DATA ANALYTICS

COURSE CODE: SGB24CA201SE
Skill Enhancement Courses
For FYUG Programmes (Honours)
Self Learning Material

SREENARAYANAGURU
OPEN UNIVERSITY

SREENARAYANAGURU OPEN UNIVERSITY
The State University for Education, Training and Research in Blended Format, Kerala

Vision

To increase access of potential learners of all categories to higher education, research and training, and ensure equity through delivery of high quality processes and outcomes fostering inclusive educational empowerment for social advancement.

Mission

To be benchmarked as a model for conservation and dissemination of knowledge and skill on blended and virtual mode in education, training and research for normal, continuing, and adult learners.

Pathway

Access and Quality define Equity.

Data Analytics

Course Code: SGB24CA201SE

Semester - III

Skill Enhancement Course
For FYUG Programmes (Honours)
Self Learning Material
(With Model Question Paper Sets)

SREENARAYANAGURU
OPEN UNIVERSITY

SREENARAYANAGURU OPEN UNIVERSITY

The State University for Education, Training and Research in Blended Format, Kerala

SREENARAYANAGURU
OPEN UNIVERSITY

DATA ANALYTICS

Course Code: SGB24CA201SE
Semester- III
Skill Enhancement Course
For FYUG Programmes (Honours)

Scan this QR Code for reading the SLM on a digital device.

Edition
October 2025

ISBN 978-81-987966-4-6

www.sgou.ac.in

Visit and Subscribe our Social Media Platforms

Message from Vice Chancellor

Dear Learner,

It is with great pleasure that I welcome you to the Four Year UG Programme offered by Sreenarayanaguru Open University.

Established in September 2020, our university aims to provide high-quality higher education through open and distance learning. Our guiding principle, 'access and quality define equity', shapes our approach to education. We are committed to maintaining the highest standards in our academic offerings.

Our university proudly bears the name of Sreenarayanaguru, a prominent Renaissance thinker of modern India. His philosophy of social reform and educational empowerment serves as a constant reminder of our dedication to excellence in all our academic pursuits.

The University is committed to fostering a future-ready learning environment that emphasizes both knowledge and practical skill development. As part of the FYUG programme, the Skill Enhancement Elective Course Data Analytics introduces learners to the principles and practices of understanding and interpreting data in meaningful ways. Designed for learners from all academic disciplines, this course offers a hands-on introduction to data-driven thinking, focusing on data collection, organization, visualization, and basic analytical techniques..

Our teaching methodology combines three key elements: Self Learning Material, Classroom Counselling, and Virtual modes. This blended approach aims to provide a rich and engaging learning experience, overcoming the limitations often associated with distance education. We are confident that this programme will enhance your understanding of statistical methods in business contexts, preparing you for various career paths and further academic pursuits.

Our learner support services are always available to address any concerns you may have during your time with us. We encourage you to reach out with any questions or feedback regarding the programme.

We wish you success in your academic journey with Sreenarayanaguru Open University.

Best regards,

Dr. Jagathy Raj V.P.
Vice Chancellor

01-10-2025

Contents

BLOCK 1

Mathematical Foundations of Data Analytics

Statistical Foundations

Learning Outcomes

After completing this unit, the learner will able to:

- to familiarise measures of central tendency
- to provide an overview of measures of dispersion
- to get awareness on outliers and normalisation
- to introduce statistical functions in the context of data analytics

Prerequisites

Consider a retail company using data analytics to enhance its business strategies. By analysing customer purchasing patterns - visits how frequent, what items are purchased, the company identifies a significant increase in online sales during specific promotional periods.

Statistical measures such as mean and standard deviation help pinpoint the most successful promotions and the products that resonate most with customers. With these insights, the company can make data-driven decisions, allocating marketing resources to focus on the most effective campaigns and adjusting inventory levels accordingly. Also, the company employs statistical measures to assess customer satisfaction through post-purchase surveys. The data reveals a correlation between positive feedback and customer loyalty. By using this information, the company optimises its customer retention strategies, tailoring promotional offers and loyalty programs to align with the preferences of satisfied customers.

In summary, through the application of statistical measures, the retail company derives actionable insights by identifying successful promotions, supports decision-making processes by allocating resources effectively, and optimises business strategies by enhancing customer satisfaction and loyalty.

Keywords

Mean, Median, Mode, Population, sample, outlier, Normalisation

Discussion

Statistical measures serve a crucial role in data analytics by providing valuable insights and summaries essential for understanding and interpreting complex datasets. These measures, including mean, median, mode, standard deviation, and others, offer a snapshot of the central tendencies, variations, and distributions within the data. They are used to describe patterns, relationships, and anomalies, aiding analysts in recognizing trends and making informed decisions. Statistical measures are fundamental for hypothesis testing, allowing the validation of assumptions and drawing conclusions about populations based on sample data. Moreover, they support data cleaning and quality assessment, helping identify outliers and ensuring the reliability of datasets. In predictive analytics, statistical measures contribute to model building, performance evaluation, and risk assessment, enabling organisations to make strategic decisions and manage uncertainties effectively.

1.1.1 Mean, Median and Mode

Mean, median, and mode are statistical measures that describe the features of data distributions. Statistical distributions fall into two main categories: one involving discrete random variables, where each term has a distinct numerical value, and the other involving continuous random variables, where data can assume an infinite number of values within an uninterrupted range, known as a probability density function. For IT professionals, a grasp of the meanings of mean, median, mode is essential for tasks such as capacity planning, load balancing, system management, maintenance, and issue troubleshooting. Additionally, familiarity with statistical terms holds significance in the expanding domain of data science.

1.1.1.1 Mean

Let us walk through a real-world example to demonstrate how to find the mean.

Consider a dataset representing the monthly salaries of 5 employees at a company: 3,000, 3,500, 4,000, 4,500, and 5,000.

Step 1: Add up all the salaries: 3,000 + 3,500 + 4,000 + 4,500 + 5,000 = 20,000.

Step 2: Count the total number of salaries in the dataset, which is 5.

Step 3: Divide the sum of the salaries by the total number of salaries: 20,000 ÷ 5 = 4,000.

Therefore, the mean monthly salary of the employees is 4,000.

The mean is determined as the average of a provided set of values. Mean serves as a gauge of Central tendency. Central tendency is an indicator that recognizes the distribution or set of data from a specific value. Hence, it can be concluded that the mean effectively characterises the entire dataset. In statistical terms, calculating the mean involves dividing the sum of observations by the total number of observations.

In a distribution with discrete random variables, the mean is the average, which is found by adding all the terms and dividing the sum by the total number of terms.

For distributions with continuous random variables, called the expected value, the mean is calculated by integrating the variable multiplied by its probability. This value is denoted by the symbol "μ".

Let's say the dataset given is X = {x1, x2, x3, . . ., xn} the mean is denoted as $\bar{x}$.

The mean of this dataset is also denoted as μ .

Mean Formula

The mean formula in statistics is defined as the sum of all observations in the given dataset divided by the total number of observations. We can illustrate it by using an example shown below.

Finding the mean of odd natural numbers up to 10 = 1, 3, 5, 7, 9

Sum of first odd natural numbers up to 10 = (1 + 3 + 5 + 7 + 9)

Number of odd natural numbers up to 10 = 5

Mean = Sum of 10 odd natural numbers/5

⇒ Mean = (1 + 3 + 5 + 7 + 9)/5

⇒ Mean = 25/5

⇒ Mean = 5

Sample Code:

```
# Python program for finding mean

# Importing the statistics module

import statistics

# list of positive integer numbers

mean_data = [1, 3, 4, 5, 7, 9, 2]

p = statistics.mean(mean_data)

# Printing the mean

Print("Mean is :", p)
```

Output:

```
Mean is : 4.428571428571429
```

1.1.1.2 Median

Let's consider a real example to illustrate how to find the median. Suppose we have a dataset representing the ages of 9 individuals: 20, 25, 30, 35, 40, 45, 50, 55, and 60.

Step 1: Arrange the data in ascending order: 20, 25, 30, 35, 40, 45, 50, 55, 60.

Step 2: Since there are 9 values in the dataset (an odd number), the median is the middle value. In this case, the middle value is the fifth value, which is 40. Therefore, the median of this group of values is 40.

The median in a distribution of discrete random variables comes on the basis of whether the count of terms in the distribution is even or odd. In cases where the term count is odd, the median corresponds to the value positioned in the middle. ensuring an equal distribution of terms greater than or equal to it and terms less than or equal to it. Conversely, if the term count is even, the median is the average of the two middle terms.

For distributions with a continuous random variable, the median is denoted by the value 'm.' This value signifies that there is a probability of at least 1/2 (50%) for a randomly selected point on the function to be less than or equal to 'm,' and a probability of at least 1/2 for a randomly selected point to be greater than or equal to 'm.'

Sample Code:

```
# Python program for finding median()

# Importing the statistics module

import statistics

median_data = [1, 3, 4, 3, 7, 9, 2, 3, 7, 8, 5]

x = statistics.median(median_data)

# Printing the median

print("Median is :", x)
```

Output:

```
Median is : 4
```

1.1.1.3 Mode

Consider a dataset representing the denomination of currency received by a water authority cash collecting staff over a day is 50, 100, 200, 50, 500, 500, 50, 200, 500, 100, 500, 500, 100 and 50.

Step 1: Count the frequency of each denomination of currency in the dataset:

- Number of 50 Rs. currency received: 4
- Number of 100 Rs. currency received: 3
- Number of 200 RS. currency received: 2
- Number of 500 Rs. currency received: 5

Step 2: Identify the value with the highest frequency, which is 5.

Therefore, the mode of the dataset is 500, indicating that 500 Rs. was the most common number among the received currency.

The mode in a distribution featuring a discrete random variable is the value of the term that appears most frequently. It's not unusual for such a distribution to exhibit more than one mode, especially when the number of terms is limited. This occurs when two or more terms occur with equal frequency, surpassing the frequency of any other terms.

A distribution characterised by two modes is termed bimodal, while a distribution with three modes is referred to as trimodal. In distributions with continuous random variables, the mode is identified as the maximum value of the function. Similar to discrete distributions, there might be multiple modes in a continuous distribution.

Sample Code:

```
# Python code for finding mode

import statistics

mode_data =[1, 3, 2, 2, 3, 3, 3, 4, 4, 7]

# Printing out mode of given data

print("Mode of given data set is % s" % (statistics.mode(mode_data)))
```

Output:

```
Mode of given data set is 3
```

1.1.2 Range, Skewness and Population

1.1.2.1 Range

Consider a dataset representing the ages of 10 individuals: 20, 25, 30, 35, 40, 45, 50,55, 60, and 65.

Step 1: Identify the highest and lowest values in the dataset. The highest value is 65, and the lowest value is 20.

Step 2: Subtract the lowest value from the highest value: 65 - 20 = 45.

Therefore, the range of ages in this dataset is 45 years.

From the above example the concept of range in a distribution involving a discrete random variable represents the gap between the maximum and minimum values.

In the case of a distribution with a continuous random variable, the range is defined as the difference between the two endpoints on the distribution curve, where the function's value drops to zero. Any value beyond the established range in a distribution results in a function value of zero.

The range function in python can also be used for generating values within the range ie., from start value to end value.

Sample Code of Python range (start, stop)

In this example, we are printing the number from 2 to 19. We are using the range function in which we are passing the starting and stopping points of the loop.

```
# printing a natural number from 2 to 20

for i in nge(2, 20):

print(i, end=" ")
```

Output:

```
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
```

1.1.2.2 Skewness

Let's demonstrate how to find skewness with an example. Suppose we have a dataset representing the daily travelling time (in minutes) of 20 employees at a company: 30, 32, 33, 35, 35, 36, 38, 40, 40, 41, 42, 42, 45, 45, 46, 47, 48, 50, 52, 55.

Step 1: Compute the mean and standard deviation of the dataset. Let's assume they are 41.6 minutes and 7.8 minutes, respectively.

Step 2: Use the formula for skewness:

$$\text{Skewness} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3}{\sigma^3}$$

After calculation, xi is each value in the dataset, $\underline{x}$ is the mean, σ is the standard deviation, and n is the number of observations.

Let's say we find the skewness to be -0.35.

Based on the computed skewness:

- If the skewness is approximately zero, the distribution is nearly symmetrical.
- If the skewness is negative, it indicates a left-skewed distribution, implying that most values are concentrated on the right side.
- If the skewness is positive, it suggests a right-skewed distribution, with most values concentrated on the left side.

In our example, with a skewness of -0.35, it indicates a slight left skew, suggesting that the majority of employees have shorter travelling times.

Skewness in statistics measures the asymmetry of the probability distribution of a real-valued random variable about its mean. A dataset can be positively skewed (tail to the right), negatively skewed (tail to the left), or have zero skewness (symmetrical distribution).

To calculate skewness, initially need to compute the mean and standard deviation of the dataset. Then, apply the formula for skewness, which involves the third moment about the mean. A positive skewness indicates that the right tail of the distribution is longer or fatter than the left tail, while a negative skewness indicates the opposite.

Skewness is used to quantify the extent of asymmetry in a graph, indicating the degree to which our data deviates from the standard symmetrical pattern. Occasionally, the normal distribution exhibits a leaning tendency on one side due to a higher probability of data being either more or less than the mean, resulting in an asymmetrical distribution. This implies an uneven distribution of data. Skewness can manifest in two forms:

Positive Skewness: In a positively skewed distribution, values are more concentrated on the right side, with the left tail extending further as in Fig 1.1.1. Consequently, statistical measures lean towards the left side, with the mean, median, and mode all being positive. In this distribution, the order is Mean > Median > Mode.

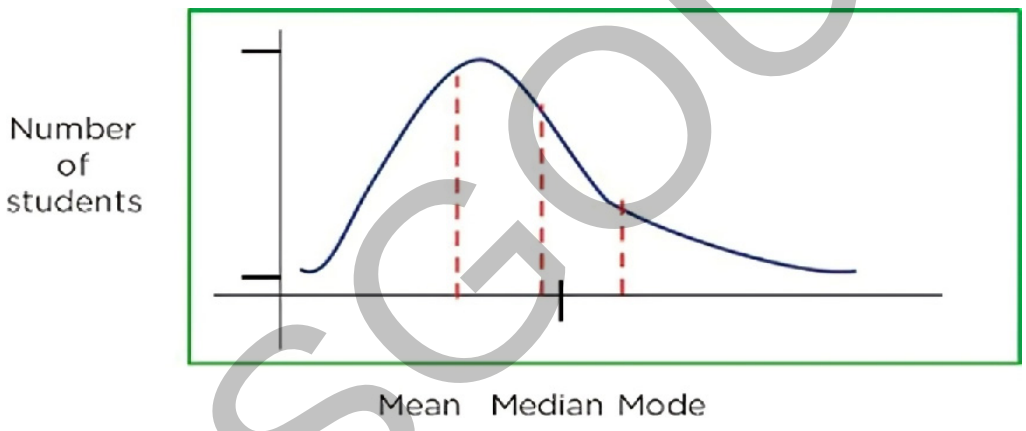


Fig. 1.1.1 Positively Skewed

Negatively Skewed: In a distribution with negative skewness, data points are more densely clustered on the right side as in Fig 1.1.2. Consequently, the mean, median, and mode shift towards the right, resulting in negative values for these measures. In this type of distribution, the order is Mode>Median>Mean.

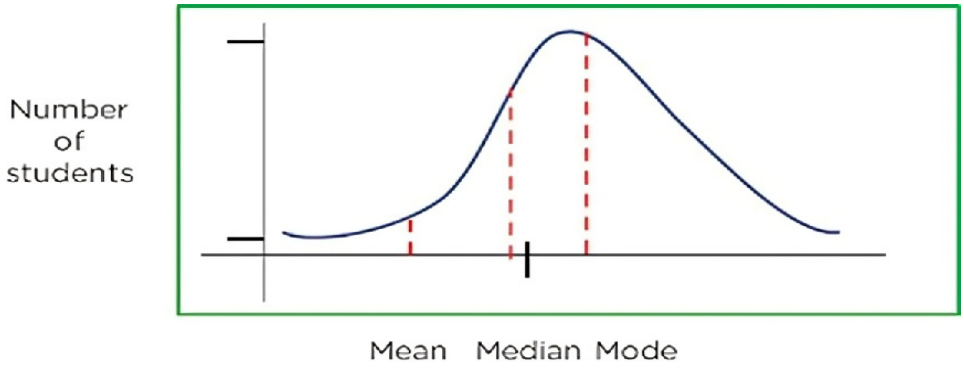


Fig. 1.1.2 Negatively Skewed

Sample Code:

1. Use skew() function to find the skewness of the data over the column axis.

```
# importing pandas as pd
import pandas as pd
# Creating the dataframe
df = pd.read_csv("nba.csv")
# skip the na values
# find skewness in each row
df.skew(axis = 1, skipna = True)
```

2. Use skew() function to find the skewness in data over the index axis.

```
# importing pandas as pd
import pandas as pd
# Creating the dataframe
df = pd.read_csv("nba.csv")
```

1.1.2.3 Population

Suppose a researcher is interested in studying the average height of all adults living in a particular city. In this case, the population would consist of all adults (age 18 and above) residing in that city. The population includes every adult resident, regardless of age, gender, occupation, or any other characteristic.

Once the population is clearly defined, the researcher can collect data from a sample—a subset of the population—to make inferences and draw conclusions about the entire population. However, it's important to note that in many cases, it may not be feasible or practical to study an entire population due to factors such as time, cost, and logistics. In such situations, researchers often use sampling techniques to study a representative sample of the population and generalise the findings to the larger population.

A population encompasses the entirety of the group under study for drawing conclusions. Conversely, a sample represents a particular subset from which data will be gathered, invariably smaller in size compared to the entire population.

A statistical population could refer to various groups depending on the context of the study.

1.1.3 Shifting and Scaling

1.1.3.1 Shifting

Let us illustrate shifting with a real-life example. Suppose you have a dataset representing the monthly temperatures (in degrees Celsius) recorded at a weather station over the course of a year. However, the temperature sensor used at the station is known to consistently measure temperatures 2 degree Celsius higher than the actual temperatures.

To correct for this known offset, you would shift the entire dataset by subtracting 2 from each recorded temperature. For instance, if the original dataset includes temperatures of 22°C, 25°C, 20°C, and 18°C, after shifting by -2°C, the adjusted temperatures would be 20°C, 23°C, 18°C, and 16°C, respectively.

This adjustment effectively aligns the dataset with the true temperatures, compensating for the systematic error introduced by the sensor. Shifting in this manner ensures that the analysis and interpretation of the data are based on accurate measurements, leading to more reliable results. Shifting data involves adding a real number, denoted as 'k', to every element within a dataset. In visual representation, this action entails elevating the entire distribution of data points by shifting them a distance of k.

"Shifting" typically refers to a transformation or adjustment made to a dataset, often by adding or subtracting a constant value to each data point. This process shifts the entire distribution of the data along the number line without changing its shape or spread. Shifting can be useful for various reasons, such as centering the data around a specific value or adjusting for a known offset.

To find shifting in statistics, you need to determine the constant value by which the dataset is being shifted and then apply this transformation to the data. The shift can be positive (adding a constant) or negative (subtracting a constant).

Shifting Data and the Mean & Median

When data is shifted the mean(μ) and median are shifted by the constant k.

