VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

MAT2001 LAB
EXERCISES SUBMISSION

Reg Num: 20BAI1158

Name: Delano Oscar Do Rosario Lourenco

Course: Statistics For Engineers

Faculty: Dr. Jaganathan B

Semester: WS 20-21

Contents

Introduction to R

Aim: To understand basic R commands and programming.

Commands

help or ?	Shows the help menu for a given command
log(x, b)	Computes the logarithm of variable x to the base b
rep(x, n)	Makes vector of size n with the value x repeated
seq(x, n, d)	Makes a sequence starting from x, incrementing by d for a total of n number of times.
sin, cos, tan, arcsin, arrcos, etc	Basic trigonometric functions
getwd()	Returns the current working directory
setwd(path)	Sets the current working directory to the given path
ls()	Lists the files in current directory
rm(f)	Removes the file with name f in the current directory
rm(lm=())	Removes all files from R workspace
cat("\f")	Clears the console
cat(data)	Outputs the data, concatenating the representation

Examples

```
> a = 10;
> b = 20;
> log(a, 10);
[1] 1
> rep(a, 5);
[1] 10 10 10 10 10
> seq(a, 10, 1);
[1] 10
> sin(a) + tan(a);
[1] 0.1043397
> cat("hello")
hello
```

Variables

Aim: To understand variables can be declared and accessed in R as well as types of variables.

An integer variable “x” with value 1

```
> x <- 1;
> x
[1] 1
```

An integer variable “y” with value 2

```
> 2 -> y;
> y;
[1] 2
```

A string variable “name” with value “Delano”

```
> name = "Delano";
> name;
[1] "Delano"
```

A float variable “I” with 0.0000089

```
> i = 0.0000089;
> i
[1] 8.9e-06
```

Vectors, Arrays, and Data Frames

Aim: To understand the concept of Vectors and Data Frames in R.

Commands

c(1,2,3, . . .)	Combines the arguments in the form of a vector
data.frame(vec1, vec2, vec3, . . .)	Used for storing data tables. It is a list of vectors of equal length.
NROW(data)	Returns the number of rows present in data
NCOL(data)	Returns the number of columns present in data

Examples

```
> y=c(1,2,3,4,5)
> y
[1] 1 2 3 4 5
```

```
> x = data.frame(0, 2, 4, 8);
> x
  X0 X2 X4 X8
1  0  2  4  8
```

```
> marks = c(6, 7, 8, 9, 10);
> numStudents = c(2, 20, 6, 8, 5);
> table = data.frame(marks, numStudents);
> table
   marks numStudents
1      6           2
2      7          20
3      8           6
4      9           8
5     10           5
```

```
> col1 = c(1, 2, 3);
> col2 = c("John", "Adam", "Jane");
> table = data.frame(col1, col2)
> table
  col1 col2
1    1 John
2    2 Adam
3    3 Jane
> NROW(table);
[1] 3
> NCOL(table);
[1] 2
```

```
> v1 = c(1, 2);
> v2 = c(6, 2);
> # add vectors
> v1 + v2;
[1] 7 4
> # subtract vectors
> v1 - v1;
[1] 0 0
> # multiply scalar
> v1 * 2;
[1] 2 4
> # member wise multiplication
> v1 * v2;
[1] 6 4
```

```
> # adding vectors of unequal dimensions
> v1 = c(1,2,3);
> v2 = c(10, 10, 10, 10, 10);
> v1 + v2
[1] 11 12 13 11 12
```

String Manipulation

Aim: To understand how to print and parse strings and variables in different formats in R.

Commands

paste(s1, s2, . . .)	Concatenates vectors, strings, etc by making them as strings.
format(x, trim, digits, nsmall, scientific)	Formats a variable "x", according to: **digits:** total number of digits, **trim:** if FALSE, logical, numeric, and complex values are right-justified to a common width: if TRUE the leading blanks for justification are suppressed., **nsmall:** minimum number of digits to the right of the decimal, **scientific:** set to true for scientific notation

Examples

```
> a = "Hello";
> b = "World.";
> c = "This is R."
> paste(a, b, c);
[1] "Hello World. This is R."
```

```
> paste("apple", "orange", "pineapple", sep = " or ");
[1] "apple or orange or pineapple"
```

```
> x = 15.892;
> # show 3 digits after rounding
> format(x, digits = 3);
[1] "15.9"
> # 6 digits after decimal
> format(x, nsmall = 6);
[1] "15.892000"
> # fit the result in 10 cells
> format(x, width = 10, justify = "left");
[1] "    15.892"
> zz = data.frame("(row 1)"= c("data1", "y"), check.names = FALSE);
> format(zz);
  (row 1)
1   data1
2       y
> # center align the data in cell
> format(zz, justify = "centre");
  (row 1)
1   data1
2     y
```

Infinity and Not a Number

Aim: To understand the concept of Inf and NaN in R.

Certain operations like dividing by 0, square root of -1 return values like Inf and NaN.

Example

```
> 1 / 0;
[1] Inf
> Inf - Inf;
[1] NaN
> sqrt(-1);
[1] NaN
Warning message:
In sqrt(-1) : NaNs produced
> 1 + sin(NA);
[1] NA
> exp(1/0);
[1] Inf
```

Inf and -Inf are positive and negative infinity whereas NaN means 'Not a Number'. These apply to numeric values and real and imaginary parts of complex values but not to values of integer vectors. Inf and NaN are reserved words in the R language.

Reading CSV Files

Aim: To understand how to read files in R.

Commands

read.csv(file, header)	Reads a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file. **file:** local or absolute path of file or file.choose(), **header:** a logical value indicating whether the file contains the names of the variables as its first line. If missing, the value is determined from the file format: header is set to TRUE if and only if the first row contains one fewer field than the number of columns.
file.choose()	Choose a file interactively

Example

marks.csv

	A	B
1	Student ID	Marks
2	10000	69
3	10001	69
4	10002	71
5	10003	1
6	10004	48
7	10005	91
8	10006	42
9	10007	85
10	10008	40
11	10009	2

```
> marks = read.csv(file.choose(), header=TRUE);
> marks;
   Student.ID Marks
1       10000    69
2       10001    69
3       10002    71
4       10003     1
5       10004    48
6       10005    91
7       10006    42
8       10007    85
9       10008    40
10      10009     2
```

Inbuilt Dataset

Aim: To understand the various inbuilt datasets available in R.

Commands

mtcars	The data was extracted from the 1974 Motor Trend US magazine and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models). It has 32 rows with 11 columns.
iris	This famous (Fisher's or Anderson's) iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica.
ToothGrowth	The response is the length of odontoblasts (cells responsible for tooth growth) in 60 guinea pigs. Each animal received one of three dose levels of vitamin C (0.5, 1, and 2 mg/day) by one of two delivery methods, orange juice or ascorbic acid (a form of vitamin C and coded as VC).

