SAVEETHA SCHOOL OF ENGINEERING

ITA0459- STATISTICS WITH R-PROGRAMMING FOR STATISTICAL COMPUTING

NAME	DIVYA.S
REGISTER NUMBER	192124069

1.Write a R program to take input from the user (name and age) and display the values. Also print the version of R installation

```
name_1= readline(prompt = "Enter ur name")
age_1= as.integer(readline(prompt = "Enter ur age"))
print(paste("YOUR NAME is:",name_1))
print(paste("YOUR AGE IS:",age_1))
print(version)
```

2. Write a R program to get the details of the objects in memory.

```
my_list=c(1,2,3)
characters= "hello" #can use any datatype
memory_size= object.size(my_list)
print(memory_size)
```

3. Write a R program to create a sequence of numbers from 20 to 50 and find the mean of numbers from 20 to 60 and sum of numbers from 51 to 91.

```
sequence_of_numbers=20:50
print(sequence_of_numbers)
mean_of_number= mean(20:60)
print(paste("MEAN OF NUMBER FROM 20 TO 60=",mean_of_number))
sum_of_number=sum(51:91)
print(paste("THE SUM OF NUMBERS FROM 51 TO 91=",sum_of_number))
```

```
}
```

4. Write a R program to create a vector which contains 10 random integer values between -50 and +50.

```
random_integer=sample(-50:50,10, replace= FALSE,prob=NULL)
print("RANDOM INTEGER FROM -50 TO +50")
print(random_integer)
```

5. Write a R program to get the first 10 Fibonacci numbers.

```
Fibonacci = numeric(10)
print(Fibonacci)
Fibonacci[1] = Fibonacci[2] =1
for (i in 3:10) Fibonacci[i] = Fibonacci[i - 2] + Fibonacci[i - 1]
print("First 10 Fibonacci numbers:")
print(Fibonacci)
print(Fibonacci[0])
```

6. Write a R program to get all prime numbers up to a given number (based on the sieve of Eratosthenes)

```
get_primes=function(n) {
 primes= c()
 for (num in 2:n) {
  is_prime=TRUE
  for (divisor in 2:sqrt(num)) {
   if (num %% divisor == 0) {
    is_prime=FALSE
    break
   }
  }
```

```
    if (is_prime) {
      primes <- c(primes, num)
    }
  }
  return(primes)
}

# Example usage:
n <- 20
prime_numbers <- get_primes(n)
print(prime_numbers)
```

7. Write a R program to print the numbers from 1 to 100 and print "Fizz" for multiples of 3, print "Buzz" for multiples of 5, and print "FizzBuzz" for multiples of both.

```
for(n in 1:100){
  if(n %% 3 == 0 & n%%5==0){print("FizzBuzz")}
  else if(n%% 3==0){print("Fizz")}
  else if(n%% 5==0){print("Buzz")}
  else print(n)
}
```

8. Write a R program to extract first 10 english letter in lower case and last 10 letters in upper case and extract letters between 22nd to 24th letters in upper case

```
print("FIRST TEN ENGLISH LETTERS IN LOWERCASE")
first_ten_lc= head(letters,10)
```

```
print(first_ten_lc)
print("LAST TEN ENGLISH LETTERS IN UPPERCASE")
last_ten_uc= tail(LETTERS,10)
print(last_ten_uc)
print("UPPERCASE LETTERS FROM 22 TO 24")
uc_from_22to24=tail(LETTERS[22:24])
print(uc_from_22to24)
```

9. Write a R program to find the factors of a given number.

```
getinput=as.numeric(readline(prompt = "ENTER ANY NUMBER"))
print(paste("the factors of",getinput,"are="))
for(n in 1:getinput){
  if((getinput %% n)==0){
    print(n)
  }
}
```

10. Write a R program to find the maximum and the minimum value of a given vector.

```
given_vector=c(12,34,56,78,199,11)
maximum=max(given_vector)
print(paste("the maximum number is=",maximum))
minimum=min(given_vector)
print(paste("the minimum number is=",minimum))
```

