Practical No. 2: Measures of Central Tendency & Dispersion

❖ Mean

In R, the mean of a vector is calculated using the mean() function. The function accepts a vector as input, and returns the average as a numeric.

✓ Calculate the Mean Scores

```
> # Creating a hypothetical dataset
> scores <- c(78, 82, 72, 91, 69, 75, 88, 80, 73, 79, 84, 70, 77, 81, 68,
90, 76, 85, 74, 87)
> # Calculating the mean
> mean_score <- mean(scores)
> mean_score

[1] 78.95
```

✓ Generate some Random Age vector between 18 to 60 years and calculate the average age.

Sample() takes a sample of the specified size from the elements of x using either with or without replacement.

```
> # Generating a vector of random ages
> set.seed(456)  # Setting seed for reproducibility
> ages <- sample(18:60, 12, replace = TRUE)  # Generating 30 random ages
                                               between 18 and 60
> # Displaying the vector of ages
> ages

 [1] 54 52 55 38 44 42 31 48 26 32 43 36

> # Calculating the mean age
> mean_age <- mean(ages)
> mean_age

[1] 41.75
```

✓ Import the CSV File of CardioGoodFitness Dataset and Calculate the Average Age.

```
> # Import the data using read.csv()
> data = read.csv("CardioGoodFitness.csv", stringsAsFactors=F)
> # Compute the mean value
> mean = mean(data$Age)
> print(mean)

[1] 28.78889
```

❖ Median

The median() Function in R, accepts a vector as an input. If there are an odd number of values in the vector, the function returns the middle value. If there are an even number of values in the vector, the function returns the average of the two medians

Syntax :

The basic syntax for calculating median in R is :

median(x, na.rm = FALSE)

Following is the description of the parameters used:

- ***x is the input vector.***
- ***na.rm is used to remove the missing values from the input vector.***

```
> # Creating a hypothetical dataset
> scores <- c(78, 82, 72, 91, 69, 75, 88, 80, 73, 79, 84, 70, 77, 81, 68,
90, 76, 85, 74, 87)
> # Calculating the Median
> median_score <- median(scores)
> median_score

[1] 78.5
```

✓ **Calculate the Median Age of CardioGoodFitness Dataset**

```
> # Import the data using read.csv()
> cardio_data = read.csv("CardioGoodFitness.csv",
+                  stringsAsFactors=F)
> head(cardio_data, 5)

  Product Age Gender Education MaritalStatus Usage Fitness Income Miles
1   TM195  18   Male        14        Single     3       4  29562   112
2   TM195  19   Male        15        Single     2       3  31836    75
3   TM195  19 Female        14     Partnered     4       3  30699    66
4   TM195  19   Male        12        Single     3       3  32973    85
5   TM195  20   Male        13     Partnered     4       2  35247    47

> # Compute the median value
> median_age = median(cardio_data$Age)
> print(median_age)

[1] 26
```

If there are ***missing values****, then the mean function returns NA. To drop the missing values from the calculation, use* ***na.rm = TRUE****. which* ***means remove the NA values****.*

```
> x <- c(12,7,3,4.2,18,2,54,-21,8,-5,NA)
> # Find mean.
> result.median <-  median(x)
> print(result.median)
[1] NA
> # Find mean dropping NA values.
> result.median <-  median(x,na.rm = TRUE)
> print(result.median)
[1] 5.6
```

✓ **Create vector marks of certain subject and Compute Mode**

Note: There is No In-Built Function for Mode. Create Custom Function for Mode.

```
> marks <- c(97, 78, 57,78, 97, 66, 87, 64, 87, 78)
> # Function to compute mode
> compute_mode <- function(x) {
+   table_x <- table(x)
+   mode_value <- as.numeric(names(table_x[table_x == max(table_x)]))
+   return(mode_value)
+ }
> # Computing mode of the vector
> mode_result <- compute_mode(marks)
> # Displaying the result
> cat("The mode of the vector marks is:", mode_result, "\n")
The mode of the vector marks is: 78
```

✓ **Create Data Frame and Compute the Mode using Custom Function.**

```
Create a Data frame and Compute the mode
> # Create Data Frame
> df=data.frame(id=c(11,22,33,44,55),
+              name=c("spark","python","R","jsp","R"),
+              price=c(144,NA,144,567,567))
> df
  id   name price
1 11  spark   144
2 22 python    NA
3 33      R   144
4 44    jsp   567
5 55      R   567
> # Usage of mode() Function
> mode(df$id)
[1] 11
> mode(df$name)
[1] "R"
> mode(df$price)
[1] 144
```

❖ Computing Range

✓ Method 1: Find range in a vector using min and max functions

Syntax:

max(vector)-min(vector)

If a vector contains NA values then we should use the na.rm function to exclude NA values

```
> # create vector
> data = c(12, 45, NA, NA, 67, 23, 45, 78, NA, 89)
>
> # display
> print(data)
 [1] 12 45 NA NA 67 23 45 78 NA 89
> # find range
> print(max(data, na.rm=TRUE)-min(data, na.rm=TRUE))

[1] 77
```

✓ Method 2: Using range() function

```
> # create dataframe
> data = data.frame(column1=c(12, 45, NA, NA, 67, 23, 45, 78, NA, 89),
+                   column2=c(34, 41, NA, NA, 27, 23, 55, 78, NA, 73))
> # display
> head(data, 3)
  column1 column2
1      12      34
2      45      41
3      NA      NA

> # find range in entire dataframe
> print(range(data$column1, na.rm = T))
[1] 12 89
```

❖ Standard Deviation

This function computes the standard deviation of the values in x. If na.rm is TRUE then missing values are removed before computation proceeds.

Syntax:

sd(x, na.rm = FALSE)

✓ **A garden contains 39 plants. The following plants were chosen at random, and their heights were recorded in cm: 38, 51, 46, 79, and 57. Calculate their heights' standard deviation.**

```
> # Heights of the plants
> heights <- c(38, 51, 46, 79, 57)
> heights
[1] 38 51 46 79 57
> # Calculate standard deviation
> std_dev <- sd(heights)
> # Print the result
> cat("Standard Deviation:", std_dev, "\n")

Standard Deviation: 15.51451
```

✓ **Data Frame Outcome represent the total of the numbers obtained by rolling two fair dice. Calculate standard deviation.**

```
> # Generate all possible outcomes of rolling two dice
> outcomes <- data.frame(Die1 = rep(1:6, each = 6), Die2 = rep(1:6, times
= 6))
> head(outcomes)
  Die1 Die2
1    1    1
2    1    2
3    1    3
4    1    4
5    1    5
6    1    6

> # Calculate the sum of the two dice for each outcome
> outcomes$Sum <- outcomes$Die1 + outcomes$Die2
> # Calculate variance and standard deviation
> std_dev_X <- sd(outcomes$Sum)
> cat("Standard Deviation of X:", std_dev_X, "\n")

Standard Deviation of X: 2.44949
```

❖ Variance

The var() function in R Language computes the sample variance of a vector. It is the measure of how much value is away from the mean value.

