Practical No.9
Using R perform the binomial and normal distribution on the data.

The binomial distribution model deals with finding the probability of success of an event which has only two possible outcomes in a series of experiments. For example, tossing of a coin always gives a head or a tail. The probability of finding exactly 3 heads in tossing a coin repeatedly for 10 times is estimated during the binomial distribution.

R has four in-built functions to generate binomial distribution. They are described below.

```
dbinom(x, size, prob)
pbinom(x, size, prob)
qbinom(p, size, prob)
rbinom(n, size, prob)
```

Following is the description of the parameters used −

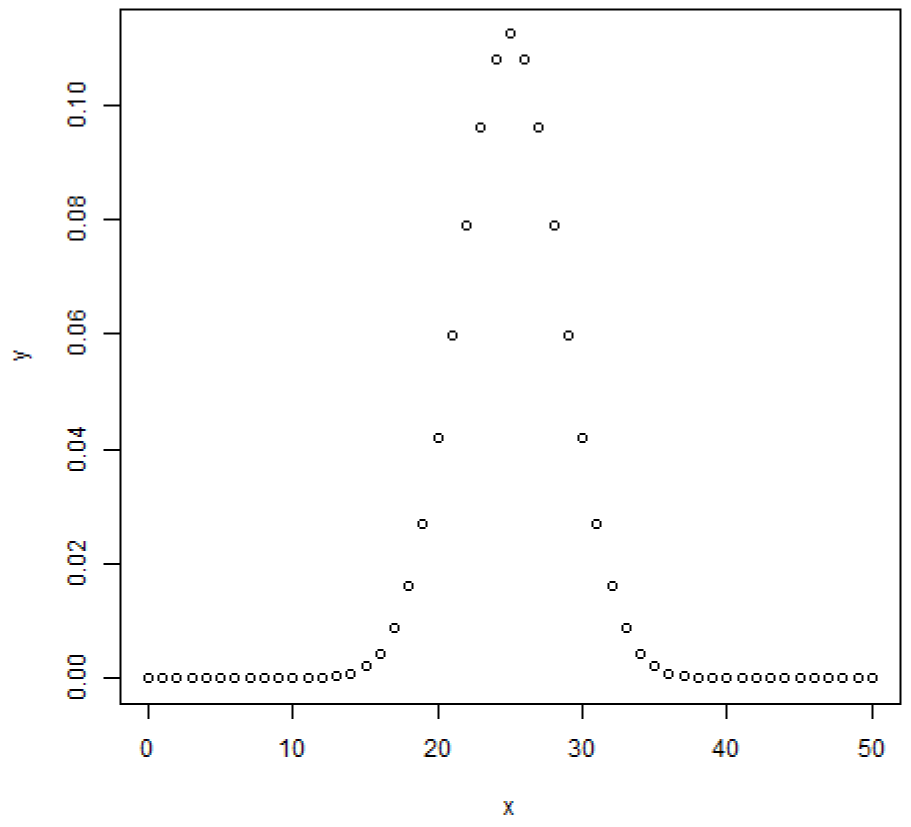
- **x** is a vector of numbers.
- **p** is a vector of probabilities.
- **n** is number of observations.
- **size** is the number of trials.
- **prob** is the probability of success of each trial.

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()
```

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



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 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
```

Suppose there are twelve multiple choice questions in an English class quiz. Each question has five possible answers, and only one of them is correct. Find the probability of having four or less correct answers if a student attempts to answer every question at random.

Solution

Since only one out of five possible answers is correct, the probability of answering a question correctly by random is 1/5=0.2. We can find the probability of having exactly 4 correct answers by random attempts as follows.

```
> dbinom(4, size=12, prob=0.2)
[1] 0.1329
```

To find the probability of having four or less correct answers by random attempts, we apply the function dbinom with x = 0,…,4.

```
 dbinom(0, size=12, prob=0.2) +
 dbinom(1, size=12, prob=0.2) +
 dbinom(2, size=12, prob=0.2) +
 dbinom(3, size=12, prob=0.2) +
 dbinom(4, size=12, prob=0.2)
[1] 0.9274
```

Alternatively, we can use the cumulative probability function for binomial distribution pbinom.

```
> pbinom(4, size=12, prob=0.2)
[1] 0.92744
```

Answer

The probability of four or less questions answered correctly by random in a twelve question multiple choice quiz is 92.7%.