Examples

```
> head(mtcars);
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
> nrow(mtcars);
[1] 32
> ncol(mtcars);
[1] 11
> mtcars["Mazda RX4", "cyl"];
[1] 6
```

```
> head(iris);
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

```
> head(ToothGrowth);
   len supp dose
1  4.2   VC  0.5
2 11.5   VC  0.5
3  7.3   VC  0.5
4  5.8   VC  0.5
5  6.4   VC  0.5
6 10.0   VC  0.5
```

Data In Tables

Aim: To understand how to manipulate, filter, and analyse data tables in R.

Commands

factor(x, labels)	Used to encode a vector as a factor (the terms 'category' and 'enumerated type' are also used for factors). If the argument ordered is TRUE, the factor levels are assumed to be ordered: **x:** vector of data, **labels:** either an optional character vector of labels for the levels (in the same order as levels after removing those in exclude) or a character string of length 1. Duplicated values in labels can be used to map different values of x to the same factor level.
subset(x, subset, select)	Returns subsets of vectors, matrices, or data frames that meet conditions: **subset:** logical expression indicating elements or rows to keep, **select:** expression indicating columns to select from a data frame
summary(data)	A generic function that produces result summaries of the results of various model fitting functions. The function invokes particular methods which depend on the class of the first argument.
table(data)	Uses the cross-classifying factors to build a contingency table of the counts at each combination of factor levels.

Examples

Consider the vector "employees".

```
> employees;
  id    name gender
1  1    John      0
2  2    Alan      0
3  3    Jane      1
4  4    Mike      0
5  5 Allison      1
```

We can convert the gender which is currently 0 for male and 1 for female using the factor function.

```
> employees$gender = factor(employees$gender, labels=c("male", "female"));
> employees;
  id    name gender
1  1    John   male
2  2    Alan   male
3  3    Jane female
4  4    Mike   male
5  5 Allison female
```

We can get only those employees who are males using the subset function.

```
> males = subset(employees, employees$gender=="male");
> males;
  id name gender
1  1 John   male
2  2 Alan   male
4  4 Mike   male
```

We can use the summary function to get a statistical analysis of the employees.

```
> summary(employees);
       id         name               gender
 Min.   :1   Length:5           male  :3
 1st Qu.:2   Class :character   female:2
 Median :3   Mode  :character
 Mean   :3
 3rd Qu.:4
 Max.   :5
```

To get a static count of the different genders in employees we can use the table function.

```
> genderCount = table(employees$gender);
> genderCount;

  male female
     3      2
```

Plotting in R

Aim: To understand various plotting techniques in R.

Commands

plot(data, type, main, sub, xlab, ylab, col)	Generic plot function which is a placeholder for other plotting functions like line, bar, pie, etc. **data:** the data to plot, **type:** the type of plot: p = points, l = line, b = both points and lines, o = overplotted, h = histogram, s = stair steps, n = no plotting, **main:** title of the plot, **sub:** subtitle of the plot, **xlab:** title for the x-axis, **ylab:** title for the y-axis, **col:** color of the plot
pie(data, labels, . . .)	Creates a pie chart. **data:** a vector of non-negative numerical quantities. The values in x are displayed as the areas of pie slices, **labels:** one or more expressions or character strings giving names for the slices.
barplot(height, . . .)	Creates a bar plot with vertical or horizontal bars. **height:** vector or matrices of the height of each bar
boxplot(formula, data, subset, . . .)	Creates a box-and-whisker plot(s) of the given (grouped) values. **formula:** a formula, such as y ~ grp, where y is a numeric vector of data values to be split into groups according to the grouping variable grp (usually a factor), **data:** a data frame (or list) from which the variables in the formula should be taken, **subset:** an optional vector specifying a subset of observations to be used for plotting
hist(data, . . .)	Computes a histogram of the given data values.
Axes and Text	
title(main, sub, xlab, ylab)	Sets the main title, subtitle, x-axis title, and y-axis title
text(location, text, pos, . . .)	Adds plaintext to a plot. **location:** location can be an x,y coordinate. Alternatively, the text can be placed interactively via mouse by specifying location as locator(1), **text:** the text to be placed, **pos:** position relative to location. 1=below, 2=left, 3=above, 4=right. If you specify pos, you can specify offset= in percent of character width
axis(side, at, labels, col, . . .)	Sets custom axes for the plot. **side:** an integer indicating the side of the graph to draw the axis (1=bottom, 2=left, 3=top, 4=right), **at:** a numeric vector indicating where tic marks should be drawn, **labels:** a character vector of labels to be placed at the tickmarks (if NULL, the at values will be used), **col:** color of the axis

legend(location, title, . . .)	Adds a legend to the plot/graph.

Examples

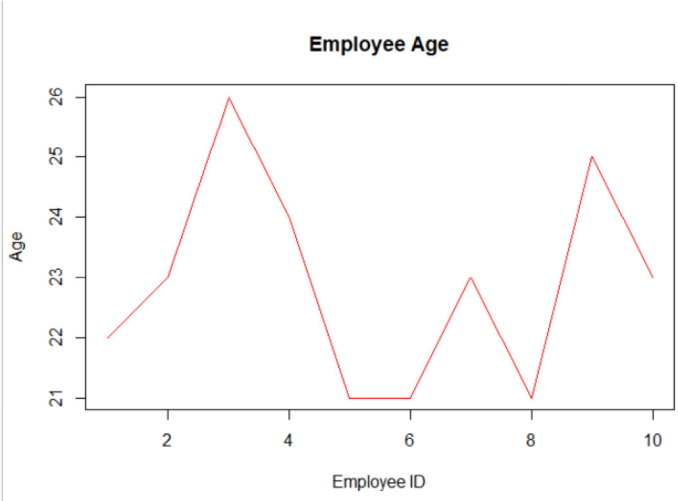
Let us graphically represent the following data in various ways:

```
> emp_info;
   id age gender   role
1   1  22   Male Intern
2   2  23   Male Intern
3   3  26 Female Junior
4   4  24 Female Junior
5   5  21 Female Senior
6   6  21   Male Senior
7   7  23   Male Junior
8   8  21   Male Senior
9   9  25   Male Junior
10 10  23   Male Intern
```

```
> summary(emp_info);
       id              age           gender      role
 Min.   : 1.00   Min.   :21.00   Male  :7   Intern:3
 1st Qu.: 3.25   1st Qu.:21.25   Female:3   Junior:4
 Median : 5.50   Median :23.00              Senior:3
 Mean   : 5.50   Mean   :22.90
 3rd Qu.: 7.75   3rd Qu.:23.75
 Max.   :10.00   Max.   :26.00
```

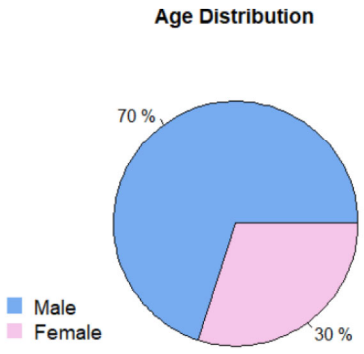
Line plot

