INDEX

Expt. No.	Date D	Date M	Date Y	Experiments - Titles	Page No.	Remarks	Signature
1	14	1	18	Basic command, array, list & frames.			
2	18	1	18	Create a matrix & perform operations addition, inverse, transpose & multiplication.			
3	18	1	18	Execute statistical freq mean, median, mode, quartiles, range, interquartile range, histogram.			
4.	25	1	18	Import data from Excel/.csv file & perform above f^n's.			
9.	15	2	18	Binomial & norm. distribution			
10.	22	2	18	Linear regression using.			
11	22	2	18	Compute least squares.			
12	1	3	18	Compute linear least square regression			
8	8	3	18	Import .csv file & perform squared test.			
5.	8	3	18	SD, variance, co-variance.			
6.	8	3	18	import .csv & draw skewness.			

Using R execute the basic commands, array list & frames.

* R-command prompt.

```
> text ← "Welcome R!"
> print (text)
[1] "Welcome R!"
```

* Datatypes

Datatype	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 o/p: [1] "numeric"
Integer	2L, 34L, 0L	v ← 2L Print (class(v)) It produces the following o/p [1] "integer"

Complex	3+2i	v ← 2+5i print(class(v)) it produces the following o/p [1] "complex
Character	'a', "good" "True"	v ← "True" print(class(v)) it produces the following o/p [1] character
Raw	"Hello" is stored as 48. 65. 6c 6c 6F	v ← charToRaw("Hello") print(class(v)) it produces the following o/p [1] "raw"

* Vectors.

```
> #Create A vectore
> car_model ← c("Skoda Rapid", "Nissan GTR", "Toyota", "Hyundai")
> print(car model)
```

O/p:

```
[1] "Skoda Rapid", "Nissan GTR" "Toyota"
[4] "Hyundai
```

* Lists

```
> # Create a list
> mylist ← list(x = c(1,5,7), c(4.4, 2.5, b), max)
> print(mylist)
```

Page No. _____

Date : | | |

O/p:

[1] 1 5 7

[[2]]
[1] 4.4 25.0 6.0

[[3]]
function (..., na.rm = FALSE). primitive ("max")

Matrices

```
> #Create a Matrix
> M= Matrix (c ('A','B','C','D','E','F','G','H','I'), nrow=3
          ncol = 3, byrow = TRUE)
```

O/P :-

	[,1]	[,2]	[,3]
[1,]	"A"	"B"	"C"
[2,]	"D"	"E"	"F"
[3,]	"G"	"H"	"I"

Array

```
> #Create an Array
> A <- array (c ('ON', 'OFF'), dim = c(3,3,2))
> print(A)
```

O/P:

, , 1

	[,1]	[,2]	[,3]
[1,]	"ON"	"OFF"	"ON"
[2,]	"OFF"	"ON"	"OFF"
[3,]	"ON"	"OFF"	"ON"

, , 2

	[,1]	[,2]	[,3]
[1,]	"OFF"	"ON"	"OFF"
[2,]	"ON"	"OFF"	"ON"
[3,]	"OFF"	"ON"	"OFF"

* Factors

```
> #Create a vector
> permission <- c('Read', 'Write', 'Execute')
>
> # Create a Factor
> F_ permission <- factor (permission)
>
>#print the factor
> print (F_permission)
```

O/P:

```
> print (F_permission)
[1] Read  Write  Execute
```

Date : | | |

Levels : Execute Read Write

```
>
> print (nlevels (f_permission))
[1] 3
```

* Data frames

```
> #Create dataframe
> Stud <- data.frame (
      + sno = c (101, 102, 103, 104, 105),
      + name = c("Madhurya", "Reyansh", "Tarush",
                 "Saloni", "Idhant"),
      + age=c (22, 23, 26, 23, 24));
>> print (stud)
```

O/p :-

	sno.	name	age
1	101	Madhurya	22
2	102	Reyansh	23
3	103	Tarush	26
4	104	Saloni	23
5	105	Idhant	24

* R-Operators

1) Arithmetic operators

Operator	Description	Example
+	Add 2 Vectors	> a ← c(1, 4, 6) > b ← c(13, 44, 57 It produces the following result: > # print(a+b) [1] 14 48 63
*	Multiply 2 vectors	> a ← c(1, 4, 6) > b ← c(13, 44, 57) It produces the following result > print(a*b) [1] 13 176 342
/	Division of 2 vectors	> a ← c(1, 4, 6) > b ← c(13, 44, 57) It produces the foll. result. > print(a/b) [1] 0.07 0.09 0.105
-	Subtract one vector from another	> a ← c(1, 4, 6) > b ← c(13, 44, 57) It produces the following result. > print(b-a) [1] 12 40 5
%%	Give the remainder of the first vector with the 2nd.	> a ← c(1, 4, 6) > b ← c(13, 44, 57) It produces the following result > print(a%%b) [1] 1 4 6

Page No. _____

Date : | | |

%/%	The result of division of first vector with second(quotient)	> a <- c(1, 4, 6) > b <- c(13, 44, 57) It produces the following result > print(a%/%b) [1] 0 0 0
^	The first vector raised to exponent of 2nd vector	> a ← c(1, 4, 6) > b ← c(13, 44, 57) It produces the following result > print(a^b) [1] 1.0000e+00 3.0485e+26 2.2627e+44
Relational Operators		
Operator <	Description checks if each element of the first vector is less than the corresponding element of the second vector	Example > a ← c(12, 34, 44) > b ← c(65, 4, 16) It produces the following result. > print(a<b) [1] True FALSE FALSE

>	Checks if each element of the first vector is greater than corresponding element of second vector	> a <- c(12, 34, 44) > b <- c(65, 4, 16) It produces the following result > print(a > b) [1] FALSE TRUE TRUE
==	Checks if each element of the first vector is equal to the corresponding element of the 2nd vector	> a <- c(12, 34, 44) > b <- c(65, 4, 16) It produces the following result > print(a == b) [1] FALSE FALSE FALSE
<=	Checks if each element of the first vector is less than or equal to the corresponding element of the 2nd vector	> a <- c(12, 34, 44) > b <- c(65, 4, 16) It produces the following result. > print(a <= b) [1] TRUE FALSE FALSE
>=	Checks if each element of the first vector is greater than or equal to the corresponding element of the 2nd vector	> a <- c(12, 34, 344) > b <- c(65, 4, 16) It produces the following result > print(a >= b) [1] FALSE TRUE TRUE
!=	Checks if each element of the first vector is unequal to the corresponding element of the 2nd vector	> a <- c(12, 34, 44) > b <- c(65, 4, 16) It produces the following result > print(a != b) [1] TRUE TRUE TRUE

