SML Practical Question Bank Solutions

Q1. Write a program to:

a) Print any built in dataset of R.

b) Get information about the dataset.

c) Find dimensions of the dataset and view the names of the variables.

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

e) Print all the values of any variable of your choice from the dataset.

f) Get statistical summary of the dataset

Write your conclusion.

Ans.

Aim: To explore a built-in R dataset by displaying its structure, dimensions, and row names, extracting values of a selected variable, and generating a statistical summary.

Code:

```r
# a) Print a built-in dataset
# Printing the built-in 'mtcars' dataset
print(mtcars)

# b) Get information about the dataset
# Using the 'str()' function to get the structure of the dataset
str(mtcars)
# Using the 'ncol()' and 'nrow()' functions to get the number of rows and columns
cat("Number of rows in the dataset:", nrow(mtcars), "\n")
cat("Number of columns in the dataset:", ncol(mtcars), "\n")

# c) Find dimensions of the dataset and view the names of the variables
# Dimensions can be obtained using the 'dim()' function
# Variable names can be viewed using the 'names()' function
cat("Dimensions of the dataset:", dim(mtcars), "\n")
cat("Names of the variables:", names(mtcars), "\n")

# d) Find name of each row in the first column
# We can use the row names along with the first column to display them
cat("Names of rows in the first column:", rownames(mtcars), "\n")

# e) Print all the values of any variable of your choice from the dataset
# Printing all the values of the 'mpg' (miles per gallon) variable
cat("Values of the 'mpg' variable:", mtcars$mpg, "\n")

# f) Get statistical summary of the dataset
# Using the 'summary()' function to get the statistical summary
```

```r
cat("Statistical summary of the dataset:\n")
summary(mtcars)
```

Conclusion:

The ‘mtcars’ built-in dataset has 352 observations of 11 variables. It is commonly used for exploratory data analysis, and to build regression models.

Q2. Write a program to:

a) Load and Print any built in dataset in R.
b) Calculate Variance.
c) Calculate Standard Deviation.
d) Calculate Range.
e) Calculate Mean Deviation and Skewness.
Write your conclusion.

Ans.

Aim: To load a built-in dataset and calculate its statistical measures such as variance, standard deviation, range, mean deviation, and skewness.

Code:

```r
# a) Load and print a built-in dataset in R
df <- data.frame(iris)
print(df)

# b) Calculate Variance
# Calculating the variance of the Sepal.Length column
cat("Variance of Sepal.Length:", var(df$Sepal.Length), "\n")

# c) Calculate Standard Deviation
# Calculating the standard deviation of Sepal.Length column
cat("Standard Deviation of Sepal.Length:", sd(df$Sepal.Length), "\n")

# d) Calculate Range
# The range is the difference between the maximum and minimum values
range_sepal_length <- range(df$Sepal.Length)
cat("Range of Sepal.Length: From", range_sepal_length[1], "to", range_sepal_length[2], "\n")

# e) Calculate Mean Deviation and Skewness
# Mean deviation is the average of absolute differences from the mean
mean_deviation_sepal_length <- mean(abs(df$Sepal.Length - mean(df$Sepal.Length)))
cat("Mean Deviation of Sepal.Length:", mean_deviation_sepal_length, "\n")
```

```r
# To calculate skewness, we need the 'e1071' package (for skewness function)
install.packages("e1071")
library(e1071)

# Skewness of Sepal.Length
cat("Skewness of Sepal.Length:", skewness(df$Sepal.Length), "\n")
```

Conclusion:
Statistical measures such as variance, standard deviation, range, mean deviation, and skewness provide useful insights into the distribution of the ‘Sepal.Length’ variable of the ‘iris’ dataset.

Q3. Write a program:

a) To perform Pearson Correlation Test to evaluate the association between two variables and interpret the result using Graph.
Write your conclusion.

Ans.

