

Experiment-1

Aim: Write a Program to implement the basic matrix operations.

Theory: Matrix operations mainly involve three algebraic operations, which are the addition of matrices, subtraction of matrices, and multiplication of matrices. Matrix is a rectangular array of numbers or expressions arranged in rows and columns. Important applications of matrices can be found in Mathematics.

R Script:

```
# Defining a matrix
A <- matrix(data=c(-3,2,893,0.17),nrow=2,ncol=2)
print(A)
# Printing a matrix column-wise
A<-matrix(data=c(1,2,3,4,5,6),nrow=2,ncol=3,byrow=FALSE)
print(A)

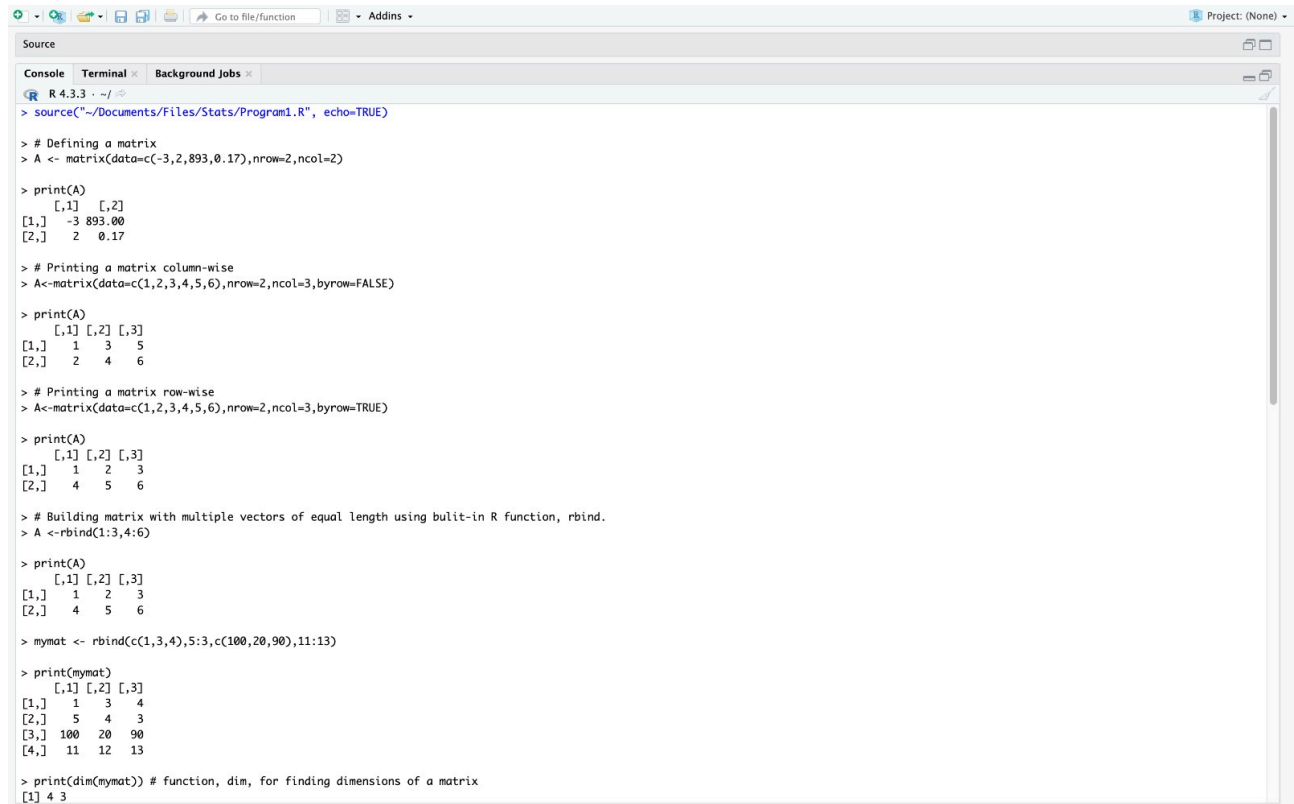
# Printing a matrix row-wise
A<-matrix(data=c(1,2,3,4,5,6),nrow=2,ncol=3,byrow=TRUE)
print(A)
# Building matrix with multiple vectors of equal length using built-in R function, rbind.
A <-rbind(1:3,4:6)
print(A)
mymat <- rbind(c(1,3,4),5:3,c(100,20,90),11:13)
print(mymat)
print(dim(mymat)) # function, dim, for finding dimensions of a matrix

# Transpose of a matrix
A <- rbind(c(2,5,2),c(6,1,4))
print(A)
print(t(A))
print(t(t(A))) # Getting back the original matrix
print(diag(x=3)) # A 3-D Identity matrix
B <- rbind(c(0,1,0),c(1,0,1))
print (A+B) # Addition of matrices
print (A-B) # Subtraction of matrices
print(A*B) # Element-wise multiplication of matrices
B <- rbind(c(1,0),c(0,0),c(0,0))
print(A%*%B) # Usual Product or multiplication of matrices

# Inverse of matrix using solve()
A <- matrix(c(3,4,1,2),nrow=2,ncol=2)
```

```
print(solve(A))
```

Output:



```
R 4.3.3 ~ /
> source("~/Documents/Files/Stats/Program1.R", echo=TRUE)

> # Defining a matrix
> A <- matrix(data=c(-3,2,893,0.17),nrow=2,ncol=2)

> print(A)
      [,1] [,2]
[1,]  -3 893.00
[2,]   2  0.17

> # Printing a matrix column-wise
> A<-matrix(data=c(1,2,3,4,5,6),nrow=2,ncol=3,byrow=FALSE)

> print(A)
      [,1] [,2] [,3]
[1,]   1   3   5
[2,]   2   4   6

> # Printing a matrix row-wise
> A<-matrix(data=c(1,2,3,4,5,6),nrow=2,ncol=3,byrow=TRUE)

> print(A)
      [,1] [,2] [,3]
[1,]   1   2   3
[2,]   4   5   6

> # Building matrix with multiple vectors of equal length using built-in R function, rbind.
> A <-rbind(1:3,4:6)

> print(A)
      [,1] [,2] [,3]
[1,]   1   2   3
[2,]   4   5   6

> mymat <- rbind(c(1,3,4),5:3,c(100,20,90),11:13)

> print(mymat)
      [,1] [,2] [,3]
[1,]   1   3   4
[2,]   5   4   3
[3,] 100  20  90
[4,]  11  12  13

> print(dim(mymat)) # function, dim, for finding dimensions of a matrix
[1] 4 3
```

```
R 4.3.3 ~ /

Source
Console Terminal Background Jobs
R 4.3.3 ~ /

> # Transpose of a matrix
> A <- rbind(c(2,5,2),c(6,1,4))

> print(A)
      [,1] [,2] [,3]
[1,]    2    5    2
[2,]    6    1    4

> print(t(A))
      [,1] [,2]
[1,]    2    6
[2,]    5    1
[3,]    2    4

> print(t(t(A))) # Getting back the original matrix
      [,1] [,2] [,3]
[1,]    2    5    2
[2,]    6    1    4

> print(diag(x=3)) # A 3-D Identity matrix
      [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1

> B <- rbind(c(0,1,0),c(1,0,1))

> print (A+B) # Addition of matrices
      [,1] [,2] [,3]
[1,]    2    6    2
[2,]    7    1    5

> print (A-B) # Subtraction of matrices
      [,1] [,2] [,3]
[1,]    2    4    2
[2,]    5    1    3

> print(A*B) # Element-wise multiplication of matrices
      [,1] [,2] [,3]
[1,]    0    5    0
[2,]    6    0    4

> B <- rbind(c(1,0),c(0,0),c(0,0))
```

```
> print(A*B*B) # Usual Product or multiplication of matrices
      [,1] [,2]
[1,]    2    0
[2,]    6    0

> # Inverse of matrix using solve()
> A <- matrix(c(3,4,1,2),nrow=2,ncol=2)

> print(solve(A))
      [,1] [,2]
[1,]    1 -0.5
[2,]   -2  1.5
> |
```

VIVA QUESTIONS

1. How can you create a 3x4 matrix with random numbers between 1 and 10 in R?

==> Answer: `matrix(runif(12, min = 1, max = 10), nrow = 3, ncol = 4)`

2. Explain the difference between element-wise and matrix multiplication in R.

==> Answer: Element-wise multiplication uses the `*` operator and multiplies corresponding elements between matrices of the same size. Matrix multiplication uses the `%*%` operator and requires compatible dimensions for multiplication (e.g., rows of first matrix equal to columns of second matrix).

3. How can you calculate the transpose of a matrix in R?

==> Answer: Use the `t()` function. For example, `t(myMatrix)` will return the transpose.

4. Explain how to extract a specific row or column from a matrix in R.

==> Answer: Use square brackets with row and column indices. For example, `myMatrix[2,]` selects the second row, and `myMatrix[, 3]` selects the third column.

5. How can you find the inverse of a square matrix in R?

==> Answer: Use the `solve()` function. It calculates the inverse if the matrix is square and non-singular. For example, `solve(myMatrix)`.

Experiment-2

Aim: Write a Program to compute descriptive statistics such as mean, median, mode, standard deviation, and fractiles such as percentiles.

Theory: Descriptive statistics describe the basic and important features of data. Descriptive statistics help simplify and summarize large amounts of data in a sensible manner. For instance, consider the Cumulative Grade Point Index (CGPI), which is used to describe the general performance of a student across a wide range of course experiences.

