R is a programming language and software environment for statistical computing and graphics. The R language is widely used among statisticians and data miners for developing statistical software and data analysis.

R commands, case sensitivity

Technically R is an expression language with a very simple syntax. It is **case sensitive**, so A and a are different symbols and would refer to different variables.

Elementary commands consist of either expressions or assignments. If an expression is given as a command, it is evaluated, printed (unless specifically made invisible), and the value is lost.

An assignment also evaluates an expression and passes the value to a variable but the result is not automatically printed.

Commands are separated either by a semi-colon (';'), or by a newline. Elementary commands can be grouped together into one compound expression by braces ('{' and '}').

Comments can be put almost anywhere, starting with a hashmark ('#'), everything to the end of the line is a comment.

If a command is not complete at the end of a line, R will give a different prompt, by default + on second and subsequent lines and continue to read input until the command is syntactically complete.

Assignment:

An assignment means naming a value, so that it can be used later. Assignment has general form

Variable = expression or value (= is an assignment operator)

```
> x = 2 + 3   # x is assigned value 5
> x
[1] 5
> x + 2
[1] 7
> x = x * 3
> x
[1] 15
> x = 2 + 3; y = -4; z = x * y      # Commands are separated by a semi-colon (';')
> x; y; z
[1] 5
[1] -4
[1] -20
> x = 2 + 3; y = -4; z = x * y; x; y; z       # Commands are separated by a semi-colon (';')
[1] 5
[1] -4
[1] -20
```

Vectors:

R operates on named data structures. The simplest such structure is the **numeric vector**, which is a single entity consisting of an ordered collection of numbers. To set up a vector named x, say, consisting of five numbers, namely 10.4, 5.6, 3.1, 6.4 and 21.7, use the R command

> x <- c(10.4, 5.6, 3.1, 6.4, 21.7)

To see what numbers is included in x type "x" and press the enter key:

> x

[1] 10.4 5.6 3.1 6.4 21.7

Accessing vectors:

Individual elements of a vector can be accessed by using indices.

> x <- c(10.4, 5.6, 3.1, 6.4, 21.7)

> x[3] # third element of vector x is accessed.

[1] 3.1

> x[1] # first element of vector x is accessed.

[1] 10.4

> x[2 : 4] # elements from second to fourth of vector x are accessed.

[1] 5.6 3.1 6.4

> x[c(2,5)] # elements having indices 2 and 5 are accessed.

[1] 5.6 21.7

> length(x) # displays number of elements in vector x.

[1] 5

> x[3 : length(x)] # elements having indices 3 to 5 of vector x are accessed.

[1] 3.1 6.4 21.7

> x[4 : 2] # elements from fourth to second reversely of vector x are accessed.

[1] 6.4 3.1 5.6

> x[x > 6] # elements of vector x having value > 6 are accessed.

[1] 10.4 6.4 21.7

> x[x < 6] # elements of vector x having value < 6 are accessed.

[1] 5.6 3.1

Subset command can also be used with vectors.

> which (x < 6) # displays index of elements of vector x whose value is < 6.

[1] 2 3

> x[-1] # elements except first are accessed.

[1] 5.6 3.1 6.4 21.7

> x[c(-2,-5)] # elements except second and fifth are accessed or x[-c(2,5)]

[1] 10.4 3.1 6.4

> x[-2 : -4] # elements except second to fourth are accessed.

[1] 10.4 21.7

> x < 6

[1] FALSE TRUE TRUE FALSE FALSE

> 1/x

[1] 0.09615385 0.17857143 0.32258065 0.15625000 0.04608295

> x

[1] 10.4 5.6 3.1 6.4 21.7

> b <- c(x, 0, x)

> b

[1] 10.4 5.6 3.1 6.4 21.7 0.0 10.4 5.6 3.1 6.4 21.7

You can store strings using both single and double quotes.

> t <- c("somaiya", "mumbai", 'new delhi')

> t

[1] "somaiya"   "mumbai"   "new delhi"

<u>Alternative way to create data vectors</u>

Vectors can be created and data can be entered alternatively by using scan function.

> x = scan()

1: 3 -5 7

4: 9 0 6.7

7: -2

8:

Read 7 items

> x

[1] 3.0 -5.0 7.0 9.0 0.0 6.7 -2.0

> y = scan()

1: 2 5 8 4 -2

6: 9 5

8:

Read 7 items

> y

[1] 2 5 8 4 -2 9 5

scan() function has many other arguments such as ***what, nmax*** etc

- ***what***: This argument indicates types of data to be accepted, by default it is numeric. For character data type set what = “character”
- ***nmax***: This argument indicates maximum number of elements to be accepted.

> t = scan(what = "character")

1: "somaiya" "vidyavihar"

3:

Read 2 items

> t

[1] "somaiya" "vidyavihar"

> x = scan(nmax = 4)

1: 5 -8 3 9 2 -11 6

Read 4 items

> x

[1] 5 -8 3 9

Generating regular sequences

R has a number of facilities for generating commonly used sequences of numbers. For example 12:20 is the vector c(12, 13, ..., 20). The colon operator has high priority within an expression, so, for example 2*12:20 is the vector c(24, 26, ..., 40). Put n <- 8 and compare the sequences 1:n-1 and 1:(n-1).

> 12:20

[1] 12 13 14 15 16 17 18 19 20

> p <- 12:20

> p

[1] 12 13 14 15 16 17 18 19 20

> q <- 3*12:20

> q

[1] 36 39 42 45 48 51 54 57 60

> n = 8

> t <- 5:(n-1)

> t

[1] 5 6 7

> w <- 5:n - 1

> w

[1] 4 5 6 7

The construction 20:12 may be used to generate a sequence backwards.

> 20:12

[1] 20 19 18 17 16 15 14 13 12

The function seq() is a more general facility for generating sequences. It has five arguments, only some of which may be specified in any one call. The first two arguments, if given, specify the beginning and end of the sequence, and if these are the only two arguments given the result is the same as the colon operator. That is **seq(12,20)** is the same vector as **12:20**.

Parameters to seq(), and to many other R functions, can also be given in **named form**, in which case the order in which they appear is irrelevant. The first two parameters may be named **from=**value and **to=**value; thus seq(12,20), seq(from=12, to=20) and seq(to=20, from=12) are all the same as 12:20. The next two parameters to seq() may be named

by=value and **length=**value, which specify a step size and a length for the sequence respectively. If neither of these is given, the default by=1 is assumed.

For example

> seq(-5, 5, by=.2) -> s3

> s3

[1] -5.0 -4.8 -4.6 -4.4 -4.2 -4.0 -3.8 -3.6 -3.4 -3.2 -3.0 -2.8 -2.6 -2.4 -2.2

[16] -2.0 -1.8 -1.6 -1.4 -1.2 -1.0 -0.8 -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6 0.8

[31] 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8 3.0 3.2 3.4 3.6 3.8

[46] 4.0 4.2 4.4 4.6 4.8 5.0

Similarly following command generates a sequence of 18 elements

> s4 <- seq(length=18, from=-5, by=.2)

> s4

[1] -5.0 -4.8 -4.6 -4.4 -4.2 -4.0 -3.8 -3.6 -3.4 -3.2 -3.0 -2.8 -2.6 -2.4 -2.2

[16] -2.0 -1.8 -1.6

rep() which can be used for replicating an object in various complicated ways. The simplest form is **s5 <- rep(x, times=5)** which will put five copies of x end-to-end in s5.

> x

[1] 305 16 122 68

> s5 <- rep(x, times=5)

> s5

[1] 305 16 122 68 305 16 122 68 305 16 122 68 305 16 122 68 305 16 122

[20] 68

Another useful version is **s6 <- rep(x, each=5)** which repeats each element of x five times before moving on to the next.

> s6 <- rep(x, each=5)

> s6

[1] 305 305 305 305 305 16 16 16 16 16 122 122 122 122 122 68 68 68 68

[20] 68

> s7 <- rep(1:4,c(2,1,2,1))

> s7

[1] 1 1 2 3 3 4

Some commonly used Built-in functions:

> x <- c(-6,9,0,-3,8,2,-5,4)

> x

[1] -6 9 0 -3 8 2 -5 4

> length(x) #Displays the number of elements of vector x

[1] 8

> max(x) #displays the maximum element of vector x

[1] 9

> min(x) #displays the minimum element of vector x

```
[1] -6
> range(x)                  #displays the range of the values of vector x
[1] -6  9
> sum(x)                    # displays sum of the values of vector x
[1] 9
> cumsum(x)                 # displays the cumulative sum of the values of vector x
[1] -6  3  3  0  8 10  5  9
> mean(x)                   # displays the mean of the values of vector x
[1] 1.125
> median(x)                 # displays the median of the values of vector x
[1] 1
> sort(x)                   # Sort the values of vector x in the increasing order
[1] -6 -5 -3  0  2  4  8  9
> sort(x, decreasing = T)   # Sort the values of vector x in the decreasing order
[1]  9  8  4  2  0 -3 -5 -6
> var(x)                    # Sample variance with denominator (n-1)
[1] 32.125
> which(x == 4)             # displays index of the required element of vector x
[1] 8
> y <- c(3,4,-5)
> prod(y)                   # displays product of the values of vector y
[1] -60
```

