 2 ...
##  $ smoker    : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ day       : Factor w/ 4 levels "Fri","Sat","Sun",..: 3 3 3 3 3 3 3 3 3
3 ...
##  $ time      : Factor w/ 2 levels "Dinner","Lunch": 1 1 1 1 1 1 1 1 1 1
...
##  $ size      : int  2 3 3 2 4 4 2 4 2 2 ...
```

```
write.csv(tips, "C:/Users/Pankaj/Documents/IMT-G/TERM 2/R PROGRAMMING/Sir
Code/5 DSA Basic Statistics - EDA/tips.csv", row.names = FALSE)

# Gender
unique(tips$sex)
```

```
## [1] Female Male
## Levels: Female Male
```

```
#Day of the week
unique(tips$day)
```

```
## [1] Sun  Sat  Thur Fri
## Levels: Fri Sat Sun Thur
```

```
#One Sample t-test - ONE GROUP [Two Tail. Ho:Mean = 2.5]
t.test(tips$tip, alternative = "two.sided", mu=2.5)
```

```
##
##  One Sample t-test
##
## data:  tips$tip
## t = 5.6253, df = 243, p-value = 5.08e-08
## alternative hypothesis: true mean is not equal to 2.5
## 95 percent confidence interval:
##  2.823799 3.172758
## sample estimates:
## mean of x
##  2.998279
```

```
#One Sample t-test - Upper Tail. Ho:Mean LE 2.5
t.test(tips$tip, alternative = "greater", mu=2.5)
```

```
##
##  One Sample t-test
##
## data:  tips$tip
## t = 5.6253, df = 243, p-value = 2.54e-08
## alternative hypothesis: true mean is greater than 2.5
## 95 percent confidence interval:
##  2.852023      Inf
## sample estimates:
## mean of x
##  2.998279
```

```
# Two Sample T-test - TWO GROUP
t.test(tip ~ sex, data = tips, var.equal = TRUE)

##
##  Two Sample t-test
##
## data:  tip by sex
## t = -1.3879, df = 242, p-value = 0.1665
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.6197558  0.1074167
## sample estimates:
## mean in group Female   mean in group Male
##             2.833448             3.089618

#Paired Two-Sample T-Test
# Dataset: Heights of Father and Son (Package:UsingR)
install.packages("UsingR", repo="https://cran.us.r-project.org")

## Installing package into 'C:/Users/91996/Documents/R/win-library/4.0'
## (as 'lib' is unspecified)

## package 'UsingR' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\91996\AppData\Local\Temp\RtmpcvlPaU\downloaded_packages

require(UsingR)

## Loading required package: UsingR

## Loading required package: MASS

## Loading required package: HistData

## Loading required package: Hmisc

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, units

##
## Attaching package: 'UsingR'
## The following object is masked from 'package:survival':
##
##     cancer
```

```
head(father.son)
```

```
##    fheight  sheight
## 1 65.04851 59.77827
## 2 63.25094 63.21404
## 3 64.95532 63.34242
## 4 65.75250 62.79238
## 5 61.13723 64.28113
## 6 63.02254 64.24221
```

```
write.csv(father.son, "C:/Users/Pankaj/Documents/IMT-G/TERM 2/R
PROGRAMMING/Sir Code/5 DSA Basic Statistics - EDA/tips.csv", row.names =
FALSE)

