INDEX

Sr. No	LIST OF EXPERIMENT	DATE	SIGNATURE
1	R AS CALCULATOR APPLICATION a. Using with and without R objects on console b. Using mathematical functions on console c. Write an R script, to create R objects for calculator application and save in a specified location in disk.		
2	DESCRIPTIVE STATISTICS IN R a. Write an R script to find basic descriptive statistics using summary, str, quartile function on mtcars & cars datasets. b. Write an R script to find subset of dataset by using subset (), aggregate () functions on iris dataset		
3	READING AND WRITING DIFFERENT TYPES OF DATASETS a. Reading different types of data sets (.txt, .csv) from Web and disk and writing in file in specific disk location. b. Reading Excel data sheet in R. c. Reading XML dataset in R		
4	VISUALIZATIONS a. Find the data distributions using box and scatter plot. b. Find the outliers using plot. c. Plot the histogram, bar chart and pie chart on sample data		
5	CORRELATION AND COVARIANCE a. Find the correlation matrix. b. Plot the correlation plot on dataset and visualize giving an overview of relationships among data on iris data. c. Analysis of covariance: variance (ANOVA), if data have categorical variables on iris data		
6	REGRESSION MODEL Import a data from web storage. Name the dataset and now do Logistic Regression to find out relation between variables that are affecting the admission of a student in a institute based on his or her GRE score, GPA obtained and rank of the student. Also check the model is fit or not. Require (foreign), require (MASS)		
7	MULTIPLE REGRESSION MODEL Apply multiple regressions, if data have a continuous Independent variable. Apply on above dataset.		
8	REGRESSION MODEL FOR PREDICTION Apply regression Model techniques to predict the data on above dataset.		
9	CLASSIFICATION MODEL a. Install relevant package for classification. b. Choose classifier for classification problem. c. Evaluate the performance of classifier.		
10	CLUSTERING MODEL a. Clustering algorithms for unsupervised classification. b. Plot the cluster data using R visualizations		

EXPERIMENT NO. 1

Aim: R AS CALCULATOR APPLICATION.

- **a.** Using with and without R objects on console
- **b.** Using mathematical functions on console
- **c.** Write an R script, to create R objects for calculator application and save in a specified location in disk.

Theory: R is a high-level programming language that is commonly used for data analysis and statistical computing. It is also very useful as a calculator due to its ability to handle mathematical operations with ease.

R AS CALCULATOR APPLICATION:

```
1.#Write an R script to create R objects for
calculator application add <- function(x, y) {
return(x + y)
}
subtract <- function(x, y) {
  return(x - y)
}
multiply <- function(x, y) {
return(x * y)
}
divide <- function(x, y) {
  return(x / y)
}
# take input from
the user
print("Select
operation.")
print("1.Add") print("2.Subtract")
print("3.Multiply") print("4.Divide") choice
= as.integer(readline(prompt="Enter
choice[1/2/3/4]: ")) num1 =
as.integer(readline(prompt="Enter first
number: ")) num2 =
as.integer(readline(prompt="Enter second
number: "))
operator <- switch(choice,"+","-","*","/")
result <- switch(choice, add(num1, num2), subtract(num1, num2), multiply(num1,
    num2), divide(num1, num2))
print(paste(num1,operator, num2,
"=", result))
```

OUTPUT:

```
[1] "Select operation."
[1] 1.Add"
[1] "2.Subtract"
[1] “3.Multiply"
[1] "4.Divide"
Enter choice[1/2/3/4]: 4
Enter first number: 20
Enter second number: 4
[1] "20 / 4 = 5"
```

EXPERIMENT NO. 2

AIM: DESCRIPTIVE STATISTICS IN R

a. Write an R script to find basic descriptive statistics using summary, str, quartile function on mtcars & cars datasets.

b. Write an R script to find subset of dataset by using subset (), aggregate () functions on iris dataset.

Dataset:

A **data set** (or **dataset**) is a collection of data that can be used by a to train machine learning models