Measures of central tendency and dispersion
Descriptive Statistics

Frequency Distribution

Frequency distribution is obtained by using function **table().** It calculates frequency of each distinct observation in a vector of observations. Syntax of the function is table(variable). Suppose, a variable mark contains marks of the students of a particular class then table(mark) gives frequency distribution of marks of students.

```
>       mark <- c(2,2,9,9,5,3,7,1,9,10,5,3,2,3,6,6,2,9,1,8)
> table(mark)
mark
1  2  3  5  6  7  8  9  10
2  4  3  2  2  1  1  4  1
```

In the similar way bivariate frequency table is constructed by using table(). If there are two factors say $X = \{x_1; x_2; ; x_n\}$ and $Y = \{y_1; y_2; ; y_n\}$ then table() counts frequency of pair $\{(x_i; y_j); i; j = 1; 2; ; n\}$. For example,

Construct bivariate frequency distribution for the following 50 pairs $\{(x_i; y_i); i = 1; 2; ; n\}$ where

x : 5; 3; 4; 3; 2; 4; 3; 3; 3; 2; 3; 1; 3; 3; 3; 5; 3; 4; 3; 1; 1; 5; 3; 4; 2;

3; 2; 4; 2; 3; 1; 2; 5; 4; 3; 2; 4; 1; 4; 5; 3; 2; 1; 2; 3; 3; 1; 5; 1; 4
y : 1; 2; 3; 4; 3; 5; 2; 3; 3; 3; 2; 5; 3; 4; 4; 3; 4; 4; 4; 1; 5; 2; 2; 2;
5; 2; 5; 4; 3; 4; 3; 1; 3; 2; 4; 2; 3; 3; 4; 2; 3; 1; 4; 2; 3; 3; 3; 1; 2; 1

Solution:

```
> # Bivariate frequency distribution
> x <- c(5,3,4,3,2,4,3,3,3,2,3,1,3,3,3,5,3,4,3,1,1,5,3,4,2,
+ 3,2,4,2,3,1,2,5,4,3,2,4,1,4,5,3,2,1,2,3,3,1,5,1,4)
> y <- c(1,2,3,4,3,5,2,3,3,3,2,5,3,4,4,3,4,4,4,1,5,2,2,2,5,
+ 2,5,4,3,4,3,1,3,2,4,2,3,3,4,2,3,1,4,2,3,3,3,1,2,1)
> table(x,y)
   y
x  1 2 3 4 5
  1 1 1 3 1 2
  2 2 2 3 0 2
  3 0 5 6 7 0
  4 1 2 2 3 1
  5 2 2 2 0 0
```

- **Measures of central tendency (For Raw Data)**

```
> x <- scan()
1: 170 151 154 160 158 154 171 156 160 157 148 165 158
14: 160 157 159 155 151 152 161 156 164 156 163 174 153 170 149 166 154
31: 166 160 160 161 154 163 164 160 148 162 167 165 158 158 176
46:
Read 45 items
> avg <- mean(x)
> avg
[1] 159.6444
> me <- median(x)
> me
[1] 160
> xt <- table(x)
> xt
x
148 149 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 170
 2   1   2   1   1   4   1   3   2   4   1   6   2   1   2   2   2   2   1   2
171 174 176
 1   1   1
> mode <- which(xt == max(xt))
> mode
160
```

12

```
> summary(x)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
148.0	155.0	160.0	159.6	164.0	176.0

- **Measures of central tendency (For Ungrouped frequency data)**

x:	150	155	160	165	170	175
f:	6	11	14	9	3	2

```
> x<- seq(150,175,5)
> x
[1] 150 155 160 165 170 175
> f <- c(6,11,14,9,3,2)
> f
[1]  6 11 14  9  3  2
> y <- rep(x,f)
> y
 [1] 150 150 150 150 150 150 155 155 155 155 155 155 155 155 155 155 155 160 160
[20] 160 160 160 160 160 160 160 160 160 160 160 160 165 165 165 165 165 165 165
[39] 165 165 170 170 170 175 175
> avg <- mean(y)
> avg
[1] 159.7778
```

OR

```
>avg <- sum(y)/length(y)
> avg
[1] 159.7778
> me <- median(y)
> me
[1] 160
> xt <- table(y)
> xt
y
150 155 160 165 170 175
  6   11  14   9   3   2
> mode <- which(xt == max(xt))
> mode
160
  3
```

> summary(y)

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
150.0	155.0	160.0	159.8	165.0	175.0

- **Measures of central tendency (For Grouped frequency data)**

C.I.	0-25	25-50	50-75	75-100	100-125
f:	5	8	13	11	3

First step calculate mid-value of each class interval

midx:	12.5	37.5	62.5	87.5	112.5

> lb <- seq(0,100,25)
> lb
[1] 0 25 50 75 100
> ub <- seq(25,125,25)
> ub
[1] 25 50 75 100 125
> midx <- (lb+ub)/2
> midx
[1] 12.5 37.5 62.5 87.5 112.5
> f <- c(5,8,13,11,3)
> f
[1] 5 8 13 11 3
> y <- rep(midx,f)
> y
[1] 12.5 12.5 12.5 12.5 12.5 37.5 37.5 37.5 37.5 37.5 37.5 37.5
[13] 37.5 62.5 62.5 62.5 62.5 62.5 62.5 62.5 62.5 62.5 62.5 62.5
[25] 62.5 62.5 87.5 87.5 87.5 87.5 87.5 87.5 87.5 87.5 87.5 87.5
[37] 87.5 112.5 112.5 112.5
> avg <- mean(y)
> avg
[1] 61.875
> avg1 <- sum(y)/length(y)
> avg1
[1] 61.875
> avg2 <- sum(midx*f)/sum(f)
> avg2
[1] 61.875
> cf <- cumsum(f)
> cf
[1] 5 13 26 37 40

```
> mincf <- min(which(cf >= sum(f)/2))
> mincf
[1] 3
> l1 = lb[mincf]; l2 = ub[mincf]
> l1;l2
[1] 50
[1] 75
> h = (l2-l1)
> h
[1] 25
> me = l1 + (h*(sum(f)/2-cf[mincf-1])/f[mincf])
> me
[1] 63.46154
> maxf <- which(f == max(f))
> maxf
[1] 3
> d1 = f[maxf] - f[maxf-1]
> d1
[1] 5
> d2 = f[maxf] - f[maxf+1]
> d2
[1] 2
> mode = lb[maxf] + (ub[maxf]-lb[maxf])*d1/(d1 + d2)
> mode
[1] 67.85714
```

- **Measures of Dispersion (For Raw Data)**

For Raw Data 1.2, 1.4, 1.3, 1.6, 1.0, 1.5, 1.7, 1.1, 1.2, 1.3

```
> x<- c(1.2, 1.4, 1.3, 1.6, 1.0, 1.5, 1.7, 1.1, 1.2, 1.3)
> x
 [1] 1.2 1.4 1.3 1.6 1.0 1.5 1.7 1.1 1.2 1.3
> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   1.200   1.300   1.330   1.475   1.700
> rng = max(x)-min(x)
> rng
[1] 0.7
> cof_rng = rng/(max(x) + min(x))
> cof_rng
[1] 0.2592593
> int_qart_rng = (1.475-1.2)
```

```
> int_qart_rng
[1] 0.275
> semi_int_qart_rng = int_qart_rng/2
> semi_int_qart_rng
[1] 0.1375
> cof_qd = int_qart_rng/(1.475+1.2)
> cof_qd
[1] 0.1028037
> dve_mean = abs(x - mean(x))
> dve_mean
 [1] 0.13 0.07 0.03 0.27 0.33 0.17 0.37 0.23 0.13 0.03
> mean_dev = dve_mean/length(x)
> mean_dev
 [1] 0.013 0.007 0.003 0.027 0.033 0.017 0.037 0.023 0.013 0.003
> mean_dev = sum(dve_mean)/length(x)
> mean_dev
[1] 0.176
> me = median(x)
> me
[1] 1.3
> dve_me = abs(x - me)
> dve_me
 [1] 0.1 0.1 0.0 0.3 0.3 0.2 0.4 0.2 0.1 0.0
> me_dev = sum(dve_me)/length(x)
> me_dev
[1] 0.17
> var_n1 = var(x)                    # this variance is with denominator (n-1)
> var_n1
[1] 0.049
> variance = (length(x)-1)*var_n1/length(x)
> variance
[1] 0.0441
> sd = variance ^ 0.5
> sd
[1] 0.21
> cv = sd/mean(x)
> cv
[1] 0.1578947
```

- **Measures of Dispersion (For Ungrouped frequency data)**

x:	150	155	160	165	170	175
f:	6	11	14	9	3	2

> x<- seq(150,175,5)

> x

[1] 150 155 160 165 170 175

> f <- c(6,11,14,9,3,2)

> f

[1] 6 11 14 9 3 2

> y <- rep(x,f)

> y

[1] 150 150 150 150 150 150 155 155 155 155 155 155 155 155 155 155 155 160 160
[20] 160 160 160 160 160 160 160 160 160 160 160 160 165 165 165 165 165 165 165
[39] 165 165 170 170 170 175 175

> summary(y)

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
150.0	155.0	160.0	159.8	165.0	175.0

Remaining commands are as above for raw data

- **Measures of Dispersion (For Grouped frequency data)**

C.I.	0-25	25-50	50-75	75-100	100-125
f:	5	8	13	11	3

First step calculate mid-value of each class interval

midx:	12.5	37.5	62.5	87.5	112.5

> lb <- seq(0,100,25)

> lb

[1] 0 25 50 75 100

> ub <- seq(25,125,25)

> ub

[1] 25 50 75 100 125

> midx <- (lb+ub)/2

> midx

[1] 12.5 37.5 62.5 87.5 112.5

> f <- c(5,8,13,11,3)

> f

[1] 5 8 13 11 3

> y <- rep(midx,f)

> y