11. Write a R program to get the unique elements of a given string and unique numbers of vector

```
unique_string="hello,my name is divya,bye,hello,bye"

unique_elements=unique(tolower(unique_string))

print(unique_elements)

unique_number=c(1,2,3,4,1,2,5,6,7)

unique_element=unique(unique_number)

print(unique_element)
```

12. Write a R program to create three vectors a,b,c with 3 integers. Combine the three vectors to become a 3×3 matrix where each column represents a vector. Print the content of the matrix

```
vector_a=c(1,2,3)

vector_b=c(4,5,6)

vector_c=c(7,8,9)

matrix_of_vectors=cbind(vector_a,vector_b,vector_c)

print("THE VECTORS INTO MATRIX")

print(matrix_of_vectors)
```

13. Write a R program to create a list of random numbers in normal distribution and count occurrences of each value.

```
print("LIST OF RANDOM NUMBERS IN ND")

random_number_nd=floor(rnorm(10,mean=20,sd=2)) #using 'floor' we can
remove decimal

print(random_number_nd)

occurance=table(random_number_nd)

print("OCCURANCES OF EACH VALUES")

print(occurance)
```

14. Write a R program to read the .csv file and display the content.

```
reading_csv=read.csv("C:/Users/divya/OneDrive/DB
PROJECT/job_skills.csv")

print(reading_csv)
```

15. Write a R program to create three vectors numeric data, character data and logical data. Display the content of the vectors and their type.

```
numeric_data=c(12,34,56)

character_data=c("hello","bye","see you")

logical_data=c(TRUE,FALSE,TRUE)

print(numeric_data)

print(typeof(numeric_data))

print(character_data)

print(typeof(character_data))

print(logical_data)

print(typeof(logical_data))
```

16. Write a R program to create a 5 x 4 matrix , 3 x 3 matrix with labels and fill the matrix by rows and 2 × 2 matrix with labels and fill the matrix by columns.

```
matrix_5into4=matrix(1:20,nrow = 5,ncol = 4)

print(matrix_5into4)

rowlabel=c("a","b","c")

collable=c("a","b","c")

matrix_3into3= matrix(1:9,nrow = 3, ncol = 3,byrow=TRUE,dimnames =
list(rowlabel,collable))

print(matrix_3into3)

rowlabel1=c("a","b")
```

```
collable2=c("a","b")

matrix_2into2= matrix(1:4,nrow = 2, ncol = 2,byrow=FALSE,dimnames =
list(rowlabel1,collable2))

print(matrix_2into2)
```

17. Write a R program to create an array, passing in a vector of values and a vector of dimensions. Also provide names for each dimension.

```
array_elements=c(1,2,3,4,5,6,7,8,9,10,11,12)

array_dimension=c(2,3,2)

array_names=list(c("R1","R2"),

            c("C1","C2","C3"),

            c("D1","D2"))

my_array=array(array_elements,dim = array_dimension,dimnames =
array_names)

print(my_array)
```

18. Write a R program to create an array with three columns, three rows, and two "tables", taking two vectors as input to the array. Print the array

```
vector_1=c(1,2,3,4,5,6,7,8,9)

 vector_2=c(10,11,12,13,14,15,16,17,18)

 combined_vectors=c(vector_1,vector_2)

 my_array=array(data = combined_vectors, dim = c(3,3,2),dimnames = NULL)

 print(my_array)
```

19. Write a R program to create a list of elements using vectors, matrices and a functions. Print the content of the list.

```
my_vector=c(1,2,3,4,5,6)
my_matrices=matrix(1:6,nrow=3,ncol=2)
my_function=function(x){
  return(x*2)
}
list_of_all=list(vector_elements=my_vector,
                 matrix_elements=my_matrices,
                 function_elements= my_function)
print(list_of_all)
```

