Practical 1

Using 'R' Execute the Basic Array, List and Frames.

1. Hello World Program:

```
> # My first program in R Programming
> s<-"Welcome to SIES"
> print (s)
[1] "Welcome to SIES"
> print (s, quote=FALSE)
[1] Welcome to SIES
```

2. R – Datatypes:

i. Numeric

ii. Integer

iii. Complex

iv. Logical

v. Character

```
> #Numeric
> x<-5
> y<-7
> z<-x+y
> class(z)
[1] "numeric"
> #Integer
> X<-51
> y<-7L
> z<-x+y
> class(z)
[1] "numeric"
> #Complex
> x<-2+3l
> y<-3-2l
> z<-x+y
> class(z)
[1] "numeric"
> # Logical
> X<-TRUE
> y<-FALSE
> z<-X&y
> class (z)
[1] "logical"
> # Character
> course<-"BSc IT"
> course
[1] "BSc IT"
> class (course)
[1] "character"
```

3. Vector

All elements must be of the same type. To combine the list of items to a vector, use the c() function and separate the items by a comma.

```
> age <- c(11,7)
> name <- c("Yashashree", "Vedshree")
> print(age)
[1] 11  7
> print(name)
[1] "Yashashree" "Vedshree"
> print(name[1])
[1] "Yashashree"
> print(name[2])
[1] "Vedshree"
```

4. Array & Matrix

Matrix is a special kind of vector. A matrix is a vector with two additional attributes: the number of rows and the number of columns. A matrix can be created with the matrix() function. Specify the nrow and ncol parameters to get the amount of rows and columns. The c() function is used to concatenate items together.

```
> s <- matrix(c(11,22,33,44), nrow=2, ncol=2)
> s
     [,1] [,2]
[1,]   11   33
[2,]   22   44
```

5. List

List can contain elements of different types.

```
> s <- list(name="Jivika", gender="F", company="PDB")
> s
$name
[1] "Jivika"

$gender
[1] "F"

$company
[1] "PDB"
```

6. Data Frame

A data frame is used for storing data tables. It is a list of vectors of equal length.

```
> name <- c("Yashashree", "Vedshree")
> age <- c(11,7)
> class <- c("6th", "1st")
> df = data.frame(name, age, class)
> df
        name age class
1 Yashashree  11   6th
2   Vedshree   7   1st
```

PRACTICAL No. 2

MATRIX COMPUTATIONS

AIM: Create a Matrix using R and Perform the operations addition, subtraction, multiplication, transpose, inverse.

```
> #Creating First matrix
> m1 <- matrix(data = 1:4, nrow = 2, ncol = 2)
> print(m1)
     [,1] [,2]
[1,]    1    3
[2,]    2    4
>
> #Creating Second matrix
> m2 <- matrix(data = 1:4, nrow = 2, ncol = 2)
> print(m2)
     [,1] [,2]
[1,]    1    3
[2,]    2    4
>
> #Adding Both Matrices
> result <- m1 + m2
> print(result)
     [,1] [,2]
[1,]    2    6
[2,]    4    8
>
> #Subtracting Both Matrices
> result <- m1 - m2
> print(result)
     [,1] [,2]
[1,]    0    0
[2,]    0    0
>
> #Multiplying Both Matrices
> result <- m1 %*% m2
> print(result)
     [,1] [,2]
[1,]    7   15
[2,]   10   22
```

```
> # transpose of matrix using t() function.
> t <- t(m1)
> print(t)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
>
> # inverse of matrix usingsolve() function
> t2 <- solve(m2)
> print(t2)
     [,1] [,2]
[1,]   -2  1.5
[2,]    1 -0.5
```

PRACTICAL No. 3

STATISTICAL FUNCTIONS

AIM: Using R Execute the statistical functions: mean, median, mode, quartiles, range, inter quartile range, histogram.