```
> emp_info;
   id age gender   role
1   1  22   Male Intern
2   2  23   Male Intern
3   3  26 Female Junior
4   4  24 Female Junior
5   5  21 Female Senior
6   6  21   Male Senior
7   7  23   Male Junior
8   8  21   Male Senior
9   9  25   Male Junior
10 10  23   Male Intern
> plot(emp_info$age, type = "l", main = "Employee Age", xlab = "Employee ID", ylab = "Age", col =
 'red');
```



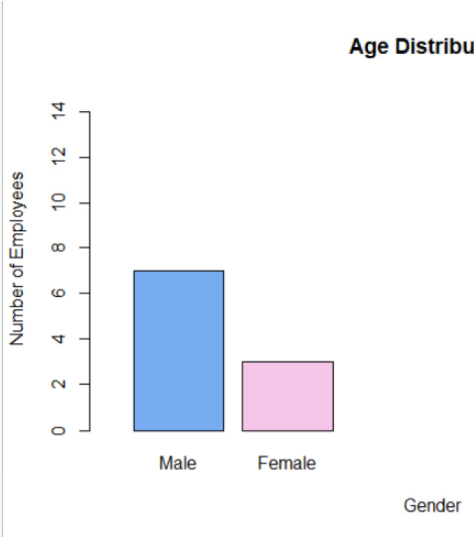
Pie Chart

```
> genders = table(emp_info$gender);
> genders;

  Male Female
     7      3
> total = sum(genders);
> percentage = 100 * genders / total;
> pie(genders, main = "Age Distribution", labels = paste(percentage, '%'), col = c('#77ACF1', '#F6C
6EA'));
> legend("bottomleft",
+        legend = c("Male", "Female"),
+        col = c('#77ACF1', '#F6C6EA'),
+        pch = c(15, 15),
+        bty = "n",
+        pt.cex = 2,
+        cex = 1.2,
+        text.col = "black",
+        horiz = F ,
+        inset = c(0.1, 0.1));
```



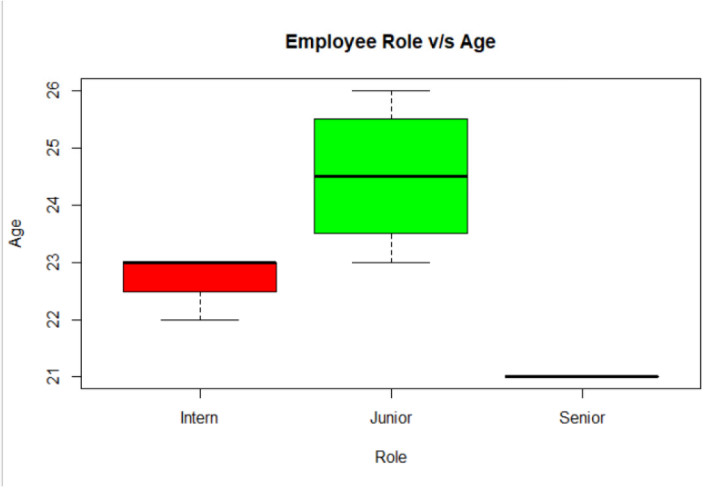
Bar Plot

```
> barplot(genders,
+         xlim = c(0, 7),
+         ylim = c(0, 15),
+         col = c('#77ACF1', '#F6C6EA'),
+         main = "Age Distribution",
+         xlab = "Gender",
+         ylab = "Number of Employees",
+         axes = T);
```



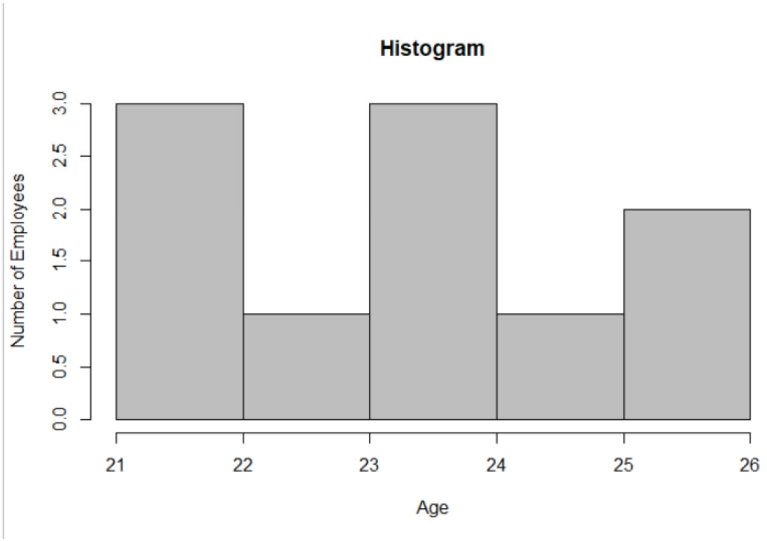
Box Plot

```
> boxplot(emp_info$age ~ emp_info$role,
+         col = c("red", "green", "blue"),
+         main = "Employee Role v/s Age",
+         xlab = "Role",
+         ylab = "Age");
```



Histogram

```
> hist(emp_info$age,
+      main = "Histogram",
+      xlab = "Age",
+      ylab = "Number of Employees", col = 'grey', right = F);
```


Histogram
Number of Employees
0.0
0.5
1.0
1.5
2.0
2.5
3.0
21
22
23
24
25
26
Age

Probability

Aim: To understand various commands related to probability and sample space in R.

Commands

sample(x, n, size)	Takes a sample of the specified size from the elements of x using either with or without replacement. **x:** either a vector of one or more elements from which to choose or a positive integer, **n:** a positive number, the number of items to choose from, **size:** a non-negative integer giving the number of items to choose
outer(X, Y, FUN, . . .)	The outer product of the arrays X and Y is the array A with dimension c(dim(X), dim(Y)) where element A[c(arrayindex.x, arrayindex.y)] = FUN(X[arrayindex.x], Y[arrayindex.y], ...). **X, Y:** First and second arguments for function FUN. Typically a vector or array, **FUN:** a function to use on the outer products, found via match.fun
choose(n, k)	Returns binomial coefficients of its absolute values. It is defined for all real numbers n and integer k. For k ≥ 1 it is defined as n(n-1)...(n-k+1) / k!, as 1 for k = 0 and as 0 for negative k. **n:** an integer **k:** an integer
factorial(x)	Returns the factorial for a non-negative integer
library(prob) tosscoin(times, makespace)	Sets up a sample space for the experiment of tossing a coin repeatedly with the outcomes "H" or "T". **times:** number of times to toss, **makespace:** if TRUE it shows the probability of each case
rolldie(times, nsides, makespace)	Sets up a sample space for the experiment of rolling a die repeatedly. **times:** number of times to toss, **nsides:** number of sides of the die, **makespace:** if TRUE it shows the probability of each case

Examples

