PRACTICAL NO. 1

Data Entry and Presentation Functions

Syntax 1:

read = read.csv("US_honey_dataset.csv")

print(read)

	X	state	colonies_number	yield_per_colony	production	stocks	average_price
1	0	Alabama	16000	58	928000	28000	62
2	1	Arizona	52000	79	4108000	986000	68
3	2	Arkansas	50000	60	3000000	900000	64
4	3	California	420000	93	39060000	4687000	60
5	4	Colorado	45000	60	2700000	1404000	68
6	5	Florida	230000	86	19780000	1780000	63
7	6	Georgia	70000	62	4340000	260000	69
8	7	Hawaii	8000	129	1032000	103000	55
9	8	Idaho	125000	48	6000000	1020000	65
10	9	Illinois	11000	74	814000	212000	102
11	10	Indiana	12000	63	756000	166000	68
12	11	Iowa	50000	68	3400000	612000	72
13	12	Kansas	17000	67	1139000	182000	71
14	13	Kentucky	3000	44	132000	30000	103
15	14	Louisiana	33000	119	3927000	275000	61
16	15	Maine	11000	45	495000	223000	100
17	16	Maryland	7000	35	245000	81000	114
18	17	Michigan	97000	92	8924000	3570000	72
19	18	Minnesota	165000	82	13530000	1218000	66
20	19	Mississippi	16000	70	1120000	146000	64
21	20	Missouri	23000	67	1541000	308000	65
22	21	Montana	106000	80	8480000	1781000	66
23	22	Nebraska	60000	73	4380000	1402000	68
24	23	Nevada	9000	29	261000	34000	96
25	24	NewJersey	8000	34	272000	57000	71
26	25	NewMexico	19000	65	1235000	247000	68
27	26	NewYork	70000	75	5250000	2100000	66
28	27	NorthCarolina	12000	52	624000	162000	81
29	28	NorthDakota	220000	108	23760000	3802000	63
30	29	Ohio	25000	62	1550000	930000	72
31	30	Oklahoma	5000	76	380000	141000	91

Syntax 2:

sdata = read.csv("US_honey_dataset.csv", header = TRUE, sep=",")

highspeed = subset(

sdata, sdata$speed == max(sdata$speed))

print(highspeed) sdata = read.csv("US_honey_dataset.csv", header = TRUE, sep=",")

highspeed = subset(

sdata, sdata$speed == max(sdata$speed))

print(highspeed)

```
[1] X                 state             colonies_number   yield_per_colony
[5] production        stocks            average_price     value_of_production
[9] year
<0 rows> (or 0-length row.names)
```

Syntax 3:

```
dataframe1 <- data.frame (
  Name = c("Juan", "Kay", "Jay", "Ray", "Aley"),
  Age = c(22, 15, 19, 30, 23),
  ID = c(101, 102, 103, 104, 105))
print(dataframe1)
print(max(dataframe1$Age))
print(min(dataframe1$ID))
```

```
  Name Age  ID
1 Juan  22 101
2  Kay  15 102
3  Jay  19 103
4  Ray  30 104
5 Aley  23 105
> print(max(dataframe1$Age))
[1] 30
> print(min(dataframe1$ID))
[1] 101
```

Syntax 4:

```
name=c("Adarsh","Ganesh","Chandan","Broo")
education=c("IPS","IAs","CS","CA")
age=c(21,22,23,24)
city=c("Than","Vira","Boisa","Andher")
df1=data.frame(name,education,age,city)
print(df1)
```

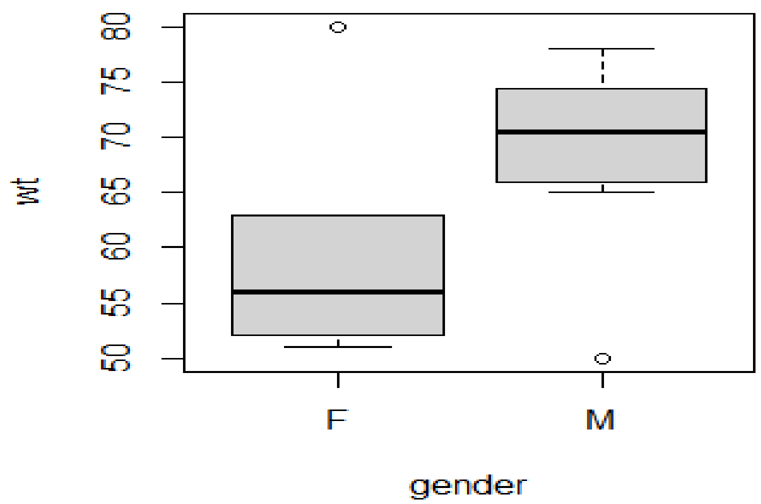
```
     name education age   city
1  Adarsh       IPS  21   Than
2  Ganesh       IAs  22   Vira
3 Chandan        CS  23  Boisa
4    Broo        CA  24 Andher
```

Data Presentation functions:

```
> wt = c(67,58,54,78,80,63,52,50,76,69,72,51,73,65)
> gender=c.factor('M','F','F','M','F','F','F','M','M','M','M','F','M','M')
> shapiro.test(wt)

        Shapiro-Wilk normality test

data:  wt
W = 0.93125, p-value = 0.3177
```