The mean is the average of a data set. We use the mean() function to calculate the mean.

```
> m <- c(97, 78, 57, 64, 87)
> r <- mean(m)
> print(r)
[1] 76.6
>
```

The median is the middle of the set of numbers. We use the median() function to calculate the median.

```
> m <- c(97, 78, 57, 64, 87)
> r <- median(m)
> print(r)
[1] 78
```

The mode is the most common number in a data set. R does not have a standard in-built function to calculate mode.

```
> # Creating getMode function
> getmode <- function(x)
+ {
+   uniqv <- unique(x)
+   uniqv[which.max(tabulate(match(x,uniqv)))]
+ }
>
> x <- c(11,22,33,4,5,12,3,2,5,2,2)
> print(getmode(x))
[1] 2
> y<- c('IT', 'IT', 'CS', 'PM', 'CS', 'OS', 'IT', 'FM')
> print(getmode(y))
[1] "IT"
```

uniqv <- unique(x)
uniqv
[1] 11 22 33 4 5 12 3 2

match(x,uniqv)
[1] 1 2 3 4 5 6 7 8 5 8 8
tabulate(match(x,uniqv))
[1] 1 1 1 1 2 1 1 3
which.max(tabulate(match(x,uniqv)))
[1] 8
uniqv[which.max(tabulate(match(x,uniqv)))]
[1] 2

A percentile is a statistical measure that indicates the value below which a percentage of data falls. For example, the 70th percentile is the value below which 70% of the observations may be found. we use the quantile() function to calculate the percentile.

```
> marks <- c(97, 78, 57, 64, 87)
> # calculate 70th percentile of marks
> result <- quantile(marks, 0.70)
> print(result)
 70%
85.2
>
```

The range() function is used to find the minimum and maximum values of a numeric vector or a sequence of values. It returns a vector containing two elements: the minimum value and the maximum value within the specified range.

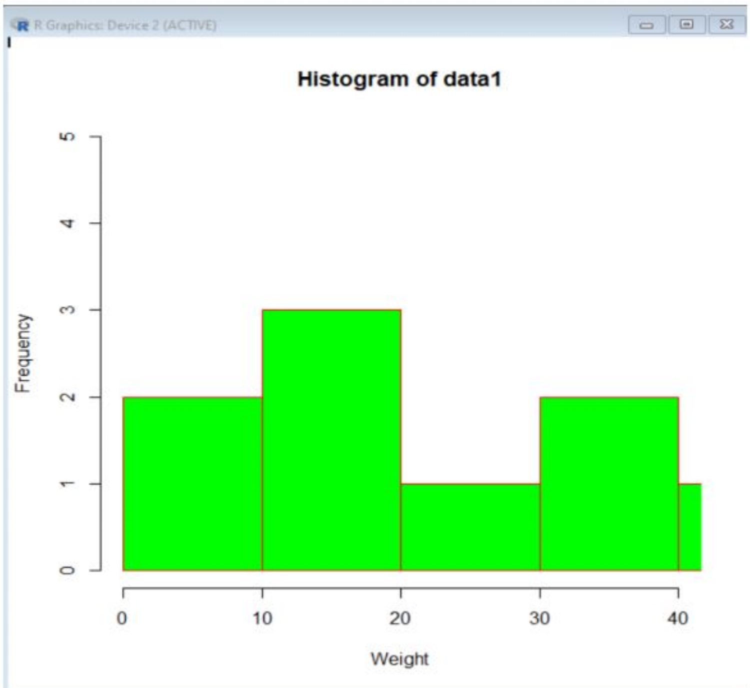
```
> arr <- c(-10, -15, 5, 19, 27, 0)
> range(arr)
[1] -15  27
> arr <- c(-10, -15, 5, 19, 27, 0)
> range(arr,na.rm = TRUE)
[1] -15  27
> values <- c(5, 12, 8, 20, 3, NA, 15)
> data_range <- range(values, na.rm = TRUE)
> c("Minimum value:", data_range[1])
[1] "Minimum value:" "3"
> c("Maximum value:", data_range[2])
[1] "Maximum value:" "20"
```

You can calculate the interquartile range (IQR) of a numeric vector using the IQR() function. The interquartile range is a measure of statistical dispersion and is the range between the first quartile (25th percentile) and the third quartile (75th percentile) of a dataset.

```
> # Create a numeric vector
> data <- c(7, 10, 15, 20, 22, 25, 30, 40, 75, 90)
> # Calculate the interquartile range
> iqr_value <- IQR(data)
> # Print the result
> cat("Interquartile Range:", iqr_value, "\n")
Interquartile Range: 21.25
```