R Script:

```
# Compute the mean, median, mode, standard deviation and percentile using R
xdata <- c(2,44.4,3,3,2,2.2,2,4)
mu = mean(xdata)
print('The mean is:')
print(mu)
m = median(xdata)
print('The median is:')
print(m)
#Finding a mode is perhaps most easily achieved by using R's table function, which gives you the
frequencies you need.
print(table(xdata))
mode = max(table(xdata))
print('The mode is:')
print(mode)
#Computing above statistics by reading data from .csv file
data <- read.csv(file="/Users/kshitizsharma/Documents/Files/Stats/Stats.csv")
print('The mean is:')
print(mean(data[["Test_marks"]]))
print('The median is:')
print(median(data[["Test_marks"]]))
#In case base R doesn't have a function for weighted median, we need to install a package such as
matrixStats, i.e. library("matrixStats")
print('The weighted mean is:')
print(weighted.mean(data[["Test_marks"]], w=data[["Attendance"]]))
print('The standard deviation is:') # Based on sample standard deviation
print(sd(data[["Test_marks"]]))
```

```
print('The Interquartile range is:')
print(IQR(data[["Test_marks"]]))
# Percentiles are calculated using quantile function in R
print(quantile(data[["Test_marks"]], p=c(.05, .25, .5, .75, .95)))
```

students_data.csv

file contents -

| Name | Attendance | Test_marks |
|---------|------------|------------|
| John | 1 | 20 |
| Michael | 5 | 18 |
| Larry | 10 | 19 |
| Andrew | 8 | 10 |
| Philip | 9 | 12 |
| Mike | 7 | 14 |

Output:

```
Source
Console Terminal Background Jobs
R 4.3.3 ~ /
> source("~/Documents/Files/Stats/Program2.R", echo=TRUE)

> # Compute the mean, median, mode, standard deviation and percentile using R
> xdata <- c(2,44.4,3,3,2,2.2,2,4)

> mu = mean(xdata)

> print('The mean is:')
[1] "The mean is:"

> print(mu)
[1] 7.825

> m = median(xdata)

> print('The median is:')
[1] "The median is:"

> print(m)
[1] 2.6

> #Finding a mode is perhaps most easily achieved by using R's table function, which gives you the frequencies you need.
> print(table(xdata))
xdata
 2  2.2  3   4 44.4
3   1   2   1   1

> mode = max(table(xdata))

> print('The mode is:')
[1] "The mode is:"

> print(mode)
[1] 3

> #Computing above statistics by reading data from .csv file
> data <- read.csv(file="/Users/kshitizsharma/Documents/Files/Stats/Stats.csv")

> print('The mean is:')
[1] "The mean is:"

> print(mean(data[["Test_marks"]]))
[1] 15.5

> print('The median is:')

[1] "The median is:"

> print(median(data[["Test_marks"]]))
[1] 16

> #In case base R doesn't have a function for weighted median, we need to install a package such as matrixStats, i.e. library("matrixStats")
> print( ... [TRUNCATED]
[1] "The weighted mean is:"

> print(weighted.mean(data[["Test_marks"]], w=data[["Attendance"]]))
[1] 14.65

> print('The standard deviation is:') # Based on sample standard deviation
[1] "The standard deviation is:"

> print(sd(data[["Test_marks"]]))
[1] 4.086563

> print('The Interquartile range is:')
[1] "The Interquartile range is:"

> print(IQR(data[["Test_marks"]]))
[1] 6.25

> # Percentiles are calculated using quantile function in R
> print(quantile(data[["Test_marks"]], p=c(.05, .25, .5, .75, .95)))
 5%  25%  50%  75%  95%
10.50 12.50 16.00 18.75 19.75
>
```

VIVA QUESTIONS

1. How can you calculate the mean, median, and mode of a numerical vector in R?

==> Answer:

- Mean: `mean(yourVector)`
- Median: `median(yourVector)`
- Mode: `table(yourVector)[which.max(table(yourVector))]` (finds the most frequent value).

2. What function calculates the standard deviation of a data set in R and explains the difference between population (σ) and sample (s) standard deviation.

==> Answer: `sd(yourVector)`.

- This calculates the sample standard deviation (s), which estimates the population standard deviation (σ) based on the sample data. σ typically requires knowledge of the entire population, whereas s uses sample data.

3. How can you find the quartiles (Q1, Q2, Q3) and interquartile range (IQR) of a data set in R?

==> Answer: Use `quantile(yourVector)`. The quartiles are automatically displayed at specific quantile values (0.25, 0.5, 0.75). IQR is the difference between Q3 and Q1 ($IQR = Q3 - Q1$).

4. Explain how to calculate the percentage of values in a data set that fall below a specific threshold in R.

==> Answer: Combine logical indexing and mean:

Example: Find percentage below 50

```
percentBelow <- mean(yourVector < 50) * 100
```

5. How can you summarize various descriptive statistics for a data frame in R?

==> Answer: Use the `summary()` function on your data frame. This provides a quick overview of measures like mean, median, minimum, maximum, and quartiles for numerical variables.

Experiment-3

Aim: Write a program to calculate Correlation using R.

Theory: In statistics, correlation or dependence is any statistical relationship, whether causal or not, between two random variables or bivariate data. It usually refers to the degree to which a pair of variables are linearly related.

R Script:

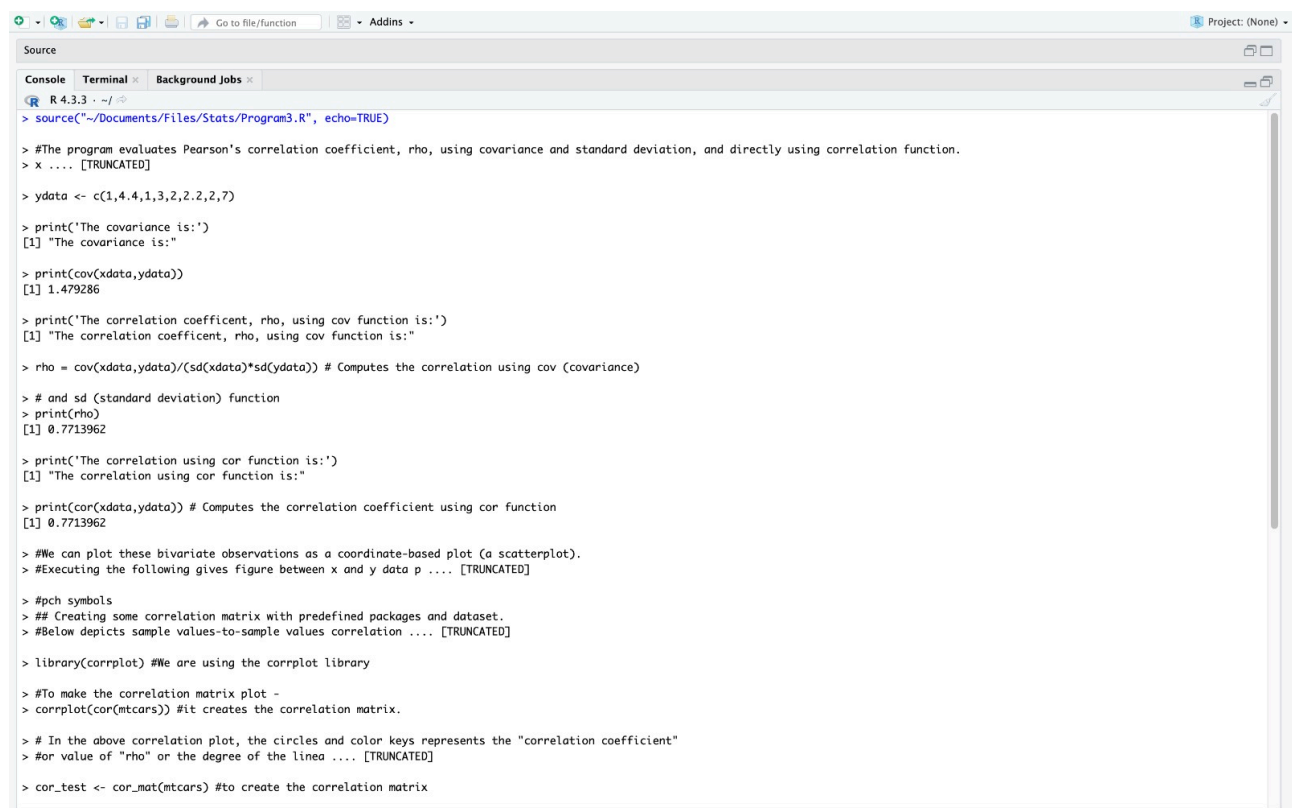
```
#The program evaluates Pearson's correlation coefficient, rho, using covariance and standard
deviation, and directly using correlation function.
xdata <- c(2,4.4,3,3,2,2.2,2,4)
ydata <- c(1,4.4,1,3,2,2.2,2,7)
print('The covariance is:')
print(cov(xdata,ydata))
print('The correlation coefficient, rho, using cov function is:')
rho = cov(xdata,ydata)/(sd(xdata)*sd(ydata)) # Computes the correlation using cov (covariance)
# and sd (standard deviation) function
print(rho)
print('The correlation using cor function is:')
print(cor(xdata,ydata)) # Computes the correlation coefficient using cor function
#We can plot these bivariate observations as a coordinate-based plot (a scatterplot).
#Executing the following gives figure between x and y data points
plot(xdata,ydata,pch=13,cex=1.5) # pch = 13, means a symbol of circle cross; cex is the size of
#pch symbols
## Creating some correlation matrix with predefined packages and dataset.
#Below depicts sample values-to-sample values correlation
data("mtcars") # loading the predefined dataset
library(corrplot) #We are using the corrplot library
#To make the correlation matrix plot -
corrplot(cor(mtcars)) #it creates the correlation matrix.
# In the above correlation plot, the circles and color keys represents the "correlation coefficient"
#or value of "rho" or the degree of the linear relationship.
#The value of rho can be -1 to +1. +1 means positive 100% correlation, i.e., if one variable
#increases, the other will also increase, or if it decreases, the other will also decrease. -1 means
#negative 100%, i.e., the other will decrease if one increases. And 0.0 means no linear
#relationship.
#The size of the circles is relative to the percentage of correlation.
## Generating a correlation matrix ##
library(rstatix)
cor_test <- cor_mat(mtcars) #to create the correlation matrix
```

```

cor_test
## Another informative is "PerformanceAnalytics", which gives the p-value,
## distribution (histograms), and correlation coefficient
library(PerformanceAnalytics)
chart.Correlation(cor(mtcars))
# In the above chart, the red stars define different levels of significance, i.e. * = 0.05, ** = 0.01, ***
= 0.001
## Using package called "Lares", we can rank the correlation and produce a gradually decreasing
## order of columns. This is useful to analyze the top most correlated variables. ##
library(lares)
corr_cross(mtcars, rm.na = T, max_pvalue = 0.05, top = 15, grid = T)

