EXPERIMENT NO 1

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

Theory:

Matrix operations involve mathematical operations performed on matrices, rectangular arrays of numbers arranged in rows and columns. Common operations include addition, subtraction, multiplication, and transposition, used in various fields such as linear algebra, statistics, computer graphics, and optimization, to solve systems of equations and analyze data. Multiplication, defined only when certain dimensions align, involves calculating the dot product between row vectors of the first matrix and column vectors of the second.
Transposition simply flips rows and columns.

Source Code:

R-script

```
# Define two matrices matrix1 <- matrix(c(1, 2, 3, 4, 5, 6), nrow =
2, byrow = TRUE) matrix2 <- matrix(c(7, 8, 9, 10, 11, 12), nrow
= 2, byrow = TRUE)

# Display the matrices
print("Matrix 1:")
print(matrix1)
print("Matrix 2:")
print(matrix2)

# Addition of matrices
addition_result <- matrix1 + matrix2
print("Addition of matrices:")
print(addition_result)

# Subtraction of matrices
subtraction_result <- matrix1 - matrix2
print("Subtraction of matrices:")
print(subtraction_result)

# Multiplication of matrices
```

```
multiplication_result <- matrix1 %*% t(matrix2) # Matrix multiplication requires transpose of the second matrix
print("Multiplication of matrices:")

print(multiplication_result)


# Transpose of matrices

transpose_matrix1 <- t(matrix1)

transpose_matrix2 <- t(matrix2)

print("Transpose of Matrix 1:")

print(transpose_matrix1)

print("Transpose of Matrix 2:")

print(transpose_matrix2)
```

Output:

```
> source("c:\\Users\\ARYAN\\PycharmProjects\\gyrados\\new.r", encoding = "UTF-$
[1] "Matrix 1:"
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[1] "Matrix 2:"
     [,1] [,2] [,3]
[1,]    7    8    9
[2,]   10   11   12
[1] "Addition of matrices:"
     [,1] [,2] [,3]
[1,]    8   10   12
[2,]   14   16   18
[1] "Subtraction of matrices:"
     [,1] [,2] [,3]
[1,]   -6   -6   -6
[2,]   -6   -6   -6
[1] "Multiplication of matrices:"
     [,1] [,2]
[1,]   50   68
[2,]  122  167
[1] "Transpose of Matrix 1:"
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
[1] "Transpose of Matrix 2:"
     [,1] [,2]
[1,]    7   10
[2,]    8   11
[3,]    9   12
>
```

EXPERIMENT NO 2

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

Theory:

Mean: Average value of dataset, sensitive to outliers.

Median: Middle value when data is sorted, less affected by outliers.

Mode: Most frequent value in dataset.

Standard Deviation: Measure of data dispersion around mean.

Percentiles: Values dividing dataset into 100 equal parts, useful for analyzing data distribution.

Source Code:

R-script

```
# Define the dataset data <- c(23, 45, 67, 89, 12,
34, 56, 78, 90, 10, 22)
# Compute mean mean_value
<- mean(data) # Compute
median median_value <-
median(data)

# Compute mode
mode_value <- names(sort(table(data), decreasing = TRUE))[1]

# Compute standard deviation
sd_value <- sd(data)

# Compute percentiles (25th, 50th, and 75th
percentiles) percentile_25 <- quantile(data, 0.25)
percentile_50 <- quantile(data, 0.50) percentile_75 <-
quantile(data, 0.75) # Print the descriptive statistics
print("Descriptive Statistics:") print(paste("Mean:",
mean_value)) print(paste("Median:", median_value))
```

```
print(paste("Mode:", mode_value))

print(paste("Standard Deviation:", sd_value))

print("Percentiles:") print(paste("25th Percentile:",

percentile_25)) print(paste("50th Percentile (Median):",

percentile_50)) print(paste("75th Percentile:",

percentile_75))
```

Output:

```
> source("c:\\Users\\ARYAN\\PycharmProjects\\gyrados\\new.r", encoding = "UTF-$
[1] "Descriptive Statistics:"
[1] "Mean: 47.8181818181818"
[1] "Median: 45"
[1] "Mode: 10"
[1] "Standard Deviation: 30.0260492966297"
[1] "Percentiles:"
[1] "25th Percentile: 22.5"
[1] "50th Percentile (Median): 45"
[1] "75th Percentile: 72.5"
```

