Program No.: 6

Aim: To implement different probability distributions in R.

Theory:

R provides various probability distributions like normal, Poisson, and uniform.

Common functions:

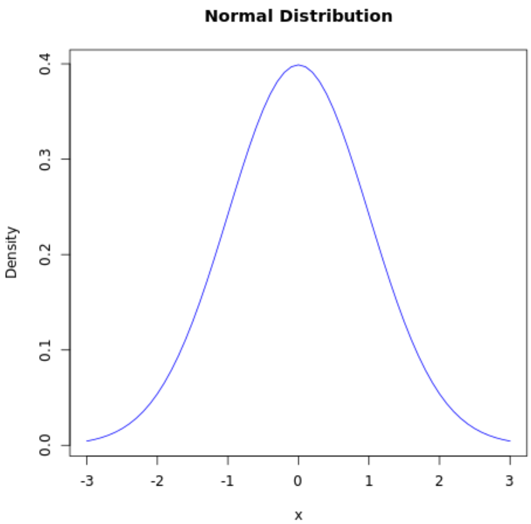
- dnorm(), pnorm() – Normal distribution
- dpois(), ppois() – Poisson distribution
- dunif(), punif() – Uniform distribution

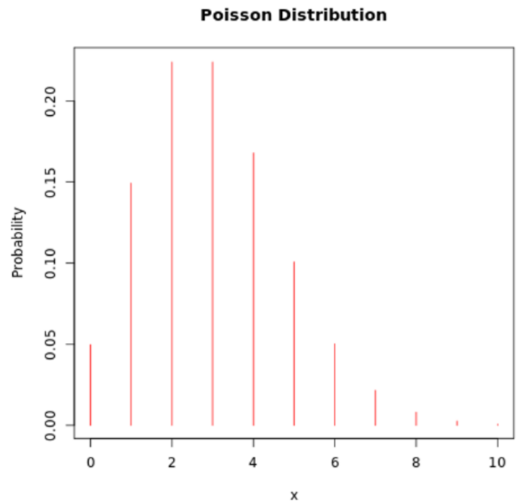
Source Code:

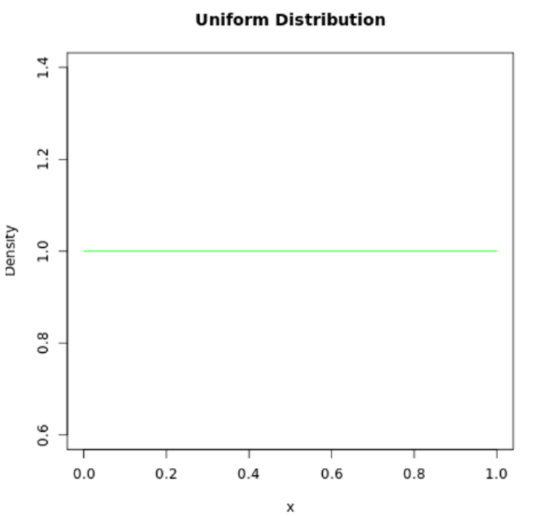
```
# Normal Distribution

x <- seq(-3, 3, by=0.1)

y <- dnorm(x, mean=0, sd=1)

plot(x, y, type="l", col="blue", main="Normal Distribution", ylab="Density")


# Poisson Distribution

x <- 0:10

y <- dpois(x, lambda=3)

plot(x, y, type="h", col="red", main="Poisson Distribution", ylab="Probability")


# Uniform Distribution

x <- seq(0, 1, by=0.01)

y <- dunif(x, min=0, max=1)
```

plot(x, y, type="l", col="green", main="Uniform Distribution", ylab="Density")

Output:







Program No.: 7

Aim: To perform exploratory data analysis (EDA) and calculate statistical measures.

Theory:

EDA helps in summarizing data using statistical measures like range, mean, variance, median, and standard deviation. These metrics describe data centrality and spread.

Source Code:

```
data <- c(12, 15, 20, 22, 25, 30, 35, 40)

range_val <- range(data)

mean_val <- mean(data)

var_val <- var(data)

median_val <- median(data)

sd_val <- sd(data)

print(paste("Range:", range_val[1], "to", range_val[2]))

print(paste("Mean:", mean_val))

print(paste("Variance:", var_val))

print(paste("Median:", median_val))

print(paste("Standard Deviation:", sd_val))
```

Output:

```
[1] "Range: 12 to 40"
[1] "Mean: 24.875"
[1] "Variance: 93.2678571428571"
[1] "Median: 23.5"
[1] "Standard Deviation: 9.65752852146227"
```

Program No.: 8

Aim: To generate random numbers and visualize their distribution using histograms.