```

Output:



```

R 4.3.3 ~ /
> source("~/Documents/Files/Stats/Program3.R", echo=TRUE)

> #The program evaluates Pearson's correlation coefficient, rho, using covariance and standard deviation, and directly using correlation function.
> x .... [TRUNCATED]

> ydata <- c(1,4,4,1,3,2,2,2,7)

> print('The covariance is:')
[1] "The covariance is:"

> print(cov(xdata,ydata))
[1] 1.479286

> print('The correlation coefficient, rho, using cov function is:')
[1] "The correlation coefficient, rho, using cov function is:"

> rho = cov(xdata,ydata)/(sd(xdata)*sd(ydata)) # Computes the correlation using cov (covariance)

> # and sd (standard deviation) function
> print(rho)
[1] 0.7713962

> print('The correlation using cor function is:')
[1] "The correlation using cor function is:"

> print(cor(xdata,ydata)) # Computes the correlation coefficient using cor function
[1] 0.7713962

> #We can plot these bivariate observations as a coordinate-based plot (a scatterplot).
> #Executing the following gives figure between x and y data p .... [TRUNCATED]

> #pch symbols
> ## Creating some correlation matrix with predefined packages and dataset.
> #Below depicts sample values-to-sample values correlation .... [TRUNCATED]

> library(corrplot) ##we are using the corrplot library

> #To make the correlation matrix plot -
> corrplot(cor(mtcars)) #it creates the correlation matrix.

> # In the above correlation plot, the circles and color keys represents the "correlation coefficient"
> #or value of "rho" or the degree of the linea .... [TRUNCATED]

> cor_test <- cor_mat(mtcars) #to create the correlation matrix

```

```

> cor_test
# A tibble: 11 x 12
  * rowname mpg cyl disp hp drat wt  qsec vs  am gear carb
  * <chr>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 mpg      1 -0.85 -0.85 -0.78 0.68 -0.87 0.42 0.66 0.6 0.48 -0.55
2 cyl     -0.85 1 0.9 0.83 -0.7 0.78 -0.59 -0.81 -0.52 -0.49 0.53
3 disp    -0.85 0.9 1 0.79 -0.71 0.89 -0.43 -0.71 -0.59 -0.56 0.39
4 hp      -0.78 0.83 0.79 1 -0.45 0.66 -0.71 -0.72 -0.24 -0.13 0.75
5 drat     0.68 -0.7 -0.71 -0.45 1 -0.71 0.091 0.44 0.71 0.7 -0.091
6 wt      -0.87 0.78 0.89 0.66 -0.71 1 -0.17 -0.55 -0.69 -0.58 0.43
7 qsec     0.42 -0.59 -0.43 -0.71 0.091 -0.17 1 0.74 -0.23 -0.21 -0.66
8 vs       0.66 -0.81 -0.71 -0.72 0.44 -0.55 0.74 1 0.17 0.21 -0.57
9 am       0.6 -0.52 -0.59 -0.24 0.71 -0.69 -0.23 0.17 1 0.79 0.058
10 gear    0.48 -0.49 -0.56 -0.13 0.7 -0.58 -0.21 0.21 0.79 1 0.27
11 carb    -0.55 0.53 0.39 0.75 -0.091 0.43 -0.66 -0.57 0.058 0.27 1

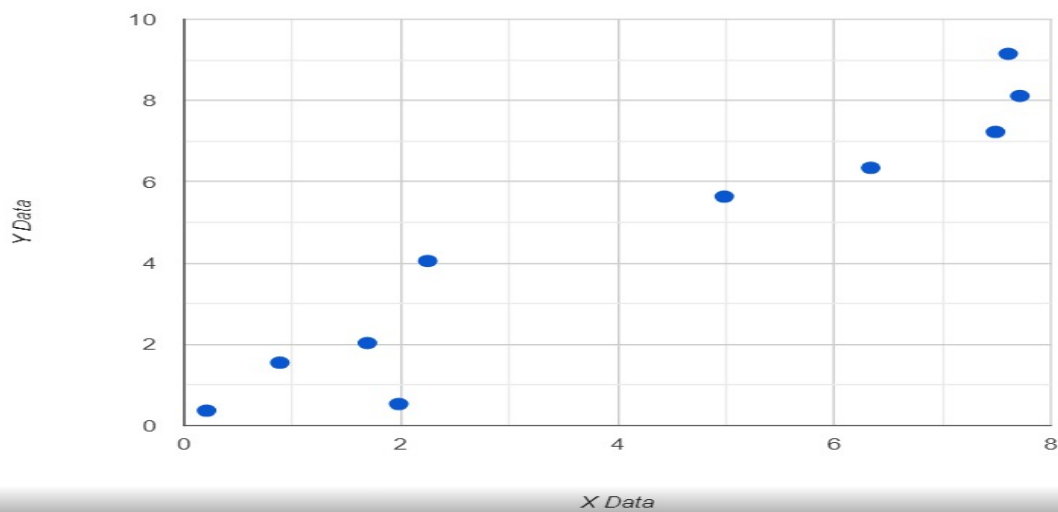
> ## Another informative is "PerformanceAnalytics", which gives the p-value,
> ## distribution (histograms), and correlation coefficient
> library(Per .... [TRUNCATED]

> chart.Correlation(cor(mtcars))

> # In the above chart, the red stars define different levels of significance, i.e. * = 0.05, ** = 0.01, *** = 0.001
> ## Using package called "Lares" .... [TRUNCATED]

> corr_cross(mtcars, rm.na = T, max_pvalue = 0.05, top = 15, grid = T)
Returning only the top 15. You may override with the 'top' argument
There were 50 or more warnings (use warnings() to see the first 50)
>

```



Ranked Cross-Correlations

15 most relevant [NAs removed]



Correlations with p-value < 0.05

VIVA QUESTIONS

1. How can you calculate Pearson's correlation coefficient between two variables in a data frame using R?

==> Answer: Use the `cor()` function. Specify the two variables from the data frame as arguments:
`correlation <- cor(data$variable1, data$variable2)`

2. Explain the difference between Pearson's correlation and Spearman's rank correlation.

==> Answer: Pearson's correlation measures the linear relationship between two continuous variables. Spearman's rank correlation assesses the monotonic relationship (increasing or decreasing trend) between any two variables, including non-parametric data (data not necessarily following a normal distribution).

3. How can you calculate Spearman's rank correlation coefficient in R?

==> Answer: Use the `cor(data$variable1, data$variable2, method = "spearman")` function. This specifies the "spearman" method for rank correlation.

4. Interpret the value of a correlation coefficient. What does a value of 0.8 indicate?

==> Answer: Correlation coefficients range from -1 to 1. A value closer to 1 indicates a strong positive correlation (variables move together), -1 indicates a strong negative correlation (variables move in opposite directions), and 0 suggests no linear relationship. 0.8 signifies a strong positive correlation.

5. How can you check for statistical significance of a correlation coefficient in R?

==> Answer: Use the `cor.test()` function. It provides the correlation coefficient and a p-value. A low p-value (typically < 0.05) suggests the correlation is statistically significant, meaning it's unlikely due to chance.

Experiment-4

Aim: Write a program to present the data as a frequency table. Find the outliers in a dataset.

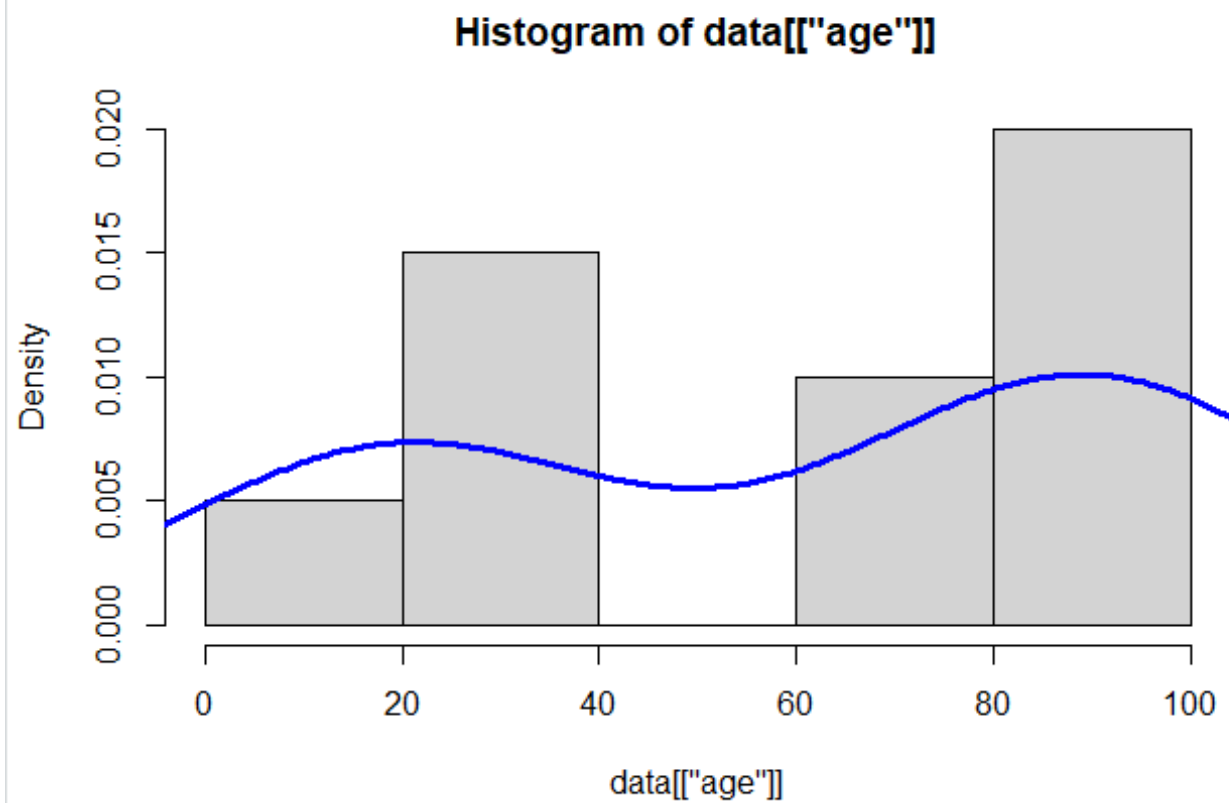
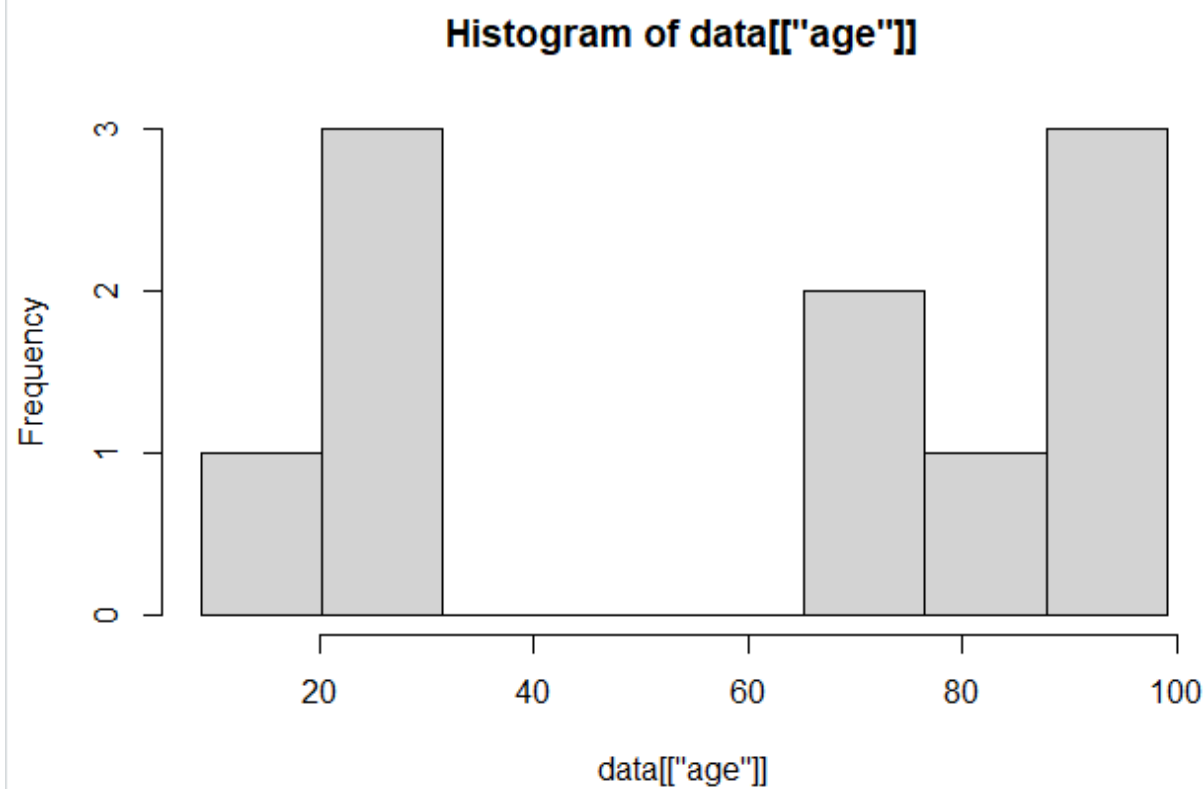
Theory: Data points far from the dataset's other points are considered outliers. This refers to the data values dispersed among other data values and upsetting the dataset's general distribution. Outlier detection is a statistical approach used to find outliers in datasets. Measurement errors, incorrect data entry, or really anomalous data values are just a few of the causes of outliers.

