**Computer Oriented Statistical Techniques**

**Practical No.1**

Q1.Using R executes the basic commands array list and frames.

1. For basic Commands:

>mytext<-“Good Morning!”

>print (mytext)

Output:

[1] Good Morning!

2. For List:

#Create a List.

>list1<-list(c(2,6,8),1,2,3,sin)

>

>#Print the List.

>print (list1)

Output:

[[1]]

[1] 2 6 8

[[2]]

[1] 1

[[3]]

[1] 2

[[4]]

[1] 3

[[5]]

function (x) .Primitive("sin")

3. For Array:

#Create an array

> A<-array(c(‘yes’,’no’),dim=c(3,3,2))

> print (A)

Output:

, , 1

[,1] [,2] [,3]

[1,] “yes” “no” “yes”

[2,] “no” “yes” “no”

[3,] “yes” “no” “yes”

, , 2

[,1] [,2] [,3]

[1,] “no” “yes” “no”

[2,] “yes” “no” “yes”

[3,] “no” “yes” “no”

4. For Data Frames

#Create the Data Frame

> BMI<-data.frame(

+ Gender =c("Female","Female","Male","Female"),

+ Height =c(152,155,165.2,170.0),

+ Weight =c(35,40,60,55),

+ Age=c(14, 16, 20, 17)

+ )

> print(BMI)

Output:

Gender Height Weight Age

1 Female 152.0 35 14

2 Female 155.0 40 16

3 Male 165.2 60 20

4 Female 170.0 55 17

**Practical No. 2**

**Create a Matrix using R and perform the operations addition, inverse, transpose, and multiplication operations.**

> # Create two 2x3 matrices.

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

> print(matrix1)

[,1] [,2] [,3]

[1,] 3 -1 2

[2,] 9 4 6

>

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

> print(matrix2)

[,1] [,2] [,3]

[1,] 5 0 3

[2,] 2 9 4

>

> # Add the matrices.

> result <- matrix1 + matrix2

> cat("Result of addition","\n")

Result of addition

> print(result)

[,1] [,2] [,3]

[1,] 8 -1 5

[2,] 11 13 10

>

> # Subtract the matrices

> result <- matrix1 - matrix2

> cat("Result of subtraction","\n")

Result of subtraction

> print(result)

[,1] [,2] [,3]

[1,] -2 -1 -1

[2,] 7 -5 2

> # Multiply the matrices.

> result <- matrix1 \* matrix2

> cat("Result of multiplication","\n")

Result of multiplication

> print(result)

[,1] [,2] [,3]

[1,] 15 0 6

[2,] 18 36 24

>

> # Divide the matrices

> result <- matrix1 / matrix2

> cat("Result of division","\n")

Result of division

> print(result)

[,1] [,2] [,3]

[1,] 0.6 -Inf 0.6666667

[2,] 4.5 0.4444444 1.5000000

Inverse of Matrix:

> solve(A)

[,1] [,2] [,3]

[1,] 0.0625 0.0625 0.125

[2,] 0.6875 -0.3125 -0.625

[3,] 0.2500 0.2500 -0.500

Transpose of Matrix:

> t(matrix1)

[,1] [,2]

[1,] 3 9

[2,] -1 4

[3,] 2 6

**Practical No. 3**

**Using R Execute the statistical functions: mean, median, mode, quartile, range, inter quartile range histogram. Using R import the data from Excel / .CSV file and Perform the above functions.**

**For Mean:**

**(a) Find the mean of 12,7,3,4,2,18,2,54,-21,8,-5.**

>#Create a Vector.

> x<-c(12,7,3,4,2,18,2,54,-21,8,-5)

>#Find Mean

> Resultmean<-mean(x)

> print(mean)

Output:

[1] 7.636364

**(b) Find the mean of 12,7,3,4,2,18,2,54,-21,8,-5 applying trim option.**

>#Create a Vector.