Practical No. 2

Measures of Central Tendance & Dispersion

```
> a=c(7,9,12,14,20,12,8,6,8,4)
> b=c(12,14,15,18,20,22,24,26,28,30)
> q=(a%*%b)/(sum(a))
> print(mean(q))
[1] 19.98
```

```
> data=c(20,25,30,35,40,45,50,55,60,65)
> q1= quantile(data,.25)
> q3= quantile(data,.75)
> iqr=q3-q1
> Qd=(q3-q1)/2
> cqd=(q3-q1)/(q3+q1)
> print(iqr)
 75%
22.5
> print(Qd)
  75%
11.25
> print(cqd)
      75%
0.2647059
```

```
> marks <- c(97, 78, 57, 64, 87)
> result <- quantile(marks, 0.70) # calculate 70th percentile of marks
> result2 <- quantile(marks, c(0.7, 0.5, 0.8))    # calculate 70th, 50th, 80th p
ercentile of marks
> print(result)
 70%
85.2
> print(result2)
 70%  50%  80%
85.2 78.0 89.0
```

```
> df1 <- data.frame (
+   Name = c("Addy", "Gannaya", "Jay", "Viru", "Aley"),
+   Age = c(22, 15, 19, 30, 23),
+   ID = c(101, 102, 103, 104, 105))
> result <- quantile(df1$Age, c(0.55, 0.27))   # calculate 55th and 27th percent
ile of the Age column
> print(result)
  55%   27%
22.20 19.24
```

Practical No. 3

Measures of Central Tendance & Dispersion

1.Arithmetic operations

```
> > x = 6
> > y = 17
> > x + y
[1] 23
> > x - y
[1] -11
> > x*y
[1] 102
> > y/x
[1] 2.833333
> > y%%x
[1] 5
> > y%/%x
[1] 2
> > y ^ x
[1] 24137569
```

2. Math Functions

```
> > max(8,99,15)
[1] 99
> > min(8,99,15)
[1] 8
> > sqrt(81)
[1] 9
> > abs(-8.9)
[1] 8.9
> > ceiling(2.6)
[1] 3
> > floor(2.6)
[1] 2
> > cos(4)
[1] -0.6536436
> > sin(4)
[1] -0.7568025
> > tan(4)
[1] 1.157821
> > log(4)
[1] 1.386294
> > exp(4)
[1] 54.59815 > >
```

3. Relational Operators

```
> > x<- 10

> > y<- 25

> x < y

[1] TRUE

> > x > y

[1] FALSE

> > x <= 5

[1] FALSE

> > y >= 20

[1] TRUE

> > y == 25

[1] TRUE

> > x != 10

[1] FALSE > > >
```

4. Operation on vectors

```
> > x = c(2,9,4)

> > y = c(6, 5, 1)

> > x + y

[1] 8 14 5

> > x

> y

[1] FALSE TRUE TRUE
```

5. Logical

```
> > x = c()

> > x = c(TRUE, FALSE, 0,6)

> > y = c(FALSE, TRUE, TRUE)

> > !x

[1] FALSE TRUE TRUE FALSE

> > x & y

[1] FALSE FALSE FALSE FALSE
```