Date : | | |

Operator	Description	Example
&	It is called Element-wise logical AND operator. It combines each element of the first vector with the corresponding element of 2nd vector & gives an output ~~true~~ TRUE if both the elements are TRUE	>a ← c (12, 34, TRUE, 2+3i) >b ← c (65, 4, FALSE, 2+3i) It produces the following result:- >print (a & b) [1] TRUE TRUE FALSE TRUE
|	It is called Element wise logical OR operator. It combines each element of the 1st vector to the corresponding element of the 2nd vector. & gives an O/p 'true' if one element is 'true'.	>a ← c (12, 34, TRUE, 2+3i) >b ← c (65, 4, FALSE, 2+3i) print (a | b) [1] TRUE TRUE TRUE TRUE
!	It is called NOT operator. Takes each element of the vector & gives the opposite logical value	>a <- c (12, 34, TRUE, 2+3i) >b <- c (65, 4, FALSE, 2+3i) It produces the following result >print (!b) [1] FALSE FALSE TRUE FALSE

Operator	Description	Example
:	colon operator. It creates the series of numbers in sequences for a vector	> a ← 3:6 > print(a) [1] 3 4 5 6
%in%	This operator is used to identify if an element belongs to a vector	> a ← 5 > b ← 10 > c ← 1:5 > print (a %in% c) [1] TRUE > print (b %in% c) [1] FALSE

Page No. _____

Date : | | |

Create a Matrix using R and Perform the operations addition, inverse, transpose & multiplication operations

Matrices

Syntax:

matrix(data, nrow, ncol, byrow, dimnames)

Example:-

```
> # Elements are arranged sequentially by rows.
> x <- matrix(c(1:10), nrow = 2, byrow = TRUE)
> print(x)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
>
> # Elements are arranged sequentially by column
> y <- matrix(c(1:10), nrow = 2, byrow = FALSE)
> print(v)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
>
> #Define the column & rows names
> rownames = c("row1", "row2")
> colnames = c("col1", "col2", "col3", "col4", "col5")
```

```
>
> P ← Matrix ( c(1:10)^3, nrow = 2, byrow = TRUE,
        dimnames = list(rownames, colnames))
> print(p)
      col1  col2  col3  col4  col5
row1     1     2     3     4     5
row2     6     7     8     9    10
```

Accessing Elements of a matrix

```
>#create matrix P
> p ← matrix(c(10:21), nrow = 4, byrow = TRUE)
> print(p)
     [,1] [,2] [,3]
[1,]   10   11   12
[2,]   13   14   15
[3,]   16   17   18
[4,]   19   20   21
>
># Access the element at 3rd row & 2nd column
> print (p[3,2])
[1] 17
>
>#Access the element at 4th row
> print (p(4,))
[1] 17 20 21
>
#Access the element at 4th row & 2nd col
> print (p(4,2))
[1] 20
```

Page No. _____

Date : / / /

* Matrix Addition & Subtraction

```
>#Create Matrix P&Q
>P <- Matrix (c(10:21), nrow = 4, byrow = TRUE)
>Q <- matrix(c(1:12), nrow=4, byrow =TRUE)
>
>#Addition
>add = P+Q
>
># Subtraction
>sub = P-Q
```

$ ># Print Both Matrix

```
>Print(p)
```

	[,1]	[,2]	[,3]
[1,]	10	11	12
[2,]	13	14	15
[3,]	16	17	18
[4,]	19	20	21

```
>print(Q)
```

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

```
>#Addition Matrix
> print(add)
```

```
     [,1] [,2] [,3]
[1,]   11   13   15
[2,]   17   19   21
[3,]   23   25   27
[4,]   29   31   33
```

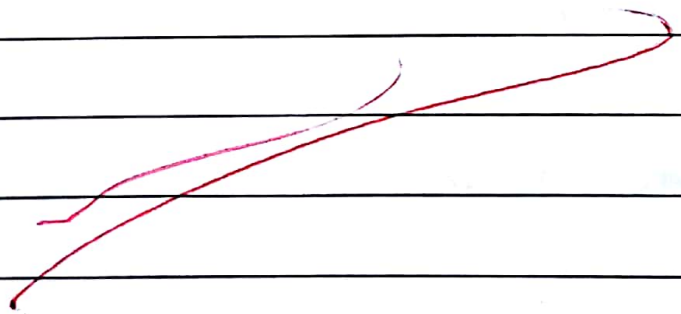
```
> # Subtraction Matrix
> print(sub)
     [,1] [,2] [,3]
[1,]    9    9    9
[2,]    9    9    9
[3,]    9    9    9
[4,]    9    9    9
```

* Multiplication & Division

```
>## print both Matrix
> print(p)
     [,1] [,2] [,3]
[1,]   10   11   12
[2,]   13   14   15
[3,]   16   17   18
[4,]   19   20   21
> print(Q)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[4,]   10   11   12
```

Date : / / /

```
> #addition Matrix
> print(multi)
     [,1] [,2] [,3]
[1,]   10   22   36
[2,]   52   78   90
[3,]  112  136  162
[4,]  190  220  252
>
> #Subtraction Matrix
> print(div)
          [,1]     [,2]    [,3]
[1,] 10.00000 5.50000 4.00
[2,]  3.25000 2.80000 2.50
[3,]  2.285714 2.12500 2.00
[4,] 1.90000 1.818182 1.75
```



Using R Execute the statistical functions:- mean, median mode, quartiles, range, inner quartile range histogram.

* Mean

Syntax :
mean(x, trim=0, na.rm= FALSE, ...)

Example
```
>#create a vector
>v<- c(10, 3, 25, 6, 22, -33)
>#find Mean
>result <- mean(v)
>print (result)
[1] 5.5
```

Apply TRIM option

```
>#create a vector
>v<- c (10,3,25,6, 22,-33)
>#find mean
>result <- mean (v)  trim = 0.2)
>print(r)
[1] 1.6
```

Applying NA option
```
>#create a vector
>v <- c(10,3,25,6,22,-33,-44,-44, NA)
>#find Mean
>r<- mean(v)
```

Page No. _____

Date : | | |

```
> print (a)
[1] NA
> a1 <- mean (v, na.rm=TRUE)
> print (a1)
[1] -1.571429
```

Median

Syntax :-

```
median (x, na.rm = FALSE)
```

Example :-

```
># create a vector
>v <- c (10, 3, 25, 6, 22, -33, -44)
># find Median
>a <- median (v)
> print (a)
[1] 6
```

Mode

```
> #Create the function
> getmode <- function (v)
+{
+ uniqv <- unique (v)
 +uniqv [which.max (tabulate (match (v, uniqv)))]
 }
># Create the no. vector
>v <- c (2,3,3,4,4,1,2,3)
```

```
#Calculate the mode using function
> r ← getmode(v)
> print(r)
[1] 3
> char v ← c("R","C","C++","R","C++")
> #Calculate the mode using function
> r ← getmode(char v)
> print(r)
[1] "R"
```