> x<-c(12,7,3,4,2,18,2,54,-21,8,-5)

>#Find Mean

> Resultmean<-mean(x,trim=0.3)

> print(mean)

Output:

[1] 7.636364

**(c) Find the mean of 12,7,3,4,2,18,2,54,-21,8,-5 applying NA option.**

> #crate a vector

> x<-c(12,7,3,4,2,18,2,54,-21,8,-5,NA)

> #Find Mean

> result.mean<-mean(x)

> print(result.mean)

[1] NA

> #Find Mean Dropping NA values

> result.mean<-mean(x,na.rm= TRUE)

> print(result.mean)

Output:

[1] 7.636364

**For median:**

**Find the median of 12,7,3,4,2,18,2,54,-21,8,-5.**

> #crate a vector

> x<-c(12,7,3,4,2,18,2,54,-21,8,-5)

> #Find Median

> result.median<-median(x)

> print(result.median)

Output;

[1] 4

**For Mode:**

**Find the mode of 2, 1, 2, 3, 1, 2, 3, 4, 1, 5, 5, 3, 2, 2, 3.**

> # Create the function.

> getmode <- function(v) {

+ uniqv <- unique(v)

+ uniqv[which.max(tabulate(match(v, uniqv)))]

+ }

>

> # Create the vector with numbers.

> v <- c(2,1,2,3,1,2,3,4,1,5,5,3,2,3)

>

> # Calculate the mode using the user function.

> result <- getmode(v)

> print(result)

Output:

[1] 2

> # Create the vector with characters.

> charv <- c("o","it","the","it","it")

>

> # Calculate the mode using the user function.

> result <- getmode(charv)

> print(result)

Output:

[1] "it"

**For Range:**

**(a)** **Find the range of the eruption duration in the data set faithful.**

> duration = faithful$eruptions

> max(duration) − min(duration)

Output:

[1] 3.5

**(b)Find the range of the following data: 45, 43, 46 , 48, 51, 46, 50, 47, 46, 45.**

> x<-c(45,43,46,48,51,46,50,47,46,45)

> max(x)

[1] 51

> min(x)

[1] 43

> max(x)-min(x)

Output:

[1] 8

**For Interquartile Range:**

**(a)** **Find the interquartile range of the eruption duration in the data set faithful.**

> duration = faithful$eruptions

> IQR(duration)

Output:

[1] 2.2915

**(b) Find the interquartile range of the following: 29, 19, 26, 12, 24, 21, 36, 25, 33.**

Arrange the given data in ascending order of magnitude, we get

12,19,21,24,26,29,33,35,36

> x<-c(12,19,21,24,26,29,33,35,36)

> IQR(x)

Output;

[1] 12

**For Quartile:**

**The junior BS Stat students of MSU-IIT have the following SASE scores: 88, 84, 83, 80, 94, 90, 81, 79, 79, 81, 85, 87, 86, 89, and 92. Determine and interpret the quartiles of these scores.**

> scores <- c(88,84,83,80,94,90,81,79,79,81,85,87,86,89,92)

> quantile(scores)

Output:

0% 25% 50% 75% 100%

79.0 81.0 85.0 88.5 94.0

**Practical No. 4**

**Using R import the data from Excel / .CSV file and Calculate the Standard deviation, variance and co-variance.**

**faithfull.csv**

| id | eruptions | waiting |
| --- | --- | --- |
| 1 | 3.6 | 79 |
| 2 | 1.8 | 54 |
| 3 | 3.333 | 74 |
| 4 | 2.283 | 62 |
| 5 | 4.533 | 85 |
| 6 | 2.883 | 55 |

> library(moments)

> skewness(duration)

[1] -0.415841

> kurtosis(duration)

[1] 1.4994

> duration = faithful$eruptions # eruption durations

> waiting = faithful$waiting # the waiting period

> cov(duration, waiting) # apply the cov function

[1] 13.97781

> duration = faithful$eruptions # eruption durations

> sd(duration)

[1] 1.141371

> duration = faithful$eruptions # eruption durations

> var(duration)