20. Write a R program to draw an empty plot and an empty plot specify the axes limits of the graphic

```
empty_plot=plot(0,xlim=c(0,10),ylim=c(0,10),type="n",xlab="X
AXIS",ylab="Y AXIS",main="EMPTY PLOT")
#TYPE="n" IS USED TO CREATE A EMPTY PLOT
print(empty_plot)
```

21. Write a R program to create an array of two 3x3 matrices each with 3 rows and 3 columns from two given two vectors. Print the second row of the second matrix of the array and the element in the 3rd row and 3rd column of the 1st matrix.

```
vector1=c(1,2,3,4,5,6,7,8,9)
vector2=c(10,11,12,13,14,15,16,17,18)
matrix1=matrix(vector1,nrow=3,ncol=3)
matrix2=matrix(vector2,nrow=3,ncol=3)
```

```
matrix_array=array(c(matrix1,matrix2),dim=c(3,3,2),dimnames= NULL)
print(matrix_array)
print(matrix_array[2, , 2]) #first element 2 indicate "ROWS", second element " " indicates columns,third elements "2" indicated dimension of array
print(matrix_array[3,3,1])
```

22. Write a R program to combine three arrays so that the first row of the first array is followed by the first row of the second array and then first row of the third

```
num1 = rbind(rep("A",3), rep("B",3), rep("C",3))
print("num1")
print(num1)
num2 = rbind(rep("P",3), rep("Q",3), rep("R",3))
print("num2")
print(num2)
num3 = rbind(rep("X",3), rep("Y",3), rep("Z",3))
print("num3")
print(num3)
a = matrix(t(cbind(num1,num2,num3)),ncol=3, byrow=T)
print("Combine three arrays, taking one row from each one by one:")
print(a)
```

23. Write a R program to create an array using four given columns, three given rows, and two given tables and display the content of the array

```
print(array(1:30,dim = c(3,4,2)))
```

24. Write a R program to create a two-dimensional 5x3 array of sequence of even integers greater than 50.

```
a=array(seq(from=50,length.out=15,by=2),c(5,3))
print(a)
```

25. Create below data frame exam_data = data.frame(name = c('Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'), score = c(12.5, 9, 16.5, 12, 9, 20, 14.5, 13.5, 8, 19), attempts = c(1, 3, 2, 3, 2, 3, 1, 1, 2, 1), qualify = c('yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes'))

```
exam_data=data.frame(name=c('anastasia','dima','katherine','james','emily','mich
ael','matthew','laura','kevin','jonas'),
                score=c(12.5,9,16.5,12,9,20,14.5,13.5,8,19),
                attempts=c(1,3,2,3,2,3,1,1,2,1),
                qualify=c('yes','no','yes','no','no','yes','yes','no','no','yes'))
               print(exam_data)
```

a. **Write a R program to extract 3rd and 5th rows with 1st and 3rd columns from a given data frame**

```
selected_rows_columns=exam_data[c(3,5),c(1,3)]
print(selected_rows_columns)
```

b. **Write a R program to add a new column named country in a given data frame Country<-c("USA","USA","USA","USA","UK","USA","USA","India","USA","USA")**

```
country=c("USA","USA","USA","USA","UK","USA","USA","India","U
SA","USA")
exam_data$Country=country
print(exam_data)
```

c. **Write a R program to add new row(s) to an existing data frame new_exam_data = data.frame(name = c('Robert', 'Sophia'),score = c(10.5, 9), attempts = c(1, 3),qualify = c('yes', 'no'))**

```
exam_data_new=data.frame(name=c('robert','sophia'),
                score=c(10.5,9),
                attempts=c(1,3),
                qualify=c('yes','no'),
                Country=c("usa","eng")
)
exam_data_bind=rbind(exam_data,exam_data_new)
print(exam_data_bind)
```

d. Write a R program to sort a given data frame by name and score

```
    sorted_exam_data=exam_data[order(exam_data$name,exam_data$score)
    , ]
    print(sorted_exam_data)
```

e. Write a R program to save the information of a data frame in a file and display the information of the file.

```
write.csv(exam_data,"examdataLOE.csv")
read_exam=read.csv("examdataLOE.csv")
print(read_exam)
```