Syntax: ***var(x)***

Parameters:

x : numeric vector

✓ **Computing variance of a vector**

```
> # Create example vector
> x <- c(1, 2, 3, 4, 5, 6, 7)
> print(x)

[1] 1 2 3 4 5 6 7
> # Apply var function in R
> var(x)
[1] 4.666667
```

✓ **Find population variance for the following: 14, 9, 21, 15, 8**

```
> # Data
> data <- c(14, 9, 21, 15, 8)
> # Calculate population variance
> population_variance <- var(data)
> # Print the result
> cat("Population Variance:", population_variance, "\n")

Population Variance: 27.3
```

✓ **Find variance when heights of 5 dogs are given as follows:**

600mm	100mm	700mm	350mm	400mm

```
> # Heights of 5 dogs
> heights <- c(600, 100, 700, 350, 400)
> # Calculate variance
> variance <- var(heights)
> # Print the result
> cat("Variance of Heights:", variance, "\n")

Variance of Heights: 54500
```

Practical No. 3: Mathematical operations & Functions

❖ Operators in R

✓ Arithmetic operators

R supports the following arithmetic operators:

- *Addition, +, which returns the sum of two numbers.*
- *Subtraction, -, which returns the difference between two numbers.*
- *Multiplication, *, which returns the product of two numbers.*
- *Division, /, which returns the quotient of two numbers.*
- *Exponents, ^, which returns the value of one number raised to the power of another.*
- *Modulus, %%, which returns the remainder of one number divided by another.*
- *Integer Division, %/%, which returns the integer quotient of two numbers.*

```
> # Addition
> a <- 5
> b <- 3
> result_addition <- a + b
> print(paste("Addition:", result_addition))
[1] "Addition: 8"
> ## Combined Arthematic Operations
> result_combined <- a + b - a * b / a
> print(paste("Combined Operations:", result_combined))
[1] "Combined Operations: 5"
> # Exponentiation
> c <- 2
> d <- 3
> result_exponentiation <- c ^ d
> print(paste("Exponentiation:", result_exponentiation))
[1] "Exponentiation: 8"
> # Modulus
> result_modulus <- a %% b
> print(paste("Modulus:", result_modulus))
[1] "Modulus: 2"
> # Integer division
> result_integer_division <- a %/% b
> print(paste("Integer Division:", result_integer_division))
[1] "Integer Division: 1"
```

✓ Comparison operators

R has the following comparison operators:

- Equal, ==, which returns TRUE if two values are equal.
- Not equal, !=, which returns TRUE if two values are not equal.
- Less than, <, which returns TRUE if left value is less than right value.
- Less than or equal to, <=, which returns TRUE if left value is less than or equal to right value.
- Greater than, >, which returns TRUE if left value is greater than right value.
- Greater than or equal to, >=, which returns TRUE if left value is greater than or equal to right value.

```
> a <- 5
> b <- 3
> result_equal <- a == b
> print(paste("Equal:", result_equal))
[1] "Equal: FALSE"
> result_not_equal <- a != b
> print(paste("Not Equal:", result_not_equal))
[1] "Not Equal: TRUE"
> result_less_than <- a < b
> print(paste("Less Than:", result_less_than))
[1] "Less Than: FALSE"
> result_greater_than_equal <- a >= b
> print(paste("Greater Than or Equal To:", result_greater_than_equal))
[1] "Greater Than or Equal To: TRUE"
```

✓ Logical operators

R has the following logical operators:

- Element-wise AND, &, for comparing each element and returning TRUE if both elements are TRUE.
- Element-wise OR, |, for comparing each element and returning TRUE if either element is TRUE.
- Logical NOT, !, which returns TRUE if the associated statement is FALSE.

```
> a <- TRUE
> b <- FALSE
> # AND
> result_and <- a & b
> print(paste("AND:", result_and))
[1] "AND: FALSE"
> # OR
> result_or <- a | b
> print(paste("OR:", result_or))
[1] "OR: TRUE"
> # NOT
> result_not <- !a
> print(paste("NOT:", result_not))
[1] "NOT: FALSE"
```

❖ Mathematical Functions in R

In R, in-built functions are pre-defined tools for various tasks (e.g., sum(), mean()), while user-defined functions are personally crafted by programmers to address specific needs using the function keyword.

✓ In-Built Functions

```
> # Sum of a vector
> numbers <- c(1, 2, 3, 4, 5)
> sum_result <- sum(numbers)
> print(sum_result)
[1] 15
> # Mean of a vector
> mean_result <- mean(numbers)
> print(mean_result)
[1] 3
> length_result <- length(numbers)
> print(length_result)
[1] 5
> median_result <- median(numbers)
> print(median_result)
[1] 3
```

✓ User Defined Functions

The basic syntax to create a user-defined function in R is as follows:

```
function_name <- function(arg1, arg2, ...) {
  # Function body with operations
  result <- some_operation(arg1, arg2, ...)
  return(result)
}
```

Example:

```
> # Function to square a number
> square <- function(x) {
+   return(x^2)
+ }
> # Call the square function
> num <- c(1:5)
> print(num)
[1] 1 2 3 4 5
> square(num)
[1]  1  4  9 16 25
```

Practical No. 4: Matrix & Data Frame operations & functions

❖ Matrix

A matrix is a two-dimensional data set with columns and rows. A column is a vertical representation of data, while a row is a horizontal representation of data. A matrix can be created with the matrix() function. Specify the nrow and ncol parameters to get the number of rows and columns:

Syntax : *The basic syntax for creating a matrix in R :*

matrix(data, nrow, ncol, byrow, dimnames)

✓ **Create a matrix taking a vector of numbers as input**

```
> # Elements are arranged sequentially by row.
> M <- matrix(c(3:14), nrow = 4, byrow = TRUE)
> print(M)

     [,1] [,2] [,3]
[1,]    3    4    5
[2,]    6    7    8
[3,]    9   10   11
[4,]   12   13   14

> # Elements are arranged sequentially by column.
> N <- matrix(c(3:14), nrow = 4, byrow = FALSE)
> print(N)

     [,1] [,2] [,3]
[1,]    3    7   11
[2,]    4    8   12
[3,]    5    9   13
[4,]    6   10   14

> # Define the column and row names.
> rownames = c("row1", "row2", "row3", "row4")
> colnames = c("col1", "col2", "col3")
> P <- matrix(c(3:14), nrow = 4, byrow = TRUE, dimnames = list(rownames, 
colnames))
> print(P)

     col1 col2 col3
row1    3    4    5
row2    6    7    8
row3    9   10   11
row4   12   13   14
```

Note: Remember the c() function is used to concatenate items together.

