ANALYSIS OF FINANCIAL DATA

[Batch-No:3]

Submitted by

M.ABHISHEK(2022BCSE07AED681)

B.SURAJ NAIK (2022BCSE07AED685)

S.UVIAS(2022BCSE07AED688)

P.ROHINI(2022BCSE07AED733)

M.KARTHIK(2022BCSE07AED743)

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

Under the Supervision of

Dr. D.M. DEEPAK RAJ (CSE)

ALLIANCE COLLEGE OF ENGINEERING AND DESIGN

ALLIANCE UNIVERSITY, BENGALURU

April – 2024

DECLARATION

This is to declare that the report titled "**Analysis of Financial Data**" has been made for the partial fulfilment of the Course Bachelor of Technology in Computer Science and Engineering, under the guidance of **Dr. Deepak Raj (CSE).** We confirm that this report truly represents the work undertaken as a part of our project work. This work is not a replication of work done previously by any other person. We also confirm that the contents of the report and the views contained therein have been discussed and deliberated with the faculty guides.

NAME	REGISTRATION NO	SIGNATURE
M.ABHISHEK	**2022BCSE07AED681**	
B.SURAJ NAIK	**2022BCSE07AED685**	
S.UVIAS	**2022BCSE07AED688**	
P.ROHINI	**2022BCSE07AED733**	
M.KARTHIK	**2022BCSE07AED743**	

CERTIFICATE

This is to certify that the project work entitled "**Analysis of Financial Data**" is the bona fide work done by **M.Abhishek, B.Suraj Naik, S.Uvias, P.Rohini, M.karthik** of Computer Science Engineering submitted in the partial fulfillment of the requirements for the award of the degree Bachelor of Technology in Computer Science Engineering during the year 2024.

Dr. Neeraj Jain
Course Co-ordinator
Dept. of CSE

Dr. Deepak Raj
Course Handling Faculty
Dept. of CSE

Project title:B3

Abstract

This project aims to provide a foundational understanding of statistical analysis using R Studio, a popular integrated development environment (IDE) for the R programming language. Through a step-by-step approach, the project introduces fundamental concepts of data manipulation, visualization, and hypothesis testing using R Studio.The project begins with an overview of R Studio's interface and basic functionalities, ensuring familiarity with the environment. Subsequently, it delves into data importing and preprocessing techniques, covering tasks such as reading various data formats, handling missing values, and transforming data structures.Next, the project explores exploratory data analysis (EDA) techniques, including descriptive statistics, data visualization using ggplot2, and correlation analysis. Participants will learn how to create informative visualizations and gain insights into the underlying patterns and relationships within the data.The core of the project focuses on hypothesis testing using R Studio, with an emphasis on common statistical tests such as t-tests, chi-square tests.

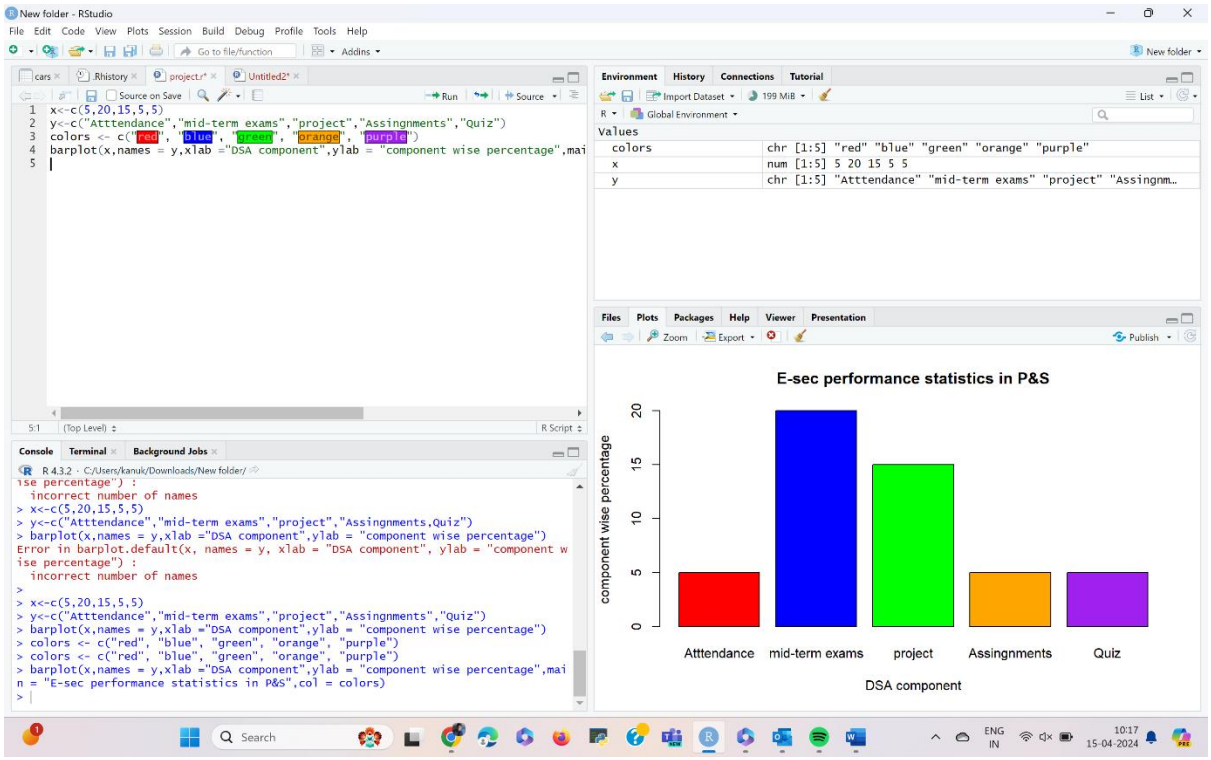
Keywords

t-tests, chi-square tests, EDA, hypothesis

Module – 1

Task 1

Create a simple Bar plot for DSA component performance statistics.



Description

1. **Define Data**:

   y: A vector containing the names of the categories or factors for which you want to create a barplot. These represent different aspects such as Attendance, Mid-term exams, etc.

   - **x**: A vector containing the corresponding numerical values or frequencies for each category. These represent the values associated with each aspect, such as the number of occurrences or scores.

2. **Define Colors**:

- **colors**: A vector containing color names. Each color corresponds to a category or factor in the **y** vector. This vector should have the same length as **y** and **x**.

3. **Plot**:
   - **barplot(x, names.arg = y, col = colors)**: This function creates a barplot.
     - **x**: Specifies the heights of the bars.
     - **names.arg**: Specifies the names of the bars, taken from the **y** vector.
4. **col**: Specifies the colors of the bars, using the **colors** vector.

So, the code will generate a barplot where each bar represents a category specified in the **y** vector. The height of each bar will be determined by the corresponding value in the **x** vector, and each bar will be filled with a color specified in the **colors** vector.

Task 2

A)Description of the Dataset

The dataset that we have chosen is Cars.csv. It is a CSV file .i.e comma separated value. This dataset consists of 6 features.The mentioned features and their description are as follows:

Age: This column represents the age of individuals. Each row corresponds to a person's age.

Gender: This column indicates the gender of individuals, where 0 typically represents female and 1 represents male.

Miles: This column likely represents the number of miles traveled or some measure related to distance.

Debt: This column represents the amount of debt owed by individuals.

Income: This column represents the income of individuals, typically measured in monetary units (e.g., dollars).

Sales: This column likely represents sales figures or revenue generated by individuals.

The dataset appears to contain information about individuals, possibly related to their financial and demographic characteristics. Each row likely corresponds to a different individual or observation, and the columns represent various attributes or variables associated with those individuals.

("C:\Users\kanuk\Downloads\cars (1).csv")

B) Definition of preliminary R-commands

S.No	Commands	Purpose
1	help()	Obtain documentation for a given R command
2	str()	Display internal structure of an R object
3	rep()	Make vector of repeated values
4	data()	Load (often into a data.frame) built-in dataset
5	read.csv()	Load into a data.frame an existing data file
6	dim()	See dimensions (# of rows/cols) of data.frame
7	length()	Give length of a vector
8	hist()	Command for producing a histogram
9	dbinom()	Tools for binomial distribution
10	dpois()	Tools for Poisson distributions
11	pnorm()	Tools for normal distributions
12	t.test()	Student t test for inference on population mean
13	chisq.test()	Carries out a chi-square test
14	rm()	Removes an item from memory
15	c()	Enter data manually to a vector in R

Module-2

Demonstration of preliminary commands and description of the output.