* Histogram

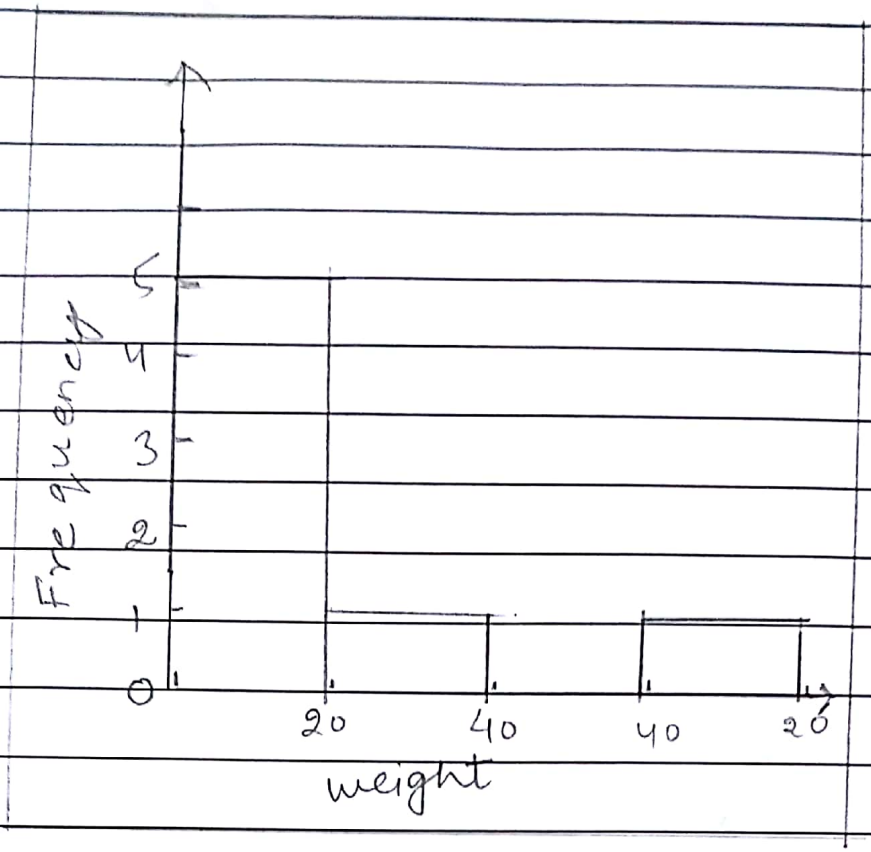
Syntax :-
hist(v, main, xlab, xlib, ylim, breaks, wl, boorder)

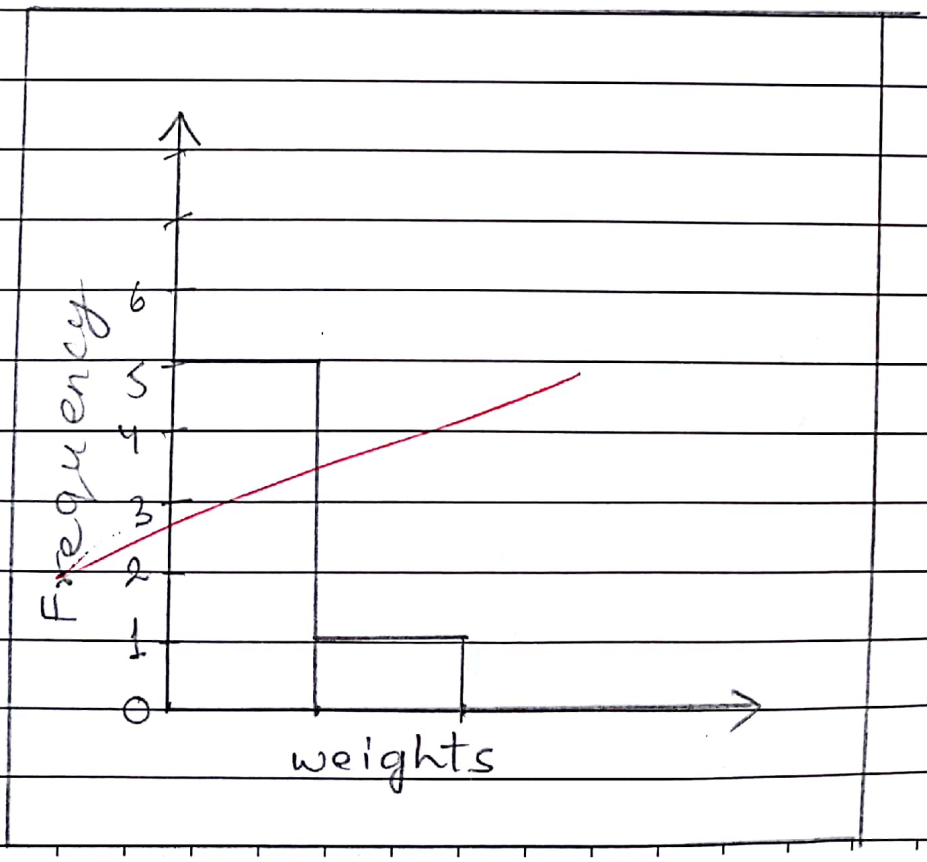
```
>#Create data for Graph
>v ← c(3,4,5,23,4,5,66)
>png(file = "Histogram.jpg")
>hist(v, xlab = "Weight", col = "red", border = "yellow")
>dev.off()
null device
1
>hist(v, xlab = "Weight", col = "red", border = "yellow")
>1

>#Create Histogram.
>hist(v, xlab = "Weight", col = "red", border = "blue", xlim =
  c(0,60), ylim = (0,7), breaks = 3)
```

Page No. _____

Date : | | |





Using R import the data from Excel / CSV file & perform the above func.

* Input as .csv file

input.csv

R.no	Age	name	dept
101	26	Saloni	IT
102	25	Idhant	HR
103	24	Rayansh	Finance
104	24	Tashish	IT
105	25	Madhuya	HR

* Reading a CSV file

```
> #first set dictionary of csv file location
> setwd ("C:/a")
> d ← read.csv ("input.csv")
> print(d)
```

O/p:

	Rno.	Age	name	Dept
1	101	26	Saloni	It
2	102	25	Idhant	HR
3	103	24	Rayansh	Finance
4	104	24	Parul	IBT
5	105	25	Madhuya	HR

Page No. _____

Date : | | |

* Analysing the CSV file

```
>d <- read.csv("input.csv")
>print(is.data.frame(d))
[1] TRUE
>print(ncol(d))
[1] 4
>print(nrow(d))
[1] 5
```

* Get the maximum age

```
># Display max age from data frame
>data <- read.csv("input.csv")
>max age <- max(data $ age)
>print(max - age)
[1] 26
```

* Means, Median, Mode

values.csv

A	B
12	45
3	76
18	22
9	11

```
> mean (d$A)
[1] 10.5
> Median (d$A)
[1] 10.5
> getmode <- function(v)
+ {
+ uniqv <- unique(v)
+ unique [which. max (tabulate(match (v, uniqv )))]
+ y
> v <- c(2,3,3,4,4,1;2,3)
> r <- getmode(v)
> print(r)
[1] 3
> charv <- c("R", "C.", "C++", "R", "C++")
> r <- getmode(charv)
> print(r)
[1] "R"
```

