Set-I

1. .(i) Write a function called kelvin_to_celsius() that takes a temperature in Kelvin and returns that temperature in Celsius (Hint: To convert from Kelvin to Celsius you subtract 273.15)
(ii) Write suitable R code to compute the mean, median ,mode of the following values c(90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20)

(iii) Write R code to find 2nd highest and 3rd Lowest value of above problem. '

Code:

```
value<-c(90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20)
mean<-mean(value)
mean
median<-median(value)
median
mode<-names(table(value))[table(value)==max(table(value))]
mode
n<-as.integer(readline(prompt="Enter a value of temperature:"))
celsius<-(n-273.15)
celsius
second_highest<-sort(unique(value),decreasing=TRUE)[2]
second_highest
third_lowest<-sort(unique(value))[3]
third_lowest
```

2. Explore the airquality dataset. It contains daily air quality measurements from New York during a period of five months:
• Ozone: mean ozone concentration (ppb),
• Solar.R: solar radiation (Langley),
• Wind: average wind speed (mph),
• Temp: maximum daily temperature in degrees Fahrenheit,
• Month: numeric month (May=5, June=6, and so on),
• Day: numeric day of the month (1-31).
i. Compute the mean temperature(don't use build in function)
ii. Extract the first five rows from airquality.
iii. Extract all columns from airquality *except* Temp and Wind
iv. Which was the coldest day during the period?

v. How many days was the wind speed greater than 17 mph?

code:

```
data(airquality)
mean<-sum(airquality$Temp)/length(airquality$Temp)
mean
five_rows<-airquality[1:5,]
five_rows
except<-airquality[ ,!(names(airquality) %in% c("Temp", "Wind"))]
except
coldest_day<-airquality$Day[which.min(airquality$Temp)]
coldest_day
speed<-sum(airquality$Wind>17)
speed
```

3. (i) Get the Summary Statistics of air quality dataset

(ii)Melt airquality data set and display as a long – format data?

(iii)Melt airquality data and specify month and day to be "ID variables"?

(iv)Cast the molten airquality data set with respect to month and date features

(v) Use cast function appropriately and compute the average of Ozone, Solar.R , Wind and temperature per month?

Code:

```
# (i) Get the summary statistics of the airquality dataset
summary(airquality)

# (ii) Melt the airquality dataset and display as long-format data
library(reshape2)
melted_data <- melt(airquality)
print(melted_data)

# (iii) Melt the airquality dataset and specify month and day as ID variables
melted_data_id <- melt(airquality, id.vars = c("Month", "Day"))
print(melted_data_id)

# (iv) Cast the molten airquality dataset with respect to month and day features
cast_data <- dcast(melted_data_id, Month + Day ~ variable)
print(cast_data)
```

```
    # (v) Use the cast function appropriately and compute the average of Ozone, Solar.R,
Wind, and Temperature per month
    cast_avg <- dcast(melted_data, Month ~ variable, mean)
    print(cast_avg)
```

4.(i) Find any missing values(na) in features and drop the missing values if its less than 10% else replace that with mean of that feature.
(ii) Apply a linear regression algorithm using Least Squares Method on "Ozone" and "Solar.R" (iii)Plot Scatter plot between Ozone and Solar and add regression line created by above model

Code:

```
# (i) Find and handle missing values
missing_values <- sum(is.na(airquality))
missing_percentage <- missing_values / nrow(airquality) * 100

if (missing_percentage < 10) {
  # Drop missing values
  airquality <- na.omit(airquality)
} else {
  # Replace missing values with mean of the feature
  airquality[is.na(airquality)] <- colMeans(airquality, na.rm = TRUE)
}

# (ii) Apply linear regression using Least Squares Method
model <- lm(Ozone ~ Solar.R, data = airquality)

# (iii) Plot scatter plot and regression line
plot(airquality$Solar.R, airquality$Ozone, xlab = "Solar.R", ylab = "Ozone", main = "Scatter
plot of Ozone vs Solar.R")
abline(model, col = "red")
```

Set-II

1. **(i)Write a function to find the factorial of a given number using "for" Loop**