[1] 1.302728

**Practical No.5**

**Using R import the data from Excel/.CSV file and draw the Skewness.**

**faithfull.csv**

| id | eruptions | waiting |
| --- | --- | --- |
| 1 | 3.6 | 79 |
| 2 | 1.8 | 54 |
| 3 | 3.333 | 74 |
| 4 | 2.283 | 62 |
| 5 | 4.533 | 85 |
| 6 | 2.883 | 55 |

> library(moments)

> skewness(duration)

[1] -0.415841

> kurtosis(duration)

[1] 1.4994

**Practical No 6**

**Import the Data from Excel/.CSV file and perform the hypothetical testing.**

**Soln: Lower Tail Test of Population mean with known Variance**

**Problem:** Suppose the manufacturer claims that the mean lifetime of a light bulb is more than 10,000 hours. In a sample of 30 light bulbs, it was found that they only last 9,900 hours on average. Assume the population standard deviation is 120 hours. At 0.05 significance level, can we reject the claim by manufacturer?

Solution:

The null hypothesis is that *µ* ≥ 10000. We begin with computing the test statistics.

> xbar=9900 #sample mean

> mu0=10000 #hypothesized value

> sigma=120 #population standard deviation

> n=30 #sample size

> z=(xbar-mu0)/(sigma/sqrt(n))

> z # Test Statistics

[1] -4.564355

We than compute the critical value at 0.05 significance level

> alpha= 0.05

> t.alpha = qt(1-alpha,df=n-1)

> t.alpha #critical value

[1] 1.699127

**Answer**

The test statistic 1.9720 is greater than the critical value of 1.699127. Hence, at 0.05significance level, we can reject the claim of the manufacturer.

**Alternative Solution**

Instead of using the critical value, we apply the **pt** function to compute the upper tail **p-value** of the test statistic. As it turns out to be less than the 0.05 significance level, we reject the null hypothesis that *µ* ≤ 2.

> pval=pt(z, df=n-1,lower.tail=FALSE)

> pval #upper tail p-value

[1] 0.9999575

**Upper Tail Test of Population mean with known Variance**

**Problem:** Suppose the food label on the cookie bag state that there is at most 2 grams of saturated fat in a single cookie. In a sample of 35 cookies, it is found that the mean amount of saturated fat per cookie is 2.1 grams. Assume that the population standard deviation is 0.25 grams. At 0.05 significance level, can we reject the claim of food label?

**Solution:**

The null hypothesis that *µ* ≤ 2, We begin with computing the test statistics.

> xbar=2.1 #sample mean xbar

> mu0=2 #hypothesized value

> sigma=0.25 #population standard deviation

> n=35 #sample size

> z=(xbar-mu0)/(sigma/sqrt(n))

> z # Test Statistics

[1] 2.366432

We than compute the critical value at 0.05 significance level

> alpha= 0.05

> z.alpha = qnorm(1-alpha)

> z.alpha

[1] 1.644854

**Answer**

The test statistics 2.3664 is greater than the critical value of 1.6449. Hence, At 0.05 significance level, we reject the claim that there is at most 2 grams of saturated fat in a cookie.

**Alternative Solution**

Instead of using the critical value, we apply the **pnorm** function to compute the upper tail **p-value** of the test statistic. As it turns out to be less than the 0.05 significance level, we reject the null hypothesis that *µ* ≤ 2.

>pval=pnorm(z, lower.tail=FALSE)

> pval

[1] 0.008980239

**Two-Tail Test of Population Mean with Known Variance**

**Problem:** Suppose the mean weight of King Penguins found in an Antarctic colony last year was 15.4 kg. In a sample of 35 Penguins same time this year in the same colony, the mean Penguin weight is 14.6 kg. Assume The population standard deviation is 2.5 kg. At 0.05 significance level, can we reject the null hypothesis that the mean Penguin weight does not differ from last year?

**Solution**

The null hypothesis that *µ* ≤ 2, We begin with computing the test statistics.