```
 [1]  12.5  12.5  12.5  12.5  12.5  37.5  37.5  37.5  37.5  37.5  37.5  37.5
[13]  37.5  62.5  62.5  62.5  62.5  62.5  62.5  62.5  62.5  62.5  62.5  62.5
[25]  62.5  62.5  87.5  87.5  87.5  87.5  87.5  87.5  87.5  87.5  87.5  87.5
[37]  87.5 112.5 112.5 112.5
> rng = ub[length(ub)] - lb[1]
> rng
[1] 125

> cf <- cumsum(f)
> cf
[1]  5 13 26 37 40
> q1_mincf <- min(which(cf >= sum(f)/4))
> q1_mincf
[1] 2
> q1_l1 = lb[q1_mincf]; q1_l2 = ub[q1_mincf]
> q1_l1;q1_l2
[1] 25
[1] 50
> h = (q1_l2-q1_l1)
> h
[1] 25
> first_quart = q1_l1 + (h*(sum(f)/4-cf[q1_mincf-1])/f[q1_mincf])
> first_quart
[1] 40.625
> x_bar = sum(f*midx)/sum(f)
> x_bar
[1] 61.875
> dev_mean = f * (midx - x_bar)^2
> dev_mean
[1] 12189.453125  4753.125000     5.078125  7223.046875  7688.671875

> variance = sum(dev_mean)/sum(f)
> variance
[1] 796.4844
> sd = sqrt(variance)
> sd
[1] 28.22205
```

- **Diagrammatic Representation of data**

Statistical data can be represented in the form of diagrams such as

- Simple bar diagram
- Multiple bar diagram
- Subdivided bar diagram
- Pie diagram or pie chart

> barplot(height, beside = T, names.arg = NULL, col = NULL, border = par("fg"), main = NULL, xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL,...)

height: Either a vector or matrix of values describing the bars which make up the plot. If height is a vector, the plot consists of a sequence of rectangular bars with heights given by the values in the vector. If height is a matrix and beside is FALSE then each bar of the plot corresponds to a column of height, with the values in the column giving the heights of stacked sub-bars making up the bar. If height is a matrix and beside is TRUE, then the values in each column are juxtaposed rather than stacked.

names.arg: A vector of names to be plotted below each bar or group of bars. If this argument is omitted, then the names are taken from the names attribute of height if this is a vector, or the column names if it is a matrix.

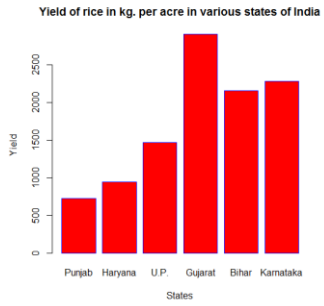
main: Overall title for the plot.

beside: A logical value. If FALSE, the columns of height are portrayed as stacked bars, and if TRUE the columns are portrayed as juxtaposed bars.

Example: The following table gives the average approximate yield of rice in kg. per acre in various states of India in 2003-04. Represent it by **Simple Bar diagram**.

State :	Punjab	Haryana	U.P.	Gujarat	Bihar	Karnataka
Yield :	728	943	1469	2903	2153	2276

```
> x <- c("Punjab", "Haryana", "U.P.", "Gujarat", "Bihar", "Karnataka")
> y <- c(728, 943, 1469, 2903, 2153, 2276)
> x
[1] "Punjab"   "Haryana"  "U.P."     "Gujarat"  "Bihar"    "Karnataka"
> y
[1]  728  943 1469 2903 2153 2276
> barplot(y, names.arg = x, col = "red", border = "blue", main = "Yield of rice in kg. per acre in various states of India", xlab = "States", ylab = "Yield")
```

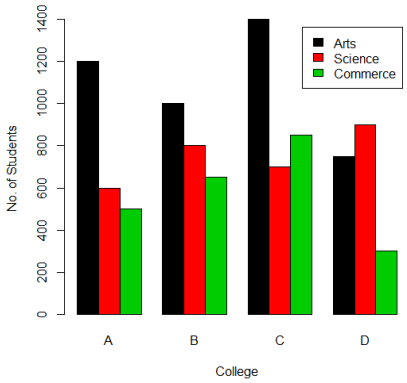


Example: Represent the following data on faculty-wise distribution of students, by **multiple bar diagram**.

College	Arts	Science	Commerce
A	1200	600	500
B	1000	800	650
C	1400	700	850
D	750	900	300

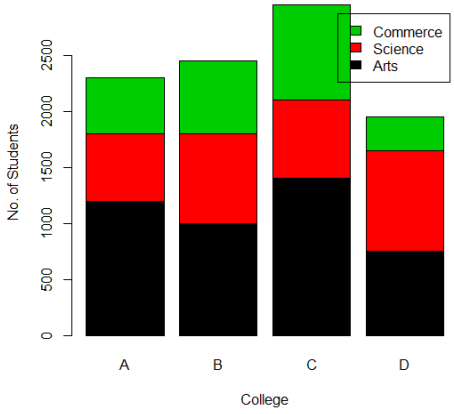
```
> clg <- c("A", "B", "C", "D")
> clgA <- c(1200, 600, 500)
> clgB <- c(1000, 800, 650)
> clgC <- c(1400, 700, 850)
> clgD <- c(750, 900, 300)
> d = data.frame(clgA, clgB, clgC, clgD)
> d
  clgA clgB clgC clgD
1 1200 1000 1400  750
2  600  800  700  900
3  500  650  850  300
> d1 = as.matrix(d)


