Data Analytics with R

Practical Component

1. Perform the following:
a) Assign different type of values to variables and display the type of variable.
Assign different types such as Double, Integer, Logical, Complex and Character and understand the difference between each data type.
b) Demonstrate Arithmetic and Logical Operations with simple examples.
C) Demonstrate generation of sequences and creation of vectors.
d) Demonstrate Creation of Matrices
e) Demonstrate the Creation of Matrices from Vectors using Binding Function.
f) Demonstrate element extraction from vectors, matrices and arrays

```
# Assigning to Variables
c <- "Hello, R!" # Character
d=3.14159 # Double
i <- 42 # Integer
l <- TRUE # Logical
cmp <- 3 + 2i # Complex

# Display variable types
cat("d is of type:", typeof(d))
cat("i is of type:", typeof(i), "\n")
cat("l is of type:", typeof(l), "\n")
cat("cmp is of type:", typeof(cmp), "\n")
cat("c is of type:", typeof(c), "\n")

# Arithmetic Operations
cat("d + i:", d + i, "\n")
cat("d * i:", d * i, "\n")
cat("d / i", d / i, "\n")

# Logical Operations
cat("Logical AND l && T:", l && T,"\n")
cat("Logical OR l || F:", l || F, "\n")
cat("Logical NOT !l:", !l, "\n")

# Generate a sequence of integers from 1 to 10
sv <- 1:10
# Create a vector
v <- c(5, 10, 15, 20, 25)
cat("Sequence of Integers:", sv, "\n")
cat("My Vector:", v, "\n")
```

```r
# Create a matrix
m <- matrix(1:12, nrow <- 3, ncol <- 4)
cat("My Matrix:\n")
cat(m)

# Create vectors
v1 <- c(1, 2, 3)
v2 <- c(4, 5, 6)
v3 <- c(7, 8, 9)

# Bind vectors into a matrix
mv <- rbind(v1, v2,v3)
cat("Matrix from Vectors:\n")
cat(mv)

# Extract elements from vectors
e1 <- mv[2]
e2 <- sv[5]

# Extract elements from matrices
e3 <- m[2, 3]
e4 <- mv[1, 2]
cat("Element from My Vector:", e1, "\n")
cat("Element from Sequence Vector:", e2, "\n")
cat("Element from My Matrix:", e3, "\n")
cat("Element from Matrix from Vectors:", e4, "\n")
```

Program 2

Assess the Financial Statement of an Organization being supplied with 2 vectors of data: Monthly Revenue and Monthly Expenses for the Financial Year. You can create your own sample data vector for this experiment. Calculate the following financial metrics:

a. Profit for each month.
b. Profit after tax for each month (Tax Rate is 30%).
c. Profit margin for each month equals to profit after tax divided by revenue.
d. Good Months – where the profit after tax was greater than the mean for the year.
e. Bad Months – where the profit after tax was less than the mean for the year.
f. The best month – where the profit after tax was max for the year.
g. The worst month – where the profit after tax was min for the year.

```r
# Sample data for monthly revenue and expenses (in $1000 units)
monthly_revenue <- c(50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 155, 165)
monthly_expenses <- c(30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85)

# Calculate profit for each month
profit <- monthly_revenue - monthly_expenses

# Calculate profit after tax for each month (Tax Rate is 30%)
tax_rate <- 0.30
profit_after_tax <- profit * (1 - tax_rate)

# Calculate profit margin for each month as a percentage
profit_margin <- (profit_after_tax / monthly_revenue) * 100

# Calculate the mean profit after tax for the year
mean_profit_after_tax <- mean(profit_after_tax)

# Determine good months, bad months, best month, and worst month
good_months <- profit_after_tax > mean_profit_after_tax
bad_months <- profit_after_tax < mean_profit_after_tax
best_month <- which.max(profit_after_tax)
worst_month <- which.min(profit_after_tax)

# Format results as vectors with appropriate units and precision
profit <- round(profit * 1000, 2)  # Convert to $1000 units
profit_after_tax <- round(profit_after_tax * 1000, 2)  # Convert to $1000 units
profit_margin <- round(profit_margin, 0)  # Remove decimal points for percentage

# Create a data frame to store the results
results <- data.frame(
  Month = 1:12,
  Profit = profit,
  ProfitAfterTax = profit_after_tax,
  ProfitMargin = profit_margin,
  GoodMonth = good_months,
  BadMonth = bad_months
)
```

```r
# Print the results
cat("Profit for each month (in $1000 units):\n")
cat(results$Profit, "\n\n")

cat("Profit after tax for each month (in $1000 units):\n")
cat(results$ProfitAfterTax, "\n\n")

cat("Profit margin for each month (in %):\n")
cat(results$ProfitMargin, "\n\n")

cat("Good Months (Profit after tax greater than mean):\n")
cat(results$Month[results$GoodMonth], "\n\n")

cat("Bad Months (Profit after tax less than mean):\n")
cat(results$Month[results$BadMonth], "\n\n")
cat("Best Month (Max Profit after tax):\n")
cat(results$Month[best_month], "\n\n")

cat("Worst Month (Min Profit after tax):\n")
cat(results$Month[worst_month], "\n\n")

# Export the results to a CSV file
write.csv(results, "financial_metrics.csv", row.names = FALSE)
```