> xbar=14.6 #sample mean xbar

> mu0=15.4 #hypothesized value

> sigma=2.5 #population standard deviation

> n=35 #sample size

> z=(xbar-mu0)/(sigma/sqrt(n))

> z

[1] -1.893146

We than compute the critical value at 0.05 significance level

> alpha= 0.05

> z.half.alpha = qnorm(1-alpha/2)

> c(-z.half.alpha, z.half.alpha)

[1] -1.959964 1.959964

**Answer**

The test statistics -1.893146 lies between the critical values -1.959964 and 1.959964. Hence, At 0.05 significance level, we do not reject the null hypothesis that the mean Penguin weight does not differ from last year.

**Alternative Solution**

Instead of using the critical value, we apply the **pnorm** function to compute the two-tail **p-value** of the test statistic. It doubles the lower tail p-value as the sample mean is less than the hypothesized value. Since it turns out to be greater than the 0.05 significance level, we do not reject the null hypothesis that µ = 15.4.

> pval= 2\*pnorm(z) #lower tail

> pval #Two-Tail p-value

[1] 0.05833852

**Two-Tailed Test of Population Mean with Unknown Variance.**

**Problem:** Suppose the mean weight of King Penguins found in an Antarctic colony last year was 15.4 kg. In a sample of 35 Penguins same time this year in the same colony, the mean Penguin weight is 14.6 kg. Assume The population standard deviation is 2.5 kg. At 0.05 significance level, can we reject the null hypothesis that the mean Penguin weight does not differ from last year?

**Solution**

The null hypothesis that *µ* ≤ 2, We begin with computing the test statistics.

> xbar=14.6 #sample mean xbar

> mu0=15.4 #hypothesized value

> sigma=2.5 #population standard deviation

> n=35 #sample size

> t=(xbar-mu0)/(sigma/sqrt(n))

> t #test statistic

[1] -1.893146

We than compute the critical value at 0.05 significance level

> alpha= 0.05

> t.half.alpha = qt(1-alpha/2,df=n-1)

> c(-t.half.alpha, t.half.alpha)

[1] -2.032245 2.032245

**Answer**

The test statistics -1.893146 lies between the critical values -2.032245 and 2.032245. Hence, At 0.05 significance level, we do not reject the null hypothesis that the mean Penguin weight does not differ from last year.

**Alternative Solution**

Instead of using the critical value, we apply the **pt** function to compute the two-tail **p-value** of the test statistic. It doubles the lower tail p-value as the sample mean is less than the hypothesized value. Since it turns out to be greater than the 0.05 significance level, we do not reject the null hypothesis that µ = 15.4.

> pval=2\*pt(t, df=n-1) #lower tail

> pval #two-tailed p-value

[1] 0.06687552

**Two-Tailed Test of Population Proportion**

**Problem:** Suppose the coin tossed turns up 12 heads out of 20 tails. At 0.05 significance level, can one reject the null hypothesis that the coin toss is fair?

**Solution:**

The null hypothesis that *p =* 0.5, We begin with computing the test statistics.

> pbar=12/20 #sample mean xbar

> p0=0.5 #hypothesized value

> n=20 #sample size

> z=(pbar-p0)/sqrt(p0\*(1-p0)/n)

> z

[1] 0.8944272

We than compute the critical value at 0.05 significance level

>alpha= 0.05

> z.half.alpha = qnorm(1-alpha/2)

> c(-z.half.alpha, z.half.alpha)

[1] -1.959964 1.959964

**Alternative Solution 1**

Instead of using the critical value, we apply the **pnorm** function to compute the two-tail **p-value** of the test statistic. It doubles the upper tail p-value as the sample projection is greater than the hypothesized value. Since it turns out to be greater than the 0.05 significance level, we do not reject the null hypothesis that p = 0.5.

> pval = 2 \* pnorm(z, lower.tail=FALSE) #upper tail

> pval

[1] 0.3710934

**Alternative Solution 2**

We apply prop.test function to compute the p-value directly. The Yates continuity correction is disabled for pedagogical reasons.