We can create a histogram to visualize the distribution of a numeric variable using the hist() function. Create a numeric vector data containing the data you want to visualize. Use the hist() function to create a histogram of the data.
hist(data, main, xlab, ylab, xlim, ylim, col, border)
main: A title for the histogram.
xlab and ylab: Labels for the x-axis and y-axis, respectively.
xlim and ylim: The limits for the x-axis and y-axis.
col: The color of the bars.
border: The color of the borders of the bars.

```
> data1 <- c(8,12,36,17,19,25,34,47,9)
> # Creating Histogram with Various Available Options
> hist(data1,xlab = "Weight",col = "green",border = "red", xlim = c(0,40), ylim = c(0,5))
```



PRACTICAL No. 4

FINDING MEAN, MEDIAN, MODE, QUARTILES, RANGE, INTER-QUARTILE RANGE, & HISTOGRAM OF EXCEL/.CSV DATA

AIM: Using R import the data from Excel/.CSV file and find mean, median, mode, quartiles, range, inter quartile range, histogram.

Working with CSV File:

1. Copy and paste .csv file in working directory.

2. Importing Data from .CSV File:

> emp <- read.csv("D:/employee.csv")

> print(emp)

```
> emp <- read.csv("D:/employee.csv")
> print(emp)
     EEID          Full.Name                   Job.Title      Department
1  E02387        Emily Davis                  Sr. Manger              IT
2  E04105     Theodore Dinh         Technical Architect              IT
3  E02572       Luna Sanders                    Director         Finance
4  E02832    Penelope Jordan     Computer Systems Manager             IT
5  E01639          Austin Vo                 Sr. Analyst         Finance
6  E00644       Joshua Gupta      Account Representative           Sales
7  E01550        Ruby Barnes                     Manager              IT
8  E04332        Luke Martin                     Analyst         Finance
9  E04533      Easton Bailey                     Manager      Accounting
10 E03838   Madeline Walker                 Sr. Analyst         Finance
11 E00591       Savannah Ali                  Sr. Manger Human Resources
12 E03344      Camila Rogers          Controls Engineer     Engineering
13 E00530          Eli Jones                     Manager Human Resources
14 E04239      Everleigh Ng                  Sr. Manger         Finance
15 E03496       Robert Yang                 Sr. Analyst      Accounting
```

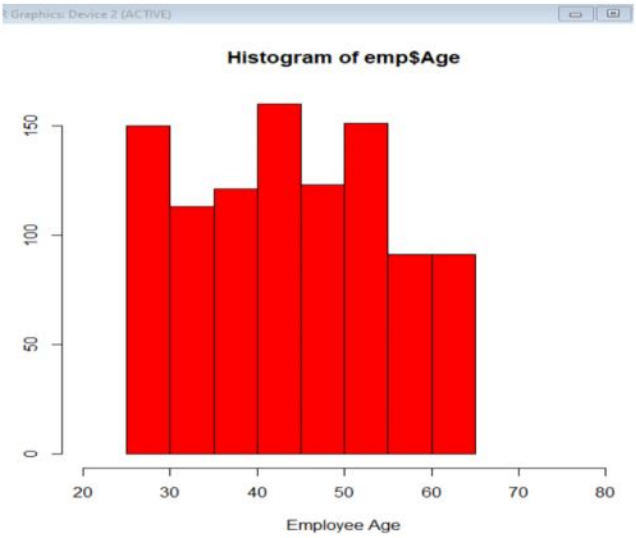
```
> r <- mean(emp$Age)
> print(r)
[1] 44.382
>
> r <- median(emp$Age)
> print(r)
[1] 45
>
> result <- quantile(emp$Age)
> print(result)
  0%  25%  50%  75% 100%
  25   35   45   54   65
>
> result <- quantile(emp$Age,0.70)
> print(result)
70%
 52
>
```

```
> data <- c(emp$Age)
> iqr_value <- IQR(data)
> print(iqr_value)
[1] 19
```

```
> data_range <- range(emp$Age)
> c("Minimum value:", data_range[1])
[1] "Minimum value:" "25"
> c("Maximum value:", data_range[2])
[1] "Maximum value:" "65"
```

```
> # Creating getMode function
> getmode <- function(x)
+ {
+   uniqv <- unique(x)
+   uniqv[which.max(tabulate(match(x,uniqv)))]
+ }
>
> x <- c(emp$Age)
> print(getmode(x))
[1] 45
```

```
> result <- hist(emp$Age, xlab = "Employee Age", col = "red", xlim = c(20,80))
```