That is to say, when you shift a data set by k, so that $f(x)= x+k$ for every x in your data set,

$f(\mu)= \mu + k$

$f(med)= med + k$

Note that k can be either positive or negative.

Measures of Spread and Relative Standing

Your standard deviation, variance, z scores and percentile values all remain unchanged when your data set is shifted. Since every point in your data set moves the exact same distance, there is no change in their relations to each other.

For a shifting operation the k= -5. The values of mean, median, Standard deviation, Variance and Z Score are as follows.

If your mean was 41 before the shift, it is now 36.

If your median was 28, it is now 23.

If your standard deviation was 16, it is still 16.

Your variance will stay the same, as will your z score.

1.1.3.2 Scaling of data

Consider a scenario where two teachers are grading their students' assignments, but each teacher uses a different scale for assigning grades.

Teacher A uses a scale where the highest possible score on an assignment is 100, while Teacher B uses a scale where the highest possible score is 50. Now, suppose a student receives a score of 80 from Teacher A and a score of 40 from Teacher B on the same assignment.

To compare the students' scores accurately and fairly, it's necessary to scale the scores to a common scale. In this case, we could scale Teacher B's scores by multiplying them by a factor of 2 to align them with Teacher A's scale. After scaling, the student's score from Teacher B would become 40 * 2 = 80, which is equivalent to the score received from Teacher A.

Similarly, if you wanted to standardise the scores on both assignments to have a mean of 0 and a standard deviation of 1, you would apply a scaling transformation known as z-score scaling. This involves subtracting the mean of the dataset from each data point and then dividing by the standard deviation.

Scaling refers to the process of transforming a dataset by multiplying or dividing each data point by a constant value. This transformation changes the scale of the data without altering its shape or distribution. Scaling is commonly used to standardise or normalise the values within a dataset, making them comparable or easier to interpret.

To find scaling in statistics, you need to determine the constant value by which the dataset is being scaled and then apply this transformation uniformly to all data points.

Suppose you have a dataset representing the heights of individuals in inches. To standardise the heights using z-score scaling, you would subtract the mean height from each individual's height and then divide by the standard deviation. This transformation ensures that the scaled heights have a mean of 0 and a standard deviation of 1, facilitating comparisons across different datasets.

In summary, finding scaling in statistics involves identifying and applying a constant transformation to a dataset to adjust its scale. This process is commonly used to standardise or normalise data, making it easier to compare or analyse across different contexts or datasets.

1.1.4 Variance, Standard Deviation and Correlation Coefficients

1.1.4.1 Variance

Variance is a statistical metric which is employed to assess the dispersion of numbers within a dataset concerning the average value or mean. It is obtained by squaring the standard deviation. Variance enables the evaluation of how spread out or condensed a distribution appears. Denoted by the symbol σ2.

Suppose we have the data set {3, 5, 8, 1} and we want to find the population variance. The mean is given as (3 + 5 + 8 + 1) / 4 = 4.25. Then by using the definition of variance we get [(3- 4.25)2 + (5 - 4.25)2 + (8 - 4.25)2 + (1 - 4.25)2] / 4 = 6.68. Thus, variance = 6.68.

In another example:

Consider a dataset representing the ages of 5 individuals in a group: 25, 30, 35, 40, and 45 years old.

Step 1: Compute the mean of the dataset. The mean is found by adding up all the values and dividing by the total number of values. In this case, the sum of the ages is 25 + 30 + 35 + 40 + 45 = 175. Since there are 5 individuals, the mean age is 175 ÷ 5 = 35 years.

Step 2: Calculate the squared differences between each data point and the mean.

- For the first individual (age 25): (25 - 35)^2 = 100
- For the second individual (age 30): (30 - 35)^2 = 25
- For the third individual (age 35): (35 - 35)^2 = 0
- For the fourth individual (age 40): (40 - 35)^2 = 25
- For the fifth individual (age 45): (45 - 35)^2 = 100

Step 3: Find the average of the squared differences.

Average squared difference = (100 + 25 + 0 + 25 + 100) ÷ 5 = 50

Therefore, the variance of the ages in this dataset is 50 years squared.

It serves as a measure to examine the extent of individual data point variations relative to the mean. Variance functions as an indicator of dispersion, a metric used to assess the extent of variability in data around a central average value. Data can be categorised into two types: grouped and ungrouped. Grouped data is characterised by its representation in class intervals, while ungrouped data consists of individual data points. Both sample and population variance can be calculated for both types of data.

Population Variance refers to the measure of how individual data points within a complete group, known as the population. This method assesses the squared distance of each data point from the mean of the entire population.

On the other hand, Sample Variance comes into play when dealing with large populations where considering each data point is impractical. In such instances, a specific number of data points are selected from the population to form a sample that serves as a representative subset. Sample variance is then determined as the average of the squared distances from the sample mean. It is crucial to note that variance is consistently calculated in relation to the sample mean.

In a more general sense, variance can be defined as the expected value of the squared differences from the mean.

The population variance is computed through the following steps:

Determine the mean (average) of the dataset.

Subtract the mean from each number in the dataset and square the result. Squaring is performed to ensure that negative values become positive, as the focus is on the distance from the mean rather than the positive or negative nature of the numbers.

Calculate the average of the squared differences.

In statistical practice, it is more common to calculate the variance for a sample. When computing the sample variance, divide by the sample size minus one.

$$\text{Variance} = S^2 = \frac{\sum(X-\bar{X})^2}{n-1}$$

1.1.4.2 Standard Deviation

Suppose you are a coach of a high school basketball team, and you want to evaluate the consistency of your players' free throw shooting performances over the past 10 games. You have recorded the number of free throws attempted and made by each player in each game.

Here's a simplified representation of the data for one player, Player A, over 10 games:

Game 1: 8 attempts, 6 made

Game 2: 9 attempts, 7 made

Game 3: 7 attempts, 5 made

Game 4: 10 attempts, 8 made

Game 5: 8 attempts, 6 made

Game 6: 9 attempts, 7 made

Game 7: 10 attempts, 8 made

Game 8: 8 attempts, 6 made

Game 9: 9 attempts, 7 made

Game 10: 7 attempts, 5 made

Steps for computing standard deviation

Step 1: Calculate the free throw percentage for each game.

Step 2: Compute the mean free throw percentage for the 10 games.

Mean Percentage=Sum of all percentages / Number of games

Step 3: Find the squared differences between each game's free throw percentage and the mean.

For Game 1: $(\text{Percentage1} - \text{Mean Percentage})^2$

Step 4: Calculate the average of these squared differences.

Average of Squared Differences=Sum of all squared differences/Number of games

Average of Squared Differences= Sum of all squared differences / Number of games

Step 5: Take the square root of the average to find the standard deviation.

Standard Deviation=Average of Squared Differences

Standard Deviation = $\sqrt{Average\ of\ Squared\ Differences}$

Standard deviation serves as a statistical metric indicating the extent of dispersion in a dataset. The term "dispersion" refers to how widely data points are spread out. Specifically, it quantifies the degree to which data points deviate from the mean or average. It helps assess whether scores are closely clustered around the average or if there is a significant number of scores well above or below the average.

On a graphical representation as in Fig 1.1.3, the bell curve, also known as a "normal distribution," is frequently employed in statistics to illustrate standard deviation. In this context, the mean or average is denoted by the Greek letter μ at the centre. Each segment, shaded from dark blue to light blue, signifies one standard deviation from the mean. For instance, 2σ indicates a distance of two standard deviations from the mean.

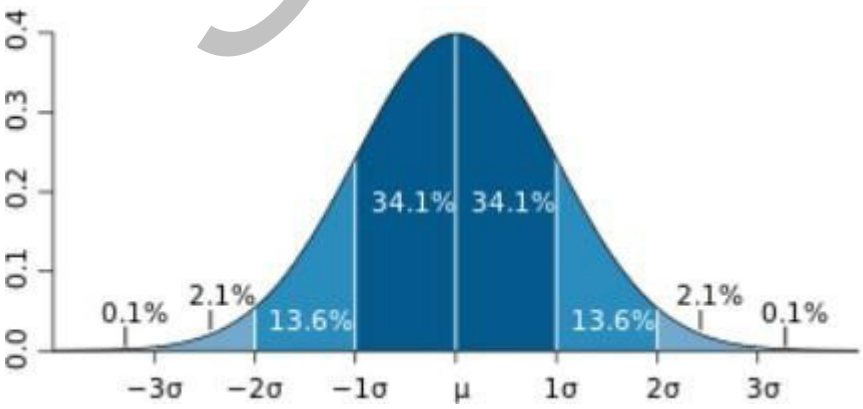


Fig. 1.1.3 Standard deviation

$$s = \sqrt{\frac{\sum(x - \bar{x})^2}{n - 1}}$$

sample standard deviation

Example Problem:

Q. Find the standard deviation for the following results:

{12, 15, 17, 20, 30, 31, 43, 44, 54}

Step1: Add the numbers up:

12 + 15 + 17 + 20 + 30 + 31 + 43 + 44 + 54 = 266.

Step2: Square the answer from Step 1:

266 x 266 = 70756

Step 3: Divide your answer from Step 2 by the number of items (n) in your set. In this example, we have 9 items, so: 70756 / 9 = 7861.777777777777 (dividing by n)

Set this number aside for a moment. You'll need it in a later step.

Step 4:Square the original numbers {12, 15, 17, 20, 30, 31, 43, 44, 54} one at a time, then add them up:

(12 x 12) + (15 x 15) + (17 x 17) + (20 x 20) + (30 x 30) + (31 x 31) + (43 x 43) + (44 x 44) + (54 x 54) = 9620

Step 5: Subtract Step 4 from Step 3.

9620 – 7861.777777777777 = 1758.2222222222226

Notice that we are not rounding yet. we should keep all of your decimal places until the very end, then rounding can be done. Rounding in the middle will lead to your answer being off just enough to get an incorrect answer. Set this number aside for a moment.

Step 6: Subtract 1 from n. We have 9 items, so n = 9:

9 – 1 = 8

Step 7: Divide Step 5 by Step 6 to get the variance: 1758.2222222222226 / 8 = 219.77777777777783

Step 8: Take the square root of Step 7:

$\sqrt{(219.77777777777783)}$ = 14.824903971958058

The standard deviation is 14.825.

1.1.4.3 Correlation Coefficients

Finding the correlation coefficient between the number of hours spent studying and the scores achieved by students in an exam. Suppose we have data for 10 students.

Table 1.1.1 Student data

Student	Study Hours (x)	Exam Score (y)
1	5	75
2	7	82
3	4	68
4	6	80
5	8	88
6	3	65
7	9	92
8	6	78
9	5	72
10	7	85

Step 1: Calculate the mean of study hours ($\underline{x}$) and exam scores (y).

$\underline{x}$ = (5+7+4+6+8+3+9+6+5+7) / 10=60 / 10=6

y = (75+82+68+80+88+65+92+78+72+85) / 10=785 / 10=78.5

Step 2: Compute the deviations from the mean for both study hours ($x - \underline{x}$) and exam scores ($y - y$).

Table 1.1.2 Deviations

Student	Study Hours (x)	Deviation from Mean (x - 6)	Exam Score (y)	Deviation from Mean (y- 78.5)
1	5	-1	75	-3.5
2	7	1	82	3.5
3	4	-2	68	-10.5
4	6	0	80	1.5
5	8	2	88	9.5
6	3	-3	65	-13.5
7	9	3	92	13.5
8	6	0	78	-0.5
9	5	-1	72	-6.5
10	7	1	85	6.5

Step 3: Calculate the product of the deviations for each student ($(x-\underline{x})(y-y)$).

Table 1.1.3 Deviation product

Student	Deviation Product
1	(-1) * (-3.5) = 3.5
2	1 * 3.5 = 3.5
3	(-2) * (-10.5) = 21
4	0 * 1.5 = 0
5	2 * 9.5 = 19
6	(-3) * (-13.5) = 40.5
7	3 * 13.5 = 40.5
8	0 * (-0.5) = 0
9	(-1) * (-6.5) = 6.5
10	1 * 6.5 = 6.5

Step 4: Find the sum of all the products of deviations.

Sum of Deviation Products = 3.5 + 3.5 + 21 + 0 + 19 + 40.5 + 40.5 + 0 + 6.5 + 6.5 = 141

Step 5: Calculate the standard deviation of study hours (σx) and exam scores (σy).

$\sigma x = ((-1)^2+1^2+(-2)^2+0^2+2^2+(-3)^2+3^2+0^2+(-1)^2+1^2)10=\sqrt{22 / 10} \approx 2.2 \approx 1.48$

$\sigma y = \sqrt{((-3.5)2 + 3.52 + (-10.5)2 + 1.52 + 9.52 + (-13.5)2 + 13.52 + (-0.5)2 + (-6.5)2 + 6.52)) / 10} \approx \sqrt{912 / 10} \approx \sqrt{91.2} \approx 9.54$

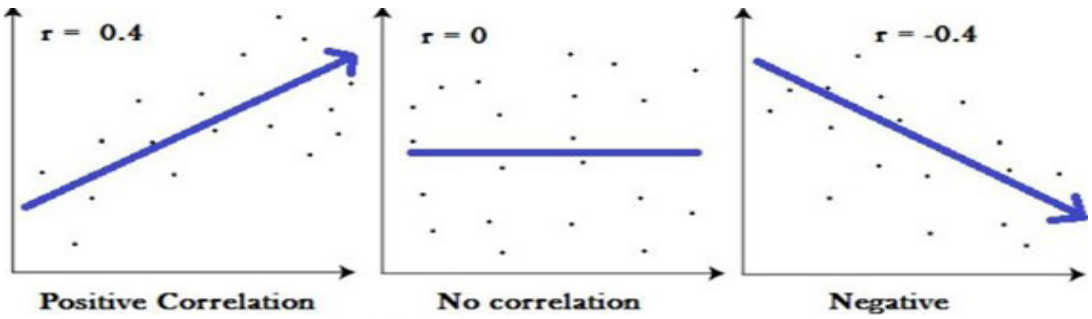
Step 6: Compute the correlation coefficient $r = \Sigma (x - \underline{x}) (y - y) / n\sigma x\sigma y = 141 / 10*1.48*9.54 \approx 141 / 141.192 \approx 0.998$

Therefore, the correlation coefficient (r) between the number of hours spent studying and the exam scores of the students is approximately 0.998. This indicates a very strong positive correlation between study hours and exam scores.

The formulas for correlation coefficients are employed to assess the strength of a relationship between data. The resulting value falls within the range of -1 to 1, where:

1 signifies a robust negative relationship.

1 signifies a robust positive relationship. A result of zero indicates the absence of any relationship.



Graphs showing a correlation of -1, 0 and +1

Fig. 1.1.4 Correlation

The correlation coefficient is a statistical measure that quantifies the strength and direction of the relationship between two variables as in Fig 1.1.3. It ranges from -1 to 1, where:

- A correlation coefficient of 1 indicates a perfect positive linear relationship between the variables. This means that as one variable increases, the other variable also increases proportionally.
- A correlation coefficient of -1 indicates a perfect negative linear relationship between the variables. This means that as one variable increases, the other variable decreases proportionally.
- A correlation coefficient of 0 indicates no linear relationship between the variables.

In essence, the correlation coefficient helps to assess the extent to which changes in one variable are associated with changes in another variable. It's important to note that correlation does not imply causation, meaning that just because two variables are correlated does not necessarily mean that one causes the other to change.

1.1.5 Outliers and Normalisation

1.1.5.1 Outliers

Consider a dataset representing the income of individuals in a particular region. Most of the data might cluster around average income levels, but there could be some extreme values or outliers that significantly deviate from the norm.

For example, let's say we have a dataset of income levels of residents in a city:

{20000, 25000, 30000, 35000, 40000, 45000, 50000, 55000, 60000, 1000000}

In this dataset, the majority of incomes fall between Rs.20,000 and Rs.60,000, representing typical earnings in the city. However, there is one individual with an income of Rs.1,000,000, which is significantly higher than the rest of the data points.

This extreme value of Rs.1,000,000 would be considered an outlier. Outliers can skew statistical analyses and machine learning models, leading to inaccurate results. In this case, the presence of the outlier might inflate the average income of the dataset, making it seem higher than it actually is. In data science, identifying and handling outliers is crucial to ensure that models and analyses are robust and accurate.

An outlier refers to a data point that significantly deviates from the average value of a set of statistics, and it may also stand out from individual samples within populations. In broader terms, an outlier represents an individual that noticeably differs from the usual characteristics in some aspect.

Outliers play a crucial role in statistical analysis as they can greatly impact overall findings. Particularly in small sample sizes, the presence of a single outlier can significantly alter averages and distort the final outcomes of a study. Statisticians endeavour to address the influence of outliers and have developed methods to identify them. For instance,

outliers can be visually detected in scatter plots where data points are graphed, or in box plots, equations are employed to determine if they surpass predefined thresholds.

Various factors can lead to the occurrence of outliers, such as misinformation provided by a subject, errors in data entry, or inaccuracies in responses. Sometimes, it is evident that outliers should be disregarded as errors, while in other cases, their exclusion may hinge on subjective judgments or established criteria where outliers are considered a natural deviation. Outliers may signal potential errors or deficiencies in the methods of sample collection. However, they can also indicate anomalies or areas worthy of further investigation, as it is not always straightforward to ascertain if outliers are erroneous. Although outliers can introduce bias into statistical analyses, they are seldom entirely eliminated from results without careful consideration and examination.

Outliers occur from various factors, some typical causes of outliers include:

- Mistakes during data gathering or measurement procedures can lead to outlier occurrences (Measurement inaccuracies).
- Outliers may emerge due to issues with the sampling technique employed (Sampling discrepancies).Certain phenomena exhibit extreme values naturally, influenced by the inherent variability of the process under investigation (Inherent variability).
- Human mistakes made during data entry can introduce outliers into the dataset (Data input errors).
- Anomalies may arise in experimental setups due to uncontrolled variables, equipment malfunctions, or unexpected occurrences (Experimental discrepancies).
- Combination of data collected from multiple populations with varying characteristics can lead to outlier presence (Combination of diverse populations).
- Outliers may be intentionally introduced to evaluate the resilience of statistical methodologies (Deliberate introduction of outliers).

Finding Outliers in a worksheet:

To draw attention to outliers within the spreadsheet, you can simply right-click on the column containing your data and select Conditional Formatting > Statistical > Outlier. This will result in every outlier in the spreadsheet being visually marked in red, or any other colour of your preference.

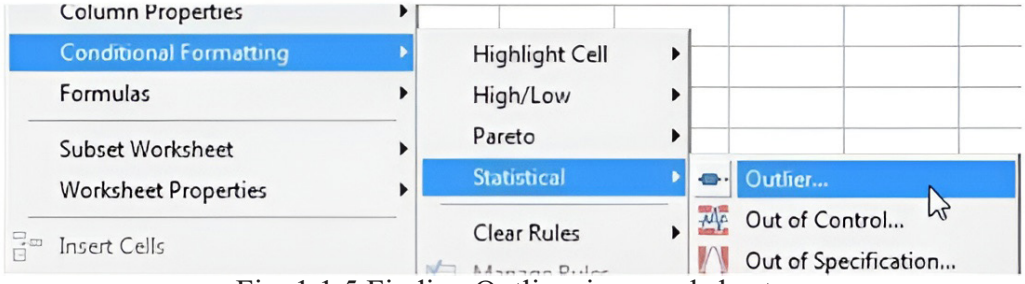


Fig. 1.1.5 Finding Outliers in a worksheet

Initially, follow the steps outlined for highlighting outliers using Conditional Formatting as in Fig 1.1.5. Afterward, right-click on the column once more and opt for Subset Worksheet > Exclude Rows with Formatted Cells to generate the revised dataset.