26. Write a R program to call the (built-in) dataset airquality. Check whether it is a data frame or not? Order the entire data frame by the first and second column. remove the variables 'Solar.R' and 'Wind' and display the data frame

```
air_data=datasets::airquality
print(air_data)
#order by 1 and 2 columns
ordering= airquality[order(airquality$Ozone,airquality$Solar.R),]
print(ordering)
#removing the variable
```

```
air_data[,c("Solar.R")]=NULL
print(air_data)
air_data[,c("Wind")]=NULL
print(air_data)
```

27. Write a R program to create a factor corresponding to height of women data set , which inbuild in R, contains height and weights for a sample of women

```
data=datasets::women
height_f=cut(data$height,3)
print(table(height_f))
```

28. 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.)

```
L=sample(LETTERS,size=50,replace=TRUE)
print(L)
print(factor(L))
print(table(L[1:5]))
```

29. Iris dataset is a very famous dataset in almost all data mining, machine learning courses, and it has been an R build-in dataset. The dataset consists of 50 samples from each of three species of Iris flowers (Iris setosa, Iris virginica and Iris versicolor). Four features(variables) were measured from each sample, they are the length and the width of sepal and petal, in centimetres. Perform the following EDA steps . (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: (vi) Average mean value of numerical varialbes by Species and Sepal.Length.Cate (vii)Create Pivot Table based on Species and Sepal.Length.Cate.

```
# Load the Iris dataset
data(iris)
# (i) Find dimension, Structure, Summary statistics, Standard Deviation of all features.
# Dimension of the dataset
dim(iris)
# Structure of the dataset
str(iris)
# Summary statistics of all features
summary(iris)
# Standard Deviation of all features
sapply(iris[, 1:4], sd)
# (ii) Find mean and standard deviation of features grouped by three species of Iris flowers.
# Mean of features grouped by species
aggregate(. ~ Species, data = iris, FUN = mean)
# Standard Deviation of features grouped by species
aggregate(. ~ Species, data = iris, FUN = sd)
# (iii) Find quantile value of sepal width and length
quantile(iris$Sepal.Length, probs = c(0.25, 0.5, 0.75))
quantile(iris$Sepal.Width, probs = c(0.25, 0.5, 0.75))
```

```
# (iv) Create a new data frame named iris1 with a new column
"Sepal.Length.Cate" that categorizes "Sepal.Length" by quantile

iris1 = iris

iris1$Sepal.Length.Cate <- cut(iris$Sepal.Length, breaks =
quantile(iris$Sepal.Length, probs = c(0, 0.25, 0.5, 0.75, 1)), labels = c("Q1",
"Q2", "Q3", "Q4"))

print(iris1)

# (v) Average value of numerical variables by two categorical variables: Species
and Sepal.Length.Cate

aggregate(. ~ Species + Sepal.Length.Cate, data = iris1, FUN = mean)

# (vi) Average mean value of numerical variables by Species and
Sepal.Length.Cate

aggregate(. ~ Species + Sepal.Length.Cate, data = iris1, FUN = mean,
FUN.aggregate = mean)

# Load the reshape2 package

library(reshape2)

# Melt the dataset

melted_data <- melt(iris1, id.vars = c("Species", "Sepal.Length.Cate"))

# Create the Pivot Table

pivot_table <- dcast(melted_data, Species + Sepal.Length.Cate ~ variable,
fun.aggregate = mean)

# Display the Pivot Table

print(pivot_table)
```