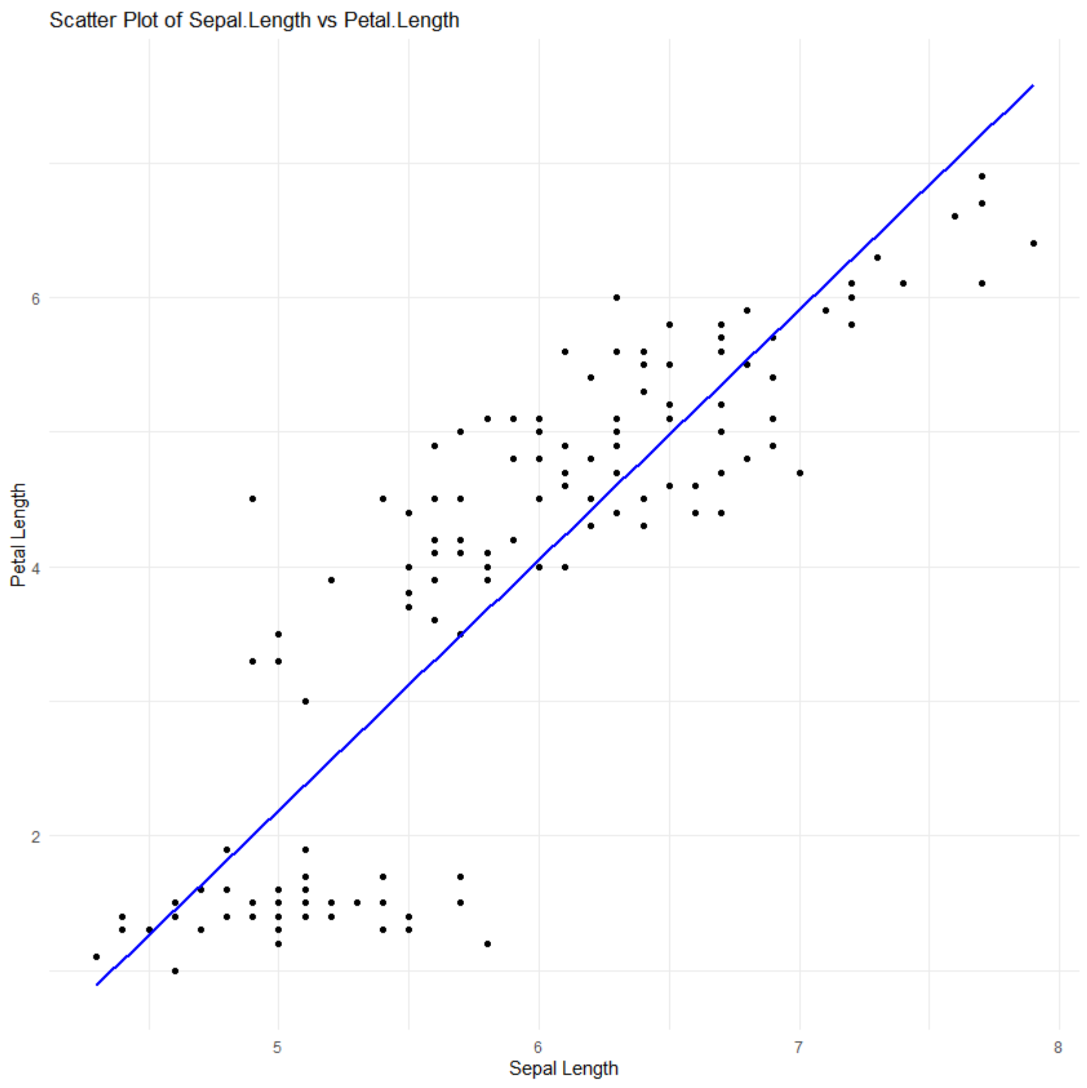
Aim: To perform Pearson Correlation Test to evaluate the association and direction of relationship between two variables, and interpret the result using a graph.

Code:

```r
# Load the 'iris' dataset
df <- data.frame(iris)

# a) Perform Pearson Correlation Test
cor_test <- cor(df$Sepal.Length, df$Petal.Length, method="pearson")

# Displaying results of the Pearson correlation test
cat("Pearson Correlation Test Results:\n")
cat("Correlation Coefficient:", cor_test, "\n")

# Interpreting the correlation coefficient
if (cor_test > 0) {
    cat("Positive correlation\n")
} else if(cor_test < 0) {
    cat("Negative correlation\n")
} else {
    cat("Zero or no correlation\n")
}
```

```
# scatter plot with a correlation line
library(ggplot2)

    ggplot(df, aes(x = Sepal.Length, y = Petal.Length)) +


    geom_point() +  # Scatter plot of the data points
    geom_smooth(method = "lm", color = "blue", se = FALSE) +  # Linear regression line
    labs(title = "Scatter Plot of Sepal.Length vs Petal.Length",
         x = "Sepal Length", y = "Petal Length") +
    theme_minimal()
```

Output (Graph):



Conclusion:

The Pearson Correlation Test shows the strength and direction of the linear relationship between 'Petal.Length' and 'Sepal.Length' variables of the 'iris' dataset.