> prop.test(12, 20, p=0.5, correct=FALSE)

1-sample proportions test without continuity correction

data: 12 out of 20, null probability 0.5

X-squared = 0.8, df = 1, p-value = 0.3711

alternative hypothesis: true p is not equal to 0.5

95 percent confidence interval:

0.3865815 0.7811935

sample estimates:

p

0.6

**Practical No 7**

**Import the data from Excel / CSV and perform the Chi-Square Test.**

library("MASS")

> print(str(Cars93))

Output:

'data.frame': 93 obs. of 27 variables:

$ Manufacturer : Factor w/ 32 levels "Acura","Audi",..: 1 1 2 2 3 4 4 4 4 5 ...

$ Model : Factor w/ 93 levels "100","190E","240",..: 49 56 9 1 6 24 54 74 73 35 ...

$ Type : Factor w/ 6 levels "Compact","Large",..: 4 3 1 3 3 3 2 2 3 2 ...

$ Min.Price : num 12.9 29.2 25.9 30.8 23.7 14.2 19.9 22.6 26.3 33 ...

$ Price : num 15.9 33.9 29.1 37.7 30 15.7 20.8 23.7 26.3 34.7 ...

$ Max.Price : num 18.8 38.7 32.3 44.6 36.2 17.3 21.7 24.9 26.3 36.3 ...

$ MPG.city : int 25 18 20 19 22 22 19 16 19 16 ...

$ MPG.highway : int 31 25 26 26 30 31 28 25 27 25 ...

$ AirBags : Factor w/ 3 levels "Driver & Passenger",..: 3 1 2 1 2 2 2 2 2 2 ...

$ DriveTrain : Factor w/ 3 levels "4WD","Front",..: 2 2 2 2 3 2 2 3 2 2 ...

$ Cylinders : Factor w/ 6 levels "3","4","5","6",..: 2 4 4 4 2 2 4 4 4 5 ...

$ EngineSize : num 1.8 3.2 2.8 2.8 3.5 2.2 3.8 5.7 3.8 4.9 ...

$ Horsepower : int 140 200 172 172 208 110 170 180 170 200 ...

$ RPM : int 6300 5500 5500 5500 5700 5200 4800 4000 4800 4100 ...

$ Rev.per.mile : int 2890 2335 2280 2535 2545 2565 1570 1320 1690 1510 ...

$ Man.trans.avail : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 1 1 1 1 ...

$ Fuel.tank.capacity: num 13.2 18 16.9 21.1 21.1 16.4 18 23 18.8 18 ...

$ Passengers : int 5 5 5 6 4 6 6 6 5 6 ...

$ Length : int 177 195 180 193 186 189 200 216 198 206 ...

$ Wheelbase : int 102 115 102 106 109 105 111 116 108 114 ...

$ Width : int 68 71 67 70 69 69 74 78 73 73 ...

$ Turn.circle : int 37 38 37 37 39 41 42 45 41 43 ...

$ Rear.seat.room : num 26.5 30 28 31 27 28 30.5 30.5 26.5 35 ...

$ Luggage.room : int 11 15 14 17 13 16 17 21 14 18 ...

$ Weight : int 2705 3560 3375 3405 3640 2880 3470 4105 3495 3620 ...

$ Origin : Factor w/ 2 levels "USA","non-USA": 2 2 2 2 2 1 1 1 1 1 ...

$ Make : Factor w/ 93 levels "Acura Integra",..: 1 2 4 3 5 6 7 9 8 10 ...

NULL

> library("MASS")

> # Load the library.

> library("MASS")

>

> # Create a data frame from the main data set.

> car.data <- data.frame(Cars93$AirBags, Cars93$Type)

>

> # Create a table with the needed variables.

> car.data = table(Cars93$AirBags, Cars93$Type)

> print(car.data)

Output:

Compact Large Midsize Small Sporty Van

Driver & Passenger 2 4 7 0 3 0

Driver only 9 7 11 5 8 3

None 5 0 4 16 3 6

> # Perform the Chi-Square test.

> print(chisq.test(car.data))

Output: Pearson's Chi-squared test

data: car.data

X-squared = 33.001, df = 10, p-value = 0.0002723

Warning message:

In chisq.test(car.data) : Chi-squared approximation may be incorrect

**Practical No 8**

**Using R perform the binomial and normal distribution on the data.**

**R- Binomial distribution**

## dbinom()

This function gives the probability density distribution at each point.