   Code:

```
function(n) {
  result <- 1
  for (i in 1:n) {
    result <- result * i
  }
  return(result)
}
number<-5
fact<-factorial(number)
fact
```

(ii) Create a 3x4 matrix with 12 random numbers between 1-100; have the matrix be filled our row by-row, instead of column-by-column. Name the columns of the matrix *uno, dos, tres, cuatro,* and the rows *x, y, z.* Scale the matrix by 10 and save the result.
(iii) Extract the column called "uno" as a vector from the original matrix and save the result

Code:

```
# Set the seed for reproducibility
set.seed(42)

# Create a 3x4 matrix filled row by row with random numbers
matrix_data <- matrix(sample(1:100, 12), nrow = 3, byrow = TRUE)

# Name the columns
colnames(matrix_data) <- c("uno", "dos", "tres", "cuatro")

# Name the rows
rownames(matrix_data) <- c("x", "y", "z")

# Scale the matrix by 10
scaled_matrix <- matrix_data * 10

# Extract the column called "uno" as a vector
column_uno <- matrix_data[, "uno"]

# Print the original matrix, scaled matrix, and the extracted column
print(matrix_data)
print(scaled_matrix)
print(column_uno)
```

2. In 1936, Edgar Anderson collected data to quantify the geographic variations of iris flowers. The data set consists of 50 samples from each of the three sub-species (*iris setosa*, *iris virginica,* and *iris versicolor*).Four features were measured in centimeters (cm): the lengths and the widths of both sepals

and petals

(i)Find dimension, Structure, Summary statistics, Standard Deviation of all features.
(ii)Find mean and standard deviation of features groped by three species of Iris flowers (Iris setosa, Iris virginica and Iris versicolor)
(iii)Find quantile value of sepal width and length
(iv)create new data frame named iris1 which have a new column name Sepal.Length.Cate that categorizes "Sepal.Length" by quantile
(v) Average value of numerical varialbes by two categorical variables: Species and Sepal.Length.Cate.

Code:

```
data(iris)
dim(iris)
str(iris)
summary(iris)
df<-data.frame(iris)
standard<-sd(df$Sepal.Length)
standard
standard<-sd(df$Sepal.Width)
standard
standard<-sd(df$Petal.Width)
standard
standard<-sd(df$Petal.Length)
standard
# Calculate mean and standard deviation of features grouped by species
aggregate(. ~ Species, data = iris, FUN = mean)
aggregate(.~Species,data=iris,FUN=sd)
quantile(df$Sepal.Width)
quantile(df$Sepal.Length)
# Create a new data frame
iris1 <- iris

# Categorize Sepal.Length by quantile and create a new column
```

```
iris1$Sepal.Length.Cate <- cut(iris1$Sepal.Length, breaks = quantile(iris1$Sepal.Length)
, labels = FALSE)
# Average value of numerical variables by Species and Sepal.Length.Cate
aggregate(. ~ Species + Sepal.Length.Cate, data = iris1, FUN = mean)
```

3. (i)Plot Scatter plot between sepals width and length grouped by Species
(ii) Plot Scatter plot between petals width and length grouped by Species
(iii)Draw the Box plot for Sepals length grouped by Species
(iv) Draw the Box plot for petals length grouped by Species
(v)Find the correlation among the four features

Code:

```
# Load the iris dataset
data(iris)

# Plot scatter plot between sepal width and length grouped by Species
plot(iris$Sepal.Width, iris$Sepal.Length, col = iris$Species, xlab = "Sepal Width", ylab = "Sepal
Length")
legend("topright", legend = levels(iris$Species), col = 1:length(levels(iris$Species)), pch = 1)

# Plot scatter plot between petal width and length grouped by Species
plot(iris$Petal.Width, iris$Petal.Length, col = iris$Species, xlab = "Petal Width", ylab = "Petal
Length")
legend("topright", legend = levels(iris$Species), col = 1:length(levels(iris$Species)), pch = 1)

# Draw box plot for Sepal length grouped by Species
boxplot(Sepal.Length ~ Species, data = iris, xlab = "Species", ylab = "Sepal Length")

# Draw box plot for Petal length grouped by Species
boxplot(Petal.Length ~ Species, data = iris, xlab = "Species", ylab = "Petal Length")

# Calculate the correlation matrix
cor_matrix <- cor(iris[, 1:4])
```

```
# Print the correlation matrix
print(cor_matrix)
```