✓ Accessing Elements of a Matrix

Elements of a matrix can be accessed by using the column and row index of the element. We consider the matrix P above to find the specific elements below.

```
> P <- matrix(c(3:14), nrow = 4)
> print(P)
     [,1] [,2] [,3]
[1,]    3    7   11
[2,]    4    8   12
[3,]    5    9   13
[4,]    6   10   14
> # Access the element at 3rd column and 1st row.
> print(P[1,3])
[1] 11
> # Access the element at 2nd column and 4th row.
> print(P[4,2])
[1] 10
> # Access only the  2nd row.
> print(P[2,])
[1]  4  8 12
> # Access only the 3rd column.
> print(P[,3])
[1] 11 12 13 14
```

✓ Matrix Arithmetic operations

```
> matrix1 <- matrix(c(3, 9, -1, 4, 2, 6), nrow = 2)
> print(matrix1)
     [,1] [,2] [,3]
[1,]    3   -1    2
[2,]    9    4    6
>
> matrix2 <- matrix(c(5, 2, 0, 9, 3, 4), nrow = 2)
> print(matrix2)
     [,1] [,2] [,3]
[1,]    5    0    3
[2,]    2    9    4
>
> # Add the matrices.
> result <- matrix1 + matrix2
> cat("Result of addition","\n")
Result of addition
> print(result)
     [,1] [,2] [,3]
[1,]    8   -1    5
[2,]   11   13   10
>
> # Subtract the matrices
> result <- matrix1 - matrix2
> cat("Result of subtraction","\n")
Result of subtraction
> print(result)
     [,1] [,2] [,3]
[1,]   -2   -1   -1
[2,]    7   -5    2

> result <- matrix1 * matrix2
> cat("Result of multiplication","\n")
Result of multiplication
> print(result)
     [,1] [,2] [,3]
[1,]   15    0    6
[2,]   18   36   24
```

✓ Matrix Functions

Number of Rows and Columns :
Use the dim() function to find the number of rows and columns in a Matrix:

```
> thismatrix <- matrix(c("apple", "banana", "cherry", "orange"), nrow = 2,
ncol = 2)
> dim(thismatrix)
[1] 2 2
```

Matrix Length :
Use the length() function to find the dimension of a Matrix:

```
> length(thismatrix)
[1] 4
```

✓ Combine two Matrices

Again, you can use the rbind() or cbind() function to combine two or more matrices together:

Example :

```
> Matrix1 <- matrix(c(1:4), nrow = 2, ncol = 2, byrow = T)
> Matrix2 <- matrix(c(5:8), nrow = 2, ncol = 2, byrow = T)
> print(Matrix1)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
> print(Matrix2)
     [,1] [,2]
[1,]    5    6
[2,]    7    8
> # Adding it as a rows
> rbind(Matrix1, Matrix2)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
[4,]    7    8
> # Adding it as a columns
> cbind(Matrix1, Matrix2)
     [,1] [,2] [,3] [,4]
[1,]    1    2    5    6
[2,]    3    4    7    8
```

❖ Data Frames

Data Frames are data displayed in a format as a table.

Use the data.frame() function to create a data frame:

```
> Data_Frame <- data.frame (
+   Training = c("Strength", "Stamina", "Other"),
+   Pulse = c(100, 150, 120),
+   Duration = c(60, 30, 45)
+ )
> # Print the data frame
> Data_Frame
  Training Pulse Duration
1 Strength   100       60
2  Stamina   150       30
3    Other   120       45
```

✓ ***Use the summary() function to summarize the data from a Data Frame:***

```
> summary(Data_Frame)

   Training              Pulse          Duration
 Length:3           Min.   :100.0   Min.   :30.0
 Class :character   1st Qu.:110.0   1st Qu.:37.5
 Mode  :character   Median :120.0   Median :45.0
                    Mean   :123.3   Mean   :45.0
                    3rd Qu.:135.0   3rd Qu.:52.5
                    Max.   :150.0   Max.   :60.0
```

✓ **Access Items**

We can use single brackets [ ], double brackets [[ ]] or $ to access columns from a data frame:

```
> Data_Frame[1]

  Training
1 Strength
2  Stamina
3    Other

> Data_Frame[["Training"]]

[1] "Strength" "Stamina"  "Other"

> Data_Frame$Training

[1] "Strength" "Stamina"  "Other"
```

✓ **Add Rows**

Use the rbind() function to add new rows in a Data Frame:

```
> New_row_DF <- rbind(Data_Frame, c("Strength", 110, 110))
> # Print the new row
> print(New_row_DF)

  Training Pulse Duration
1 Strength   100       60
2  Stamina   150       30
3    Other   120       45
4 Strength   110      110
```

✓ **Add Columns**

Use the cbind() function to add new columns in a Data Frame:

```
> # Add a new column Method 1
> New_col_DF <- cbind(Data_Frame, Steps = c(1000, 6000, 2000))
> # Print the new column
> print(New_col_DF)

  Training Pulse Duration Steps
1 Strength   100       60  1000
2  Stamina   150       30  6000
3    Other   120       45  2000
```

```
> # Add a new column Method 2
> New_col_DF$Target <- c("Yes", "No", "Yes")
> # Print the new column
> print(New_col_DF)

  Training Pulse Duration Steps Target
1 Strength   100       60  1000    Yes
2  Stamina   150       30  6000     No
3    Other   120       45  2000    Yes
```

✓ **Remove Rows and Columns**

Use the c() function to remove rows and columns in a Data Frame:

```
> # Remove the first row and column
> Data_Frame_New <- New_col_DF[-c(1), -c(1)]
> # Print the new data frame
> Data_Frame_New

  Pulse Duration Steps Target
2   150       30  6000     No
3   120       45  2000    Yes
```

Use the length() function to find the number of columns in a Data Frame (similar to ncol()):

```
> length(Data_Frame_New)

[1] 4
```