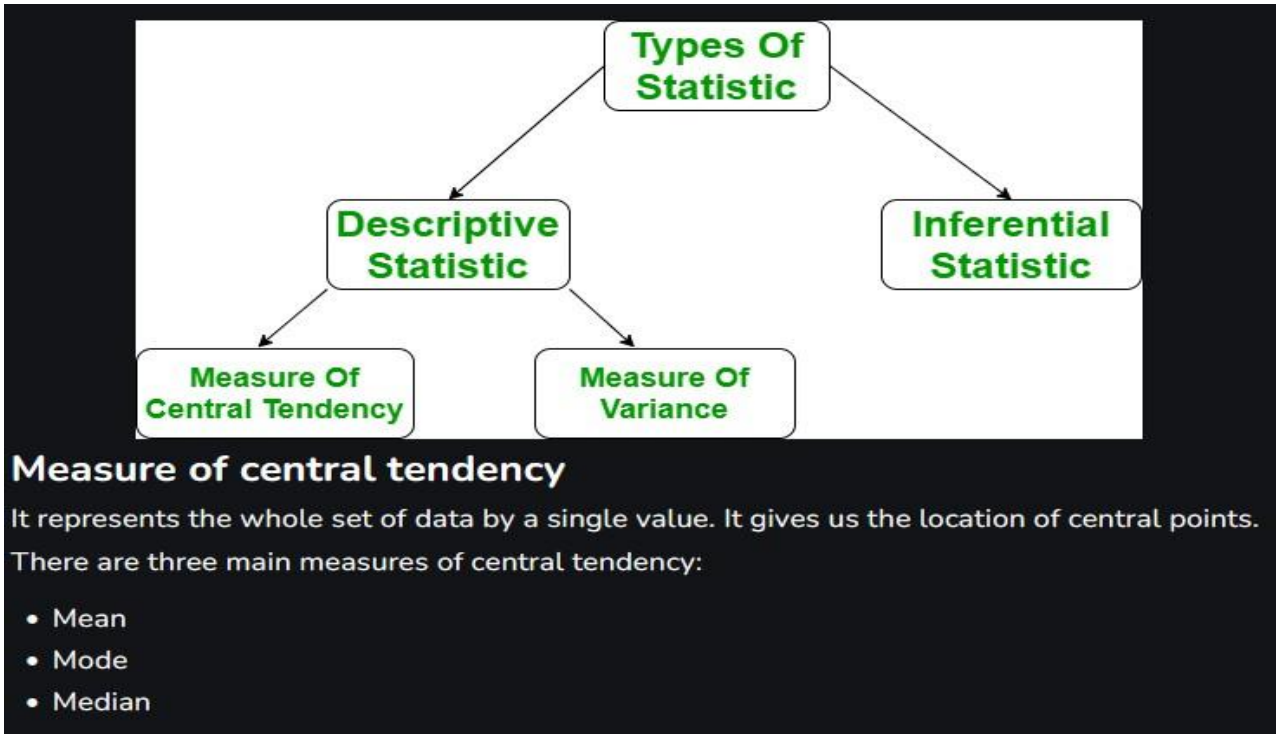
Taxes and Home Prices (1)
http://lib.stat.cmu.edu/DASL/Stories/hometax.html

House	Sale price (100$)	Size (sqft)	Age (years)
Avalon	2050	2650	13
Cross Winds	2080	2600	*
The White House	2150	2554	6
The Rectory	2150	2921	3
Larchwood	1999	2580	4
Orchard House	1900	2580	4
Shangri-La	1800	2774	2
The Stables	1560	1920	1
Cobweb Cottage	1450	2150	*
Nairn House	1449	1710	1

(2) (3) (4) (5)

Descriptive Statics:

In Descriptive statistics in R Programming Language, we describe our data with the help of various representative methods using charts, graphs, tables, excel files, etc. In the descriptive analysis, we describe our data in some manner and present it in a meaningful way so that it can be easily understood.



Import your data into R:

Before doing any computation, first of all, we need to prepare our data, save our data in external .txt or .csv files and it's a best practice to save the file in the current directory.

Note : Download mtcars dataset from kaggle and save in current working directory.(documents)

1)str()function The str() function takes a single object as an argument and compactly shows us the structure of the input object. It shows us details like length, data type, names and other specifics cs about the components of the object. str(mtcars)

Code:

 # Display internal structure of dataframe df <- read.csv("mtcars.csv")

str(df)

output:

2) Summary() Function

The summary() function in R takes a single object as an argument. It returns summary statistics like the mean, median, minimum, maximum, 1st quartile, 3rd quartile, etc., for each component or variable in the object.

df <- read.csv("mtcars.csv")

summary(df)

OUTPUT:

For example, here's a snippet of the output for the variables gear and carb:

Gear	carb
Min. :3.000	Min. :1.000
1st Qu.:3.000	1st Qu.:2.000
Median :4.000	Median :2.000
Mean :3.688	Mean :2.812
3rd Qu.:4.000	3rd Qu.:4.000
Max. :5.000	Max. :8.000

[For each of the 11 variables in the dataset, we can see the following summary statistics:]

- **Min**: The minimum value.
- **1st Qu**: The value of the first quartile (25th percentile).
- **Median**: The median value.
- **Mean**: The mean value.
- **3rd Qu**: The value of the third quartile (75th percentile).
- **Max**: The maximum value.

3) QUARTILE FUNCTIONS

Quartiles divide data into four equal parts:

- The **first quartile** (Q1) or **lower quartile** marks the point below which 25% of the data fall.
- The **second quartile** (Q2) is the median, dividing the data into two equal halves (50% below it).
- The **third quartile** (Q3) or **upper quartile** is the point below which 75% of the data fall.

CODE:-

```
# Load the built-in 'mtcars' dataset

data("mtcars")

# Display the top few rows of the data

head(mtcars)

# Install and load the dplyr package (only needs to be installed once)

install.packages('dplyr')
```

```
library(dplyr)
# Using tapply to apply the quantile function to multiple groups (by 'gear')
do.call("rbind", tapply(mtcars$mpg, mtcars$gear, quantile))
```

	0%	25%	50%	75%	100%
3	10.4	14.5	15.5	18.4	21.5
4	17.8	21.0	22.8	28.075	33.9
5	15.0	15.8	19.7	26.0	30.4

Exercise-2(B) Aim: Write an R script to find subsets of the Iris dataset using subset() and aggregate() functions.

