SML - Practical - QB - Solutions - By - Th3_

1: Write a program to:

a. Print any built in data set of R

```
data()  # will give all inbuilt datasets
```

b. Get information about the data set.

```
df <- data.frame(iris)
df
str(df)  # information of dataset
```

c. Find dimensions of the data set and view the names of the variables

```
cat("Dimensions of inbuilt dataset iris are:",dim(df),"\n")
```

d. Find the name of each row in the first column.

```
cat("Names of each rows in 1st column are:",rownames(df),"\n")
```

e. Print all the values of any variable of your choice from the data set.

```
cat("All values in column petal length",(df$Petal.Length),"\n")
```

f. Get a statistical summary of the data set.

```
cat("Summary of iris dataset is:",summary(df))
```

Conclusion: The iris dataset contains 150 observations of iris flowers with 4 numeric attributes and a categorical variable. Exploring its structure, dimensions, and summary statistics provides a clear understanding of its features for analysis.

2: Write a program:

a. To Load and Print any built in data set in R.

```
# sample dataset
monthly_sales <- c(120, 150, 130, 160, 170, 180, 160, 170, 200, 210, 190,220)
```

b. Calculate Variance.

```
sales-vari <- var(monthly-sales)
cat ("The variance in monthly sales is:", sales-vari)
```

c. Calculate Standard Deviation.

```
sales-sd <- sd(monthly_sales)
```

d. Calculate Range

```
range(iris$Sepal.Length)
```

e. Calculate Mean Deviation and Skewness.

```
# Sample data
data<- c(5, 7, 8, 9, 10)

# MEAN
mean_value<- mean(data)

 # Calculate mean deviation
mean_deviation<- mean(abs(data - mean_value))

# Print the result
print(paste("Mean Deviation:", mean_deviation))

#SKEWNESS
install.packages("e1071") # Installing the package
library(e1071) # Load the package

skewness_value<- skewness(data)
print(skewness_value)
```

Conclusion: These calculations summarize variability and symmetry of data distributions.

4: Write a program:

a. To perform Spearman Rank Correlation Test to evaluate the association between two variables and interpret the result.

```
x <- mtcars$mpg; y <- mtcars$wt
cat("Spearman Correlation:", cor(x, y, method = "spearman"), "\n")
print(cor.test(x, y, method = "spearman"))
result = cor(x,y,method ="spearman")

#using cor() method
if(result>0) {
print("Positive Correlation")
} else if(result<0) {
print("Negative Correlation")
} else {
print("Zero Correlation")
}
```

Conclusion: Spearman correlation captures monotonic relationships.

SML - Practical - QB - Solutions - By - Th3_

5: Write a program:

a. Calculate the probability of getting heads when flipping a fair coin

```
# Probability of heads
outcomes <- c("Heads", "Tails")
classical_prob <- length(outcomes[outcomes == "Heads"]) / length(outcomes)
cat("Probability of getting Heads:", classical_prob, "\n")
```

b. Calculate the probability of drawing a spade from a standard deck of 52 cards.

```
# Probability of drawing a spade
deck <- rep(c("Spades", "Hearts", "Diamonds", "Clubs"), each = 13)
classical_prob_spade <- length(deck[deck == "Spades"]) / length(deck)
cat("Probability of drawing a Spade:", classical_prob_spade, "\n")
```

Conclusion: Probabilities quantify the likelihood of outcomes.

7: Write a program:

a. To Use Bayes Theorem in R.

```
bayesTheorem <- function(pA, pB, pBA) {
  return((pA * pBA) / pB)
}

# Example probabilities
pRain <- 0.2
pCloudy <- 0.4
pCloudyRain <- 0.85

# Bayes theorem result
pRainCloudy <- bayesTheorem(pRain, pCloudy, pCloudyRain)
cat("P(Rain | Cloudy):", pRainCloudy, "\n")
```

Conclusion: Bayes theorem allows updating beliefs based on new evidence.

8: Write a program:

a. For implementation of Extrapolation in R.

```
extrapolate <- function(x, y, xp) if (xp < min(x)) y[1] + diff(y) / diff(x) * (xp - x[1]) else y[2] + diff(y) /
diff(x) * (xp - x[2])
cat("Extrapolated:", extrapolate(c(0.3, 0.5), c(1.8, 2.1), 1.2), "\n")
```

Conclusion: Extrapolation extends patterns in existing data.

10: Write a program:

a. Based on Chi-Square Distribution using dchisq, pchisq, qchisq and rchisq functions

```
# dchisq
df <- 6
vec <- 1:4
print("Density function values:")
print(dchisq(vec, df))

# pchisq
df <- 5
print("Calculating P(X ≤ 5):")
print(pchisq(5, df, lower.tail = TRUE))
print("Calculating P(X > 5):")
print(1 - pchisq(5, df))

# qchisq
print("75th percentile (Q):")
print(qchisq(0.75, df))

# rchisq
x <- rchisq(50000, df)
hist(x, freq = FALSE, xlim = c(0, 16), ylim = c(0, 0.2))
curve(dchisq(x, df), from = 0, to = 15, col = 'red', lwd = 2, add = TRUE)
```

Conclusion: Chi-square functions are useful for hypothesis testing and modeling variability.