Working with Excel File:

1. Install the readxl package

    install.packages("readxl")

2. Copy and paste . xlsx file in working directory.

3. Load the readxl package

    library(readxl)

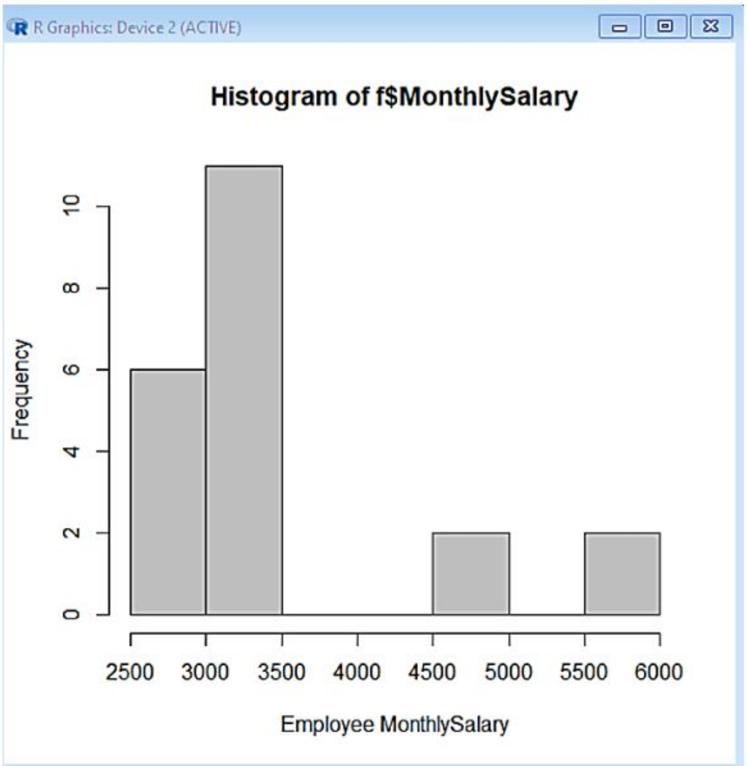
4. Read the Excel file

f <- read_excel("D:\r cost pract\emp.xlsx")

print(emp)

```
> library(readxl)
Warning message:
package 'readxl' was built under R version 4.3.2
> f <- read_excel("D:/r cost pract/emp.xlsx")
> print(f)
# A tibble: 21 × 6
   Name    JoiningDate         EmailAddress    Department MonthlySalary JobStatus
   <chr>   <dttm>              <chr>           <chr>              <dbl> <chr>
 1 Mark    2021-12-31 00:00:00 Mark@demomail… Human Res…          5830 Permanent
 2 Brian   2021-12-31 00:00:00 Brian@demomai… Sales              3450 Permanent
 3 Alan    2022-01-14 00:00:00 Alan@demomail… Legal              4920 Permanent
 4 Tony    2022-01-14 00:00:00 Tony@demomail… Retail             2785 Permanent
 5 Agatha  2022-02-01 00:00:00 Agatha@demoma… Sales              3450 Permanent
 6 Lana    2022-02-01 00:00:00 Lana@demomail… Accounting         3125 Permanent
 7 Heather 2022-08-04 00:00:00 Heather@demom… Accounting         3125 Temporary
 8 Ben     2022-08-04 00:00:00 Ben@demomail.… Sales              3450 Temporary
 9 Caitlyn 2022-03-01 00:00:00 Caitlyn@demom… Retail             2785 Permanent
10 Gibbs   2022-03-01 00:00:00 Gibbs@demomai… Retail             2785 Permanent
# ℹ 11 more rows
# ℹ Use `print(n = ...)` to see more rows
>
> r <- mean(f$MonthlySalary)
> print(r)
[1] 3501.905
>
> r <- median(f$MonthlySalary)
> print(r)
[1] 3125
>
> result <- quantile(f$MonthlySalary)
> print(result)
  0%  25%  50%  75% 100%
2500 2785 3125 3450 5830
>
> data_range <- range(f$MonthlySalary)
> c("Minimum value:", data_range[1])
[1] "Minimum value:" "2500"
> c("Maximum value:", data_range[2])
[1] "Maximum value:" "5830"
>
> data <- c(f$MonthlySalary)
> iqr_value <- IQR(data)
> print(iqr_value)
[1] 665
```

```
> # Creating getMode function
> getmode <- function(x)
+  {
+    uniqv <- unique(x)
+    uniqv[which.max(tabulate(match(x,uniqv)))]
+  }
>
> x <- c(f$MonthlySalary)
> print(getmode(x))
[1] 3450
>
> result <- hist(f$MonthlySalary, xlab = "Employee MonthlySalary", col = "gray")
>
```