Finding Outliers in a Graph:

If we want to visually identify them in a graph and see where your outliers are positioned relative to the rest of your data, you can utilise the Graph > Boxplot feature as in Fig 1.1.6.

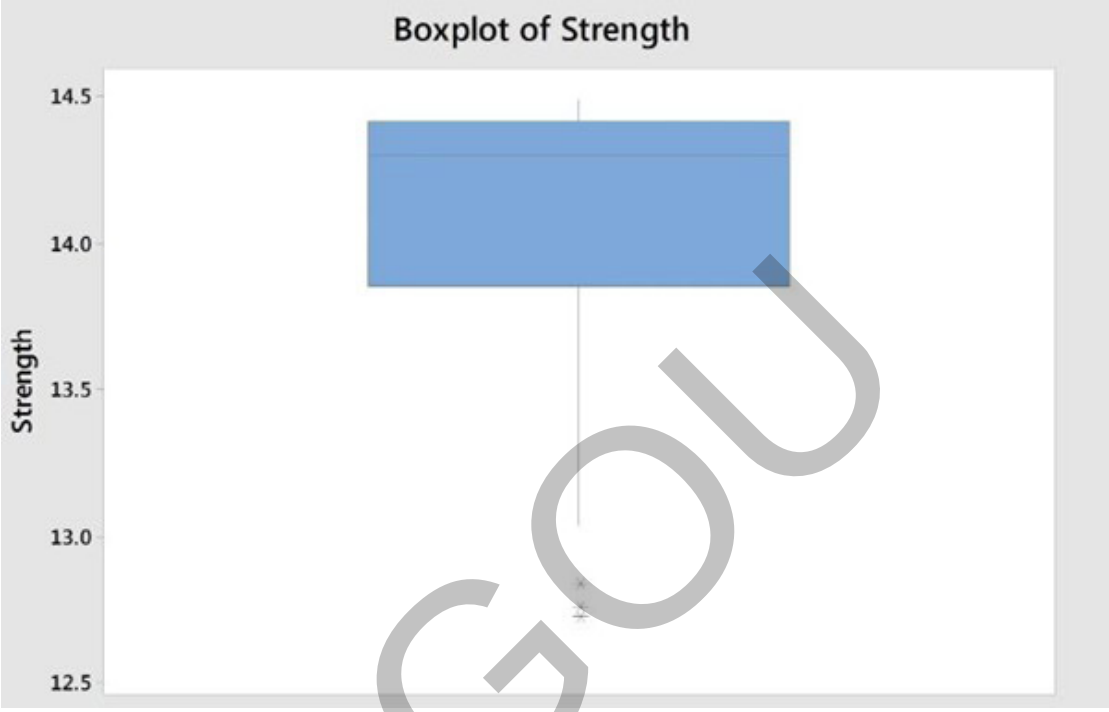


Fig. 1.1.6 Finding Outliers in a Graph
See the e-content for a color version of this image

1.1.5.2 Normalisation

Normalisation involves transforming data values to conform to a standardised scale or distribution, facilitating meaningful comparisons. This process encompasses aligning the scale of values to a consistent metric or adjusting time scales for comparative analysis of similar periods. For instance, when dealing with health data containing annual height measurements in feet and daily weight measurements in pounds, normalisation might entail adjusting the values to represent percentages within the range spanning from the minimum to the maximum values. Normalisation is a crucial concept in statistics used to standardise data, making it easier to interpret and compare across different scales. This process transforms the data so that it conforms to a common scale without distorting the differences in the range and distribution of values. In this one-page exploration, we'll delve into the significance of normalisation in statistics and explore common methods used for normalisation.

Significance of Normalisation:

In statistical analysis, datasets often contain variables with vastly different scales. For instance, consider a dataset comprising income and age variables. Income might range from a few thousand dollars to millions, while age typically ranges from 0 to 100. When analysing such data, the inherent differences in scales can skew the results and overshadow the importance of certain variables. Normalisation resolves this issue by bringing all variables to a standard scale, facilitating fair comparison and analysis.

Normalisation Methods in Statistics

Here are some common normalisation methods:

1. Min-Max Scaling
2. Z-Score Normalisation
3. Decimal Scaling
4. Standardisation
5. Logarithmic Transformation

One of the most commonly used normalisation method is Z-Score Normalisation

Choosing the Right Method

The best normalisation method depends on the specific characteristics of your data and the intended analysis. Consider factors like:

- **Distribution of data:** Is it symmetrical or skewed?
- **Presence of outliers:** Are there extreme values that need to be addressed?
- **Desired range:** Do you need specific limits for the normalized data?
- **Interpretability:** How important is it to preserve the original data meaning?

By understanding these methods and their limitations, you can choose the most suitable approach for your statistical analysis and ensure meaningful comparisons.

Normalisation plays a pivotal role in statistical analysis by ensuring fair comparisons and accurate interpretations of data. By standardising data to a common scale, normalisation enables researchers to focus on the intrinsic relationships between variables without being influenced by differences in scales. Understanding the various methods of normalisation equips analysts with the tools to preprocess data effectively, leading to more robust and reliable statistical analyses.

Recap

- Statistical measures provide crucial insights and summaries of data
- Central Tendency: Mean, Median, and Mode
- Mean: Average of all values
- Median: Middle value when ordered
- Mode: Most frequent value.
- Range: Difference between highest and lowest values
- Skewness: Quantifies asymmetry in data distribution
- Shifting and: Adjusting data values
- Scaling: Data values to fit a specific range
- Variance: Measures of data spread around the mean
- Standard Deviation: indicating data dispersion.
- Correlation: Strength and direction of relationships between two variables

Objective Type Questions

1. What measure summarises the "center" of a data set?
2. Which measure is most influenced by outliers?
3. For a symmetrical distribution, which measures are usually similar?
4. What does a negative value for skewness indicate?
5. What does "shifting" data do?
6. What does "scaling" data do?
7. Which formula represents the population variance?
8. How can you identify outliers in a scatter plot?
9. What is the main purpose of data normalisation?

Answers to Objective Type Questions

1. Mean
2. Mean
3. Mean, Mode, Median
4. Data is clustered on the left side
5. Moves all values by a constant amount
6. Fits values within a specific range
7. Ó(x - ì)^2 / n
8. Strength and direction of a relationship
9. To facilitate comparisons between different data sets

Assignments

1. Discuss the concepts of shifting and scaling in data preprocessing. Demonstrate with an example how these transformations affect mean and median but not variance or standard deviation.
2. Write about measures of centra tendency.
3. Explain about measures of dispersion.
4. Using a small dataset, identify an outlier and perform min-max normalisation to scale the data between 0 and Explain how normalisation helps in improving model performance in data analytics.
5. Using the example of study hours and exam scores, explain the steps to compute the correlation coefficient. Interpret the meaning of a correlation coefficient close to +1, -1, and 0.

Reference

1. Freedman, D., Pisani, R., & Purves, R. (2007). *Statistics* (4th ed.). W. W. Norton & Company.
2. Utts, J. M., & Heckard, R. F. (2014). *Mind on statistics* (5th ed.). Cengage Learning.

3. Agresti, A., & Franklin, C. (2017). *Statistics: The art and science of learning from data* (4th ed.). Pearson.

4. Moore, D. S., McCabe, G. P., & Craig, B. A. (2017). *Introduction to the practice of statistics* (9th ed.). W. H. Freeman.

5. Field, A. (2018). *Discovering statistics using IBM SPSS statistics* (5th ed.). SAGE Publications.

Suggested Reading

1. Agarwal, B. L. (2013). *Basic statistics.* New Age International Publishers.

2. Bhat, B. R., Sri Venkata Ramana T, & Rao Madhava K. S. (1977). *Statistics: A beginners Text* Vol. 2. New Age International (P) Ltd., New Delhi.

3. Dekking, F. M., & others. (2005). *A Modern Introduction to Probability and Statistics.* Springer Verlag, New York.

Probability for Exploratory Data Analysis

Learning Outcomes

After completing this unit, the learner will able to:

- to familiarise basic probability rules
- to provide an overview of Venn diagrams for probability representations
- to get awareness on dependent and independent events
- to introduce the concept of conditional probability and Bayes theorem

Prerequisites

As we scroll through news on our phone, we see many articles about new gadgets. If we're curious, we might click on one, and then we'll start getting lots of advertisements for that same gadget. That's not magic, it's data analytics using probability. Companies use calculations to guess how likely you are to buy something based on your clicks and online behaviour. This helps them show you advertisements for things you might actually want, just like predicting how many rainy days there might be helps plan picnics. Probability in data analytics helps companies understand what you and millions of others might do, making their marketing smarter and, hopefully, making your online experience more relevant.

Probability helps predict the likelihood of a customer completing a purchase based on their browsing history, previous purchases, and clicked advertisements. This knowledge drives personalised recommendations, targeted ads, and dynamic pricing strategies, ultimately boosting sales and improving the customer experience.

In Summary: Probability is not just about numbers; it's about understanding the shades of grey in data. By embracing its power, data analysts can transform uncertainties into valuable insights, driving better decisions and achieving better outcomes across various domains.

Keywords

Independent Events, Disjoint Events, Conditional Probability, Bayes Theorem, Normal Distribution

Discussion

Probability is an important part of data analytics because it helps us deal with uncertainty and make smarter decisions based on data. One key concept in probability is “Bayes' Theorem”, which allows us to update our predictions when we get new information. For example, if a medical test shows a positive result, Bayes' Theorem helps doctors decide how likely it is that the person actually has the disease, considering both the test result and previous data. Another useful concept is the “probability distribution”, which shows how data is spread out. A common type is the “normal distribution”, which looks like a bell-shaped curve. It helps analysts understand patterns in data and predict how often certain values will occur. For instance, heights of people often follow a normal distribution, which helps us estimate how many people are likely to be above or below a certain height.

1.2.1 Introduction to Probability

Probability means how likely something is to happen. When we are unsure about what will happen, we use probability to think about the chances of different outcomes. The study of these chances is called statistics.

A simple example is flipping a coin. When you flip a coin, it can land on either heads or tails. To find out how likely it is to get heads, we use a basic formula. Even though we often say there's a 50% chance for heads and 50% for tails, probability helps us understand and calculate this more clearly.

The probability equation is as follows:

$$\text{Probability of an event} = \frac{\textit{Number of ways it can happen}}{\textit{Total number of possible outcomes}}$$

This is usually written as:

P(A) = (Number of ways A can happen) / (Total number of outcomes)

The result is always between 0 and 1, where:

- 0 means the event is impossible
- 1 means the event is certain.

You can also write probability as a percentage (for example, 0.5 = 50%).

If P(A) > P(B), it means event A is more likely to happen than event B.

If P(A) = P(B), it means both events have an equal chance of happening.

1.2.2 Rules of Probability

The value of a probability is always between 0 and 1.

- 0 means the event cannot happen (impossible).
- 1 means the event will definitely happen (certain).

The total of all probabilities for every possible outcome in a situation always adds up to 1.

1. Addition Rule (For "Or" Events)

This rule helps us find the chance that one event, or both events, will happen.

If two events can't happen at the same time (called mutually exclusive), use:

P(A or B) = P(A) + P(B)

(Example: Rolling a 2 or rolling a 3 on a die.)

If events can happen at the same time, use:

P(A or B) = P(A) + P(B) - P(A and B)

(We subtract the part that is counted twice.)

2. Multiplication Rule (For "And" Events)

This rule is used to find the chance that two events happen together.

If events are independent (one doesn't affect the other), use:

P(A and B) = P(A) × P(B)

(Example: Tossing a coin and rolling a die.)

If events are **dependent** (the first affects the second), use:

P(A and B) = P(A) × P(B|A)

Here, P(B|A) means the chance of B happening given that A already happened.

3. Conditional Probability

This tells us the chance of an event happening after another event has already happened.

Written as: **P(A|B) = P(A and B) / P(B)**

("P A given B" = chance of both A and B happening ÷ chance of B happening)

4. Mutually Exclusive Events

These are events that cannot happen at the same time.

Example: Rolling a 2 and rolling an odd number on a die.

Since 2 is even, it can't be both 2 and odd at the same time, so the events are mutually exclusive.

1.2.3 Venn Diagram

A Venn diagram is a visual way to show the relationship between different events in a sample space or Venn diagram serves as a graphical representation to illustrate the connections among different sets or occurrences. Typically, it comprises circles that intersect partially, each circle representing distinct events or sets.

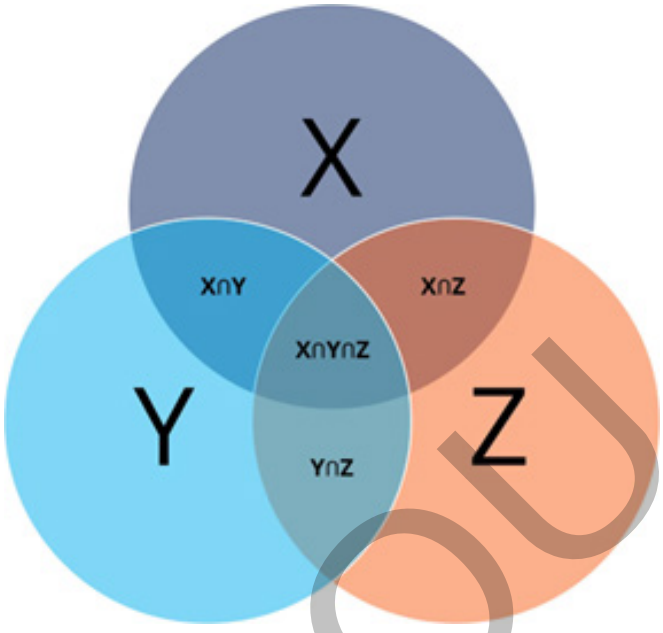


Fig. 1.2.1 Venn Diagram for three sets labeled X, Y, and Z and the corresponding relationships between elements in each set
See the e-content for a color version of this image

1.2.3.1 Steps for calculating Probability Using a Venn Diagram

Step 1: Count the number enclosed by the event you are being asked about.

Step 2: Calculate the probability by using the number from step 1 in the numerator and using the total number represented in the entire diagram in the denominator. Convert your final answer to a percentage.

These principles and methods will guide us in computing probabilities using Venn diagram examples.

The Venn diagram below shows the results of a survey of 300 high school students on whether they participate in football, band, or drama. Find the probability of a randomly selected high schooler participating in band or drama (or both) but not football. Round to the nearest tenth of a percent.

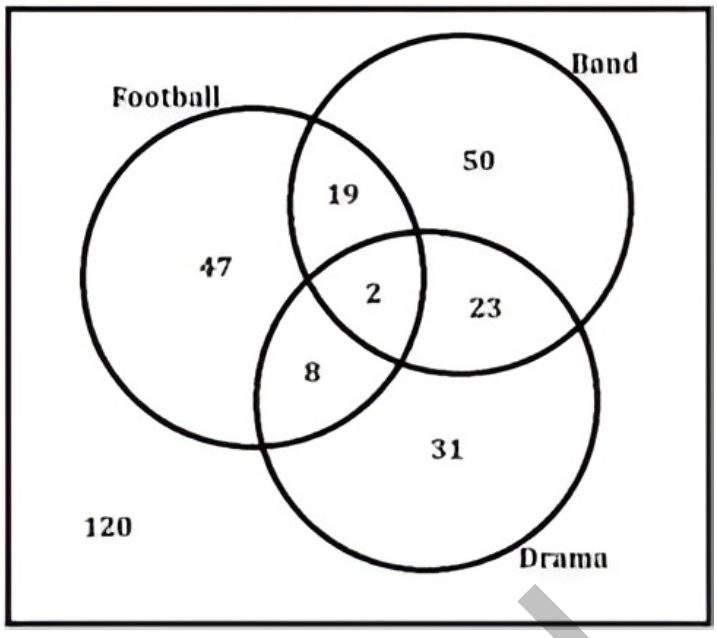


Fig. 1.2.2 Venn Diagram of 300 high school students on whether they participate in football, band, or drama

Step 1: Count the number enclosed by the event you are being asked about. Counting the students:50 participate in only band 31 participate in only drama,23 participate in both band and drama, but not football

This is a total of 50 + 31 + 23 = 104 students who participate in band or drama (or both) but not football.

Step 2: Calculate the probability by using the number from step 1 in the numerator and using the total number represented in the entire diagram in the denominator. Convert your final answer to a percentage.

Calculating the probability:

P(participates in band or drama (or both) but not football) = 104/300 = 0.34666…

Rounding to the nearest tenth of a percent, there is a 34.7% chance that a randomly selected student will participate in band or drama (or both) but not football.

1.2.4 Disjoint Events and Non-disjoint Events

1.2.4.1 Disjoint Events

Consider the below cases:

The outcome of a single coin toss cannot be a head and a tail. A student cannot both fail and pass an exam.A single card drawn from a deck cannot be an ace and a queen at the same time. All the above cases are disjoint events.

Disjoint Events, by definition, cannot happen at the same time. Another term is mutually exclusive.

In probability, disjoint event A and B is expressed as: P(A and B) = 0

In Venn diagram representation, we represent each event by the circles; if event A and B are disjoint, we end up with two circles that don't touch each other.

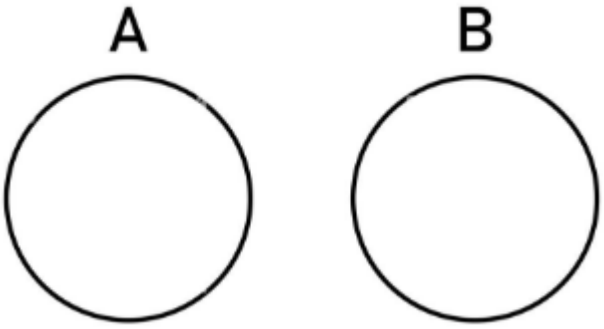


Fig. 1.2.3 Disjoint event

Union of disjoint events

For disjoint events A and B, the probability of A or B happening is simply the probability of A plus the probability of B.

$$P(A \text{ or } B) = P(A) + P(B)$$

1.2.4.2 Non-disjoint Events

Two events A and B are non-disjoint if:

$$P(A \cap B) \neq 0$$

This means the events intersect, or share outcomes, so both can happen simultaneously.

A student can get an A in Mathematics and A in History at the same time.