30. Randomly Sample the iris dataset such as 80% data for training and 20% for test and create Logistics regression with train data, use species as target and petals width and length as feature variables , Predict the probability of the model using test data, Create Confusion matrix for above test model

```
# Load the required library

library(caret)

# Load the Iris dataset

data("iris")

# Set a random seed for reproducibility

set.seed(123)

# Split the dataset into training and testing sets (80% training, 20% testing)

inTrain <- createDataPartition(iris$Species, p = 0.8, list = FALSE)

training_data <- iris[inTrain, ]

testing_data <- iris[-inTrain, ]

# Build a logistic regression model

model <- glm(Species ~ Petal.Length + Petal.Width, data = training_data,
family = binomial)

# Make predictions on the test data

predictions <- predict(model, newdata = testing_data, type = "response")

# Convert predicted probabilities to class labels

predicted_classes <- ifelse(predictions > 0.5, "versicolor", "not versicolor")

# Create a confusion matrix

confusion_matrix <- table(predicted = predicted_classes, actual =
testing_data$Species)

# Print the confusion matrix

print(confusion_matrix)
```

31. (i)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)

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

```
values=c(90,50,70,80,70,60,20,30,80,90,20)

print(paste("the mean",mean(values)))

print(paste("the median",median(values)))

values_mode=as.numeric(names(sort(table(values),decreasing=TRUE)[1]))

print(paste("the mode is",values_mode))

high=sort(values,decreasing = TRUE)

print(paste("the second hightest",max(high[2])))

high=sort(values,decreasing = FALSE)

print(paste("the second LOWEST",max(high[3])))
```

32. 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).

```
data=datasets::airquality

print(data)
```

i. Compute the mean temperature(don't use build in function)

```
mean_temp= sum(data$Temp)/length(data$Temp)

print(mean_temp)
```

ii.Extract the first five rows from airquality.

```
first_five_rows= head(data)
print(first_five_rows)
```

iii.Extract all columns from airquality except Temp and Wind.

```
selected_columns= airquality[, !(names(airquality) %in% c("Temp", "Wind"))]
print(selected_columns)
```

iv.Which was the coldest day during the period?

```
coldest_day =airquality[which.min(airquality$Temp), ]
print(coldest_day)
```

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

```
wind_gt_17_count <- sum(airquality$Wind > 17)
print(wind_gt_17_count)
```

33. (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?

```
library(reshape2)
data=datasets::airquality
summary_airquality=summary(data)
print(summary_airquality)
melted_data_air= melt(data)
print(melted_data_air)
melted_data_id= melt(airquality,id.vars = c("Month","Day"))
print(melted_data_id)
casted_data= dcast(melted_data_id,Month + Day ~ variable)
```

```
print(casted_data)

average_per_month=dcast(melted_data_id,Month~variable,fun.aggregate =
mean)

print(average_per_month)
```

34.(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

```
data=datasets::airquality

model=lm(Ozone~Solar.R,data=airquality)

print(model)

a=airquality$Ozone

b=airquality$Solar.R

plot(b,a,main = "SCATTER PLOT BETWEEN OZONE AND SOLAR.R",xlab
= "SOLAR.R",ylab =
"OZONE",col=c("blue","pink"),pch=19,abline(model,col="green"))

missing_percent= colMeans(is.na(data)) * 5

threshold = 10

cols_to_keep = names(missing_percent[missing_percent <= threshold])

data = data[complete.cases(data[cols_to_keep]), ]

data = replace(data, is.na(data), colMeans(data, na.rm = TRUE))

print(data)
```

35. Load dataset named ChickWeight, (i).Order the data frame, in ascending order by feature name “weight” grouped by feature “diet” and Extract the last 6 records from order data frame. (ii).a Perform melting

function based on “Chick", "Time", "Diet" features as ID variables b. Perform cast function to display the mean value of weight grouped by Diet c. Perform cast function to display the mode of weight grouped by Diet

```
library(dplyr)
library(reshape2)

# Load the ChickWeight dataset
data=datasets::ChickWeight
datadf=data.frame(data)
print(datadf)
# (i) Order the data frame, group by diet, and extract the last 6 records
order_datset=datadf[order(datadf$Diet)]
print(tail(order_datset))
# (ii) (a) Perform melting function
melted_data <- melt(ChickWeight, id.vars = c("Chick", "Time", "Diet"))
print(melted_data)
# (ii) (b) Perform cast function to display the mean value of weight grouped by
Diet
cast_mean <- dcast(melted_data, Diet ~ variable, mean)
print(cast_mean)
# (ii) (c) Perform cast function to display the mode of weight grouped by Diet
cast_mode <- dcast(melted_data, Diet ~ variable, mode)
print(cast_mode)
```