```
> # Select 5 numbers from 1-100
> sample(1:100, 5);
[1] 54 71 19 77 21
```

```
> # Select 10 outcomes from 0-1
> # replace = T is suitable for modelling
> # coin tosses or throws of a die
> sample(0:1, 10, replace = T);
 [1] 0 1 1 1 1 1 0 0 1 1
```

```
> # Roll 2 dice
> dice = as.vector(outer(1:6, 1:6, paste));
> dice;
 [1] "1 1" "2 1" "3 1" "4 1" "5 1" "6 1" "1 2" "2 2" "3 2" "4 2" "5 2" "6 2" "1 3" "2 3"
[15] "3 3" "4 3" "5 3" "6 3" "1 4" "2 4" "3 4" "4 4" "5 4" "6 4" "1 5" "2 5" "3 5" "4 5"
[29] "5 5" "6 5" "1 6" "2 6" "3 6" "4 6" "5 6" "6 6"
```

```
> # Product of face values when rolling 2 dice
> faceProd = as.vector(outer(1:6, 1:6));
> faceProd;
 [1]  1  2  3  4  5  6  2  4  6  8 10 12  3  6  9 12 15 18  4  8 12 16 20 24  5 10 15 20
[29] 25 30  6 12 18 24 30 36
```

```
> # Toss a unbiased coin 10 times
> sample(c("H", "T"), 10, replace = T);
 [1] "H" "H" "H" "H" "H" "H" "T" "H" "T" "T"
```

```
> # Toss a biased coin 10 times
> sample(c("H", "T"), 10, replace = T, prob = c(0.9, 0.1));
 [1] "H" "H" "H" "H" "T" "H" "H" "H" "H" "H"
```

```
> # Combination 5C2
> choose(5, 2);
[1] 10
```

```
> # Permutation 5P2
> n = 5;
> r = 2;
> p = factorial(n) / factorial(r);
> p;
[1] 60
```

```
> # Binomial coefficients
> choose(5, 0:5);
[1]  1  5 10 10  5  1
```

```
> # Pascal's Triangle
> N = 10;
> for (i in 0:(N-1)) {
+   s = "";
+   for(k in 0:(N-i)) s = paste(s, "  ", sep="");
+   for(j in 0:i) {
+     s = paste(s, sprintf("%3d ", choose(i, j)), sep="");
+   }
+   print(s);
+ }
[1] "                        1 "
[1] "                      1   1 "
[1] "                    1   2   1 "
[1] "                  1   3   3   1 "
[1] "                1   4   6   4   1 "
[1] "              1   5  10  10   5   1 "
[1] "            1   6  15  20  15   6   1 "
[1] "          1   7  21  35  35  21   7   1 "
[1] "        1   8  28  56  70  56  28   8   1 "
[1] "      1   9  36  84 126 126  84  36   9   1 "
```

```
> # Tossing n coins without probability
> library(prob);
> tosscoin(2);
  toss1 toss2
1     H     H
2     T     H
3     H     T
4     T     T
> # With probability
> tosscoin(2, makespace = TRUE);
  toss1 toss2 probs
1     H     H  0.25
2     T     H  0.25
3     H     T  0.25
4     T     T  0.25
```

```
> # Rolling 2 3-sided dice
> rolldie(2, nsides = 3);
  X1 X2
1  1  1
2  2  1
3  3  1
4  1  2
5  2  2
6  3  2
7  1  3
8  2  3
9  3  3
```

```
> # Expectation, Variance and Standard Deviation
> x =  c(   0,     1,     2, 3);
> px = c(0.25, 0.125, 0.125, 0.5);
> expect = sum(x * px);
> expect;
[1] 1.875
> variance = sum((x ^ 2 * px)) - (expect ^ 2);
> variance;
[1] 1.609375
> stdDev = sqrt(variance)
> stdDev;
[1] 1.268611
```

Binomial Probability Distribution

Aim: To understand various commands related to a binomial probability distribution in R.

Commands

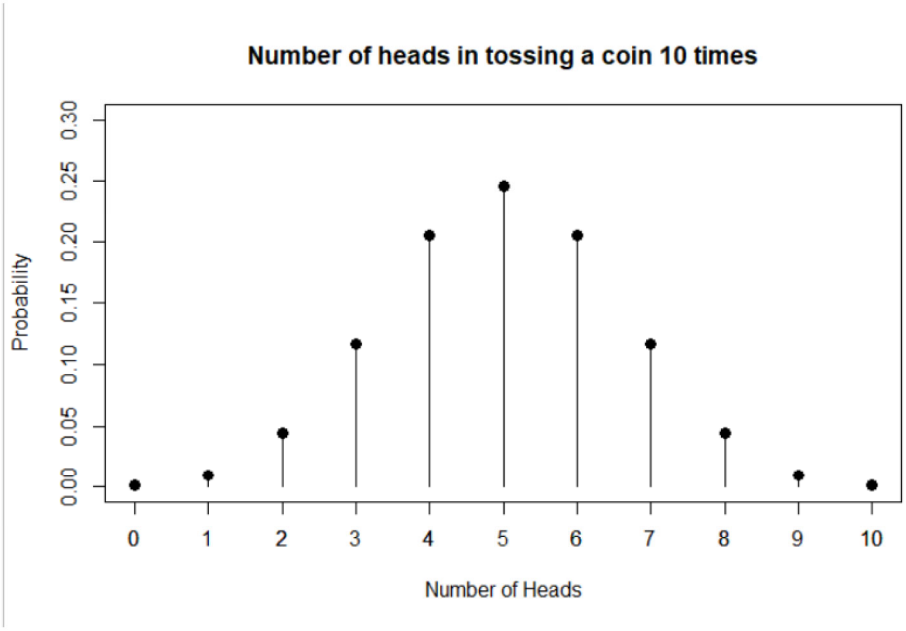
dbinom(x, size, prob)	Returns the binomial distribution probability of x in n number of observations where prob is the probability. **x:** vector of quantiles, **size:** number of observations, **prob:** probability of success on each trial.
pbinom(q, size, prob, lower.tail)	Returns the c.d.f of the binomial distribution. **q:** vector of quantiles, **size:** number of observations, **prob:** probability of success on each trial, **lower.tail:** logical; if TRUE (default), probabilities are P[X ≤ x], otherwise, P[X > x]
qbinom(p, size, prob, lower.tail)	Calculates the "inverse c. d. f." of the binomial distribution. It gives the quantile function. The quantile is defined as the smallest value x such that F(x) >= p, where F is the distribution function. **p:** vector of probabilities
rbinom(n, size, prob)	Generates a required number of random values of given probability from a given sample.

Examples