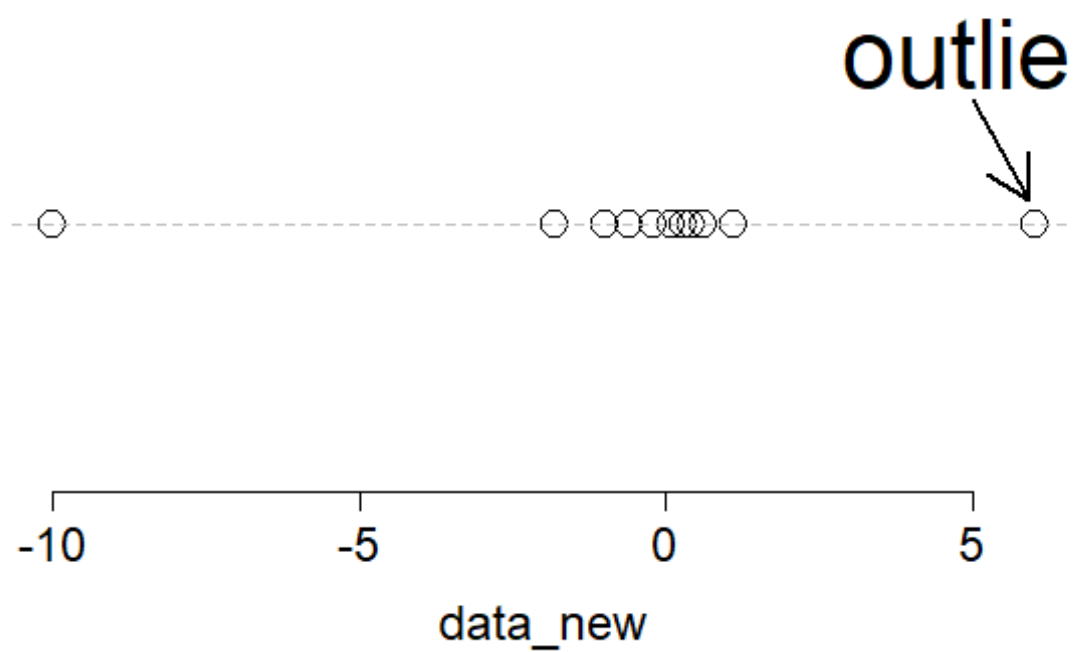
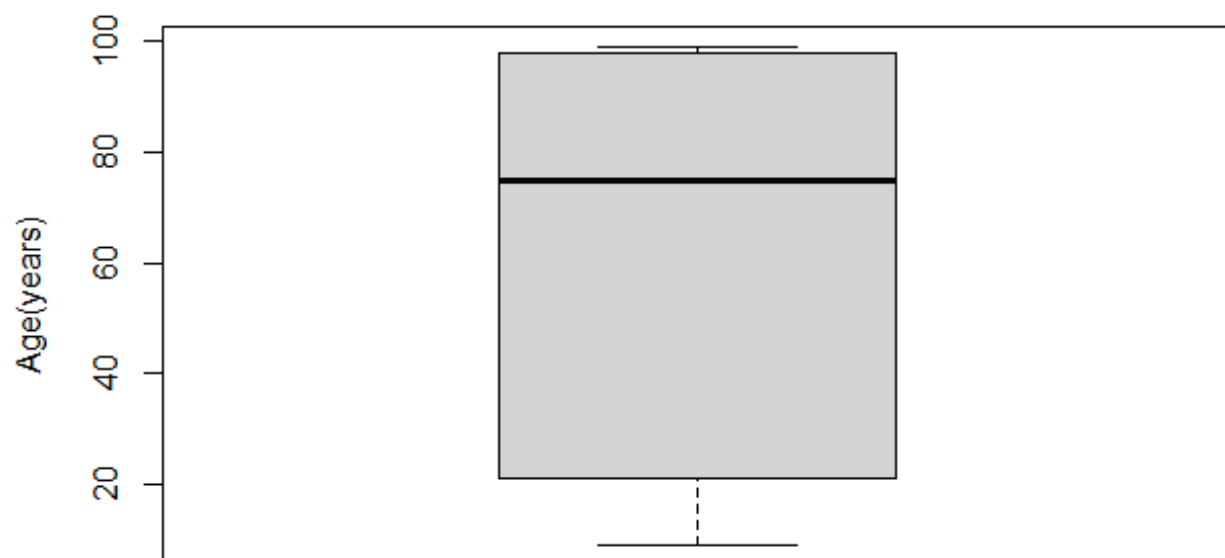
R Script:

```
data = read.csv(file="/Users/kshitizsharma/Documents/Files/Stats/Stats2.csv")
break_interval<-seq(from=min(data[["age"]]),to=max(data[["age"]]),length=9)
age_freq<-cut(data[["age"]],breaks=break_interval,right=TRUE,include.lowest=TRUE)
table(age_freq)
break_interval2 <- seq(from=min(data[["age"]]),to=max(data[["age"]]), 5)
age_freq2 <- cut(data[["age"]], breaks=break_interval2,right=TRUE, include.lowest = TRUE)
table(age_freq2)
hist(data[["age"]], breaks=break_interval)
hist(data[["age"]], freq=FALSE)
lines(density(data[["age"]]), lwd=3, col="blue")
df <- data.frame(team=c('A', 'A', 'A', 'A', 'B', 'B', 'B', 'B'),
                 position=c('G', 'G', 'G', 'F', 'G', 'F', 'F', 'C'))
df
library(dplyr)
df %>% group_by(team, position) %>% summarize(Freq=n())
boxplot(data[["age"]], ylab="Age(years)")
data_new <- c(0.6,-0.6,0.1,-0.2,-1.0,0.4,0.3,-1.8,1.1,6.0,-10)
plot(data_new,rep(0,11), yaxt="n",ylab="",bty="n",cex=2,cex.axis=1.5,cex.lab=1.5)
abline(h=0,col="gray",lty=2)
arrows(5,0.5,5.9,0.1,lwd=2)
text(5,0.7,labels="outlier",cex=3)
```

Output:

```
> data = read.csv(file="C:/Users/PC/Desktop/New folder (2)/age_data.csv")
> break_interval<-seq(from=min(data[["age"]]),to=max(data[["age"]]),length=9)
> age_freq<-cut(data[["age"]],breaks=break_interval,right=TRUE,include.lowest=TRUE)
> table(age_freq)
age_freq
 [9,20.2] (20.2,31.5] (31.5,42.8] (42.8,54] (54,65.2] (65.2,76.5] (76.5,87.8]
      1          3          0          0          0          2          1
 (87.8,99]
      3
> break_interval2 <- seq(from=min(data[["age"]]),to=max(data[["age"]]), 5)
> age_freq2 <- cut(data[["age"]], breaks=break_interval2,right=TRUE, include.lowest = TRUE)
> table(age_freq2)
age_freq2
 [9,14] (14,19] (19,24] (24,29] (29,34] (34,39] (39,44] (44,49] (49,54] (54,59] (59,64]
      1          0          2          1          0          0          0          0          0          0          0
 (64,69] (69,74] (74,79] (79,84] (84,89] (89,94] (94,99]
      0          0          2          0          1          0          3
> hist(data[["age"]], breaks=break_interval)
> hist(data[["age"]], freq=FALSE)
> lines(density(data[["age"]]), lwd=3, col="blue")
> df <- data.frame(team=c('A', 'A', 'A', 'A', 'B', 'B', 'B', 'B'),
+                   position=c('G', 'G', 'G', 'F', 'G', 'F', 'F', 'C'))
> df
  team position
1   A         G
2   A         G
3   A         G
4   A         F
5   B         G
6   B         F
7   B         F
8   B         C
> library(dplyr)
```





VIVA QUESTIONS

1. How can you create a frequency table for a categorical variable in R?

==> Answer: There are two main ways:

- `table(yourData$categoricalVariable)`: This creates a simple frequency table showing the counts of each category.
- `dplyr::count(yourData, categoricalVariable)` (using the `dplyr` package): This offers more flexibility, allowing you to group by additional variables or calculate summary statistics along with frequencies.