PRACTICAL No. 5

FINDING STANDARD DEVIATION,

VARIANCE & CO-VARIANCE

OF EXCEL/.CSV DATA

AIM: Using R import the data from Excel/.CSV file and find standard deviation, variance and co-variance.

- **Standard Deviation (sd()):**

Syntax: sd(data)

Description: Calculates the standard deviation of a numeric vector or a column in a dataset. In short it shows how spread out the data is.

- **Variance (var()):**

Syntax: var(data)

Description: Computes the variance of a numeric vector or a column in a dataset. In short it measures how much data values deviate from the mean.

- **Covariance (cov()):**

Syntax: cov(data1, data2) or cov(data) for the covariance matrix of multiple variables.

Description: Computes the covariance between two numeric vectors or columns in a dataset. In short it measures how two sets of data change together. A positive value indicates a direct relationship (both tend to increase or decrease together), while a negative value suggests an inverse relationship (one tends to increase as the other decreases).

p5.xlsx

Month	Gold	Bitcoin
Jan	0.1	1
Feb	0.08	2
Mar	-0.05	3
Apr	0.1	4
May	0.06	5
Jun	0.08	6
Jul	0.1	7
Aug	0.03	8
Sep	0.08	9
Oct	0.05	10
Nov	0.06	11
Dec	0.07	12

p51.csv

Month	Gold	Bitcoin
Jan	0.1	1
Feb	0.08	2
Mar	0.05	3
Apr	0.1	4
May	0.06	5
Jun	0.08	6
Jul	0.1	7
Aug	0.03	8
Sep	0.08	9
Oct	0.05	10
Nov	0.06	11
Dec	0.07	12

Working with Excel File:

```
> library(readxl)
> f <- read_excel("D:/r cost pract/p5.xlsx")
> print(f)
# A tibble: 12 × 3
   Month  Gold Bitcoin
   <chr> <dbl>   <dbl>
 1 Jan    0.1        1
 2 Feb    0.08       2
 3 Mar   -0.05       3
 4 Apr    0.1        4
 5 May    0.06       5
 6 Jun    0.08       6
 7 Jul    0.1        7
 8 Aug    0.03       8
 9 Sep    0.08       9
10 Oct    0.05      10
11 Nov    0.06      11
12 Dec    0.07      12
>
> r <- sd(f$Gold)
> print(r)
[1] 0.04163332
>
> r <- var(f$Gold)
> print(r)
[1] 0.001733333
>
> r <- cov(f$Gold, f$Bitcoin)
> print(r)
[1] 0.0009090909
```

Working with CSV File:

```
> f <- read.csv("D:/r cost pract/p51.csv")
> print(f)
   Month Gold Bitcoin
1    Jan 0.10       1
2    Feb 0.08       2
3    Mar 0.05       3
4    Apr 0.10       4
5    May 0.06       5
6    Jun 0.08       6
7    Jul 0.10       7
8    Aug 0.03       8
9    Sep 0.08       9
10   Oct 0.05      10
11   Nov 0.06      11
12   Dec 0.07      12
>
> r <- sd(f$Gold)
> print(r)
[1] 0.02249579
>
> r <- var(f$Gold)
> print(r)
[1] 0.0005060606
>
> r <- cov(f$Gold, f$Bitcoin)
> print(r)
[1] -0.03090909
```

PRACTICAL No. 6

FINDING SKEWNESS & KURTOSIS

OF EXCEL/.CSV DATA

AIM: Using R import the data from Excel/.CSV file and find skewness and kutosis.