> Create a sample of 50 numbers which are incremented by 1.

> x <- seq(0,50,by = 1)

>

> # Create the binomial distribution.

> y <- dbinom(x,50,0.5)

>

> # Give the chart file a name.

> png(file = "dbinom.png")

>

> # Plot the graph for this sample.

> plot(x,y)

>

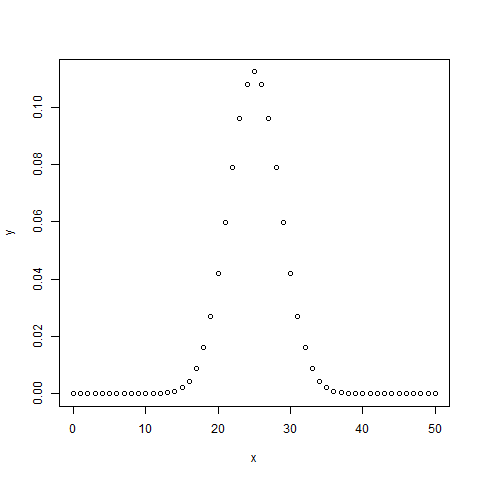
> # Save the file.

> dev.off()

null device

1

> plot(x,y)



## pbinom()

This function gives the cumulative probability of an event. It is a single value representing the probability.

># Probability of getting 26 or less heads from a 51 tosses of a coin.

>x <- pbinom(26,51,0.5)

>print(x)

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

[1] 0.610116

## qbinom()

This function takes the probability value and gives a number whose cumulative value matches the probability value.

># How many heads will have a probability of 0.25 will come out when a coin is tossed 51 times.

>x <- qbinom(0.25,51,1/2)

>print(x)

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

[1] 23

## rbinom()

This function generates the required number of random values of given probability from a given sample.

># Find 8 random values from a sample of 150 with probability of 0.4.

>x <- rbinom(8,150,.4)

>print(x)

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

[1] 58 61 59 66 55 60 61 67

**R- Normal Distribution**

## dnorm()

This function gives height of the probability distribution at each point for a given mean and standard deviation.

># Create a sequence of numbers between -10 and 10 incrementing by 0.1.

>x <- seq(-10, 10, by = .1)

># Choose the mean as 2.5 and standard deviation as 0.5.

>y <- dnorm(x, mean = 2.5, sd = 0.5)

># Give the chart file a name.

>png(file = "dnorm.png")

>plot(x,y)

>#Save the file.

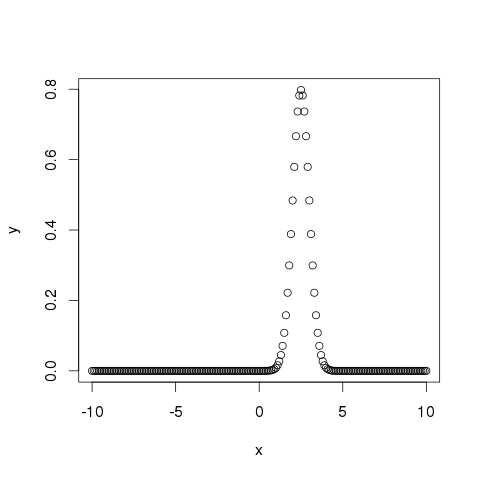
>dev.off()

windows

2

> plot(x,y)

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



## pnorm()

This function gives the probability of a normally distributed random number to be less that the value of a given number. It is also called "Cumulative Distribution Function".

># Create a sequence of numbers between -10 and 10 incrementing by 0.2.

>x <- seq(-10,10,by = .2)

># Choose the mean as 2.5 and standard deviation as 2.

>y <- pnorm(x, mean = 2.5, sd = 2)

># Give the chart file a name.

>png(file = "pnorm.png")

># Plot the graph.

>plot(x,y)

>#Save the file.