EXPERIMENT NO 3

Aim: Write a program to calculate Correlation Coefficient using R

Theory:

The correlation coefficient measures the strength and direction of the linear relationship between two variables. It ranges from -1 to 1, where 1 indicates a perfect positive correlation, -1 indicates a perfect negative correlation, and 0 indicates no correlation. Positive values signify direct relationships, while negative values signify inverse relationships

Source Code:

R-script

```
# Set working directory (optional, but recommended for organized projects)

# setwd("/path/to/your/project/directory") # Replace with your actual directory path

# The program evaluates Pearson's correlation coefficient


xdata <- c(2,4.4,3,3,2,2.2,2,4)

ydata <- c(1,4.4,1,3,2,2.2,2,7)


# Covariance and correlation using base R

functions 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))

print(rho)


# Correlation using cor function print('The

correlation using cor function is:')

print(cor(xdata, ydata))


# Scatter plot plot(xdata, ydata, pch=13, cex=1.5, main = "Scatter Plot of xdata vs ydata") #

Add main title


# Correlation matrix with corrplot package
```

library(corrplot) corr_matrix <- cor(mtcars) # Store correlation matrix for later use corrplot(corr_matrix, main = "Correlation Matrix of mtcars") # Add main title

Correlation heatmap with PerformanceAnalytics package

library(PerformanceAnalytics) chart.Correlation(corr_matrix, main = "Correlation Heatmap of mtcars") # Add main title

dev.new() # Clear any existing plots

plot(xdata, ydata, pch=13, cex=1.5)

dev.hold() # Overlap plots on the same device

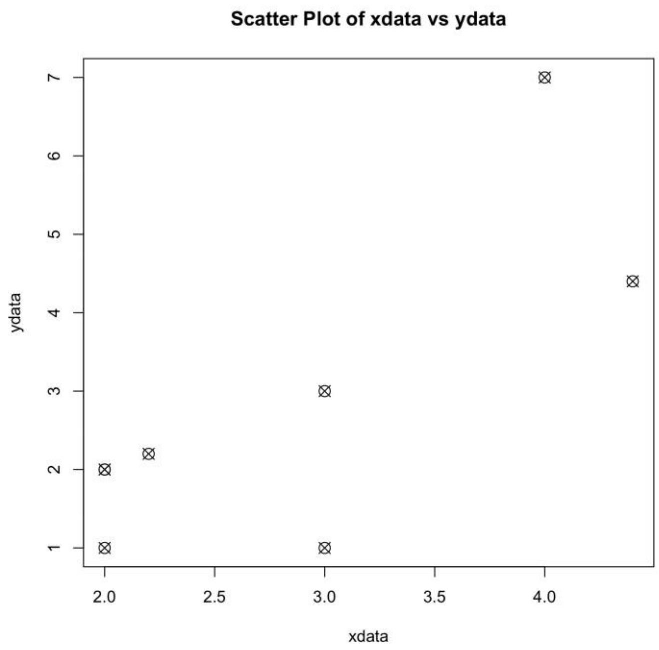
Output:

```
> 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:')
[1] "The covariance is:"
> print(cov(xdata,ydata))
[1] 1.479286
> print('The correlation coefficent, rho, using cov function is:')
[1] "The correlation coefficent, 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 #We can plot these bivariate observations as a coordinate-based plot
(a scatterplot). #Executing the following gives figure between x and y data
points
[1] 0.7713962
>  32
[1] 32
```