SUBSETTING your data does not change the content of your data; it simply selects the portion most relevant to your analysis. In general, there are three main ways to subset the rows and columns of your dataset: by index, by name, and by value.

```
# Display the number of rows in the iris dataset
nrow(iris)
# Display row names
rownames(iris)
# Display column names
colnames(iris)
# Subset rows where species is 'setosa'
subset_setosa <- subset(iris, Species == "setosa")
print(subset_setosa)
# Display the first row
print(iris[1,])
# Display all rows in the second and fifth columns
print(iris[, c(2, 5)])
# Display the first row, for columns 2 to 5
print(iris[1, 2:5])
```

Additionally, here's how to use the aggregate() function on the Iris dataset to get summary statistics:

CODE

```
# Aggregate to find the mean of each numeric variable by species
aggregate_data <- aggregate(. ~ Species, data = iris, FUN = mean)
print(aggregate_data)
```

Syntax:

aggregate(x, by, FUN)

x: a variable to aggregate

by: A list of variables to group by

FUN: the summary statistic to compute

EXPERIMENT NO. 3

READING AND WRITING DIFFERENT TYPES OF DATASETS

a. Reading different types of data sets (.txt, .csv) from Web and disk and writing in file in specific disk location.

b. Reading Excel data sheet in R.

c. Reading XML dataset in R

3(a) Reading different types of data sets (.txt, .csv) from Web and disk and writing in file in specific disk location.

THEORY:

Reading and writing datasets is a fundamental aspect of data analysis and data science, and R programming provides various functions to handle different types of datasets. In R, a dataset can exist in multiple formats, such as CSV, Excel, text files, databases, and more. The goal is to read these datasets, perform data manipulation or analysis, and then save them back in the same or different format.

File reading in R

One of the important formats to store a file is in a text file. R provides various methods that one can read data from a text file.

- **read.delim()**: This method is used for reading "tab-separated value" files (".txt"). By default, point (".") is used as decimal points. ***Syntax:*** *read.delim(file, header = TRUE, sep = "\t", dec = ".", ...)* ***Parameters:***
- *file: the path to the file containing the data to be read into R.*
- *header: a logical value. If TRUE, read.delim() assumes that your file has a header row, so row 1 is the name of each column. If that's not the case, you can add the argument header = FALSE.*
- *sep: the field separator character. "\t" is used for a tab-delimited file. dec: the character used in the file for decimal points.*

Example:

R

```
# R program reading a text file

# Read a text file using read.delim()
myData = read.delim("geeksforgeeks.txt", header =
FALSE)
print(myData)
```

Output:

1 A computer science portal for geeks.

Note: The above R code, assumes that the file "geeksforgeeks.txt" is in your current working directory. To know your current working directory, type the function **getwd()** in R console.

- **read.delim2()**: This method is used for reading "tab-separated value" files (".txt"). By default, point (",") is used as decimal points. ***Syntax:*** *read.delim2(file, header = TRUE, sep = "\t", dec = ",", ...)* ***Parameters:***
- *file: the path to the file containing the data to be read into R.*
- *header: a logical value. If TRUE, read.delim2() assumes that your file has a header row, so row 1 is the name of each column. If that's not the case, you can add the argument header = FALSE.*
- *sep: the field separator character. "\t" is used for a tab-delimited file.*
- *dec: the character used in the file for decimal points.*

Example:

R

```
# R program reading a text file

# Read a text file using read.delim2
myData = read.delim2("geeksforgeeks.txt", header =
FALSE)
print(myData)
```

Output:

1 A computer science portal for geeks.

- **file.choose()**: In R it's also possible to choose a file interactively using the function **file.choose()**, and if you're a beginner in R programming then this method is very useful for you.

Example:

R

```
# R program reading a text file using file.choose()

myFile = read.delim(file.choose(), header = FALSE)
# If you use the code above in RStudio
# you will be asked to choose a file
print(myFile)
```

Output:

1 A computer science portal for geeks.

Reading a file in a table format

Another popular format to store a file is in a tabular format. R provides various methods that one can read data from a tabular formatted data file.

read.table(): read.table() is a general function that can be used to read a file in table format. The data will be imported as a data frame.

Syntax: *read.table(file, header = FALSE, sep = "", dec = ".")*

Parameters:

- *file: the path to the file containing the data to be imported into R.*
- *header: logical value. If TRUE, read.table() assumes that your file has a header row, so row 1 is the name of each column. If that's not the case, you can add the argument header = FALSE.*
- *sep: the field separator character*
- *dec: the character used in the file for decimal points.*

Example:

R

```
# R program to read a file in table format

# Using read.table() myData = read.table("basic.csv") print(myData)
```

Output:

1 Name,Age,Qualification,Address

2 Amiya,18,MCA,BBS

3 Niru,23,Msc,BLS

4 Debi,23,BCA,SBP5 Biku,56,ISC,JJP **read.csv()**: read.csv() is used for reading "comma separated value" files (".csv"). In this also the data will be imported as a data frame.

Syntax: *read.csv(file, header = TRUE, sep = ",", dec = ".", ...)*

Parameters:

- *file: the path to the file containing the data to be imported into R.*
- *header: logical value. If TRUE, read.csv() assumes that your file has a header row, so row 1 is the name of each column. If that's not the case, you can add the argument header = FALSE.*
- *sep: the field separator character*
- *dec: the character used in the file for decimal points.*

Example:

R

```
# R program to read a file in table format

# Using read.csv() myData = read.csv("basic.csv") print(myData)
```

Output:

Name Age Qualification Address

1 Amiya 18 MCA BBS

2 Niru 23 Msc BLS

3 Debi 23 BCA SBP4 Biku 56 ISC JJP **read.csv2()**: read.csv() is used for variant used in countries that use a comma “,” as decimal point and a semicolon “;” as field separators.