Practical No. 4

Matrix and Data Frame operations and its functions

```
> A1=array(1:36,c(3,3,4))
> print(A1)
, , 1

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

, , 2

     [,1] [,2] [,3]
[1,]   10   13   16
[2,]   11   14   17
[3,]   12   15   18

, , 3

     [,1] [,2] [,3]
[1,]   19   22   25
[2,]   20   23   26
[3,]   21   24   27

, , 4

     [,1] [,2] [,3]
[1,]   28   31   34
[2,]   29   32   35
[3,]   30   33   36

> A1[,2,3]     #
[1] 22 23 24
> A1[,,2]
     [,1] [,2] [,3]
[1,]   10   13   16
[2,]   11   14   17
[3,]   12   15   18


> name=c("Ramesh","Suresh","Mukesh","Rakesh")
> education=c("10th","12th","8th","PHD")
> age=c(28,26,25,29)
> city=c("Thane","Virar","Boisar","Andheri")
> df=data.frame(name,education,age,city)
> print(df)
    name education age    city
1 Ramesh      10th  28   Thane
2 Suresh      12th  26   Virar
3 Mukesh       8th  25  Boisar
4 Rakesh       PHD  29 Andheri
> df[,1]
[1] "Ramesh" "Suresh" "Mukesh" "Rakesh"
> df[2,1]
[1] "Suresh"
> df[2,]
    name education age  city
2 Suresh      12th  26 Virar
> df[2:3,]
    name education age   city
2 Suresh      12th  26  Virar
3 Mukesh       8th  25 Boisar
```

```
> name=c("Adarsh","Ganesh","Chandan","Broo")
> education=c("IPS","IAs","CS","CA")
> age=c(21,22,23,24)
> city=c("Than","Vira","Boisa","Andher")
> df1=data.frame(name,education,age,city)
> print(df1)
     name education age   city
1  Adarsh       IPS  21   Than
2  Ganesh       IAs  22   Vira
3 Chandan        CS  23  Boisa
4    Broo        CA  24 Andher

> d2= rbind(df,df1)
> print(d2)
     name education age    city
1  Ramesh      10th  28   Thane
2  Suresh      12th  26   Virar
3  Mukesh       8th  25  Boisar
4  Rakesh       PHD  29 Andheri
5  Adarsh       IPS  21    Than
6  Ganesh       IAs  22    Vira
7 Chandan        CS  23   Boisa
8    Broo        CA  24  Andher

> df$city
[1] "Thane"   "Virar"   "Boisar"  "Andheri"

> cat("Dimension:", dim(airquality))
Dimension: 153 6> cat("\nRow:", nrow(airquality))

Row: 153> cat("\nCol:", ncol(airquality))

Col: 6>


> marks <- c(97, 78, 57, 64, 87)
> result <- quantile(marks, 0.70) # calculate 70th percentile of marks
> result2 <- quantile(marks, c(0.7, 0.5, 0.8))    # calculate 70th, 50th, 80th p
ercentile of marks
> print(result)
 70% 
85.2 
> print(result2)
 70%  50%  80% 
85.2 78.0 89.0 


> df1 <- data.frame (
+   Name = c("Addy", "Gannaya", "Jay", "Viru", "Aley"),
+   Age = c(22, 15, 19, 30, 23),
+   ID = c(101, 102, 103, 104, 105))
> result <- quantile(df1$Age, c(0.55, 0.27))   # calculate 55th and 27th percent
ile of the Age column
> print(result)
  55%   27% 
22.20 19.24 


> quan= c(10,35,40,5)
> df$quan= quan
> df1
     Name Age  ID
1    Addy  22 101
2 Gannaya  15 102
3     Jay  19 103
4    Viru  30 104
5    Aley  23 105
> df1$ID
[1] 101 102 103 104 105
```

```
> subset(df1, subset= price > 5)
        Name Age  ID
1       Addy  22 101
2    Gannaya  15 102
3        Jay  19 103
4       Viru  30 104
5       Aley  23 105
NA      <NA>  NA  NA
NA.1    <NA>  NA  NA
```

```
> vec1 <- c(28,64,63,43,56,46,87,34,73)
> vec2 <- c(53,37,29,45,68,33,76,49,30)
> vec3 <- c(12,44,36,75,36,93,34,64,18)
> vec1=((vec1-min(vec1))/(max(vec1)-min(vec1)))
> vec1
[1] 0.0000000 0.6101695 0.5932203 0.2542373 0.4745763 0.3050847 1.0000000 0.1016
949 0.7627119
```

```
> rescale=function(vec){
+   vec=((vec-min(vec))/(max(vec)-min(vec)))
+   return(vec)
+ }
> rescale(vec1)
[1] 0.0000000 0.6101695 0.5932203 0.2542373 0.4745763 0.3050847 1.0000000 0.1016
949 0.7627119
> rescale(vec2)
[1] 0.51063830 0.17021277 0.00000000 0.34042553 0.82978723 0.08510638 1.00000000
0.42553191 0.02127660
> rescale(vec3)
[1] 0.00000000 0.39506173 0.29629630 0.77777778 0.29629630 1.00000000 0.27160494
0.64197531 0.07407407
```

```
> vec1 = c(28,64,63,43,56,46,87,34,73)
> reciprocal = function(vec) vec = 1/vec
> rvec1 = reciprocal(vec1)
> rvec1
[1] 0.03571429 0.01562500 0.01587302 0.02325581 0.01785714 0.02173913 0.01149425
0.02941176 0.01369863
```

```
> a= matrix(1:9,3,3)
> b = matrix(10:18,3,3)
> print(a%*%b)  # % sign is imp for multipying corresponding elements
     [,1] [,2] [,3]
[1,]  138  174  210
[2,]  171  216  261
[3,]  204  258  312
>      #OR
> a= matrix(1:8,nrow=2)
> b = matrix(8:19,nrow=4)
> print(a%*%b)
     [,1] [,2] [,3]
[1,]  162  226  290
[2,]  200  280  360
```