> d1
     clgA clgB clgC clgD
[1,] 1200 1000 1400  750
[2,]  600  800  700  900
[3,]  500  650  850  300
```

> barplot(d1, beside = T, names.arg = clg, col = 1:2:3, legend = c("Arts", "Science", "Commerce"),xlab = "College", ylab = "No. of Students")

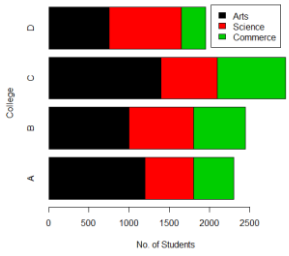


For the above example draw **subdivided bar diagram.**

> barplot(d1, beside = F, names.arg = clg, col = 1:2:3:4, legend = c("A", "B", "C", "D"),xlab = "College", ylab = "No. of Students")



> barplot(d1, beside = F, horiz = T, names.arg = clg, col = 1:2:3, legend = c("Arts", "Science", "Commerce"),ylab = "College", xlab = "No. of Students")



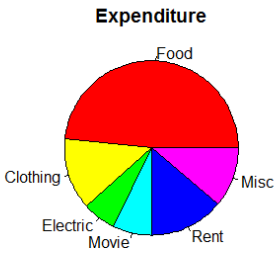
Example: Represent the following data by a **pie diagram**:

Item :	Food	Clothing	Recreation	Indian	Rent	Miscellaneous
Expenditure (in Rs.)	87	24	11	13	25	20

> itm <- c("Food", "Clothing", "Electric", "Movie", "Rent", "Misc")

> exp ,- c(87, 24, 11, 13, 25, 20)

> pie(exp, main = "Expenditure", labels = itm, radius = 1, col=rainbow(length(exp)))



Graphical Representation of data:

Statistical data can be represented in the form of graphs such as

- Histogram
- Frequency polygon
- Ogive curve

R supports commands hist, plot, lines, points etc for drawing above graphs.

Histogram:

> hist(x, breaks = classlimits, freq/probability = False/True, density = NULL, col = NULL, border = NULL, main = paste("Histogram of" , xname), xlim = range(breaks), ylim = NULL, xlab = xname, ylab=yname, axes = TRUE . . . .)

x: A vector of values for which the histogram is desired.

breaks: A vector giving breakpoints (class limits) for histogram. This can be done using c() or seq(). For eg: **breaks=c(100, 300, 500, 700)** Compute a histogram for the raw data values and set the bins (bars) such that they run from 100 to 300, 300 to 500 and 500 to 700. However, the `c()` function can make your code very messy sometimes. That is why you can instead use **breaks=**`seq(x, y, z)`. The values of x, y and z are determined by yourself and represent, in order of appearance, the begin number of the x-axis, the end number of the x-axis and the interval in which these numbers appear.

> brk <- seq(148,178,5)

> hist(x, breaks = brk)

This command creates histogram with class limits 148 to 153, 153 to 158, 158 to 163, 163 to 168, 168 to 173, 173 to 178.

Note that you can also combine the two functions:

> hist(x, breaks=c(100, seq(200,700, 150)))

Make a histogram for the vector x, start at 100 on the x-axis, and from values 200 to 700, make the bins 150 wide

freq/probability: logical; if TRUE, the histogram graphic is a representation of frequencies; if FALSE, probability densities, are plotted (so that the histogram has a total

area of one). Defaults to TRUE *if and only if* breaks are equidistant (and probability is not specified).

density: the density of shading lines, in lines per inch. The default value of NULL means that no shading lines are drawn.

col: a colour to be used to fill the bars. The default of `NULL` yields unfilled bars.

border: the color of the border around the bars. The default is to use the standard foreground color.

main: Overall title for the plot.

> brk <- seq(148,178,5)

> xnme = "Heights"

> hist(x, breaks = brk, freq = FALSE, main = paste("Histogram of" , xnme))

> x <- scan()

1: 170 151 154 160 158 154 171 156 160 157 148 165 158

14: 160 157 159 155 151 152 161 156 164 156 163 174 153 170 149 166 154

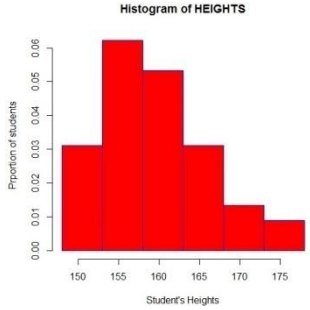
31: 166 160 160 161 154 163 164 160 148 162 167 165 158 158 176

46:

Read 45 items

> hist(x)

> hist(x, breaks = brk, freq = FALSE, col = "red", border = "blue", main = paste("Histogram of" , xnme), xlab = "Student's Heights", ylab="Prportion of students")



Histogram for ungrouped frequency data

x:	150	155	160	165	170	175
f:	6	11	14	9	3	2

> x <- seq(150,175,5)

> f <- c(6,11,14,9,3,2)

> y <- rep(x,f)

> hist(y)

> t = seq(147.5,177.5,5)

> hist(y, breaks = t)

Histogram for grouped frequency data

C.I.	0-25	25-50	50-75	75-100	100-125
f:	5	8	13	11	3

> midx <- seq(12.5,112.5,25)

> f <- c(5,8,13,11,3)

> cls_limit <- seq(0,125,25)

> y <- rep(midx,f)

> hist(y)

> hist(y, breaks=cls_limit)

Frequency polygon:

It is obtained by joining the points (x_i, f_i) where x_i is the midpoint of the i^{th} class interval and f_i is the corresponding frequency.

> lb <- seq(0,100,25)

> ub <- seq(25, 125, 25)

> midx <- (lb+ub)/2

> f <- c(5,8,13,11,3)

> x0 <- c(0, midx, 125)

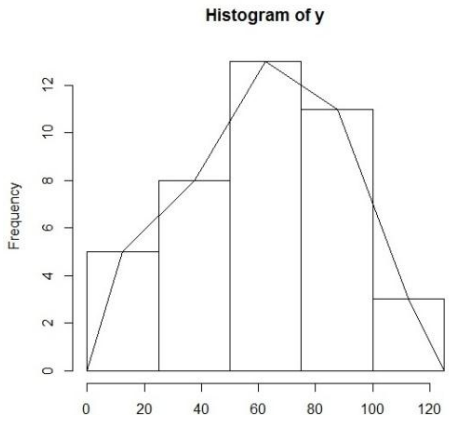
> f0 <- c(0,f,0)

> y <- rep(midx,f)

> bks <- seq(0,125,25)

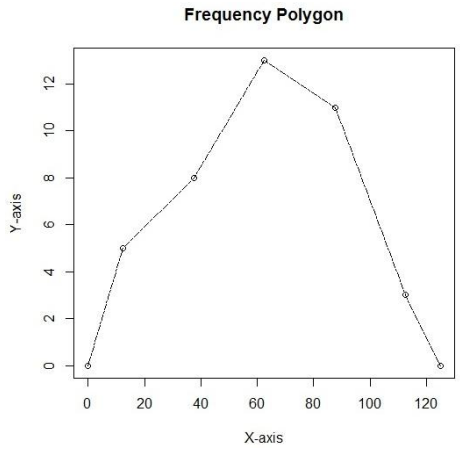
> hist(y,breaks=bks)

> lines(x0, f0)



OR

> plot(x0,f0, main = "Frequency Polygon", xlab ="X-axis", ylab = "Y-axis", type = "o", lty =6, xlim = range(min(x0),max(x0)))



Binomial, Poisson, Hypergeometric distribution

Binomial Distribution:

The binomial distribution is a discrete probability distribution. It describes the outcome of n independent trials called Bernoulli trials in an experiment. Each trial is assumed to have only two outcomes, either success or failure. If the probability of a successful trial is p, then the probability of having x successful outcomes in an experiment of n independent trials is as follows.

$$P(X = x) = \binom{n}{x} p^x q^{n-x} \qquad , x = 0,1,2, \dots, n$$

Notation: X ~ B(n, p) where,
n : number of independent Bernoulli trials
p: Probability of success in a single trial.

R commands for Binomial distribution –

- dbinom(x, size, prob , log = FALSE) : Gives the probability at x i.e. P(X=x)
- pbinom(q, size, prob, lower.tail=TRUE, log.p= FALSE): Gives the cumulative probability i.e. P(X<=x)
- rbinom(n, size, prob) : Takes a random sample of size n from Binomial distribution
- qbinom(p, size, prob, lower.tail = TRUE, log.p=FALSE) : Finds quantiles of Binomial distribution. It is the smallest integer *x* such that $P(X \leq x) \geq p$

where,

x, q	vector of quantiles.
p	vector of probabilities.
n	number of observations. If length(n) > 1, the length is taken to be the number required.
size	number of trials (zero or more).
prob	probability of success on each trial.
log, log.p	logical; if TRUE, probabilities p are given as log(p).
lower.tail	logical if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.

Examples :

1) Let X ~ B(12,0.4). Calculate i) P(X=2) ii) P(X ≤ 3) iii) P(X ≥ 8) iv) P(4 ≤ X ≤ 6).

Solution:

i) rp1= dbinom(2, 12,0.4)

ii) rp2 = pbinom(3, 12, 0.4)

OR

rp2 = sum(dbinom(0:3,12,0.4))

iii) rp3 = pbinom(8, 12,0.4,lower.tail= FALSE) + dbinom(8,12.0.4)

iv) rp4 = sum(dbinom(4:6,12,0.4)

2) Let X~B(n=8, p= 0.3). Find k such that i) P(X ≤ k) = 0.2552

Solution:

i) k = qbinom(0.2552, 8, 0.3, lower.tail = TRUE)

3) Draw a random sample of size 8 from a binomial distribution with n=6 and p = 1/3. Find mean and variance of sample.

Solution :

```
r = rbinom(8, 6, 1/3)
m = mean(r)
```

4) If the probability that any person 70 years old will be dead within a year is 0.05, find the probability that out of a group of 7 such persons i) exactly one ii) none iii) at least two iv) not more than one

Solution : X ~ B(7, 0.05)

i) rp1= P(X=1) = dbinom(1,7,0.05)

ii) rp2 = P(X=0) = dniom(0,7,0.05)

iii) rp3= P(X ≥ 1) = pbinom(1, 7, 0.05 , lower.tail= FALSE) +dbinom(1,7,0.05)

iv) rp4= P(X ≤ 1) = pbinom(1, 7, 0.05 , lower.tail= TRUE)

5) Fit a Binomial distribution to the following data and find the expected frequencies. Plot observed and expected frequencies and comment on the adequacy of the model.

x :	0	1	2	3	4	5
f :	6	15	25	42	18	4

Solution :

```
x = 0:5
f = c(6, 15, 25, 42, 18, 4)
m = sum(f*x)/sum(f)
n = max(x); p = m/n; q = 1-p
px = dbinom(x,n,p)
px = round(px,6)
ef = sum(f)*px
ef = round(pf,0)
d = data.frame(x,f, “fx”=f*x,”probability”= px, “exp.freq”=ef)
print (d)
plot(f,ef,”P”,pch=16)
abline(0,1)
```

Poisson distribution:

This is a distribution associated with rare events. e.g. traffic accidents, typing errors etc. The Poisson distribution is the probability distribution of independent event occurrences in an interval. If λ is the mean occurrence per interval, then the probability of having x occurrences within a given interval is:

$$P(X = x) = \frac{e^{-\lambda}\lambda^{x}}{x!} \quad , \quad x = 0,1,2, \ldots$$

Notation: X ~ P(λ)

R commands for Poisson distribution –

- dpois(x, lambda, log = FALSE) : Gives the probability at X i.e. P(X=x)
- ppois(q, lambda, lower.tail = TRUE, log.p = FALSE) : Gives the cumulative probability i.e. P(X<=x)
- rpois(n, lambda) : Takes a random sample of size n from Poisson distribution
- qpois(p, lambda, lower.tail = TRUE, log.p = FALSE) : Finds quantiles of Poisson distribution. It is the smallest integer *x* such that $P(X \leq x) \geq p$

Examples:

1) Let X ~ P(λ=0.5). Find i) P(X=2) ii) P(X>1) iii) P(3≤X≤5) iv) P(X<4).

Solution :

i) rp1= dpois(2, 0.5)

ii) rp2 = ppois(1, 0.5,lower.tail= FALSE)

iii) rp3 = sum(dpois(3:5,0.5)

iv) rp4 = ppois(4, 0.5) – dpois(4,0.5)

OR

rp4 = sum(dpois(0:3,0.5))

2) X is Poisson random variable such that P(X=0) = 0.5, find P(X > 1).

Solution:

lambda = - log (0.5)

rp = ppois(1, lambda, lower.tail = FALSE)

3) It is claimed that the chance of twin in human births is 0.001 and the chance of a triplet is 0.0001. Compute the probabilities of observing at least i) 5 sets of twins ii) 4 sets of triplets in a data of 10,000 human births.

Solution:

i) Let X : number of twin human births.

X ~ P(lambda = 10)

rp = P(X=5) = dpois(5, 10)

ii) Let Y : number of triplet human births.

Y ~ P(lambda = 1)

rp = P(Y=4) = dpois(4, 10)

4) It is claimed that 3 persons in every 1000 is having cataract problem. Find the probability that a random sample of 2500 people will contain fewer than 10 people with cataract problem.

Solution :

X ~ P(λ = 7.5)

rp = P(X < 10) = sum(dpois(0:9, 7.5))

5) Fit Poisson distribution to the following data and find the expected frequencies. Plot observed and expected frequencies and comment on the adequacy of the model.

x :	0	1	2	3	4	5	6
f :	93	67	50	30	16	5	2

Solution :