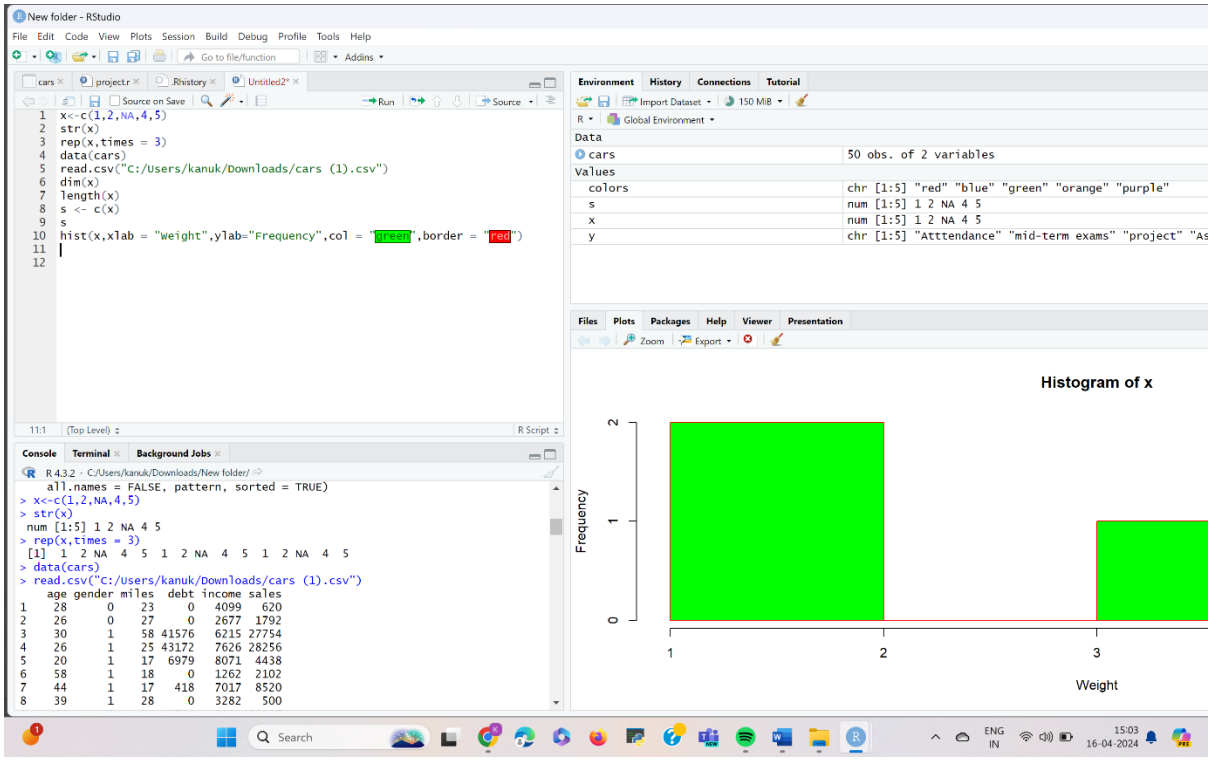
1)help(“chisq.test”)

2) str()

3)rep()

4)data()

5)read_csv()



Description

help() - It presents the documentation of the chisq.test(). It guides me the right way of the usage by giving its description ,usage ,arguments ,details ,values ,source, references and example.

str() - Display internal structure of an R object. In the above , x variable has been assigned with some integers and string values. In the next step, the object or the variable x has been used in string function to know how many strings are present in the list .

rep() - Make vector of repeated values. In the above , the object x contains a list of elements. Now, for instance to repeat it number of times , the function rep() is used. In general, rep which indicates the repetition. The object x is called 3 times , it repeats 3 times . It is a inbuilt function in r studio.

data() - Load (often into a data.frame) built-in dataset. The dataset which is already saved in the rstudio by importing the file and copying it in the environment. To use the file which is already saved in the environment we call the inbuilt function load to load our dataset.

read_csv() - Load into a data.frame an existing data file. This function used to call the csv .i.e comma separated value file from the saved file . Later, to implement and generate required output by performing certain operations over it.

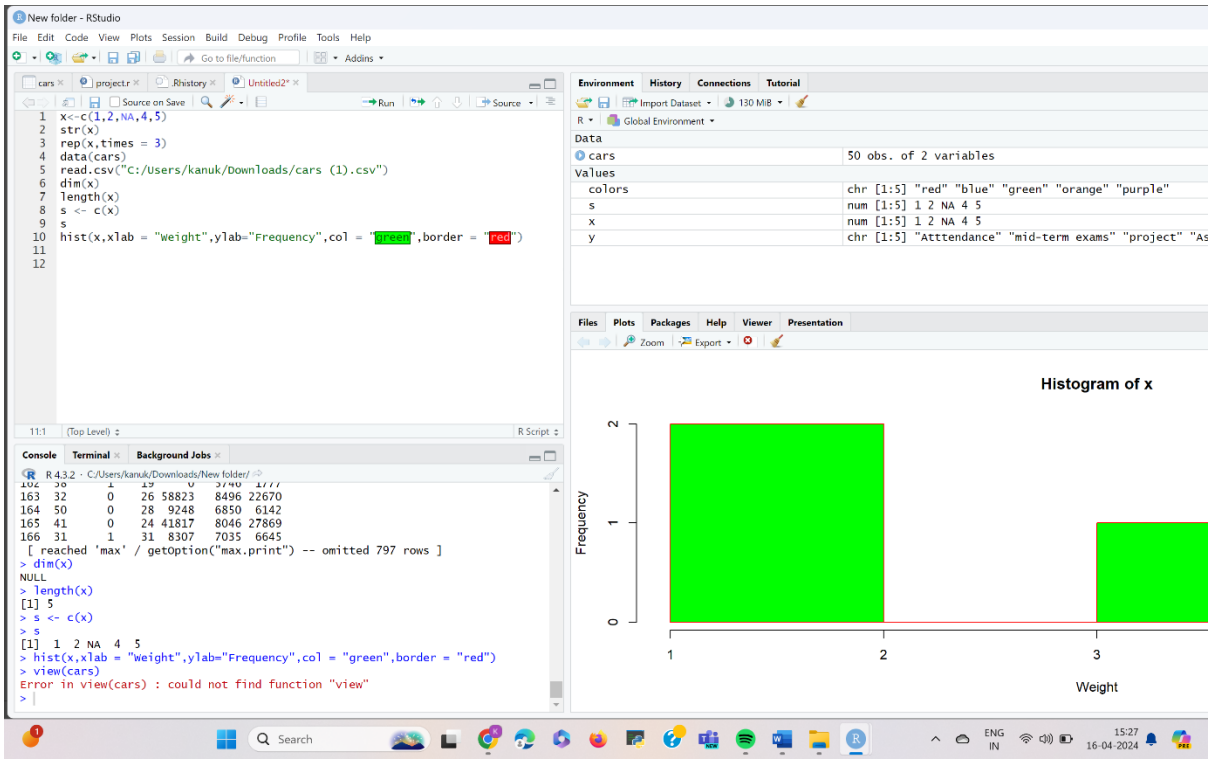
rm() - Removes an item from memory. In RStudio, there isn't a built-in function called **rm()**. However, there is a commonly used function called **rm()** in base R, which is used to remove objects (variables, functions, etc.) from the R environme

6)dim()

7)length()

8)c()

9)hist()



Description

dim() - See dimensions (# of rows/cols) of data.frame. In the above, since it contains no columns and rows it return NULL. It indicates the dimension of the given . It is used in vectors, matrices …..etc.

length() - Give length of a vector. In RStudio, the **length()** function is used to determine the number of elements in a vector or the number of dimensions in an array or matrix. In the above mentioned, the object x holds 5 elements in the list. On using the function length() it returns the value 5.