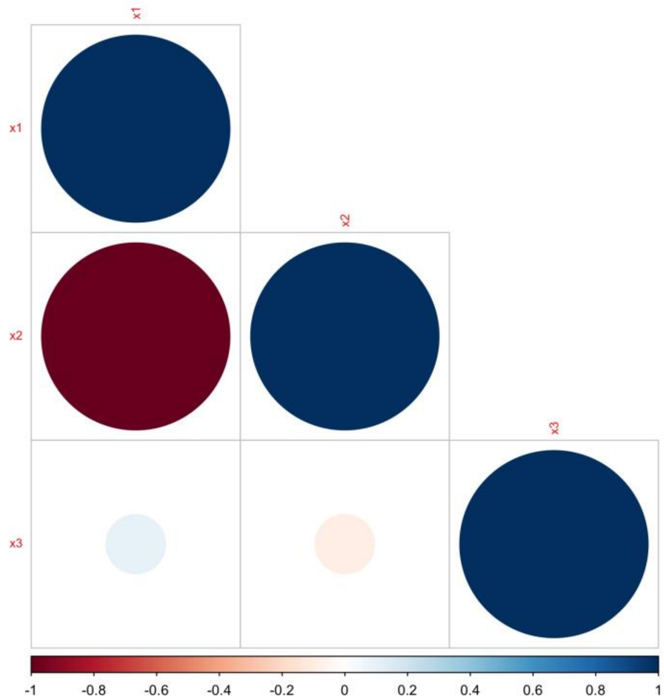
```
Correlation Matrix:
> print(correlation_matrix)
            [,1]        [,2]        [,3]         [,4]        [,5]         [,6]
[1,]  0.11216160 -0.09679700  0.11216160 -0.010755222  0.05070319  0.038411508
[2,] -0.21412670  0.18479428 -0.21412670  0.020532697 -0.09679700 -0.073331062
[3,] -0.02379186  0.02053270 -0.02379186  0.002281411 -0.01075522 -0.008147896
[4,] -0.02379186  0.02053270 -0.02379186  0.002281411 -0.01075522 -0.008147896
[5,]  0.11216160 -0.09679700  0.11216160 -0.010755222  0.05070319  0.038411508
[6,]  0.08497091 -0.07333106  0.08497091 -0.008147896  0.03841151  0.029099628
[7,]  0.11216160 -0.09679700  0.11216160 -0.010755222  0.05070319  0.038411508
[8,] -0.15974532  0.13786240 -0.15974532  0.015318044 -0.07221364 -0.054707300
            [,7]        [,8]
[1,]  0.05070319 -0.25658888
[2,] -0.09679700  0.48985149
[3,] -0.01075522  0.05442794
[4,] -0.01075522  0.05442794
[5,]  0.05070319 -0.25658888
[6,]  0.03841151 -0.19438551
[7,]  0.05070319 -0.25658888
[8,] -0.07221364  0.36544476
```

```
+   print("\nNo variables with absolute correlation greater than 0.5 found.")
+ }
[1] "\nNo variables with absolute correlation greater than 0.5 found."
```

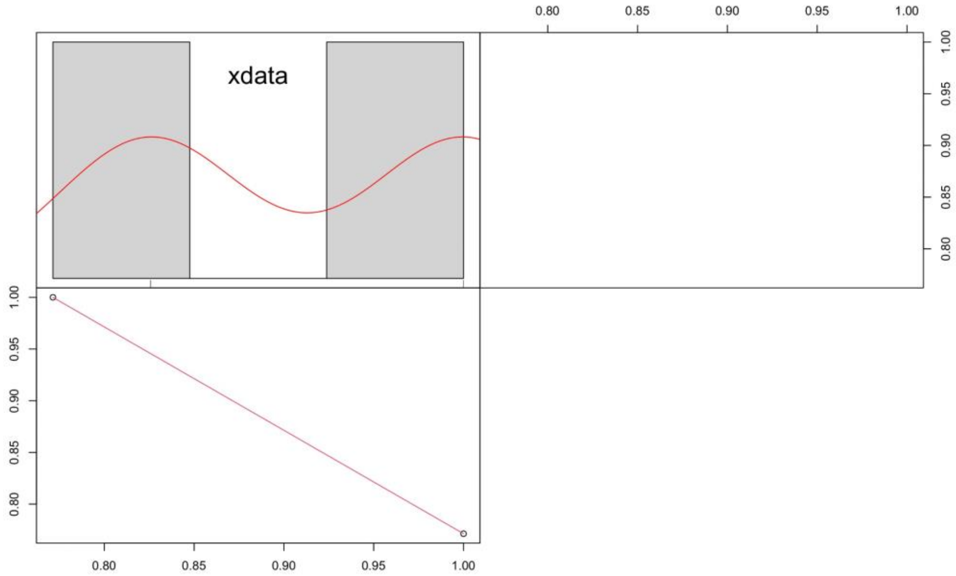
SCATTER PLOT



CORRELATION MATRIX PLOT



CORRELATION CHART USING PERFORMANCE ANALYTICS



EXPERIMENT NO 4

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

Theory:

A frequency table is a tabular summary of the frequency of occurrences of values or categories in a dataset, displaying the count or percentage of each value or category.

Outliers are data points that significantly deviate from the rest of the dataset, often indicating anomalies, errors, or rare events, and are typically identified using statistical methods such as the interquartile range.