Normal Distribution:

In a random collection of data from independent sources, it is generally observed that the distribution of data is normal. Which means, on plotting a graph with the value of the variable in the horizontal axis and the count of the values in the vertical axis we get a bell shape curve. The center of the curve represents the mean of the data set. In the graph, fifty percent of values lie to the left of the mean and the other fifty percent lie to the right of the graph. This is referred as normal distribution in statistics.

R has four in built functions to generate normal distribution. They are described below.

```
dnorm(x, mean, sd)
pnorm(x, mean, sd)
qnorm(p, mean, sd)
rnorm(n, mean, sd)
```

Following is the description of the parameters used in above functions −

- **x** is a vector of numbers.
- **p** is a vector of probabilities.
- **n** is number of observations(sample size).
- **mean** is the mean value of the sample data. It's default value is zero.
- **sd** is the standard deviation. It's default value is 1.

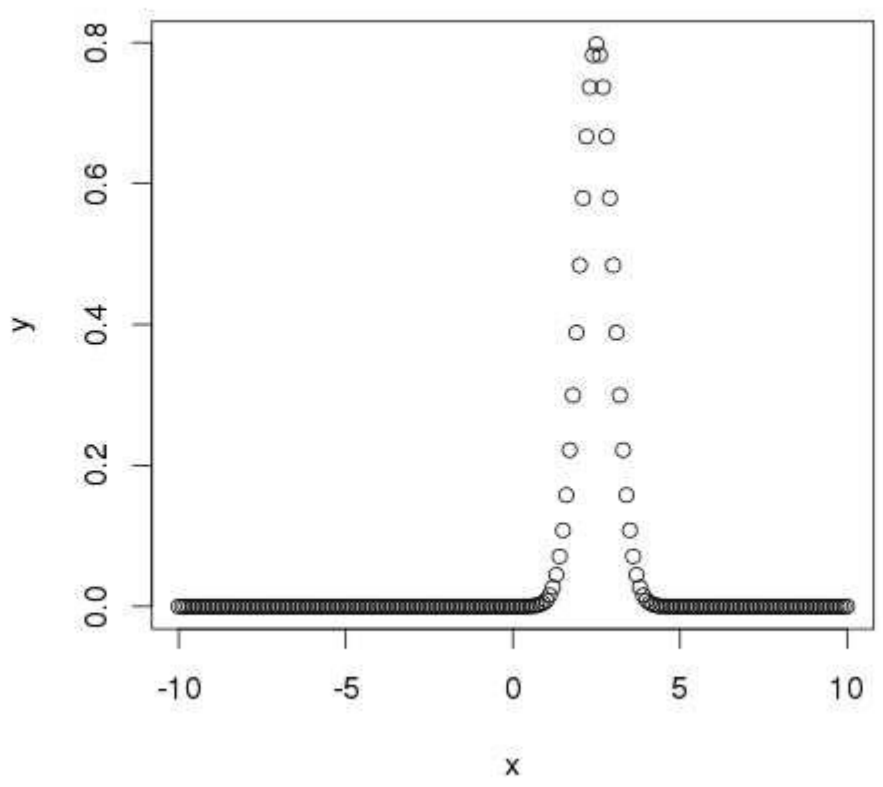
dnorm()