Practical No. 5

Correlation and Regression Functions

```
> x=c(1,2,3,4,5,6,7)
> y=c(1,3,6,2,7,4,5)
> result=cor(x,y,method="kendall")
> print(result)
[1] 0.4285714
> result=cor.test(x,y,method="pearson")
> print(result)

        Pearson's product-moment correlation

data:  x and y
t = 1.4186, df = 5, p-value = 0.2152
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.3643187  0.9183058
sample estimates:
      cor
0.5357143

> shapiro.test(mtcars$wt)

        Shapiro-Wilk normality test

data:  mtcars$wt
W = 0.94326, p-value = 0.09265


> res <- cor.test(mtcars$hp, mtcars$mpg, method = "pearson")
> res

        Pearson's product-moment correlation

data:  mtcars$hp and mtcars$mpg
t = -6.7424, df = 30, p-value = 1.788e-07
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.8852686 -0.5860994
sample estimates:
       cor
-0.7761684

> x=c(10,20,30,40,50,60,70)
> y=c(8,6,14,16,10,20,24)
> z=cor.test(x,y,method='pearson')
> z

        Pearson's product-moment correlation

data:  x and y
t = 3.6145, df = 5, p-value = 0.01531
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.2707625 0.9774828
sample estimates:
      cor
0.8504201

> p=cov(x,y,method='pearson')
> p
[1] 120
>
> Father=c(65,66,67,67,68,69,71,73)
> Daug=c(67,68,64,69,72,70,69,73)
> z=cor.test(Father,Daug,method='pearson')
```

```
> z

        Pearson's product-moment correlation

data:  Father and Daug
t = 2.0717, df = 6, p-value = 0.08369
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.1080788  0.9281049
sample estimates:
      cor
0.6457766

> a=cov(Father,Daug,method='pearson')
> a
[1] 4.857143
>
> 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
>
> Hist=c(35,23,47,17,10,43,96,28)
> Algebra=c(30,33,45,23,84,91,24,31)
> relation=cor.test(Hist,Algebra)
> relation

        Pearson's product-moment correlation

data:  Hist and Algebra
t = -0.62361, df = 6, p-value = 0.5558
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.8104841  0.5543273
sample estimates:
       cor
-0.2467188

> res=cor.test(Hist,Algebra,method='spearman')
> res

        Spearman's rank correlation rho

data:  Hist and Algebra
S = 86, p-value = 0.9768
alternative hypothesis: true rho is not equal to 0
sample estimates:
        rho
-0.02380952


> print(mtcars)
                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
```

```
Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2

> my_data = mtcars[, c(1,3,4,5,6,7)]
> head(my_data, 6)
                   mpg disp  hp drat    wt  qsec
Mazda RX4         21.0  160 110 3.90 2.620 16.46
Mazda RX4 Wag     21.0  160 110 3.90 2.875 17.02
Datsun 710        22.8  108  93 3.85 2.320 18.61
Hornet 4 Drive    21.4  258 110 3.08 3.215 19.44
Hornet Sportabout 18.7  360 175 3.15 3.440 17.02
Valiant           18.1  225 105 2.76 3.460 20.22


> res = cor(my_data)
> round(res, 2)
       mpg  disp    hp  drat    wt  qsec
mpg   1.00 -0.85 -0.78  0.68 -0.87  0.42
disp -0.85  1.00  0.79 -0.71  0.89 -0.43
hp   -0.78  0.79  1.00 -0.45  0.66 -0.71
drat  0.68 -0.71 -0.45  1.00 -0.71  0.09
wt   -0.87  0.89  0.66 -0.71  1.00 -0.17
qsec  0.42 -0.43 -0.71  0.09 -0.17  1.00


> a = c(2,4,6,8,10)
> b = c(1,11,3,33,5)
> print(cov(a, b, method = "pearson"))
[1] 15


> a <- c(2,4,6,8)
> b <- c(1,11,3,33)
> covar = cov(a,b)
> print(covar)
[1] 29.33333


> x = c(1, 3, 5, 10)
> y = c(2, 4, 6, 20)
> # Print covariance using different methods
> print(cov(x, y))  #bydefault pearson
[1] 30.66667
> print(cov(x, y, method = "pearson"))
[1] 30.66667
> print(cov(x, y, method = "kendall"))
[1] 12
> print(cov(x, y, method = "spearman"))
[1] 1.666667
```

Practical No. 6

Probability and Conditional Probability functions

```
data=data.frame(A = c("a1","a1","a1","a2","a2","a2"),B=c("b1","b1","b2","b1","b2","b2"))
> contigency_table = table(data$A,data$B)
> conditional_probability_table=prop.table(contigency_table,margin=1)
> print(conditional_probability_table)

          b1        b2
  a1 0.6666667 0.3333333
  a2 0.3333333 0.6666667
```

```
> weather_data=data.frame(Cloudy=c("Yes","Yes","No","No"),Rain=c("Yes","No","Yes","No"),Frequency=c(30,
20,10,40))
> weather_data
  Cloudy Rain Frequency
1    Yes  Yes        30
2    Yes   No        20
3     No  Yes        10
4     No   No        40
```

```
> # total freq. of cloudy days
> total_cloudy=sum(weather_data$Frequency[weather_data$Cloudy=="Yes"])
> #freq. of rainy days when its cloudy
> rainy_and_cloudy=weather_data$Frequency[weather_data$Cloudy == "Yes" & weather_data$Rain == "Yes"
]
> #conditional prob. of rain given clouds
> P_rain_given_cloudy = rainy_and_cloudy/total_cloudy
> print(P_rain_given_cloudy)
[1] 0.6
```

```
> student_data = data.frame(Attendance=c("High","High","Low","Low"),Pass=c("Yes","No","Yes","No"),Freque
ncy=c(80,20,30,70))
> total_high_attendance = sum(student_data$Frequency[student_data$Attendance == "High"])
> pass_and_high_attendance =student_data$Frequency[student_data$Attendance == "High" & student_data
$Pass == "Yes"]
> P_pass_given_high_attendance = pass_and_high_attendance / total_high_attendance
> print(P_pass_given_high_attendance)
[1] 0.8
```

```
> total_low_attendance = sum(student_data$Frequency[student_data$Attendance == "Low"])
> pass_and_low_attendance =student_data$Frequency[student_data$Attendance == "Low" & student_data$
Pass == "Yes"]
> P_pass_given_low_attendance = pass_and_low_attendance / total_low_attendance
> print(P_pass_given_low_attendance)
```

```
[1] 0.3
```

```
> bayesTheorem=function(pA,pB,pBA){
+   pAB=pA*pBA/pB
+   return(pAB)
+ }
> pRain=0.2
> pCloudy=0.4
> pCloudyRain=0.85
> bayesTheorem(pRain,pCloudy,pCloudyRain)
[1] 0.425
```

```
> BayesTheorem=function(P_EventA,P_EventB,P_EventBGivenEventA){
+   P_EventAGivenEventB=P_EventA*P_EventBGivenEventA / P_EventB
+   return(P_EventAGivenEventB)
+ }
> PRain=0.30
> PWalk=0.50
> PWalkGrain=0.10
> BayesTheorem(PRain,PWalk,PWalkGrain)
[1] 0.06
```

```
> psunny = 0.65
> pcloudy = 0.20
> pcloudysunny = 0.30
> BayesTheorem(psunny,pcloudy,pcloudysunny)
[1] 0.975
```