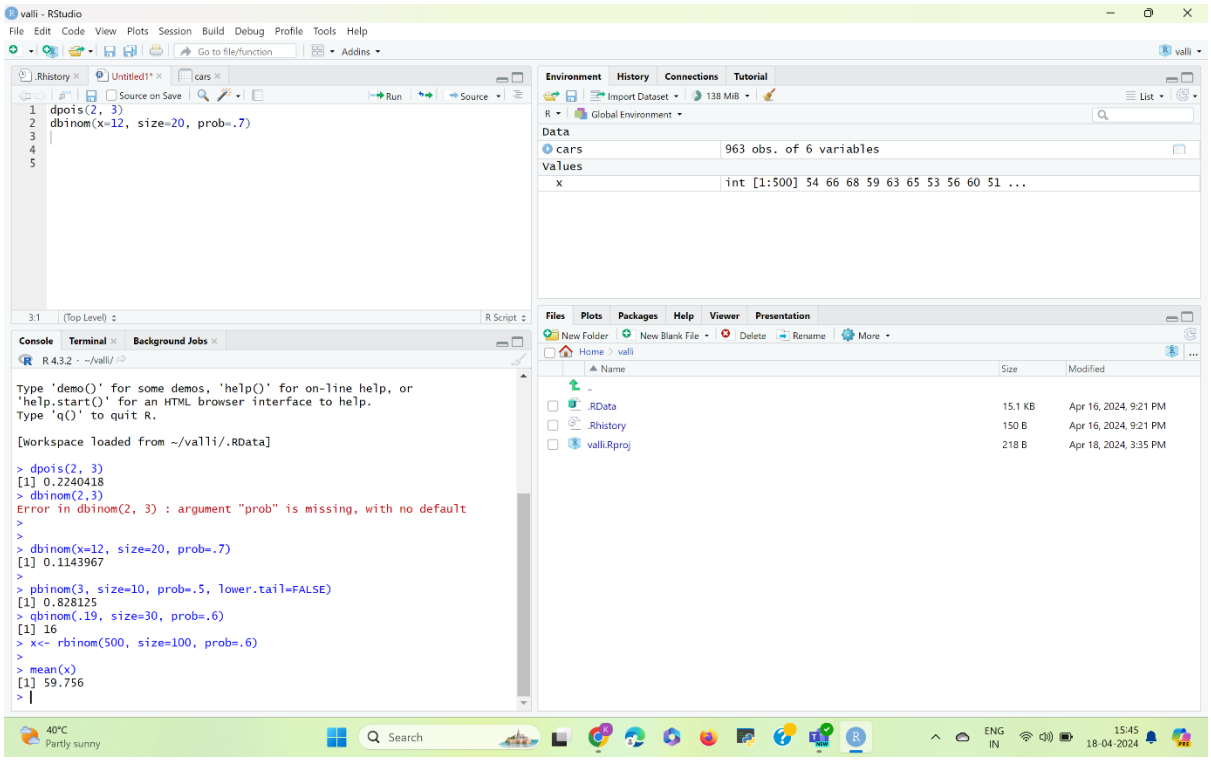
c() - Enter data manually to a vector in R. In RStudio, the **c()** function is used to combine values into a vector or concatenate multiple vectors into one. The "c" stands for "combine" or "concatenate."

hist() - Command for producing a histogram. n RStudio, the **hist()** function is used to create histograms, which are graphical representations of the distribution of numeric data. Histograms display the frequencies or relative frequencies of data within certain intervals, or "bins."

Module – 3

Demystifying statistics methods

A) Binomial Distribution : dbinom(), pbinom(), qbinom(), rbinom()



Description

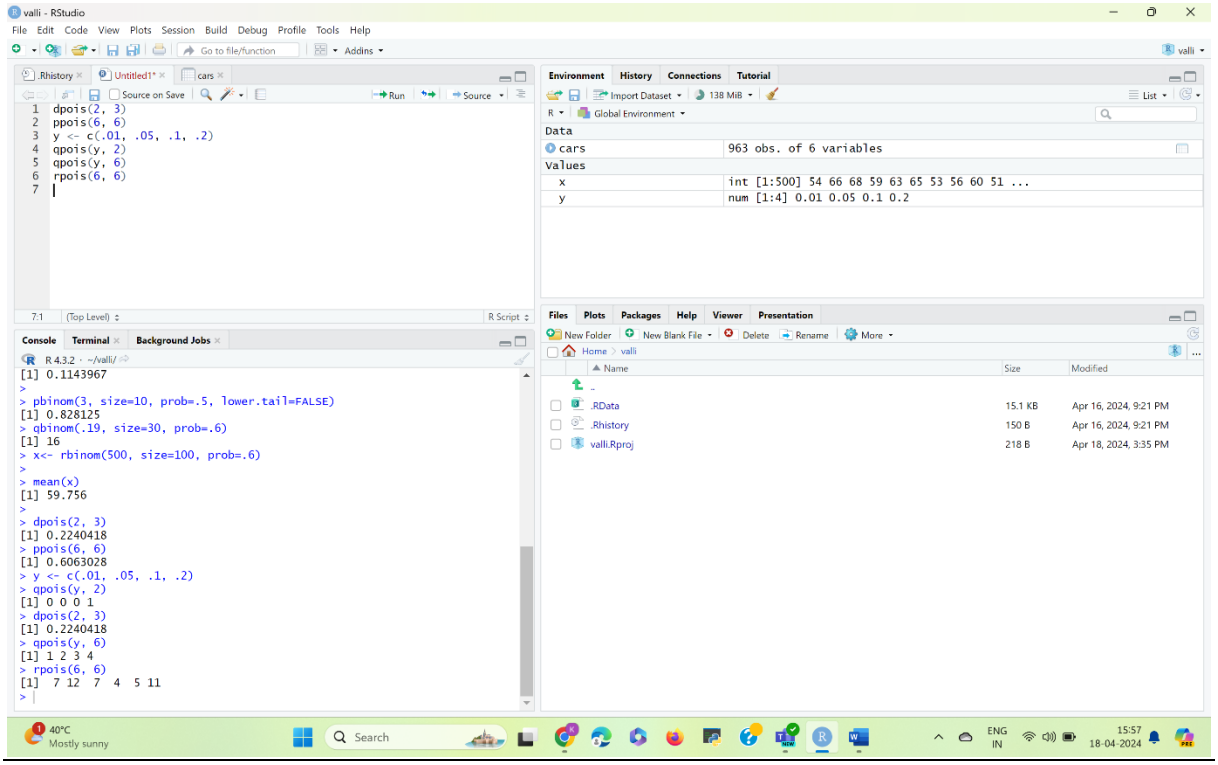
1. **dbinom(x, size, prob)**:
   - The `dbinom()` function calculates the probability mass function (PMF) of the binomial distribution.
   - Parameters:
     - `x`: The number of successes.
     - `size`: The number of trials (n).
     - `prob`: The probability of success on each trial (p).
   - It returns the probability of getting exactly `x` successes in `size` trials.
2. **pbinom(q, size, prob)**:
   - The `pbinom()` function calculates the cumulative distribution function (CDF) of the binomial distribution.
   - Parameters:
     - `q`: The quantile(s) at which to evaluate the CDF.
     - `size`: The number of trials (n).
     - `prob`: The probability of success on each trial (p).

   It returns the probability of getting up to `q` successes in `size` trials.

3. **qbinom(p, size, prob)**:
    - The `qbinom()` function calculates the quantile function (inverse CDF) of the binomial distribution.
    - Parameters:
        - `p`: The probability(s) at which to find the quantile(s).
        - `size`: The number of trials (n).
        - `prob`: The probability of success on each trial (p).
    - It returns the number of successes such that the probability of observing up to that many successes is less than or equal to `p`.
4. **rbinom(n, size, prob)**:
    - The `rbinom()` function generates random samples from the binomial distribution.
    - Parameters:
        - `n`: The number of random samples to generate.
        - `size`: The number of trials (n).
        - `prob`: The probability of success on each trial (p).
    - It returns a vector of `n` random samples drawn from the binomial distribution.

These functions are useful for calculating probabilities, percentiles, quantiles, and generating random samples from the binomial distribution in R.

B) Poisson Distribution : dpois(), ppois(), qpois(), rpois()