Source Code:

R-script

```
# Sample data for age age <- c(25, 32, 28, 41,
20, 35, 23, 18, 45, 62)


# Frequency tables with different binning strategies


# Unequal intervals (adjust number of bins as needed) break_interval <-
seq(from=min(age),to=max(age), length=9) age_freq <- cut(age,
breaks=break_interval,right=TRUE, include.lowest = TRUE) table(age_freq) #
Print frequency table


# Equal intervals (adjust number of bins as needed) break_interval2 <-
seq(from=min(age),to=max(age), length=5) age_freq2 <- cut(age,
breaks=break_interval2,right=TRUE, include.lowest = TRUE) table(age_freq2) #
Print frequency table


# Histogram with breakpoints
hist(age, breaks=break_interval)


# Density plot hist(age, freq=FALSE)
lines(density(age), lwd=3, col="blue")
```

Frequency table by group (replace with your own data structure)
This example demonstrates by team and position team <-

c("A", "A", "A", "B", "B", "B", "C", "C", "C", "D")

position <- c("Manager", "Engineer", "Engineer", "Manager", "Sales", "Marketing", "Director", "Analyst", "Analyst", "Intern")

df <- data.frame(team, position)

library(dplyr)

df %>% group_by(team, position) %>% summarize(Freq=n())

Boxplot for age distribution

boxplot(age, ylab="Age(years)")

Identifying outliers (example with sample data) outliers

<- c(0.6,-0.6,0.1,-0.2,-1.0,0.4,0.3,-1.8,1.1,6.0,-10)

plot(outliers,rep(0,length(outliers)),yaxt="n",ylab="",bty="n",cex=2,cex.axis=1.5,cex.lab=1.5) abline(h=0,col="gray",lty=2)

arrows(length(outliers),0.5,length(outliers)+0.1,0.1,lwd=2)

text(length(outliers)+0.2,0.7,labels="outlier",cex=3)

Output:

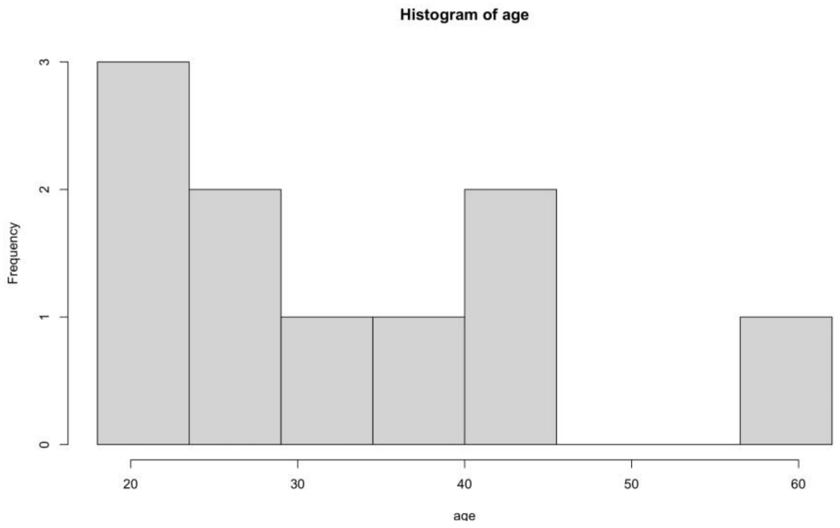
```
> table(age_freq)  # Print frequency table
age_freq
[18,23.5] (23.5,29] (29,34.5] (34.5,40] (40,45.5] (45.5,51] (51,56.5]
        3         2         1         1         2         0         0
(56.5,62]
        1
```

```
> table(age_freq2)  # Print frequency table
age_freq2
[18,29] (29,40] (40,51] (51,62]
      5       2       2       1
>
```