Page No. _____

Date : / / /

PRACTICAL 9

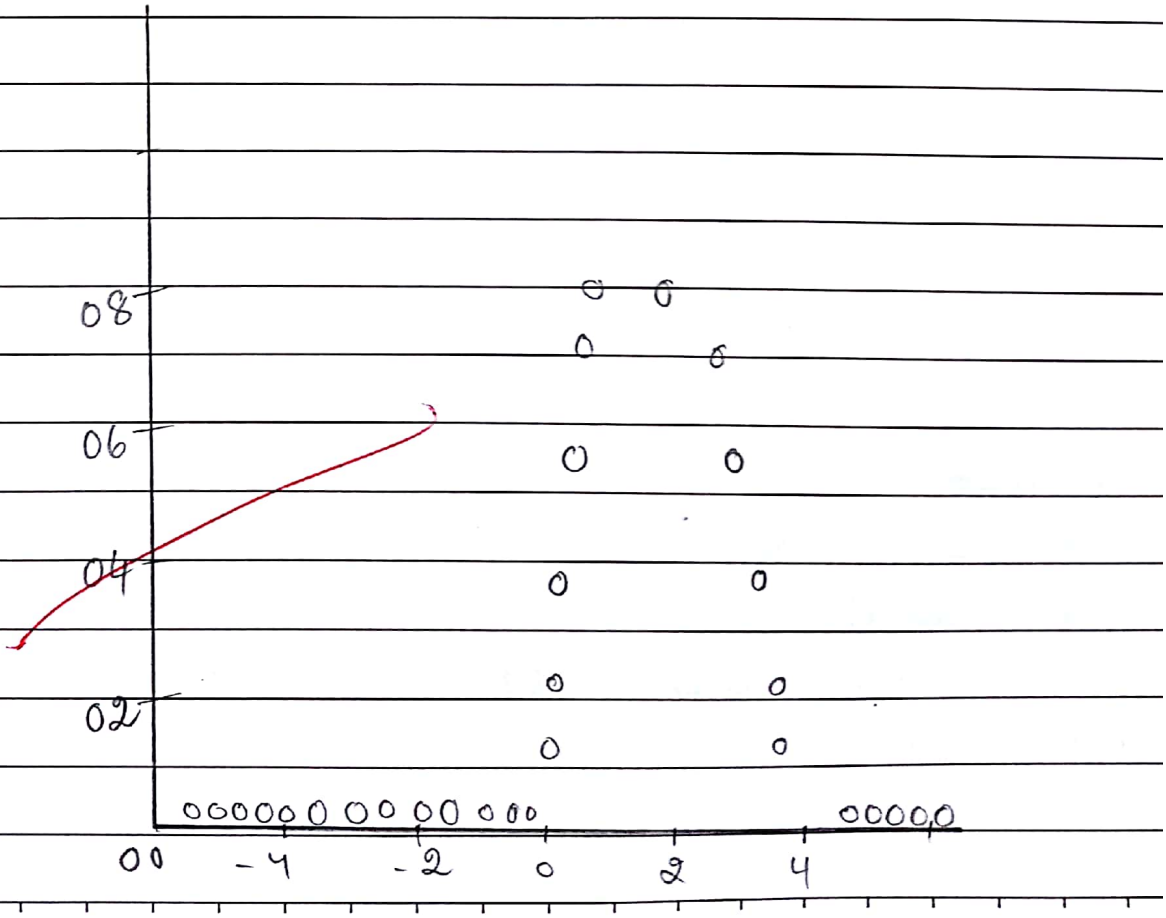
Using R perform binomial & normal distribution on the data.

* Normal distribution

1) dnorm()

```
> # create a sequence of -5 to 5 increment by 0.2
> a ← seq(-5, 5, by = 0.2)
> # choose mean as 1.5 & std deviation .
> b ← dnorm(a, mean = 1.5, sd = 0.5)
> png(file = "Dnoorm.png")
> plot(a, b)
> dev.off()
  null device
> plot(a, b)
```

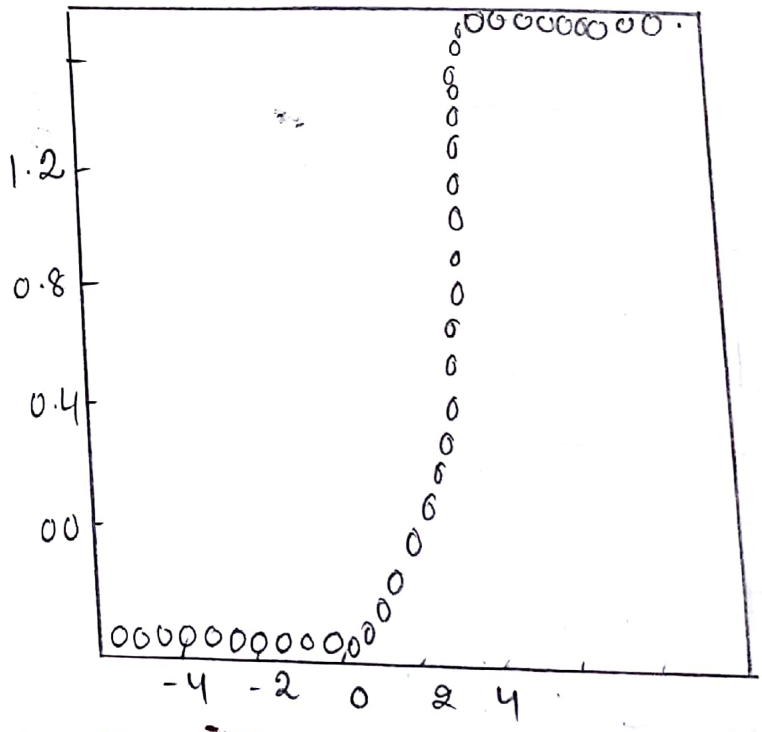
Result :-



2) pnorm( )

```
> #create a sequence of -5 to 5 increment by 0.2
> a <- seq(-5, 5, by = 0.2)
> png( file = "pnorm.png")
> b <- pnorm (a, mean = 1.5, sd = 0.5)
> plot(a, b)
> dev.off ()
  null device

> plot (a,b)
```