2. How can you identify outliers in a numerical variable using a boxplot in R?

==> Answer: Use `boxplot(yourData$numericalVariable)`. Points falling outside the whiskers (beyond 1.5 times the interquartile range from the quartiles) are potential outliers.

3. How can you calculate Interquartile Range (IQR) and use it to identify outliers in R?

==> Answer: Use `IQR(yourData$numericalVariable)`. Values less than $Q1 - 1.5 * IQR$ or greater than $Q3 + 1.5 * IQR$ are potential outliers ($Q1$ and $Q3$ are the first and third quartiles, respectively).

4. What function can be used to identify potential outliers based on z-scores in R?

==> Answer: Use `abs(zscore(yourData$numericalVariable)) > 3`. This calculates the absolute value of the z-scores and identifies observations with a z-score greater than 3 (or less than -3) as potential outliers.

5. Explain the difference between identifying and treating outliers in a data set.

==> Answer: Identifying outliers involves techniques like boxplots, IQR, or z-scores to flag potential data points that deviate significantly from the majority. Treating outliers requires further investigation. You might decide to remove them if they are errors, Winsorize them (cap their values to a certain threshold), or keep them if they represent valid but extreme values.

Experiment-5

Aim: Write a program to implement concepts of probability and distributions in R.

Theory: Classical probability, often referred to as “a priori” probability, is a branch of probability theory that deals with situations where all possible outcomes are equally likely. It provides a foundational understanding of how probability works and forms the basis for more advanced probability concepts.

R Script:

```
##### Binomial Distribution #####
dbinom(x=5,size=8,prob=1/6) # Binomial distribution for Pr(X=5), no. of trials, n = 8,
# probability of success = 1/6
x_prob_binomial = dbinom(x=0:8,size=8,prob=1/6)
print(x_prob_binomial)
print(sum(x_prob_binomial))
print(round(x_prob_binomial, 3))
barplot(x_prob_binomial,names.arg=0:8,space=0,xlab="x",ylab="Pr(X = x)", col = "green")
pbinom(q=3,size=8,prob=1/6) #Evaluating cumulative distribution function for Binomial
# distribution, i.e. Pr(X <= 3)

##### Poisson Distribution #####
dpois(x=3,lambda=3.22) # Poisson distribution. The dpois function provides the individual
# Poisson mass function probabilities
# Pr(X = x) for the Poisson distribution. The ppois function provides the left cumulative
# probabilities, i.e. Pr(X <= x)
dpois(x=0,lambda=3.22)
round(dpois(0:10,3.22),3)
barplot(dpois(x=0:10,lambda=3.22),ylim=c(0,0.25),space =
0,names.arg=0:10,ylab="Pr(X=x)",xlab="x", col = "blue")
ppois(q=2,lambda=3.22)

##### Uniform Distribution #####
min <- 0
max <- 100
# Specify x-values for the function
xpos <- seq(min, max , by = 0.5)
# supplying corresponding y coordinations
ypos <- dunif(xpos, min = 10, max = 80)
# plotting the graph
plot(ypos , type="o")
## CDF for Uniform distribution function
```

```

min <- 0
max <- 60
# calculating punif value
punif(15 , min =min , max = max)

##### Normal Distribution #####
# Generating sequence of
# numbers from -15 to +10
x<-seq(-15,10)
# calculate the Normal distribution function
# based on the mean and sd of data
pdf<- dnorm(x,mean(x),sd(x))
# Plotting the PDF(Normal Distribution Function
plot(x,pdf)
# CDF of Normal Distribution
# Generating sequence of
# numbers from -15 to +10
x<-seq(-15,10)
# calculate the Cumulative distribution
# function based on the mean and sd of data
cdf<-ecdf(x)
# Plotting the CDF
plot(cdf,xlabel="x",ylabel="y",main="CDF Graph")
# Plotting CDF using gbutils package
# install and load the gbutils package
library(gbutils)
# Calculating the CDF Normal Distribution Function
cdf1 <- pnorm(x, mean = -2.5, sd = 7.64)
# Plotting the CDF Normal Distribution Function
plotpdf(cdf1, cdf = cdf1,main='CDF Plot')
#### QQ-Plot for a sample of 100
# values randomly generated from a normal distribution; as expected, the points
# closely follow the line. If the points roughly fall on the diagonal line, then the sample
# distribution can be
# considered close to normal. #####
norm_samp <- rnorm(100)
qqnorm(norm_samp)
abline(a=0, b=1, col='grey')

##### Student's t-distribution #####
x <- seq(-6, 6, length = 100) # seq function can be used to generate 100 points between say -6 to 6
df = c(1,4,10,30)
colour = c("red", "orange", "green", "yellow", "black")
# Plot a normal distribution

```

```

plot(x,dnorm(x), type = "l" , lty = 2, xlab= "t-value", ylab="Density", main = "Comparison of
tdistributions", col = "black") # type = "l" is for line,
# lty = 2 or lty = "dashed" is for dashed line
#Add the t-distributions to the plot
for (i in 1:4){
  lines(x, dt(x, df[i]), col = colour[i])
}
#Add a legend
legend("topright", c("df = 1", "df = 4", "df = 10", "df = 30", "normal"), col = colour , title =
  "t=distributions", lty = c(1, 1, 1, 1, 2))

```

```

##### Chi-square distribution #####
df = 5 #Defining degrees of freedom
vec <- 0:4
#Calculating for the Density function values (pdf) in the interval [0, 4]
print ("Calculating for the values [0, 4]")
dchisq(vec, df = df) # pdf of distribution
pchisq(4, df = df, lower.tail = TRUE) # CDF of distribution, lower.tail = TRUE means  $P(X \leq x)$ 
# x) and if lower.tail = FALSE means  $P(X > x)$ .
# computing probability values of 50,000 random values with 4 degrees of freedom
x <- rchisq(50000, df = 4)
hist(x, freq = FALSE, xlim = c(0, 16), ylim = c(0, 0.2), col = 'gray')
curve(dchisq(x, df = 4), from = 0, to = 15, n = 5000, col = 'red', lwd = 2, add = T)