```
x = 0:6
f = c(93, 67, 50, 30, 16,5,2)
m = sum(f*x)/sum(f)
n = max(x); lambda = m/n
px = dpois(x, lambda)
px = round(px,6)
ef = sum(f)*px
ef = round(pf,0)
d = data.frame(x,f, “fx”=f*x,”probability”= px, “exp.freq”=ef)
print (d)
plot(f,ef,”P”,pch=16)
abline(0,1)
```

Hypergeometric Distribution:

Suppose a population consists of a finite number of N items. Let M out of these N items possess attribute A and the remaining N-M items do not possess attribute A. Suppose n items are drawn at random without replacement from the population. Let X denotes the number of items possessing attribute A that are drawn in the sample.

A discrete random variable X is said to have hypergeometric distribution with parameters k, m and n if its p.m.f. is given by

$$P(X = x) = \frac{\binom{m}{x}\binom{n}{k-x}}{\binom{m+n}{k}} \quad , x = 0,1,2, \dots, \min\{n, k\}$$

$$= 0 \qquad \text{o.w}$$

commands for Hypergeometric distribution –

- dhyper(x, m, n, k, log = FALSE) : Gives the probability at X i.e. P(X=x)
- phyper(q, m, n, k, lower.tail = TRUE, log.p = FALSE) : Gives the cumulative probability i.e. P(X<=x)
- qhyper(p, m, n, k, lower.tail = TRUE, log.p = FALSE) : Finds quantiles of Hypergeometric distribution.
- rhyper(nn, m, n, k) : Takes a random sample of size n from Hypergeometric distribution

Arguments

x, q	vector of quantiles representing the number of white balls drawn without replacement from an urn which contains both black and white balls.
M	the number of white balls in the urn.
N	the number of black balls in the urn.
K	the number of balls drawn from the urn.

P probability, it must be between 0 and 1.

Nn number of observations. If length(nn) > 1, the length is taken to be the number required.

log, log.p logical; if TRUE, probabilities p are given as log(p).

lower.tail logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.

Examples :

1) A factory has 400 employees, of which 280 are permanent and the rest are temporary. Twelve employees are selected at random without replacement. Find the probability that i) at most 3 temporary employees ii) exactly 4 temporary employees are selected in the sample.

Solution:

i) rp = P(X≤3) = phyper(x=3, m=120, n=280, k=12)

ii) rp = P(X = 4) = dhyper(x=4, m=120, n=280, k=12)

2)A dealer has 12 Maruti zen cars and 8 santro cars in his shop. If a sample of 4 cars is selected at random from the shop, find the probability that i) at most one santro cars is selected ii) 3 maruti zen cars are selected.

Solution :

i) rp1= P(X<=2) = phyper(2, m=8,n=12,k=4)

ii) rp2 = P(X=1) = dhyper(1, m=8,n=12,k=4)

3) Among 500 employees of a factory, 200 are the union members, while the others are not. If 10 employees are selected at random for a special training programme, find the probability that i) 4 of them ii) less than 3 iii) more than 5 will be the union members while the others are not when sampling is done without replacement.

Solution :

j) rp = P(X = 4) = dhyper(x=4, m=200, n=300, k=10)

ii) rp = P(X < 4) = sum (dhyper(x= 0:3, m=200, n=300, k=10))

iii)rp = P(X >5) = phyper(q=5, m=200, n=300, k=10, lower.tail=FALSE)

Session – V
Normal, t, χ^2 probabilities

For every distribution there are four commands. The commands for each distribution are prepended with a letter to indicate the functionality:

"r"	returns randomly generated numbers
"d"	returns the height of the probability density function
"p"	returns the cumulative density function
"q"	returns the inverse cumulative density function (quantiles)

R- Commands for Normal distribution:

- rnorm(n, mean, sd) : This command is used for generating random sample of size n from normal distribution with the specified values of mean and sd.
- dnorm(x, mean, sd): Given a set of values it returns the height of the probability distribution at each point of normal distribution with specified values of mean and sd.
- pnorm(x, mean, sd): This command is to determine CDF at x of normal distribution with specified values of mean and sd. This gives area left of x.
- qnorm(p, mean, sd): This command is used to obtain p^{th} quantile (x) of normal distribution with specified values of mean and sd.

Following command generates 5 random numbers from standard normal distribution

> rnorm(5)

[1] -0.1589664 0.3369436 -0.7376138 1.3808449 0.3558597

Following command generates 5 random numbers from normal distribution with mean 15 and sd 4.

> rnorm(5,15,4)

[1] 14.048153 16.243795 14.624714 15.920500 9.295134

Following command gives height of standard normal probability distribution at the point x = 0.

> dnorm(0)

[1] 0.3989423

Following command gives height of normal probability distribution with ($\mu = 4$, $\sigma = 10$) at the point x = 0.

> dnorm(0,mean=4,sd=10)

[1] 0.03682701

> v <- c(0,1,2)

> v

[1] 0 1 2

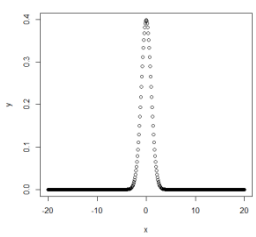
Following command gives height of standard normal probability distribution at the point x = 0, 1, 2.

> dnorm(v)

[1] 0.39894228 0.24197072 0.05399097

Following commands plots standard normal probability curve

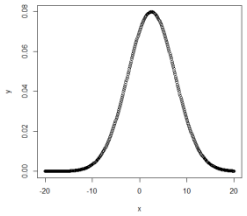
> x <- seq(-20,20,by=.1); y <- dnorm(x); plot(x,y)



Following commands plot normal probability curve with ($\mu = 2.5$, $\sigma = 5$)

> y <- dnorm(x,mean=2.5,sd=5)

> plot(x,y)



Following command gives $P(Z \leq 1.96)$ and $P(Z \leq 1.645)$

> pnorm(1.96)

[1] 0.9750021

> pnorm(1.645)

[1] 0.9500151

Following command gives $P(x \leq 18)$ where $X \sim N(\mu = 20, \sigma = 3)$

> pnorm(18, 20, 3)

[1] 0.2524925

Following commands displays values of a and b such that $P(Z \leq a) = 0.25$; $P(Z \leq b) = 0.5$

> qnorm(0.25)

[1] -0.6744898

> qnorm(0.5)

[1] 0

Following command displays the value c such that $P(X \leq c) = 0.30$ where $X \sim N(\mu=15, \sigma=4)$

> qnorm(0.30,15,4)

[1] 12.9024

R- Commands for t distribution:

The commands follow the same kind of naming convention, and the names of the commands are dt, pt, qt, and rt.

- rt(n,df): This function is used to draw a random sample of size n from t distribution with specified degrees of freedom
- dt(n,df): This function is used to obtain density function of t distribution for vector x and with specified degrees of freedom
- pt(x,df): This function is used to obtain cumulative distribution function of t distribution for vector x and with specified degrees of freedom
- qt(p,df):This function is used to obtain p^{th} quantile of t distribution with specified degrees of freedom

Following commands generates random numbers according to t-distribution

> rt(3,df=10)

[1] -0.1012297 0.6990362 -0.3691327

> rt(3,df=10)

[1] 0.1457079 -2.6634936 1.5102245

> rt(5,df=12)

[1] -2.8224425 0.8844997 -2.5102270 -0.3751747 -1.3321576

> rt(5,df=12)

[1] -0.5246495 0.1651533 1.6191017 0.2533841 1.8454552

Following command displays y-coordinate of t-probability curve with degrees of freedom 10 at t = 4 i.e. $f(t)_{n=10} = f(4)_{n=10} = 0.002031034$

> y <- dt(4,10)

> y

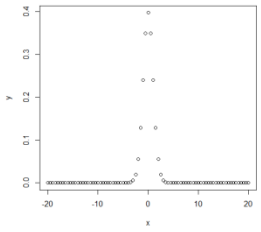
[1] 0.002031034

Following commands plot t-probability curve with degrees of freedom 50

> x <- seq(-20,20,by=.5)

> y <- dt(x,df=50)

> plot(x,y)



Following command displays cumulative probability of t-distribution with 10 degrees of freedom i.e. $P(t_{n=10} \leq -3)$

> pt(-3,df=10)

[1] 0.006671828

Following command displays cumulative probability of t-distribution with 10 degrees of freedom i.e. $P(t_{n=10} \leq -3)$, $P(t_{n=10} \leq -4)$, $P(t_{n=10} \leq -2)$, $P(t_{n=10} \leq -1)$,

> x = c(-3,-4,-2,-1)

> pt(x,10)

[1] 0.006671828 0.001259166 0.036694017 0.170446566

Following command displays inverse cumulative probability distribution i.e. $P(t_{n=10} \leq a) = 0.05$

> qt(0.05,df=10)

[1] -1.812461

Following command displays values of b, c and d i.e. inverse cumulative probabilities such that $P(t_{n=7} \leq b) = 0.005$, $P(t_{n=7} \leq c) = 0.025$, $P(t_{n=7} \leq d) = 0.05$

> v <- c(0.005,.025,.05)

> qt(v,df=7)

[1] -3.499483 -2.364624 -1.894579

R- Commands for χ^2 distribution:

There are four functions that can be used to generate the values associated with the Chi-Squared distribution. These commands work just like the commands for the t-distribution. The commands follow the same kind of naming convention, and the names of the commands are **dchisq, pchisq, qchisq, and rchisq.**

Testing of hypothesis

R-command for large samples

Problem: Test the hypothesis H_0: $\mu = 10$ against H_1: $\mu \neq 10$. A random sample of size 400 is drawn gives mean 10.2 and standard deviation 2.25. Use LOS = 5%.

This is large sample test problem. Decision about acceptance or rejection of null hypothesis can be taken using p-value. If p-value is less than LOS then reject null hypothesis

> n=400; mean=10.2; sd=2.25; m0=10
> zcal = (mean-m0)/(sd/sqrt(n))
> zcal
[1] 1.777778
> pvalue = 2*(1-pnorm(abs(zcal))) or > pvalue = 2*pnorm(abs(zcal), lower.tail=F)
> pvalue
[1] 0.07544036

Conclusion: Since p-value > LOS, therefore do not reject H_0 at 5% LOS.

Problem: Test the hypothesis H_0: $\mu \geq 50$ against H_1: $\mu < 50$. A random sample of size 65 is drawn gives mean 47.8 and standard deviation 10. Use LOS = 5%.

> n=65;mean=47.8;sd=10;m0=50
> zcal = (mean-m0)/(sd/sqrt(n))
> zcal
[1] -1.773697
> pvalue = pnorm(zcal)
> pvalue
[1] 0.03805667

Conclusion: Since p-value < LOS, therefore reject H_0 at 5% LOS.

Problem: Two random samples of sizes 1000 and 2000 are drawn from two populations with same standard deviation 2.5 gives means 67.5 and 68 respectively. Test the hypothesis H_0: $\mu_1 = \mu_2$ against H_1: $\mu_1 \neq \mu_2$. Use 5% LOS.

> n1=1000; n2=2000; xbar1 = 67.5; xbar2 = 68; s1 = 2.5; s2 = 2.5
> zcal = (xbar1 - xbar2)/sqrt(s1^2/n1 + s2^2/n2)
> zcal
[1] -5.163978
> pvalue = 2*(1-pnorm(abs(zcal)))
> pvalue
[1] 2.417564e-07 is approximately zero

Conclusion: Since p-value < LOS, therefore reject H_0 at 5% LOS.

Problem: A trucking firm suspects that that the average lifetime of 28,000 miles claimed for certain tyre is too high. To check this claim the firm puts 40 of these tyres on trucks and gets a mean lifetime of 27,563 miles and a standard deviation of 1,348 miles. What will the trucking firm conclude at 0.01 level of significance if it tests the null hypothesis $\mu = 28{,}000$ miles against an appropriate alternative? Assume Normal distribution. Find 'p' value and interpret the value.

H_0: $\mu = 28000$ against H_1: $\mu < 28000$, n = 40,
sample mean = 27563, sample sd = 1348 $\alpha = 0.01$