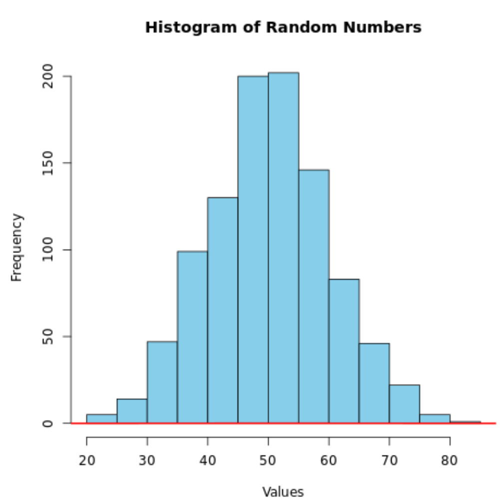
Theory:

R provides random number generators like rnorm(), runif(), rpois(). Histograms help visualize the frequency distribution.

Source Code:

```
# Generating random numbers

set.seed(123)

data <- rnorm(1000, mean=50, sd=10)


# Density and histogram

hist(data, breaks=20, col="skyblue", main="Histogram of Random Numbers", xlab="Values")

lines(density(data), col="red", lwd=2)
```

Output:



Program No.: 9

Aim: To study correlation using scatter plots and investigate relationships between variables.

Theory:

Correlation measures the strength and direction of a linear relationship between two variables.

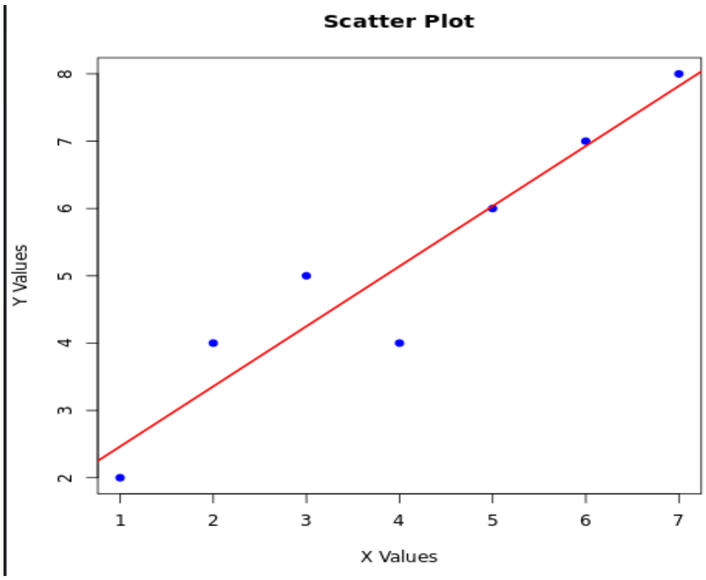
- cor(x, y) gives correlation coefficient.
- plot(x, y) visualizes the relationship.

Source Code:

```
x <- c(1, 2, 3, 4, 5, 6, 7)

y <- c(2, 4, 5, 4, 6, 7, 8)


# Correlation

correlation <- cor(x, y)

print(paste("Correlation Coefficient:", correlation))


# Scatter plot

plot(x, y, main="Scatter Plot", xlab="X Values", ylab="Y Values", col="blue", pch=19)

abline(lm(y~x), col="red", lwd=2)
```

Output:

```
[1] "Correlation Coefficient: 0.947622554473629"
```


Scatter Plot
Y Values
X Values
1
2
3
4
5
6
7
8

Program No.: 10

Aim: To perform statistical inference using contingency tables and chi-square test in R.

Theory:

Statistical inference draws conclusions about populations based on sample data.

- Contingency tables summarize categorical data.
- Chi-square test (chisq.test()) checks the association between variables.

Source Code:

```
# Creating contingency table

data <- matrix(c(20, 15, 30, 35), nrow=2, byrow=TRUE)

rownames(data) <- c("Group1", "Group2")

colnames(data) <- c("Success", "Failure")

print(data)

# Chi-square test

test_result <- chisq.test(data)

print(test_result)
```

Output:

```
       Success Failure
Group1      20      15
Group2      30      35

        Pearson's Chi-squared test with Yates' continuity correction

data:  data
X-squared = 0.7033, df = 1, p-value = 0.4017
```

Program No.: 11

Aim: To implement linear and logistic regression models in R.