Syntax: *read.csv2(file, header = TRUE, sep = “;”, dec = “,”, ...)*
Parameters:

- *file: the path to the file containing the data to be imported into R.*
- *header: logical value. If TRUE, read.csv2() assumes that your file has a header row, so row 1 is the name of each column. If that’s not the case, you can add the argument header = FALSE.*
- *sep: the field separator character*
- *dec: the character used in the file for decimal points.*

Example:

R

```
# R program to read a file in table format

# Using read.csv2() myData = read.csv2("basic.csv") print(myData)
```

Output:
Name.Age.Qualification.Address

1 Amiya,18,MCA,BBS

2 Niru,23,Msc,BLS

3 Debi,23,BCA,SBP4 Biku,56,ISC,JJP **file.choose()**: You can also use **file.choose()** with **read.csv()** just like before. **Example:**

R

```
# R program to read a file in table format

# Using file.choose() inside read.csv() myData = read.csv(file.choose()) # If you use the code above in RStudio # you will be asked to choose a file print(myData)
```

Output:
Name Age Qualification Address

1 Amiya 18 MCA BBS

2 Niru 23 Msc BLS

3 Debi 23 BCA SBP4 Biku 56 ISC JJP **read_csv()**: This method is also used for to read a comma (“,”) separated values by using the help of readr package.

Syntax: *read_csv(file, col_names = TRUE)*
Parameters:

- *file: the path to the file containing the data to be read into R.*
- *col_names: Either TRUE, FALSE, or a character vector specifying column names. If TRUE, the first row of the input will be used as the column names.*

Example:

R

```
# R program to read a file in table format
# using readr package

# Import the readr library
library(readr)

# Using read_csv() method
myData = read_csv("basic.csv", col_names = TRUE)
print(myData)
```

Output:

Parsed with column specification:

```
cols(
  Name = col_character(),
  Age = col_double(),
  Qualification = col_character(),
  Address = col_character()
)
# A tibble: 4 x 4
  Name    Age Qualification Address

1 Amiya    18 MCA           BBS
2 Niru     23 Msc           BLS
3 Debi     23 BCA           SBP
4 Biku     56 ISC           JJP
```

Reading a file from the internet

It's possible to use the functions **read.delim()**, **read.csv()** and **read.table()** to import files from the web.

Example:

R

```
# R program to read a file from the internet

# Using read.delim() myData =
read.delim("http://www.sthda.com/upload/boxplot_for mat.txt")
print(head(myData))
```

Output:

Nom variable Group

1 IND1 10 A

2 IND2 7 A

3 IND3 20 A

4 IND4 14 A

5 IND5 14 A6 IND6 12 A

b) Reading Excel data sheet in R. #Install openxlsx package

```
install.packages("openxlsx
")


# Load openxlsx
library(openxlsx) # Read
excel file
read.xlsx('/Users/admin/new_f
ile.xlsx')
```

EXPERIMENT NO. 4

Aim: VISUALIZATIONS

a. Find the data distributions using box and scatter plot.

b. Find the outliers using plot.

c. Plot the histogram, bar chart and pie chart on sample data.

Theory: Visualizations in R programming refer to the graphical representations of data using different plots and charts. R programming offers a wide range of tools for creating informative and appealing visualizations, including scatter plots, bar graphs, histograms, box plots, and more.

Creating visualizations in R programming typically involves the following steps:

1. Importing data into R using the read.csv or read.table functions

2. Transforming and cleaning data as needed

3. Selecting the appropriate type of plot for the data and using the corresponding R function to generate the plot

4. Customizing the plot with labels, titles, colors, and other features as desired

5. Saving the visualization to a file or displaying it directly in the R console Overall, visualizations in R programming are an essential tool for exploring and communicating complex data sets and can be used in a variety of settings, from research and academia to business and industry.

a. Find the data distributions using box and scatter plot.

1. Box Plot: A box plot is a graphical representation of the distribution of a continuous variable through their quartiles. It can also indicate the presence of outliers. In R, the boxplot() function is used to create a box plot.

Syntax: boxplot(x, y, data, notch, varwidth, main, xlab, ylab, col)

- x: a numeric vector of values to be plotted on the x-axis
- y: a numeric vector of values to be plotted on the y-axis
- data: a data frame containing the variables to be plotted
- notch: a logical value indicating whether notches should be drawn around the median
- varwidth: a logical value indicating whether the width of the box should be proportional to the square root of the sample size
- main: the title of the plot
- xlab: the label for the x-axis
- ylab: the label for the y-axis
- col: the color of the boxes and whiskers

2. Scatter Plot: A scatter plot is a graphical representation of the relationship between two continuous variables. It displays the values of two variables as points on a two-dimensional graph, where the position of each point is determined by its x and y values. In R, the plot() function is used to create a scatter plot.

Syntax:

plot(x, y, data, main, xlab, ylab, col)

- x: a numeric vector of values to be plotted on the x-axis
- y: a numeric vector of values to be plotted on the y-axis
- data: a data frame containing the variables to be plotted
- main: the title of the plot
- xlab: the label for the x-axis
- ylab: the label for the y-axis
- col: the color of the points

b. Find the outliers using plot.