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()
```

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

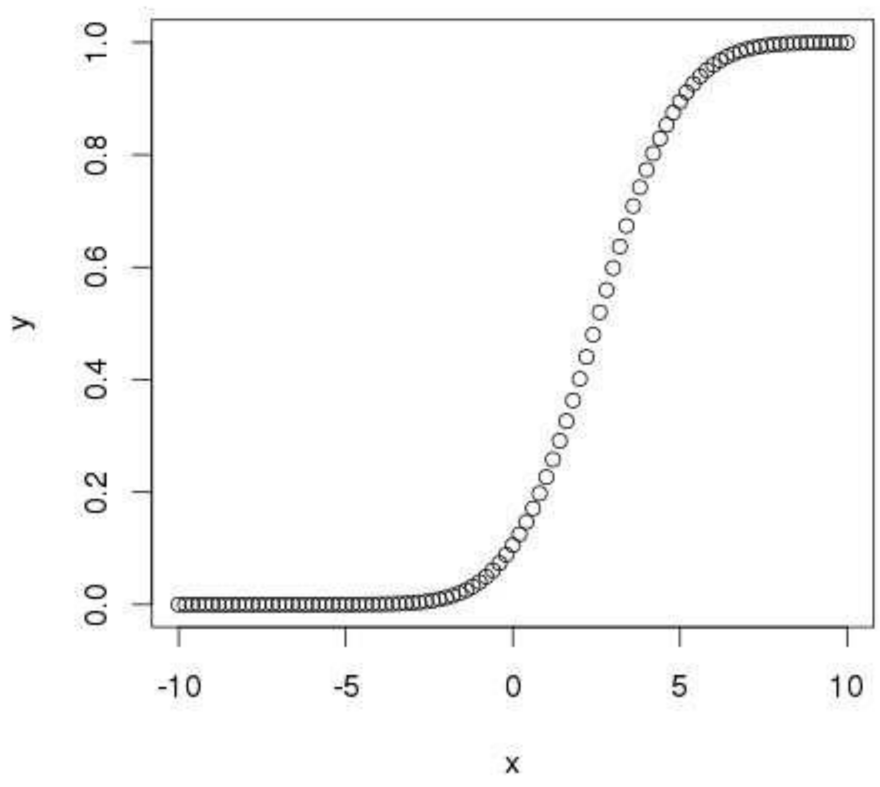


pnorm()

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".

```r
# 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()
```

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



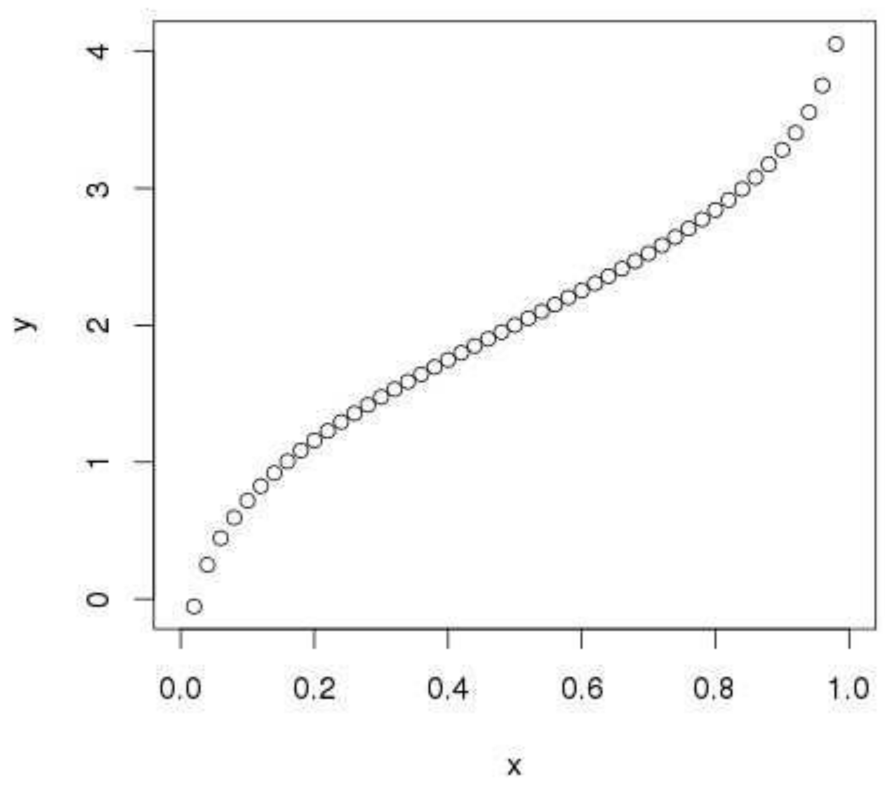
qnorm()

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()
```