```
> # Find prob of getting 3 twos in rolling a die 5 times.
> # n = 5, x = 3, p = 1/6
> dbinom(3, size = 5, prob = 1 / 6);
[1] 0.03215021
```

Plotting binomial distribution

```
> # Plotting binomial distribution
> # Prob of number of heads in tossing a coin 10 times
> n = 10;
> p = 1 / 2;
> x = 0:n;
> probs = dbinom(x, size = n, prob = p);
> plot(x, probs, type = 'h',
+      ylim = c(0, 0.3),
+      xlab = "Number of Heads",
+      ylab = "Probability",
+      main = "Number of heads in tossing a coin 10 times");
> axis(1, x);
> points(x, probs, pch = 16, cex = 1.25);
```




```
> # Prob of getting at most 2 heads in tossing a coin 5 times.
> # n = 5, x <= 1, p = 1 / 2
> pbinom(q = 2, size = 5, prob = 1 / 2);
[1] 0.5
```

```
> # Find the 10th quantile of a binomial distribution
> # with 10 trials and prob of success on each trial = 0.4
> qbinom(0.10, size = 10, prob = 0.4);
[1] 2
```

```
> # Generate a vector that shows the number
> # of successes of 10 binomial experiments with
> # 100 trials where the probability of success
> # on each trial is 0.3.
> results <- rbinom(10, size = 100, prob = 0.3);
> results;
 [1] 27 34 33 33 29 33 26 27 39 31
```

Poisson Probability Distribution

Aim: To understand various commands related to Poisson probability distribution in R.

Commands

dpois(x, lambda)	Returns the Poisson distribution probability of x with lambda as mean. **x:** vector of (non-negative integer) quantiles, **lambda:** vector of (non-negative) means
ppois(q, lamda, lower.tail)	Finds the probability that a certain number of successes or less occur based on an average rate of success. **q:** vector of quantiles, **lower.tail:** logical; if TRUE (default), probabilities are P[X ≤ x], otherwise, P[X > x]
qpois(p, lambda, lower.tail)	Finds the number of successes that corresponds to a certain percentile based on an average rate of success. **p:** percentile
rpois(n, lambda)	Generates a list of random variables that follow a Poisson distribution with a certain average rate of success: **n:** number of random variables to generate

Examples

```
> # A shop makes 100 sales per hour.
> # In a given hour, find prob that
> # the shop makes exactly 80 sales.
> dpois(x = 80, lambda = 100);
[1] 0.005197854
```

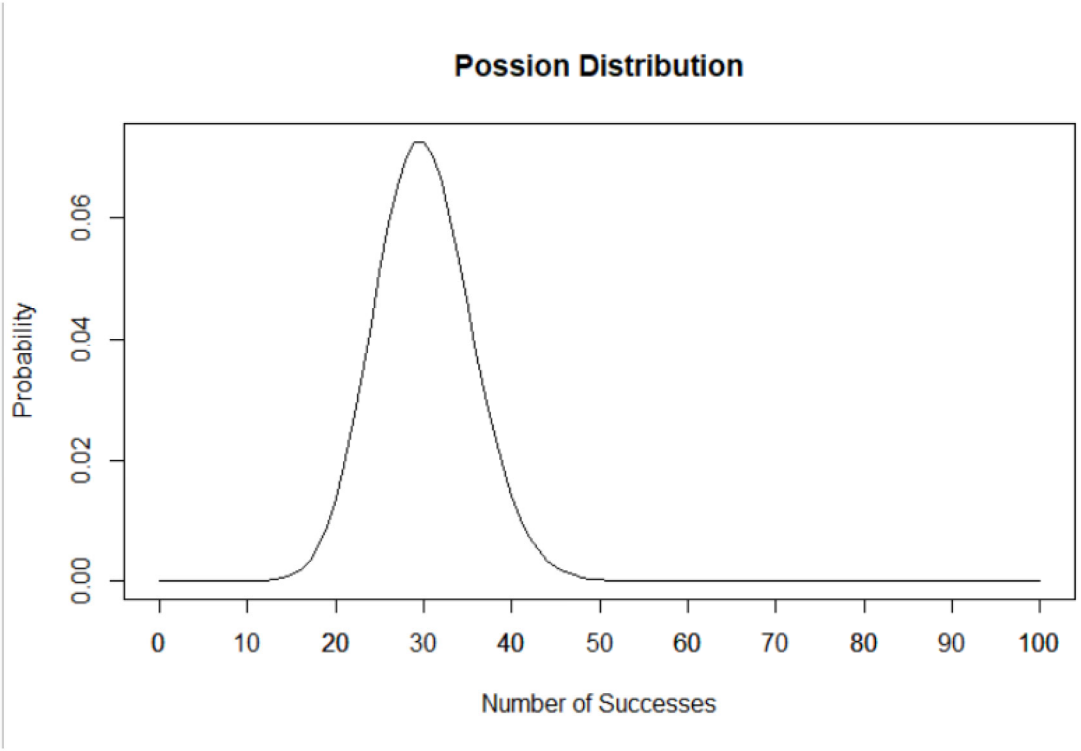
```
> # A shop makes 100 sales per hour.
> # In a given hour, find prob that
> # the shop makes at least 80 sales.
> ppois(q = 80, lambda = 100);
[1] 0.02264918
```

```
> # A shop makes 100 sales per hour.
> # Find the number of sales the shop needs
> # to make to be at the 90th percentile
> # for sales in an hour.
> qpois(p = 0.9, lambda = 100);
[1] 113
```

```
> # Generate a list of 10 random variables
> # that follow a Poisson distribution
> # with a rate of success equal to 7.
> rpois(n = 10, lambda = 7);
 [1] 2 4 4 9 8 4 8 7 7 1
```

Plotting Poisson distribution

```
> # Plot a poisson distribution for a
> # range of success from 0-100 with
> # average number of successes = 30.
> x = 0:100;
> px = dpois(x, lambda = 30);
> plot(x, px, type = "l");
> axis(1, seq(0, 100, 10));
> plot(x, px, type = "l",
+      xlab = "Number of Successes",
+      ylab = "Probability",
+      main = "Possion Distribution");
> axis(1, seq(0, 100, 10));
```



Normal Probability Distribution

Aim: To understand various commands related to a Normal probability distribution in R.

Commands

dnorm(x, mean, sd)	Returns the value of the probability density function of the normal distribution given a certain random variable x, a population mean, and population standard deviation. **x:** vector of quantiles, **mean:** vector of means, **sd:** vector of standard deviations
pnorm(q, mean, sd, lower.tail)	Returns the value of the cumulative density function of the normal distribution given a certain random variable q, a population mean, and population standard deviation. **q:** vector of quantiles, **lower.tail:** logical; if TRUE (default), probabilities are P[X ≤ x] otherwise, P[X > x].
qnorm(p, mean, sd, lower.tail)	Returns the value of the inverse cumulative density function of the normal distribution given a certain random variable p, a population mean, and population standard deviation. **p:** vector of probabilities
rnorm(n, mean, sd)	Generates a vector of normally distributed random variables given a vector length n, a population mean, and population standard deviation. **n:** number of observations

Examples