Outliers, as the name suggests, are the data points that lie away from the other points of the dataset. That is the data values that appear away from other data values and hence disturb the overall distribution of the dataset. This is usually assumed as an abnormal distribution of the data values.

c. Plot the histogram, bar chart and pie chart on sample data.

1. Histogram: A histogram is a graph that displays the distribution of a continuous variable. It is created by dividing the range of values into a series of intervals (bins) and counting the number of observations that fall into each bin. In R, the hist() function is used to create a histogram.

Syntax:

hist(x, breaks, col, xlab, ylab, main)

- x: a numeric vector of values to be plotted
- breaks: the number of intervals (bins) to use in the histogram
- col: the fill color of the bars
- xlab: the label for the x-axis
- ylab: the label for the y-axis
- main: the title of the plot

2. Bar Chart: A bar chart is a graph that displays the distribution of a categorical variable. It is created by plotting the frequency or proportion of each category as a bar. In R, the barplot() function is used to create a bar chart.

Syntax:

barplot(height, names.arg, col, xlab, ylab, main)

- height: a numeric vector of values representing the heights of the bars

- names.arg: a character vector of labels for each bar
- xlab: the label for the x-axis
- ylab: the label for the y-axis
- main: the title of the plot

3. Pie Chart: A pie chart is a graph that displays the relative proportions of different categories as slices of a circle. In R, the pie() function is used to create a pie chart.

Syntax: pie(x, labels, col, main)

- x: a numeric vector of values representing the sizes of each slice
- labels: a character vector of labels for each slice
- col: a vector of colors for each slice
- main: the title of the plot

Conclusion: In this practical, we learned how to use R programming to create different types of data visualizations. We used box plots and scatter plots to identify data distributions and outliers. We also created a histogram, bar chart, and pie chart to visualize sample data. Data visualization is an essential tool to explore and understand data, and R provides many functions and packages to create different types of plots and visualizations.

EXPERIMENT NO. 5

CORRELATION AND COVARIANCE

a. Find the correlation matrix.

b. Plot the correlation plot on dataset and visualize giving an overview of relationships among data on iris data.

c. Analysis of covariance: variance (ANOVA), if data have categorical variables on iris data

To calculate the correlation matrix for the numeric variables in the iris dataset:

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

# Calculate the correlation matrix for numeric variables cor_matrix <- cor(iris[, 1:4])  # The
first 4 columns are numeric (Sepal.Length, Sepal.Width, Petal.Length, Petal.Width)

# Print the correlation matrix

print(cor_matrix) OUTPUT:
```

```
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
```

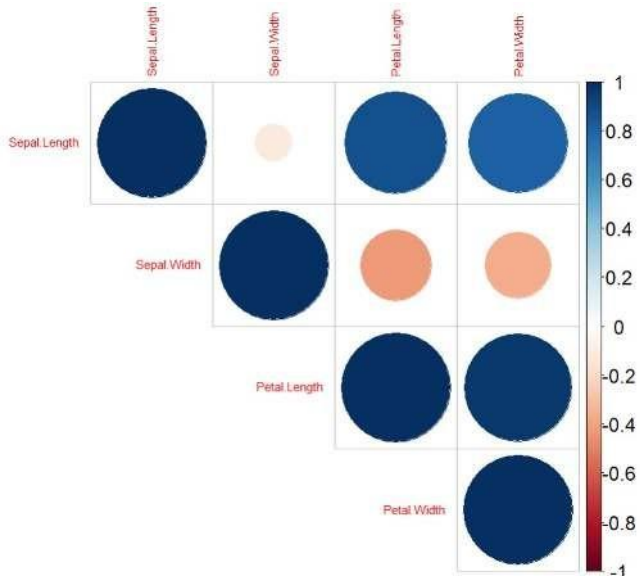
a) Plot the Correlation Plot

To visualize the correlation matrix, you can use the corrplot package, which provides a nice graphical representation:

```
# Install and load the corrplot package if not already installed
install.packages("corrplot") library(corrplot)

# Plot the correlation matrix corrplot(cor_matrix, method = "circle",
type = "upper", tl.cex = 0.5)
```

OUTPUT:



a) Analysis of Covariance (ANCOVA) with Categorical Variables

In the context of the iris dataset, you might want to analyze how the mean of a numeric variable varies across different species (categorical variable) using ANOVA, which is a special case of ANOVA when there is only one categorical variable.

Here's how to perform ANOVA on one of the numeric variables, say Sepal.Length, by species: #

Perform ANOVA to see if there's a significant difference in Sepal.Length across Species

anova_result <- aov(Sepal.Length ~ Species, data = iris)

Summary of the ANOVA summary(anova_result)

```
> summary(anova_result)
             Df Sum Sq Mean Sq F value Pr(>F)
Species       2  63.21  31.606   119.3 <2e-16 ***
Residuals   147  38.96   0.265
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Explanation of Each Column

- **Df: Degrees of Freedom**
  - Species: The number of categories (species) minus one, which is 2 (3 species - 1).
  - Residuals: The degrees of freedom for the residuals (total observations - number of groups), which is 147 in this case.
- **Sum Sq: Sum of Squares**
  - Species: The variation in Sepal.Length due to differences between species, which is 63.21.