Skewness: It tells you how lopsided a distribution is. (Imagine a group of numbers lined up. If most of the numbers gather towards one end, it's lopsided or skewed. We're looking at how much the data leans or tilts to one side instead of being balanced or symmetrical like a mirror image.)

- Positive skewness: Data is skewed to the right (tail on the right).
- Negative skewness: Data is skewed to the left (tail on the left).
- Skewness of 0: Perfectly symmetrical.

Kurtosis: Describes the tails and the peak of a distribution.(Kurtosis tells us how much the center of a group of numbers is squeezed together or spread out.)

- High kurtosis (>3): Tails are heavier, distribution is more peaked.
- Low kurtosis (<3): Tails are lighter, distribution is less peaked.
- Kurtosis of 3: Standard (normal distribution).

p5.xlsx

Month	Gold	Bitcoin
Jan	0.1	1
Feb	0.08	2
Mar	-0.05	3
Apr	0.1	4
May	0.06	5
Jun	0.08	6
Jul	0.1	7
Aug	0.03	8
Sep	0.08	9
Oct	0.05	10
Nov	0.06	11
Dec	0.07	12

p51.csv

Month	Gold	Bitcoin
Jan	0.1	1
Feb	0.08	2
Mar	0.05	3
Apr	0.1	4
May	0.06	5
Jun	0.08	6
Jul	0.1	7
Aug	0.03	8
Sep	0.08	9
Oct	0.05	10
Nov	0.06	11
Dec	0.07	12

```
> install.packages("moments")
Installing package into 'C:/Users/sonal/AppData/Local/R/win-library/4.3'
(as 'lib' is unspecified)
--- Please select a CRAN mirror for use in this session ---
trying URL 'https://cloud.r-project.org/bin/windows/contrib/4.3/moments_0.14.1.zip'
Content type 'application/zip' length 56091 bytes (54 KB)
downloaded 54 KB

package 'moments' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\sonal\AppData\Local\Temp\RtmpUPi2x2\downloaded_packages
```

Using EXCEL DATA

```
> library(moments)
> library(readxl)
> f <- read_excel("D:/r cost pract/p5.xlsx")
> print(f)
# A tibble: 12 × 3
   Month  Gold Bitcoin
   <chr> <dbl>   <dbl>
 1 Jan    0.1        1
 2 Feb    0.08       2
 3 Mar   -0.05       3
 4 Apr    0.1        4
 5 May    0.06       5
 6 Jun    0.08       6
 7 Jul    0.1        7
 8 Aug    0.03       8
 9 Sep    0.08       9
10 Oct    0.05      10
11 Nov    0.06      11
12 Dec    0.07      12
> r <- skewness(f$Gold)
> print(r)
[1] -1.754061
> r <- kurtosis(f$Gold)
> print(r)
[1] 5.674312
```

Using .CSV DATA

```
> library(moments)
> library(readxl)
> f <- read.csv("D:/r cost pract/p51.csv")
> print(f)
   Month Gold Bitcoin
1    Jan 0.10       1
2    Feb 0.08       2
3    Mar 0.05       3
4    Apr 0.10       4
5    May 0.06       5
6    Jun 0.08       6
7    Jul 0.10       7
8    Aug 0.03       8
9    Sep 0.08       9
10   Oct 0.05      10
11   Nov 0.06      11
12   Dec 0.07      12
> r <- skewness(f$Gold)
> print(r)
[1] -0.2159291
> r <- kurtosis(f$Gold)
> print(r)
[1] 2.106529
```

PRACTICAL No. 7

Import the data from Excel / .CSV and perform the hypothetical testing.