#ANOVA  - Comparing Multiple Groups
# Tip by the Day of the Week
str(tips)
```

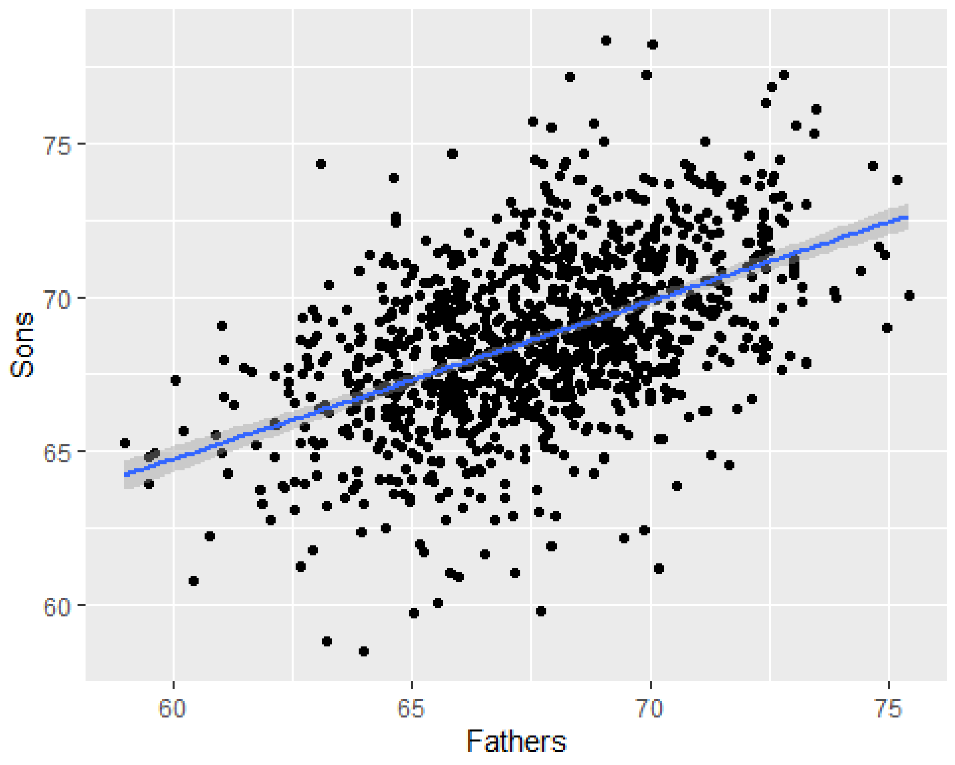
```
## 'data.frame':    244 obs. of  7 variables:
##  $ total_bill: num  17 10.3 21 23.7 24.6 ...
##  $ tip       : num  1.01 1.66 3.5 3.31 3.61 4.71 2 3.12 1.96 3.23 ...
##  $ sex       : Factor w/ 2 levels "Female","Male": 1 2 2 2 1 2 2 2 2 2 ...
##  $ smoker    : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ day       : Factor w/ 4 levels "Fri","Sat","Sun",..: 3 3 3 3 3 3 3 3 3
3 ...
##  $ time      : Factor w/ 2 levels "Dinner","Lunch": 1 1 1 1 1 1 1 1 1 1
...
##  $ size      : int  2 3 3 2 4 4 2 4 2 2 ...
```

```
tipAnova = aov(tip ~ day, tips)
summary(tipAnova)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## day           3    9.5   3.175   1.672  0.174
## Residuals   240  455.7   1.899
```

LINEAR REGRESSION

```r
# Simple Linear Regression (SLR)
# Dataset: father.son.
# Using fathers' heights to predit sons' heights using SLR.
# Fathers height as predictor(Indep - X) and
# Son's height as the response /Target(Dep - Y)
require(UsingR)
require(ggplot2)
head(father.son)

##    fheight  sheight
## 1 65.04851 59.77827
## 2 63.25094 63.21404
## 3 64.95532 63.34242
## 4 65.75250 62.79238
## 5 61.13723 64.28113
## 6 63.02254 64.24221

ggplot(father.son, aes(x=fheight, y=sheight))+geom_point()+
  geom_smooth(method="lm")+labs(x="Fathers", y="Sons")

## `geom_smooth()` using formula 'y ~ x'
```



```r
heightsLM = lm(sheight ~ fheight, data = father.son)
heightsLM

##
## Call:
## lm(formula = sheight ~ fheight, data = father.son)
##
## Coefficients:
## (Intercept)      fheight
##     33.8866       0.5141

summary(heightsLM)

##
## Call:
## lm(formula = sheight ~ fheight, data = father.son)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.8772 -1.5144 -0.0079  1.6285  8.9685
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 33.88660    1.83235   18.49   <2e-16 ***
## fheight      0.51409    0.02705   19.01   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.437 on 1076 degrees of freedom
## Multiple R-squared:  0.2513, Adjusted R-squared:  0.2506
## F-statistic: 361.2 on 1 and 1076 DF,  p-value: < 2.2e-16
```

THE END