Program 3

Develop a program to create two 3 X 3 matrices A and B and perform the following operations a) Transpose of the matrix b) addition c) subtraction d) multiplication

```r
# Create matrices A and B
A = matrix(1:9, nrow = 3, ncol = 3)
B = matrix(9:1, nrow = 3, ncol = 3)

# a) Transpose of the matrix
A_t = t(A)
B_t = t(B)

# b) Addition
sum = A + B

# c) Subtraction
diff = A - B
```

```
# d) Multiplication
prod = A %*% B

# Print the results
cat("Matrix A:\n")
print(A)
cat("Matrix B:\n")
print(B)
cat("Transpose of A:\n")
print(A_t)
cat("Transpose of B:\n")
print(B_t)
cat("Addition of A and B:\n")
print(sum)
cat("Subtraction of A and B:\n")
print(diff)
cat("Multiplication of A and B:\n")
print(prod)
```

Program 4:

Develop a program to find the factorial of given number using recursive function calls.

```
# Recursive function to calculate the factorial
fact = function(n) {
  if (n == 0) {
    return(1)
  } else {
    return(n * fact(n - 1))
  }
}
# Input a number from the user
n = as.integer(readline("Enter a non-negative integer: "))

if (n < 0) {
  cat("Factorial is not defined for negative numbers.\n")
} else {
  result = fact(n)
  cat("The factorial of", n, "is", result, "\n")
}
```

Program 5:

Develop an R Program using functions to find all the prime numbers up to a specified number by the method of Sieve of Eratosthenes

```r
# Function to find all prime numbers up to a specified number using the Sieve of Eratosthenes
sieve_of_eratosthenes <- function(n) {
  if (n < 2) {
    cat("No prime numbers in the specified range.\n")
    return()
  }

is_prime <- rep(TRUE, n)
  is_prime[1] <- FALSE  # 1 is not prime

  p <- 2
  while (p^2 <= n) {
    if (is_prime[p]) {
      for (i in seq(p^2, n + 1, by = p)){
        is_prime[i] <- FALSE
      }
    }
    p <- p + 1
  }

  primes <- which(is_prime)
  cat("Prime numbers up to", n, "are:\n", primes, "\n")
}

# Input a number from the user
n <- as.integer(readline("Enter a positive integer: "))
sieve_of_eratosthenes(n)
```

Program 7:

Develop R program to create a Data Frame with following details and do the following operations.

itemCode	itemCategory	itemPrice
1001	Electronics	700
1002	Desktop Supplies	300
1003	Office Supplies	350
1004	USB	400
1005	CD Drive	800

a) Subset the Data frame and display the details of only those items whose price is greater than or equal to 350.

b) Subset the Data frame and display only the items where the category is either "Office Supplies" or "Desktop Supplies"