```
> n=40;mean=27563;sd=1348;m0=28000
> zcal = (mean-m0)/(sd/sqrt(n))
> zcal
[1] -2.050319
> pvalue = pnorm(zcal)
> pvalue
[1] 0.02016663
```

Conclusion: Since p-value > LOS, therefore do not reject H_0 at 1% LOS.

Problem: Experience has shown that 20% of a manufactured product is of the top quality. In one day's production of 400 articles only 50 are of top quality. Test the hypothesis that experience of 20% is wrong.

H_0: P = 0.20 against H_1: P≠ 0.20 n = 400,
Sample proportion p = 50/400 = 0.125 $\alpha = 0.05$

```
> n=400; p = 0.125; P = 0.20
> zcal = (p - P)/sqrt(P*(1-P)/n)
> zcal
[1] -3.75
> pvalue = 2*(1-pnorm(abs(zcal)))
> pvalue
[1] 0.0001768346
```

Conclusion: Since p-value < LOS, therefore reject H_0 at 5% LOS.

Problem: From each of two consignments of apples, a sample of size 200 is drawn, and number of rotten apples counted. Test whether the proportion of rotten apples in the two consignments are significantly different?

	Sample size	No. of rotten apples
Consignment A	200	44
Consignment B	200	30

n1 = 200, p1 = 44/200, n2 = 200, p2 = 30/200, $\alpha = 0.05$
H_0: $P_1 = P_2$ against H_1: $P_1 \neq P_2$

```
> n1=200; p1 = 44/200; n2 = 200; p2 = 30/200
> p = ((n1 * p1) + (n2 * p2))/(n1 + n2)
> q = 1 - p
> p;q
[1] 0.185
[1] 0.815
> se = sqrt(p*q*(1/n1 + 1/n2))
> se
[1] 0.03882976
```

> zcal = (p1 - p2)/se
> zcal
[1] 1.802741
> pvalue = 2*(1-pnorm(abs(zcal)))
> pvalue
[1] 0.07142888
Conclusion: Since p-value > LOS, therefore do not reject H_0 at 5% LOS.

R-command for small samples
To test single population mean:

- Check normality of two populations using Shapiro-Wilk normality test or Q-Q plot.
- t.test(x, mu, alt, conf.level)

Where x: vector of observations
mu: specified value of true mean i.e. μ_0
alt: alternative hypothesis with three options alt = "less", alt = "greater" and by default alt = "two-sided"
conf.level: confidence coefficient (1 - α). Default option is 0.95

Analytical methods of checking assumption of normality of parent population:
If the sample size is small, Q-Q plot or boxplot fails to judge about normality of parent population. For small sample size, an analytical test, known as Shapiro-Wilk test is designed to check normality of population. The R-command to test normality of population is
shapiro.test(x) where x: vector of observations

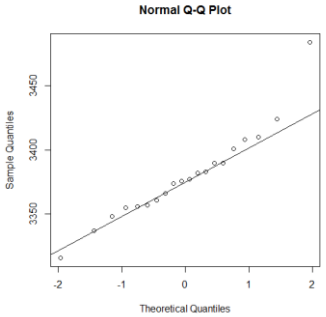
This command returns value of test-statistic W and p-value of test statistic.
H_0: population is normally distributed.
Thus if the p-value < α, then the null hypothesis is rejected and there is evidence that the data tested are not from a normally distributed population. In other words, the data are not normal. On the contrary, if the p-value > α, then the null hypothesis that the data came from a normally distributed population cannot be rejected. However, since the test is biased by sample size, the test may be statistically significant from a normal distribution in any large samples. Thus a Q–Q plot is required for verification in addition to the test.

Problem: Write R-command, for the following data to test the hypothesis Ho: μ = 3400 against H_1: μ < 3400 at 5% LOS.

3366	3337	3361	3410	3316	3357	3348	3356	3376	3382
3377	3355	3408	3401	3390	3424	3383	3374	3384	3390

For applying t-test, basic assumption is, sample is drawn from normal population.
> qqnorm(x)

> qqline(x)



For small sample, it is not possible to judge about normality of parent population using Q-Q plot. Therefore we use Shapiro-Wilk test as follows

> x<- c(3366, 3337, 3361, 3410, 3316, 3357, 3348, 3356, 3376, 3382, 3377, 3355, 3408, 3401, 3390, 3424, 3383, 3374, 3484, 3390)

> x

[1] 3366 3337 3361 3410 3316 3357 3348 3356 3376 3382 3377 3355 3408 3401 3390 3424 3383 3374 3484 3390

> shapiro.test(x)

Shapiro-Wilk normality test

data: x

W = 0.9332, p-value = 0.1777

Since p-value > α, therefore the null hypothesis that the data came from a normally distributed population cannot be rejected. Hence we can apply t-test for testing population mean.

H_0: $\mu = 3400$ H_1: $\mu < 3400$

> t.test(x, mu=3400, alt="less")

One Sample t-test

data: x

t = -2.5268, df = 19, p-value = 0.01027

alternative hypothesis: true mean is less than 3400

95 percent confidence interval:

-Inf 3393.607

sample estimates:

mean of x

3379.75

Interpretation: t_{cal} = -2.5268,

degrees of freedom = 19,

p-value = 0.01027

Since p-value < α (0.05), therefore decision is to reject H_o at 5% level of significance.

95% C.I = ($-\infty$, 3393.607)

Note: Since alternative hypothesis is one sided, lower limit of the class interval is -∞.

Point estimate = $\bar{x} = 3379.5$

Decision about acceptance or rejection of null hypothesis can be taken on the basis of confidence interval also. In the above example $\mu_0 = 3400$ lies outside the 95% C.I., hence H_0 is rejected at 5% level of significance.

Problem: The following data refer to the amount of coffee (in ounces) filled by a machine in six randomly picked jars: 15.7, 15.9, 16.3, 16.2, 15.7 and 15.9. Is the true mean amount of coffee in a jar is 16 ounces? Use LOS 5%.

$H_0: \mu = 16$ $H_1: \mu \neq 16$

Since sample size n = 6 is small, we have to use t-test, which is based on the normality of parent population. Therefore first we have to use Shapiro-Wilk normality test.