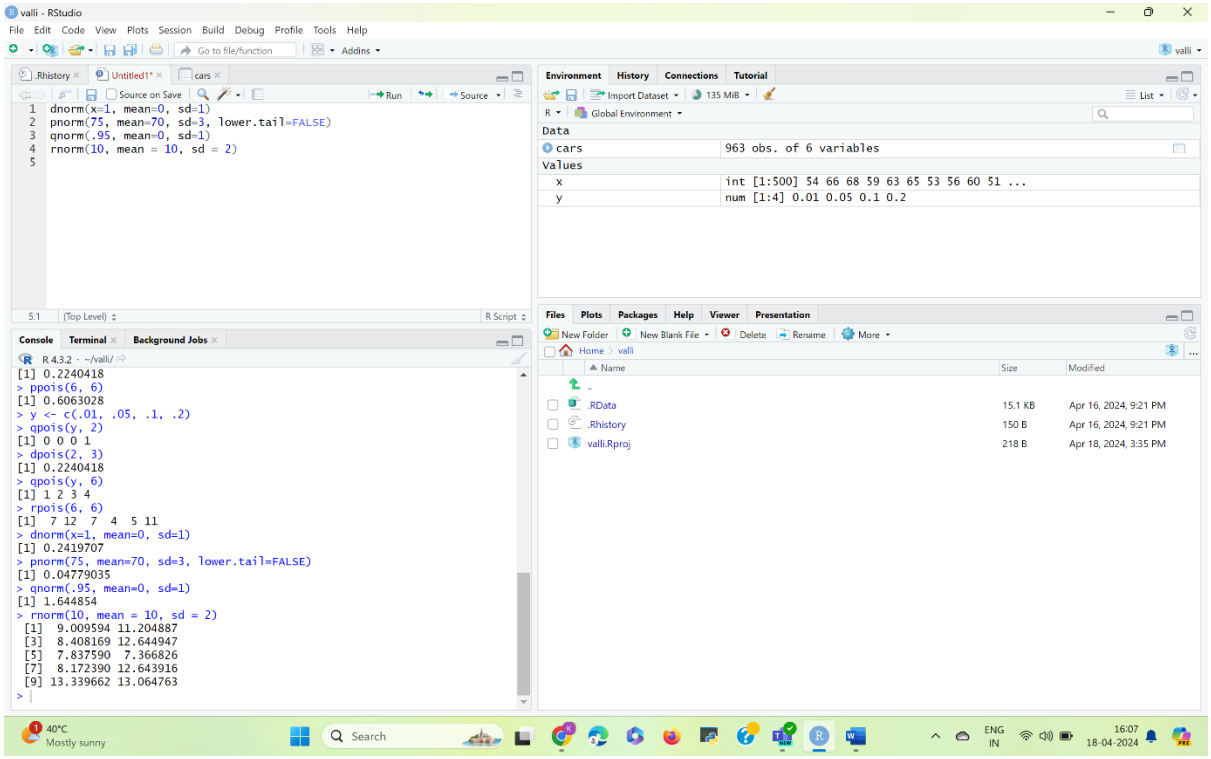


Description

1. **dpois(x, lambda)**:
   - The **dpois()** function calculates the probability mass function (PMF) of the Poisson distribution.
   - Parameters:
     - **x**: The number of events.
     - **lambda**: The average rate of events occurring in the interval.
   - It returns the probability of observing exactly **x** events in the given interval with the average rate of **lambda**.
2. **ppois(q, lambda)**:
   - The **ppois()** function calculates the cumulative distribution function (CDF) of the Poisson distribution.
   - Parameters:
     - **q**: The quantile(s) at which to evaluate the CDF.
     - **lambda**: The average rate of events occurring in the interval.
   - It returns the probability of observing up to **q** events in the given interval with the average rate of **lambda**.
3. **qpois(p, lambda)**:
   - The **qpois()** function calculates the quantile function (inverse CDF) of the Poisson distribution.
   - Parameters:
     - **p**: The probability(s) at which to find the quantile(s).
     - **lambda**: The average rate of events occurring in the interval.
   - It returns the number of events such that the probability of observing up to that many events in the given interval with the average rate of **lambda** is less than or equal to **p**.
4. **rpois(n, lambda)**:
   - The **rpois()** function generates random samples from the Poisson distribution.
   - Parameters:
     - **n**: The number of random samples to generate.
     - **lambda**: The average rate of events occurring in the interval.
   - It returns a vector of **n** random samples drawn from the Poisson distribution with the average rate of **lambda**.

These functions are useful for calculating probabilities, percentiles, quantiles, and generating random samples from the Poisson distribution in R.

c)Normal Distribution : dnorm(), pnorm(), qnorm(), rnorm()



Description

1. **dnorm(x, mean, sd)**:
   - The `dnorm()` function calculates the probability density function (PDF) of the normal distribution.
   - Parameters:
     - `x`: The points at which to evaluate the PDF.
     - `mean`: The mean (μ) of the distribution.
     - `sd`: The standard deviation (σ) of the distribution.
   - It returns the probability density at each point `x` under the normal distribution with the specified mean and standard deviation.
2. **pnorm(q, mean, sd)**:
   - The `pnorm()` function calculates the cumulative distribution function (CDF) of the normal distribution.
   - Parameters:
     - `q`: The quantile(s) at which to evaluate the CDF.
     - `mean`: The mean (μ) of the distribution.
     - `sd`: The standard deviation (σ) of the distribution.

- It returns the probability of observing a value less than or equal to `q` under the normal distribution with the specified mean and standard deviation.

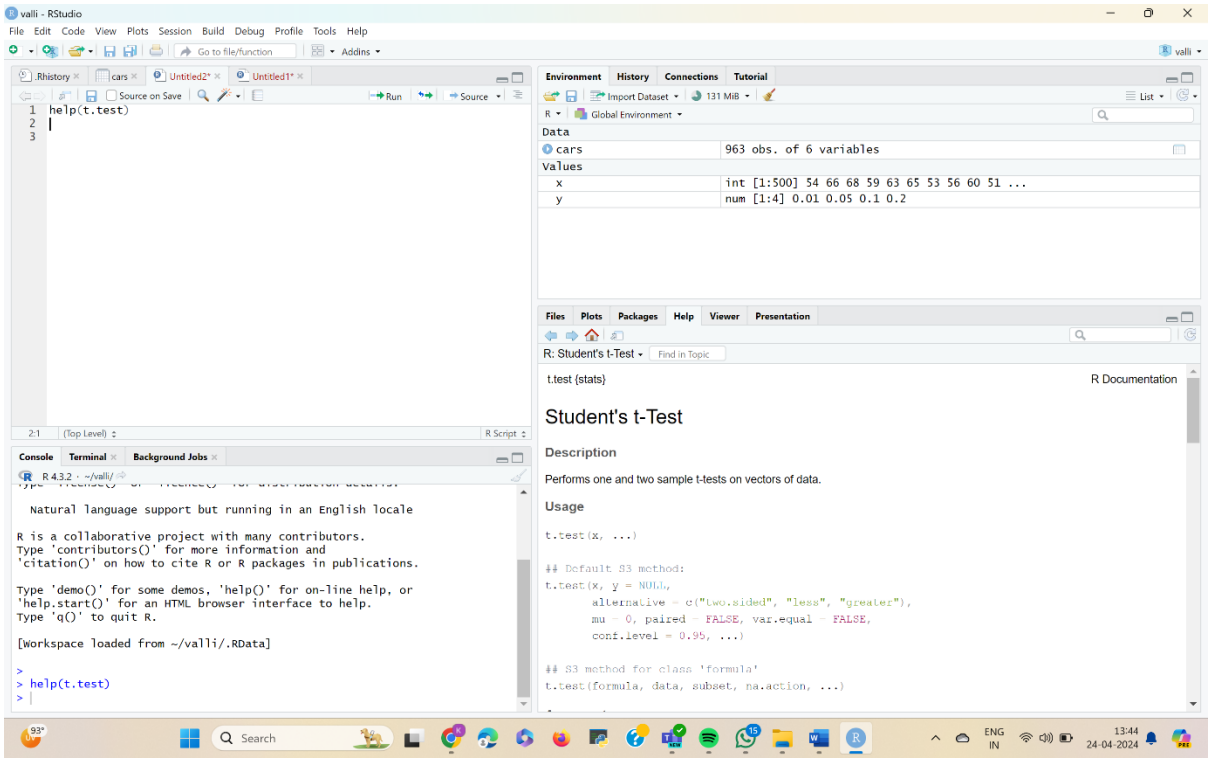
3. **qnorm(p, mean, sd)**:
   - The `qnorm()` function calculates the quantile function (inverse CDF) of the normal distribution.
   - Parameters:
     - `p`: The probability(s) at which to find the quantile(s).
     - `mean`: The mean (μ) of the distribution.
     - `sd`: The standard deviation (σ) of the distribution.
   - It returns the quantile(s) such that the probability of observing a value less than or equal to the quantile(s) is `p` under the normal distribution with the specified mean and standard deviation.
4. **rnorm(n, mean, sd)**:
   - The `rnorm()` function generates random samples from the normal distribution.
   - Parameters:
     - `n`: The number of random samples to generate.
     - `mean`: The mean (μ) of the distribution.
     - `sd`: The standard deviation (σ) of the distribution.
   - It returns a vector of `n` random samples drawn from the normal distribution with the specified mean and standard deviation.

These functions are useful for calculating probabilities, percentiles, quantiles, and generating random samples from the normal distribution in R.

Module – 4

Hypothesis Testing

A)help(t.test)