Use the dim() function to find the amount of rows and columns in a Data Frame:

```
> dim(Data_Frame_New)

[1] 2 4
```

You can also use the ncol() function to find the number of columns and nrow() to find the number of rows:

```
> ncol(Data_Frame_New)
[1] 4

> nrow(Data_Frame_New)
[1] 2
```

✓ **Converting Matrix into Data Frame**

```
> # Your original matrix
> thismatrix <- matrix(c("apple", "banana", "cherry", "orange", "mango",
"pineapple"), nrow = 3, ncol = 2)
> # Convert the matrix to a data frame
> df <- as.data.frame(thismatrix)
> # Print the resulting data frame
> print(df)
      V1        V2
1  apple    orange
2 banana     mango
3 cherry pineapple
```

```
> # Provide custom column names
> colnames(df) <- c("Fruit1", "Fruit2")
> # Print the data frame with custom column names
> print(df)
  Fruit1    Fruit2
1  apple    orange
2 banana     mango
3 cherry pineapple
```

- ✓ *create a data frame with two columns containing certain numbers and then add a third column that contains Boolean values based on whether the value in the first column is greater than the value in the second column.*

```
> # Generate data for the first two columns
> col1 <- c(5, 8, 12, 3, 6)
> col2 <- c(2, 7, 9, 4, 5)
> # Create the data frame
> df <- data.frame(Column1 = col1, Column2 = col2)
> # Add a third column with Boolean values
> df$GreaterColumn1 <- df$Column1 > df$Column2
> # Print the resulting data frame
> print(df)

  Column1 Column2 GreaterColumn1
1       5       2           TRUE
2       8       7           TRUE
3      12       9           TRUE
4       3       4          FALSE
5       6       5           TRUE
```

Practical No. 5 Correlation and regression functions

- **Correlation coefficient can be computed using the functions cor() or cor.test(): cor() computes the correlation coefficient**
- **Generally, it lies between -1 and +1. It is a scaled version of covariance and provides the direction and strength of a relationship.**
- **To determine if the correlation coefficient between two variables is statistically significant, you can perform a correlation test in R using the following syntax:**

Syntax:

cor(x, y, method = "pearson")

cor.test(x, y, method = "pearson")

Parameters:

x, y: numeric vectors with the same length

method: correlation method

✓ Example 1: Using cor() method

```
> x = c(1, 2, 3, 4, 5, 6, 7) # vector 1
> y = c(1, 3, 6, 2, 7, 4, 5) # Vector 2
> # Calculating Correlation coefficient Using cor() method
> result = cor(x, y, method = "pearson")
> # Print the result
> cat("Pearson correlation coefficient is:", result)

Pearson correlation coefficient is: 0.5357143
```

✓ Example 2: Using cor.test() method

```
> x = c(1, 2, 3, 4, 5, 6, 7)
> y = c(1, 3, 6, 2, 7, 4, 5)
> # Calculating Correlation coefficient Using cor.test() method
> result = cor.test(x, y, method = "pearson")
> # Print the result
> print(result)

        Pearson's product-moment correlation

data:  x and y
t = 1.4186, df = 5, p-value = 0.2152
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.3643187  0.9183058
sample estimates:
      cor
0.5357143
```

✓ From the following data, compute Karl Pearson's coefficient of correlation

Price(Rs.)	10	20	30	40	50	60	70
Supply (Units)	8	6	14	16	10	20	24

```
> price = c(10,20,30,40,50,60,70)
> supply = c(8,6,14,16,10,20,24)
> correlation = cor.test(price, supply, method='pearson')
> correlation

        Pearson's product-moment correlation

data:  price and supply
t = 3.6145, df = 5, p-value = 0.01531
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.2707625 0.9774828
sample estimates:
      cor
0.8504201
```

❖ Regression

- ***The lm() function creates a linear regression model in R. This function takes an R formula Y ~ X where Y is the outcome variable and X is the predictor variable.***
- ***To create a multiple linear regression model in R, add additional predictor variables using +.***

✓ perform a simple linear regression on numeric vector

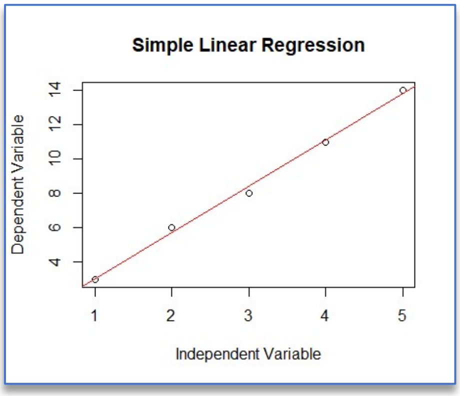
```
> # Create a numeric vector
> independent_variable <- c(1, 2, 3, 4, 5)
> dependent_variable <- c(3, 6, 8, 11, 14)
> # Create a data frame
> data <- data.frame(IndependentVar = independent_variable, DependentVar =
dependent_variable)
> # Fit a simple linear regression model
> model <- lm(DependentVar ~ IndependentVar, data = data)
> print(model)

Call:
lm(formula = DependentVar ~ IndependentVar, data = data)

Coefficients:
   (Intercept)  IndependentVar
           0.3             2.7
```

✓ Visualize the above linear regression using plot function

```
> # Plot the data points and regression line
> plot(independent_variable, dependent_variable, main = "Simple Linear
Regression",
+      xlab = "Independent Variable", ylab = "Dependent Variable")
> abline(model, col = "red")
```



✓ Predict values using the model

```
> new_data <- data.frame(IndependentVar = c(6, 7, 8))
> predicted_values <- predict(model, newdata = new_data)
> print(predicted_values)

   1    2    3
16.5 19.2 21.9
```

✓ Print the summary of the regression model

```
> summary(model)

Call:
lm(formula = DependentVar ~ IndependentVar, data = data)

Residuals:
         1          2          3          4          5
 8.882e-16  3.000e-01 -4.000e-01 -1.000e-01  2.000e-01

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)      0.3000     0.3317   0.905 0.432389
IndependentVar   2.7000     0.1000  27.000 0.000111 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3162 on 3 degrees of freedom
Multiple R-squared:  0.9959,   Adjusted R-squared:  0.9945
F-statistic:   729 on 1 and 3 DF,  p-value: 0.0001115
```

Practical No. 6 Probability & Conditional probability functions

Probability is a fundamental concept in statistics that measures the likelihood of events occurring. It ranges from 0 (impossible event) to 1 (certain event). Conditional probability is the probability of an event occurring given that another event has already occurred. In R, you can use various functions to calculate probabilities and explore the relationships between events.