```
> # Find the value of the standard normal
> # distribution at x = 0
> dnorm(x = 0, mean = 0, sd = 1);
[1] 0.3989423
```

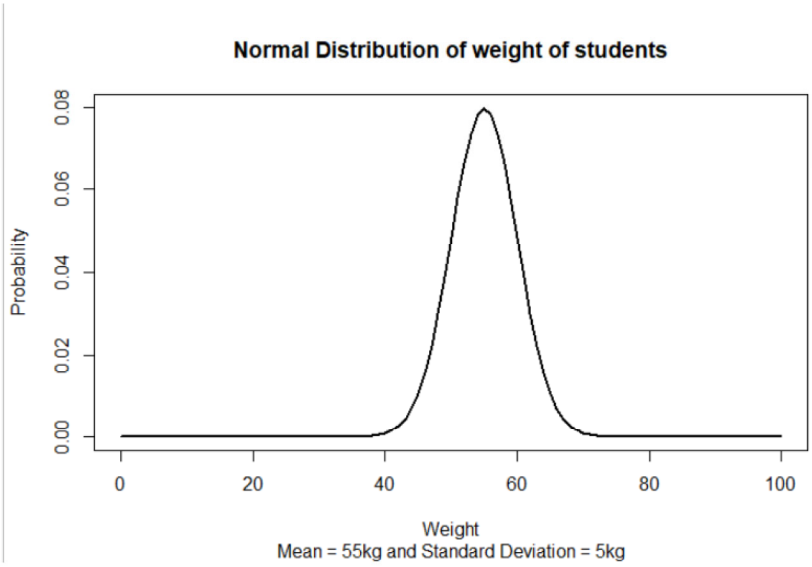
```
> # Find the percentage of students who weigh
> # more than 70kg in a population with mean = 55kg
> # and standard deviation = 5kg
> pnorm(70, mean = 66, sd = 5, lower.tail = F);
[1] 0.2118554
```

```
> # Find the Z-score of the 95th quantile of the
> # standard normal distribution.
> qnorm(0.95);
[1] 1.644854
```

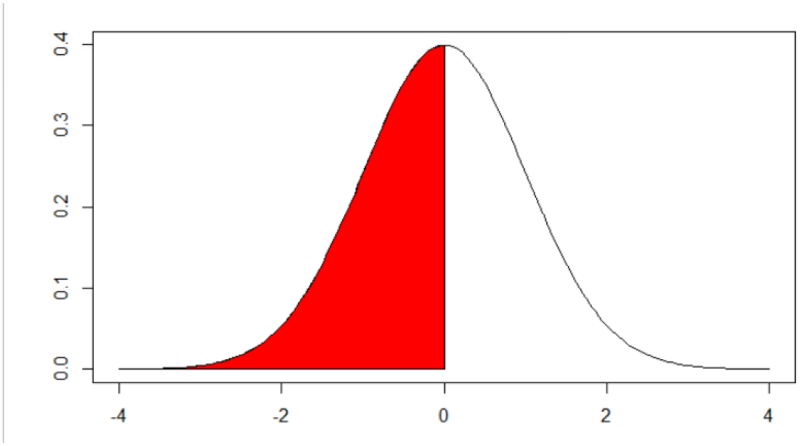
```
> # Find weights of 10 students if mean = 55 and
> # standard deviation = 5, given weight is normally
> # distributed.
> round(rnorm(10, mean = 55, sd = 5), 2);
 [1] 53.92 57.17 49.56 53.74 46.24 53.34 50.74 57.33 63.74 53.34
```

Plotting normal distribution

```
> # Plotting normal distribution
> # mean = 55, standard deviation = 5
> x = 0:100;
> px = dnorm(x, mean = 55, sd = 5);
> plot(x, px, type = "l", lwd = 2,
+      xlab = "Weight", ylab = "Probability",
+      main = "Normal Distribution of weight of students",
+      sub = "Mean = 55kg and Standard Deviation = 5kg");
```




```
> # Find area of curve to left of mean
> # of the standard normal distribution
> x = seq(-4, 4, length = 200);
> y = dnorm(x);
> plot(x,y, type='l', xlab = "", ylab = "");
> x1 = seq(-4, 0, length = 100);
> y1 = dnorm(x1);
> polygon(c(-4, x1, 0), c(0, y1, 0), col = 'red');
> pnorm(0);
[1] 0.5
```


0.4
0.3
0.2
0.1
0.0
-4
-2
0
2
4

Correlation

Aim: To understand various commands and techniques related to correlation in R.

Commands

var(x)	Computes the variance of x.
cor(x, y, method)	Computes the correlation between x and y. **method:** a character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman".
cov(x, y, method)	Computes the covariance between x and y.
cor.test(x, y, method)	Test for the association between paired samples, using one of Pearson's product moment correlation coefficient, Kendall's tau or Spearman's rho.

Examples

Covariance using Karl Pearson's formula

```
> x = c(15, 25, 35, 45, 55, 65);
> y = c(302.38, 193.63, 185.46, 198.49, 224.30, 288.71);
> # Correlation using Karl Pearson Formula
> cov(x, y) / sqrt(var(x) * var(y));
[1] 0.03847689
> # Correlation using inbuilt R function
> cor(x,y);
[1] 0.03847689
```

```
> # Correlation using Spearman Formula
> x = c(15, 25, 35, 45, 55, 65);
> y = c(302.38, 193.63, 185.46, 198.49, 224.30, 288.71);
> x1 = rank(x);
> x2 = rank(y);
> d = x2 - x1;
> di = d ^ 2;
> r = 1 - (6 * sum(di)) / (6 * (36 - 1));
> r;
[1] 0.08571429
> # Correlation using inbuilt R function
> cor(x, y, method = 'spearman');
[1] 0.08571429
```

```
> # Optimized Spearman correlation Formula
> x = c(15, 25, 35, 45, 55, 65);
> y = c(302.38, 193.63, 185.46, 198.49, 224.30, 288.71);
> x1 = rank(x);
> y1 = rank(y);
> sum1 = sum(table(x1) ^ 3 - table(x1));
> sum2 = sum(table(y1) ^ 3 - table(y1));
> sum3 = (sum1 + sum2) / 6;
> d = y1 - x1;
> di = d ^ 2;
> r = 1 - (6 *(sum(di)) + sum3) / (6 * (36 - 1));
> r;
[1] 0.08571429
> cor(x, y, method = 'spearman');
[1] 0.08571429
```