36. a. Create Box plot for “weight” grouped by “Diet” b. Create a Histogram for “weight” features belong to Diet- 1 category c. Create Scatter plot for “ weight” vs “Time” grouped by Diet

```
data=datasets::ChickWeight

boxplot(data$weight,data$Diet,main="BOX
PLOT",xlab="WEIGHT",ylab="DIET",col="pink")

diet_data= ChickWeight[ChickWeight$Diet == 1, ]

print(diet_data)

hist(diet_data$weight,main = "HIST OF WEIGHT AND DIET-1",col = "red")

plot(data$weight,data$Time,main = "SCATTER PLOT OF WEIGHT AND
TIME",xlab="WEIGHT",ylab =
"TIME",type="p",pch=19,col=c("pink","cyan"))
```

37. a. Create multi regression model to find a weight of the chicken , by “Time” and “Diet” as as predictor variables b. Predict weight for Time=10 and Diet=1 c. Find the error in model for same

```
data=datasets::ChickWeight

factor_data <- as.factor(ChickWeight$Diet)

multi_reg_model <- lm(weight ~ Time + Diet, data = ChickWeight)

print(multi_reg_model)

new_data <- data.frame(Time = 10, Diet = factor(1))  # Ensure 'Diet' is a factor

predicted_weight <- predict(multi_reg_model, newdata = new_data)

cat("Predicted Weight for Time=10 and Diet=1:", predicted_weight, "\n")

# Calculate Error

actual_weight <- ChickWeight$weight[ChickWeight$Time == 10 &
ChickWeight$Diet == 1]

error <- actual_weight - predicted_weight

cat("Error in Model for Same:", error, "\n")
```

38. .For this exercise, use the (built-in) dataset Titanic. a. Draw a Bar chart to show details of "Survived" on the Titanic based on passenger Class b. Modify the above plot based on gender of people who survived c. Draw histogram plot to show distribution of feature "Age"

```
library(ggplot2)

datatitanic=datasets::Titanic


# Convert the dataset to a data frame

titanic_df <- as.data.frame(datatitanic)

print(titanic_df)


# a. Draw a Bar chart to show details of "Survived" on the Titanic based on passenger Class

bar_chart_class <- ggplot(titanic_df, aes(x = Class, fill = factor(Survived))) +

  geom_bar(position = "dodge", stat = "count") +

  labs(title = "Survival on Titanic by Passenger Class",

       x = "Passenger Class",

       y = "Count",

       fill = "Survived")


print(bar_chart_class)


# b. Modify the above plot based on the gender of people who survived

bar_chart_gender <- ggplot(titanic_df, aes(x = Class, fill = factor(Survived), group = Sex)) +

  geom_bar(position = "dodge", stat = "count") +

  labs(title = "Survival on Titanic by Passenger Class and Gender",
       x = "Passenger Class",
       y = "Count",
       fill = "Survived") +
  facet_grid(.~Sex)

print(bar_chart_gender)

# c. Draw a histogram plot to show the distribution of the feature "Age"
histogram_age <- ggplot(titanic_df, aes(x = Age)) +
  geom_bar( fill = "blue", color = "black", alpha = 0.7) +
  labs(title = "Distribution of Age on Titanic",
       x = "Age",
       y = "Count")

print(histogram_age)
```