    - Residuals: The variation not explained by species (within-group variation), which is 38.96.
- **Mean Sq: Mean Square**
    - Species: Average variation explained by species, calculated as Sum Sq / Df, which is 31.606.
    - Residuals: Average within-group variation, which is 0.265.
- **F value:** The ratio of the variance explained by the model (Species) to the variance not explained (Residuals). It tells you how much of the variation in Sepal.Length is explained by species compared to the residual variation. In this case, it is 119.3.
- **Pr(>F):** p-value
    - This value tests the null hypothesis that the means of Sepal.Length are equal across species. A very small p-value (in this case, <2e-16) indicates that there is strong evidence against the null hypothesis, suggesting that at least one species mean is significantly different from the others. **Interpretation**
- **F value:** A large F value (119.3) indicates that the variation between species is much larger compared to the variation within species, suggesting that the species have different means for Sepal.Length.
- **p-value:** The p-value is less than 0.001 (indicated by ***), which is much lower than the common significance level of 0.05. This means that the differences in Sepal.Length between species are statistically significant.

  Conclusion: There are significant differences in Sepal.Length between different species of the iris dataset. The species explain a substantial portion of the variability in Sepal.Length, far beyond what would be expected by chance alone.

EXPERIMENT NO. 6

Aim: REGRESSION MODEL Import a data from web storage. Name the dataset and now do Logistic Regression to find out relation between variables that are affecting the admission of a student in a institute based on his or her GRE score, GPA obtained and rank of the student. Also check the model is fit or not. Require (foreign), require (MASS)

>library (rio)
>data<-import("binary.sas7bdat")

Data Cleaning:

Looking at the structure of data set

>str(data)

```
## 'data.frame':    400 obs. of  4 variables:
##  $ ADMIT: num  0 1 1 1 0 1 1 0 1 0 ...
##  $ GRE  : num  380 660 800 640 520 760 560 400 540 700 ...
##  $ GPA  : num  3.61 3.67 4 3.19 2.93 ...
##  $ RANK : num  3 3 1 4 4 2 1 2 3 2 ...
##  - attr(*, "label")= chr "LOGIT"
```

Variables **ADMIT** and **RANK** are of type numeric but they should be factor variables since were are not going to perform any mathematical operations on them.

```
>data$ADMIT<-as.factor(data$ADMIT)
>data$RANK<- as.factor(data$RANK)
>str (data)
# 'data.frame':    400 obs. of  4 variables:
##  $ ADMIT: Factor w/ 2 levels "0","1": 1 2 2 2 1 2 2 1 2 1 ...
##  $ GRE  : num  380 660 800 640 520 760 560 400 540 700 ...
##  $ GPA  : num  3.61 3.67 4 3.19 2.93 ...
##  $ RANK : Factor w/ 4 levels "1","2","3","4": 3 3 1 4 4 2 1 2 3 2 ...
##  - attr(*, "label")= chr "LOGIT"
```

Looking at the summary of the dataset

```
>summary (data
##  ADMIT       GRE             GPA       RANK
##  0:273   Min.   :220.0   Min.   :2.260   1: 61
##  1:127   1st Qu.:520.0   1st Qu.:3.130   2:151
##          Median :580.0   Median :3.395   3:121
##          Mean   :587.7   Mean   :3.390   4: 67
##          3rd Qu.:660.0   3rd Qu.:3.670
##          Max.   :800.0   Max.   :4.000
```

From the summary statistics we observe

- Most of students did not get admitted
- There are no missing data values(NAs).

Checking for multi collineality

>Plot(data$GPA,data$GRE,col="red")

> cor (data$GRE, data$GPA)

Exploratoty Data Analysis.

We will explore the relationship between dependent and independent variables by way of visualization.

GRE

Since GRE is numeric variable and dependent variable is factor variable, we plot a box plot

Library(ggplot2)
Ggplot(data,aes(ADMIT,GRE,fill=ADMIT))+

```
library (ggplot2)   #For plotting
ggplot(data,aes(ADMIT,GRE,fill=ADMIT))+
  geom_boxplot()+
  theme_bw()+
  xlab("Admit")+
  ylab("GRE" )+
  ggtitle  ( "ADMIT BY GRE")
```

The two box plots are differents in terms of displacement, and hence *GRE* is significant variable.

GPA

```
ggplot(data,aes(ADMIT,GPA,fill=ADMIT))+
  geom_boxplot()+
  theme_bw()+
  xlab("Admit")+
  ylab("GPA")+
  ggtitle("ADMIT BY GPA" )
```

There is clear difference in displacement between the two box plots, hence *GPA* is an important predictor.

RANK

RANK is a factor variable and since the dependent variable is a factor variable we plot a bar plot.

```
ggplot(data,aes(RANK,ADMIT,fill=ADMIT))+
  geom_col()+
  xlab("RANK")+
  ylab(  "COUNT-      )+
  ggtitle( "ADMIT BY )
```

Modelling

Data Splitting

Before we fit a model, we need to split the dataset into training and test dataset to be able to assess the performance of the model with the unseen test.

EXPERIMENT NO. 7

Aim : Apply multiple regressions, if data have a continuous Independent variable. Apply on above dataset. tidyverse for data manipulation and visualization.