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

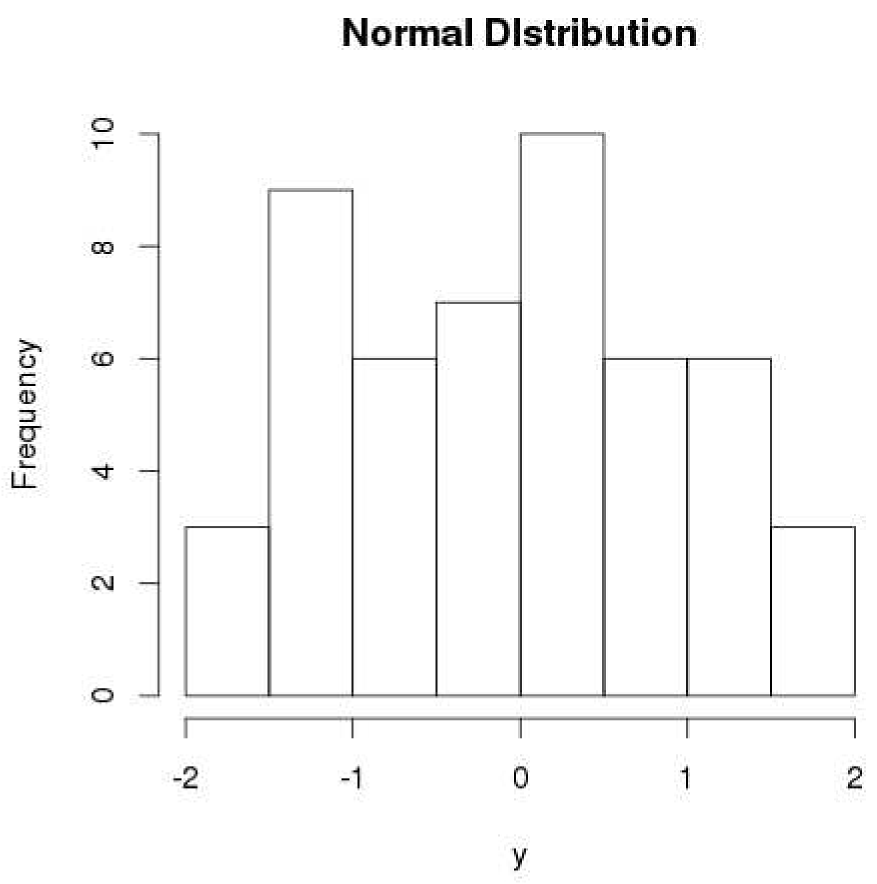


rnorm()

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()
```

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



Assume that the test scores of a college entrance exam fits a normal distribution. Furthermore, the mean test score is 72, and the standard deviation is 15.2. What is the percentage of students scoring 84 or more in the exam?

Solution

We apply the function pnorm of the normal distribution with mean 72 and standard deviation 15.2. Since we are looking for the percentage of students scoring higher than 84, we are interested in the *upper tail* of the normal distribution.

```
> pnorm(84, mean=72, sd=15.2, lower.tail=FALSE)
[1] 0.21492
```

Q. Suppose the number of games in which major league baseball players play during their careers is normally distributed with mean equal to 1500 games and standard deviation equal to 350 games. Use R to solve the following problems.

(a) What percentage play in fewer than 750 games?

(b) What percentage play in more than 2000 games?

(c) Find the 90th percentile for the number of games played during a career.

Solution:

```r
mu = 1500
s =350
x <- seq(0,4000, by =150)
curve(dnorm(x,mu,s),xlim=c(0,4000),main='Standard Normal', xlab =  "z")
abline(v=1500, lty = 2, col ="red")
# percentage play fewer than 750 games
k = 750
codx <- c(0,seq(0,750,150),750)
cody <- c(0,dnorm(seq(0,750,150),mu,s),0)
polygon(codx,cody,col='green')
p750 = round(pnorm(k, mean =mu, sd =s), digit = 3)
# percentage play in more than 2000


k = 2000
codx <- c(2000,seq(2000,4000,150),4000)
cody <- c(0,dnorm(seq(2000,4000,150),mu,s),0)
polygon(codx,cody,col='pink')
p2000 = round(pnorm(k, mean =mu, sd =s), digit = 3)
# percentage play in more than 2000
round(pnorm(k, mean =mu, sd =s, lower.tail = FALSE), digit = 3)
## [1] 0.077
# find the 90th percentile for the number of games played during a career
q90 <- qnorm(0.9, mean = mu, sd = s,
             lower.tail = TRUE, log.p = FALSE)
abline(v=1948.543, lty= 7, col = "blue")
```