4.(i) Randomly Sample the iris dataset such as 50% data for training and 50% for test (ii)find summary statistics of above train and test dataset.
(iii)Create Logistics regression with train data
(iv)Predict the probability of the model using test data
(v)Create Confusion matrix for above test model

```
# Step 1: Load the required packages
library(caret)
library(dplyr)

# Step 2: Load and split the iris dataset
data(iris)
set.seed(123)  # For reproducibility
train_indices <- createDataPartition(iris$Species, p = 0.5, list = FALSE)
train_data <- iris[train_indices, ]
test_data <- iris[-train_indices, ]

# Step 3: Summary statistics of the train and test datasets
train_summary <- summary(train_data)
test_summary <- summary(test_data)
print(train_summary)
print(test_summary)

# Step 4: Create a logistic regression model using train data
logistic_model <- glm(Species ~ ., data = train_data, family = binomial)

# Step 5: Predict the probability of the model using test data
predicted_probs <- predict(logistic_model, newdata = test_data, type = "response")

# Step 6: Create a confusion matrix for the test model
predicted_classes <- ifelse(predicted_probs > 0.5, "virginica", "non-virginica")
confusion_matrix <- table(predicted = predicted_classes, actual = test_data$Species)
```

```
print(confusion_matrix)
```

Set-III

1. Suppose you track your commute times for two weeks (10 days) and you find the following times in minutes 17 16 20 24 22 15 21 15 17 22 Enter this into R as vector data type. (i)create function maxi to find the longest commute time, the function avger to find the average and the function mini to find the minimum.

(ii)Oops, the 24 was a mistake. It should have been 18. How can you fix this? Do so, and then find the new average using above functions.

(iii)How many times was your commute 20 minutes or more?

Code:

```
minutes<-c(17,16,20,24,22,15,21,15,17,22 )
min<-min(minutes)
min
max<-max(minutes)
max
minutes[4]=18
minutes
mean<-mean(minutes)
mean
sum(minutes>=20)
```

2. There is a popular built-in data set in R called "mtcars" (Motor Trend Car Road Tests), which is retrieved from the 1974 Motor Trend US Magazine.

(i)Find the dimension of the data set

(ii)Give the statistical summary of the features.

(iii)Find the largest and smallest value of the variable hp (horsepower).

(iv)Give the mean of mileage per gallon (mpg) with respect to transmission model (feature named as 'am')

(v)Give the median of horsepower (hp) with respect to cylinder displacement(cyl)

```
data(mtcars)
dim(mtcars)
summary(mtcars)
```

```
largest<-max(mtcars$hp)
largest
smallest<-min(mtcars$hp)
smallest
# Step 5: Mean of mileage per gallon (mpg) with respect to transmission model (am)
mean_mpg_am <- tapply(mtcars$mpg, mtcars$am, mean)
print(mean_mpg_am)

# Step 6: Median of horsepower (hp) with respect to cylinder displacement (cyl)
median_hp_cyl <- tapply(mtcars$hp, mtcars$cyl, median)
print(median_hp_cyl)
```

3.(i)Create Scatter plot mpg vs hp, grouped by transmission model (feature named as 'am') (ii)Create Box plot for mpg with respect to transmission model (feature named as 'am') (iii)Create histogram plot which shows statistical distribution of hp (iv)Draw the Bar Chart to show car distribution with respect to number of gears grouped by cylinder.(Grouped or multiple bar chart)