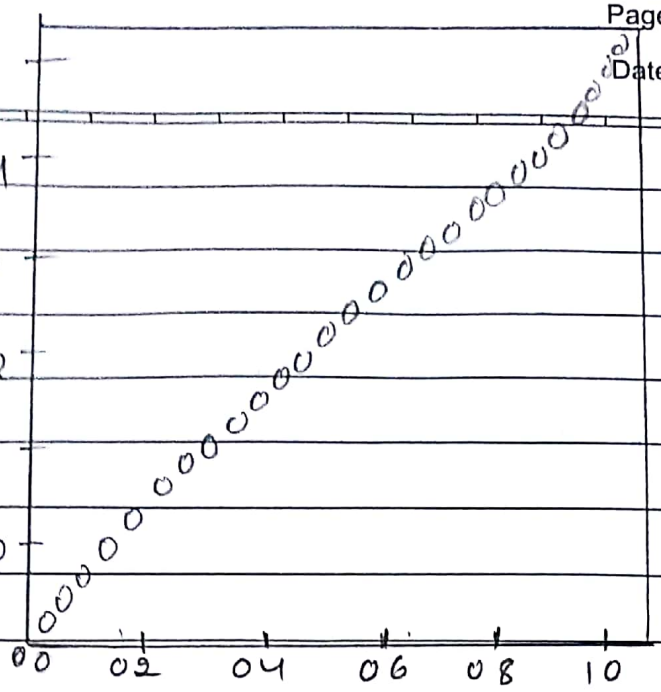
3) qnorm

```
> #create a sequnie
> a <- seq(0, 1, by=0.01)
> b <- qnorm (a, mean=2, sd = 1)
> plot (a,b)
```

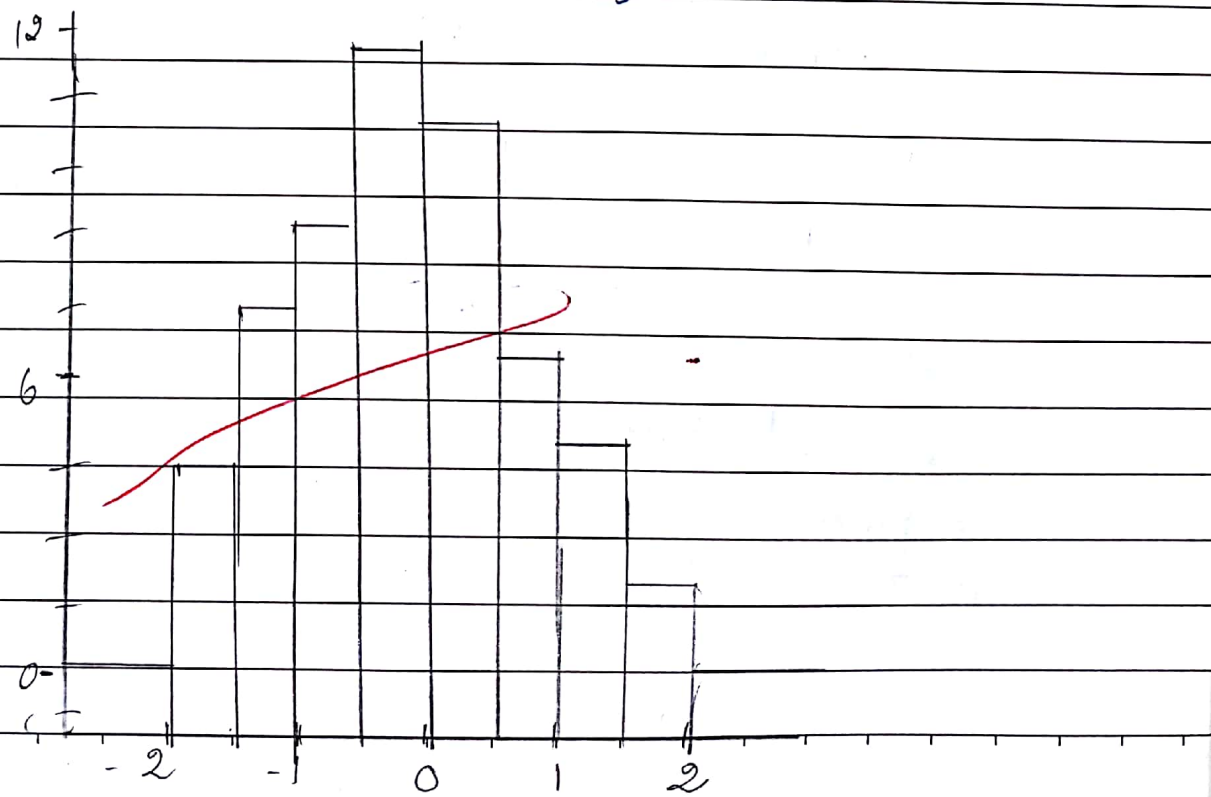
Page No. _____

Date : | | |

4) Rnorm( )



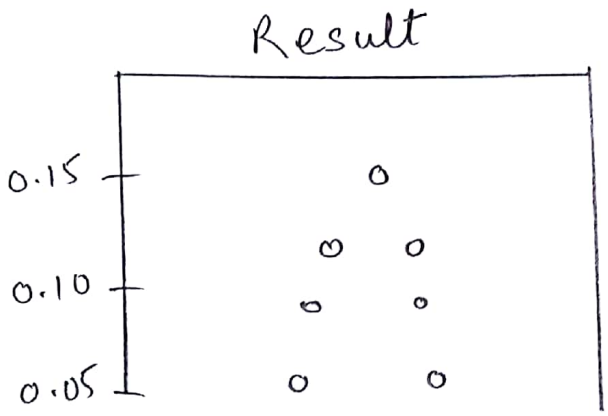

```
>#create a sample of 50 no.s
>a<- Rnorm(50)
>png(file = "Rnorm.png")
>hist(a, main = "Normal Distribution")
>dev.off()
 null device
           1
>hist(a, main = "Normal distribution")
```



* Binomial Distribution

1) dbinom ( )

```
># create a sample of 50 numbers
> a ← seq (1, 30, by =2)
```



2) Pbinorm

```
> # probability
> a ← pbinorm (12,34, 0.4)
> print(a)
[ c ] 0.354171.
```

3) qbinom c)

```
> a ← qbinorm (0.12,34,1/4)
> print(a)
[c] 6
```

Page No. _____

Date : \ \ \

4) rbinom()

```
> # find 5 random values from rbinom(5,30,0.4)
> a <- rbinom(5,30,0.4)
> print(a)
[1] 10 13 8 9 12.
```

Practical 10

perform the linear Regrenion using R.