Practical No. 7

Binomial and Poisson Distribution & Plotting of its PMF, PDF, CDF

```
> dpois(2, lambda = 3)  #first argument is x= prob & 2nd is lambda
[1] 0.2240418

> dpois(6,6)
[1] 0.1606231

> dpois(0:10, 5)
 [1] 0.006737947 0.033689735 0.084224337 0.140373896 0.175467370 0.175467370 0.146222808
 [8] 0.104444863 0.065278039 0.036265577 0.018132789

> ppois(2,3)
[1] 0.4231901

> ppois(6,6)
[1] 0.6063028

> rpois(2,3)
[1] 5 5

> rpois(6,6)
[1]  6 11  4  4 10  4

> y = c(0.1,0.5,0.1,0.2)
> qpois(y,2)
[1] 0 2 0 1

> qpois(y,6)
[1] 3 6 3 4

> #lambda=12cars crossing bridge on an avg per min.
> #find prob of having seventeen or more cars crossing the bridge in a particular time
> #so x=17-1=16
> ppois(16, 12, lower.tail = FALSE)
[1] 0.101291

> #Prob of making 2 to 4 sales in a week if the avg sales rate is 3 per week
> dpois(2,3)+dpois(3,3)+dpois(3,4)  #OR
[1] 0.6434504

> ppois(4,3)-dpois(1,3)-dpois(0,3)
[1] 0.616115

> #baseball player has a p=0.3 batting avg. find prob of x<=150 hits in n=500 at bats
> ppois(150, 0.300*500)
[1] 0.5216972
```

```
> # p=0.1%, n=1000, prob of 0 patient died
> dpois(0.001*1000, 1)
[1] 0.3678794

> ppois(1, (1/2000)*1000, lower.tail = FALSE)
[1] 0.09020401

> dpois(6, (1/1000)*3000)
[1] 0.05040941

> #3rd Question
> dpois(0, 0.5) #no defective
[1] 0.6065307

> ppois(1, 0.5, lower=FALSE)  #2 or more defective
[1] 0.09020401
> ppois(2, 0.5) # more than 2 defective
[1] 0.9856123

> #4th question
> ppois(5,3, lower=FALSE)
[1] 0.08391794
> 1-ppois(5,3)
[1] 0.08391794

> #Bacteria question
> dpois(0,6)
[1] 0.002478752
> dpois(1,6)
[1] 0.01487251
> dpois(2,6)
[1] 0.04461754
> dpois(3,6)
[1] 0.08923508
> ppois(3,6, lower.tail = FALSE)  #less than 4
[1] 0.8487961

> #Confusion. Check the Council question
> ppois(5, 5, lower.tail = FALSE) #OR
[1] 0.3840393
> 1-ppois(5,5)
[1] 0.3840393
```

Practical No. 8

Normal Distribution & Plotting of its PMF, PDF, CDF

```
> pnorm(70,50,15)-pnorm(49,50,15) # 50-1= 49 used because if used 50 then it will start from 51 but we want
50 also
[1] 0.4353652

> pnorm(100, 90, 10, lower.tail = FALSE)
[1] 0.1586553

> #Mean=30, SD=4
> pnorm(39,30,4)  #Que=x<40 so that means x<=39
[1] 0.9877755

> pnorm(21,30,4,lower=FALSE) #x>21 so that means x=>22. Hence x should be atleast 22.
[1] 0.9877755

> pnorm(34,30,4)-pnorm(30,30,4,lower=FALSE) #30<x<35
[1] 0.3413447

> #Class Questions
> pnorm(585,500,100)
[1] 0.8023375

> pnorm(5.04,5,0.02)-pnorm(4.95,5,0.02)
[1] 0.9710402

> pnorm(5.02,5,0.02)-pnorm(4.97,5,0.02)
[1] 0.7745375

> #Birthweight of babies
> #Mean=7.5, SD=1, prob >7
> pnorm(7,7.5,1,lower=FALSE)
[1] 0.6914625

> #Height of Males
> pnorm(68,70,3)
[1] 0.2524925
> pnorm(70,70,3, lower=FALSE)
[1] 0.5

> #shoe size
> pnorm(9,10,1) #x<10
[1] 0.1586553

> #blood pressure
> dnorm(100,80,20)  #x=100
[1] 0.01209854
> pnorm(99,80,20) #X<100
[1] 0.8289439
```

```
> pnorm(100,80,20,lower=FALSE)  #x>100
[1] 0.1586553

> dnorm(100,80,20)+ pnorm(99,80,20)+ pnorm(100,80,20,lower=FALSE)
[1] 0.9996977

> #fraudulent transaction
> pbinom(1,50,0.02,lower=FALSE) #x>1
[1] 0.2642286

> pbinom(2,50,0.02,FALSE) #x>=3
[1] 0.07842775

> #shopping returns per week
> pbinom(5,50,0.1,FALSE)  #x>5
[1] 0.383877

> pbinom(19,50,0.1) #x<20. If we took 20 then it will indicate 21. So we will take 19 to get 20
[1] 1

> #Mean=12, SD=2
> pnorm(12,12,2)-pnorm(6,12,2)  #between 7 to 12
[1] 0.4986501

> pnorm(80,70,10)
[1] 0.8413447

> pnorm(59,70,10,FALSE) #x>=60 i.e Atleast 60
[1] 0.8643339

> pnorm(59,70,10)
[1] 0.1356661

> pnorm(499,450,100)-pnorm(400,450,100) #400<x<500
[1] 0.3793955
```