(v)Draw Pie chart which shows the percentage of distribution by number of gears.

```
# Step 1: Load the mtcars dataset
data(mtcars)

# Step 2: Create a scatter plot of mpg vs hp, grouped by transmission model (am)
plot(mpg ~ hp, data = mtcars, col = am, pch = 16, xlab = "Horsepower", ylab = "Mileage per Gallon")
legend("topright", legend = c("Automatic", "Manual"), col = c(1, 2), pch = 16)

# Step 3: Create a box plot of mpg with respect to transmission model (am)
boxplot(mpg ~ am, data = mtcars, xlab = "Transmission Model", ylab = "Mileage per Gallon",
        col = c("red", "blue"), names = c("Automatic", "Manual"))
# Step 4: Create a histogram plot of hp to show the statistical distribution
hist(mtcars$hp, xlab = "Horsepower", ylab = "Frequency", main = "Distribution of Horsepower")
# Step 5: Create a grouped bar chart to show car distribution by number of gears (gear) grouped by
cylinders (cyl)
table_data <- table(mtcars$cyl, mtcars$gear)
barplot(table_data, beside = TRUE, legend = rownames(table_data),
        xlab = "Number of Gears", ylab = "Car Distribution", main = "Car Distribution by Gears and
Cylinders")

# Step 6: Create a pie chart to show the percentage distribution by number of gears (gear)
gear_counts <- table(mtcars$gear)
pie(gear_counts, labels = paste(names(gear_counts), "gears"),
    main = "Percentage Distribution by Number of Gears", col = rainbow(length(gear_counts)))
```

4. (i)Generate a multiple regression model using the built-in dataset mtcars. Establish the relationship between "mpg" as a response variable with "disp","hp" and "wt" as predictor variables . (ii)Plot the multiple regression line model with above model parameters.
(iii) Predict the mileage of the car with dsp=221, hp=102 and wt=2.91

```
# Step 1: Load the mtcars dataset
data(mtcars)

# Step 2: Create the multiple regression model
model <- lm(mpg ~ disp + hp + wt, data = mtcars)

# Step 3: Print the model summary
summary(model)

# Step 4: Plot the multiple regression line
plot(mpg ~ disp, data = mtcars, xlab = "Displacement", ylab = "Mileage per Gallon")
abline(model, col = "red")

# Step 5: Predict the mileage of a car with disp = 221, hp = 102, and wt = 2.91
new_data <- data.frame(disp = 221, hp = 102, wt = 2.91)
predicted_mpg <- predict(model, newdata = new_data)
print(paste("Predicted mileage:", predicted_mpg))
```

Set IV

1. (i) Write a function in R programming to print generate Fibonacci sequence using Recursion in R .
(ii) Find sum of natural numbers up-to 10, without formula using loop statement.
(iii) create a vector 1:10 and Find a square of each number and store that in a separate list.

```
# Step 1: Create the vector from 1 to 10
vector <- 1:10
```

```
# Step 2: Calculate the square of each number and store them in a separate list
squared_list <- vector^2

# Step 3: Print the squared list
print(squared_list)
```

2. **mtcars(motor trend car road test) comprises fuel consumption, performance and 10 aspects of automobile design for 32 automobiles. It comes pre-installed with dplyr package in R.**

(i)Find the dimension of the data set

(ii)Give the statistical summary of the features.

(iii)Print the categorical features in Dataset

(iv)Find the average weight(wt) grouped by Engine shape(vs)

(v)Find the largest and smallest value of the variable weight with respect to Engine shape

```
# Step 1: Load the dplyr package
library(dplyr)

# Step 2: Access the mtcars dataset
data(mtcars)

# (i) Find the dimension of the dataset
dimension <- dim(mtcars)
print(paste("Dimensions of the dataset:", dimension[1], "rows and", dimension[2], "columns"))

# (ii) Give the statistical summary of the features
summary_stats <- summary(mtcars)
print(summary_stats)

# (iii) Print the categorical features in the dataset
categorical_features <- sapply(mtcars, is.factor)
print(names(mtcars)[categorical_features])

# (iv) Find the average weight (wt) grouped by Engine shape (vs)
```

```
average_weight <- mtcars %>%
  group_by(vs) %>%
  summarize(avg_weight = mean(wt))
print(average_weight)

# (v) Find the largest and smallest value of the variable weight with respect to Engine shape
largest_weight <- mtcars %>%
  group_by(vs) %>%
  filter(wt == max(wt)) %>%
  select(vs, wt)
smallest_weight <- mtcars %>%
  group_by(vs) %>%
  filter(wt == min(wt)) %>%
  select(vs, wt)
print(paste("Largest weight per Engine shape:", largest_weight))
print(paste("Smallest weight per Engine shape:", smallest_weight))
```