Q4. Write a program:

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

Ans.

Aim: To perform Spearman Rank Correlation Test to evaluate the association and direction of relationship between two variables, and interpret the result using a graph.

Code:

```r
# Load the 'mtcars' dataset
df <- data.frame(mtcars)

# a) Perform Spearman Rank Correlation Test
cor_test <- cor(df$mpg, df$wt, method="spearman")

# Displaying results of the Spearman Rank Correlation Test
cat("Spearman Rank Correlation Test Results:\n")
cat("Correlation Coefficient:", cor_test, "\n")

# Interpreting the correlation coefficient
if (cor_test > 0) {
    cat("Positive correlation\n")
} else if(cor_test < 0) {
    cat("Negative correlation\n")
} else {
    cat("Zero or no correlation\n")
}
```

Conclusion:
The Spearman Correlation Test assesses how well the relationship between the two variables ‘mpg’ and ‘wt’ can be described using a monotonic function.

Q5. Write a program:

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

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

Write your conclusion.

Ans.

Aim: To calculate the probability of getting a head when flipping a fair coin, and the probability of drawing a spade from a standard deck of cards.

Code:

```r
# a) Calculate the probability of getting heads when flipping a fair coin
outcomes <- c("heads", "tails")

total_outcomes <- length(outcomes) # Total no. of possible outcomes
favourable_outcomes <- length(outcomes[outcomes=="heads"])
# No. of outcomes in which we get a head

prob_head <- favourable_outcomes / total_outcomes
cat("Probability of getting a head when flipping a fair coin:", prob_head, "\n")

# Calculate the probability of drawing a spade from a standard deck of 52 cards.
deck <- rep(c("spades", "hearts", "diamonds", "clubs"), each = 13)

total_cards <- length(deck)
spades <- length(deck[deck == "spades"])

prob_spade <- spades / total_cards
cat("Probability of drawing a spade from a standard deck of cards:", prob_spade, "\n")
```

Conclusion:
Basic probability principles can be used to determine the likelihood of simple events like getting a head when flipping a coin, and drawing a particular card based on total and favorable outcomes.

Q6. Write a program:

a) To Calculate Conditional Probability.

Write your conclusion.

Ans.

Aim: To calculate the conditional probability P(B|A), which is the probability of event B occurring, given that event A has already occurred.

Code:

```r
# a) Calculate Conditional Probability P(B|A)

# Defining the sample space (for a fair 6-sided die)
sample_space <- 1:6

# Defining events
A <- c(2, 4, 6) # (even numbers)
B <- c(4, 5, 6) # (numbers greater than 3)

# P(A) - Probability of event A occurring (even numbers)
P_A <- length(A) / length(sample_space)

# P(A ∩ B) - Probability of both event A and event B occurring (even and greater than 3)
A_and_B <- intersect(A, B)
P_A_and_B <- length(A_and_B) / length(sample_space)

# Conditional Probability P(B|A) = P(A ∩ B) / P(A)
P_B_given_A <- P_A_and_B / P_A

# Display the result
cat("The conditional probability P(B|A) is:", P_B_given_A, "\n")
```

Conclusion:

Conditional probability accurately provides the likelihood of an event B occurring, given that another event A is true.

Q7. Write a program:

a) To Use Bayes Theorem in R.

Write your conclusion.

Ans.

Aim: To use Bayes Theorem in R, to calculate conditional probability P(R|C).