Practical no. 9

Hypothesis Testing for different conditions functions

```
x=c(109,112,106,110,108,115,99,108,104,111)
> shapiro.test(x)

        Shapiro-Wilk normality test

data:  x
W = 0.96555, p-value = 0.8468

> t.test(x, mu=100, alpha= 0.05, alternate='two.sided')

        One Sample t-test

data:  x
t = 5.8047, df = 9, p-value = 0.0002579
alternative hypothesis: true mean is not equal to 100
95 percent confidence interval:
 105.0044 111.3956
sample estimates:
mean of x
   108.2

> t.test(x, alpha= 0.05, alternate='two.sided')

        One Sample t-test

data:  x
t = 76.594, df = 9, p-value = 5.577e-14
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 105.0044 111.3956
sample estimates:
mean of x
   108.2

> sample=c(12,14,15,17,19,20,22,25,28,30)
> hypothesized_mean=18
> t_test= t.test(sample, mu=hypothesized_mean)
> print(t_test)      #We fail

        One Sample t-test

data:  sample
t = 1.1531, df = 9, p-value = 0.2786
alternative hypothesis: true mean is not equal to 18
95 percent confidence interval:
 15.88408 24.51592
sample estimates:
```

```
mean of x = 20.2

> x=c(3,7,11,0,7,0,4,5,6,2)
> t.test(x)              #Output: We fail to accept the null hypothesis bcoz value <0.05

        One Sample t-test

data:  x
t = 4.1367, df = 9, p-value = 0.002534
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 2.0392 6.9608
sample estimates:
mean of x
      4.5

> set.seed(150)
> data=data.frame(value=rnorm(30, mean=50, sd=10))
> test=t.test(data$value, mu=50)
> test                  #Output: We fail to reject the null hypothesis

        One Sample t-test

data:  data$value
t = 0.57321, df = 29, p-value = 0.5709
alternative hypothesis: true mean is not equal to 50
95 percent confidence interval:
 47.02585 55.29045
sample estimates:
mean of x
 51.15815

>
> set.seed(123)
> sugar_cookie=rnorm(30,mean=9.99, sd=0.04)
> head(sugar_cookie)
[1]  9.967581  9.980793 10.052348  9.992820  9.995172 10.058603
> t.test(sugar_cookie, mu=10)     #We fail to reject the null hypothesis because Value >0.05

        One Sample t-test

data:  sugar_cookie
t = -1.6588, df = 29, p-value = 0.1079
alternative hypothesis: true mean is not equal to 10
95 percent confidence interval:
  9.973463 10.002769
sample estimates:
mean of x
 9.988116
```

```
> set.seed(123)
> before=rnorm(7, mean=50000, sd=50)
> after=rnorm(7, mean=50075, sd=50)
> t.test(before, after, paired= T)    #or write var.equal in place of paired

        Paired t-test

data:  before and after
t = -2.6102, df = 6, p-value = 0.04011
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 -97.618586  -3.152003
sample estimates:
mean difference
      -50.38529

>
> set.seed(0)
> shop1 = rnorm(50, mean=140, sd=4.5)
> shop2 = rnorm(50, mean=150, sd=4)
> t.test(shop1, shop2, paired = F)       #We fail to accept bcoz p <0.05

        Welch Two Sample t-test

data:  shop1 and shop2
t = -13.158, df = 94.837, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -11.483435  -8.472434
sample estimates:
mean of x mean of y
 140.1077  150.0856

>
> set.seed(0)
> sweet1 = c(rnorm(100, mean=14, sd=0.3))
> sweet2 = c(rnorm(100, mean=13, sd=0.2))
> t.test(sweet1, sweet2, paired = T)       #We fail to accept bcoz p <0.05

        Paired t-test

data:  sweet1 and sweet2
t = 33.06, df = 99, p-value < 2.2e-16
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 0.9549378 1.0768839
sample estimates:
mean difference
       1.015911
```

```
> before=c(12.2,14.6,13.4,11.2,12.7,10.4,15.8,13.9,9.5,14.2)
> after=c(13.5,15.2,13.6,12.8,13.7,11.3,16.5,13.4,8.7,14.6)
> data=data.frame(subject=rep(c(1:10),2), time=rep(c("before","after"), each=10), score=c(before,after))
> data
  subject   time score
1       1 before  12.2                11      1  after  13.5
2       2 before  14.6                12      2  after  15.2
3       3 before  13.4                13      3  after  13.6
4       4 before  11.2                14      4  after  12.8
5       5 before  12.7                15      5  after  13.7
6       6 before  10.4                16      6  after  11.3
7       7 before  15.8                17      7  after  16.5
8       8 before  13.9                18      8  after  13.4
9       9 before   9.5                19      9  after   8.7
10     10 before  14.2                20     10  after  14.6
> t.test(score~time, data=data, paired=T)

        Paired t-test

data:  score by time
t = 2.272, df = 9, p-value = 0.0492
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 0.002344255 1.077655745
sample estimates:
mean difference
          0.54

>
> #Two Sample test bcoz paired is false
> set.seed(125)
> g1=c(rnorm(100, mean=24, sd=3))
> g2=c(rnorm(100, mean=43, sd=2.4))
> t.test(g1,g2)        #bydefault paired = False if paired not used

        Welch Two Sample t-test

data:  g1 and g2
t = -47.765, df = 179.99, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -19.51569 -17.96722
sample estimates:
mean of x mean of y
 24.30063  43.04208
```