```r
results <- c(p750,p2000, q90)
results
## [1]    0.016    0.923 1948.543
```

Problem 3. Find the area under the standard normal curve in the following case: (a) Between z =0.81 and z =1.94 (b) to the right of z = -1.28 (c) To the right of z = 2.05 or to the left of z = -1.44.

```r
mu = 0
s =1
```

```
par(mfrow=c(2,2))
x <- seq(-4,4, by =1)
curve(dnorm(x,mu,s),xlim=c(-4,4),
      main='Standard Normal', xlab =  "z")
abline(v=0, lty= 2, col = "red")
#Between z =0.81  and z =1.94
codx <- c(0.81, seq(0.81,1.94,0.1),1.94)
cody <- c(0,dnorm(seq(0.81,1.94,0.1),mu,s),0)
polygon(codx,cody,col='green')
r1 <- pnorm(1.94, mean = mu , sd = s)- pnorm(0.81, mean = mu , sd = s)
# (b) to the right of z = -1.28
curve(dnorm(x,mu,s),xlim=c(-4,4),
      main='Standard Normal', xlab =  "z")
codx <- c(-1.28, seq(-1.28,4,0.1),4)
cody <- c(0,dnorm(seq(-1.28,4,0.1),mu,s),0)
polygon(codx,cody,col='green')
r2 <- 1-pnorm(-1.28, mean = mu , sd = s)

#(c) To the right of z = 2.05 or to the left of z = -1.44.
curve(dnorm(x,mu,s),xlim=c(-4,4),
       main='Standard Normal', xlab =  "z")
codx <- c(2.05, seq(2.05,4,0.1),4)
cody <- c(0,dnorm(seq(2.05,4,0.1),mu,s),0)
polygon(codx,cody,col='green')

codx <- c(-4, seq(-4,-1.44,0.1),-1.44)
cody <- c(0,dnorm(seq(-4,-1.44,0.1),mu,s),0)
polygon(codx,cody,col='green')
r3 <- 1-pnorm(2.05, mean = mu , sd = s) + pnorm(-1.44, mean = mu , sd = s)
Results <- c(r1,r2,r3)
Results
## [1] 0.18278024 0.89972743 0.09511591
```

Problem 4 . The time spent watching tv per week by middle - school students has a normal distribution with a mean 20.5 and standard deviation =5.5 hrs. Find the student percent who watch less than 25hrs per week ?

```r
mu = 20.5

s =5.5

par(mfrow=c(1,2))

#Find the student percent who watch less than 25hrs per week ?

r25 <- pnorm(25, mean = mu , sd = s)


codx <- c(0,seq(0,25,1),25)

cody <- c(0,dnorm(seq(0,25,1),mu,s),0)

curve(dnorm(x,mu,s),xlim=c(0,40),main=' Less 25Hrs perWeek on TV')

abline(v=25, lty= 2, col = "blue")

polygon(codx,cody,col='red')


#Find the percent who watch over 30 hours per weekdaysDate()?

r30 <- pnorm(30, mean = mu , sd = s, lower.tail = FALSE)

curve(dnorm(x,mu,s),xlim=c(0,40),main='More 30Hrs per Week on TV')

abline(v=30, lty= 2, col = "blue")

codx <- c(30,seq(30,40,1),40)

cody <- c(0,dnorm(seq(30,40,1),mu,s),0)

polygon(codx,cody,col='red')
```

```r
results <- c(r25, r30)

results
```

```
## [1] 0.79337331 0.04205935
```

Practical No.10

Regression analysis is a very widely used statistical tool to establish a relationship model between two variables. One of these variable is called predictor variable whose value is gathered through experiments. The other variable is called response variable whose value is derived from the predictor variable.

In Linear Regression these two variables are related through an equation, where exponent (power) of both these variables is 1. Mathematically a linear relationship represents a straight line when plotted as a graph. A non-linear relationship where the exponent of any variable is not equal to 1 creates a curve.

The general mathematical equation for a linear regression is −