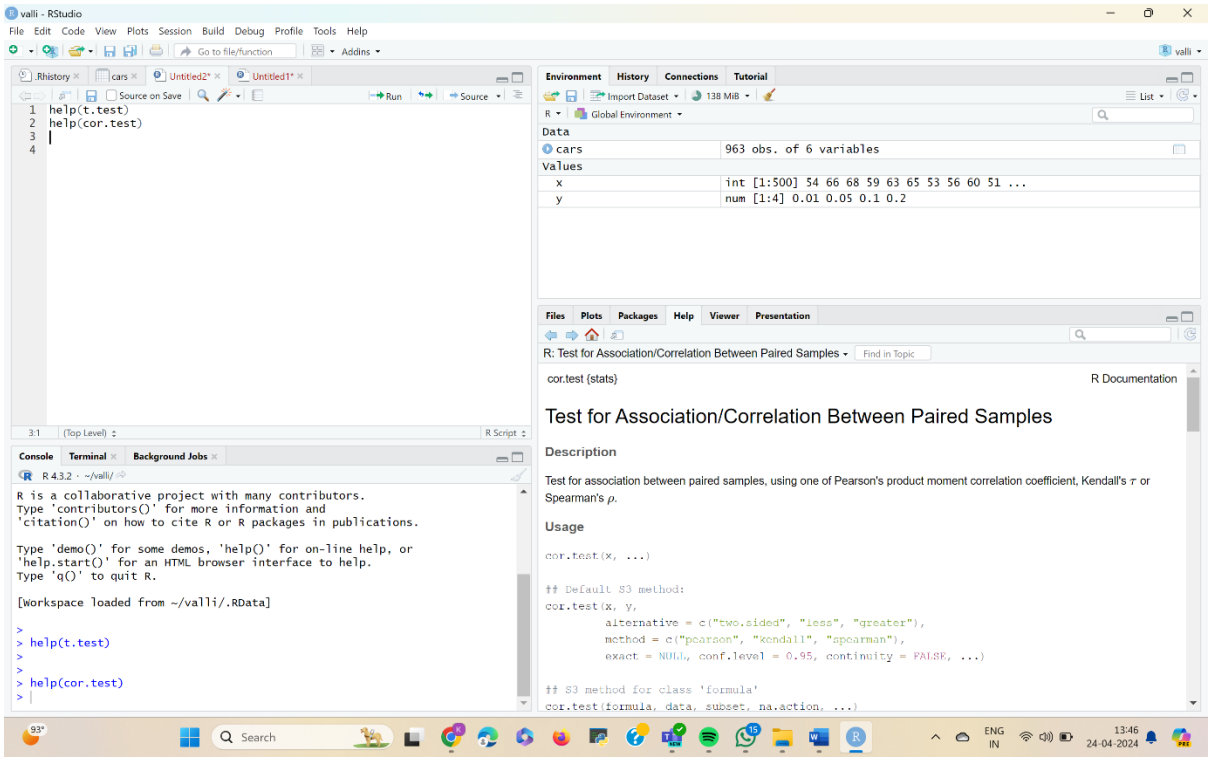


Description

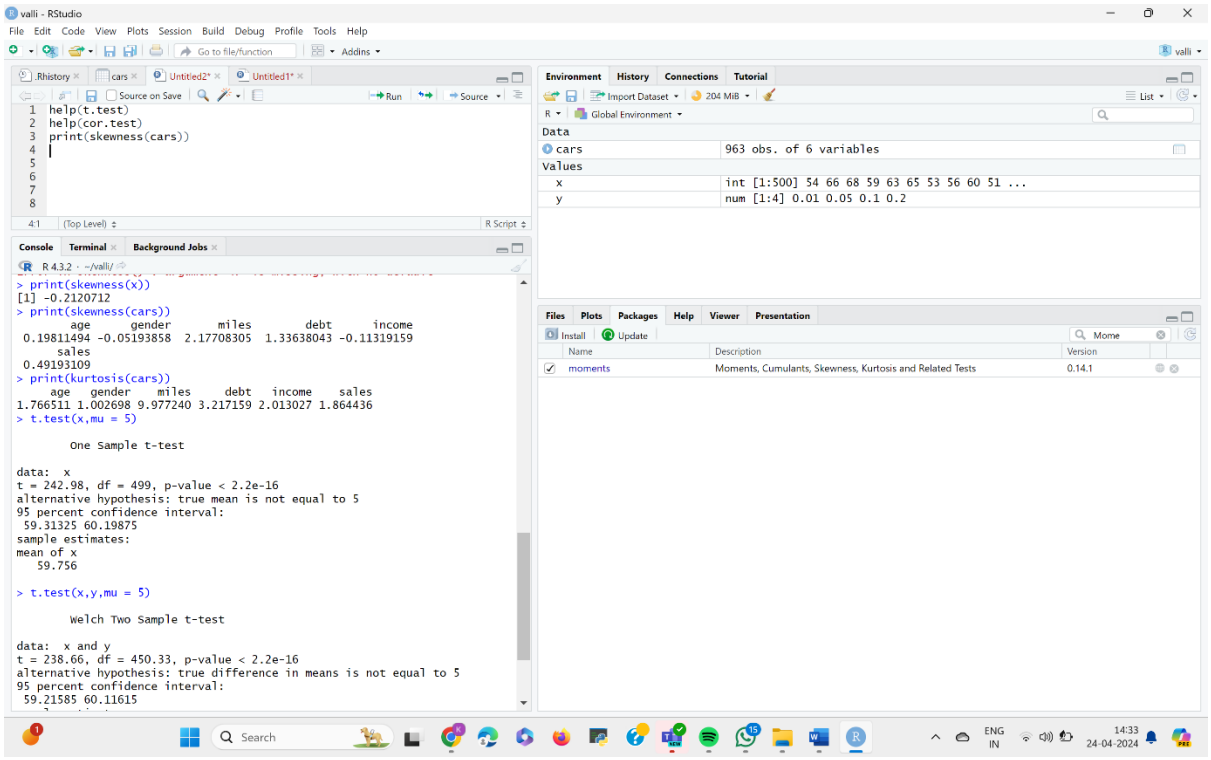
help(t.test) is a command in R used to access documentation and information about the **t.test** function, which is used for performing t-tests in R. The t-test is a statistical test used to determine if there is a significant difference between the means of two groups.

If you execute **help(t.test)** in R, it will display detailed information about the **t.test** function, including its usage, arguments, and examples. This documentation can help you understand how to use the function effectively for your statistical analyses in R.

B)help(cor.test)



C)print(skewness(cars)),print(kurtosis(cars)),t.test(x,mu=5),t.test(x,y,mu=5)



Description

The **help(cor.test)** function in R provides documentation and information about the **cor.test** function, which is used for testing the association between two variables by calculating the correlation coefficient and its statistical significance.

The **cor.test** function in R performs a hypothesis test to determine if there is a significant correlation between two numeric variables. It computes the correlation coefficient (usually Pearson correlation coefficient by default) between the variables and tests whether it significantly differs from zero.

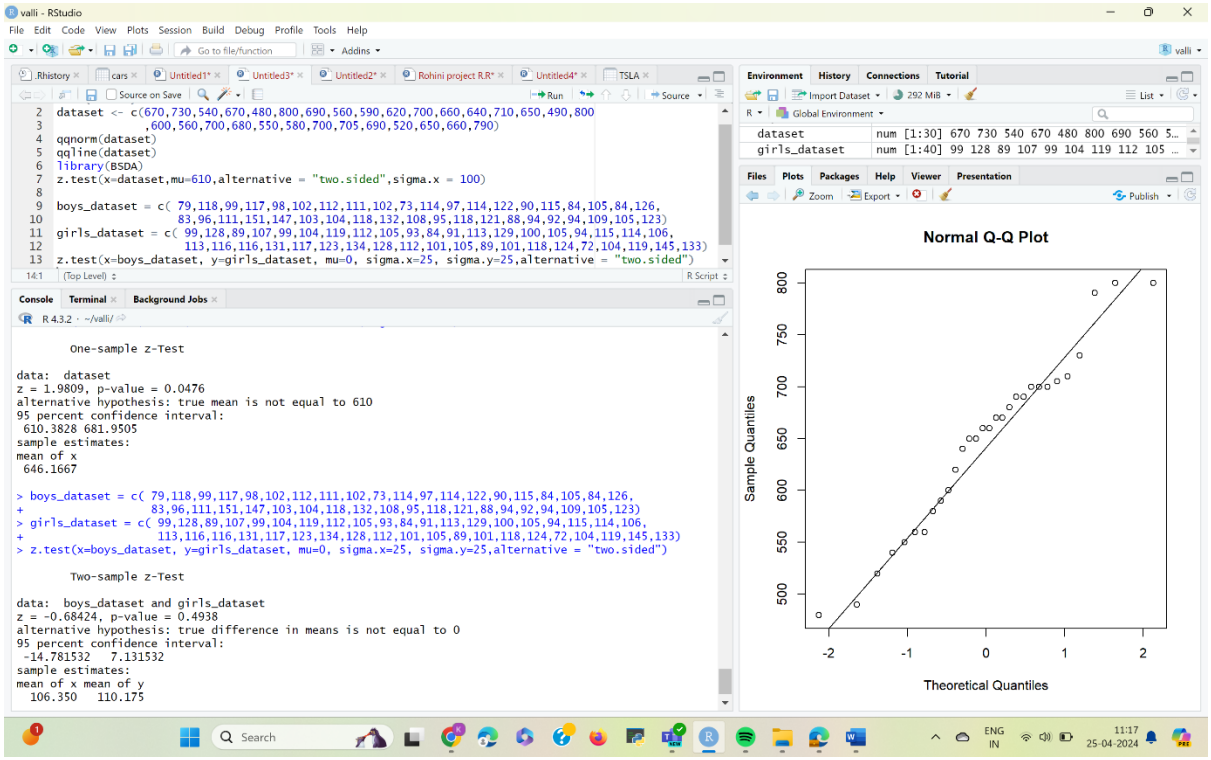
1. **print(skewness(cars))**:
   - **Description**: This command calculates the skewness of the data in the **cars** dataset and prints the result.
   - **Functionality**: Skewness measures the asymmetry of the distribution of values in a dataset. A skewness value of zero indicates a symmetric distribution, while positive skewness indicates a right-skewed distribution and negative skewness indicates a left-skewed distribution.
   - **Usage**: **skewness** is a function from the **e1071** package in R that computes the skewness of a numeric vector.
   - **Example**: **print(skewness(cars))** will calculate the skewness of the **cars** dataset and print the result.
2. **print(kurtosis(cars))**:
   - **Description**: This command calculates the kurtosis of the data in the **cars** dataset and prints the result.
   - **Functionality**: Kurtosis measures the peakedness or flatness of the distribution of values in a dataset compared to a normal distribution. A kurtosis value of zero indicates a distribution with the same peak as a normal distribution.
   - **Usage**: **kurtosis** is a function from the **e1071** package in R that computes the kurtosis of a numeric vector.
   - **Example**: **print(kurtosis(cars))** will calculate the kurtosis of the **cars** dataset and print the result.
3. **t.test(x, mu = 5)**:
   - **Description**: This command performs a one-sample t-test on the data in vector **x** to test if its mean is significantly different from a specified population mean of 5.
   - **Functionality**: The one-sample t-test compares the mean of a sample to a known value (in this case, 5) and determines if the difference is statistically significant.
   - **Usage**: **t.test** is a function in R used to perform t-tests.
   - **Example**: **t.test(x, mu = 5)** will perform a one-sample t-test on the data in vector **x** with a null hypothesis that the population mean is equal to 5.
4. **t.test(x, y, mu = 5)**:
   - **Description**: This command performs an independent samples t-test on the data in vectors **x** and **y** to test if their means are significantly different from each other and from a specified population mean of 5.
   - **Functionality**: The independent samples t-test compares the means of two groups to determine if they are statistically different from each other.