In the Venn diagram representation of events A and B that are non-disjoint, we have two circles that overlap, or in other words, join, which indicates that the probability of event A and B happening at the same time is non-zero. So it is some number between 0 and 1.

$$P(A \text{ and } B) \neq 0$$

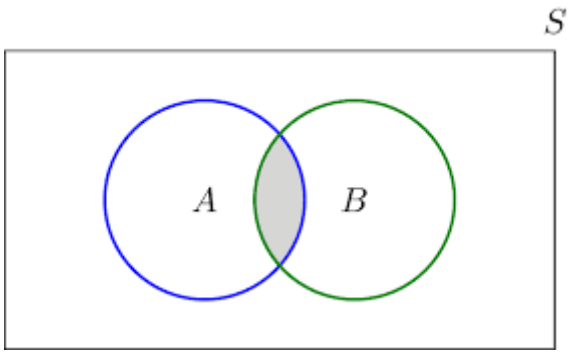


Fig. 1.2.4 Non Disjoint Set

Union of non-disjoint event

For non-disjoint events A and B, the probability of A or B happening is the P(A) plus P(B) minus the probability of A and B happening at the same time (Avoid double counting).

$$P(A \text{ or } B) = P(A) + P(B) - P(A \text{ and } B)$$

Example:

A single card is drawn from a standard deck of 52 playing cards.Let:

- Event A: Drawing a heart
- Event B: Drawing a queen

What is the probability that the card is a heart or a queen?

Total number of cards = 52

Count the outcomes

- Number of hearts = 13
- Number of queens = 4
- Overlap (queen of hearts) = 1 card (counts in both A and B)

Since one card (queen of hearts) is in both events, they are non-disjoint.

Use the union formula:

$$P(A \text{ or } B)=P(A)+P(B)-P(A \text{ and } B)$$

$$P(\text{Heart})=13/52,$$

$$P(\text{Queen})=4/52,$$

$$P(\text{Heart and Queen})=1/52$$

$$P(\text{Heart or Queen})=13/52+4/52-1/52$$

$$=16/52=4/13$$

1.2.5 Sample Space

A sample space is a collection of all possible outcomes of a trial. For example:

A couple has two kids, the sample space for the sex of these 2 kids is:

$$S= \{MM, FF, FM, MF\}$$

M for Male, F for Female

If you toss a fair coin twice, the sample space are:

S = {HH, TT, HT, TH}

H for Head, T for Tail

1.2.6 Dependent and independent Events

1.2.6.1 Dependent Events

A dependent event in probability is an event whose outcome is influenced by the outcome of another event that has already occurred. If the happening of one event affects the chances of another event happening, then those events are dependent.

Here's an example:

- Let's say you pick a ball from a bag containing 3 red balls and 2 blue balls.
- Event A: Picking a red ball.
- Event B: Picking another red ball after putting the first one back.

Event A and B are dependent because when you pick a red ball in Event A, you're leaving fewer red balls in the bag, making it less likely to pick another red ball in Event B (compared to if you hadn't taken out any balls).

Here are some key points about dependent events:

- The probability of a dependent event happening depends on the outcome of the event that happened before it.
- You can't use the simple multiplication rule (multiplying the probabilities of each event) to find the probability of two dependent events occurring together. You need to use conditional probability, which takes into account the order of events.
- Many real-life events are dependent. For example, the probability of rain tomorrow depends on the weather today.

Here are some other examples of dependent events:

- Drawing two aces from a deck of cards without replacing the first one.
- Rolling a dice, getting a 6, and then rolling another 6.
- Robbing a bank and going to jail.
- Not paying your water bill on time and having your water supply cut off.
- Boarding a plane first and finding a good seat.
- Parking illegally and getting a parking ticket. Parking illegally increases your odds of getting a ticket.
- Buying ten lottery tickets and winning the lottery. The more tickets you buy, the greater your odds of winning.
- Reckless driving and getting in a traffic accident.

1.2.6.2 Independent Events

The probability of one event does not change based on the outcome of the other event.

When two events are independent, one event does not influence the probability of another event. Independent events can be observed in a scenario involving tossing two separate coins. When flipping the first coin, the outcome of landing heads or tails is entirely independent of the outcome of flipping the second coin. Whether the first coin lands heads or tails does not influence the result of the second coin toss. Each coin toss represents an independent event because the outcome of one does not affect the outcome of the other.

An independent event is an event that has no connection to another event's chances of happening (or not happening). In other words, the event has no effect on the probability of another event occurring. Independent events in probability are no different from independent events in real life.

Some of the other examples of independent events in real life are:

- Where you work has no effect on what colour car you drive.
- Buying a lottery ticket has no effect on having a child with blue eyes.
- Owning a dog and growing your own herb garden.
- Winning a race and running out of milk.
- Taking a cab home and finding your favourite movie on cable.
- Getting a parking ticket and playing cricket with friends.

Checking for Independence of Events

There are four formulas for checking for independence of events:

P(B | A) = P(B)	—	A doesn't affect B
P(A | B) = P(A)	—	B doesn't affect A
P(B | A) = P(B | not A)	—	B is unaffected by A occurring or not
P(A and B) = P(A) * P(B)	—	Joint = product

Example 1: Flipping a Coin

Let's say:

- Event A = Getting Heads
- Event B = Getting Tails

The probability of getting heads or tails is always 50% (0.5).

Now, if you already flipped the coin once and got heads, there is a 50% chance of getting tails on the next flip.

Each coin flip is independent; the outcome of one flip doesn't affect the next.

This matches the condition:

$$P(B \mid A)=P(B)=0.5$$

This tells us the events are independent.

Example 2: Drawing Cards

There are 52 cards in a deck.

There are 4 Jacks.

$$P(\text{Jack})=4/52=1/13\approx 7.69\%$$

With Replacement (Independent Events)

- You draw a card.
- Look at it.
- Put it back.
- Shuffle and draw again.

The number of cards stays the same (52), so your chance of drawing a Jack doesn't change.

Each draw is independent.

Without Replacement (Dependent Events)

- You draw a card and don't put it back.
- Now the deck has 51 cards.

Let's say you didn't draw a Jack on the first try.

Now:

$$P(\text{Jack})=4/51\approx 7.84\%$$

The chance of drawing a Jack increased because the total number of cards decreased and you know the Jack wasn't drawn the first time. So, without replacement, events are dependent, what happens in one trial affects the next.

How to tell if an event is Dependent or Independent

Distinguishing between dependent and independent events is crucial when solving probability problems. For instance, winning the lottery depends on purchasing a ticket, winning the lottery and buying ticket dependent events. However, events like driving to

work and winning the lottery are unrelated, making them independent. Understanding the dependency between events is essential as it significantly affects the likelihood of occurrence. Figuring out whether events are dependent or independent events can be challenging.

1.2.7 Marginal Probability

Marginal Probability is a fundamental concept in probability theory and statistics. It refers to the probability of the single event occurring irrespective of the outcomes of the other related events. It often appears in the margins (or totals) of a probability table. It helps us focus on just one part of the data. The Marginal probabilities are essential for understanding joint distributions and are commonly used in various fields including economics, engineering and social sciences.

For example, suppose a teacher has a class of 20 students. Out of these, 12 are boys and 8 are girls. If we randomly select one student from the class, the marginal probability of selecting a girl is simply the number of girls divided by the total number of students. That is:

$$P(Girl)=8/20=0.4$$

This means there is a 40% chance of selecting a girl from the class. Notice that we are only interested in the probability of picking a girl, not considering any other factor (like age, grade, or performance). That's why this is a marginal probability, it focuses on one variable alone.

Formula for Marginal Probability

Formula for marginal probability are different for different variables:

- **Discrete Random Variables**
- **Continuous Random Variables**

For Discrete Random Variables

Let X and Y be two discrete random variables with the joint probability mass function P(X = x, Y = y). The marginal probability mass function of X is obtained by summing over all possible values of Y:

$$P(X=x)=\sum_{y} P(X=x,Y=y)$$

Similarly, the marginal probability mass function of Y is:

$$P(Y=y)=\sum_{x} P(X=x,Y=y)$$

For Continuous Random Variables

For continuous random variables, the joint probability density function $f_{X,Y}(x,y)$ is used. The marginal probability density function of the X is obtained by the integrating over all values of

Y: $f(X=x)=\int_{-\infty}^{\infty} f\ (x,y)dy$

The marginal probability density function of Y is:

$$f(Y=y)=\int_{-\infty}^{\infty} f\ (x,y)dx$$

If two fair dice are rolled. Calculate the marginal probability of getting a 3 on first die.

Solution:

Sample space of rolling two dice = 36

Number of favourable outcomes = 6

Marginal Probability P (First die = 3) = 6 / 36

Marginal Probability P (First die = 3) = 1 / 6

1.2.8 Joint Probability

A statistical measure that calculates the likelihood of two events occurring together and at the same point in time is called Joint probability.

Let A and B be the two events, joint probability is the probability of event B occurring at the same time that event A occurs.

Formula for Joint Probability

Notation to represent the joint probability can take a few different forms. The following formula represents the joint probability of events with intersection, P (A∩B)

Where, A, B= Two events

P(A and B),P(A∩B)=The joint probability of A and B

The symbol "∩" in a joint probability is called an intersection. The probability of event A and event B happening is the same thing as the point where A and B intersect. Hence, the joint probability is also called the intersection of two or more events. We can represent this relation using a Venn diagram as shown below.

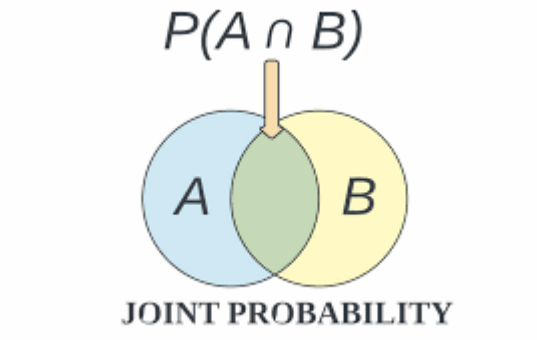


Fig. 1.2.5 Joint probability
See the e-content for a color version of this image

Let us see some examples of how to find the joint probability with solutions.

Example: Find the probability that the number three will occur twice when two dice are rolled at the same time.

Solution:

Number of possible outcomes when a die is rolled = 6

i.e. {1, 2, 3, 4, 5, 6}

Let A be the event of occurring 3 on the first die and B be the event of occurring 3 on the second die. Both the dice have six possible outcomes, the probability of a three occurring on each die is 1/6.

P(A) =1/6

P(B)=1/6

P(A,B) = 1/6 x 1/6 = 1/36

1.2.9 Bayes Theorem

Bayes theorem is a theorem in probability and statistics, named after the Reverend Thomas Bayes, that helps in determining the probability of an event that is based on some event that has already occurred.

Bayes rule has many applications such as Bayesian inference, an example application in the healthcare sector to determine the chances of developing health problems with an increase in age and other factors.The Bayes theorem is based on finding P(A | B) when P(B | A) is given. Bayes Law is a method to determine the probability of an event based on the occurrences of prior events. It is used to calculate conditional probability. Bayes theorem calculates the probability based on the hypothesis.

Now, let us state and prove Bayes Theorem. Bayes rule states that the conditional probability of an event A, given the occurrence of another event B, is equal to the product of the likelihood of B, given A and the probability of A divided by the probability of B. It is given as:

$$P(A|B) = (P(B|A) * P(A))/P(B)$$

- P(A) = how likely A happens (Prior knowledge)- The probability of a hypothesis is true before any evidence is present.
- P(B) = how likely B happens (Marginalisation)- The probability of observing the evidence.
- P(A|B) = how likely A happens given that B has happened (Posterior)-The probability of a hypothesis is true given the evidence.
- P(B|A) = how likely B happens given that A has happened (Likelihood)- The probability of seeing the evidence if the hypothesis is true.

When two events A and B are independent, P(A|B) = P(A) and P(B|A) = P(B)

Conditional probability can be calculated using the Bayes theorem for continuous random variables.

1.2.10 Normal Distribution/Gaussian Distribution

Normal Distribution is also called the Gaussian Distribution, is the most significant continuous probability distribution. Sometimes it is also called a bell curve. A large number of random variables are either nearly or exactly represented by the normal distribution, in every physical science and economics. Furthermore, it can be used to approximate other probability distributions, therefore supporting the usage of the word 'normal 'as in the one, mostly used.

The Normal Distribution is defined by the probability density function for a continuous random variable in a system. Let us say, f(x) is the probability density function and X is the random variable. Hence, it defines a function which is integrated between the range or interval (x to x+dx), giving the probability of random variable X, by considering the values between x and x+dx.

$$f(x) \geq 0 \ \forall \ x \ \Gamma(-\infty,+\infty)$$

This means:

- The function f(x) is non-negative for all real numbers x.
- In other words, f(x) can never be negative, which makes sense because probabilities cannot be negative.

And $\int_{-\infty}^{+\infty} f(x)dx = 1$

This means:

- The total area under the curve of f(x) over the entire real line equals 1.
- This reflects the fact that the total probability across all possible values of a continuous random variable must be 1.

Normal Distribution Formula

The probability density function of normal or gaussian distribution is given by;

$$f(x,\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

Where, x = variable

μ = mean

σ = standard deviation

Normal Distribution Problems

Question 1: Calculate the probability density function of normal distribution using the following data. x = 3, μ = 4 and σ = 2.

Solution:

Given, variable, x = 3

Mean = 4 and

Standard deviation = 2

By the formula of the probability density of normal distribution, we can write

$$f(3,4,2) = \frac{1}{2\sqrt{2\pi}} e^{\frac{-(3-2)^2}{2\times 2^2}}$$

Hence, f(3,4,2) = 1.106

Question 2: If the value of the random variable is 2, mean is 5 and the standard deviation is 4, then find the probability density function of the gaussian distribution.

Solution: Given,

Variable, x = 2

Mean = 5

Standard deviation = 4

By the formula of the probability density of normal distribution, we can write;

$$f(2,2,4) = \frac{1}{4\sqrt{2\pi}} e^{\frac{-(2-2)^2}{2\times 4^2}}$$

$f(2,2,4) = 1/(4\sqrt{2\pi})\ e^0$

f(2,2,4) = 0.0997

There are two main parameters of normal distribution in statistics namely mean and standard deviation. The location and scale parameters of the given normal distribution can be estimated using these two parameters.

Normal Distribution Properties

Some of the important properties of the normal distribution are listed below:

- In a normal distribution, the mean, median and mode are equal.(i.e., Mean = Median= Mode). The total area under the curve should be equal to 1.
- The normally distributed curve should be symmetric at the centre.
- There should be exactly half of the values to the right of the centre and exactly half of the values are to the left of the centre.

- The normal distribution should be defined by the mean and standard deviation. The normal distribution curve must have only one peak. (i.e. Unimodal)
- The curve approaches the x-axis, but it never touches, and it extends farther away from the mean.

Applications of Normal Distributions

The normal distributions are closely associated with many things such as :

- Marks scored on the test
- Heights of different persons
- Size of objects produced by the machine
- Blood pressure and so on.

Recap

- Probability measures the likelihood of an event occurring, expressed as a number between 0 and 1.
- Basic Rules of probability.
    - Addition Rule:
    - Mutually exclusive: P(A or B) = P(A) + P(B). Mutually exclusive: Cannot happen at the same time.
    - Not mutually exclusive: P(A or B) = P(A) + P(B) - P(A and B).
    - Multiplication Rule:
- Independent events: P(A and B) = P(A) * P(B).
- Dependent events: Use conditional probability.
- Conditional Probability: P(A|B) = P(A and B) / P(B).
- Venn Diagrams: Graphical representation of probabilities using overlapping circles.
- Disjoint Events: Mutually exclusive events, represented by non-touching circles in Venn diagrams.
- Dependent vs. Independent
- Bayes' Theorem: Conditional probability based on prior events.
- Normal Distribution: Bell-shaped curve.

Objective Type Questions

1. What is the range of possible values for a probability?
2. Which of the following events are mutually exclusive?
3. What is the formula for calculating the probability of event A happening given that event B has already happened (conditional probability)?
4. In a coin toss, what is the probability of getting heads and tails?
5. What is the difference between dependent and independent events?
6. What is the sum rule of probability?
7. What is the multiplication rule of probability?
8. What is the complement rule of probability?
9. What is the probability of drawing an ace from a standard deck of cards?
10. What is the probability of rolling a 3 and then a 5 on two separate rolls of a dice?

Answers to Objective Type Questions

1. 0 to 1
2. Drawing a red card or a heart from a deck of cards.
3. P(A | B) = P(A and B) / P(B)
4. 0% (Heads and tails are mutually exclusive events in a single coin toss).
5. The outcome of one dependent event affects the probability of the other, while the outcome of one independent event doesn't affect the other.
6. P(A or B) = P(A) + P(B) (mutually exclusive events) or b) P(A or B) = P(A) + P(B) - P(A and B) (not mutually exclusive events) depending on the events.
7. P(A and B) = P(A) * P(B) (independent events)
8. P(not A) = 1 - P(A)
9. 4/52
10. 1/6 * 1/6 = 1/36

Assignments

1. Write about basic rules of probability.
2. Explain about Bayes theorem.
3. What is gaussian distribution? How it is useful in data analytics?

Reference

1. Grinstead, C. M., & Snell, J. L. (2012). *Introduction to probability* (2nd ed.). American Mathematical Society. https://math.dartmouth.edu/~prob/prob/prob.pdf
2. Devore, J. L. (2015). *Probability and statistics for engineering and the sciences* (9th ed.). Cengage Learning.
3. Montgomery, D. C., & Runger, G. C. (2014). *Applied statistics and probability for engineers* (6th ed.). Wiley.

Suggested Reading

1. Seema Acharya, Subhasini Chellappan (2015), *Big Data Analytics*, Wiley.
2. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). *An Introduction to Statistical Learning: with Applications in R*. Springer.
3. Dekking, F. M., & others. (2005). *A Modern Introduction to Probability and Statistics.* Springer Verlag, New York

BLOCK 2
Introduction to Data Analytics

Concepts of Data Analytics

Learning Outcomes

At the end of this unit, the learners will be able to:

- explain the concept of data and its types, including structured, semi-structured, and unstructured data
- describe the stages of data processing and their importance in converting raw data into meaningful information
- discuss the key concepts of data analysis and analytics, including techniques like descriptive, predictive, and prescriptive analytics
- recognize the importance of data quality, governance, and ethical use in effective data-driven decision-making

Prerequisites

You already know that data is everywhere whether on your phone, computer, or the internet. We use data every day when checking social media, shopping online, or tracking the weather. This unit will help you understand what data really is, the different types of data, and how it is prepared and used for analysis. You will also learn why handling large amounts of data, known as Big Data, is important in today's world, and explore how unstructured data, which is not organized in tables, can also provide valuable insights. Before studying this topic, learners should have a basic understanding of computer fundamentals, data storage, and how information supports decision-making. Knowledge of spreadsheet tools like Microsoft Excel and basic mathematical reasoning helps in understanding how raw data is processed and analyzed. The importance of this topic lies in the fact that data powers nearly every modern activity organizations depend on for planning, forecasting, and improving performance. For example, companies like Amazon use customer data to recommend products and enhance user experience, demonstrating how data analytics influences real-world decisions. However, data processing has limitations such as data quality issues, privacy risks, high processing costs, and the potential for misinterpretation if not handled properly. Therefore, having a basic understanding of computing concepts and logical thinking is essential before exploring this topic further.

Keywords

Data Preprocessing, Big Data, Data Warehousing, Data analytics life cycle

Discussion

In today's world, we are surrounded by huge amounts of information, and organisations in every field are collecting more data than ever before. Data analytics is a method that helps make sense of all this data. It allows organisations to find useful information and hidden patterns that can help them improve the way they work. By using data analytics, organisations can better understand their customers, how their business is running, and what changes are happening in the market. This helps them make smart decisions, work more efficiently, come up with new ideas, and stay ahead of their competitors. In short, data analytics helps organisations grow and succeed. It helps them make better use of time, money, and people. As data becomes more important in every area of life and business, knowing how to use it well is a key skill for staying successful in today's fast-changing world.