>library(tidverse)

We'll use the marketing data set [datarium package], which contains the impact of the amount of money spent on three advertising medias (youtube, facebook and newspaper) on sales. First install the datarium package using devtools::install_github("kassmbara/datarium"), then load and inspect the marketing data as follow:

>data("marketing", package = "datarium")
> head(marketing, 4)

```
##   youtube facebook newspaper sales
## 1   276.1     45.4      83.0  26.5


## 2    53.4     47.2      54.1  12.5
## 3    20.6     55.1      83.2  11.2
## 4   181.8     49.6      70.2  22.2
```

Building model

We want to build a model for estimating sales based on the advertising budget invested in youtube, facebook and newspaper, as follow:

sales = b0 + b1*youtube + b2*facebook + b3*newspaper You can compute the model coefficients in R as follow:

>model <- lm(sales ~ youtube + facebook + newspaper, data = marketing)

>summary(model)

```
####Call:
## lm(formula = sales ~ youtube + facebook + newspaper, data = marketing)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -10.59  -1.07   0.29   1.43   3.40
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.52667    0.37429    9.42   <2e-16 ***
## youtube      0.04576    0.00139   32.81   <2e-16 ***
## facebook     0.18853    0.00861   21.89   <2e-16 ***
## newspaper   -0.00104    0.00587   -0.18     0.86
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01
'*' 0.05 '.' 0.1 ' ' 1 ##
## Residual standard error: 2.02 on 196 degrees of freedom
## Multiple R-squared:  0.897,  Adjusted R-squared:  0.896
## F-statistic:  570 on 3 and 196 DF,  p-value: <2e-16
```

Interpretation

The first step in interpreting the multiple regression analysis is to examine the F-statistic and the associated pvalue, at the bottom of model summary.

In our example, it can be seen that p-value of the F-statistic is < 2.2e-16, which is highly significant. This means that, at least, one of the predictor variables is significantly related to the outcome variable.

To see which predictor variables are significant, you can examine the coefficients table, which shows the estimate of regression beta coefficients and the associated t-statitic p-values:

```
>summary(model)$coefficient
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.52667    0.37429   9.422 1.27e-17
## youtube      0.04576    0.00139  32.809 1.51e-81
## facebook     0.18853    0.00861  21.893 1.51e-54
## newspaper   -0.00104    0.00587  -0.177 8.60e-01
```

As the newspaper variable is not significant, it is possible to remove it from the model:

```
>model  <- lm(sales ~ youtube + facebook, data = marketing)
>summary(model)
##
## Call:
## lm(formula = sales ~ youtube + facebook, data = marketing)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -10.557  -1.050   0.291   1.405   3.399
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.50532    0.35339    9.92   <2e-16 ***
## youtube      0.04575    0.00139   32.91   <2e-16 ***
## facebook     0.18799    0.00804
23.38   <2e-16 *** ## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.02 on 197 degrees of freedom
## Multiple R-squared:  0.897,  Adjusted R-squared:  0.896
## F-statistic:  860 on 2 and 197 DF,  p-value: <2e-16
```

Finally, our model equation can be written as follow: sales = 3.5 + 0.045*youtube + 0.187*facebook. The confidence interval of the model coefficient can be extracted as follow:

```
>confint(model)
```

```
##              2.5 % 97.5 %
##
(Intercept)
2.808
4.2022 ##
youtube
0.043
0.0485
## facebook    0.172 0.2038
```

Model accuracy assessment

Residual Standard Error (RSE), or sigma:
The RSE estimate gives a measure of error of prediction. The lower the RSE, the more accurate the model (on the data in hand).

The error rate can be estimated by dividing the RSE by the mean outcome variable:

```
>sigma(model)/mean(marketing$sales)
```

```
## [1] 0.12
```

EXPERIMENT NO. 8

REGRESSION MODEL FOR PREDICTION:

Aim: Apply regression Model techniques to predict the data on above dataset.

Predicting Blood pressure using Age by Regression in R

Now we are taking a dataset of Blood pressure and Age and with the help of the data train a linear regression model in R which will be able to predict blood pressure at ages that are not present in our dataset. Equation of the regression line in our dataset

BP = 98.7147 + 0.9709 Age

Importing dataset

Importing a dataset of Age vs Blood Pressure which is a CSV file using function read.csv( ) in R and storing this dataset into a data frame bp.

>bp <- read.csv("bp.csv")

Creating data frame for predicting values

Creating a data frame which will store Age 53. And this data frame will be used to predict blood pressure at Age
53 after creating a linear regression model.

>p <- as.data.frame(53)

>colnames(p) <- "Age"

Calculating the correlation between Age and Blood pressure

We can also verify our above analysis that there is a correlation between Blood pressure and Age by taking the help of cor( ) function in R which is used to calculate the correlation between two variables. **>cor(bpBP,bpAge)**

[1] 0.6575673

Creating a Linear regression model

Now with the help of lm( ) function, we are going to make a linear model. lm( ) function has two attributes first is a formula where we will use "BP ~ Age" because Age is an independent variable and Blood pressure is a dependent variable and the second is data, where we will give the name of the data frame containing data which is in this case, is data frame bp. model <- lm(BP ~ Age, data = bp)

Summary of our linear regression

model summary(model)

Call:
lm(formula = BP
~ Age, data = bp) #
Residuals:
Min 1Q Median 3Q Max
-21.724 -6.994 -0.520 2.931 75.654
##
Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept) 98.7147 10.0005 9.871 1.28e-10 ***
Age 0.9709 0.2102 4.618 7.87e-05 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
Residual standard error: 17.31 on 28 degrees of freedom
Multiple R-squared: 0.4324, Adjusted R-squared: 0.4121
F-statistic: 21.33 on 1 and 28 DF, p-value: 7.867e-05

Interpretation of the model

Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept) 98.7147 10.0005 9.871 1.28e-10 ***
Age 0.9709 0.2102 4.618 7.87e-05 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
B0 = 98.7147 (Y- intercept)
B1 = 0.9709 (Age coefficient)
BP = 98.7147 + 0.9709 Age

EXPERIMENT NO. 9

Aim: 9a. Install relevant package for classification.
9b. Choose classifier for classification problem
9c. Evaluate the performance of classifier

The R package "party" is used to create decision trees.