Aim: Import the data from Excel / .CSV and perform the hypothetical testing.

Hypotheses

A hypothesis is a smart guess about something you're investigating. It's an idea you can test to see if it's true or not.

Example

Scenario: You work for a company that claims their coffee vending machine dispenses an average of 250ml of coffee per cup. You want to test this claim.

Hypotheses:

- **Null Hypothesis (H0)**: The mean amount of coffee dispensed by the machine is 250ml.
- **Alternative Hypothesis (H1)**: The mean amount of coffee dispensed by the machine is different from 250ml.

Steps:

1. **Collect Data**: Take a sample of, say, 30 cups of coffee from the vending machine and measure the amount of coffee in each cup.
2. **Formulate Hypotheses**: Based on the company's claim, set up the null and alternative hypotheses.
3. **Perform the Test**: Using a one-sample t-test in R or other statistical software:

```
# Sample data (replace with actual data)
coffee_amounts <- c(245, 255, 248, 252, 253, 249, 247, 251, 250,
247, 246, 248, 252, 254, 249, 253, 250, 251, 247, 246, 249, 251, 254,
248, 250, 253, 247, 248, 250, 252)

# Performing one-sample t-test
t_test_result <- t.test(coffee_amounts, mu = 250)
t_test_result
```

```
> library(readxl)
> f <- read_excel("D:/r cost pract/p5.xlsx")
> t_test_result <- t.test(f$Gold, f$Bitcoin)
> print(t_test_result)

        Welch Two Sample t-test

data:  f$Gold and f$Bitcoin
t = -6.1837, df = 11.003, p-value = 6.864e-05
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -8.727603 -4.145730
sample estimates:
 mean of x  mean of y
0.06333333 6.50000000

> f <- read.csv("D:/r cost pract/p51.csv")
> t_test_result <- t.test(f$Gold, f$Bitcoin)
> print(t_test_result)

        Welch Two Sample t-test

data:  f$Gold and f$Bitcoin
t = -6.176, df = 11.001, p-value = 6.945e-05
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -8.719214 -4.137453
sample estimates:
 mean of x  mean of y
0.07166667 6.50000000
```

Practical No. 8

Import the data from Excel / .CSV and perform the Chi-squared Test.

The Chi-squared test

The Chi-squared test is a statistical tool used to check if there's a meaningful relationship between categories. The Chi-squared test checks if categories are linked or if they're not related at all. It helps us figure out if there's a connection or if it's just random chance.

Example

You have a dataset of students and their favorite subjects, and you want to test if there's a significant association between gender and favorite subject.

Student	Gender	FavSubject
Student 1	Male	Math
Student 2	Female	Science
Student 3	Male	Math
Student 4	Female	Math
Student 5	Male	Science
Student 6	Male	Math
Student 7	Female	Science
Student 8	Male	Math
Student 9	Female	Math
Student 10	Male	Science
Student 11	Male	Math
Student 12	Female	Science
Student 13	Male	Math
Student 14	Female	Math
Student 15	Male	Science
Student 16	Male	Math
Student 17	Female	Science
Student 18	Male	Math
Student 19	Female	Math
Student 20	Male	Science
Student 21	Male	Math
Student 22	Female	Science
Student 23	Male	Math
Student 24	Female	Math
Student 25	Male	Science

This dataset contains categorical variables: Gender (Male/Female) and Favorite Subject (Math/Science). We'll perform a Chi-squared test to determine if there's a relationship between these variables.

```
> data <- read.csv("D:/r cost pract/student.csv")
> a <- table(data$Gender, data$FavSubject)
> r <- chisq.test(a)
Warning message:
In chisq.test(a) : Chi-squared approximation may be incorrect
> print(r)

        Pearson's Chi-squared test with Yates' continuity correction

data:  a
X-squared = 0.17361, df = 1, p-value = 0.6769
```

```
> library(readxl)
> data <- read_excel ("D:/r cost pract/student.xlsx")
> a <- table(data$Gender, data$FavSubject)
> r <- chisq.test(a)
Warning message:
In chisq.test(a) : Chi-squared approximation may be incorrect
> print(r)

        Pearson's Chi-squared test with Yates' continuity correction

data:  a
X-squared = 0.17361, df = 1, p-value = 0.6769
```

Note: Here Warning shows because of less data.