- Linear Regression

  $y = ax + 6$

- lm() function

  lm (formula, data)

create relation model and get the coffice.

```
> # create date of dyf largs
>v ← c(23,45, 66, 50)
> u ← c (2, 3, 56)
> r ← lm (v ~ u)
> print (r)
```

Call:

lm (formula = v ~ v)

coefficients :'

(intercept)

12.7 u 8.7

get the summery of relationship

```
> # create data of diff length
> v ← c (23, 45, 66, 50)
> u ← (2, 3, 5, 6)
> r ← lon (v ~ u)
> print (summary) (r) )
```

call;

Page No. _____

Date : | | |

lm (formula = V ~ u)

Residuals

1	2	3	4
7.1	6.2	9.8	-8.9

Coefficients:

	Estimate	Stdmar	tvalue	Pr(>|t|)
(Intercept)	12.700	15.630	0.813	0.502
u	8.700	3.634	2.394	0.139

Residual standard error 11.49 on 2 DF
Multiple R-squared: 0.7413
Adjusted R-squared: 0.619
F-statistic: 5.732 on 1 & 2 DF
P-value : 0.139

Predict @:
Syntax - (predict object newdata)
predict the value from series

```
> # create data of diff. length.
> v <- c (23, 45, 66, 58)
> u <- c (2, 3, 5, 6)
> r <- lm (v ~ u)
> # find relational value for 4 from
> s <- data.frame (u = 4)
> p <- lm (v ~ u)
> result <- predict (r, s)
> print (result
```

47.5

Visualize the Regression Graphically:

```
> # create predictor and response values.
>m <- c(1,2,3.4,5,6)
>t <- c(25,22, 30 34, 45,26)
> # Label to chart
> png(file = "linear, png")
> plot(t,m,co = "red" main = "maths, temperature
+ abline(lm(m~t))
+ lwx = 1.6,
+ pch = 19
+ x lab = "month"
+ y lab = 'temperature")
```

Page No. _____

Date : | | |

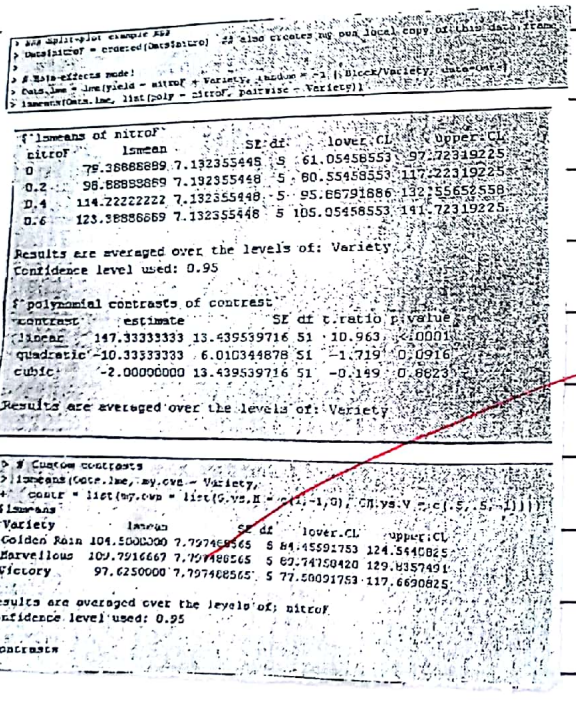
PRACTICAL 11

Computer the least square means using R.

. Least square mean

```
> instal packages ("Ts mean")
> setwd ("c. /r folder")
> library ("ls means)
```

```
Loading required packages ("Is mean")
> Instal packages ("Is mean)"
> Setwd ("c l r folder")
> library ("Is means)
    Loading required package estimability.
```



```
> ### split-plot example ###
> Oats$nitrof = ordered(Oats$nitro)   ## also creates my own local copy of this data frame
>
> # Main-effects model
> Oats.lme = lme(yield ~ nitrof + Variety, random = ~1 | Block/Variety, data=Oats)
> lsmeans(Oats.lme, list(poly ~ nitrof, pairwise ~ Variety))
$`lsmeans of nitrof`
 nitrof     lsmean       SE df lower.CL upper.CL
 0      79.38888889 7.192355448  5 61.05458553 97.72319225
 0.2    96.88888889 7.192355448  5 78.55458553 115.22319225
 0.4   114.22222222 7.192355448  5 95.88791886 132.55652558
 0.6   123.38888889 7.192355448  5 105.05458553 141.72319225

Results are averaged over the levels of: Variety
Confidence level used: 0.95

$`polynomial contrasts of contrast`
 contrast      estimate       SE df t.ratio p.value
 linear      147.33333333 13.439539716 51  10.963  <.0001
 quadratic   -10.33333333  6.010344878 51  -1.719  0.0916
 cubic        -2.00000000 13.439539716 51  -0.149  0.8823

Results are averaged over the levels of: Variety
```

```
> # Custom contrasts
> lsmeans(Oats.lme, ~my.cvs ~ Variety,
+   contr = list(my.cvs = list(G.vs.H = c(1,-1,0), GH.vs.V = c(.5, .5, -1))))
$lsmeans
 Variety      lsmean       SE df lower.CL upper.CL
 Golden Rain 104.5000000 7.797468565  5 84.45591753 124.5440825
 Marvellous  109.7916667 7.797468565  5 89.74758420 129.8357491
 Victory      97.6250000 7.797468565  5 77.58091753 117.6690825

Results are averaged over the levels of: nitrof
Confidence level used: 0.95

$contrasts
```

```
> # Model with interaction, follow-ups at levels of other factors
> Oats$int.lme = update(Oats.lme, . ~ . + nitrof:Variety)
> lsmeans(Oats$int.lme, list(revpairwise ~ Variety | nitrof, poly ~ nitrof | Variety))
$`lsmeans of Variety | nitrof`
nitrof = 0:
 Variety      lsmean       SE df lower.CL upper.CL
 Golden Rain 80.00000000 9.106954224  5 56.58908889 103.41017111
 Marvellous  86.66666667 9.106954224  5 63.25575555 110.07683777
 Victory     71.50000000 9.106954224  5 48.08908889 94.91017111

nitrof = 0.2:
 Variety      lsmean       SE df lower.CL upper.CL
 Golden Rain 98.50000000 9.106954224  5 75.00908889 121.91017111
 Marvellous 105.50000000 9.106954224  5 82.00908889 128.91017111
 Victory     89.66666667 9.106954224  5 66.25575556 113.07683777

nitrof = 0.4:
 Variety      lsmean       SE df lower.CL upper.CL
 Golden Rain 114.66666667 9.106954224  5 91.25575556 138.07683777
 Marvellous  117.16666667 9.106954224  5 93.75575555 140.57683777
 Victory     110.83333333 9.106954224  5 87.42216223 134.24450444

nitrof = 0.6:
```

```
> # The following will produce the above results for Victory only
> lsmeans(Oats$int.lme, poly ~ nitrof, for reduce = function(nof, lev) coef(lm(1, 1))
NOTE: Results may be misleading due to involvement in interactions
$lsmeans
 nitrof      lsmean       SE df lower.CL upper.CL
 0        71.50000000 9.106954224  5 48.08908889 94.91017111
 0.2      89.66666667 9.106954224  5 66.25575556 113.07683777
 0.4     110.83333333 9.106954224  5 87.42216223 134.24450444
 0.6     118.50000000 9.106954224  5 95.08908889 141.91017111

Results are averaged over the levels of: Variety
Confidence level used: 0.95

$contrasts
 contrast      estimate       SE df t.ratio p.value
 linear      162.1666667 24.29564907 45   6.675  <.0001
 quadratic   -10.5000000 10.85534657 45  -0.966  0.3390
 cubic       -16.5000000 24.29564907 45  -0.679  0.5035

Results are averaged over the levels of: Variety
```

```
l a fake covariate that has some predictive ability
eed(271828, "default", "default")  # for reproducibility of resul
fakex = sqrt(Oats$yield) + rnorm(nrow(Oats), 5, .25)
ov.lme = update(Oats.lme, . ~ . + poly(fakex,1))
```

Page No. _____

Date : / / /

PRACTICAL - 12

Computer the linear square Regression

> # five pairs consists of a year & the means interest rate :

> years ← c (2000, 2001, 2002, 2003, 2004)

> rate ← c (9.34, 8.50, 7.62, 6.93, 6.60)

> # find the correlation between the year & mean interest rates.