```

Output:

```
R 4.3.3 ~ /
> source("~/Documents/Files/Stats/Program5.R", echo=TRUE)

> ##### Binomial Distribution #####
> dbinom(x=5,size=8,prob=1/6) # Binomial distribution for Pr(X=5), no. of trials, n = 8,
[1] 0.004167619

> # probability of success = 1/6
> x_prob_binomial = dbinom(x=0:8,size=8,prob=1/6)

> print(x_prob_binomial)
[1] 2.325680e-01 3.721089e-01 2.604762e-01 1.041905e-01 2.604762e-02 4.167619e-03 4.167619e-04 2.381497e-05
[9] 5.953742e-07

> print(sum(x_prob_binomial))
[1] 1

> print(round(x_prob_binomial, 3))
[1] 0.233 0.372 0.260 0.104 0.026 0.004 0.000 0.000 0.000

> barplot(x_prob_binomial,names.arg=0:8,space=0,xlab="x",ylab="Pr(X = x)", col = "green")

> pbinom(q=3,size=8,prob=1/6) #Evaluating cumulative distribution function for Binomial
[1] 0.9693436

> # distribution, i.e. Pr(X <= 3)
>
> ##### Poisson Distribution #####
> dpois(x=3,lambda=3.22) # Poisson distribution. The dpois function prov .... [TRUNCATED]
[1] 0.2223249

> # Poisson mass function probabilities
> # Pr(X = x) for the Poisson distribution. The ppois function provides the left cumulative
> # probabilities, .... [TRUNCATED]
[1] 0.03995506

> round(dpois(0:10,3.22),3)
[1] 0.040 0.129 0.207 0.222 0.179 0.115 0.062 0.028 0.011 0.004 0.001

> barplot(dpois(x=0:10,lambda=3.22),ylim=c(0,0.25),space = 0,names.arg=0:10,ylab="Pr(X=x)",xlab="x", col = "blue")

> ppois(q=2,lambda=3.22)
[1] 0.3757454

> ##### Uniform Distribution #####
> min <- 0
```

```
R 4.3.3 ~ /
> max <- 100

> # Specify x-values for the function
> xpos <- seq(min, max , by = 0.5)

> # supplying corresponding y coordinations
> ypos <- dunif(xpos, min = 10, max = 80)

> # plotting the graph
> plot(ypos , type="o")

> ## CDF for Uniform distribution function
> min <- 0

> max <- 60

> # calculating punif value
> punif (15 , min =min , max = max)
[1] 0.25

> ##### Normal Distribution #####
> # Generating sequence of
> # numbers from -15 to +10
> x<-seq(-15,10)

> # calculate the Normal distribution function
> # based on the mean and sd of data
> pdf<- dnorm(x,mean(x),sd(x))

> # Plotting the PDF(Normal Distribution Function)
> plot(x,pdf)

> # CDF of Normal Distribution
> # Generating sequence of
> # numbers from -15 to +10
> x<-seq(-15,10)

> # calculate the Cumulative distribution
> # function based on the mean and sd of data
> cdf<-ecdf(x)

> # Plotting the CDF
> plot(cdf,xlabel="x",ylabel="y",main="CDF Graph")
```

```
Source
Console Terminal Background Jobs
R 4.3.3 ~ /