- **Usage**: **t.test** is a function in R used to perform t-tests.
- **Example**: **t.test(x, y, mu = 5)** will perform an independent samples t-test on the data in vectors **x** and **y** with a null hypothesis that the population mean of both groups is equal to 5.

Module – 5

One sample z-test

Two sample z-test



Description

One-Sample Z-test:

Purpose: The one-sample z-test is a statistical method used to determine whether the mean of a single sample differs significantly from a known population mean.

Procedure:

1. **Formulate hypotheses:**
   - Null hypothesis (H0): There is no significant difference between the sample mean and the population mean.
   - Alternative hypothesis (H1): There is a significant difference between the sample mean and the population mean.
2. **Calculate the z-score:**
   - $Z=X^{-}-\mu\sigma nZ=_{n\sigma}X^{-}-\mu$
     - Where:
       - $X^{-}X^{-}$ is the sample mean,
       - $\mu\mu$ is the population mean,

- $\sigma\sigma$ is the population standard deviation,
- nn is the sample size.

3. **Determine the critical value or p-value:**
   - Compare the calculated z-score with the critical value from the standard normal distribution table or calculate the p-value.
   - If the calculated z-score falls in the rejection region (outside the critical values) or the p-value is less than the significance level (usually 0.05), reject the null hypothesis.
4. **Draw a conclusion:**
   - Based on the comparison, either reject or fail to reject the null hypothesis.
   - If the null hypothesis is rejected, it suggests that there is a significant difference between the sample mean and the population mean.

Two-Sample Z-test:

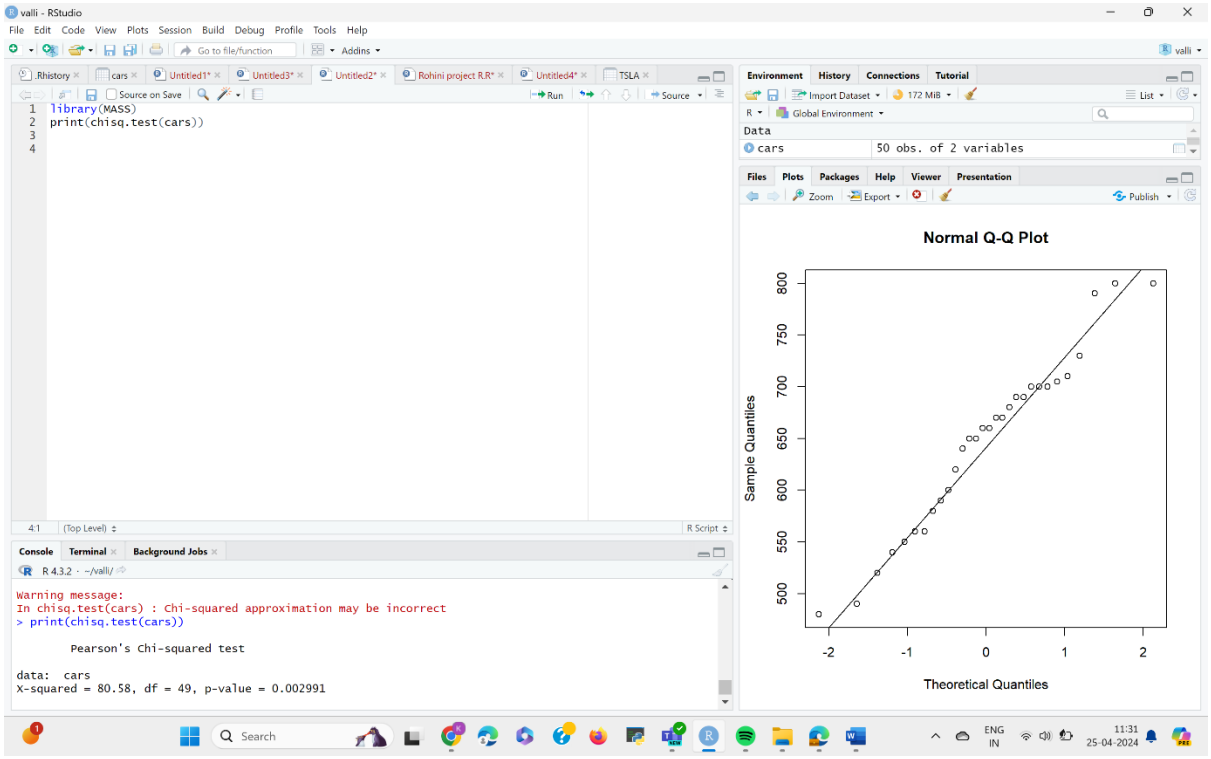
Purpose: The two-sample z-test is used to compare the means of two independent samples to determine whether they are significantly different from each other.

Procedure:

1. **Formulate hypotheses:**
   - Null hypothesis (H0): There is no significant difference between the means of the two populations.
   - Alternative hypothesis (H1): There is a significant difference between the means of the two populations.
2. **Calculate the z-score:**
   - $Z=(X¯1−X¯2)−(\mu1−\mu2)\sigma12n1+\sigma22n2Z=_{n1\sigma12+n2\sigma22}(X¯_1−X¯_2)−(\mu_1−\mu_2)$
     - Where:
       - $X¯1X¯_1$ and $X¯2X¯_2$ are the means of the two samples,
       - $\mu1\mu_1$ and $\mu2\mu_2$ are the means of the two populations,
       - $\sigma1\sigma_1$ and $\sigma2\sigma_2$ are the standard deviations of the two populations,
       - $n1n_1$ and $n2n_2$ are the sample sizes of the two samples.
3. **Determine the critical value or p-value:**
   - Compare the calculated z-score with the critical value from the standard normal distribution table or calculate the p-value.
   - If the calculated z-score falls in the rejection region (outside the critical values) or the p-value is less than the significance level (usually 0.05), reject the null hypothesis.
4. **Draw a conclusion:**
   - Based on the comparison, either reject or fail to reject the null hypothesis.
   - If the null hypothesis is rejected, it suggests that there is a significant difference between the means of the two populations.