- ✓ **In a deck of cards, what is the probability of drawing a red card?**

```
> # Probability of drawing a red card
> total_cards <- 52
> red_cards <- 26
> probability_red <- red_cards / total_cards
> probability_red
[1] 0.5
```

- ✓ **A six-sided die is rolled. What is the probability of rolling an even number?**

```
> # Probability of rolling an even number
> total_outcomes <- 6
> even_outcomes <- 3  # Numbers 2, 4, 6 are even
> probability_even <- even_outcomes / total_outcomes
> probability_even
[1] 0.5
```

- ✓ **In a group of students, 60% play basketball, and 40% play soccer. If 30% play both sports, what is the probability that a student chosen at random plays at least one of the sports?**

```
> # Probability of playing at least one sport
> probability_basketball <- 0.6
> probability_soccer <- 0.4
> probability_both <- 0.3
>
> # Probability of playing at least one sport = P(Basketball) + P(Soccer)
- P(Both)
> probability_at_least_one <- probability_basketball + probability_soccer
- probability_both
> probability_at_least_one

[1] 0.7
```

- ✓ **Suppose you have information about students' study habits and exam performance. The data is stored in a data frame named student_data. Use R to find the conditional probability of passing the exam given that a student has low attendance.**

```
> # Data frame with student information
> student_data <- data.frame(
+   Attendance = c("High", "High", "Low", "Low"),
+   Pass = c("Yes", "No", "Yes", "No"),
+   Frequency = c(80, 20, 30, 70)
+ )
> print(student_data)

  Attendance Pass Frequency
1       High  Yes        80
2       High   No        20
3        Low  Yes        30
4        Low   No        70

> # Total frequency of students with low attendance
> total_low_attendance <-
sum(student_data$Frequency[student_data$Attendance == "Low"])
> # Frequency of students who pass the exam with low attendance
> pass_and_low_attendance <-
student_data$Frequency[student_data$Attendance == "Low" &
student_data$Pass == "Yes"]
> # Conditional probability of passing the exam given low attendance
> P_pass_given_low_attendance <- pass_and_low_attendance /
total_low_attendance
> P_pass_given_low_attendance

[1] 0.3
```

- ✓ **Consider a survey where people were asked about their preferences for two brands, A and B. The data is stored in a data frame named brand_survey. Calculate the conditional probability that a person prefers brand B given that they are aged 25 or older.**

```
> # Data frame with brand survey information
> brand_survey <- data.frame(
+   Age = c("18-24", "25-34", "18-24", "25-34"),
+   Preference = c("A", "B", "A", "B"),
+   Frequency = c(30, 40, 20, 50)
+ )
> print(brand_survey)

    Age Preference Frequency
1 18-24          A        30
2 25-34          B        40
3 18-24          A        20
4 25-34          B        50
> # Total frequency of people aged 25 or older
> total_25_or_older <- sum(brand_survey$Frequency[brand_survey$Age == "25-
34"])
> # Frequency of people who prefer brand B aged 25 or older
> prefer_B_and_25_or_older <- brand_survey$Frequency[brand_survey$Age ==
"25-34" & brand_survey$Preference == "B"]
> # Conditional probability of preferring brand B given age 25 or older
> P_prefer_B_given_25_or_older <- prefer_B_and_25_or_older /
total_25_or_older
> P_prefer_B_given_25_or_older

[1] 0.4444444 0.5555556
```

- ✓ **Imagine a population of individuals who have either a high or low level of physical activity. The data is stored in a data frame named activity_data. Use R to find the conditional probability of having a high level of physical activity given that an individual has a healthy BMI.**

```
> # Data frame with activity information
> activity_data <- data.frame(
+   PhysicalActivity = c("High", "Low", "High", "Low"),
+   BMI = c("Healthy", "Overweight", "Healthy", "Overweight"),
+   Frequency = c(40, 30, 25, 35)
+ )
> print(activity_data)
  PhysicalActivity        BMI Frequency
1             High    Healthy        40
2              Low Overweight        30
3             High    Healthy        25
4              Low Overweight        35

> # Total frequency of individuals with a healthy BMI
> total_healthy_bmi <- sum(activity_data$Frequency[activity_data$BMI ==
"Healthy"])
> # Frequency of individuals with high physical activity and a healthy BMI
> high_activity_and_healthy_bmi <-
activity_data$Frequency[activity_data$BMI == "Healthy" &
activity_data$PhysicalActivity == "High"]
> # Conditional probability of having high physical activity given a
healthy BMI
> P_high_activity_given_healthy_bmi <- high_activity_and_healthy_bmi /
total_healthy_bmi
> P_high_activity_given_healthy_bmi

[1] 0.6153846 0.3846154
```

Practical No. 7 Binomial & Poisson distribution and plotting of its pdf, cdf, pmf functions

❖ **BINOMIAL DISTRIBUTION**

✓ A coin is tossed 4 times ,what is the probability of getting no heads , getting one head?

```
> dbinom(0,4,0.5) #Getting no head
[1] 0.0625
> dbinom(1,4,0.5) #getting one head
[1] 0.25
```

✓ A coin is tossed 4 times, what is the probability of getting atleast 2 heads ?

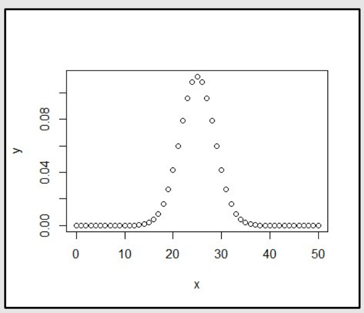
```
> pbinom(1,4,0.5, lower.tail = FALSE)
[1] 0.6875
```

✓ Create a sample of 50 numbers which are increased by 1, create binomial distribution and plot graph

```
> #create a sample of 50 numbers which are increased by 1
> x=seq(0,50,by=1)
> x

 [1]  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
48 49 50

> #creat a binomial distribution
> y=dbinom(x,50,0.5)
> #plot the graph for this sample
> plot(x,y)
```



✓ If 4 survive out of 10 with probability of 50%

```
> dbinom(4,10,0.5)

[1] 0.2050781
```

✓ A person makes 70% of his throw attempts, and if he shoots 20 throws, so what will be the probability that the person makes exactly 12 of them attempt.

```
> dbinom(x=12, size=20, prob=0.7)

[1] 0.1143967
```