Practical No. 10

Analysis of Variance functions

```
df = read.csv("anova.csv")   #Easy work do by csv file
> df
   runs players
1   18       A
2   25       A
3   20       A
4   11       A
5   18       A
6   24       A
7    9       B
8    7       B
9    8       B
10  13       B
11  11       B
12  30       B
13  15       C
14  14       C
15  12       C
16  30       C
17  25       C
18  17       C
19  22       D
20  18       D
21   9       D
22  12       D
23  15       D
24  16       D
25   9       E
26  28       E
27  15       E
28  20       E
29  16       E
30  20       E
31  10       F
32  13       F
33  17       F
34  23       F
35   8       F
36  30       F

> model=aov(runs~players, data=df)
> model
Call:
   aov(formula = runs ~ players, data = df)

Terms:
                 players Residuals
Sum of Squares  171.2222 1404.3333
Deg. of Freedom        5        30

Residual standard error: 6.841865
Estimated effects may be unbalanced
> summary(model)
            Df Sum Sq Mean Sq F value Pr(>F)
players      5  171.2   34.24   0.732  0.605
Residuals   30 1404.3   46.81

> data=c(6,7,3,8,5,5,3,7,5,4,3,4)
> land=factor(c('A','A','A','A','B','B','B','B','C','C','C','C'))
> df= aov(data~land)
> summary(df)
            Df Sum Sq Mean Sq F value Pr(>F)
land         2      8   4.000     1.5  0.274
Residuals    9     24   2.667
```

```
> data=c(16,10,11,9,9,10,9,14,12,11,15,8,8,10,18,12,6,13,13,12,13,11,10,7,14)
> group=factor(c('A','B','C','D','E','E','C','A','B','D','B','D','E','C','A','D','E','B','A','C','C','A','D','E','B'))
> summary(aov(data~group))
            Df Sum Sq Mean Sq F value   Pr(>F)    
group        4  122.6   30.64   8.281 0.000414 ***
Residuals   20   74.0    3.70                     
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
> data=c(55,65,58,59,72,66,57,57,47,53,74,58)
> wheat=factor(c('w1','w1','w1','w1','w2','w2','w2','w2','w3','w3','w3','w3'))
> summary(aov(data~wheat))
            Df Sum Sq Mean Sq F value Pr(>F)
wheat        2   54.2   27.08   0.395  0.685
Residuals    9  616.7   68.53               
```

```
> data=c(5,7,9,4,8,6,4,5,6,7,4,7)
> salesman=factor(c('A','A','A','B','B','B','C','C','C','D','D','D'))
> summary(aov(data~salesman))
            Df Sum Sq Mean Sq F value Pr(>F)
salesman     3      6       2   0.667  0.596
Residuals    8     24       3               
```

```
> mileage=c(13,12,12,11,11,10,11,13,12,10,11,9,13,11,12,10,14,11,13,10,13,10,14,8)
> brand=factor(c('W','W','W','W','W','W','X','X','X','X','X','X','Y','Y','Y','Y','Y','Y','Z','Z','Z','Z','Z','Z'))
> summary(aov(mileage~brand))
            Df Sum Sq Mean Sq F value Pr(>F)
brand        3   2.17  0.7222   0.269  0.847
Residuals   20  53.67  2.6833               
```

```
> sales=c(23,20,40,34,35,34,30,34,20,29,45,45,25,28,30,40)
> emp=factor(c('A','B','D','A','C','A','B','D','B','C','C','A','C','D','B','D'))
> summary(aov(sales~emp))
            Df Sum Sq Mean Sq F value Pr(>F)
emp          3    270   90.00   1.617  0.237
Residuals   12    668   55.67               
```

```
> df=read.csv('crop.data.csv')
> df
   density block fertilizer    yield
1        1     1          1 177.2287
2        2     2          1 177.5500
3        1     3          1 176.4085
4        2     4          1 177.7036
5        1     1          1 177.1255
6        2     2          1 176.7783
7        1     3          1 176.7463
8        2     4          1 177.0612
9        1     1          1 176.2749
10       2     2          1 177.9672
11       1     3          1 176.6013
12       2     4          1 177.0305
13       1     1          1 177.4795
14       2     2          1 176.8741
15       1     3          1 176.1144
16       2     4          1 176.0084
17       1     1          1 176.1083
18       2     2          1 178.3574
19       1     3          1 177.2624
20       2     4          1 176.9188
21       1     1          1 176.2390
22       2     2          1 176.5731
```