Frequency table by group

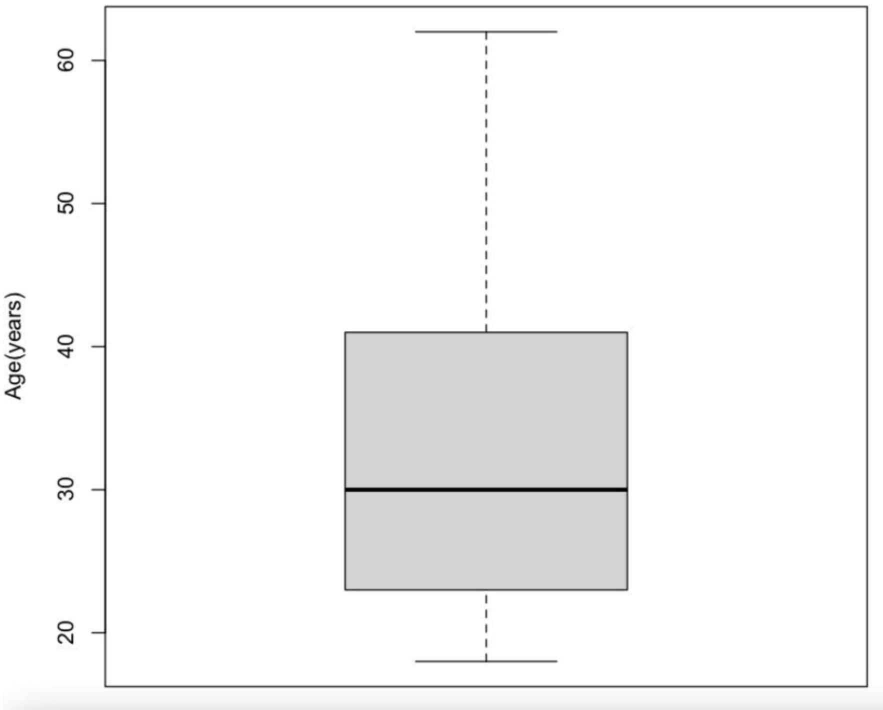
```
> df %>% group_by(team, position) %>% summarize(Freq=n())
`summarise()` has grouped output by 'team'. You can override using the
`.groups` argument.
# A tibble: 8 × 3
# Groups:   team [4]
  team  position   Freq
  <chr> <chr>     <int>
1 A     Engineer      2
2 A     Manager       1
3 B     Manager       1
4 B     Marketing     1
5 B     Sales         1
6 C     Analyst       2
7 C     Director      1
8 D     Intern        1
```

HISTOGRAM

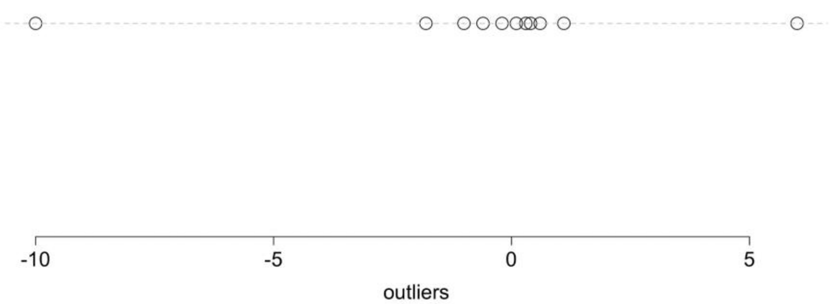


DENSITY PLOT

BOX PLOT-



OUTLIERS-



EXPERIMENT NO 5

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

Theory:

Probability distributions describe the likelihood of outcomes in a dataset or random experiment. Common distributions include the normal distribution, representing symmetrical data; the Poisson distribution, modeling rare events; and the binomial distribution, for binary outcomes. They help analyze uncertainty and make predictions in various fields, including statistics and finance.

Source Code:

R-script

Binomial Distribution

```
# Probability of getting 5 successes in 8 trials with success probability 1/6

x_prob_binomial <- choose(8, 5) * (1/6)^5 * (1 - 1/6)^3 # Probability of

each outcome (0 to 8 successes) all_probs <- sapply(0:8, function(x)

choose(8, x) * (1/6)^x * (1 - 1/6)^(8-x))


# Print all probabilities and their sum (should be close to

1) cat("Probabilities of each outcome (0-8 successes):\n")

print(all_probs) cat("Sum of probabilities (should be close

to 1):\n") print(sum(all_probs)) # Handles potential NAs

# Round probabilities to 3 decimal places and create a barplot

rounded_probs <- round(all_probs, 3)

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

# Cumulative probability of at most 3 successes p_less_than_3 <-

sum(all_probs[1:4]) # Sum probabilities from 0 to 3 successes cat("Cumulative

probability (at most 3 successes):\n") print(p_less_than_3) # This represents

Pr(X <= 3)
```