3.Use ggplot package to plot below EDA questions label the plot accordingly (i)Create weight(wt) vs displacement(disp) scatter plot factor by Engine Shape(vs) (ii) Create horsepower (hp) vs mileage (mgp) scatter plot factor by Engine Shape(vs) (iv)In above(ii) plot , Separate columns according to cylinders(cyl) size

(v) Create histogram plot for horsepower (hp) with bin-width size of 5

Code:

```
library(ggplot2)

# Scatter plot of wt vs disp, factor by vs
ggplot(mtcars, aes(x = disp, y = wt, color = factor(vs))) +
  geom_point() +
  labs(x = "Displacement", y = "Weight", color = "Engine Shape") +
  theme_minimal()
# Scatter plot of hp vs mpg, factor by vs
ggplot(mtcars, aes(x = mpg, y = hp, color = factor(vs))) +
  geom_point() +
  labs(x = "Mileage (mpg)", y = "Horsepower (hp)", color = "Engine Shape") +
  theme_minimal()
# Scatter plot of hp vs mpg, factor by vs with separate columns for cyl
ggplot(mtcars, aes(x = mpg, y = hp, color = factor(vs))) +
  geom_point() +
  facet_wrap(~ cyl) +
  labs(x = "Mileage (mpg)", y = "Horsepower (hp)", color = "Engine Shape") +
  theme_minimal()
```

4. Performing Logistic regression on dataset to predict the cars Engine shape(vs) . (i)Do the EDA analysis and find the features which is impact the Engine shape and use this for model.

(ii) Split the data set randomly with 80:20 ration to create train and test dataset and create logistic model

(iii)Create the Confusion matrix among prediction and test data.

Code:

```
# (i) EDA analysis to identify the features impacting Engine shape (vs)
# You can perform exploratory data analysis (EDA) techniques such as data visualization, correlation analysis,
and feature selection methods (e.g., chi-square test, feature importance, etc.) to identify the features that
impact the Engine shape (vs). This step will depend on the specific dataset and its characteristics.

# Once you have identified the relevant features, you can proceed to the next steps.

# (ii) Split the dataset into train and test datasets
set.seed(123)  # Set seed for reproducibility
train_indices <- sample(nrow(mtcars), nrow(mtcars) * 0.8)  # 80% random indices for training
train_data <- mtcars[train_indices, ]  # Training dataset
test_data <- mtcars[-train_indices, ]  # Test dataset

# (iii) Create logistic regression model
model <- glm(vs ~ mpg + cyl + hp, data = train_data, family = binomial)

# (iv) Make predictions on the test dataset
predictions <- predict(model, newdata = test_data, type = "response")

# (v) Create confusion matrix
```

```
threshold <- 0.5  # Threshold for classification
predicted_classes <- ifelse(predictions > threshold, 1, 0)  # Convert probabilities to classes
confusion_matrix <- table(predicted_classes, test_data$vs)
print(confusion_matrix)
```

Set-V

1.(i) Write a R program to extract the five of the levels of factor created from a random sample from the LETTERS (Part of the base R distribution.)

(ii)Write R function to find the range of given vector. Range=Max-Min

Sample input, C<-(9,8,7,6,5,4,3,2,1), output=8

(iii)Wirte the R function to find the number of vowels in given string

Sample input c<- "matrix", output<-2

```
# Set seed for reproducibility
set.seed(123)

# Create a random sample from LETTERS
sample_letters <- sample(LETTERS, size = 20, replace = TRUE)
# Convert the sample to a factor
factor_sample <- as.factor(sample_letters)
# Extract five levels from the factor
five_levels <- levels(factor_sample)[1:5]
(ii)
# Function to find the range of a vector
find_range <- function(vec) {
  range <- max(vec) - min(vec)
  return(range)
}
# Sample input
C <- c(9, 8, 7, 6, 5, 4, 3, 2, 1)
```

```
# Call the function with the vector
result <- find_range(C)
# Print the range
print(result)
```