```
> # Testing correlation between 2 samples
> x = c(15, 25, 35, 45, 55, 65);
> y = c(302.38, 193.63, 185.46, 198.49, 224.30, 288.71);
> cor.test(x, y);

        Pearson's product-moment correlation

data:  x and y
t = 0.077011, df = 4, p-value = 0.9423
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.7980031  0.8242983
sample estimates:
       cor
0.03847689

> cor.test(x, y, method = "spearman");

        Spearman's rank correlation rho

data:  x and y
S = 32, p-value = 0.9194
alternative hypothesis: true rho is not equal to 0
sample estimates:
       rho
0.08571429

> cor.test(x, y, method = "kendall");

        Kendall's rank correlation tau

data:  x and y
T = 9, p-value = 0.7194
alternative hypothesis: true tau is not equal to 0
sample estimates:
tau
0.2
```

Linear Regression

Aim: To understand various commands related to linear regression in R.

Commands

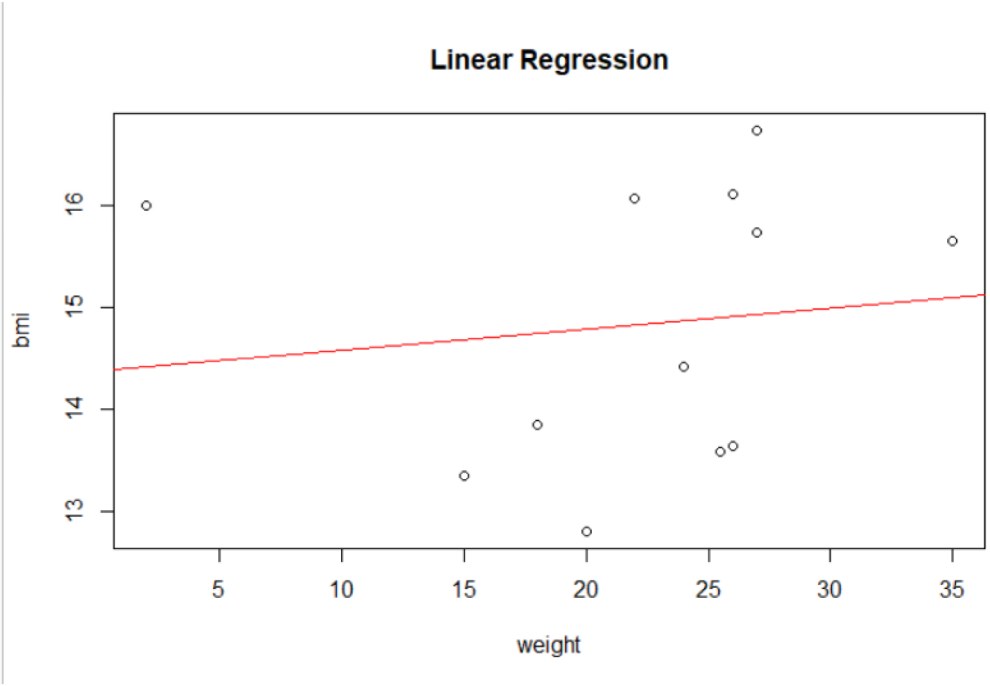
lm(formula, data, . . .)	Used to fit linear models. It can be used to carry out regression, single stratum analysis of variance and analysis of covariance: **formula:** a symbolic description of the model to be fitted, **data:** an optional data frame, list or environment containing the variables in the model
predict.lm(object, newdata)	Predicted values based on linear model object: **object:** an object of class lm, **newdata:** an optional data frame in which to look for variables with which to predict.
coef(model)	A generic function which extracts model coefficients from objects returned by modeling functions. coefficients is an alias for it.

Examples

```
> height = c(175,168,170,171,169,165,165,160,180,186);
> weight = c(80,68,72,75,70,65,62,60,85,90);
> model = lm(height~weight);
> print(model3);

Call:
lm(formula = height ~ weight2)

Coefficients:
(Intercept)      weight2
   115.2002       0.7662

> # predict the height of a person with weight = 50kg
> predict(model, data.frame(weight = 50));
       1
153.5082
```

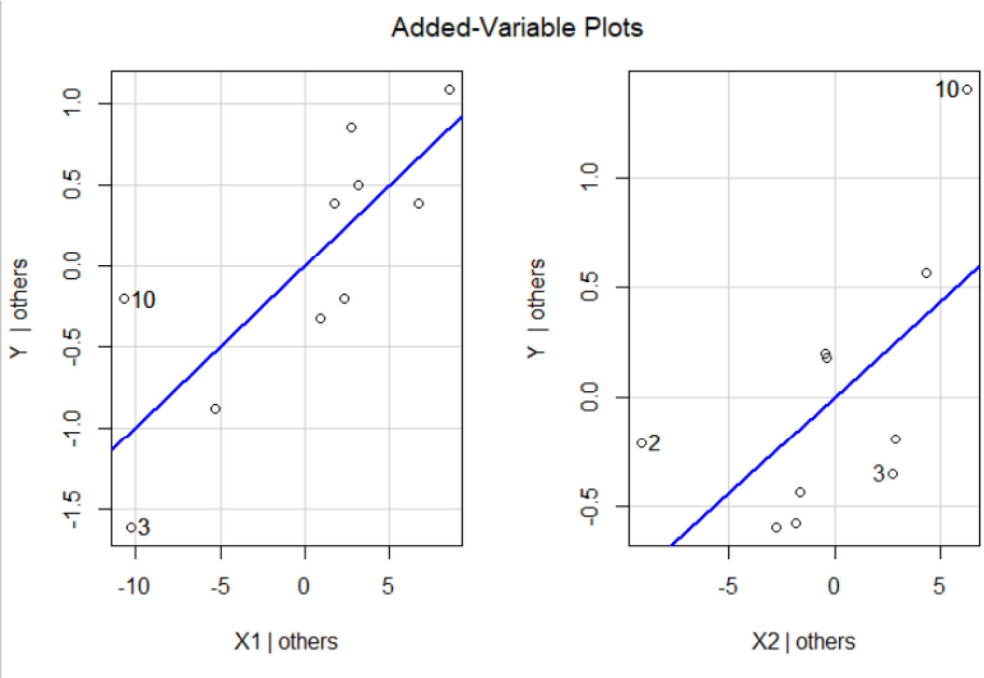
```
> # The BMI depends on weight linearly
> weight = c(15,26,27,2,25.5,27,35,18,22,20,26,24);
> bmi = c(13.35,16.12,16.74,16.00,13.59,15.73,15.65,13.85,16.07,12.80,13.65,14.42);
> model <- lm(bmi~weight);
> summary.lm(model);

Call:
lm(formula = bmi ~ weight)