> # Plotting CDF using gbutils package
> # install and load the gbutils package
> library(gbutils)

> # Calculating the CDF Normal Distribution Function
> cdf1 <- pnorm(x, mean = -2.5, sd = 7.64)

> # Plotting the CDF Normal Distribution Function
> plotpdf(cdf1, cdf = cdf1, main='CDF Plot')

> ##### QQ-Plot for a sample of 100
> # values randomly generated from a normal distribution; as expected, the points
> # closely follow the line. If t .... [TRUNCATED]

> qqnorm(norm_samp)

> abline(a=0, b=1, col='grey')

> ##### Student's t-distribution #####
> x <- seq(-6, 6, length = 100) # seq function can be used to generate 100 points between say -6 to 6

> df = c(1,4,10,30)

> colour = c("red", "orange", "green", "yellow", "black")

> # Plot a normal distribution
> plot(x, dnorm(x), type = "l", lty = 2, xlab= "t-value", ylab="Density", main = "Comparison of tdistributions", col = .... [TRUNCATED]

> # lty = 2 or lty = "dashed" is for dashed line
> #Add the t-distributions to the plot
> for (i in 1:4){
+   lines(x, dt(x, df[i]), col = colour[i])
+   .... [TRUNCATED]

> #Add a legend
> legend("topright", c("df = 1", "df = 4", "df = 10", "df = 30", "normal"), col = colour , title =
+   "t=distributions", lty = .... [TRUNCATED]

> ##### Chi-square distribution #####
> df = 5 #Defining degrees of freedom

> vec <- 0:4

> #Calculating for the Density function values (pdf) in the interval [0, 4]

> print ("Calculating for the values [0, 4]")
[1] "Calculating for the values [0, 4]"

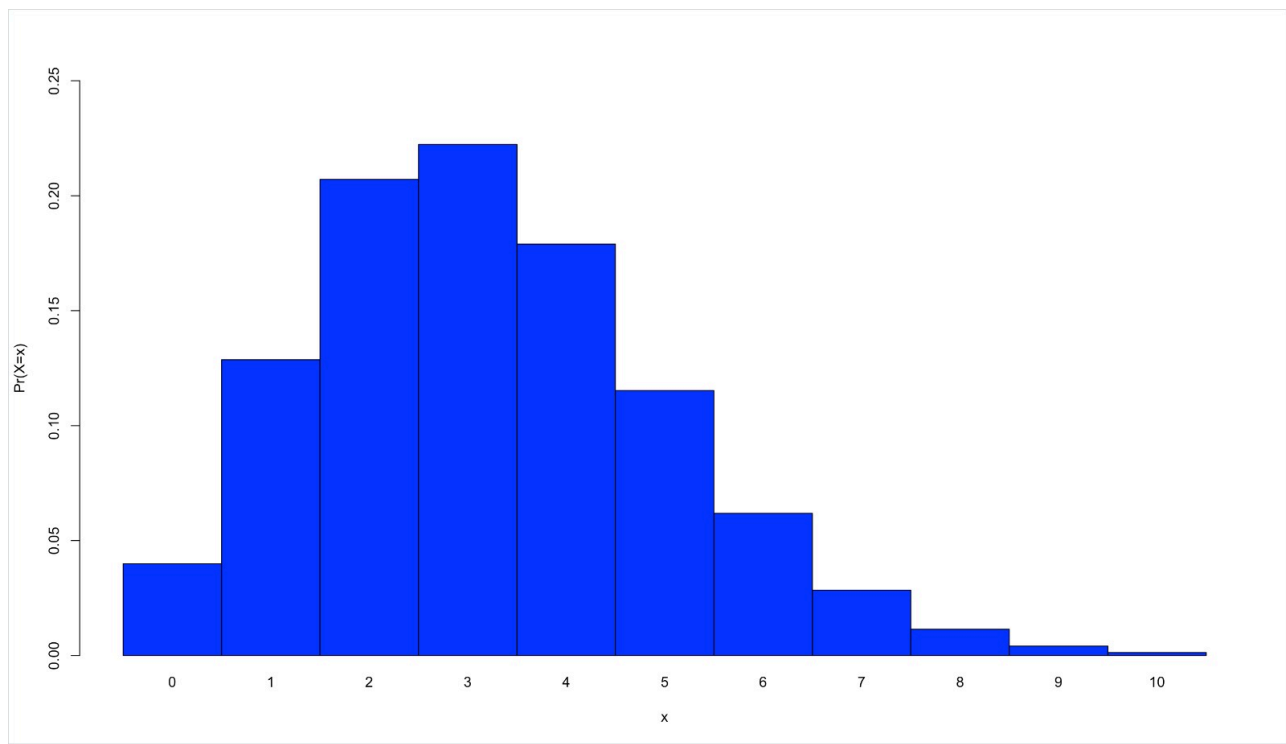
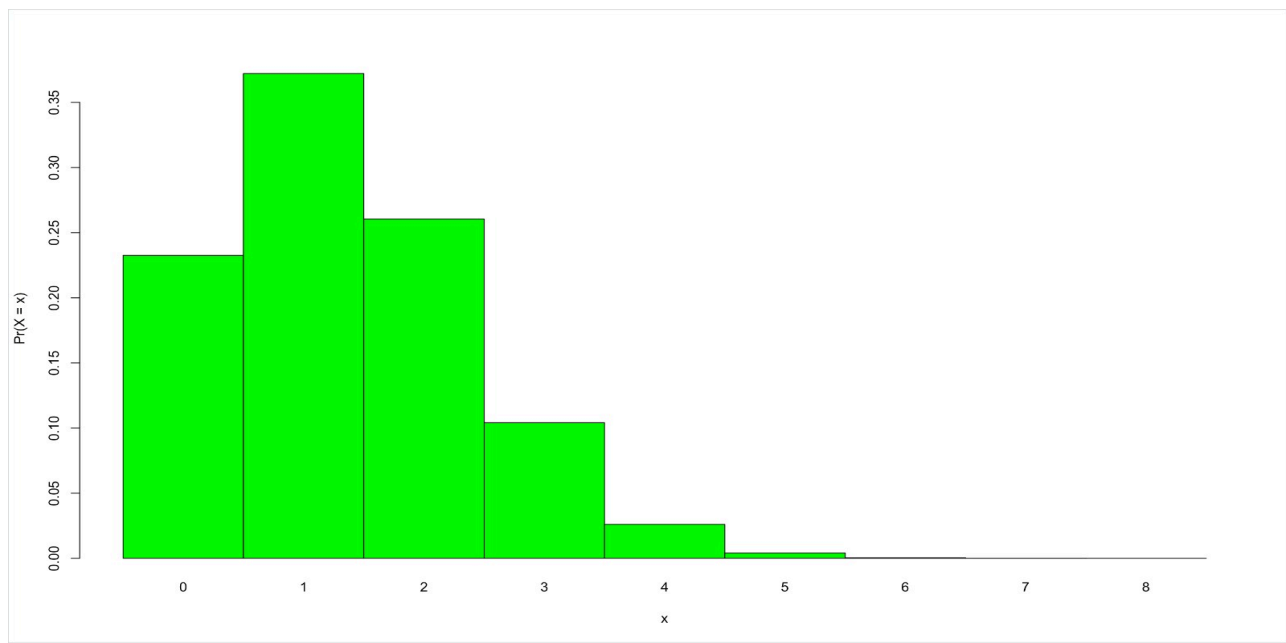
> dchisq(vec, df = df) # pdf of distribution
[1] 0.00000000 0.08065691 0.13836917 0.15418033 0.14397591

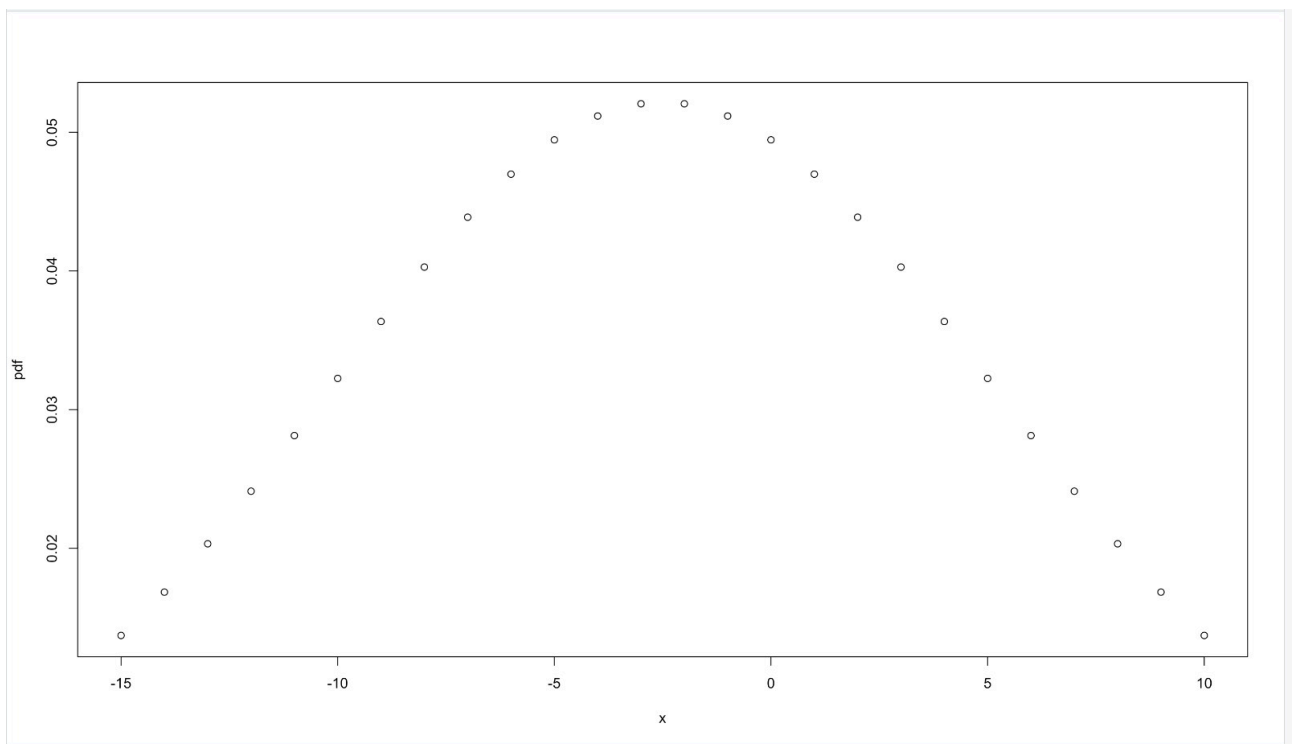
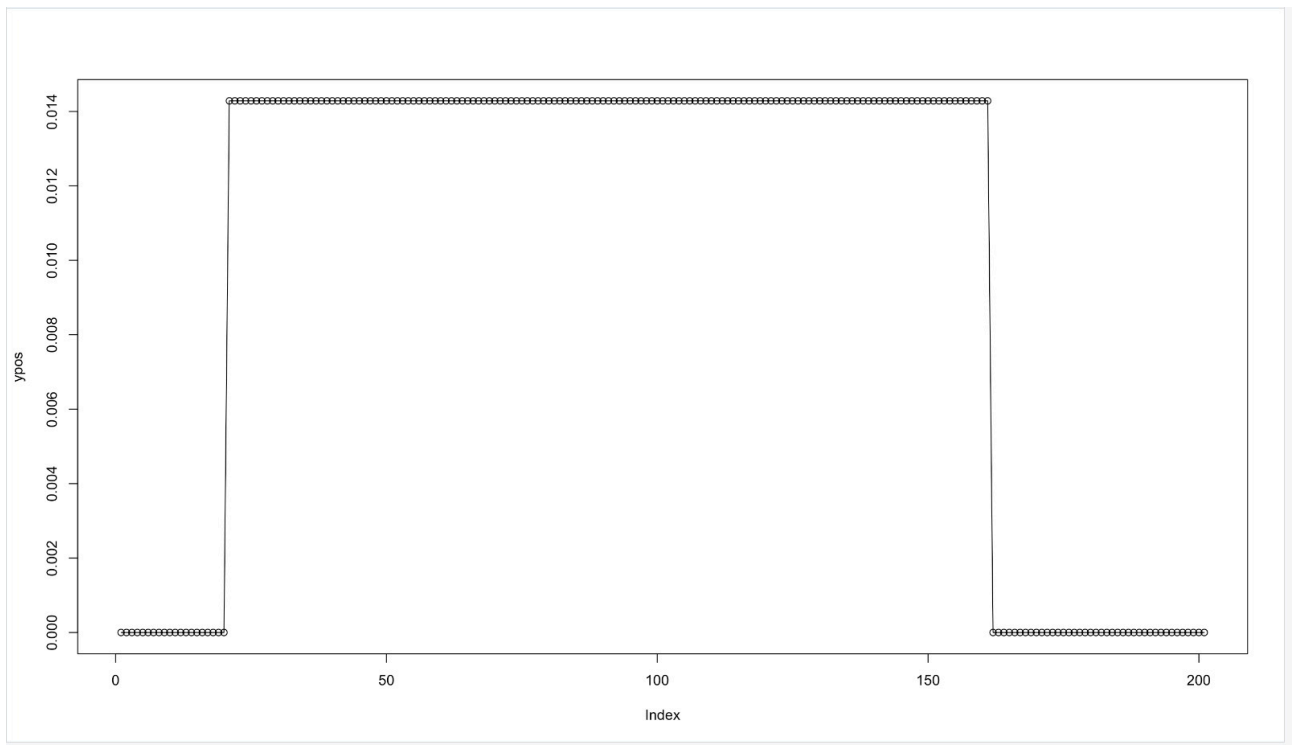
> pchisq(4, df = df, lower.tail = TRUE) # CDF of distribution, lower.tail = TRUE means P(X <=
[1] 0.450584

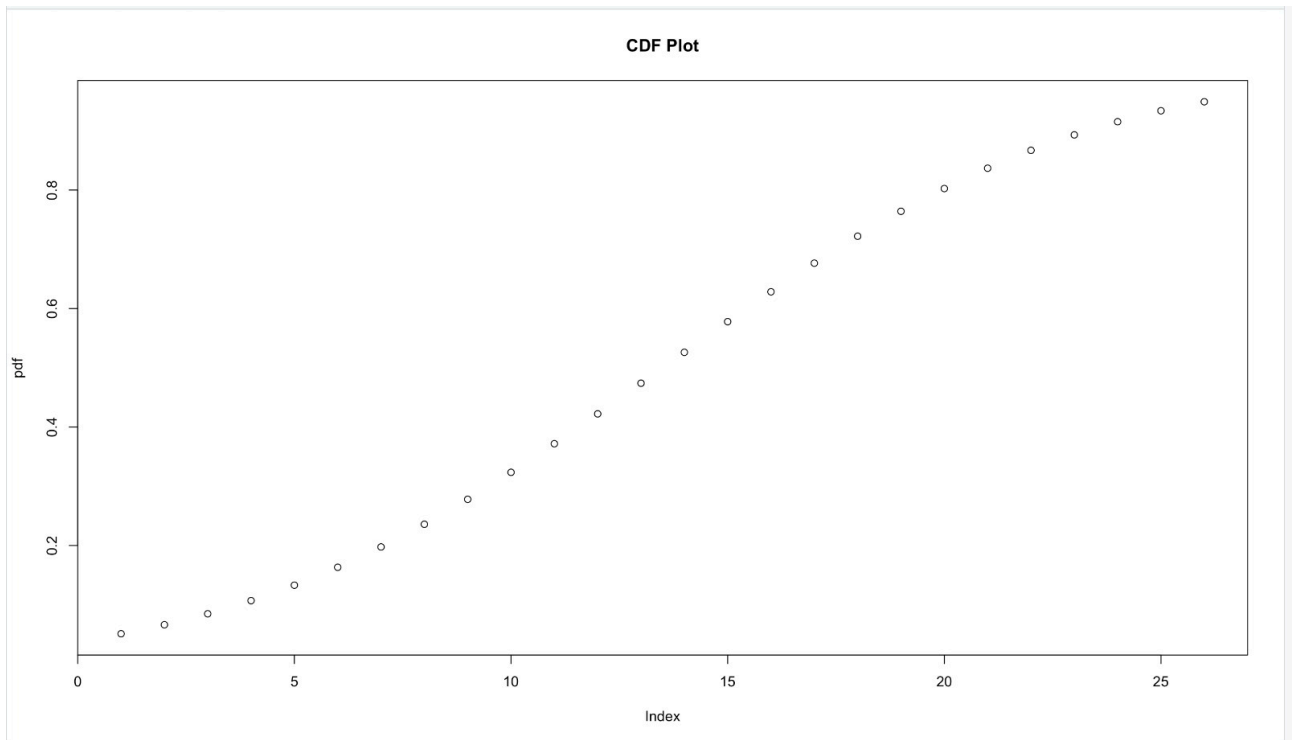
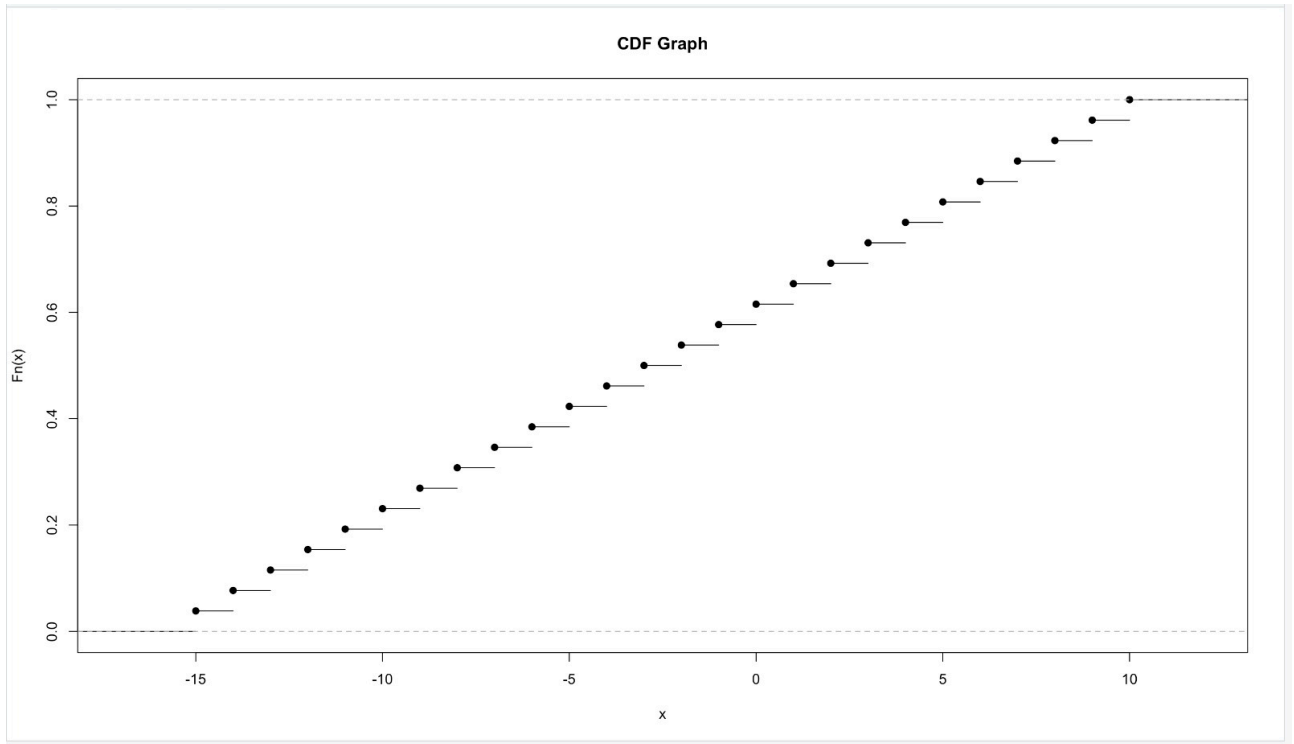
> # x) and if lower.tail = FALSE means P(X > x).
> # computing probability values of 50,000 random values with 4 degrees of freedom
> x <- rchisq(500 .... [TRUNCATED]

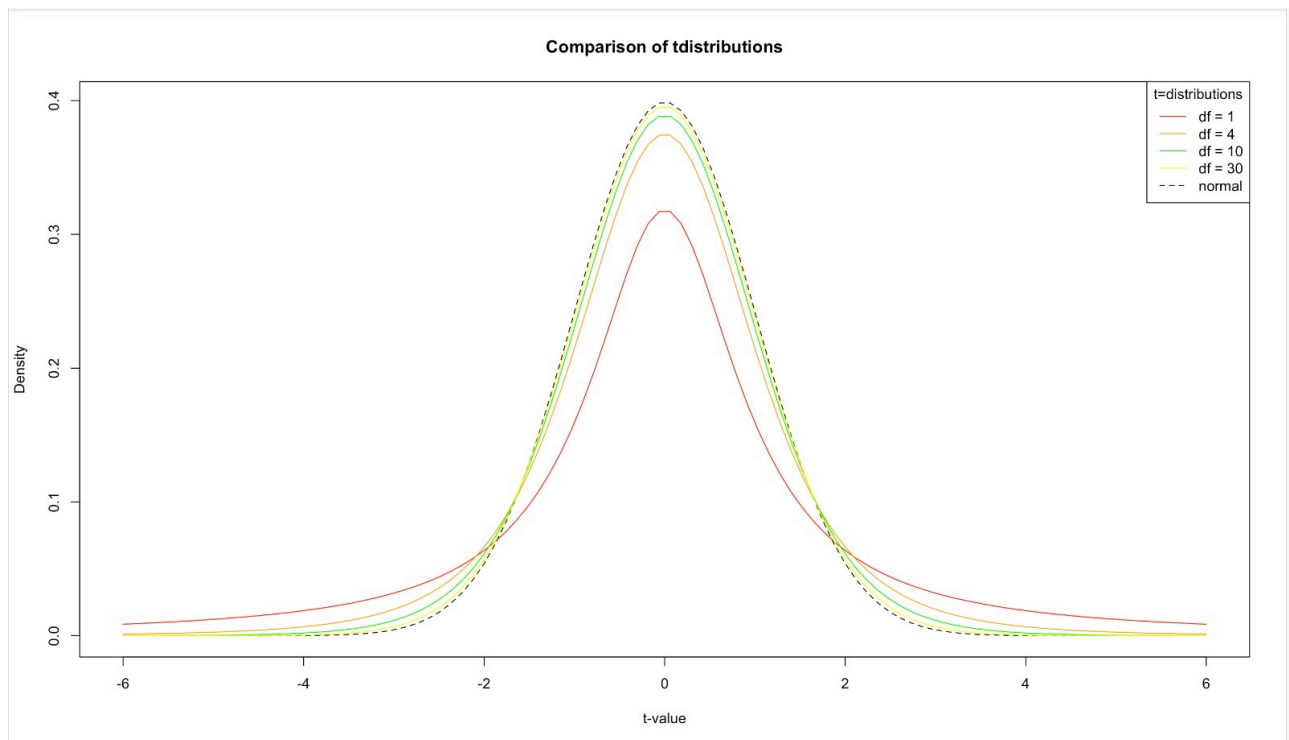
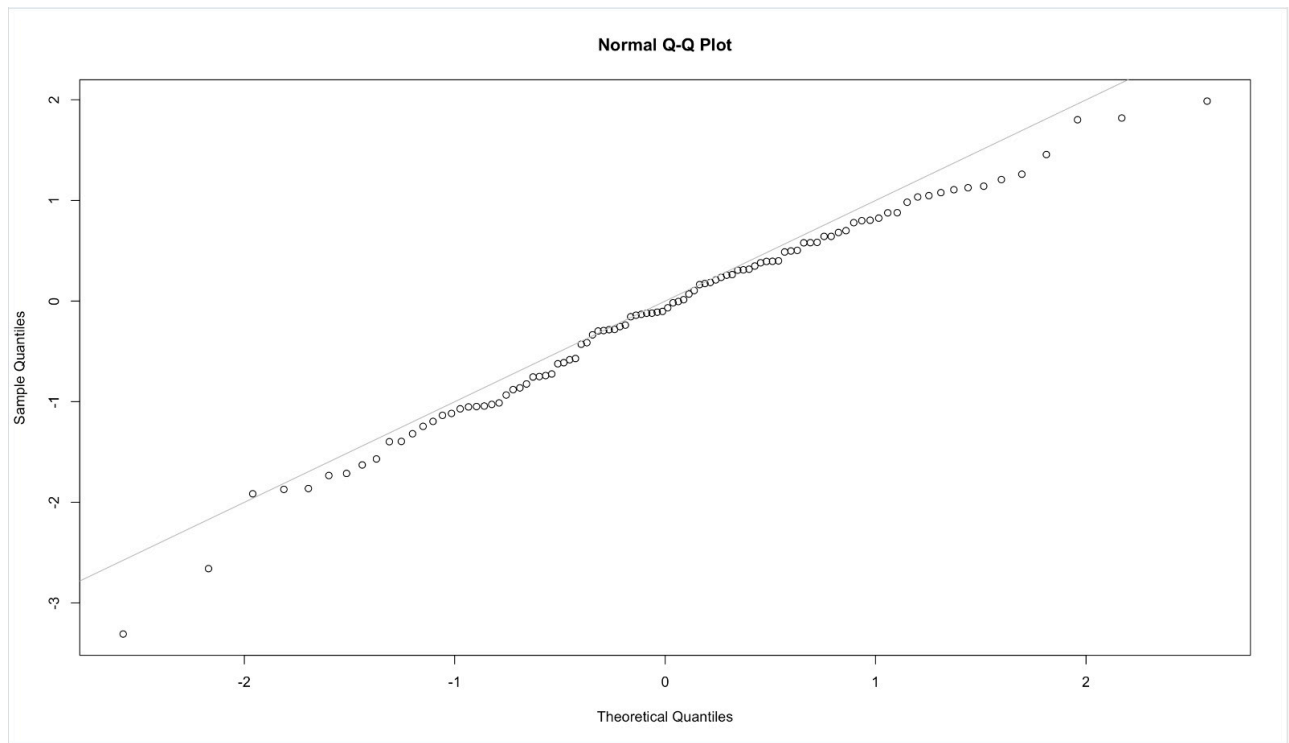
> hist(x, freq = FALSE, xlim = c(0, 16), ylim = c(0, 0.2), col = 'gray')

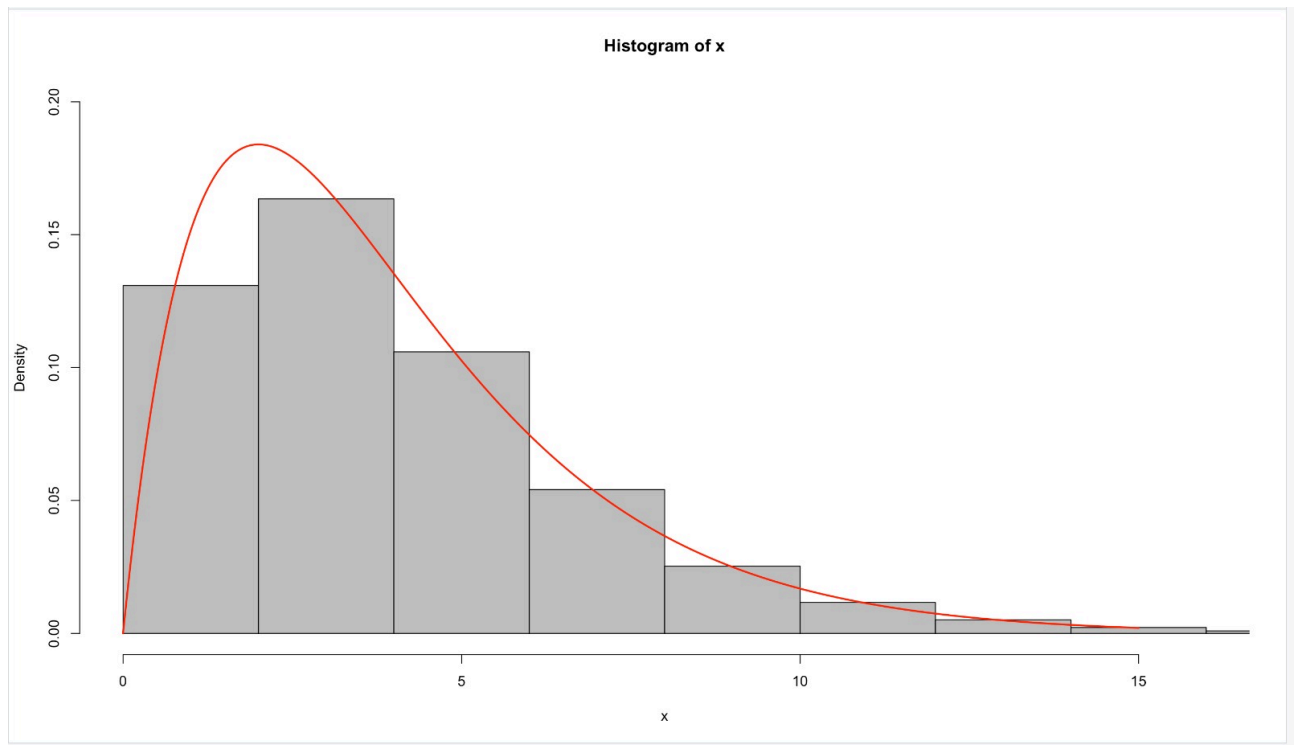
> curve(dchisq(x, df = 4), from = 0, to = 15, n = 5000, col = 'red', lwd = 2, add = T)
There were 24 warnings (use warnings() to see them)
> |
```