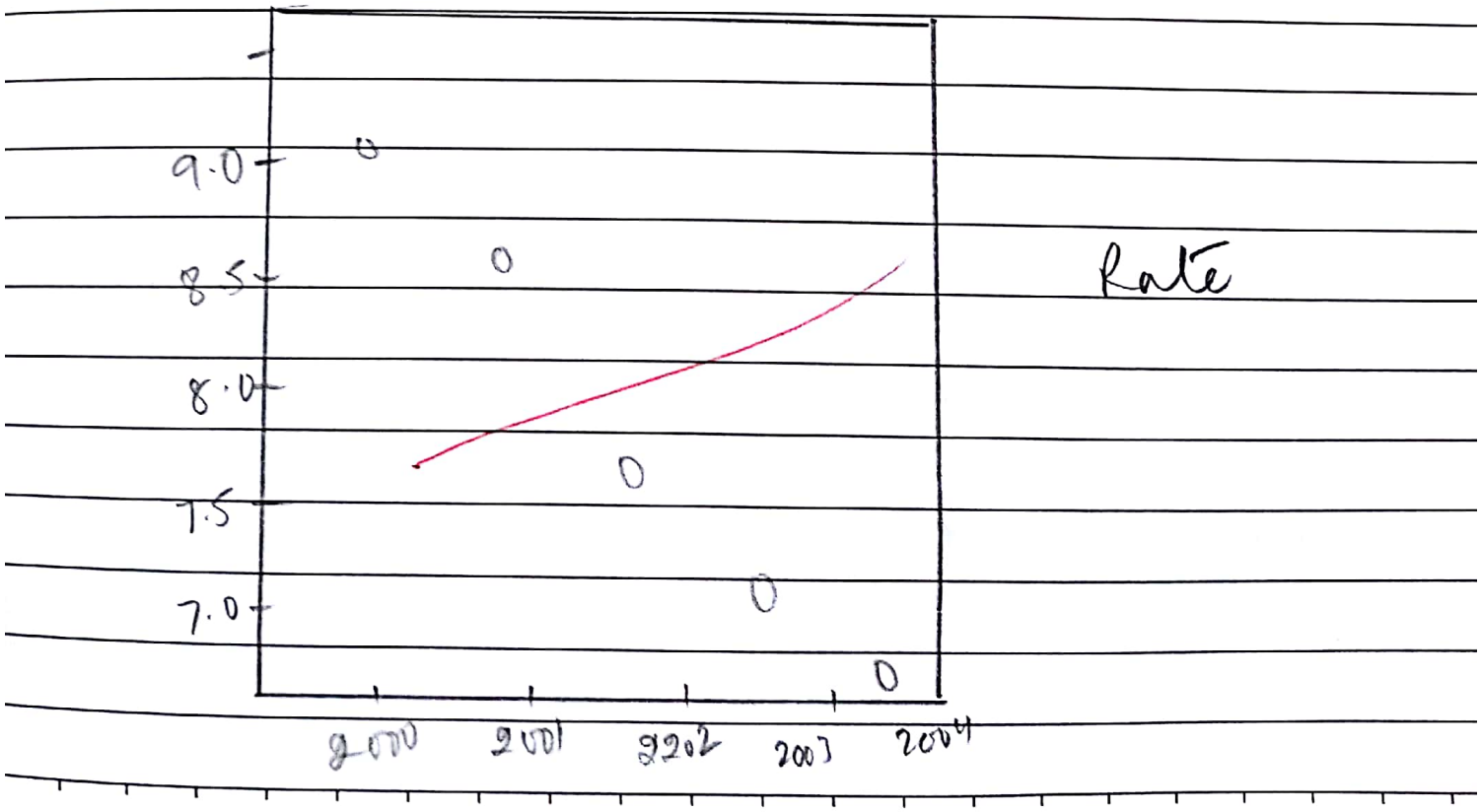
> plot (year, rate,

+ main = " commercial Banks, Insterest Rates for" year car loan

subs = " h # p : // www federdaam gov/ release / g 19 (2085805 / ")

> cor (year, rate)

[1] - 0.9880131155.



```
> fit <- lm(rate ~ year)
> fit

Call:
lm(formula = rate ~ year)

Coefficients:
(Intercept)         year
   1419.208       -0.705
```

you would like to know what else is stored in the variable you can use the attributes ommand:

```
> attributes(fit)
$names
 [1] "coefficients"  "residuals"     "effects"       "rank"          "fitted.values"
 [6] "assign"        "qr"            "df.residual"   "xlevels"       "call"
[11] "terms"         "model"

$class
[1] "lm"
```

One of the things you should notice is the coefficients variable within fit. You can print out the y-intercept and slope by accessing this part of the variable:

```
> fit$coefficients[1]
(Intercept)
   1419.208
> fit$coefficients[[1]]
[1] 1419.208
> fit$coefficients[2]
  year
-0.705
> fit$coefficients[[2]]
[1] -0.705
>
```

Note that if you just want to get the number you should use two square braces. So if you want to get an estimate of the interest rate in the year 2015 you can use the formula for a line:

```
> fit$coefficients[[2]]*2015+fit$coefficients[[1]]
[1] -1.367
>
```

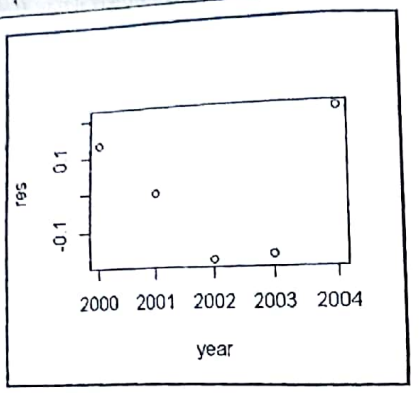
Page No. _____

Date : / / /

ics A Practical Approach to R

A better use for this formula would be to calculate the residuals and plot them:

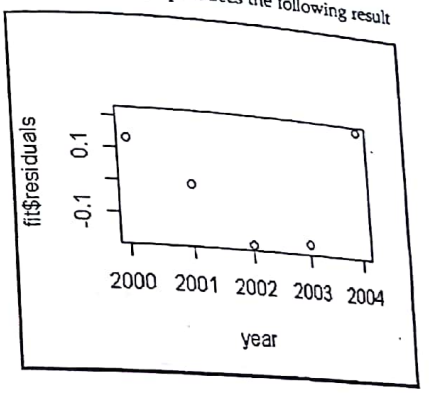
```
> res <- rate - (fit$coefficients[[2]]*year+fit$coefficients[[1]])
> res
[1]  0.132000000001 -0.003000000000 -0.178000000000 -0.163000000000  0.212000000000
> plot(year,res)
> |
```