Theory:

Regression analysis models relationships between dependent and independent variables.

- **Linear Regression:** For continuous dependent variables using lm().
- **Logistic Regression:** For binary dependent variables using glm(family=binomial).

Source Code:

```
# Linear Regression

x <- c(1, 2, 3, 4, 5)

y <- c(2, 4, 5, 4, 5)

model1 <- lm(y ~ x)

print(summary(model1))

plot(x, y, main="Linear Regression", xlab="X", ylab="Y", col="blue", pch=19)

abline(model1, col="red", lwd=2)


# Logistic Regression

data <- data.frame(

  score = c(45, 55, 60, 70, 80, 85, 95),

  pass = c(0, 0, 0, 1, 1, 1, 1)

)

model2 <- glm(pass ~ score, data=data, family=binomial)

print(summary(model2))
```

plot(data$score, data$pass, main="Logistic Regression", xlab="Score", ylab="Pass/Fail", col="green", pch=19)

curve(predict(model2, data.frame(score=x), type="response"), add=TRUE, col="red", lwd=2)

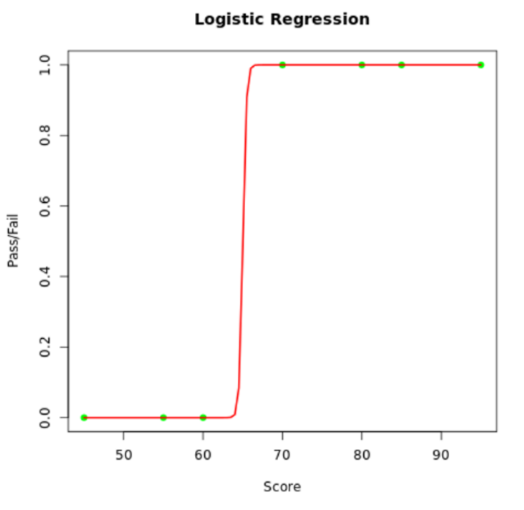
Output:

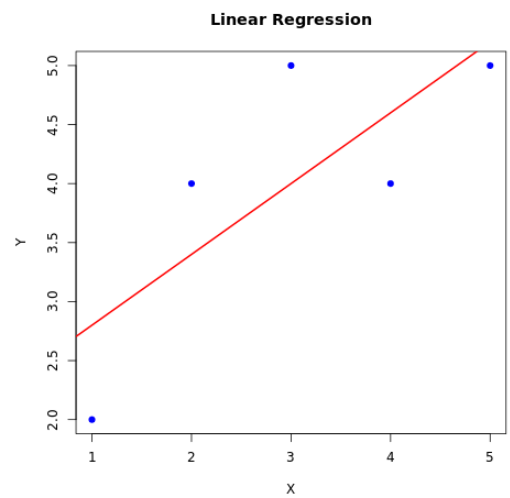
```
Call:
lm(formula = y ~ x)

Residuals:
   1    2    3    4    5
-0.8  0.6  1.0 -0.6 -0.2

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.2000     0.9381   2.345    0.101
x             0.6000     0.2828   2.121    0.124
```

```
Residual standard error: 0.8944 on 3 degrees of freedom
Multiple R-squared:   0.6,    Adjusted R-squared:  0.4667
F-statistic:   4.5 on 1 and 3 DF,  p-value: 0.124

Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred

Call:
glm(formula = pass ~ score, family = binomial, data = data)
```

```
Deviance Residuals:
        1          2          3          4          5          6
-2.110e-08 -2.110e-08 -1.125e-05  1.182e-05  2.110e-08  2.110e-08
        7
 2.110e-08

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  -304.691 688779.521       0        1
score           4.687  10525.799       0        1

(Dispersion parameter for binomial family taken to be 1)
```

```
    Null deviance: 9.5607e+00  on 6  degrees of freedom
Residual deviance: 2.6628e-10  on 5  degrees of freedom
AIC: 4

Number of Fisher Scoring iterations: 25
```


Logistic Regression
Pass/Fail
0.0
0.2
0.4
0.6
0.8
1.0
50
60
70
80
90
Score


Linear Regression
Y
2.0
2.5
3.0
3.5
4.0
4.5
5.0
1
2
3
4
5
X