❖ POISSON DISTRIBUTION

✓ A website receives an average of 5 visitors per hour. What is the probability of exactly 3 visitors in the next hour?

```
> lambda <- 5  # Average visitors per hour
> k <- 3       # Number of visitors
>
> # Poisson probability for exactly 3 visitors
> prob_3_visitors <- dpois(k, lambda)
> prob_3_visitors

[1] 0.1403739
```

✓ A call center receives an average of 10 calls per minute. What is the probability of receiving fewer than 5 calls in the next minute?

```
> lambda <- 10  # Average calls per minute
> k <- 4        # Number of calls (one less than 5)
>
> # Poisson probability for fewer than 5 calls
> prob_fewer_than_5_calls <- ppois(k, lambda)
> prob_fewer_than_5_calls
[1] 0.02925269
```

✓ A factory produces, on average, 2 defective items per day. What is the probability of having exactly 1 defective item tomorrow?

```
> lambda <- 2  # Average defective items per day
> k <- 1       # Number of defective items
>
> # Poisson probability for exactly 1 defective item
> prob_1_defective <- dpois(k, lambda)
> prob_1_defective
[1] 0.2706706
```

✓ A website receives an average of 5 visitors per hour. Find the number of visitors such that the probability of having fewer than that number is 0.3.

```
> lambda <- 5  # Average visitors per hour
> p_value <- 0.3
>
> # Find the number of visitors for the given probability
> num_visitors <- qpois(p_value, lambda)
> num_visitors
[1] 4
```

❖ A student receives, on average, 4 spam emails per day. Find the number of spam emails such that the probability of having at most that number is 0.2.

```
> lambda <- 4  # Average spam emails per day
> p_value <- 0.2
>
> # Find the number of spam emails for the given probability
> num_spam <- qpois(p_value, lambda)
> num_spam
[1] 2
```

Practical No. 8 Normal distribution and plotting of its pdf, cdf, pmf functions

- If N~(450,100), what is the probability P(400<=x<=500)?

```
> pnorm(500,450,100)-pnorm(399,450,100)

[1] 0.3864367
```

- If N~(450,100), what is the probability P(x>630)?

```
> pnorm(630,450,100, lower.tail=FALSE)

[1] 0.03593032
```

- The weights of a certain species of birds are normally distributed with a mean of 120 grams and a standard deviation of 15 grams. What is the probability that a randomly selected bird weighs between 110 and 130 grams?

```
> # Given data
> mean_weight <- 120
> std_dev_weight <- 15
> x_lower <- 110
> x_upper <- 130
>
> # Probability of weighing between 110 and 130 grams
> prob_between_110_and_130 <- pnorm(x_upper, mean_weight, std_dev_weight)
- pnorm(x_lower, mean_weight, std_dev_weight)
> prob_between_110_and_130
[1] 0.4950149
```

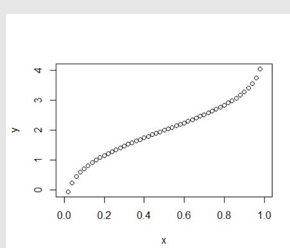
- The IQ scores of a population are normally distributed with a mean of 100 and a standard deviation of 15. What is the 90th percentile of IQ scores?

```
> # Given data
> mean_IQ <- 100
> std_dev_IQ <- 15
>
> # Find the 90th percentile IQ score
> IQ_90th_percentile <- qnorm(0.9, mean_IQ, std_dev_IQ)
> IQ_90th_percentile

[1] 119.2233
```

- Plotting Function

```
> x=seq(0,1,by=0.02)
> y=qnorm(x,mean=2,sd=1)
> plot(x,y)
```



❖ If N~(45,7),what is the probability p(x=40) p(x>=50) p(x<=60) p(52<=X=68)?

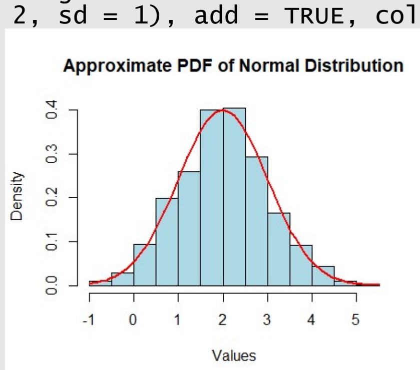
```
> dnorm(40,45,7)
[1] 0.04415934
> pnorm(49,45,7,lower.tail = FALSE)
[1] 0.2838546
> pnorm(60,45,7)
[1] 0.9839377
> pnorm(68,45,7)-pnorm(51,45,7)
[1] 0.1951743
```

❖ If N~(450,100), what is the probability P(x)=480?

```
> pnorm(479,450,100, lower.tail=FALSE)
[1] 0.3859081
```

❖ PDF of Normal Distribution

```
> # Generate random samples from a normal distribution
> set.seed(123)  # for reproducibility
> data <- rnorm(1000, mean = 2, sd = 1)
>
> # Create a histogram to approximate the PDF
> hist(data, probability = TRUE, col = "lightblue", main = "Approximate
PDF of Normal Distribution", xlab = "Values")
> # Add a curve representing the true PDF of the normal distribution
> curve(dnorm(x, mean = 2, sd = 1), add = TRUE, col = "red", lwd = 2)
```



Practical No. 9 Hypothesis testing for different conditions functions

❖ **Using the Student's T-test in R**

The Student's T-test is a method for comparing two samples. It can be implemented to determine whether the samples are different. This is a parametric test, and the data should be normally distributed.

R can handle the various versions of T-test using the t.test() command. The test can be used to deal with two- and one-sample tests as well as paired tests.

✓ One-Sample T-test

```
> x = rnorm(10)
> t.test(x, mu = 5)

        One Sample t-test

data:  x
t = -14.505, df = 9, p-value = 1.509e-07
alternative hypothesis: true mean is not equal to 5
95 percent confidence interval:
 -0.2295253  1.1815272
sample estimates:
mean of x
 0.476001
```

Q. Is there any significant difference in the mean weight of plant heights.

```
> #create vector to hold plant heights
> my_data <- c(14, 14, 16, 13, 12, 17, 15, 14, 15, 13, 15, 14)
> #perform one sample t-test
> t.test(my_data, mu=15)
        One Sample t-test

data:  my_data
t = -1.6848, df = 11, p-value = 0.1201
alternative hypothesis: true mean is not equal to 15
95 percent confidence interval:
 13.46244 15.20423
sample estimates:
mean of x
 14.33333
```

✓ Two-Sample T-test

```
> x = rnorm(10)
> y = rnorm(10)
> t.test(x,y)

	Welch Two Sample t-test

data:  x and y
t = -1.0633, df = 17.886, p-value = 0.3018
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.9785666  0.6492211
sample estimates:
  mean of x   mean of y
-0.07905064  0.58562207
```

✓ Paired T-test

```
> set.seed(123)
> # sales before the program
> sales_before <- rnorm(7, mean = 50000, sd = 50)
> # sales after the program.This has higher mean
> sales_after <- rnorm(7, mean = 50075, sd = 50)
> # draw the distribution
> t.test(sales_before, sales_after,var.equal = TRUE)

	Two Sample t-test

data:  sales_before and sales_after
t = -2.2245, df = 12, p-value = 0.04606
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -99.735277  -1.035312
sample estimates:
mean of x mean of y
 50022.46  50072.84
```

❖ Chi-sq test

The R function chisq.test() can be used as follow:

chisq.test(x, p)

- x: a numeric vector
- p: a vector of probabilities of the same length of x.