A small p-value (less than 0.05) means there's likely a real connection between things we're comparing. A big p-value (more than 0.05) suggests the connection might be just random chance.

Practical 9

Using R/Python perform the binomial and normal distribution on the data.

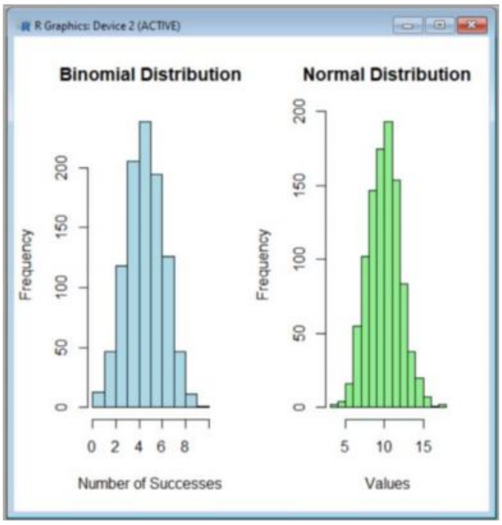
```
binomial_data <- rbinom(1000, 10, 0.5)

normal_data <- rnorm(1000, 10, 2)

par(mfrow = c(1, 2))

hist(binomial_data, main = "Binomial Distribution", xlab = "Number of Successes", col = "lightblue")

hist(normal_data, main = "Normal Distribution", xlab = "Values", col = "lightgreen")
```



Binomial Distribution

binomial_data <- rbinom(1000, 10, 0.5)

Generates 1000 random values from a binomial distribution with 10 trials and a success probability of 0.5.

Results are stored in the variable 'binomial_data'.

Normal Distribution

normal_data <- rnorm(1000, 10, 2)

Generates 1000 random values from a normal distribution with mean 10 and standard deviation 2.

Results are stored in the variable 'normal_data'.

Set up a 1x2 grid for plotting

par(mfrow = c(1, 2))

Sets up a grid for plotting with one row and two columns.

Plotting Binomial Distribution

hist(binomial_data, main = "Binomial Distribution", xlab = "Number of Successes", col = "lightblue")

Creates a histogram of 'binomial_data' with a title, x-axis label, and light blue color.

Plotting Normal Distribution

hist(normal_data, main = "Normal Distribution", xlab = "Values", col = "lightgreen")

Creates a histogram of 'normal_data' with a title, x-axis label, and light green color.

Practical 10

Perform the Linear Regression using R/Python.

Linear regression is a way to understand and predict the relationship between two or more variables by fitting a straight line to the data. It helps us see how changes in one variable are associated with changes in another. Linear regression finds a straight line that best fits data points, helping predict one thing based on another. For example, predicting salary based on years of experience.

It's like drawing a straight line through dots on a graph to understand and predict how things are connected.

Dependent variable (y): Salary

Independent variable (x): Years of experience

User ID	Gender	Age	Salary
15624510	Male	19	19000
15810944	Male	35	20000
15668575	Female	26	43000
15603246	Female	27	57000
15804002	Male	19	76000
15728773	Male	27	58000
15598044	Female	27	84000
15694829	Female	32	150000
15600575	Male	25	33000
15727311	Female	35	65000
15570769	Female	26	80000
15606274	Female	26	52000
15746139	Male	20	86000
15704987	Male	32	18000
15628972	Female	18	82000
15697686	Male	29	80000
15704583	Male	46	28000

```
a <- read.csv("D:/r cost pract/emps.csv")

linear_model <- lm(a$Salary ~ a$Age)

summary(linear_model)

plot(a$Age, a$Salary, main = "Salary vs Age", xlab = "Age", ylab = "Salary")

abline(linear_model, col = "blue")
```

y ~ x (where y is the dependent variable and x is the independent variable)

The summary() function will provide information about the regression coefficients, p-values, R-squared value, and other statistics related to the regression model.

The abline() function in R is used to add lines to plots