2.1.1 Concept of Data

Data simply means facts or information that we collect by observing, measuring, researching, or recording something. This can include numbers, names, dates, or descriptions about people, objects, or events. At first, data is just raw information. But when we study and analyse it, we can turn it into useful knowledge. This process gives us something called statistics, which help us understand what the data is trying to tell us.

Data is all around us and helps us understand what's going on in the world. For example, on social media platforms like Facebook, data shows us what topics are trending, how people are feeling, and where they are posting from. This kind of information helps companies plan advertisements and even helps in emergency situations. Data is also useful for checking ideas and making predictions. For example, a college store can look at past records to guess how many books to order for the next academic year.

Learning about Data Science, which is the study of how to collect and work with data and can help us gain useful skills for many jobs. Even though data can sometimes be complex, it is very important because it helps us make better decisions, understand problems, and plan for the future.

2.1.2 Data Usage

We use data in many ways in our daily lives, especially through smart applications that help us make decisions or offer suggestions. Some of the most common uses of data today include:

Forecasting: Forecasting means predicting what might happen in the future by looking at what has happened in the past. For example, weather apps use past weather data to predict if it will rain tomorrow.

Classification: Classification is a machine learning task that involves categorising data points into predefined classes or categories. It aims to assign labels to input data based on its features, enabling the system to distinguish between different classes.

Pattern and Anomaly Detection: Pattern and anomaly detection involves identifying regular patterns or abnormal behaviour within data. It encompasses recognizing recurring patterns or trends as well as detecting deviations or outliers that do not conform to expected behaviour.

Recommendations: Recommendation systems look at your likes, choices, or past activities to suggest things you might like. For example, when YouTube shows you videos similar to what you've watched, or when shopping websites suggest products, they're using recommendations.

Image Recognition: Image recognition is the process of identifying and classifying objects, people, or scenes within images. It involves using machine learning algorithms to analyse visual features and patterns in images to recognize and categorise objects accurately.

2.1.3 Data Processing

Data processing refers to the systematic procedure of collecting, organizing, and transforming raw data into meaningful information that can support decision making. This transformation can be carried out manually by individuals or automatically by machines. The overall process typically follows a structured sequence known as the data processing cycle, which consists of three primary stages: input, processing, and output.

Input: The input stage involves the collection and preparation of data in a format that can be understood and interpreted by a computing system.

Processing: During the processing stage, the collected data is subjected to various operations such as calculations, comparisons, or transformations to convert it into more useful and structured information.

Output: The final stage is the output, where the results of the processing phase are presented in a usable form. This output can take various forms such as reports, charts, documents, or direct actions.

Information and Its Characteristics

Once data has been processed and organized, it becomes information that is, data that is meaningful and relevant to the user. Information is essential for effective decision-making, and in order to be useful, it must meet certain quality criteria:

- **Accuracy:** Information must be correct and free from errors.
- **Completeness:** Information should be whole and not missing any relevant parts.

- **Timeliness:** Information must be available when it is needed to support timely decisions.

Applications of Data Processing

In the current digital environment, data processing plays a crucial role in various applications. Some common examples include:

- **Predictive analysis** is used to guess what might happen in the future. For example, it can help find out when a customer is likely to stop using a service.
- **Customer segmentation** means dividing customers into smaller groups based on things they have in common, like their interests, habits, or age.
- **Fraud detection** is the process of spotting activities that look unusual or suspicious, like someone using your bank card in another country.
- **Recommendation systems** suggest products, videos, or other items that a person might like, based on what they've liked or done before.
- **Pricing strategies** use data to help companies set better prices for their products or services so they can earn more or stay competitive.

2.1.4 Types of Data

Statistical data can be classified in various ways, but one of the most common methods is to divide it into qualitative and quantitative types. Qualitative data refers to non-numerical information that can be grouped into categories. For example, hair colour is qualitative data because it can be categorized as "blond," "brown," "red," and so on.

In contrast, quantitative data involves numerical values that can be measured and compared. For instance, height is considered quantitative data because it can be measured in units such as inches or centimeters. Although there are several ways to organize data, distinguishing between qualitative and quantitative data is fundamental in the study of statistics.

2.1.4.1 Qualitative Data (Categorical Data)

Qualitative data refers to information that describes qualities or characteristics and is not represented by numbers. This type of data is often used to categorize or label items or individuals into different groups.

Types of Qualitative Data:

1. Nominal Data

Nominal data consists of names, labels, or categories that do not follow a specific order. These categories are used to classify data but cannot be arranged in a meaningful sequence. Examples of nominal data include gender (such as male and female), religion (like Hindu, Muslim, or Christian), and nationality (such as Indian or American).

In some cases, numbers may be used to represent different categories, but these numbers do not indicate quantity or order. For instance, a satisfaction rating scale from 1 to 5, where 1 means "Very Unsatisfied" and 5 means "Very Satisfied" can be considered nominal if the numbers are only used to label the satisfaction levels and not to measure actual differences between them.

2. Ordinal Data

Ordinal data is a type of qualitative data that represents categories with a meaningful order or ranking. This means that the data not only categorizes items but also shows a sequence or level of importance. Common examples of ordinal data include social class (such as upper class, middle class, and lower class), opinions or feedback (like excellent, good, and poor), and satisfaction ratings (such as Very Satisfied, Satisfied, Neutral, Unsatisfied, and Very Unsatisfied).

Unlike nominal data, which simply groups items without any order, ordinal data shows a clear progression from one level to another. For instance, in a satisfaction survey, knowing that someone is "Very Satisfied" versus "Neutral" gives more insight because there is an implied ranking in the responses.

2.1.4.2 Quantitative Data (Numerical Data)

Quantitative data, also known as numerical data, refers to information that can be measured and expressed using numbers. This type of data is useful because it allows us to count, compare, and perform calculations. Quantitative data is essential in many areas where precise measurement is needed.

Types of Quantitative Data:

1. Discrete Data

Discrete data consists of values that can be counted and separated into distinct units. Examples include counting the number of people in a room or the number of cars passing on a road. This type of data is typically gathered through surveys or experiments and is often presented using tables or graphs. One benefit of discrete data is that it is easy to understand. However, obtaining accurate results can be challenging when dealing with a small sample size. Additionally, discrete data can only represent a limited set of possible values. For example, if we were counting how many students were in each grade at a school, that data would be discrete because there's a set number of possibilities (from 0 to the most students in any one grade).

2. Continuous Data

Continuous data refers to information that can take any value within a specific range. Unlike discrete data, which consists of distinct, separate values, continuous data flows smoothly like points on a line. Collecting continuous data often requires precise instruments and careful measurement, making it more difficult to gather accurately. Analyzing it can also be challenging due to the possibility of small errors. However, continuous data offers more detailed insights than discrete data and allows for better

predictions and deeper analysis. This makes it especially valuable in areas such as weather forecasting and medical research.

For example, temperature is continuous data because it can be any number within a certain range, like 32.1°F, 32.5°F, or 33°F, and so on.

2.1.5 Data Preprocessing Techniques

Data preprocessing is a vital step in data mining and machine learning that involves converting raw data into a format that can be easily understood and processed by machines. Real world data often contains inconsistencies, inaccuracies, or missing values, which can negatively affect the performance of machine learning models. Preprocessing helps address these issues to improve model accuracy and reliability.

Machines cannot interpret data in the form of text, images, or videos directly; they require numerical representations. Therefore, simply feeding raw images or unprocessed text into a model is ineffective. Data preprocessing ensures that data is structured and formatted in a way that algorithms can interpret and learn from more effectively.

The process of data preprocessing typically includes four main steps:

1. Data Cleaning
2. Data Integration
3. Data Transformation
4. Data Reduction

2.1.5.1 Data Cleaning

Data Cleaning is an essential step in the data preprocessing process, especially because real world data is often incomplete, inconsistent, or irrelevant. Datasets may have missing values, errors, duplicates, or noise, all of which can negatively impact the performance of machine learning models. Cleaning the data helps ensure that the information fed into the model is accurate and meaningful.

1. Handling Missing Data

Missing data is a common issue that can arise during data collection or due to technical errors. There are several ways to handle missing values:

- **Removing Rows or Columns**: If a row or column contains mostly missing values, it is often best to remove it to avoid affecting the analysis.
- **Eliminating Duplicates**: Repeated rows or columns should be removed to prevent bias in the model.
- **Estimating Missing Values**: If only a few values are missing, they can be estimated using techniques like:

- **Mean or Median Imputation** : replacing missing values with the average or median.
- **Mode Imputation** : using the most frequent value for categorical data.

2. Handling Noisy Data

Noisy data includes random errors or irrelevant values that make it difficult for machines to learn effectively. Common techniques for handling noisy data include:

- **Binning**: Data is sorted into bins or intervals, and values within each bin are smoothed, typically by replacing them with the mean or boundary values of the bin.
- **Clustering**: Similar data points are grouped together, and noise or outliers that don't belong to any cluster can be identified and treated.
- **Regression**: A regression function is fitted to the data to identify trends and reduce noise by smoothing the data based on the function's predictions.

2.1.5.2 Data Integration

Data integration is the process of combining data from multiple sources into a unified and consistent view. These sources may include different databases, files, or systems within an organization. The goal is to provide a comprehensive dataset that can be used for analysis or machine learning.

To ensure that data from different sources aligns correctly, metadata which is data about data is used. Metadata helps describe the structure, format, and meaning of the data, making it easier to resolve differences and avoid inconsistencies during integration. By integrating data effectively, organizations can eliminate data silos, improve data consistency, and create a more complete and accurate dataset for analysis or decision-making.

2.1.5.3 Data Transformation

This step involves converting raw data into formats that are more suitable for data mining and machine learning. It ensures the data is consistent, meaningful, and optimized for efficient processing by algorithms. Data transformation improves data quality and enhances model performance, speed, and accuracy. Key techniques include:

1. **Normalization**: Normalization scales data values to fall within a specific range, such as 0.0 to 1.0 or -1.0 to 1.0. This ensures that no single feature dominates due to its scale and helps improve model performance.
2. **Concept Hierarchy Generation**: This involves transforming low-level data into higher-level concepts. For example, raw age values can be grouped into categories like "young," "middle-aged," or "elderly" to simplify analysis.
3. **Smoothing**: Smoothing techniques help remove noise from data. Methods such as binning, clustering, or regression can be used to make data more consistent and easier to interpret.

4. **Aggregation**: Aggregation combines data to produce summaries. For instance, daily sales data can be aggregated into monthly totals to reduce complexity and reveal broader trends.

2.1.5.4 Data Reduction

Data reduction techniques help simplify large datasets while preserving essential information, making processing faster and more efficient. Key techniques are:

1. **Dimensionality Reduction**: This reduces the number of input features or variables. For example, in image processing, hundreds of features may be reduced using techniques like Principal Component Analysis (PCA).

2. **Numerosity Reduction**: This replaces large volumes of data with smaller representations, such as using mathematical models, histograms, or clustering to represent the data.

Preprocessing of Text Data

Preparing text data is a vital step in any natural language processing or machine learning task involving textual input. Since machine learning models require numerical input, raw text must be converted into meaningful features. This involves cleaning the data by removing unnecessary elements such as punctuation, stop words, URLs, and spelling errors. Additionally, techniques like stemming and lemmatization are used to reduce words to their root or base forms, helping to avoid redundancy in feature representation.

Steps for Text Preprocessing

1. **Load the Text Data:** Read the text data from the source (e.g., a file or dataset) and store it in a variable for processing.

2. **Convert to List Format:** If the text is in a DataFrame (e.g., a pandas DataFrame), convert the relevant column into a list using df['column_name']. tolist() for easy iteration.

3. **Initialize Preprocessing Tools:** Set up and configure necessary preprocessing objects or libraries (e.g., using NLTK, spaCy, or custom functions).

4. **Apply Preprocessing Techniques:** Iterate through the list of text entries, applying techniques such as:

    - Removing punctuation and special characters
    - Eliminating stop words
    - Converting text to lowercase
    - Tokenizing the text
    - Performing stemming or lemmatization

5. **Store the Processed Text:** Save the cleaned and processed text in a new list or DataFrame column for further analysis or model training.

2.1.6 Need for Big Data

Big Data refers to extremely large and complex datasets that are generated at high speed from various sources such as business transactions, social media, sensors, and digital devices. It plays a crucial role in modern businesses by enabling advanced analytics, predictive modeling, and machine learning. With the right tools and technologies, organizations can analyze Big Data to gain deeper insights, improve decision making, and enhance operational efficiency.

Technologies that support Big Data analytics, such as Hadoop, Spark, and NoSQL databases, have become essential components of contemporary data management systems. To fully understand the significance of Big Data, it's important to explore its key characteristics, often referred to as the Five V's of Big Data:

1. **Volume**: Volume refers to the massive amount of data generated every second, measured in units like gigabytes, terabytes, zettabytes, or even yottabytes. Managing and processing such huge volumes of data requires distributed storage and parallel computing systems, such as Hadoop and cloud platforms.

2. **Velocity**: Velocity is the speed at which data is generated, transmitted, and processed. With real-time data streaming in from sources like IoT sensors, social media, and online transactions, organizations must have the capability to process and analyze data at high speeds to stay competitive.

3. **Variety**: Big Data comes in various formats structured (like databases), semi-structured (like XML or JSON), and unstructured (like emails, images, and videos). Managing this variety of data types is a major challenge, requiring flexible systems that can accommodate different data sources and formats.

4. **Veracity**: Veracity deals with the trustworthiness and quality of data. Inconsistent or inaccurate data can lead to faulty analyses and poor business decisions. Ensuring data accuracy, reliability, and integrity is critical in Big Data environments.

5. **Value**:Ultimately, the value of Big Data lies in the insights it provides. Data that aligns with business goals can be transformed into actionable intelligence. Data scientists and analysts play a vital role in cleaning, processing, and interpreting data to unlock its full potential and contribute to business growth.

2.1.6.1 Types of Big Data

Big Data comes in various forms, each with its own structure, source, and use case. Understanding these types is essential for selecting appropriate tools and techniques for storage, processing, and analysis. The major types of Big Data are:

1. **Structured Data:** Structured data is highly organized and follows a predefined data model. It is easily stored, retrieved, and processed using traditional relational databases. Structured data typically appears in tabular formats such as rows and columns, making it simple for both humans and machines to understand. Common examples include business transaction records, employee databases, and sales spreadsheets.

2. **Semi-Structured Data:** Semi-structured data does not follow the strict structure of relational databases but still contains identifiable tags or markers to separate elements. This type of data bridges the gap between structured and unstructured data. It is often stored in formats like XML, JSON, or CSV files, where data elements are not organized in a table but still maintain some hierarchical or attribute-based structure.

3. **Unstructured Data:** Unstructured data lacks any predefined format or organization. It is typically generated in large volumes from sources like social media platforms, audio and video content, emails, and instant messages. This type of data is complex and requires advanced processing techniques, such as Natural Language Processing (NLP) or image recognition, to extract meaningful insights. Examples include YouTube videos, WhatsApp messages, or customer reviews on e-commerce sites.

4. **Geospatial Data:** Geospatial data refers to information that includes geographic location components. It captures the position, shape, and features of physical objects on or near the Earth's surface. This data type can be static (e.g., the location of a building) or dynamic (e.g., GPS coordinates of a moving vehicle). Geospatial data is essential in fields like urban planning, navigation, and logistics.

5. **Machine or Operational Logging Data:** Machine-generated data is produced automatically by software applications, devices, or sensors without human intervention. It is typically stored in log files and includes records of events or operations within systems. Examples include server logs, application logs, call detail records, and network monitoring data. Such data is crucial for system diagnostics, performance monitoring, and cybersecurity.

6. **Open-Source Data:** Open source data consists of publicly available datasets that can be freely accessed, used, and modified. These datasets are often published by government bodies, research institutions, or community platforms. They provide cost effective opportunities for data analysis and innovation. Examples include the Google Public Data Explorer, World Bank Open Data, and Kaggle Datasets.

2.1.6.2 Why is Big data used by many organisations?

Big Data plays a vital role in modern business operations by offering insights that drive better decision-making, reduce costs, and improve customer satisfaction. Below are key reasons why organisations increasingly rely on Big Data analytics:

1. **Cost Savings:** Big Data platforms such as Apache Hadoop and Apache Spark offer scalable and cost effective solutions for storing and processing massive volumes of data. By analysing return patterns, for instance, companies can predict and reduce return related costs often 1.5 times higher than the original shipping cost by taking preventive actions.

2. **Driving efficiency:** Big Data tools help companies quickly look at information coming from many places like sensors, websites, and social media. This helps them make faster decisions. These tools can also do some regular tasks automatically, so employees have more time to do important work. This makes the company work better and faster.

3. **Analysing the market:** Big Data helps companies understand what's happening in the market and what customers like. By looking at what people buy, businesses can find out which products are popular and guess what customers might want next. This helps them stay ahead of their competitors and build better relationships with both customers and suppliers.

4. **Enhancing Customer Experience:** Big Data helps businesses understand what customers like by studying their feedback, shopping habits, and how they interact with the company. This allows businesses to offer products and ads that match customer interests. As a result, they spend less money on useless ads and keep customers happy and loyal.

5. **Supporting Innovation:** Big Data helps companies come up with new ideas by using real feedback and customer behaviour. By watching the market in real-time, businesses can quickly improve or create products and stay ahead of their competitors.

6. **Detecting fraud:** In areas like banking and government, Big Data helps find fraud. With the help of Artificial Intelligence (AI) and Machine Learning (ML), it can spot unusual activities in transactions, helping organisations stop fraud and keep services safe.

7. **Improving Productivity:** Modern Big Data tools help data scientists and analysts work faster by quickly analysing large amounts of data. This means they can find useful insights sooner, finish projects faster, and get better results.

8. **Enabling Agility:** Big Data helps businesses react quickly to changes. By studying customer data, they can spot new trends early and change their plans faster. It also helps them manage risks better and keep improving their products and services.

In summary, Big Data helps businesses make faster and smarter decisions. It gives them a better understanding of what customers need, helps solve customer problems quickly, and allows them to stay ahead of their competitors in the market.

2.1.7 Data Warehousing

A Data Warehouse is different from a regular database. It is designed to store large volumes of data collected from various sources like files, other databases, and external systems. The main purpose of a data warehouse is to provide meaningful insights and support decision-making through data analysis.

For example, a college can use a data warehouse to quickly find out how many computer science students got placed in jobs over the past ten years, including information like salary ranges and the number of students placed each year.

While regular databases are used for daily operations and can handle small to medium amounts of data, a data warehouse is used to store and analyze large amounts of historical data. Unlike transactional databases that focus on day to day tasks, data warehouses are optimized for analytics, reporting, and identifying long-term trends, helping organizations make better strategic decisions.

2.1.7.1 Benefits of Data Warehousing

1. **Better Business Analysis:** A data warehouse allows businesses to look at old data and find useful patterns or trends. This helps them make smarter decisions based on facts, not guesses.

2. **Faster Query Performance:** Data warehouses are designed to handle big and complex questions (queries). They can process information much faster than normal databases, saving time.