VIVA QUESTIONS

1. How can you simulate 1000 coin flips and calculate the probability of getting heads in R?

==> Answer:

- Generate random data: `flips <- rbinom(1000, 1, 0.5)` (1000 trials, success probability 0.5 for heads).
- Calculate probability: `mean(flips == 1)` (checks for heads and takes the mean, effectively calculating the proportion of heads).

2. Explain the Central Limit Theorem and how it relates to sampling distributions in R.

==> Answer: The Central Limit Theorem states that as the sample size increases, the sampling distribution of the mean (average) approaches a normal distribution regardless of the original population's distribution (under certain conditions). In R, you can simulate this by drawing samples from different distributions (e.g., uniform, binomial) with increasing sample sizes and observing the resulting distribution of the means.

3. How can you calculate the probability of drawing a specific value (e.g., number 5) from a normal distribution with a mean of 10 and standard deviation of 2 in R?

==> Answer: Use the `pnorm()` function: `pnorm(5, mean = 10, sd = 2)`. This calculates the cumulative probability up to 5, allowing you to interpret the probability of getting exactly 5 or a value less than 5.

4. Explain the difference between the probability density function (PDF) and cumulative distribution function (CDF) of a distribution and how to visualize them in R.

==> Answer: PDF shows the probability of a specific value occurring. CDF represents the probability of a value being less than or equal to a specific value. Visualize PDF with `dnorm()` (e.g., `dnorm(x, mean = 10, sd = 2)`) and plot the function. Visualize CDF with `pnorm()` (as used in question 3) and plot the function.

5. How can you perform a chi-square test for independence between two categorical variables in R?

==> Answer: Use the `chisq.test(table(yourData$variable1, yourData$variable2))` function. This tests if the observed distribution of categories is independent of each other.