23	1	3	1	176.0393
24	2	4	1	176.8179
25	1	1	1	176.1606
26	2	2	1	177.2264
27	1	3	1	175.9385
28	2	4	1	177.1649
29	1	1	1	175.3608
30	2	2	1	177.2770
31	1	3	1	175.9454
32	2	4	1	175.8828
33	1	1	2	176.4793
34	2	2	2	176.0443
35	1	3	2	177.4125
36	2	4	2	177.3608
37	1	1	2	177.3855
38	2	2	2	176.9758
39	1	3	2	177.3798
40	2	4	2	177.9980
41	1	1	2	176.4349
42	2	2	2	176.9333
43	1	3	2	175.9835
44	2	4	2	177.0341
45	1	1	2	176.4368
46	2	2	2	176.0677
47	1	3	2	177.1210
48	2	4	2	177.1977
49	1	1	2	176.6037
50	2	2	2	177.2082
51	1	3	2	177.1488
52	2	4	2	176.8191
53	1	1	2	176.9991
54	2	2	2	178.1346
55	1	3	2	176.4292
56	2	4	2	176.6683
57	1	1	2	176.8959
58	2	2	2	177.7795
59	1	3	2	176.4145
60	2	4	2	176.8789
61	1	1	2	177.5807
62	2	2	2	176.9573
63	1	3	2	175.7475
64	2	4	2	177.3526
65	1	1	3	177.1042
66	2	2	3	178.0796
67	1	3	3	176.9034
68	2	4	3	177.5403
69	1	1	3	177.0327
70	2	2	3	178.2860
71	1	3	3	176.4054
72	2	4	3	176.4308
73	1	1	3	177.3963
74	2	2	3	176.9256
75	1	3	3	177.0550
76	2	4	3	177.3442
77	1	1	3	177.1284
78	2	2	3	177.1683
79	1	3	3	176.3539
80	2	4	3	179.0609
81	1	1	3	176.3005
82	2	2	3	177.5934
83	1	3	3	177.1152
84	2	4	3	177.7945
85	1	1	3	177.0040
86	2	2	3	178.0369
87	1	3	3	177.7014
88	2	4	3	177.6328
89	1	1	3	177.6523
90	2	2	3	177.1004
91	1	3	3	177.1880
92	2	4	3	177.4053
93	1	1	3	178.1416
94	2	2	3	177.7106
95	1	3	3	177.6873
96	2	4	3	177.1182

```
> df$density = factor(df$density)
> df$block = factor(df$block)
> df$fertilizer = factor(df$fertilizer)
> summary(aov(yield~density+fertilizer, data=df))   #We fail to accept the null hypothesis
           Df Sum Sq Mean Sq F value   Pr(>F)
density     1  5.122   5.122  15.316 0.000174 ***
fertilizer  2  6.068   3.034   9.073 0.000253 ***
Residuals  92 30.765   0.334
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Practical No. 11

Non-Parametric Tests functions

```
> #Q) Are color equally common?
> tulip = c(81,50,27)
> ress = chisq.test(tulip, p=c(1/3,1/3,1/3))  # OR p=c(rep(1/2, 3)) i.e. repeat
> ress

        Chi-squared test for given probabilities

data:  tulip
X-squared = 27.886, df = 2, p-value = 8.803e-07

> #Q) Comparing observed to expected proportions
> tulip = c(81,50,27)
> res = chisq.test(tulip, p=c(1/2,1/3,1/6))
> res

        Chi-squared test for given probabilities

data:  tulip
X-squared = 0.20253, df = 2, p-value = 0.9037

> obs = c(12,8,15,5,16,4)
> a= chisq.test(obs, p=c(rep(1/6,6)))
> a

        Chi-squared test for given probabilities

data:  obs
X-squared = 13, df = 5, p-value = 0.02338

> low=c(25,30)
> med=c(35,10)
> hi=c(50,50)
> dframe=data.frame(low,med,hi)
> rownames(dframe)= c("male","female")
> cat("Contingency table sex vs. usage: \n\n")
Contingency table sex vs. usage:

> print(dframe)
    low med hi
male    25  35 50
female  30  10 50
> m=chisq.test(dframe)
> m

        Pearson's Chi-squared test

data:  dframe
X-squared = 12.468, df = 2, p-value = 0.001961
```

```
> #First method
> demo=c(120,110)
> repu=c(90,95)
> inde=c(40,45)
> dframe=data.frame(demo,repu,inde)
> rownames(dframe)= c("male","female")
> print(dframe)
       demo repu inde
male    120   90   40
female  110   95   45
> a=chisq.test(dframe)
> a

        Pearson's Chi-squared test

data:  dframe
X-squared = 0.86404, df = 2, p-value = 0.6492

> #OR Second Method
> data=matrix(c(120,110,90,95,40,45), ncol=3, byrow=TRUE)
> colnames(data)= c("Demo","Rep","Inde")
> rownames(data)= c("Male","Female")
> data1= as.table(data)
> data1
       Demo Rep Inde
Male    120 110   90
Female   95  40   45
> a=chisq.test(dframe)
> a

        Pearson's Chi-squared test

data:  dframe
X-squared = 0.86404, df = 2, p-value = 0.6492

> obs_freq=c(212,147,103,50,46,42)
> exp_freq=c(rep(100,6))
> chisq.test(obs_freq, p=c(rep(1/6,6)))

        Chi-squared test for given probabilities

data:  obs_freq
X-squared = 235.42, df = 5, p-value < 2.2e-16
```