Install R Package

Use the below command in R console to install the package. You also have to install the dependent packages if any.

>install.packages("party")

The package "party" has the function **ctree()** which is used to create and analyze decison tree.

Syntax

The basic syntax for creating a decision tree in R is −

>ctree(formula, data)
INPUT DATA:
>library(party)
>print(head(reading skills))

When we execute the above code, it produces the following result and chart −

	nativeSpeaker	**age**	**shoeSize**	**score**
1	yes	5	24.83189	32.29385
2	yes	6	25.95238	36.63105
3	no	11	30.42170	49.60593
4	yes	7	28.66450	40.28456
5	yes	11	31.88207	55.46085
6	yes	10	30.07843	52.83124

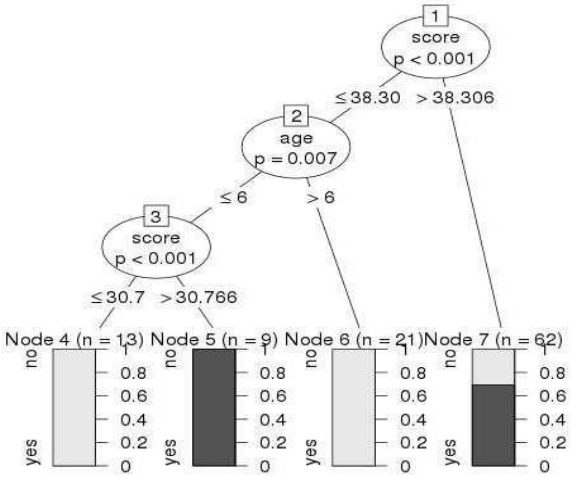
Loading required package: methods

```
# dependent packages.
library(party)

# Create the input data frame.
input.dat <- readingSkills[c(1:105),]

# Give the chart file a name.
png(file = "decision_tree.png")

# Create the tree.
```



Loading required package: grid
.....

Example

We will use the **ctree()** function to create the decision tree and see its graph.

```
# Load the party package. It will automatically load other
  output.tree <- ctree(
  nativeSpeaker ~ age + shoeSize + score,
  data = input.dat)

# Plot the tree.
plot(output.tree)

# Save the file.
dev.off()
```

When we execute the above code, it produces the

following result − null device

1

EXPERIMENT NO. 10

Aim: 10a. Clustering algorithms for unsupervised classification
10b. Plot the cluster data

k-means clustering

```
str(x)
##  num [1:300, 1:2] 3.37 1.44 2.36 2.63 2.4 ...
head(x)
##          [,1]     [,2]
## [1,] 3.370958 1.995379
## [2,] 1.435302 2.760242
## [3,] 2.363128 2.038991
## [4,] 2.632863 2.735072
## [5,] 2.404268 1.853527
## [6,] 1.893875 1.942113
# Create the k-means model: km.out
km.out <- kmeans(x, centers = 3, nstart = 20)

# Inspect the result
summary(km.out)
##              Length Class  Mode
## cluster      300    -none- numeric
```

using R visualizations

```
## centers        6    -none- numeric
## totss          1    -none- numeric
## withinss       3    -none- numeric
## tot.withinss   1    -none- numeric
## betweenss      1    -none- numeric
## size           3    -none- numeric
## iter           1    -none- numeric
## ifault         1    -none- numeric
```

– Results of kmeans()

```
# Print the cluster membership component of the model
km.out$cluster
##   [1] 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [36] 2 3 3 3 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2
##  [71] 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1
## [106] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [141] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [176] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [211] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [246] 1 1 1 1 1 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 3 3 2
## [281] 3 3 3 3 3 3 2 3 3 3 3 3 3 2 3 3 3 2 3 3
# Print the km.out object
km.out
## K-means clustering with 3 clusters of sizes 150, 98, 52
##
## Cluster means:
##         [,1]        [,2]
## 1 -5.0556758  1.96991743
## 2  2.2171113  2.05110690
## 3  0.6642455 -0.09132968
##
## Clustering vector:
##   [1] 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [36] 2 3 3 3 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2
##  [71] 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1
## [106] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [141] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [176] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
## [211] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [246] 1 1 1 1 1 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 3 3 2
## [281] 3 3 3 3 3 3 2 3 3 3 3 3 3 2 3 3 3 2 3 3
##
## Within cluster sum of squares by cluster:
## [1] 295.16925 148.64781  95.50625
##  (between_SS / total_SS =  87.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

– Visualizing and interpreting results of kmeans()

```
# Scatter plot of x
plot(x,
  col = km.out$cluster,
  main = "k-means with 3 clusters",
  xlab = "",
  ylab = "")
```