✓ Find out any significant correlation between the types of car sold and the type of Air bags it has. If correlation is observed we can estimate which types of cars can sell better with what types of air bags.

```
> # Load the library.
> library("MASS")
> # Create a data frame from the main data set.
> car.data <- data.frame(Cars93$AirBags, Cars93$Type)
> # Create a table with the needed variables.
> car.data = table(Cars93$AirBags, Cars93$Type)
> print(car.data)
```

```
                 Compact Large Midsize Small Sporty Van
  Driver & Passenger       2     4       7     0      3   0
  Driver only              9     7      11     5      8   3
  None                     5     0       4    16      3   6
> # Perform the Chi-Square test.
> print(chisq.test(car.data))

        Pearson's Chi-squared test

data:  car.data
X-squared = 33.001, df = 10, p-value = 0.0002723
```

- ✓ Are the colors equally common?

```
> tulip <- c(81, 50, 27)
> res <- chisq.test(tulip, p = c(1/3, 1/3, 1/3))
> res

        Chi-squared test for given probabilities

data:  tulip
X-squared = 27.886, df = 2, p-value = 8.803e-07
```

The p-value of the test is 8.80310^{-7}, which is less than the significance level alpha = 0.05. We can conclude that the colors are significantly not commonly distributed

- ✓ comparing observed to expected proportions

```
> tulip <- c(81, 50, 27)
> res <- chisq.test(tulip, p = c(1/2, 1/3, 1/6))
> res

        Chi-squared test for given probabilities

data:  tulip
X-squared = 0.20253, df = 2, p-value = 0.9037
```

The p-value of the test is 0.9037, which is greater than the significance level alpha = 0.05. We can conclude that the observed proportions are not significantly different from the expected proportions.

- ✓ The following table shows the results of the survey:

	Republican	Democrat	Independent	Total
Male	120	90	40	250
Female	110	95	45	250
Total	230	185	85	500

```
> #create table
> data <- matrix(c(120, 90, 40, 110, 95, 45), ncol=3, byrow=TRUE)
> colnames(data) <- c("Rep","Dem","Ind")
> rownames(data) <- c("Male","Female")
> data <- as.table(data)
> data
       Rep Dem Ind
```

```
Male   120  90  40
Female 110  95  45
> #Perform Chi-Square Test of Independence
> chisq.test(data)

        Pearson's Chi-squared test

data:  data
X-squared = 0.86404, df = 2, p-value = 0.6492
```

❖ Sign Test in R

The sign test is used to compare the medians of paired or matched observations.

```
> # Create a vector of data
> data <- c(-1, 2, -3, 4, -5, 6, -7, 8)
> # Perform sign test using the binomial test function
> binom_test_result <- binom.test(sum(data > 0), length(data), p=0.5,
alternative="two.sided")
> # View the results
> print(binom_test_result)

        Exact binomial test

data:  sum(data > 0) and length(data)
number of successes = 4, number of trials = 8, p-value = 1
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.1570128 0.8429872
sample estimates:
probability of success
                   0.5
```

- A soft drink company has invented a new drink, and would like to find out if it will be as popular as the existing favorite drink. For this purpose, its research department arranges 18 participants for taste testing. Each participant tries both drinks in random order before giving his or her opinion.

```
> binom.test(5, 18)

        Exact binomial test

data:  5 and 18
number of successes = 5, number of trials = 18, p-value = 0.09625
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.09694921 0.53480197
sample estimates:
probability of success
             0.2777778
```

Practical No. 10: Analysis of Variance functions

- ✓ A researcher wants to determine if there is a significant difference in the average scores of three teaching methods. The scores of students taught with Method A, Method B, and Method C are collected. Perform an ANOVA test at a significance level of 0.05.

```
> # Given data
> scores_method_A <- c(80, 85, 88, 92, 78)
> scores_method_B <- c(75, 82, 88, 90, 79)
> scores_method_C <- c(82, 86, 84, 88, 80)
> # Combine data into a data frame
> data_anova <- data.frame(
+   Scores = c(scores_method_A, scores_method_B, scores_method_C),
+   Method = rep(c("A", "B", "C"), each = 5)
+ )
> # Perform ANOVA test
> anova_result <- aov(Scores ~ Method, data = data_anova)
> summary(anova_result)
            Df Sum Sq Mean Sq F value Pr(>F)
Method       2    8.4    4.20   0.155  0.858
Residuals   12  326.0   27.17
```

- ✓ A study is conducted to compare the average weights of four different diets. The weights of participants after following each diet for 8 weeks are collected. Conduct an ANOVA test at a significance level of 0.01.

```
> # Given data
> weights_diet_1 <- c(160, 155, 165, 158, 162)
> weights_diet_2 <- c(150, 155, 148, 145, 152)
> weights_diet_3 <- c(165, 170, 168, 175, 160)
> weights_diet_4 <- c(140, 145, 138, 142, 135)
>
> # Combine data into a data frame
> data_anova_diet <- data.frame(
+   Weights = c(weights_diet_1, weights_diet_2, weights_diet_3,
weights_diet_4),
+   Diet = rep(c("Diet 1", "Diet 2", "Diet 3", "Diet 4"), each = 5)
+ )
>
> # Perform ANOVA test
> anova_result_diet <- aov(Weights ~ Diet, data = data_anova_diet)
> summary(anova_result_diet)
            Df Sum Sq Mean Sq F value   Pr(>F)
Diet         3 2161.6   720.5   38.53 1.51e-07 ***
Residuals   16  299.2    18.7
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

✓ An experiment is conducted to compare the tensile strengths of materials produced by three different manufacturers. The tensile strengths (in MPa) of samples from each manufacturer are recorded. Conduct an ANOVA test at a significance level of 0.05.