(iii)

```
# Function to count the number of vowels in a string
count_vowels <- function(string) {
  vowels <- c("a", "e", "i", "o", "u")
  count <- sum(strsplit(tolower(string), "")[[1]] %in% vowels)
  return(count)
}

# Sample input
c <- "matrix"
# Call the function with the string
result <- count_vowels(c)
# Print the count
print(result)
```

2.Load inbuild dataset “ChickWeight” in R

(i) Explore the summary of Data set, like number of Features and its type. Finds the number of records for each features

(ii)Extract last 6 records of dataset

(iii) order the data frame, in ascending order by feature name “weight” grouped by feature “diet” (iv)Perform melting function based on “Chick","Time","Diet" features as ID variables

(v)Perform cast function to display the mean value of weight grouped by Diet

Code:

```
# Load the ChickWeight dataset
data(ChickWeight)

# (i) Explore the summary of the dataset
summary(ChickWeight)
str(ChickWeight)
table(ChickWeight$weight, ChickWeight$Time, ChickWeight$Chick, ChickWeight$Diet)
```

```
# (ii) Extract last 6 records of the dataset
last_6_records <- tail(ChickWeight, 6)
print(last_6_records)

# (iii) Order the data frame in ascending order by the "weight" feature, grouped by "diet"
ordered_data <- ChickWeight[order(ChickWeight$weight), ]
print(ordered_data)

# (iv) Perform melting function based on "Chick", "Time", "Diet" features as ID variables
library(reshape2)
melted_data <- melt(ChickWeight, id.vars = c("Chick", "Time", "Diet"))
print(melted_data)

# (v) Perform cast function to display the mean value of weight grouped by Diet
cast_data <- dcast(melted_data, Diet ~ ., mean)
print(cast_data)
```

3.(i)Get the Statistical Summary of "ChickWeight" dataset

(ii)Create Box plot for "weight" grouped by "Diet"

(iii)Create a Histogram for "Weight" features belong to Diet- 1 category

(iv) Create a Histogram for "Weight" features belong to Diet- 4 category

(v) Create Scatter plot for weight vs Time grouped by Diet

Code:

```
# Load the ChickWeight dataset
data(ChickWeight)

# (i) Get the statistical summary of the ChickWeight dataset
summary(ChickWeight)

# (ii) Create a box plot for "weight" grouped by "Diet"
library(ggplot2)
ggplot(ChickWeight, aes(x = factor(Diet), y = weight)) +
  geom_boxplot() +
  labs(x = "Diet", y = "Weight") +
  theme_minimal()
```

```
# (iii) Create a histogram for "weight" features belonging to Diet-1 category
hist(ChickWeight$weight[ChickWeight$Diet == 1], main = "Histogram of Weight (Diet 1)", xlab = "Weight")

# (iv) Create a histogram for "weight" features belonging to Diet-4 category
hist(ChickWeight$weight[ChickWeight$Diet == 4], main = "Histogram of Weight (Diet 4)", xlab = "Weight")

# (v) Create a scatter plot for weight vs Time grouped by Diet
ggplot(ChickWeight, aes(x = Time, y = weight, color = factor(Diet))) +
  geom_point() +
  labs(x = "Time", y = "Weight", color = "Diet") +
  theme_minimal()
```

4.(i) Create multi regression model to find a weight of the chicken , by “Time” and “Diet” as as predictor variables

(ii) Predict weight for Time=10 and Diet=1

(iii)Find the error(MAE) in model for same

Code:

```
# Load the ChickWeight dataset
data(ChickWeight)

# (i) Create a multiple regression model to predict weight using Time and Diet as predictors
model <- lm(weight ~ Time + Diet, data = ChickWeight)

# (ii) Predict weight for Time=10 and Diet=1
new_data <- data.frame(Time = 10, Diet = 1)
prediction <- predict(model, newdata = new_data)
print(prediction)

# (iii) Find the Mean Absolute Error (MAE) in the model for the prediction
actual_weight <- ChickWeight$weight[ChickWeight$Time == 10 & ChickWeight$Diet == 1]
mae <- mean(abs(prediction - actual_weight))
print(mae)
```