c) Create another Data Frame called "item-details" with three different fields itemCode, ItemQtyonHand and ItemReorderLvl and merge the two frames

```
#Develop R program to create a Data Frame with following details and do
#the following operations:
#itemCode itemCategory      itemPrice
#1001     Electronics       700
#1002     Desktop Supplies  300
#1003     Office Supplies   350
#1004     USB               400
#1005     CD Drive          800
#a) Subset the Data frame and display the details of only those item
#   whose price is greater than or equal to 350.
#b) Subset the Data frame and display only the items where the category
#   is either “Office Supplies” or “Desktop Supplies”
#c) Create another Data Frame called “item-details” with three different
#   fields itemCode, ItemQtyonHand and ItemReorderLvl and merge the two frames

# Creating the data frame
itemCode <- c(1001:1005)#, 1002, 1003, 1004, 1005)
itemCategory <- c("Electronics", "Desktop Supplies", "Office Supplies", "USB",
"CD Drive")
itemPrice <- c(700, 300, 350, 400, 800)

# Creating the data frame using the above vectors
items_df <- data.frame(itemCode, itemCategory, itemPrice)

# Displaying the created data frame
print("Data Frame with Items Information:")
print(items_df)
```

```r
# Summary statistics of itemPrice
print(summary(items_df$itemPrice))

# Filtering items with price greater than 350
high_priced_items <- subset(items_df, itemPrice > 350)
print("Items with Price greater than 400:")
print(high_priced_items)

# Subset the data frame for items with category as "Office Supplies" or "Desktop
Supplies"
filtered_items <- subset(items_df, itemCategory %in% c("Office Supplies",
"Desktop Supplies"))

# Display the subsetted data frame
print("Items with 'Office Supplies' or 'Desktop Supplies' category:")
print(filtered_items)

# Creating the 'item-details' data frame
itemCode <- c(1001, 1002, 1003, 1004, 1005)
ItemQtyonHand <- c(20, 15, 30, 10, 25)
ItemReorderLvl <- c(5, 10, 8, 3, 7)

# Creating the data frame using the above vectors
item_details <- data.frame(itemCode, ItemQtyonHand, ItemReorderLvl)

# Displaying the created data frame
print("Data Frame 'item-details':")
print(item_details)

# Merging the two data frames based on 'itemCode'
merged_data <- merge(items_df, item_details, by = "itemCode")

# Displaying the merged data frame
print("Merged Data Frame:")
print(merged_data)
```

Program 6:

The built-in data set mammals contain data on body weight versus brain weight. Develop R commands to:

a) Find the Pearson and Spearman correlation coefficients. Are they similar?

b) Plot the data using the plot command.

c) Plot the logarithm (log) of each variable and see if that makes a difference.

```r
#The built-in data set mammals contain data on body weight versus brain weight.
#Develop R commands to:
#  a) Find the Pearson and Spearman correlation coefficients. Are they similar?
#  b) Plot the data using the plot command.
#  c) Plot the logarithm (log) of each variable and see if that makes a
difference.

data("mammals", package = "MASS")
# a) Finding the Pearson and Spearman correlation coefficients
pcorr <- cor(mammals$body, mammals$brain, method = "pearson")
scorr <- cor(mammals$body, mammals$brain, method = "spearman")

# Displaying the correlation coefficients
cat("Pearson correlation coefficient: ", pcorr, "\n")
cat("Spearman correlation coefficient: ", scorr, "\n")

# b) Plotting the data using the plot command
plot(mammals$body, mammals$brain, xlab = "Body Weight", ylab = "Brain Weight",
main = "Mammals' Body Weight vs. Brain Weight")

# c) Plotting the logarithm (log) of each variable and checking the difference
log_body <- log(mammals$body)
log_brain <- log(mammals$brain)

plot(log_body, log_brain, xlab = "Log Body Weight", ylab = "Log Brain Weight",
main = "Log of Mammals' Body Weight vs. Brain Weight")
```

Program 8

Let us use the built-in dataset air quality which has Daily air quality measurements in New York, May to September 1973. Develop R program to generate histogram by using appropriate arguments for the following statements.

a) Assigning names, using the air quality data set.

b) Change colors of the Histogram

c) Remove Axis and Add labels to Histogram

d) Change Axis limits of a Histogram

e) Add Density curve to the histogram.

```r
# Load the air quality dataset
data("airquality", package = "datasets")
```

```r
# a) Assigning names
names(airquality) <- c("Ozone", "Solar.R", "Wind", "Temp", "Month", "Day")

# b) Change colors of the Histogram
hist(airquality$Ozone, col = "skyblue", main = "Histogram of Ozone", xlab =
"Ozone Levels", ylab = "Frequency")

# c) Remove Axis and Add labels to Histogram
hist(airquality$Ozone, col = "lightgreen", main = "", xlab = "", ylab = "", axes
= FALSE)
title(xlab = "Ozone Levels", ylab = "Frequency", main = "Histogram of Ozone")

# d) Change Axis limits of a Histogram
hist(airquality$Ozone, col = "lightpink", main = "Histogram of Ozone", xlab =
"Ozone Levels", ylab = "Frequency", xlim = c(0, 150), ylim = c(0, 40))

# e) Add Density curve to the histogram
# Remove missing values in 'Ozone' column
cleaned_data <- na.omit(airquality$Ozone)

# Create a histogram of 'Ozone' column
hist(cleaned_data, col = "lightblue", main = "Histogram of Ozone", xlab = "Ozone
Levels", ylab = "Frequency")

# Add a density curve to the histogram
lines(density(cleaned_data), col = "red")
```