```
> # Given data
> strength_manufacturer_A <- c(50, 55, 48, 52, 49)
> strength_manufacturer_B <- c(60, 58, 62, 55, 59)
> strength_manufacturer_C <- c(45, 48, 50, 47, 52)
>
> # Combine data into a data frame
> data_anova_strength <- data.frame(
+   TensileStrength = c(strength_manufacturer_A, strength_manufacturer_B,
strength_manufacturer_C),
+   Manufacturer = rep(c("A", "B", "C"), each = 5)
+ )
>
> # Perform ANOVA test
> anova_result_strength <- aov(TensileStrength ~ Manufacturer, data =
data_anova_strength)
> summary(anova_result_strength)
             Df Sum Sq Mean Sq F value   Pr(>F)
Manufacturer  2  296.5  148.27    20.5 0.000135 ***
Residuals    12   86.8    7.23
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

❖ **Two Way Anova**

A pharmaceutical company is testing the effectiveness of a new drug. The response variable is the recovery time (in hours) after surgery. Two factors, Drug Type (A and B) and Dosage (Low and High), are considered. Perform a Two-Way ANOVA at a significance level of 0.01.

```
> # Given data
> data_anova_drug <- read.table(header=TRUE, text="
+   RecoveryTime DrugType Dosage
+   8            A        Low
+   10           B        Low
+   9            A        High
+   12           B        High
+   7            A        Low
+   11           B        Low
+   10           A        High
+   13           B        High
+ ")
>
> # Perform Two-Way ANOVA
> anova_result_drug <- aov(RecoveryTime ~ DrugType * Dosage, data =
data_anova_drug)
> summary(anova_result_drug)
                Df Sum Sq Mean Sq F value  Pr(>F)
DrugType         1     18    18.0      36 0.00388 **
Dosage           1      8     8.0      16 0.01613 *
DrugType:Dosage  1      0     0.0       0 1.00000
Residuals        4      2     0.5
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

An agricultural experiment is conducted to study the yield of crops. The factors considered are Fertilizer Type (A, B, and C) and Irrigation Level (Low and High). The response variable is the crop yield (in tons). Perform a Two-Way ANOVA at a significance level of 0.05.

```
> # Given data
> data_anova_yield <- read.table(header=TRUE, text="
+   CropYield FertilizerType IrrigationLevel
+   5         A              Low
+   6         B              Low
+   4         C              Low
+   7         A              High
+   5         B              High
+   6         C              High
+   4         A              Low
+   7         B              Low
+   5         C              Low
+   8         A              High
+   6         B              High
+   7         C              High
+ ")
>
> # Perform Two-Way ANOVA
> anova_result_yield <- aov(CropYield ~ FertilizerType * IrrigationLevel,
data = data_anova_yield)
> summary(anova_result_yield)
                               Df Sum Sq Mean Sq F value Pr(>F)
FertilizerType                  2  0.667   0.333   0.667 0.5477
IrrigationLevel                 1  5.333   5.333  10.667 0.0171 *
FertilizerType:IrrigationLevel  2  8.667   4.333   8.667 0.0170 *
Residuals                       6  3.000   0.500
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Practical No. 11: Non-Parametric Test functions

❖ Wilcoxon signed rank test

```
> set.seed(1234)
> my_data <- data.frame(
+  name = paste0(rep("M_", 10), 1:10),
+  weight = round(rnorm(10, 20, 2), 1)
+ )
> # Print the first 10 rows of the data
> head(my_data, 10)
   name weight
1   M_1   17.6
2   M_2   20.6
3   M_3   22.2
4   M_4   15.3
5   M_5   20.9
6   M_6   21.0
7   M_7   18.9
8   M_8   18.9
9   M_9   18.9
10 M_10   18.2
> # Statistical summaries of weight
> summary(my_data$weight)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  15.30   18.38   18.90   19.25   20.82   22.20
> # One-sample wilcoxon test
> res <- wilcox.test(my_data$weight, mu = 25)
Warning message:
In wilcox.test.default(my_data$weight, mu = 25) :
  cannot compute exact p-value with ties
> # Printing the results
> res

        Wilcoxon signed rank test with continuity correction

data:  my_data$weight
V = 0, p-value = 0.005793
alternative hypothesis: true location is not equal to 25

> # print only the p-value
> res$p.value
[1] 0.005793045
```

❖ Sign Test for One-sample Data

```
> Data <- read.table(header=TRUE, stringsAsFactors=TRUE, text='
+ Speaker Rater Likert
+ "Maggie Simpson" 1 3
+ "Maggie Simpson" 2 4
+ "Maggie Simpson" 3 5
+ "Maggie Simpson" 4 4
+ "Maggie Simpson" 5 4
+ "Maggie Simpson" 6 4
+ "Maggie Simpson" 7 4
+ "Maggie Simpson" 8 3
+ "Maggie Simpson" 9 2
+ "Maggie Simpson" 10 5
+ ')
> library(psych)
> str(Data)
'data.frame':  10 obs. of  3 variables:
 $ Speaker: Factor w/ 1 level "Maggie Simpson": 1 1 1 1 1 1 1 1 1 1
 $ Rater  : int  1 2 3 4 5 6 7 8 9 10
 $ Likert : int  3 4 5 4 4 4 4 3 2 5
```

```
> summary(Data)
           Speaker       Rater          Likert
 Maggie Simpson:10   Min.   : 1.00   Min.   :2.00
                     1st Qu.: 3.25   1st Qu.:3.25
                     Median : 5.50   Median :4.00
                     Mean   : 5.50   Mean   :3.80
                     3rd Qu.: 7.75   3rd Qu.:4.00
                     Max.   :10.00   Max.   :5.00
> library(DescTools)
> SignTest(Data$Likert,
+  mu = 3)

        One-sample Sign-Test

data:  Data$Likert
S = 7, number of differences = 8, p-value = 0.07031
alternative hypothesis: true median is not equal to 3
97.9 percent confidence interval:
 3 5
sample estimates:
median of the differences
                        4
```

- **Perform two-sided sign test using the binomial test function**

```
> # Create a vector of data
> data <- c(-1, 2, -3, 4, -5, 6, -7, 8)
> binom_test_result <- binom.test(sum(data > 0), length(data), p=0.5,
+  alternative="two.sided")
> # View the results
> print(binom_test_result)

        Exact binomial test

data:  sum(data > 0) and length(data)
number of successes = 4, number of trials = 8, p-value = 1
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.1570128 0.8429872
sample estimates:
probability of success
                   0.5
```