3. **Improved Data Quality:** Data stored in a warehouse is cleaned and organised. Since it's kept separate from other sources, it stays accurate and consistent, which makes analysis more reliable.

4. **Access to Historical Data:** With a data warehouse, companies can keep years of past data. This helps them review how their business performed over time and find helpful insights whenever needed.

2.1.7.2 Features of Data Warehousing

1. **Data Integration:** A data warehouse combines information from many different sources into one place. This helps remove duplicates and mismatched data, making everything easier to manage and understand.

2. **Historical Data Storage:** It stores old data over time. This lets businesses look back, find trends, and make better plans for the future.

3. **Query and Analysis:** Data warehouses come with tools that let users search, explore, and study the data in different ways. This helps in spotting patterns and making smart decisions.

4. **Data Transformation:** Before storing, the data is cleaned, filtered, and formatted. This makes sure the information is accurate, consistent, and ready to use.

5. **Data Mining:** A data warehouse helps in digging deep into data to find hidden patterns and connections. This can help businesses spot new chances, predict future trends, and avoid risks.

6. **Data Security:** It uses strong security methods like passwords, encryption, and backups to protect the data from being lost or seen by the wrong people.

2.1.7.3 Limitations of Data Warehousing

1. **High Cost:** Building a data warehouse can be expensive. It requires powerful computers, specialised software, and skilled professionals to manage it.

2. **Skilled Personnel Required:** Organisations need to hire experts like data engineers and administrators to make sure the system works smoothly and efficiently.

3. **Time-Consuming Setup:** Setting up a data warehouse takes a lot of time. It needs careful planning, design, and testing before it can be used effectively.

4. **Data Integration Challenges:** Combining data from different sources can be difficult. It takes effort to make sure all the information is accurate and fits well together.

5. **Security Concerns:** Data warehouses often store sensitive and important information. Strong security measures are needed to protect the data from unauthorised access or loss.

2.1.8 Data Analytics Life Cycle

In today's digital world, data is like gold! Just like people go through different stages in life, data also follows a journey. It gets created, cleaned, analysed, and used in many ways. To make this process easier and more organised, experts follow a step by step guide called the data analytics life cycle. Each phase in this cycle plays an important role in turning raw data into useful insights.

2.1.8.1 Phases in Data Analytics Life Cycle

Phase 1: Discovery (What's the Problem?)

In this phase, the team talks with people involved in the project (stakeholders) to understand the main goals, problems, and what they expect. They try to figure out the important questions that need answers and what a successful result would look like. The team also checks what resources they have, like where the data will come from, which tools and technologies they can use, and the skills of the team members.

Phase 2: Data Preparation (Get it Ready!)

Once the data is found, it is collected from different places and made ready for analysis. This step includes cleaning the data, fixing errors, changing formats, and combining data from different sources. The team removes anything that could affect the results, like wrong values, missing information, or repeated records. Data engineers and analysts work together to make sure the data is clean and ready to use. Tools like Hadoop, OpenRefine, and Excel help make this job easier and faster.

Phase 3: Model Planning (Pick the Right Tools)

In this phase, the team studies the data to find patterns, trends, and relationships. This process is called Exploratory Data Analysis (EDA). Based on what they learn, they decide which methods or machine learning models will work best. They also divide the data into three parts: one for training the model, one for testing it, and one for final checking (validation). Tools like MATLAB, STATISTICA, or Python (using Pandas and Scikit-learn) help in this process. Making a good plan at this stage is very important because it helps build a strong and accurate model later.

Phase 4: Model Building (Make it Work!)

In this phase, the team builds the actual models using the prepared data. They create different sets of data for training, testing, and checking the model. The team makes sure the tools and computers they use are strong enough to handle the work. Both free tools like WEKA and R, and paid tools like MATLAB, can be used to build and test the models. The goal is to create a model that gives accurate and reliable results.

Phase 5: Share the Results (Tell Everyone!)

In this phase, the team checks how well the model is working and looks at the results. They find the most important insights and measure how useful they are. Then, they explain everything clearly to stakeholders (people involved in the project) using simple language, charts, and visuals. The goal is to help everyone understand the findings and make smart decisions based on them.

Phase 6: Put it to Use (Spread the Benefits!)

In this final phase, the team starts by using the model in a small test project, called a pilot. This helps them find any problems and make improvements before using it on a larger scale. Once the model works well, they use it in the real environment. They also share final reports, presentations, and the code using open-source tools like WEKA and SQL, so others can use and understand the work.

2.1.9 Analytics for Unstructured Data

What is Unstructured Data?

Not all data fits neatly into rows and columns like in a spreadsheet. Think about emails, social media posts, videos, images, or PDFs, this type of data is called unstructured data. It doesn't follow a fixed format and can be messy or scattered. Even though it's tricky to manage, unstructured data can reveal valuable insights. Businesses use it to understand customer behavior, predict future trends, and make smart decisions.

Common Sources of Unstructured Data

- Emails
- PDF documents
- Audio files
- Social media posts
- Images
- Videos

Why is it Challenging?

Unstructured data is hard to work with because it doesn't follow a fixed format. It's like puzzle pieces scattered around the information is there, but it's not organised neatly. Regular tools can't easily read or understand this kind of messy data.

2.1.9.1 How to Analyse Unstructured Data (Key Steps & Tips)

1. **Prepare the Data:** Before analysing unstructured data like emails or videos, you need to get it ready. This means organising and cleaning the data so that it can be easily used for analysis. If the data is messy or incomplete, it won't give good results.

2. **Know Your Goals:** Be clear about what you want to find out. For example, if you want to identify a face in a video, focus on features like eye colour or face shape. Store this information in easy to use formats like JSON. Tools like MongoDB help manage this kind of data.

3. **Find Data Faster:** Use metadata, which is information about your data, to quickly find what you need. For example, if you have thousands of documents, metadata like the title or author can help you search and find the right files much faster.

4. **Choose the Right Tools:** Different problems need different solutions. For example, if you want to detect theft in CCTV videos, you need advanced tools like deep learning. But for simpler tasks, like grouping families based on their milk purchases, simple analysis is enough. Always pick the tools that fit your task.

5. **Pick Reliable Sources:** Not all data is useful. Choose data sources that are trustworthy and relevant to your problem. For example, if you have data from social media, devices, and recordings, decide which sources are important for your analysis and focus on those. Data lakes like MongoDB Atlas Data Lake help combine data from many places.

6. **Evaluate Your Technologies:** Choose tools that can handle large amounts of data and give you the answers you need. Some tools are very powerful

but also costly and complex. Look for tools that offer features like predictive analytics, easy data searching, and integration with other systems.

7. **Get Real-Time Insights:** Sometimes you need answers quickly, like for preventing fraud or sending personalised offers. Tools like MongoDB can collect data from many sources and analyse it fast, giving you up-to-date information when you need it.

8. **Store and Integrate in Data Lakes:** Data lakes are special storage systems that keep all your unstructured data in its original form. For example, MongoDB Atlas Data Lake lets you search and analyse all your data from one place, including files stored in cloud services like Amazon S3, which makes your work easier and faster.

9. **Clean Up the Mess:** Make sure your data is clean before analysis. If there is too much noise or irrelevant information, your results will not be accurate. Cleaning your data helps you get better and more trustworthy insights.

Recap

Concept of Data

- Data refers to raw facts or information collected through observation, measurement, or research.
- It can be numbers, names, dates, or descriptions.

Data Usage

- Data is used in everyday applications such as:
- Forecasting (e.g., weather prediction)
- Classification (e.g., sorting emails into spam/not spam)
- Pattern & Anomaly Detection (e.g., detecting unusual behavior)
- Recommendation Systems (e.g., YouTube, shopping suggestions)
- Image Recognition (e.g., face detection in photos)

Data Processing

- Data Processing is converting raw data into meaningful information.
- It has three stages:
    - Input: Collecting and preparing data.
    - Processing: Performing calculations and transformations.
    - Output: Presenting results (e.g., reports, charts).

Types of Data

- Data is mainly classified into:
- Qualitative (Categorical): Non-numerical, based on categories.
  - Nominal: No specific order (e.g., gender, religion)
  - Ordinal: Ordered categories (e.g., satisfaction levels)
- Quantitative (Numerical): Measurable values.
  - Discrete: Countable values (e.g., number of students)
  - Continuous: Measurable on a continuous scale (e.g., temperature)

Data Preprocessing

Involves preparing raw data for machine learning or analysis.

Includes four key steps:

1. Data Cleaning
   - Handling missing data (removal or imputation)
   - Removing duplicates
   - Reducing noise (using binning, clustering, regression)
2. Data Integration
   - Combining data from multiple sources
   - Using metadata to resolve inconsistencies
3. Data Transformation : It is the process of converting data into a suitable format for analysis.
4. Data Reduction: It involves minimizing the volume of data while preserving its integrity and analytical value.
   - Big Data involves massive, fast, and diverse data sets (Five V's: Volume, Velocity, Variety, Veracity, Value).
   - Types of Big Data:
     - Structured (databases)
     - Semi-structured (XML, JSON)
     - Unstructured (videos, emails)
     - Geospatial data
     - Machine/operational logs
     - Open-source data

Data Warehousing

- Stores integrated data from multiple sources for analysis.
- Features:
    - Data integration and cleaning
    - Historical data storage
    - Query and analysis tools
    - Data mining capabilities
    - Security and backup mechanisms

Data Analytics Life Cycle

- Phases:

1. Discovery: Define objectives and resources
2. Data Preparation: Clean and format data
3. Model Planning: Explore data, select algorithms
4. Model Building: Create and validate models
5. Share Results: Present insights effectively
6. Put it to Use: Deploy and monitor models

Analytics for Unstructured Data

- Unstructured data includes emails, social media, images, videos.
- Challenges: No predefined format, harder to process.
- Requires specialized tools and methods to extract valuable insights.
- Helps in understanding customer sentiment, behavior, and trends.

Objective Type Questions

1. What is the raw fact that has no meaning on its own?
2. What is the term for converting raw data into meaningful information?
3. What type of data is expressed in numbers?
4. What type of data describes categories or labels?

5. What is the technique of handling missing values called?
6. Which step combines data from different sources?
7. What type of data is stored in a data warehouse?
8. What is the term for extremely large and complex data sets?
9. Which type of data includes videos and emails?
10. What is the first phase in the data analytics life cycle?

Answers to Objective Type Questions

1. Data
2. Processing
3. Quantitative
4. Qualitative
5. Imputation
6. Integration
7. Historical
8. BigData
9. Unstructured
10. Discovery

Assignments

1. List and describe any four common types of data with examples.
2. What are the main steps in data preprocessing? Briefly explain each.
3. Differentiate between qualitative data and quantitative data with suitable examples.
4. Differentiate between structured and unstructured data with examples.
5. Explain the different phases of the Data Analytics Life Cycle.

Reference

1. Provost, F., & Fawcett, T. (2013). *Data Science for Business: What You Need to Know About Data Mining and Data-Analytic Thinking.* O'Reilly Media.

2. Han, J., Kamber, M., & Pei, J. (2022). *Data Mining: Concepts and Techniques* (4th ed.). Morgan Kaufmann.

3. Kotu, V., & Deshpande, B. (2019). *Data Science: Concepts and Practice* (2nd ed.). Morgan Kaufmann.

4. Shmueli, G., Bruce, P. C., Gedeck, P., & Patel, N. R. (2020). *Data Mining for Business Analytics: Concepts, Techniques, and Applications in Python.* Wiley.

5. Marr, B. (2016). *Big Data in Practice: How 45 Successful Companies Used Big Data Analytics to Deliver Extraordinary Results.* Wiley.

Suggested Reading

1. Agarwal, B. L. (2013). *Basic statistics*. New Age International Publishers

2. Bhat, B. R., Sri Venkata Ramana T, & Rao Madhava K. S. (1977). *Statistics: A beginners Text Vol. 2*. New Age International (P) Ltd., New Delhi.

3. Dekking, F. M., & others. (2005). *A Modern Introduction to Probability and Statistics.* Springer Verlag, New York.

4. Seema Acharya, Subhasini Chellappan (2015), *Big Data Analytics,* Wiley

Familiarisation of Different Algorithms for Data Analytics

Learning Outcomes

At the end of this unit, the learners will be able to:

- familiarize with the basic principles of classification algorithms
- recognize the key features and components of clustering algorithms
- understand the fundamental steps involved in clustering methods
- introduce the essential steps in model evaluation techniques

Prerequisites

Artificial Intelligence (AI) and Machine Learning (ML) are closely connected fields that lie at the heart of modern technological advancements. AI refers to the broader concept of developing intelligent systems capable of performing tasks that typically require human intelligence. These tasks span diverse areas, including speech recognition, natural language processing, computer vision, and autonomous decision-making. Within this broad field, Machine Learning serves as a key subset, focusing on the development of algorithms that enable systems to learn from data and make informed decisions or predictions without being explicitly programmed. By continuously training on data, ML models enhance their accuracy and uncover meaningful patterns that drive intelligent behavior. Collectively, AI and ML are transforming industries such as healthcare, finance, and transportation, and are redefining the ways we interact with technology in everyday life.

Machine Learning, as a branch of computer science, empowers machines to learn and adapt from experience autonomously. It revolves around the use of algorithms—structured sets of instructions—that analyze data and support problem-solving. These algorithms enable systems to recognize patterns and make predictions across a wide range of applications, from weather prediction to customized recommendations. Although the subject may seem complex at first, our journey begins with a focus on fundamental principles. Get ready to explore the fascinating world of machine learning as we embark on this academic journey together.

Keywords

Naive Bayes Classification, Decision tree classification, K-means Clustering, Agglomerative Clustering, Hyperparameter Tuning

Discussion

2.2.1 Introduction to Machine Learning

Consider a scenario in which a retail company seeks to optimise its marketing strategy by identifying customer segments based on their purchasing behaviour. Utilising machine learning algorithms, the company can analyse historical transaction data to categorise customers into distinct groups, each exhibiting unique preferences and tendencies. Classification algorithms such as the Naïve Bayes Classifier and the Decision Trees Classifier enable the company to assign customers to predefined segments, thereby facilitating targeted marketing campaigns tailored to specific consumer demographics.

On the other hand, envision a hospital that wants to improve how it treats patients by studying their medical records. The goal is to group patients with similar health conditions so that doctors can give them the most suitable care.To do this, the hospital uses special computer techniques called clustering algorithms like K-Means, DBSCAN, and Agglomerative Clustering. These tools help find hidden patterns in large sets of patient data.Once the patients are grouped into clusters based on similar health features, doctors can better understand what each group needs. This allows them to create personalised treatment plans for each group, leading to better care, quicker recovery, and higher patient satisfaction.

In essence, the integration of classification and clustering algorithms into AI systems empowers organisations across diverse industries to extract actionable insights from complex datasets, driving innovation and informed decision-making. As illustrated, machine learning algorithms serve as invaluable tools in unlocking the potential of data, enabling organisations to gain a deeper understanding of their customers, patients, and stakeholders, and driving progress in the digital era.

2.2.2 Machine Learning Paradigms

Machine learning is a key part of modern artificial intelligence (AI) that allows computers to learn from data and make decisions on their own. Two important types of machine learning techniques are classification and clustering. Classification is used to sort data into known categories, like deciding if an email is spam or not. Clustering, on the other hand, groups similar data together without predefined labels, like grouping customers with similar buying habits. These methods help us organize, understand, and make predictions from data in a simple and effective way.

Classification algorithms, within the realm of supervised learning, are designed to assign input data points to predefined categories or classes based on their features. These algorithms leverage labelled training data to learn the underlying patterns

and relationships between input features and output labels. Noteworthy examples of classification algorithms include the Naïve Bayes Classifier and the Decision Trees Classifier. Naive Bayes relies on Bayes' theorem and conditional independence assumptions to calculate the probability of each class given the input features, making it particularly suitable for text classification tasks. On the other hand, Decision Trees work by asking a series of questions to split the data into groups. This step-by-step process helps sort items into different categories. They are easy to understand and explain because the decisions are shown clearly, like a flowchart.

In contrast, clustering algorithms operate in an unsupervised learning setting, where the objective is to partition the input data into cohesive groups or clusters based on their intrinsic similarities. These algorithms do not require labelled data and instead focus on uncovering hidden structures within the data itself. Key clustering algorithms include K-Means, DBSCAN (Density-Based Spatial Clustering of Applications with Noise), and Agglomeration Clustering. K-Means clustering divides the data into a predetermined number of clusters by iteratively optimising cluster centroids to minimise the within-cluster variance. DBSCAN, on the other hand, identifies clusters based on density connectivity, effectively capturing irregularly shaped clusters and handling noise. Agglomeration clustering, also known as hierarchical clustering, constructs a hierarchy of clusters by iteratively merging the most similar data points or clusters until a termination criterion is met, offering flexibility in cluster interpretation and analysis.

Classification and clustering are important types of machine learning algorithms that help computers automatically organize and understand data. By using these algorithms, AI systems can find patterns, make decisions, and solve problems without human intervention. These techniques are useful in many areas, such as healthcare, finance, marketing, and engineering, helping people make better decisions and discover new insights from data.

2.2.3 Naive Bayes Classifier: Unveiling Probabilistic Classification

Suppose we have a dataset containing emails labelled as either "spam" or "not spam," along with the presence of specific keywords within each email. To classify a new email as either spam or not spam, the Naive Bayes Classifier calculates the probability of the email belonging to each class based on the occurrence of keywords. Given a new email with the words "discount," "offer," and "sale," the classifier computes the likelihood of it being spam or not spam based on the frequency of these words in both categories. By applying Bayes' theorem, the classifier assigns the email to the class with the highest probability, thereby determining its classification.

The Naive Bayes Classifier is a probabilistic algorithm grounded in Bayes' theorem, which calculates the probability of a hypothesis given observed evidence. Despite its simplistic assumptions, the Naïve Bayes Classifier is renowned for its efficiency and effectiveness in various classification tasks, particularly in natural language processing and spam filtering.

2.2.3.1 Naive Bayes ClassifierAlgorithm

1. Start
2. Input: Training dataset with labelled examples New instance to classify
3. Preprocessing: Calculate prior probabilities and likelihoods
4. For each class c:

   Calculate prior probability P(c)

5. For each feature xi:

   Calculate likelihood P(xi|c)

6. Calculate posterior probabilities for each class
7. Select the class with the highest posterior probability as the predicted class for the new instance
8. Output: Predicted class for the new instance
9. End

2.2.4 Decision Trees Classifier: Navigating Decision-making through Tree Structures

Imagine a scenario in which a loan officer needs to decide whether to approve or deny a loan application based on various attributes such as income, credit score, and employment status. Utilising a decision tree classifier, the loan officer can construct a tree structure where each node represents a decision based on a specific attribute. For instance, the first node may split the dataset based on the applicant's credit score, with subsequent nodes further dividing the data based on income and employment status. Ultimately, the decision tree leads to leaf nodes corresponding to the final loan approval decision, providing a transparent and interpretable framework for loan application assessment.

The Decision Trees Classifier employs a hierarchical tree structure to partition the feature space into regions, facilitating decision-making based on a series of if-else conditions. This algorithm is prized for its interpretability, as decision trees can be visually represented, offering intuitive insights into the decision-making process.