Code:

```r
# a) Use Bayes Theorem in R

# Suppose we know the following probabilities:
# P(rain) = 0.20
# P(cloudy) = 0.40
# P(cloudy|rain) = 0.85
# To calculate P(rain|cloudy):

P_R <- 0.2
P_C <- 0.4
P_CR <- 0.85

P_RC <- P_CR * P_R / P_C # P(rain|cloudy)
cat("P(rain|cloudy):", P_RC)
```

Conclusion:

Bayes Theorem provides a way to update the probability estimate for an event based on new found evidence.

Q8. Write a program:

a) For implementation of Extrapolation in R.

Write your conclusion.

Ans.

Aim: To implement Extrapolation in R.

Code:

```r
extrapolation <- function(x, y, xp) {
    n <- length(x)

    if (xp < min(x)) { # if value to be predicted is smaller than available range
        slope <- (y[2] - y[1]) / (x[2] - x[1])
        yp <- y[1] + slope * (xp - x[1])

    } else if (xp > max(x)) { # if value to be predicted is larger than available range
        slope <- (y[n] - y[n - 1]) / (x[n] - x[n - 1])
        yp <- y[n] + slope * (xp - x[n])

    } else { # if value to be predicted falls within the available range
        stop("xp must be outside the range of x for extrapolation") # Stops program execution
    }

    return (yp)
}

#Main Program
x <-c( 0.3, 0.5, 0.7, 0.9)
y <- c(1.8, 2.1, 2.4, 2.7)
xp <- 1.2

extrapolated_value <- extrapolation(x, y, xp)
cat("Extrapolated value at x = 1.2 is", extrapolated_value, "\n")
```

Conclusion:

Extrapolation can be used to predict values outside the known range for data that has a linear relationship.

Q9. Write a program to:

a) Generate Samples using Sampling Functions.
Write your conclusion.

Ans.

Aim: To generate samples using different sampling functions.

Code:

```r
#a) Generate Samples using Sampling Functions

population <- data.frame(mtcars)

# 1: Simple Random Sample (using the 'sample()' function)
simple_sample <- sample(population$cyl, size = 5, replace = FALSE)
cat("Sample generated through simple random sampling:", simple_sample, "\n")

# 2: Stratified Sampling (using the 'createDataPartition()' function)
# Generate a sample of size 5 from the population with replacement
install.packages("caret")
library("caret")
strfied_sample <- createDataPartition(population$mpg, p=0.5, list = FALSE)
cat("Sample generated through stratified sampling:", strfied_sample, "\n")

# 3. Systematic Sampling (using the 'seq()' function)
sys_sample <- seq(from=0, to=50, by = 10)
cat("Sample generated through systematic sampling:", sys_sample, "\n")

# 4. Biased Sampling (using a vector)
index <- c(1:5)
cat("Sample generated through biased sampling:", index, "\n")
```

Conclusion:
Different sampling functions in R can be used to generate samples based on different sampling methods.

Q10. Write a program:

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

Ans.

Aim: To explore Chi-Square Distribution functions such as dchisq, pchisq, qchisq, and rchisq.

Code:

```r
# a) Based on Chi-Square Distribution using dchisq, pchisq, qchisq, rchisq functions

# Defining the intervals (x) and degrees of freedom (df)
x <- seq(0, 5, by=1)
df <- 2

# 1. dchisq(): Chi-Square Probability Density Function (PDF)
dchisq_vals <- dchisq(x, df)
cat("The value of the probability density function up to x = 5 is:\n", dchisq_vals, "\n\n")

# 2.pchisq(): Chi-Square Cumulative Distribution Function (CDF)
pchisq_vals <- pchisq(x, df)
cat("The cumulative probability up to x = 5 is:\n", pchisq_vals, "\n\n")

# 3. qchisq(): Quantile function (Inverse CDF)
# Calculating the quantile for the cumulative probability 0.95
qchisq_vals <- qchisq(0.95, df)
cat("The quantile for the cumulative probability 0.95 is:", qchisq_vals, "\n\n")

# 4. rchisq(): Generate random values from a Chi-Square distribution
rchisq_vals <- rchisq(10, df)
cat("10 random values from a Chi-Square distribution with df = 2:\n", rchisq_vals, "\n")
```

Conclusion:
The different chi-square distribution functions in R can be used to calculate probability densities, cumulative probabilities, quantiles of cumulative probabilities, critical values, and to generate random values from a Chi-Square distribution.