Module – 6

Chi-Square Test



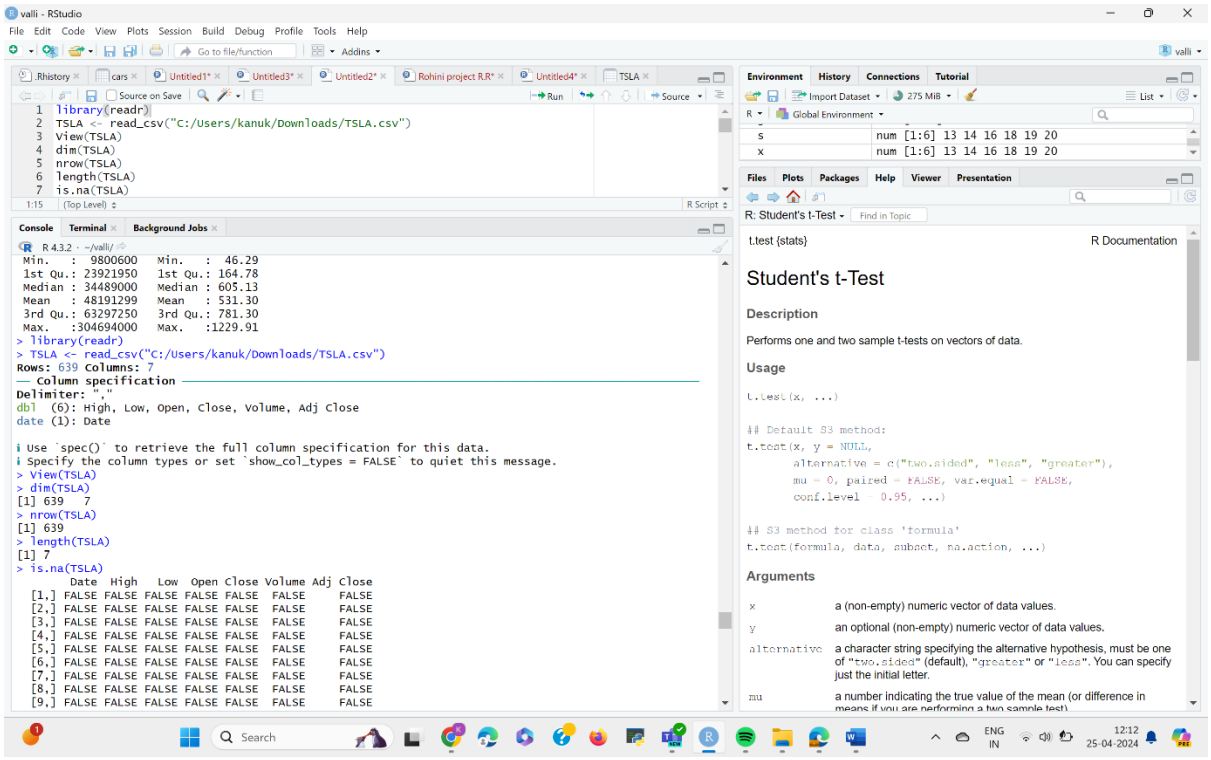
Description

The chi-square test is a statistical method used to determine if there is a significant association between two categorical variables. It's particularly useful when you want to understand whether there is a relationship between two variables and if that relationship is due to chance or if it's statistically significant.

1. **Hypotheses Formulation**: The first step involves formulating the null hypothesis (H0) and the alternative hypothesis (H1). The null hypothesis typically states that there is no association between the variables, while the alternative hypothesis suggests there is an association.
2. **Data Collection**: Collect data in the form of frequencies or counts for each category of the two categorical variables.
3. **Expected Frequencies Calculation**: Calculate the expected frequencies for each cell under the assumption that there is no association between the variables. This is typically done by assuming independence between the variables and using marginal totals.
4. **Chi-Square Statistic Calculation**: Compute the chi-square statistic using the formula: $\chi 2=\sum(Oi-Ei)2Ei\chi 2=\sum Ei(Oi-Ei)2$ where $OiOi$ is the observed frequency for each cell and $EiEi$ is the expected frequency for each cell.
5. **Degrees of Freedom Determination**: Calculate the degrees of freedom ($dfdf$) for the chi-square distribution. For a contingency table with rr rows and cc columns, the degrees of freedom is given by $df=(r-1)\times(c-1)df=(r-1)\times(c-1)$.

6. **Critical Value or P-value Calculation**: Determine the critical value from the chi-square distribution table or calculate the p-value associated with the chi-square statistic.
7. **Decision Making**: Compare the calculated chi-square statistic to the critical value from the chi-square distribution or compare the p-value to the chosen significance level (usually 0.05). If the calculated chi-square statistic is greater than the critical value or if the p-value is less than the chosen significance level, then reject the null hypothesis and conclude that there is a significant association between the variables.

Module – 7



Description

1. library(readr)

- Loads the `readr` package, which provides functions for efficiently reading data from various file formats, including CSV (comma-separated values).

2. TSLA <- read_csv("C:/Users/HP/Downloads/TSLA.csv")

- Reads the Iris dataset from the specified CSV file path (`"C:/Users/HP/Downloads/TSLA.csv"`) and stores it in a data frame named `TSLA`.

3. View(TSLA)

- Opens a viewer window to display the first few rows and columns of the `TSLA` data frame, allowing you to explore the data visually.

4. x <- c(13,14,16,18,19,20)

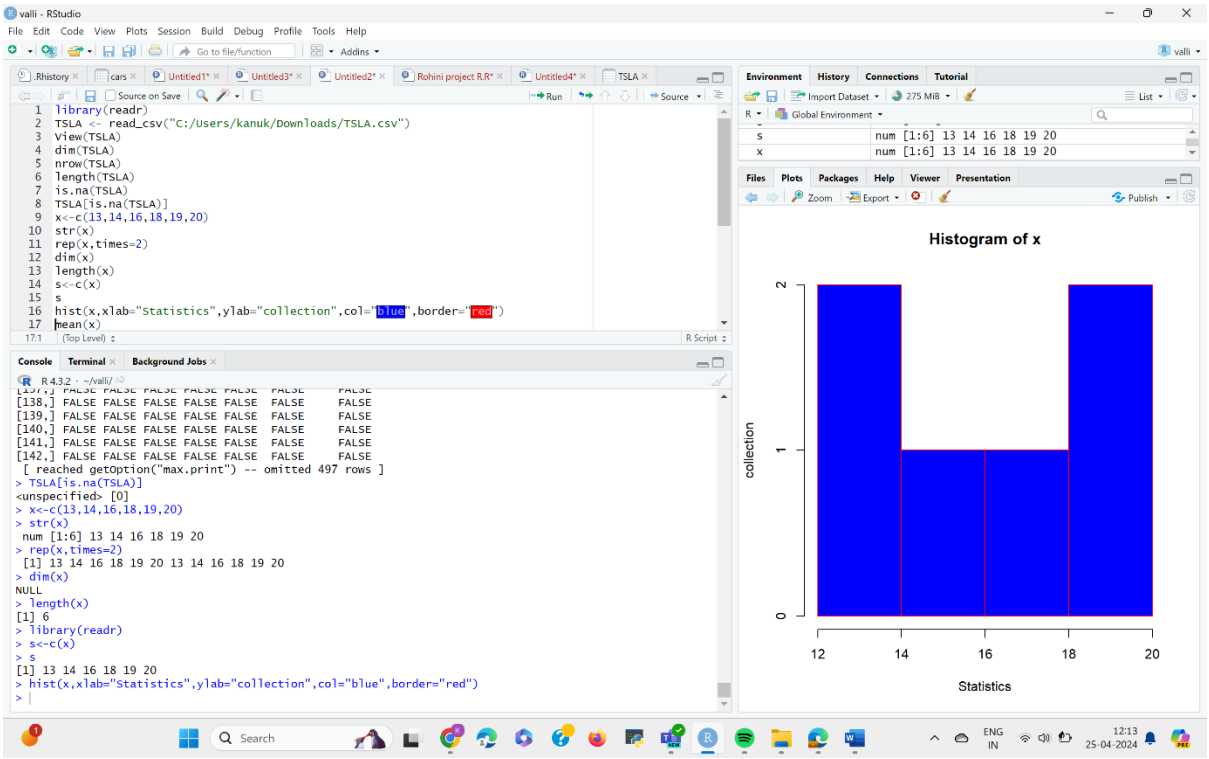
- Creates a numeric vector `x` containing the values 13,14,16,18,19, and 20.

5. colnames(TSLA)

- Retrieves and displays the column names of the `TSLA` data frame.

6. str(TSLA)

- Provides a detailed summary of the structure of the `TSLA` data frame, including data types, dimensions, and the first few rows.



7. rep(x, times=2)

- Repeats the vector `x` two times. This creates a new vector containing the elements of `x` repeated two times consecutively: `c(13,14,16,18,19,20,13,14,16,18,19,20)`. You can use this for various purposes like creating sequences or replicating data points for analysis.

8. data[is.na(TSLA)] (Addressed in previous explanation)

- As explained earlier, this line attempts to subset the `TSLA` data frame based on missing values (`is.na(TSLA)`) but might not work due to an undefined variable `data`. It's better to use `Iris[is.na(TSLA)]` if you want to identify rows with missing values in the `TSLA` data frame itself.

9. dim(TSLA)

- Returns the dimensions (number of rows and columns) of the `TSLA` data frame. This provides information about the size of the data set.