Residuals:
     Min       1Q   Median       3Q      Max
-1.98431 -1.26858  0.05782  1.22168  1.81358

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 14.37825    1.22506  11.737  3.6e-07 ***
weight       0.02030    0.05185   0.392    0.704
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.406 on 10 degrees of freedom
Multiple R-squared:  0.0151,    Adjusted R-squared:  -0.08339
F-statistic: 0.1533 on 1 and 10 DF,  p-value: 0.7036

> plot(weight, bmi, main = "Linear Regression");
> abline(model, col = 'red');
```

Here we see that, bmi = 0.02030 * weight + 14.37825.



Multiple Linear Regression

```
> # multiple linear regression
> # Y depends on X1 and on X2
> Y = c(1, 2, 1, 3, 2, 3, 3, 4, 4, 3);
> X1 = c(40, 45, 38, 50, 48, 55, 53, 55, 58,40);
> X2 = c(25, 20, 30, 30, 28, 30, 34, 36, 32, 34);
> input = data.frame(Y, X1, X2);
> reg_model = lm(formula = Y ~ X1+X2, data = input);
> summary(reg_model);

Call:
lm(formula = Y ~ X1 + X2, data = input)

Residuals:
     Min       1Q   Median       3Q      Max
-0.59080 -0.39823 -0.05028  0.23136  0.85910

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -4.83038    1.45112  -3.329  0.01261 *
X1           0.09980    0.02792   3.574  0.00905 **
X2           0.08763    0.04242   2.066  0.07769 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5526 on 7 degrees of freedom
Multiple R-squared:  0.7945,    Adjusted R-squared:  0.7357
F-statistic: 13.53 on 2 and 7 DF,  p-value: 0.003937

> library(car);
> avPlots(reg_model);
```



From the plot we see that the slope of the lines for both the plots is positive which matches with the coefficients from the summary of the model.

Hence Y = 0.0998 * X1 + 0.0876 * X2 – 4.8303

Testing Hypothesis (Z Test)

Aim: To understand various commands and techniques related to testing hypothesis in R.

Theory

Test for significance of single mean

Test Statistic: $Z = \frac{\bar{x} - \mu}{\sigma/\sqrt{n}}$

Test for significance of population proportion

Test Statistic: $Z = \frac{\mathrm{p} - \mathrm{p_0}}{\sqrt{\mathrm{p_0 q_0}/n}}$

Type of Test	Null Hypothesis	Alternate Hypothesis	Reject Null Hypothesis when
Two Tail	$\mu = \mu_0$	$\mu \neq \mu_0$	$\|z\| > \|z_a\|$
Right Tail	$\mu \geq \mu_0$	$\mu < \mu_0$	$z \geq z_a$
Left Tail	$\mu \leq \mu_0$	$\mu > \mu_0$	$z \leq -z_a$

Examples

Left tail test

```
> # A company claims that mean lifetime of its product
> # is more than 10000 hrs. In a sample of 30 products
> # it is found that they only last 9000 hrs on average.
> # Assume population standard deviation is 120 hrs.
> # At 5% significance can we reject the claim by the company?
>
> # Null hypothesis: u > 10000
> # Alternate hypothesis: u <= 10000
>
> xbar = 9900;
> u = 10000;
> sd = 120;
> n = 30;
> z = (xbar - u) / (sd / sqrt(n));
> round(z, 3);
[1] -4.564
>
> alpha = 0.05;
> za = round(qnorm(1-alpha), 3);
> za;
[1] 1.645
>
> # Since -4.564 is not in (-1.645, 1.645) null hypothesis is
> # rejected at 5% significance.
>
> pval = pnorm(z);
> pval;
[1] 2.505166e-06
>
> # Here also, since lower tail pvalue is less than significance
> # level 0.05, we reject the null hypothesis that mean lifetime is
> # more than 10000 hrs.
```

Right tail test

```
> # A food brand advertises that there is at most 2g of fats
> # in a cookie. In a sample of 35 cookies, it is found that the
> # mean amount of fats is 2.1g per cookie. Assume the population
> # standard deviation 0.25g. At 5% significance, can we reject the
> # claim of the food brand?
>
> # Null hypothesis: u <= 2
> # Alternate hypothesis: u > 2
>
> xbar = 2.1;
> u = 2;
> sd = 0.25;
> n = 35;
> z = (xbar - u) / (sd / sqrt(n));
> round(z, 3);
[1] 2.366
>
> alpha = 0.05;
> za = round(qnorm(1-alpha), 3);
> za;
[1] 1.645
>
> # Since 2.366 is not in (-1.645, 1.645) null hypothesis is
> # rejected at 5% significance.
>
> pval = pnorm(z);
> 1-pval;
[1] 0.008980239
>
> # Here also, since upper tail pvalue is less than significance
> # level 0.05, we reject the null hypothesis that amount of
> # fat is less than 2g.
```

Two tail test

```
> # The mean lifetime of 100 bulbs is found to be 1580 hrs with
> # standard deviation of 90 hrs. Test the hypothesis at 1% level
> # of significance that the mean lifetime of the bulbs is 1600 hrs.
>
> # Null hypothesis: u = 1600
> # Alternate hypothesis: u != 1600
>
> xbar = 1600;
> u = 1580;
> sd = 90;
> n = 100;
> z = (xbar - u) / (sd / sqrt(n));
> round(z, 3);
[1] 2.222
>
> alpha = 0.01;
> za = round(qnorm(1-alpha/2), 3);
> za;
[1] 2.576
>
> # Since 2.222 is in (-2.576, 2.576) null hypothesis is
> # accepted at 1% level of significance.
>
> pval = 2*pnorm(z);
> pval;
[1] 1.973732
>
> # Here also, since pvalue is more than significance
> # level 0.01, we accept the null hypothesis that lifetime of bulb is
> # 1600 hrs.
```

Test for population proportion

```
> # Suppose 60% of citizens voted in last election. 85 out of 148 people
> # in a telephone survey said that they voted in current election. At
> # 5% level of significance, can we reject the null hypothesis that the
> # proportion of voters in the population is above 60% this year?
>
> # Null hypothesis: p > 60/100
> # Alternate hypothesis: p <= 60/100
>
> p = 85/148;
> p0 = 60/100;
> n = 148;
> q0 = 1-p0;
> z = (p-p0)/sqrt((p0*q0)/n);
> round(z, 3);
[1] -0.638
> alpha = 0.05;
> za = round(qnorm(1-alpha), 3);
> za;
[1] 1.645
>
> # Since -0.638 is in (-1.645, 1.645) null hypothesis is
> # accepted at 5% level of significance.
>
> pval = pnorm(z);
> pval;
[1] 0.2618676
>
> # Here also, since pvalue is more than significance
> # level 0.05, hence we accept the null hypothesis that proportion of
> # voters is above 60%.
```

THE END