Program 9

Design a data frame in R for storing about 20 employee details. Create a CSV file named "input.csv" that defines all the required information about the employee such as id, name, salary, start_date, dept. Import into R and do the following analysis.
a) Find the total number rows & columns
b) Find the maximum salary
c) Retrieve the details of the employee with maximum salary
d) Retrieve all the employees working in the IT Department.
e) Retrieve the employees in the IT Department whose salary is greater than 20000 and write these details into another file "output.csv"

```r
# Importing the CSV file into R
emp_data <- read.csv("empdata.csv")

# a) Find the total number rows & columns
num_rows <- nrow(emp_data)
```

```r
num_cols <- ncol(emp_data)
cat("Total number of rows:", num_rows, "\n")
cat("Total number of columns:", num_cols, "\n")

# b) Find the maximum salary
max_salary <- max(emp_data$salary)
cat("Maximum salary:", max_salary, "\n")

# c) Retrieve the details of the employee with maximum salary
employee_max_salary <- emp_data[emp_data$salary == max_salary, ]
cat("Employee with maximum salary:\n")
print(employee_max_salary)

# d) Retrieve all the employees working in the IT Department
emp_IT <- subset(emp_data, dept == "IT")
cat("Employees working in the IT Department:\n")
print(emp_IT)

# e) Retrieve the employees in the IT Department whose salary is greater than
20000
employees_IT_high_salary <- subset(emp_IT, salary > 20000)
cat("Employees in the IT Department with salary > 20000:\n")
print(employees_IT_high_salary)

# Write these employees to a new CSV file
write.csv(employees_IT_high_salary, "output.csv", row.names = FALSE)
```

Program 10

Using the built in dataset mtcars which is a popular dataset consisting of the design and fuel consumption patterns of 32 different automobiles. The data was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973-74 models). Format A data frame with 32 observations on 11 variables : [1] mpg Miles/(US) gallon, [2] cyl Number of cylinders [3] disp Displacement (cu.in.), [4] hp Gross horsepower [5] drat Rear axle ratio,[6] wt Weight (lb/1000) [7] qsec 1/4 mile time, [8] vs V/S, [9] am Transmission (0 = automatic, 1 = manual), [10] gear Number of forward gears, [11] carb Number of carburetors. Develop R program, to solve the following:
a) What is the total number of observations and variables in the dataset?
b) Find the car with the largest hp and the least hp using suitable functions
c) Plot histogram / density for each variable and determine whether continuous variables are normally distributed or not. If not, what is their skewness?
d) What is the average difference of gross horse power(hp) between automobiles with 3 and 4 number of cylinders(cyl)? Also determine the difference in their standard deviations.
e) Which pair of variables has the highest Pearson correlation?

```r
# Load the mtcars dataset
data(mtcars)

# a) Total number of observations and variables
observations <- nrow(mtcars)
variables <- ncol(mtcars)
cat("Total number of observations:", observations, "\n")
cat("Total number of variables:", variables, "\n")

# b) Car with the largest and least hp
car_largest_hp <- mtcars[which.max(mtcars$hp), ]
car_least_hp <- mtcars[which.min(mtcars$hp), ]
cat("Car with the largest hp:\n")
print(car_largest_hp)
cat("Car with the least hp:\n")
print(car_least_hp)

# c) Histogram / density plot and skewness
par(mfrow = c(4, 3), mar = c(3, 3, 1, 1))  # Adjusting margin size
for (i in 1:ncol(mtcars)) {
  hist(mtcars[, i], main = names(mtcars)[i], xlab = "", col = "skyblue")
  lines(density(mtcars[, i]), col = "red")  # Adding density curve
}

# Calculate skewness
library(e1071)
skew <- sapply(mtcars, skewness)
cat("Skewness of variables:\n")
print(skew)

# d) Average difference and standard deviation between hp for 3 and 4 cylinders
hp_diff_mean <- mean(mtcars$hp[mtcars$cyl == 3]) - mean(mtcars$hp[mtcars$cyl ==
4])
hp_diff_sd <- sd(mtcars$hp[mtcars$cyl == 3]) - sd(mtcars$hp[mtcars$cyl == 4])
cat("Average difference in hp between 3 and 4 cylinders:", hp_diff_mean, "\n")
cat("Difference in standard deviations of hp between 3 and 4 cylinders:",
hp_diff_sd, "\n")

# e) Pair of variables with the highest Pearson correlation
cor_matrix <- cor(mtcars)
diag(cor_matrix) <- 0  # Exclude diagonal values
max_corr <- which(cor_matrix == max(cor_matrix), arr.ind = TRUE)
```

```r
cat("Pair of variables with the highest Pearson correlation:",
rownames(cor_matrix)[max_corr[1,1]], "and", colnames(cor_matrix)[max_corr[1,2]],
"\n")

# c) Histogram / density plot and skewness
par(mfrow = c(4, 3))  # To display histograms for each variable in a grid
for (i in 1:ncol(mtcars)) {
  hist(mtcars[, i], main = names(mtcars)[i], xlab = "", col = "skyblue")
  lines(density(mtcars[, i]), col = "red")  # Adding density curve
}

# Calculate skewness
library(e1071) #functions for data analysis & ML
skew <- sapply(mtcars, skewness)#apply the skewness() function to each column
cat("Skewness of variables:\n")
print(skew)

# d) Average difference and standard deviation between hp for 3 and 4 cylinders
hp_diff_mean <- mean(mtcars$hp[mtcars$cyl == 3]) - mean(mtcars$hp[mtcars$cyl ==
4])
hp_diff_sd <- sd(mtcars$hp[mtcars$cyl == 3]) - sd(mtcars$hp[mtcars$cyl == 4])
cat("Average difference in hp between 3 and 4 cylinders:", hp_diff_mean, "\n")
cat("Difference in standard deviations of hp between 3 and 4 cylinders:",
hp_diff_sd, "\n")

# e) Pair of variables with the highest Pearson correlation
cor_matrix <- cor(mtcars)
diag(cor_matrix) <- 0  # Exclude diagonal values
max_corr <- which(cor_matrix == max(cor_matrix), arr.ind = TRUE)
cat("Pair of variables with the highest Pearson correlation:",
rownames(cor_matrix)[max_corr[1,1]], "and", colnames(cor_matrix)[max_corr[1,2]],
"\n")
```