39. Explore the USArrests dataset, contains the number of arrests for murder, assault, and rape for each of the 50 states in 1973. It also contains the percentage of people in the state who live in an urban area. (i) a. Explore the summary of Data set, like number of Features and its type. Find the number of records for each feature. Print the statistical feature of data b. Print the state which saw the largest total number of rape c. Print the states with the max & min crime rates for murder (ii).a. Find the correlation among the features b. Print the states which have assault arrests more than median of the country c. Print the states are in the bottom 25% of murder (iii). a. Create a histogram and density plot of murder arrests by US stat b. Create the plot that shows the relationship between murder arrest rate and proportion of the population that is urbanised by state. Then enrich the chart by adding assault arrest rates (by colouring the points from blue (low) to red (high)). c. Draw a bar graph to show the murder rate for each of the 50 states .

```
usdataset=datasets::USArrests
create=as.data.frame(usdataset)
print(create)
#(i)a. Explore the summary of Data set, like number of Features and its type.
Find the number
#of records for each feature. Print the statistical feature of data
summarydata=str(usdataset)
print(summarydata)
statsitical=summary(usdataset)
print(statsitical)
#b. Print the state which saw the largest total number of rape
max_rape=usdataset[which.max(usdataset$Rape), ]
print(max_rape)
#c. Print the states with the max & min crime rates for murder
max_murder_state=usdataset[which.max(usdataset$Murder), ]
print(max_murder_state)
min_murder_state=usdataset[which.min(usdataset$Murder), ]
print(min_murder_state)
#(ii).a. Find the correlation among the features
correlation=cor(usdataset)
print(correlation)
#b. Print the states which have assault arrests more than median of the country
median_assault <- median(usdataset$Assault)
states_higher_than_median <- rownames(USArrests)[USArrests$Assault >
median_assault]
cat("States with assault arrests more than median:", states_higher_than_median,
"\n")
```

#c. Print the states are in the bottom 25% of murder

bottom_25_percent_murder <- rownames(USArrests)[USArrests$Murder < quantile(USArrests$Murder, 0.25)]

cat("States in the bottom 25% of murder:", bottom_25_percent_murder, "\n")

#a. Create a histogram and density plot of murder arrests by US state

hist(usdataset$Murder,main="murder arrests by US state",col="blue")

density_plot=density(usdataset$Murder)

plot(density_plot,main="density of murder arrest",xlab="density")

#b. Create the plot that shows the relationship between murder arrest rate and proportion

#of the population that is urbanised by state. Then enrich the chart by adding assault

#arrest rates (by colouring the points from blue (low) to red (high)).

# Plot relationship between murder arrest rate and urban population with color-coded assault rates

# Plot relationship between murder arrest rate and urban population with color-coded assault rates

plot(USArrests$UrbanPop, USArrests$Murder, col=USArrests$Assault, pch=16,main="Relationship between Murder Arrest Rate, Urban Population, and Assault Arrest Rate",xlab="Urban Population", ylab="Murder Arrest Rate")

#c. Draw a bar graph to show the murder rate for each of the 50 states .

barplot(USArrests$Murder, names.arg=rownames(USArrests), main="Murder Rate by State", xlab="State", ylab="Murder Rate",col="purple")

40. 4. a. Create a data frame based on below table.

Month 1 2 3 4 5 6 7 8 9 10 11 12

Spends 1000 4000 5000 4500 3000 4000 9000 11000 15000 12000 7000 3000 Sales 9914 40487 54324 50044 34719 42551 94871 118914 158484 131348 78504 36284

b. Create a regression model for that data frame table to show the amount of sales(Sales) based on the how much the company spends (Spends) in advertising c. Predict the Sales if Spend=1350

```
# Step a: Create a data frame
data <- data.frame(
  Month = 1:12,
  Spends = c(1000, 4000, 5000, 4500, 3000, 4000, 9000, 11000, 15000, 12000, 7000, 3000),
  Sales = c(9914, 40487, 54324, 50044, 34719, 42551, 94871, 118914, 158484, 131348, 78504, 36284)
)

# Step b: Create a regression model
model <- lm(Sales ~ Spends, data = data)

# Summary of the regression model
summary(model)

# Step c: Predict Sales if Spend = 13500
new_data <- data.frame(Spends = 13500)
predicted_sales <- predict(model, newdata = new_data)

# Print the predicted sales
cat("Predicted Sales for Spend = 13500:", predicted_sales, "\n")
```