```
y = ax + b
```

Following is the description of the parameters used −

- **y** is the response variable.

- **x** is the predictor variable.
- **a** and **b** are constants which are called the coefficients.

Steps to Establish a Regression

A simple example of regression is predicting weight of a person when his height is known. To do this we need to have the relationship between height and weight of a person.

The steps to create the relationship is −

- Carry out the experiment of gathering a sample of observed values of height and corresponding weight.
- Create a relationship model using the **lm()** functions in R.
- Find the coefficients from the model created and create the mathematical equation using these
- Get a summary of the relationship model to know the average error in prediction. Also called **residuals**.
- To predict the weight of new persons, use the **predict()** function in 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

Live Demo

```
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

Live Demo

```
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

Live Demo

```
# 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 −

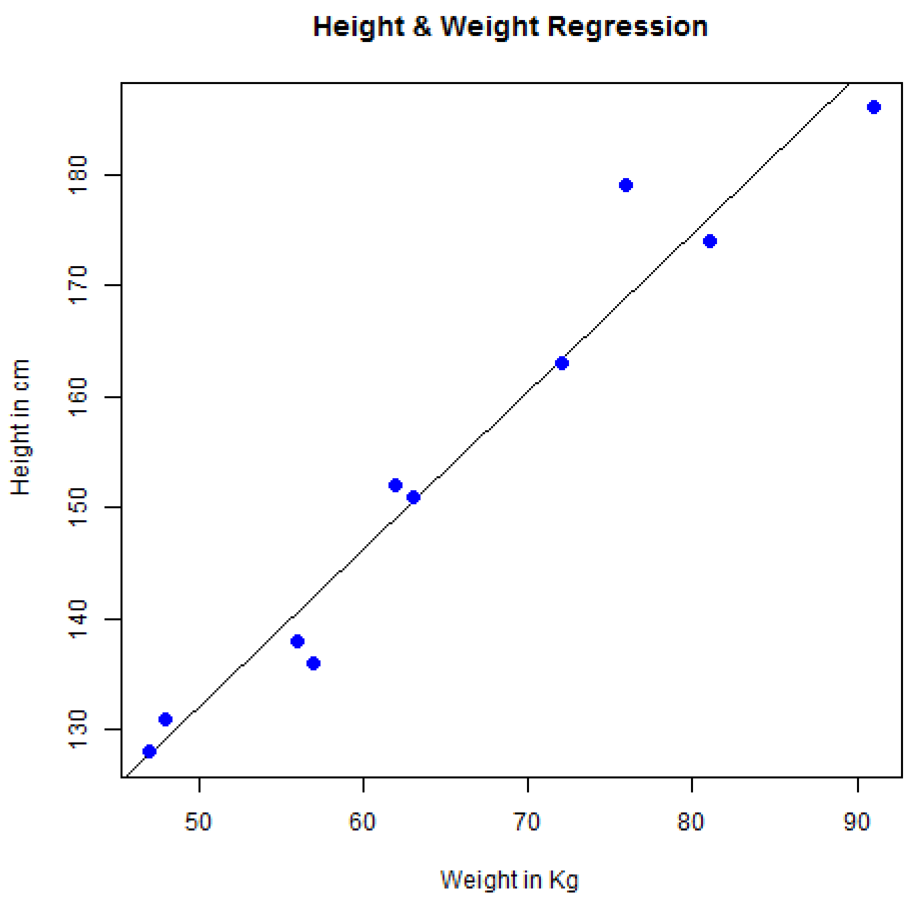
```
       1 
76.22869 
```

Visualize the Regression Graphically

Live Demo

```
# 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 −


Height & Weight Regression
Height in cm
130
140
150
160
170
180
50
60
70
80
90
Weight in Kg