Poisson Distribution

```
Probability of 3 occurrences with lambda = 3.22 lambda <- 3.22

prob_3_occurrences <- exp(-lambda) * (lambda^3) /

factorial(3) Probabilities of 0 to 10 occurrences
```

all_probs_poisson <- sapply(0:10, function(x) exp(-lambda) *

(lambda^x) / factorial(x))

Round probabilities to 3 decimal places

rounded_probs_poisson <- round(all_probs_poisson, 3)

#Barplot with adjusted y-axis limits for better visualization

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

Cumulative probability of at most 2 occurrences p_less_than_2 <-

sum(all_probs_poisson[1:3]) # Sum probabilities from 0 to 2 occurrences cat("Cumulative

probability (at most 2 occurrences):\n") print(p_less_than_2) # This represents Pr(X <= 2)

Uniform Distribution min <- 0 max <- 100

Sequence of x-values for the uniform distribution function

xpos <- seq(min, max, by = 0.5)

Removed commented-out code for demonstration purposes

Calculate the cumulative distribution function (CDF)

punif_value <- ((xpos - min) / (max - min))

cat("Cumulative probability (X <= 15):\n")

print(punif_value[31]) # Probability for X = 15 (index 31)

Normal Distribution

Generate sequence of numbers from -15 to +10 x <- seq(-15, 10, length = 101)

Increased number of points for smoother curve # Calculate the probability

density function (PDF) assuming unknown mean and standard deviation pdf_normal <- (1 / (sqrt(2 * pi) * sd(x))) * exp(-((x - mean(x))^2) / (2 *

(sd(x))^2))

Plot the PDF

plot(x, pdf_normal, type = "l", lwd = 2, xlab = "x", ylab = "Density", main = "Normal Distribution PDF")

Calculate the CDF using an empirical approach cdf_normal <-

sapply(x, function(val) sum(x <= val) / length(x))

Plot the CDF

plot(x, cdf_normal, type = "l", lwd = 2, xlab = "x", ylab = "Cumulative Probability", main = "Normal Distribution CDF (empirical)")

QQ-Plot for normality check

norm_samp <- rnorm(100) qq

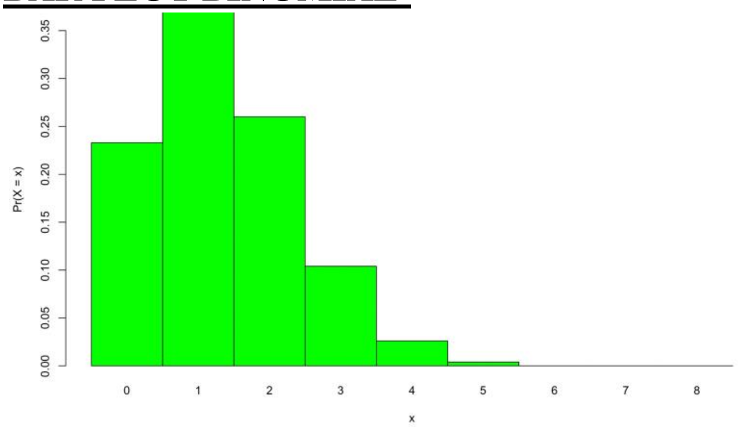
Output:

BINOMIAL DISTRIBUTION

```
> #### Binomial Distribution
>
> # Probability of getting 5 successes in 8 trials with success probability 1/6
> x_prob_binomial <- choose(8, 5) * (1/6)^5 * (1 - 1/6)^3
>
> # Probability of each outcome (0 to 8 successes)
> all_probs <- sapply(0:8, function(x) choose(8, x) * (1/6)^x * (1 - 1/6)^(8-
x))
>
> # Print all probabilities and their sum (should be close to 1)
> cat("Probabilities of each outcome (0-8 successes):\n")
Probabilities of each outcome (0-8 successes):
> print(all_probs)
[1] 2.325680e-01 3.721089e-01 2.604762e-01 1.041905e-01 2.604762e-02
[6] 4.167619e-03 4.167619e-04 2.381497e-05 5.953742e-07
> cat("Sum of probabilities (should be close to 1):\n")
Sum of probabilities (should be close to 1):
> print(sum(all_probs))  # Handles potential NAs
[1] 1
```

```
successes
> cat("Cumulative probability (at most 3 successes):\n")
Cumulative probability (at most 3 successes):
> print(p_less_than_3)  # This represents Pr(X <= 3)
[1] 0.9693436
>
```