Program 11

Demonstrate the progression of salary with years of experience using a suitable data set (You can create your own dataset).
a Plot the graph visualizing the best fit line on the plot of the given data points.
b Plot a curve of Actual Values vs. Predicted values to show their correlation and performance of the model.
c Interpret the meaning of the slope and y-intercept of the line with respect to the given data.
d Implement using lm function.
e Save the graphs and coefficients in files.
f Attach the predicted values of salaries as a new column to the original data set and save the data as a new CSV file.

```r
# Generating sample data for demonstration
set.seed(123)
Years_of_Experience <- 1:40
Salary <- 30000 + 1500 * Years_of_Experience + rnorm(40, mean = 0, sd = 2000)  # Generating salaries

# Create a data frame with the generated data
data <- data.frame(Years_of_Experience, round(Salary, digits = -1 ))

# Perform linear regression using lm() function
model <- lm(Salary ~ Years_of_Experience, data = data)

# Plot the data points and the best-fit line
plot(Salary ~ Years_of_Experience, data, col = "blue", main = "Salary vs. Years of Experience")
abline(model, col = "red")

# Save the plot as an image file (e.g., PNG)
png("Salary_Experience_Plot03.png")
plot(Salary ~ Years_of_Experience, data, col = "blue", main = "Salary vs. Years of Experience")
abline(model, col = "red")
dev.off()

# Generate predicted values from the model
predicted_values <- round( predict(model), digits=0)

# Plotting Actual vs. Predicted values
plot(Salary, predicted_values, main = "Actual vs. Predicted Salaries", xlab = "Actual Salary", ylab = "Predicted Salary", col = "green")

# Save the plot as an image file (e.g., PNG)
jpeg("Actual_vs_Predicted_Salary.jpg")
plot(Salary, predicted_values, main = "Actual vs. Predicted Salaries", xlab = "Actual Salary", ylab = "Predicted Salary", col = "green")
dev.off()

# Attach predicted values as a new column to the original dataset
data$Predicted_Salary <- predicted_values

# Save the dataset as a new CSV file
write.csv(data, "Salary_Experience_Predictions.csv", row.names = FALSE)
```