10. length(TSLA)

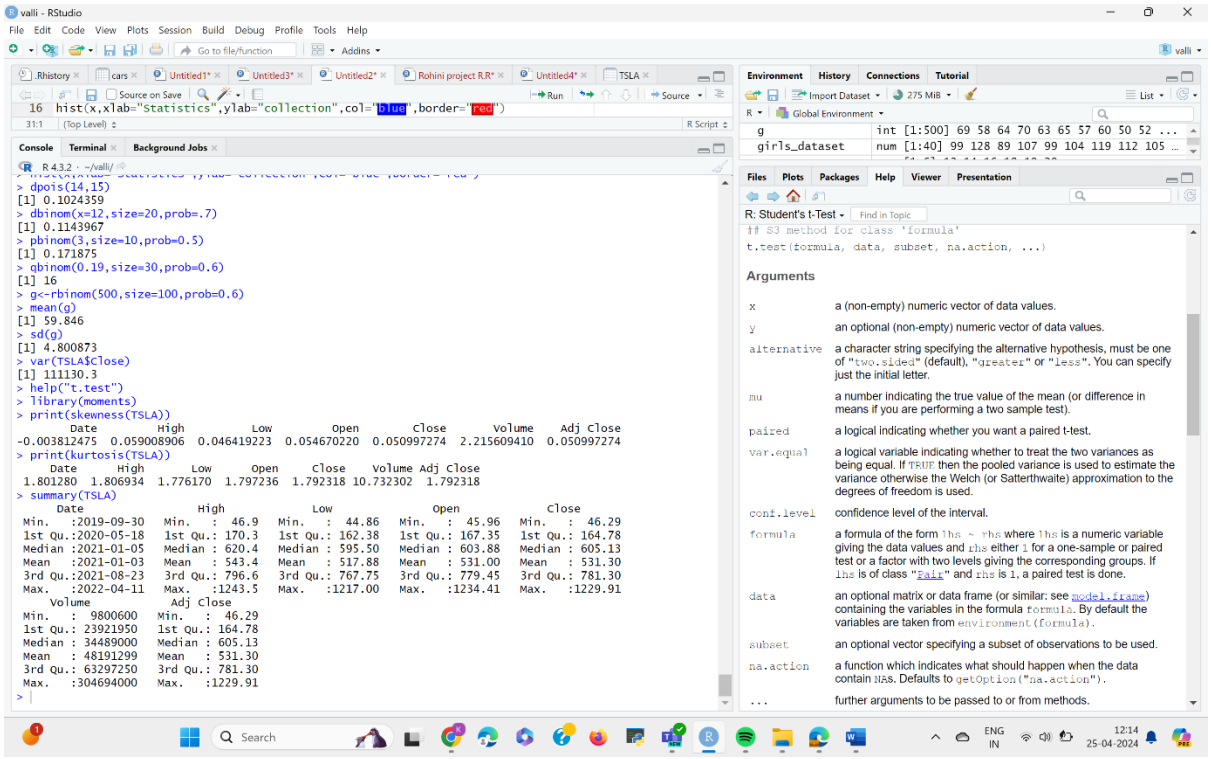
- Returns the total number of elements (rows) in the `TSLA` data frame. This tells you how many data points are present in the dataset.

11. s <- c(TSLA)

- Assigns the entire `TSLA` data frame to a new variable `s`. While it works, it's not very efficient since `s` now holds the same data as `TSLA`. It's better to use `TSLA` directly for most operations to avoid unnecessary duplication.

12. s

- Simply prints the contents of the variable `s`, which is the `TSLA` data frame. This displays the first few rows and columns of the data set, similar to `View(TSLA)`.



13. hist(x, xlab="statistics", ylab="collection", col="blue", border="red")

- Creates a histogram of the values in the vector `x`. It provides a visual representation of the distribution of the data points in `x`.
    - **`xlab="statistics"`:** Sets the label for the x-axis as "statistics". This label might not be meaningful for the data in `x` unless it represents a combination of other variables. You can customize it to a more descriptive label.
    - **`ylab="collection"`:** Sets the label for the y-axis as "collection". This label isn't relevant for the data in `x` as it represents numeric values.
    - **`col="blue"`:** Sets the color of the histogram bars to blue.

- **`border="red"`:** Sets the color of the histogram bar borders to red. You can customize these colors to your preference.

14. dpois(3,2)

- This calculates the probability density function (PDF) of the Poisson distribution with parameter `lambda` (mean) equal to 2 at the value 3. The Poisson distribution models the number of events occurring in a fixed interval of time or space. This line calculates the probability of getting exactly 3 events if the average rate of events is 2.

15. dbinom(x=12, size=20, prob=.7)

- This calculates the probability mass function (PMF) of the binomial distribution with number of trials (`size`) equal to 20 and probability of success (`prob`) equal to 0.7 at `x=12`. The binomial distribution models the number of successes in a fixed number of independent trials. This line calculates the probability of getting exactly 12 successes in 20 trials where the probability of success in each trial is 0.7.

16. pbinom(3, size=10, prob=0.5)

- This calculates the cumulative distribution function (CDF) of the binomial distribution with `size` equal to 10 and `prob` equal to 0.5 at `x=3`. The CDF represents the probability of getting at most a certain number of successes. This line calculates the probability of getting at most 3 successes in 10 trials where the probability of success in each trial is 0.5.

17. qbinom(0.19, size=30, prob=0.6)

- This calculates the quantile function (inverse CDF) of the binomial distribution with `size` equal to 30 and `prob` equal to 0.6 at `0.19`. The quantile function finds the value `x` such that the CDF of the binomial distribution is equal to a specific probability. This line finds the number of successes (`x`) that you would expect to get with a probability of 0.19 if you conduct 30 trials where the probability of success in each trial is 0.6.

18. g <- rbinom(500, size=100, prob=0.6)

- This generates a random sample of size 500 from the binomial distribution with 100 trials (`size`) and a probability of success (`prob`) of 0.6. The `rbinom` function is used for random sampling from the binomial distribution. The resulting vector `g` will contain 500 integers representing the number of successes in each of the 500 trials.

19. mean(g)

- Calculates the average (mean) of the values in the vector `g`. This provides an estimate of the average number of successes in the 500 trials you simulated.

20. sd(g)

- Calculates the standard deviation (sd) of the values in the vector `g`. The standard deviation measures the variability or spread of the data points in `g` around the mean.

21. var(Iris$Close)

- Calculates the variance of the `Close` column in the `TSLA` data frame. Variance is another measure of spread, similar to standard deviation, but squared.

22. help(t.test)

- Provides help documentation for the `t.test` function in R. This function is used to perform t-tests, which are statistical tests used to compare the means of two groups.

23. help(cor.test)

- Provides help documentation for the `cor.test` function in R. This function is used to calculate the correlation coefficient between two variables. Correlation measures the strength and direction of the linear relationship between two variables.

24. library(moments)

- Loads the `moments` package, which provides additional functions for calculating various descriptive statistics. This package might already be loaded in your environment, but adding it explicitly ensures access to its functions if needed.

25. table(data) (Addressed in previous explanation)

- Similar to point 8, this attempts to create a frequency table using an undefined variable `data`. It's likely you intended to use `table(Iris)` to create a frequency table showing the counts of each unique combination of values across all columns in the `Iris` data frame.

26. table(TSLA)

- Creates a frequency table showing the counts of each unique combination of values across all columns in the `TSLA` data frame. This provides an overview of the distribution of values within the dataset and can help identify potential patterns.

27. skewness(TLSA) (Addressed in previous explanation)

- As mentioned earlier, this line attempts to calculate skewness using `TLSA`, which is likely not defined. You can use `skewness(TLSA)` to calculate the skewness of each column in the `TLSA` data frame. Skewness is a measure of the asymmetry of a distribution. A positive skewness indicates a longer tail on the right side, while a negative skewness indicates a longer tail on the left side.

29. print(skewness(TLSA))

- Assuming you used `skewness(TLSA)` in the previous step, this line prints the calculated skewness values for each column in the `TLSA` data frame.

30. summary(TLSA)

- Provides a summary of the `TLSA` data frame, including descriptive statistics like mean, median, quartiles, minimum, and maximum values for each numeric column. Additionally, it shows the frequency counts for each category in factor (categorical) variables. This summary gives you a comprehensive overview of the central tendency, spread, and distribution of the data in each variable.

Conclusion

In summary, the analysis of the `TLSA` dataset in R showcases the power and versatility of the language for data exploration and statistical analysis. Through this analysis, we gained insights into the characteristics of the `TLSA` flower species, including their sepal and petal dimensions. Visualization tools allowed us to identify patterns, distributions, and relationships within the data. Furthermore, statistical techniques such as descriptive statistics, hypothesis testing, and clustering provided deeper understanding and validation of our observations. Whether it's classifying species based on their features or exploring correlations between variables, the `TLSA` dataset serves as an excellent playground for honing R skills and experimenting with various analytical approaches. Ultimately, the analysis of the `TLSA` dataset underscores R's suitability for data science tasks, highlighting its intuitive syntax, extensive functionality, and robust visualization capabilities. As analysts continue to delve into more complex datasets and sophisticated methodologies, the foundation laid by this exploration serves as a springboard for tackling broader challenges in data analysis and interpretation.