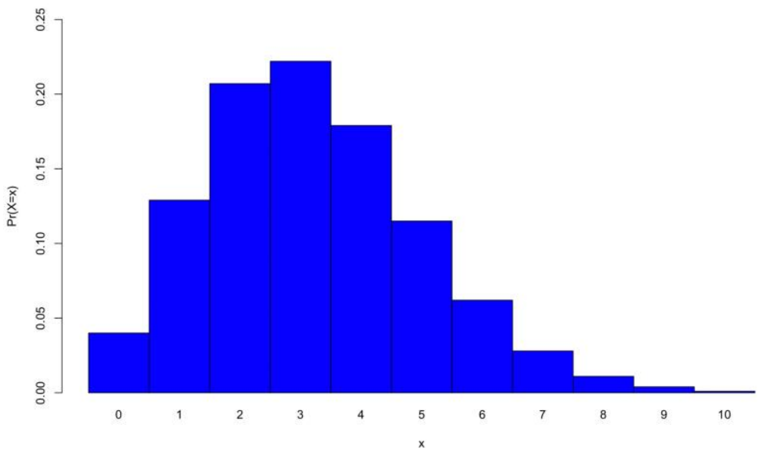
BAR PLOT BINOMIAL-



POISSON DISTRIBUTION-

```
>
> # Cumulative probability (at most 2 occurrences)
> p_less_than_2 <- sum(all_probs_poisson[1:3])
>
> # Print cumulative probability
> cat("Cumulative probability (at most 2 occurrences):\n")
Cumulative probability (at most 2 occurrences):
> print(p_less_than_2)
[1] 0.3757454
```

BAR PLOT POISSON DISTRIBUTION-

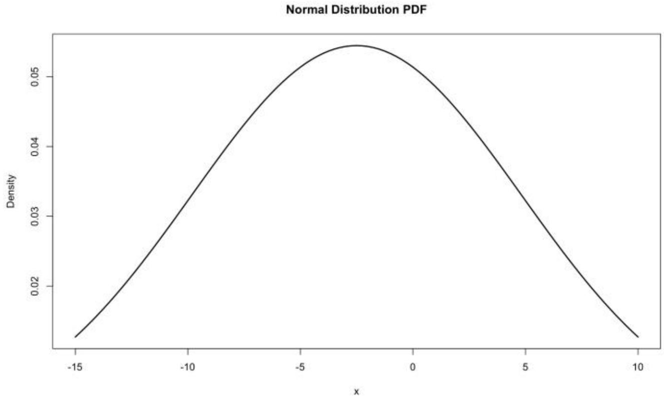


UNIFORM DISTRIBUTION-

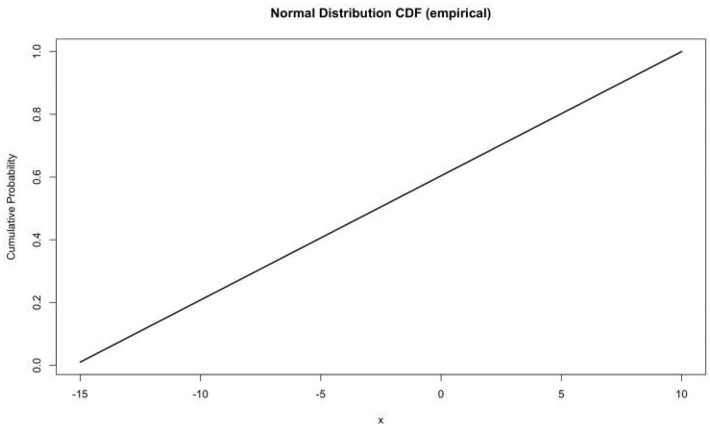
```
>
> # Sequence of x-values for the uniform distribution functi
> xpos <- seq(min, max, by = 0.5)
>
> # Removed commented-out code for demonstration purposes
>
> # Calculate the cumulative distribution function (CDF)
> punif_value <- ((xpos - min) / (max - min))
> cat("Cumulative probability (X <= 15):\n")
Cumulative probability (X <= 15):
> print(punif_value[31])  # Probability for X = 15 (index 31
[1] 0.15
```

NORMAL DISTRIBUTION-

PDF PLOT-



CDF PLOT-



NORMAL Q-Q PLOT