2.2.4.1 Decision Trees ClassifierAlgorithm

1. Start

2. Input: Training dataset with labeled examples New instance to classify

3. Preprocessing: Select attribute for the root node based on a splitting criterion (e.g., information gain)

4. If stopping criteria met (e.g., maximum tree depth reached, no further split possible):

   Output majority class label for leaf node

5. Else:

   Split data based on selected attribute

   Recur on each subset until stopping criteria met

6. Output: Predicted class for the new instance based on traversing the decision tree

7. End

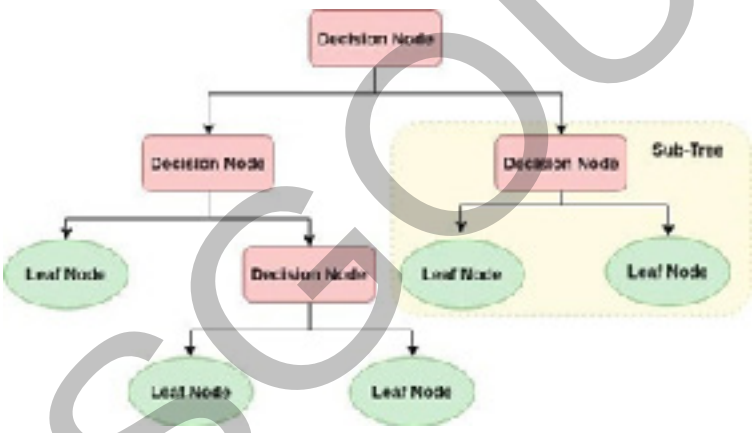


Fig. 2.2.1 Decision tree classification

2.2.5 Understanding Partition Clustering Algorithms: A Comprehensive Overview

Clustering algorithms serve as indispensable tools in unsupervised machine learning, facilitating the grouping of similar data points without prior knowledge of group assignments. Among these algorithms, partition clustering stands out for its ability to partition data into distinct groups. In this discourse, we delve into three eminent partition clustering algorithms: K-Means, Density-Based Spatial Clustering of Applications with Noise (DBSCAN), and Agglomerative Clustering, shedding light on their intricate methodologies and illustrative applications.

2.2.5.1 K-Means Clustering

Imagine you have a bag of coloured marbles, and you want to group them based on their colours. Here's how K-Means clustering works:

1. **Initialization:** First, decide how many groups (K) you want to create. Let's say you want to separate the marbles into three groups based on colour.

2. **Group Assignment:** Randomly pick three marbles to represent the centres of your groups. Now, for each marble in the bag, assign it to the group whose centre it's closest to.

3. **Update Centers:** Once all marbles are assigned, recalculate the centre of each group by averaging the positions of the marbles in that group.

4. **Repeat:** Keep repeating steps 2 and 3 until the centres of the groups stop moving significantly. You've now clustered your marbles based on colour.

Suppose we have a dataset of customer purchases with attributes like spending habits and product preferences. K-Means can group customers into clusters based on similarities in their purchasing behaviour, helping businesses target specific customer segments more effectively.

In K-Means clustering, data points are partitioned into K clusters based on their proximity to cluster centroids. Initially, K centroids are randomly selected, and each data point is assigned to the nearest centroid. The centroids are then recalculated based on the mean of the data points in each cluster, and the assignment process is repeated iteratively until convergence. The final result is well-defined clusters that minimise intra-cluster variance.

In a retail industry, K-Means clustering can be utilised to segment customers based on their purchasing behaviour. For instance, a supermarket chain may use K-Means to group customers into clusters such as frequent buyers, occasional shoppers, and seasonal purchasers. This segmentation enables targeted marketing strategies, personalised promotions, and tailored product recommendations, thereby enhancing customer satisfaction and loyalty.

2.2.5.2 Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

Now, imagine you have a map of a city with various landmarks like parks, schools, and shopping malls. You want to categorize these landmarks according to how close they are to one another. Here's how DBSCAN works:

1. Core Points: Choose a starting landmark and draw a circle around it. If there are enough other landmarks (based on a minimum threshold) within this circle, mark this landmark as a "core point."

2. Expand Clusters: Now, expand the cluster by finding all other landmarks within the circle of the core point. If any of these landmarks are also core points, expand the cluster further.

3. Noise Detection: If a landmark is not within the circle of any core point, it's considered an "outlier" or "noise."

4. Repeat: Keep repeating steps 1 to 3 for each landmark until all landmarks are assigned to clusters or marked as noise.

Imagine a scenario where urban planners need to identify areas of high population density for infrastructural development. DBSCAN can be employed to analyse geographical data, distinguishing densely populated regions from sparsely populated ones. By identifying clusters of high-density areas, urban planners can make informed decisions regarding the allocation of resources, construction of amenities, and implementation of public services.

DBSCAN identifies clusters in a dataset based on density connectivity, where clusters are areas of high data density separated by areas of low density. The algorithm requires two parameters: epsilon (ε), which defines the maximum distance between two points for them to be considered neighbours, and minPts, which specifies the minimum number of points required to form a dense region. DBSCAN iteratively expands clusters from core points until all points are either assigned to a cluster or marked as noise.

2.2.6 Agglomerative Clustering

Agglomerative clustering is a hierarchical clustering technique that starts by considering each data point as a single cluster and then merges the closest pairs of clusters iteratively until only one cluster remains. Here's an illustrative explanation of the agglomerative clustering process:

1. Initialization: Begin with each data point as a single cluster. Calculate the distance or dissimilarity between all pairs of clusters.

2. Merge closest clusters: Identify the two closest clusters based on a chosen distance metric (e.g., Euclidean distance). Merge these clusters into a single cluster.

3. Update distance matrix: Recalculate the distance matrix to reflect the new cluster configurations. There are different linkage criteria for updating the distance matrix, including single linkage (minimum distance between points in different clusters), complete linkage (maximum distance between points in different clusters), and average linkage (average distance between points in different clusters).

4. Repeat: Repeat steps 2 and 3 until only one cluster remains, or until a specified number of clusters is reached.

5. Dendrogram: A dendrogram can visually represent the hierarchical clustering process. It displays the sequence of cluster merges and allows for the identification of clusters at different levels of granularity.

The agglomerative clustering process can be visually represented using a dendrogram, which shows the hierarchical relationships between clusters and allows for the interpretation of cluster structures at different levels of granularity.

Agglomerative clustering is a hierarchical clustering method that starts with each data point as a separate cluster and merges the closest clusters iteratively until only one cluster remains. The algorithm uses a linkage criterion, such as single linkage, complete linkage, or average linkage, to determine the distance between clusters. The result is a dendrogram that illustrates the hierarchical relationships between clusters.

2.2.7 Regression

Regression analysis serves as a powerful tool in statistics, offering insights into the relationships between a dependent variable (the one we aim to predict) and one or more independent variables (those used to guide the prediction). Three primary regression models are : Linear, Gaussian, and Polynomial Regression.

2.2.7.1 Linear Regression

Imagine investigating the connection between house size (independent variable) and selling price (dependent variable). Linear regression assumes a simple, straight-line relationship between these factors. Its algorithm seeks the optimal straight line that minimises the distance between actual data points and their predicted values. This line can be expressed as:

$$y = mx + b$$

where:

- y represents the predicted value of the dependent variable
- x embodies the independent variable
- m embodies the slope, reflecting the change in y for a unit change in x
- b represents the y-intercept, the value of y when x is zero

2.2.7.2 Gaussian Regression

Gaussian regression, also known as Gaussian process regression, is a non-parametric technique used to model the relationship between variables. Unlike linear regression, which assumes a linear relationship, Gaussian regression is adept at handling non-linear relationships. It operates under the premise of a Gaussian distribution, assuming that the data follows a smooth underlying function. In Gaussian regression, errors around

predictions are expected to conform to a bell-shaped curve, allowing for the optimization of a smooth curve that minimises the disparity between predicted and actual values. This method is particularly advantageous when dealing with data that doesn't adhere to a linear pattern or is affected by noise. Gaussian regression excels in scenarios where the relationship between variables is complex and not well-defined, making it a valuable tool in machine learning applications, especially with small datasets or instances where the underlying function is nonlinear.

2.2.7.3 Polynomial Regression

Polynomial regression is a method used in regression analysis to model the relationship between the independent variable x and the dependent variable y as an n-th degree polynomial. The equation of a polynomial regression model is represented as

$y = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$, where $a_0, a_1, \ldots, a_n$ are the coefficients of the polynomial terms.

The assumption of polynomial regression is that the relationship between variables can be adequately captured by a polynomial function. However, a major limitation is that the polynomial regression may suffer from overfitting if the degree of the polynomial is too high. This means the model considers the noise in the data while model building rather than just the underlying pattern.Polynomial regression finds applications in various fields such as physics, engineering, and biology, where the relationship between variables is non-linear and can be approximated by a polynomial function. It is particularly useful in curve fitting and trend analysis, allowing practitioners to analyse and predict trends in datasets that exhibit non-linear behaviour. However, careful consideration must be given to selecting the appropriate degree of the polynomial to avoid overfitting and ensure the model's reliability and generalizability.

2.2.8 Model Evaluation

Just like judging a cake without tasting it wouldn't give you the full picture, evaluating your machine learning model solely on its creation leaves crucial questions unanswered. Evaluating your model's performance unlocks its true potential. It reveals how well it generalises unseen data, identifies hidden weaknesses, and ultimately helps you make informed decisions based on its predictions.

In the realm of machine learning, model evaluation plays a crucial role in assessing a model's effectiveness and optimising its performance. Two key techniques employed in this process are cross-validation and hyperparameter tuning.

2.2.8.1 Cross-Validation

Cross-validation is a robust technique for evaluating the generalisation capability of a machine learning model. It aims to estimate how well the model performs on unseen data, which is crucial for real-world applications.

Here's how it works:

- Split the data: The dataset is divided into folds (usually k folds, like k = 5 or 10).
- Iterative training and testing:

For each fold:

- One fold is held out as the test set for evaluation.
- The remaining k-1 folds are combined as the training set.
- The model is trained on this training set.
- The trained model is then used to predict on the held-out test set.
- This process is repeated for each fold such that every data point gets a chance to be in the test set.

Performance evaluation: The performance of the model is averaged across all the folds, providing a more reliable estimate of its generalisation ability than a single train-test split.

2.2.8.2 Hyperparameter Tuning

Hyperparameters are high-level settings of a machine learning algorithm that control its learning process but are not learned from the data itself. Examples include learning rate, number of hidden layers in a neural network, or the regularisation parameter in a support vector machine.

Hyperparameter tuning is the process of finding the optimal values for these hyperparameters to improve the model's performance. It involves an iterative process of:

- Defining a search space: This involves identifying the range of possible values for each hyperparameter.
- Selecting a search method: Various techniques exist, such as grid search, random search, or Bayesian optimization, to explore the search space efficiently.
- Training and evaluating the model: For each combination of hyperparameter values, the model is trained and evaluated using cross-validation.
- Identifying the best hyperparameters: The combination that yields the best performance on the validation set is chosen as the optimal set.
- By effectively utilising both cross-validation and hyperparameter tuning, you can build robust and generalizable machine learning models that perform well on unseen data.

Recap

1. Machine Learning Paradigms:
    - Classification: Supervised learning, assigns data points to predefined classes.
    - Clustering: Unsupervised learning, groups data based on similarities.
2. Naïve Bayes Classifier:
    - Spam classification using probability of keyword occurrence.
    - Efficient, grounded in Bayes' theorem for text and spam tasks.
3. Decision Trees Classifier:
    - Transparent, interpretable framework for decision-making.
4. Partition Clustering Algorithms:
    - Essential in unsupervised learning (K-Means, DBSCAN, Agglomeration).
    - K-Means: Proximity-based clustering for customer segmentation.
5. K-Means Clustering:
    - Groups data points into K clusters based on proximity.
    - Iterative process for convergence, effective for customer segmentation.
6. DBSCAN:
    - Identifies clusters based on density connectivity.
    - Useful for urban planning to identify high-density areas.
7. Agglomerative Clustering:
    - Merges closest clusters hierarchically.
    - Visualized with dendrogram, useful for hierarchical relationships.
8. Regression:
    - Models relationships between variables (Linear, Gaussian, Polynomial).
9. Linear Regression:
    - Assumes a simple, straight-line relationship.
    - Equation: $y = mx + b$.

10. Gaussian Regression:

- Non-parametric, handles non-linear relationships.
- Assumes Gaussian distribution for errors around predictions.

11. Polynomial Regression:

- Models as an n-th degree polynomial.
- Useful in various fields for non-linear relationships.

12. Model Evaluation:

- Crucial for assessing effectiveness (Cross-validation, Hyperparameter tuning).
- Cross-validation: Iterative training and testing for generalisation ability.
- Hyperparameter tuning: Find optimal settings for improved performance.

Objective Type Questions

1. Which machine learning paradigm assigns data points to predefined classes based on features?
2. Name two classification algorithms mentioned in the text.
3. In which type of learning do clustering algorithms operate?
4. What is the key objective of the Naïve Bayes Classifier?
5. Which algorithm is used for loan approval decisions based on hierarchical decision nodes?
6. What is the primary purpose of partition clustering algorithms?
7. Name one clustering algorithm used for customer segmentation in retail.
8. How does K-Means clustering assign data points to clusters?
9. Which clustering algorithm is useful for identifying high-density areas in urban planning?
10. What does DBSCAN stand for?

11. How does Agglomerative Clustering merge clusters?

12. What type of regression assumes a linear relationship between variables?

13. What is the equation for Linear Regression?

14. Which regression technique handles non-linear relationships and assumes a Gaussian distribution for errors?

15. What is the primary purpose of model evaluation in machine learning?

16. What technique is used for estimating how well a model performs on unseen data?

17. What is hyperparameter tuning used for in machine learning?

18. Name one application of polynomial regression.

19. Which clustering algorithm starts with each data point as a separate cluster and merges them iteratively?

Answers to Objective Type Questions

1. Classification.
2. Naïve Bayes and Decision Trees.
3. Unsupervised learning.
4. To calculate the probability of data belonging to each class.
5. Decision Trees Classifier.
6. To group similar data points without prior knowledge of group assignments.
7. K-Means.
8. Based on proximity to cluster centroids.
9. DBSCAN (Density-Based Spatial Clustering of Applications with Noise).
10. Density-Based Spatial Clustering of Applications with Noise.
11. Hierarchically, by merging closest clusters.
12. Linear Regression.

13. $y = mx + b$.

14. Gaussian Regression.

15. To assess the effectiveness and performance of the model.

16. Cross-validation.

17. To find optimal settings for improving model performance.

18. Curve fitting and trend analysis.

19. Agglomerative Clustering.

Assignments

1. Explain the difference between classification and clustering algorithms. Provide examples of how each type of algorithm is applied in real-world scenarios, drawing from the examples given in the text.

2. Choose one classification algorithm and one clustering algorithm mentioned in the text. Describe how each algorithm works and provide a detailed explanation of how they are applied in the specific examples of retail marketing optimization and healthcare treatment personalization.

3. Compare and contrast linear regression, Gaussian regression, and polynomial regression. Discuss the strengths and limitations of each type of regression and provide examples of real-world applications where each type would be most suitable.

4. Explain the importance of model evaluation in machine learning. Discuss two model evaluation techniques mentioned in the text (cross-validation and hyperparameter tuning) and describe how they are used to assess and improve the performance of machine learning models.

5. Imagine you are a data scientist working for a retail company. Design a machine learning project to optimise the company's marketing strategy using classification and clustering algorithms. Outline the steps you would take, including data preprocessing, algorithm selection, model evaluation, and implementation of the optimised strategy.

Reference

1. Bishop, C. M. (2006). *Pattern recognition and machine learning.* Springer.
2. Raschka, S. (2015). *Python machine learning.* Packt Publishing Ltd.
3. Alpaydin, E. (2020). *Introduction to machine learning*. MIT Press.

Suggested Reading

1. Bishop, C. M. (2006). *Pattern recognition and machine learning.* Springer.
2. Raschka, S. (2015). *Python machine learning.* Packt Publishing Ltd.
3. Alpaydin, E. (2020). *Introduction to machine learning*. MIT Press.

BLOCK 3
Data Visualisation and Techniques

Data Visualisation Concepts

Learning Outcomes

After completing this unit, the learner will able to:

- identify how visualizations aid in understanding, interpreting, and communicating complex data effectively
- define appropriate visualization techniques for different types of quantitative data
- describe the use of various types of charts (e.g., bar charts, line graphs, pie charts, histograms, scatter plots)
- recognize the differences between numerical (quantitative) and non-numerical (qualitative) data

Prerequisites

Assume a teacher preparing a report on student performance for a parent-teacher meeting. To make the data easy to understand, the teacher uses bar graphs to show test scores, a pie chart to display the percentage of students in each grade category, and a map to show where students live in relation to the school. This simple real-world example shows how different types of visualizations can help communicate various forms of data more clearly. To fully understand or create such visualizations, learners need some basic skills and knowledge, which serve as important prerequisites for this topic.

First, learners should have a clear understanding of data types, particularly the difference between numerical (quantitative) and non-numerical (qualitative) data. They should also know how to perform basic calculations such as averages, percentages, and frequencies, which are commonly used in creating meaningful charts. These statistical concepts help in summarizing and interpreting raw data before it is visualized. Additionally, familiarity with spreadsheet tools like Microsoft Excel or Google Sheets is important, as these are widely used for organizing data and generating simple graphs and charts.

Second, learners should have some level of visual literacy and map-reading ability, especially for topics like geospatial data visualization. They should be comfortable interpreting visuals like maps, understanding directions, scales, and symbols, and recognizing how data is distributed over a geographic area. For example, reading a map that shows the number of COVID-19 cases in different cities requires understanding how color intensity or size of markers represents higher or lower values. These skills, along with critical thinking to identify misleading visuals or incorrect interpretations, form the foundation for mastering data visualization effectively.

Keywords

Graphs, Charts, Tables, Geospatial Data, Quantitative Data, Qualitative Data.

Discussion

3.1.1 Overview of Data Visualization

Our brains are highly adept at processing and understanding visual information compared to textual or auditory information. This phenomenon is often referred to as the "picture superiority effect." Our brains find visual representations of data more accessible and memorable because of efficiency in processing, higher retention, ease of comprehension, engagement and attention, facilitation of pattern recognition and emotional impact. So data visualisation is a powerful tool in data analytics allowing individuals to harness the innate capabilities of the human brain to process and understand complex information more effectively.

Companies and institutions increasingly prioritise gathering data from customers, internal processes, and services. Today, businesses require comprehensive analysis ranging from understanding their target market to tracking individual sales. Consequently, there has been a proliferation of data visualisation tools designed to present business data to management in a clear and efficient manner, aiding them in good decision making.

> Data visualization is the process of presenting information or data in a visual manner, such as maps, graphs, or charts, to make it simpler to comprehend and evaluate.

Data visualisation is a crucial component of the data science process, indicating that once data is gathered, processed, and modelled, it should be represented visually to draw conclusions. Additionally, within the broader framework of data presentation architecture (DPA), data visualisation plays a significant role. DPA focuses on efficiently identifying, organising, processing, formatting, and delivering data. Primary aim of data visualisation is to facilitate the identification of patterns, trends, and anomalies within extensive datasets. Other terms like information graphics, information visualisation, and statistical graphics are sometimes used interchangeably with data visualisation.