```
> x <- c(15.7, 15.9, 16.3, 16.2, 15.7, 15.9)
> x
[1] 15.7 15.9 16.3 16.2 15.7 15.9
> shapiro.test(x)
        Shapiro-Wilk normality test
data:  x
W = 0.8788, p-value = 0.2636
```

Since p-value > α, therefore the null hypothesis that the data came from a normally distributed population cannot be rejected. Hence we can apply t-test for testing population mean.

```
> t.test(x, mu=16)
        One Sample t-test
data:  x
t = -0.488, df = 5, p-value = 0.6462
alternative hypothesis: true mean is not equal to 16
95 percent confidence interval:
 15.68659 16.21341
sample estimates:
mean of x
   15.95
```

Interpretation: $t_{cal} = -0.488$,
degrees of freedom = 5,
p-value = 0.6462

Since p-value > α (0.05), therefore decision is, do not reject H_o at 5% level of significance.
95% C.I = (15.68659 16.21341)
Point estimate = $\bar{x} = 15.95$

Decision about acceptance or rejection of null hypothesis can be taken on the basis of confidence interval also. In the above example $\mu_0 = 16$ lies within the 95% C.I., hence H_0 is accepted at 5% level of significance.

To test equality of two population means for two independent normal populations with equal variances:

- Check normality of two populations using Shapiro-Wilk normality test or Q-Q plot.

- Check for equality of variances using F-test for which R-command is

var.test(x, y, ratio = 1, alternative = c("two.sided", "less", "greater"), conf.level = 0.95)

Where x , y: numeric vectors of data values

ratio: the hypothesized ratio of the population variances of x and y

alternative: a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter

conf.level: confidence level

- t.test(x, y, alt = ”less or greater”, var.equal = T)

Default value of alt is two.sided and var.equal is F.

Problem: Below are given the gain in weights (in lbs) of pigs fed on two diets A and B

Gain in weight

Diet A:25, 32, 30, 43, 24, 14, 32, 24, 31, 31, 35, 25

Diet B:44, 34, 22, 10, 47, 31, 40, 30, 32, 35, 18, 21, 35, 29, 22

Test, if the two diets differ significantly as regards their effect on increase in weight. Use LOS 5%.

> x <- c(25,32,30,34,24,14,32,24,30,31,35,25)

> y <- c(44,34,22,10,47,31,40,30,32,35,18,21,35,29,22)

> shapiro.test(x)

Shapiro-Wilk normality test

data: x

W = 0.8875, p-value = 0.1095

Conclusion1: Since p-value > LOS, therefore null hypothesis of normality of population X is accepted.

> shapiro.test(y)

Shapiro-Wilk normality test

data: y

W = 0.9772, p-value = 0.947

Conclusion2: Since p-value > LOS, therefore null hypothesis of normality of population Y is accepted.

> var.test(x, y, ratio = 1)

F test to compare two variances

data: x and y

F = 0.343, num df = 11, denom df = 14, p-value = 0.08144

alternative hypothesis: true ratio of variances is not equal to 1

95 percent confidence interval:

0.1108401 1.1520871

sample estimates:

ratio of variances

0.3430045

Conclusion3: Since p-value > LOS, therefore null hypothesis of equality of two population variances is accepted.

Since both the assumptions; normality of two populations and equality of two population variances are satisfied therefore we can apply t-test.

> t.test(x, y, var.equal = T)

Two Sample t-test

data: x and y

t = -0.6103, df = 25, p-value = 0.5472

alternative hypothesis: true difference in means is not equal to 0

95 percent confidence interval:

-8.749507 4.749507

sample estimates:

mean of x mean of y

28 30

Final Conclusion: Since p-value > LOS, therefore null hypothesis of equality of two population means is accepted and we may conclude that two diets do not differ significantly as regards their effect on increase in the weight.

[Note: If assumption of normality of parent population fails, then use non-parametric methods. Also for normal populations, if equality of population variances is rejected by F-test, then usual t-test is not applicable. Instead, Welch's variant of t-test is used. To carry out this test, R-command is t.test(x, y, mu = 0, alt). In this command argument var.equal is not required, as its default value is FALSE.]

Paired t-test

This test is based on the assumption of normality of population differences.

- Check normality of two populations differences using Shapiro-Wilk normality test using R-command shapiro.test(d) where d = x - y
- t.test(d, mu = 0, conf.level) or t.test(x, y, paired = T, conf.level)

Problem: Eleven school boys were given a test in mathematics. They were given a month's tuition and a second test was held at the end of it. Do the marks give evidence that the student's have benefited by the extra coaching? Use LOS 5%.

Marks in test-1: 23, 20, 19, 21, 18, 20, 18, 17, 23, 16, 19

Marks in test-2: 24, 19, 22, 18, 20, 22, 20, 20, 23, 20, 17

> x <- c(23, 20, 19, 21, 18, 20, 18, 17, 23, 16, 19)

> y <- c(24, 19, 22, 18, 20, 22, 20, 20, 23, 20, 17)

> d = x -y

> x; y; d

[1] 23 20 19 21 18 20 18 17 23 16 19

[1] 24 19 22 18 20 22 20 20 23 20 17

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

> shapiro.test(d)

Shapiro-Wilk normality test

data: d

W = 0.9352, p-value = 0.4656

Conclusion: Since p-value > LOS, therefore hypothesis of normality of two populations differences is accepted. Hence we can use paired t-test.

> t.test(d, mu = 0)

```
        One Sample t-test
data:  d
t = -1.4832, df = 10, p-value = 0.1688
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -2.5022109  0.5022109
sample estimates:
mean of x
      -1
```

OR

```
> t.test(x, y, paired = T)
        Paired t-test
data:  x and y
t = -1.4832, df = 10, p-value = 0.1688
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2.5022109  0.5022109
sample estimates:
mean of the differences
                  -1
```

```
> t.test(d, mu = 0,conf.level = 0.99)          #99% confidence interval
        One Sample t-test
data:  d
t = -1.4832, df = 10, p-value = 0.1688
alternative hypothesis: true mean is not equal to 0
99 percent confidence interval:
 -3.136723  1.136723
sample estimates:
mean of x
      -1
```

Final Conclusion: Since p-value > LOS, therefore null hypothesis that there is no significant difference between the marks obtained by students before and after tuition is accepted i.e. marks do not give any evidence that the student's have benefited by the extra coaching.

Analysis of variance

One-Way ANOVA

We are often interested in determining whether the means from more than two populations or groups are equal or not. To test whether the difference in means is statistically significant we can perform analysis of variance (ANOVA) using the R function **aov()**. If the ANOVA F-test shows there is a significant difference in means between the groups we may want to perform multiple comparisons between all pair-wise means to determine how they differ.

- The first step in our analysis is to graphically compare the means of the variable of interest across groups. It is possible to create side-by-side boxplots of measurements organized in groups using the function plot().

plot(response ~ factor, data=data_name)

Where response is the name of the response variable and factor the variable that separates the data into groups. Both variables should be contained in a data frame called data_name.

- The second step, the R function aov() can be used for fitting ANOVA models. The general form is

aov(response ~ factor, data=data_name)

Where response represents the response variable and factor the variable that separates the data into groups. Both variables should be contained in the data frame called data_name. Once the ANOVA model is fit, one can look at the results using the **summary()** function. This produces the standard ANOVA table.

Problem: A drug company tested three formulations of a pain relief medicine for migraine headache sufferers. For the experiment 27 volunteers were selected and 9 were randomly assigned to one of three drug formulations. The subjects were instructed to take the drug during their next migraine headache episode and to report their pain on a scale of 1 to 10 (10 being most pain).

Drug A	4 5 4 3 2 4 3 4 4
Drug B	6 8 4 5 4 6 5 8 6
Drug C	6 7 6 6 7 5 6 5 5

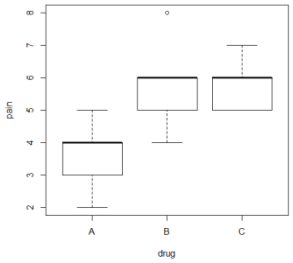
To make side-by-side boxplots of the response variable "pain" grouped by the factor variable "drug" we must first read in the data into the appropriate format.

> pain = c(4, 5, 4, 3, 2, 4, 3, 4, 4, 6, 8, 4, 5, 4, 6, 5, 8, 6, 6, 7, 6, 6, 7, 5, 6, 5, 5)
> drug = c(rep("A",9), rep("B",9), rep("C",9))
> migraine = data.frame(pain,drug)
> migraine