That is a bit messy, but fortunately there are easier ways to get the residuals. Two ot ways are shown below:

```
> residuals(fit)
     1      2      3      4      5
 0.132 -0.003 -0.178 -0.163  0.212
> fit$residuals
     1      2      3      4      5
 0.132 -0.003 -0.178 -0.163  0.212
> plot(year,fit$residuals)
> |
```

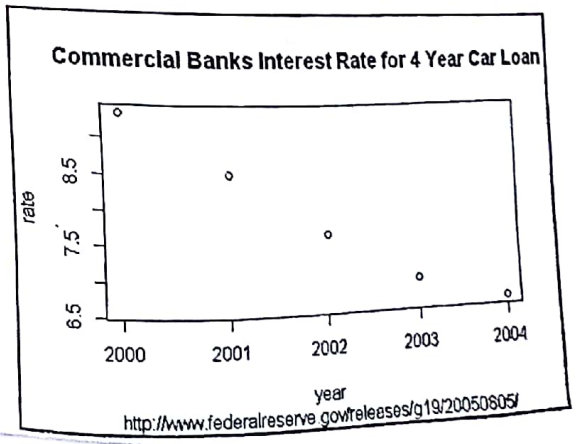
When we execute the above code, it produces the following result



If you want to plot the regression line on the same plot as your scatter plot you can use the abline function along with your variable fit:

```
> plot(year,rate,
+      main="Commercial Banks Interest Rate for 4 Year Car Loan",
+      sub="http://www.federalreserve.gov/releases/g19/20050805/")
> |
```

When we execute the above code, it produces the following result



Finally, as a teaser for the kinds of analyses you might see later, you can get the results of an F-test by asking R for a summary of the fit variable:

```
> summary(fit)

Call:
lm(formula = rate ~ year)

Residuals:
     1      2      3      4      5
 0.132 -0.003 -0.178 -0.163  0.212

Coefficients:
              Estimate    Std. Error   t value  Pr(>|t|)
(Intercept) 1419.20800000  126.94956528  11.17931 0.0015341 **
year          -0.70500000    0.06341136 -11.11788 0.0015592 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2005243 on 3 degrees of freedom
Multiple R-squared:  0.9763047, Adjusted R-squared:  0.9684062
F-statistic: 123.6073 on 1 and 3 DF,  p-value: 0.001559188
```

PRACTICAL - 8

Page No. _____
Date : / / /

→ Import the data from Excel / csv file and the chp - squared Test
syntaxi : chisq. test (data)

> Install - packages ("mass")
> library ("MASS")
> print cs # (cars93))
> # Create a date frame from the main data
> car - datas: date frame (cars 93 & An bags Cars 93 & Type)
> car - data: table (cars 93 & AirBags, Cars 93 & Type)
> print (car - data)

	Compact	large	Midsize	small	sporty	van
driver & Passanger	2	4	7	0	3	0
driver only	9	7	11	5	8	3
None	5	0	4	16	3	6

> Print (str) (cars 93)
. data frame": 93 obs. of 27 variables :
$ Manufacturer: Factors w/32 levels "Acura" "Audi.
$ Model: Factor w/93 level "100, 190 E" 11 22 744443
240" 49 56 91
6 2 4 5 4 1 4 7 3 35

$ Type: Factor w/ levels compact" large;, 43 13,32

min- Parsce: num 12.9.29.2 25.9. 308 21,

14.21, 9.9

22.6 26.3000

$ Price-: num 15.9 33.9 29.1 37.7 30 15.7

20.8 23.7 26.3

$ Max. Price num 18.8, 38.7. 32.3,

44.6. 36.9, 17.3, 12.7

24.9 26.3

$ MPG. City 8 int 25 18 20 19 22 19 16 16

$ MPG highway: int 31 25 26 26 30 31 28 25,24

$ Air bag and factors w/3 levels, drivers

.. passangers... 31 21222, 2222

$ Drivetrain: factors w/6, levels "4 '40" "front-

$ Cylinders: Factors w/6, levels '3 "4 "5 "6-

2, 2.444 222

$ Engine size: int. 8 1.2.28.28 15,22

3.8, 5.73.8

$ Horsepower: int 140 200 172 172 208 16

170 188 170

Page No. _____

Date : / / /

RPM : Int 140 200 172 172 208 110 170
180 ~~170~~

Fuel tanks : capacity. num 13.2.18 16.9 21.7
21.1 21.116 4 18

Passangers : Prt 5 5 5 6 4 6 6 6 56 ...

PRACTICAL - 5

Using R read the data from Excel csv file and calculate the standard deviation, variance, co-variance

Standard Deviation:

The standard deviation of a observation variable in the square root of its variance problems

Find the standard deviation of the eruption duration in the data set faithful.

Solution:

We apply the set function to compute the standard deviation of eruptions

```
> duration = faithful$eruptions
# the durations
> var(duration)
[1] 1.3027
```

Page No. _____

Date : | | |

co variance.

The W variance of two variables x & y in a data measures how the two are linearly related.

Positive covariance would indicate a positive linear relationship between the variables and a negative coverian e. would indicate the opposite.

The sample covariance is defined in terms of the sample mean as

$$s_{xy} = \frac{1}{n-1} \sum_{101} (x_i - \bar{x})(y_i - \bar{y})$$

Similarly, the population covariance is defined in terms of population mean μ_x μ_y as:

$$\sigma_{xy}. r_{xy} = \frac{1}{N} \sum_{i=}^{n} (x_i - \mu_x)(y_i - \mu_y)$$

problem. -

Find the convariance of eruption duration & wait time in the date set set faithful observe is there is any linear relation ship by the 2 variables.

Solution

We apply the cor function to compute the covariance of eruptions and waiting

> duration - faithful $ eruptions
> waiting: faithful $ waiting
> cov(duration, waiting) # apply the cor function (+) 12.9 %.

Page No. _____

PRACTICAL 6.

Date : / / /

→ using import the data from excel and csv file.
draw the skewness.

skewness.

The skewness of a data population is defined by following formula, where v_2 & v_3 are the second & third central moments.

Intutively the skewness is a measure of symmetry, rule the negative skewness indicates that the mean data values in less than the median & the data distribution is left skewed positively skewness would indicate that the mean of the data value is larger than the median. & the data distributed is right :-

Problem :-

Find the skewness of eruption durations is the data faithful.

Page No. _____

Date : | | |

Solutions :-

We apply the function skewerness from e1071 to computer the steven coefficient of erruption package is not in one lib. It has to see installed & loaded in R workspace

> Install packages("e1071")

> Liberary (e1071)

> duration = faithful $ eruptions #
eruptions duration.
skewness (duration)
[1] - 0.41355.