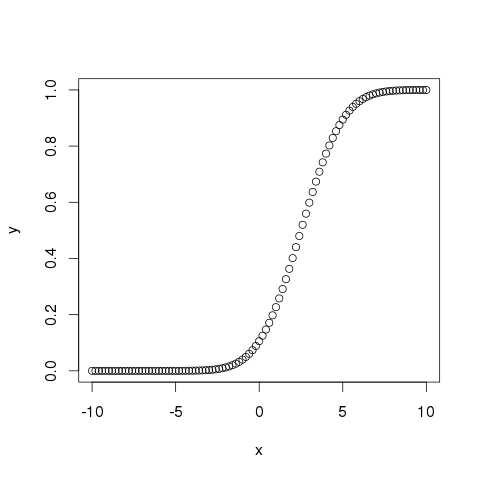
>dev.off()

Windows

2

>plot(x,y)

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



## qnorm()

This function takes the probability value and gives a number whose cumulative value matches the probability value.

> # Create a sequence of probability values incrementing by 0.02.

> x <- seq(0, 1, by = 0.02)

> # Choose the mean as 2 and standard deviation as 3.

> y <- qnorm(x, mean = 2, sd = 1)

> # Give the chart file a name.

> png(file = "qnorm.png")

> # Plot the graph.

> plot(x,y)

> # Save the file.

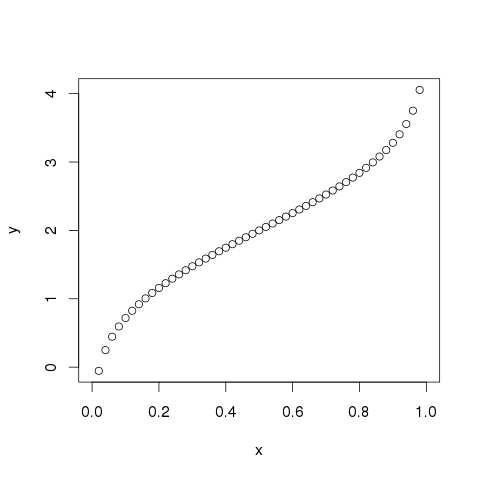
> dev.off()

windows

2

> plot(x,y)

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



## rnorm()

This function is used to generate random numbers whose distribution is normal. It takes the sample size as input and generates that many random numbers. We draw a histogram to show the distribution of the generated numbers.

> # Create a sample of 50 numbers which are normally distributed.

> y <- rnorm(50)

> # Give the chart file a name.

> png(file = "rnorm.png")

> # Plot the histogram for this sample.

> hist(y, main = "Normal Distribution")

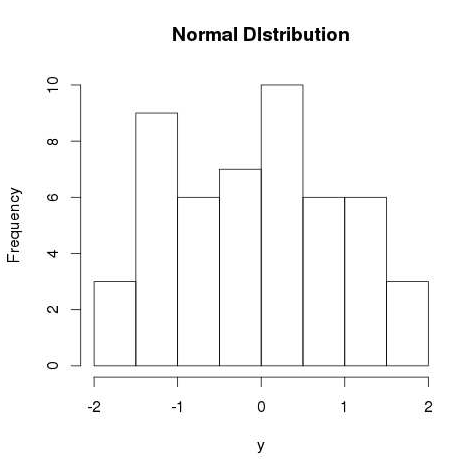
> # Save the file.

> dev.off()

windows

2

> hist(y, main = "Normal Distribution")

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

**Practical No 9**

**Perform the linear regression using R.**

### Input Data

Below is the sample data representing the observations −

# Values of height

151, 174, 138, 186, 128, 136, 179, 163, 152, 131

# Values of weight.

63, 81, 56, 91, 47, 57, 76, 72, 62, 48

## lm() Function

This function creates the relationship model between the predictor and the response variable.

### Syntax

The basic syntax for **lm()** function in linear regression is −

lm(formula,data)

Following is the description of the parameters used −

* **formula** is a symbol presenting the relation between x and y.
* **data** is the vector on which the formula will be applied.

### Create Relationship Model & get the Coefficients

>x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)

>y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)

># Apply the lm() function.