```
   pain drug
1     4    A
2     5    A
3     4    A
4     3    A
5     2    A
:     :    :
25    6    C
26    5    C
27    5    C
```

We can now make the boxplots by typing:

> **plot(pain ~ drug, data = migraine)**



From the boxplots it appears that the mean pain for drug A is lower than that for drugs B and C.

Next, the R function aov() can be used for fitting ANOVA models.

> results = aov(pain ~ drug, data = migraine)

> summary(results)

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
drug	2	28.22	14.111	11.91	0.000256 ***
Residuals	24	28.44	1.185		

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

Studying the output of the ANOVA table above we see that the F-statistic is 11.91 with a p-value equal to 0.0003. We clearly reject the null hypothesis of equal means for all three drug groups.

The ANOVA F-test answers the question whether there are significant differences in the K population means. However, it does not provide us with any information about how they differ. Therefore when you reject H_0 in ANOVA, additional analyses are required to determine what is driving the difference in means.

Multiple comparisons procedure is Tukey‟s method (a.k.a.Tukey's Honest Significance Test). The function TukeyHSD() creates a set of confidence intervals on the differences between means with the specified family-wise probability of coverage. The general form is

TukeyHSD(x, conf.level = 0.95)

Here x is a fitted model object (e.g., an aov fit) and conf.level is the confidence level.

> TukeyHSD(results, conf.level = 0.95)

Tukey multiple comparisons of means

95% family-wise confidence level

Fit: aov(formula = pain ~ drug, data = migraine)

$drug

	diff	lwr	upr	p adj
B-A	2.1111111	0.8295028	3.392719	0.0011107
C-A	2.2222222	0.9406139	3.503831	0.0006453
C-B	0.1111111	-1.1704972	1.392719	0.9745173

These results show that the B-A and C-A differences are significant(p=0.0011 and p=0.00065, respectively), while the C-B difference is not(p=0.97).

Two-Way ANOVA

Two-way ANOVA is used to compare the means of populations that are classified in two different ways, or the mean responses in an experiment with two factors. We fit two-way ANOVA models in R using the function **aov()**.

aov(Response~ FactorA+ FactorB)

fits a two-way ANOVA model without interactions. Here both Factor A and Factor B are categorical variables, while Response is quantitative.

Problem: A tea company appoints four salesmen A, B, C and D and observes their sales in three seasons – summer, winter and monsoon. The figures (in lakhs) of sales are given in the following table:

Season\ Salesman	A	B	C	D
Summer	36	36	21	35
Winter	28	29	31	32
Monsoon	26	28	29	29

i) Do the salesmen significantly differ in performance?
ii) Is there significant difference between the seasons?

```
> sales = c(36,36,21,35,28,29,31,32,26,28,29,29)
> f1 = c(rep(1:3, rep(4,3)))
> f2 = rep(c("A","B","C","D"),3)
> season = factor(f1)
> salesmen = factor(f2)
> ans = aov(sales~ season + salesmen)
> summary(ans)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
season	2	32	16.00	0.706	0.531
salesmen	3	42	14.00	0.618	0.629
Residuals	6	136	22.67		

Since p-value > LOS for both factors season and salesmen, therefore H_0 is accepted.

Salesmen do not differ significantly in their performance and also there is no significant difference between seasons.

R as a programming language

While R is perhaps best known as a statistical tool for analyzing data or for making graphs, it is also really useful as a simple programming language and compiler. In R, a program is just any group of commands that you wish to run as a set, to achieve some output.

Using Text Editors and ".R" Files in R

By using a text editor, we can write whole groups of commands and have the computer run them separately or all together. Further, text editors allow you to save your program for later use.

There are three different types of windows that are used by R: console, graphics, and text editor windows. The window where you enter line commands is the R Console. When you used the "plot" command, it opened a new window, which is the graphics window. Text editor windows are just simple text editors that are smart enough to interact with R.

On a PC, go to "File" and open "New script". To execute commands, either highlights the command(s) or put the cursor anywhere on that line and push the button in upper corner of the main R window for "Run line or selection."

Creating a new document (or script) opens a simple text editor in R. You can then enter multiple lines of commands that are not executed until you are ready. And, instead of executing commands one by one, you can execute them all at once or any set of them together. You can also save the file (usually as a "____.R" file) and rerun these commands at a later time.

A function is a program in R that can be run using a single command, such as "mean" or "plot". Such functions might be convenient for you to make if, for example, you want to run the same program over and over, perhaps just changing one or two variables.

The procedure for writing any other functions is similar, involving three key steps:

1. Define the function,
2. Load the function into the R session,
3. Use the function.

For R to be able to execute your function, it needs first to be read into memory. This is just like loading a library, until you do it the functions contained within it cannot be called.

There are two methods for loading functions into the memory:

1. Copy the function text and paste it into the console
2. Use the source() function to load your functions from file.

Put your functions into a file with an extension .R and save this file within the R folder in You can then read the function into memory by calling:

source("____.R")

```
factorial1 = function(N)
{
  prod =1
  for (i in 1:N)
    {
      prod = prod * i
    }
  return(prod)
}
ncr = function(n, r)
 {
   num = factorial1(n)
   deno1 = factorial1(r)
   deno2 = factorial1(n-r)
   ans = num/(deno1*deno2)
   return(ans)
 }
absolute = function(x)
 {
   if(x < 0)
    {
     cat("Modulus of ", x ," = ", -x, "\n")
    }
   else
    {
      cat("Modulus of ", x ," = ", x, "\n")
    }
}
```

Save above script as say first.R

```
>source("first.R")
> ncr(5,3)
[1] 10
> ncr(7,4)
[1] 35
> absolute(-5)
Modulus of  -5  =  5
> absolute(5)
Modulus of  5  =  5
```

Matrix Operation

To form a matrix you can use following syntax.

matrix(data =, nrow =, ncol= ,byrow="FALSE").

data: Actual data may be written in any of the variable or values by using function c().

nrow: Number of rows of a matrix

ncol: Number of columns of a matrix

byrow: It specifies whether matrix values are filled row wise or column wise. FALSE is by default i.e. column wise. If you want row wise then use TRUE.

For example,

```
>a <- c(1,2,3,4,5,6,7,8,9,10,11,12)
>A <- matrix(data=a, nrow=3, ncol=4, byrow="TRUE")
>A
     [,1]  [,2]  [,3]  [,4]
[1,] 1     2     3     4
[2,] 5     6     7     8
[3,] 9     10    11    12
```

Specific values in a vector or in a matrix are referenced using square brackets ([ ]). For example,

```
> A[2,4]
[1] 8
> A[3,]
[1] 9  10  11  12
> A[c(2,3),1]
[1] 5 9
```

Matrix operators are provided in the Table

Operation or Function	Description
A * B	Element-wise multiplication
A %*% B	Matrix multiplication
t(A)	Transpose
diag(x)	Creates diagonal matrix with elements of x in the principal diagonal
diag(A)	Returns a vector containing the elements of the principal diagonal
diag(k)	If k is a scalar, this creates a k x k identity matrix.
solve(A,b)	Returns vector x in the equation b = Ax
solve(A)	Inverse of A where A is a square matrix.
cbind(A,B,...)	Combine matrices(vectors) horizontally. Returns a matrix.
rbind(A,B,...)	Combine matrices(vectors) vertically. Returns a matrix.

Problem: Create matrix A and B using R command

$$A = \begin{bmatrix} -3 & 2 & 6 \\ 4 & 0 & -1 \\ 5 & 2 & 3 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 5 & 4 \\ -3 & 2 & 6 \\ 0 & -4 & 3 \end{bmatrix}$$

a) Create matrix A using vector a = (-3, 2, 6, 4, 0, -1, 5, 2, 3)
b) Create matrix B using vector b = (1, -3, 0, 5, 2, -4, 4, 6, 3)
c) Create matrix C which is **element wise** multiplication of matrices A and B
d) Create matrix D which is **matrix** multiplication of matrices A and B
e) Create matrix T which is transpose of matrix B
f) Combine matrices A and B horizontally
g) Combine matrices A and B vertically
h) Returns a vector containing the elements of the principal diagonal of matrix B
i) Returns a vector containing the elements of the principal diagonal of matrix B

Ans:

> a = c(-3, 2, 6, 4, 0, -1, 5, 2, 3)
> b = c(1, -3, 0, 5, 2, -4, 4, 6, 3)
a) >A = matrix(a, nrow = 3, ncol = 3, byrow = "TRUE")
>A
b) > B = matrix(b, nrow = 3, ncol = 3)
*c) > C = A * B*
d) > D = A %% B*
e) > T = t(B)
f) > P = solve(A)
g) >cbind(A,B)
h) >rbind(A,B)
i) >diag(B)

Problem: Create diagonal matrix D with elements of (-3, 0, 9, 2.3) in the principal diagonal

Ans:

> t = c(-3, 0, 9, 2.3)
> D = diag(t)

Problem: Create 4×4 identity matrix

Ans:

>diag(4)

Problem: Write R command to solve following system of equations

$$\begin{aligned} -3x_1 + 2x_2 + 6x_3 &= 4 \\ 4x_1 - x_3 &= -6 \\ 5x_1 + 2x_2 + 3x_3 &= 3 \end{aligned}$$

Ans:

> a = c(-3, 2, 6, 4, 0, -1, 5, 2, 3)
> A = matrix(a, nrow = 3, ncol = 3, byrow = "TRUE")
> b = c(4, -6, 3)
> x = solve(A,b)
>x

*** * * * ***