>relation <- lm(y~x)

>print(relation)

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

Call:

lm(formula = y ~ x)

Coefficients:

(Intercept) x

-38.4551 0.6746

### Get the Summary of the Relationship

>x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)

>y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)

># Apply the lm() function.

>relation <- lm(y~x)

>print(summary(relation))

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

Call:

lm(formula = y ~ x)

Residuals:

Min 1Q Median 3Q Max

-6.3002 -1.6629 0.0412 1.8944 3.9775

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) - 38.45509 8.04901 -4.778 0.00139 \*\*

x 0.67461 0.05191 12.997 1.16e-06 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.253 on 8 degrees of freedom

Multiple R-squared: 0.9548, Adjusted R-squared: 0.9491

F-statistic: 168.9 on 1 and 8 DF, p-value: 1.164e-06

## predict() Function

### Syntax

The basic syntax for predict() in linear regression is −

predict(object, newdata)

Following is the description of the parameters used −

* **object** is the formula which is already created using the lm() function.
* **newdata** is the vector containing the new value for predictor variable.

### Predict the weight of new persons

# The predictor vector.

x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)

# The resposne vector.

y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)

# Apply the lm() function.

relation <- lm(y~x)

# Find weight of a person with height 170.

a <- data.frame(x = 170)

result <- predict(relation,a)

print(result)

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

1

76.22869

### Visualize the Regression Graphically

# Create the predictor and response variable.

x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)

y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)

relation <- lm(y~x)

# Give the chart file a name.

png(file = "linearregression.png")

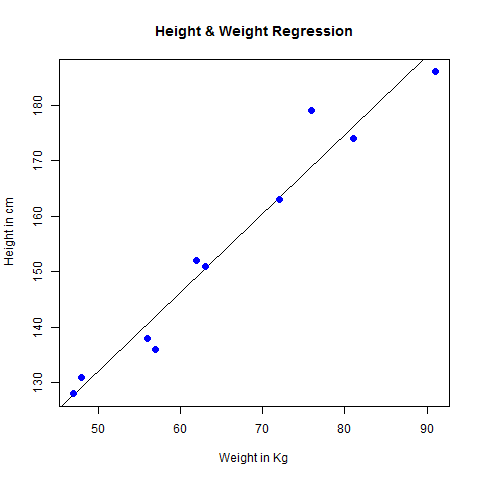
# Plot the chart.

plot(y,x,col = "blue",main = "Height & Weight Regression",

abline(lm(x~y)),cex = 1.3,pch = 16,xlab = "Weight in Kg",ylab = "Height in cm")

# Save the file.

dev.off()

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

**Practical No 10**

**Compute the Linear Least Square Regression.**

The way that this relationship is defined in the *lm* command is that you write the vector containing the response variable, a tilde (“~”), and a vector containing the explanatory variable:

> fit <- lm(rate ~ year)

> fit

Output:

Call:

lm(formula = rate ~ year)

Coefficients:

(Intercept) year

1419.208 -0.705

When you make the call to *lm* it returns a variable with a lot of information in it. If you are just learning about least squares regression you are probably only interested in two things at this point, the slope and the y-intercept. If you just type the name of the variable returned by *lm* it will print out this minimal information to the screen. (See above.)

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

> attributes(fit)

Output:

$names

[1] "coefficients" "residuals" "effects" "rank"

[5] "fitted.values" "assign" "qr" "df.residual"

[9] "xlevels" "call" "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

So if you just wait long enough, the banks will pay you to take a car!

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] 0.132 -0.003 -0.178 -0.163 0.212

> plot(year,res)

That is a bit messy, but fortunately there are easier ways to get the residuals. Two other 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)

>

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/")

> abline(fit)

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)

Output:

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) 1419.20800 126.94957 11.18 0.00153 \*\*

year -0.70500 0.06341 -11.12 0.00156 \*\*

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2005 on 3 degrees of freedom

Multiple R-Squared: 0.9763, Adjusted R-squared: 0.9684

F-statistic: 123.6 on 1 and 3 DF, p-value: 0.001559