Data visualisation is used by teachers to display student test results, by computer scientists exploring advancements in artificial intelligence (AI) or by executives looking to share information with stakeholders. It also plays an important role in big data projects. As businesses accumulated massive collections of data during the early years of the big data trend, they needed a way to get an overview of their data quickly and easily. Visualisation tools were a natural fit. When a data scientist is writing advanced predictive analytics or machine learning (ML) algorithms, it becomes important to visualise the outputs to monitor results and ensure that models are performing as intended. This is because visualisations of complex algorithms are generally easier to interpret than numerical outputs.

3.1.2 Purpose of Data Visualization

In today's data-driven world, the ability to understand and communicate information effectively is more important than ever. This is where data visualization plays a crucial role. The purpose of data visualization is to present complex data in a clear, concise, and visually engaging manner, allowing viewers to quickly grasp patterns, trends, and insights that might be difficult to identify in raw numbers or text. Whether it's a simple bar chart showing monthly sales or an interactive map displaying population density, visualizations help transform abstract data into meaningful, accessible stories that support decision-making, problem-solving, and communication across a wide range of fields. Main purposes of Data Visualization include the following points:

1. **Simplifies and improves understanding of complex data:** Data visualisation helps in understanding complex datasets by representing them in a visual format such as charts, graphs, or maps. It allows analysts to quickly identify patterns, trends, and outliers that may not be apparent from raw data. This is because humans are much better at processing visual information than textual information.

2. **Communication of insights:** Visualisations make it easier to communicate insights and findings to stakeholders who may not have a background in data analysis. Visual representations are often more intuitive and engaging than raw numbers or text-based reports.

3. **Identification of patterns and trends:** Data visualisation can help people to identify patterns and trends in data that they might not otherwise be able to see. This is because visual representations can highlight relationships between different variables that are not always apparent in raw data.

4. **Detecting Anomalies and Outliers:** Visualisations make it easier to spot anomalies and outliers in the data, which may indicate errors or interesting phenomena worth investigating further.

5. **Comparing Data:** Visualisations enable analysts to compare different datasets or subsets of data effectively. This comparison can reveal relationships between variables and help in understanding the impact of various factors on the data.

6. **Increases accessibility:** Not everyone is a data expert. Data visualisation makes data more accessible to a wider audience, from executives to operational teams, enhancing overall data literacy within the organisation.

7. **Real-time monitoring:** With the rise of interactive dashboards, businesses can monitor their operations in real-time. This can help with tasks like tracking sales performance, monitoring supply chains, and managing operational efficiency.

8. **Identify areas that need attention or improvement:** Visualisation of data can highlight areas where a business can improve. This could be a department not reaching targets, a product not performing well, or a process that needs streamlining.

9. **Predictive analysis:** Advanced visualisation tools enable businesses to predict future trends based on historical data. This can be useful for forecasting sales, demand, and other important business metrics.

10. **Storytelling:** Data visualisations can be used to tell a story or convey a message effectively. By combining visual elements with narrative techniques, analysts can create compelling data-driven narratives that resonate with their audience.

11. **Exploratory Data Analysis (EDA):** Data visualisation is an integral part of exploratory data analysis, where analysts explore datasets to understand their structure, distributions, and relationships between variables.

12. **Interactive Visualisations:** Interactive visualisations allow users to explore data dynamically by interacting with the visualisations. Users can filter, drill down, and zoom in on specific data points, enhancing their understanding and exploration of the data.

13. **Decision Making:** Data visualisation aids in decision-making by providing decision- makers with clear, concise, and visually appealing representations of relevant information. This enables more informed and data-driven decision-making processes.

14. **Performance Monitoring:** Visualisations are often used to monitor key performance indicators (KPIs) and track progress towards goals. Real-time or regularly updated visualisations help organisations stay informed about their performance and take timely action when necessary.

15. **Risk Management:** Data visualisation can help organisations understand complex scenarios that involve risks and uncertainties in a better way. The visual simplification of data can assist in identifying the potential areas of risk.

3.1.3 Key Concepts in Data Visualization

In an age where vast amounts of data are generated daily, the ability to interpret and communicate information effectively has become increasingly important. Data visualization plays a vital role in this process by transforming raw data into visual formats such as charts, graphs, and maps that are easier to understand and analyze. Rather than relying solely on numbers and text, visualizations help reveal patterns, trends, and relationships that may otherwise go unnoticed. Whether used in academic research, business reporting, or public communication, data visualization enhances comprehension, supports better decision-making, and allows for more compelling storytelling. This section explores the key concepts, principles, and techniques essential for creating effective and meaningful data visualizations.

3.1.3.1 Visual Encodings

Let's say we're sketching a mango tree. First, we arrange the position of the tree on our canvas, followed by determining its height and width. Once the basic structure is drawn, we select colours for the stem, roots, leaves, and fruit that resemble those found on a real mango tree. This is visual encoding of data.

Data encodings are the visual elements used to represent data in a visualisation. Visual encodings are the methods by which data values are represented visually in a chart or graph. They act as the link between abstract data and the viewer's perception, making complex information more accessible and interpretable. Common visual encodings include position, length, angle, area, shape, and color. Choosing appropriate visual encodings is essential for effectively representing data and conveying insights.

- **Position:** The location of a data point on the chart or graph.
- **Length:** The length of a bar or line segment.
- **Angle:** The angle of a line or sector.
- **Colour:** The colour of a data point or symbol.
- **Size:** The size of a data point or symbol.

3.1.3.2 Choosing the Right Visual Format

Suppose I need to represent students' progress yearly. I can do this by using a table or a graph, etc. If I use a line chart, you can easily track individual students' progress over time and identify any patterns or trends. If I use a bar chart, it helps us to easily compare the progress of different students in a specific year or across multiple years. If I use a table, it allows for easy comparison and reference but may not be as visually engaging as graphical representations. So depending on the type of data, purpose of visualisation, and audience, we can select a visualisation method.

Data visualization is an essential tool in making data insights more accessible and understandable. However, with the vast array of visual formats available, selecting the appropriate one for your data is crucial to ensuring your message is clear and impactful. The right visual format not only presents data but also highlights key trends, patterns, and outliers effectively.

Choosing the right visual format depends on several factors including the purpose of the visualisation, the type of data being presented, and the audience you're targeting. Here are some key considerations for choosing the right visual format:

1. **Understand the Type of Data**

    a. **Categorical Data:** This refers to data that can be divided into distinct groups or categories (e.g., colors, brands, types). For categorical data, bar charts, pie charts, and stacked bar charts are common choices. These visuals allow for easy comparison between different categories.

    b. **Numerical Data:** This data consists of numbers that can be ordered or measured on a scale. Line graphs, scatter plots, and histograms are well-suited for numerical data as they can show relationships, distributions, or trends over time.

    c. **Time-Series Data:** When data is collected or measured at different time points (e.g., monthly sales), line charts, area charts, or time-series plots are effective in showcasing trends and changes over time.

    d. **Geospatial Data:** Data that has a geographic component (e.g., locations, regions) can be visualized using maps. Heat maps or choropleth maps can be useful for showing geographical patterns.

2. **Purpose of the Visualization**

    a. **Comparison:** If the goal is to compare different groups or categories, bar charts or column charts are often the best choice. These formats make it easy to contrast values between distinct categories.

    b. **Trend Identification:** Line charts, area charts, or time series plots are ideal for showing trends or changes over time. They highlight increases, decreases, or cycles in data.

    c. **Distribution:** To show how data is distributed across a range of values, histograms, box plots, or density plots are appropriate. These formats allow for the visualization of the spread, central tendency, and variation within a dataset.

    d. **Relationship:** Scatter plots or bubble charts are useful for displaying the relationship between two or more variables. These formats can show correlations, clusters, or outliers.

3. **Audience and Context**

   a. **Expert Audience:** If your audience is familiar with the data and its complexities, you can use more sophisticated visualizations like heat maps, treemaps, or network graphs. These formats offer deeper insights but may require a higher level of expertise to interpret.

   b. **General Audience:** For a broader or less data-savvy audience, keep the visuals simple and intuitive. Use bar charts, line graphs, or pie charts, and avoid overloading the viewer with too much information at once. Simplicity often enhances comprehension.

   c. **Context of Use:** Consider where and how your visualization will be presented. For example, an interactive dashboard might allow for more complex and dynamic visualizations, while a printed report would require static charts that can still tell a compelling story.

4. **Clarity and Aesthetics**

   a. **Simplicity Over Complexity:** Avoid cluttering your visual with too much data. A good visualization should focus on the key message. Remove unnecessary gridlines, 3D effects, and excessive colors to improve clarity.

   b. **Color Usage:** Colors should be used strategically to emphasize key data points or patterns. Consistent color schemes help guide the viewer's eye and aid in differentiating categories or trends. However, avoid overuse of colors, which can overwhelm or confuse the audience.

   c. **Legibility:** Ensure that text, labels, and legends are readable. Consider the size, font, and placement of text to avoid crowding the visual.

5. **Interactivity**

If your visualization will be explored interactively, such as in a dashboard, you may opt for more dynamic visual formats like interactive maps, pie charts with drill-down capabilities, or time sliders for time-series data. Interactivity allows the user to explore data from different angles, providing a more engaging and personalized experience.

6. **Scalability**

Consider how the visualization will scale. Some formats, like scatter plots or bubble charts, can become difficult to read when there is a large amount of data. If you are dealing with a high volume of data, consider aggregating it or using formats that can handle large datasets, such as heat maps or treemaps.

Some common visual formats and their best use cases are shown in the table (Table:3.1.1.) below.

Table 3.1.1 Common visual formats and their best use cases

VISUAL FORMATS	USE CASES
Bar Charts	Best for comparing different categories or discrete variables.
Line Charts	Ideal for showing trends over time or continuous data.
Pie Charts	Useful for representing parts of a whole, but should be limited to fewer categories to avoid clutter.
Histograms	Best for showing the distribution of numerical data.
Scatter Plots	Used to display relationships between two continuous variables.
Heat Maps	Ideal for showing data density or intensity across a grid or space.
Box Plots	Useful for showing the spread and summary statistics of numerical data.

Choosing the right visual format is about understanding the data, the audience, and the key message you want to convey. A well-chosen visualization simplifies complex data and aids in decision-making by providing insights at a glance. Whether you're comparing data points, highlighting trends, or showing relationships, aligning your visualization type with your data's structure and the audience's needs ensures that your message is both clear and compelling.

3.1.3.3 Design Principles for an Effective Data Visualization

Have you seen the building plan of a house? When considering the building plan of a house, it's crucial to ensure simplicity and clarity, aligning with our budget and the site's position. A well-crafted plan should effortlessly delineate the layout, distinguishing bedrooms, living halls, kitchens, dining halls, and toilets. Consistency in design throughout each section is key,ensuring seamless transitions and easy comprehension. Additionally, precise measurements of room sizes are paramount, as every detail in the plan should accurately reflect reality. Just as a house plan must adhere to these principles, so too should every visualisation, ensuring accuracy, clarity, and coherence for effective communication of insights. The design principles for an effective visualization are:

a. **Clarity:** The visualisation should be clear and easy to understand. Avoid cluttering the chart with too much data or unnecessary elements.

b. **Accuracy:** The data should be accurately represented in the visualisation. Double- check your data and ensure the visual encodings are consistent.

c. **Relevance:** The visualisation should be relevant to the audience and the message you want to convey. Choose a visual format that effectively communicates your findings.

d. **Consistency:** Use consistent visual encodings throughout your visualisation. This helps viewers easily compare data points and identify patterns.

e. **Aesthetics:** While clarity and accuracy are essential, don't neglect the aesthetics of your visualisation. Use colour, fonts, and other design elements to create a visually appealing and engaging chart or graph.

f. **Interactivity:** Interactive visualisations allow users to explore data dynamically by interacting with visual elements. Interactivity enhances the analytical process by enabling users to filter, drill down, and zoom in on specific data points.

g. **Data Storytelling:** Data storytelling involves using visualisations to communicate a narrative or message effectively. By combining visual elements with narrative techniques, analysts can create engaging stories that resonate with their audience and drive action.

h. **Dashboard Design:** Dashboards are a collection of visualisations arranged on a single screen to provide an overview of key metrics and insights. Effective dashboard design involves selecting relevant visualisations, organising them logically, and ensuring clarity and usability.

i. **Colour Theory:** Color plays a crucial role in data visualisation, helping to distinguish between different categories, highlight trends, and draw attention to key insights. Understanding colour theory and applying it appropriately can improve the readability and effectiveness of visualisations.

j. **Data Preparation:** Before creating visualisations, it is essential to prepare the data properly. This may involve cleaning and transforming the data, handling missing values, and aggregating or summarising the data as needed.

k. **Audience Consideration:** When designing visualisations, it is important to consider the audience's background, knowledge, and preferences. Tailoring visualisations to the audience ensures that the information is presented in a way that is relevant and understandable to them.

3.1.4 Challenges in Data Visualization

Have you ever watched an ad while at the cinema or during a similar experience? An ad must adhere to several principles to effectively capture and retain viewers' attention. It must be engaging for the audience, informative, and relevant to their interests, demographics, and preferences, ensuring that the message resonates with them on a personal level. Additionally, it should be memorable, maintain consistent branding, include a clear call to action, be respectful of context, and demonstrate adaptability. While data visualisation is a powerful tool in data analytics, it also comes with its own set of challenges. Here are some common challenges faced in data visualisation:

3.1.4.1 Data-related challenges

a. **Data quality:** Garbage in, garbage out. One of the primary challenges is ensuring the quality and integrity of the data being visualised. Inaccurate, incomplete, or inconsistent data can lead to misleading visualisations.

b. **Large datasets:** Big data brings big challenges. Visualising massive datasets effectively requires choosing the right tools and techniques to avoid overwhelming viewers with information overload.

c. **Data complexity:** Complex data with multiple variables can be difficult to represent clearly in a single visualisation. Finding the right balance between simplicity and comprehensiveness is crucial.

d. **Interpretation and Misinterpretation:** Visualisations can sometimes be misinterpreted, leading to incorrect conclusions or decisions. Ensuring that visualisations are clear, accurately represent the underlying data, and are interpreted correctly by users is crucial.

3.1.4.2 Design-related challenges

a. **Choosing the right chart type:** Selecting the wrong chart type can distort the data and misrepresent the message. Understanding the strengths and weaknesses of different chart types is essential.

b. **Visual Clutter and Overplotting:** Visual clutter, including too many data points, labels, or visual elements, can make visualisations difficult to interpret. Overplotting, where data points overlap and obscure each other, can also be a problem, particularly in scatter plots or density plots.

c. **Color Selection and Accessibility:** Choosing appropriate colours for visualisations is important for effective communication and accessibility. Poor colour choices can lead to confusion or misinterpretation, particularly for users with colour vision deficiencies.

d. **Misleading or subjective elements:** Using misleading scales, manipulative axes, or subjective colour schemes can lead to biassed interpretations. Emphasise neutrality and objectivity in your visualisations.

3.1.4.3 User-related challenges

a. **Audience understanding:** Consider your audience's level of data literacy and tailor your visualisation accordingly. Complex visualisations might be lost on an uninformed audience.

b. **Accessibility:** Ensure your visualisations are accessible to people with disabilities, including colour blindness. Use appropriate colour palettes and alternative text descriptions.

c. **Interactive limitations:** Static visualisations might limit exploration and understanding. Consider interactive elements for deeper engagement and analysis. Interaction can be challenging. Incorporating interactive features and designing intuitive user interfaces requires careful consideration of user needs and preferences.

d. **User Engagement:** Designing visualisations that engage users and encourage exploration.

3.1.4.4 Technological challenges

a. **Tool limitations:** Not all visualisation tools are created equal. Choose a tool that aligns with your data size, complexity, and desired functionalities.

b. **Integration with Analytics Workflow:** Integrating data from different sources into a cohesive visualisation can be challenging, requiring data cleaning and harmonisation efforts. Integrating data visualisation into the broader data analytics workflow, including data preparation, analysis, and reporting, can be complex. Ensuring seamless integration and interoperability with other analytics tools and platforms is essential for a smooth workflow.

c. **Dynamic data updates:** Keeping visualisations updated with real-time or constantly changing data can be challenging and require automated solutions.

d. **Performance and Scalability:** Visualising large datasets or real-time data streams can pose performance and scalability challenges. Ensuring that visualisations remain responsive and usable even with large amounts of data can be difficult.

e. **Data Privacy and Security:** Visualising sensitive or confidential data raises concerns about data privacy and security. Ensuring that sensitive information is appropriately anonymized or protected in visualisations is essential to avoid privacy breaches.

3.1.5 Common Data Visualisation Use Cases

Data visualization plays a critical role in transforming raw data into meaningful insights by presenting information in a clear and visually engaging format. It allows users to identify patterns, trends, and relationships that might otherwise go unnoticed in spreadsheets or raw datasets. From business intelligence to scientific research, data visualization is used across a wide range of fields to support analysis, decision-making, and communication. Understanding the common use cases helps individuals and organizations apply the right visualization techniques to specific problems whether it's tracking performance, comparing categories, analyzing trends, or communicating findings effectively. Common use cases for data visualisation include the following:

1. **Business Intelligence and Analytics:** Data visualisation is extensively used in business intelligence and analytics to analyse sales trends, customer behaviour, market segmentation, and other Key Performance Indicators (KPIs). It helps organisations make data-driven decisions and identify areas for improvement.

2. **Financial Analysis:** In finance, data visualisation is used to analyse stock market trends, track investment performance, visualise financial statements,

and identify patterns in economic data. It helps investors and financial analysts understand complex financial data more easily.

3. **Marketing and Advertising:** Marketers use data visualisation to analyse campaign performance, track website traffic, monitor social media engagement, and visualise customer demographics. It helps them identify successful strategies, optimise marketing efforts, and target specific customer segments more effectively.

4. **Healthcare and Medical Research:** Data visualisation plays a crucial role in healthcare for visualising patient data, tracking disease outbreaks, analysing medical research findings, and identifying healthcare trends. It helps healthcare professionals make informed decisions and improve patient outcomes.

5. **Supply Chain Management:** In supply chain management, data visualisation is used to track inventory levels, monitor logistics operations, optimise transportation routes, and identify supply chain bottlenecks. It helps companies streamline their supply chain processes and improve efficiency.

6. **Education and Academia:** Educators and researchers use data visualisation to present research findings, visualise educational performance data, and facilitate data-driven decision-making in education policy. It helps educators identify student learning trends and personalise instruction.

7. **Human Resources:** In HR, data visualisation is used to analyse employee turnover rates, track workforce demographics, visualise performance metrics, and identify talent acquisition trends. It helps HR professionals make strategic decisions related to recruitment, retention, and employee development.

8. **Geospatial Analysis:** Geospatial data visualisation is used in fields such as urban planning, environmental science, and transportation to visualise spatial data, analyse geographic patterns, and make location-based decisions. It helps professionals understand spatial relationships and solve complex spatial problems.

9. **Scientific Research:** Scientists use data visualisation to visualise experimental results, analyse scientific data, and present research findings in fields such as biology, chemistry, physics, and environmental science. It helps researchers communicate complex scientific concepts more effectively.

10. **Government and Public Policy:** Governments use data visualisation to analyse census data, track economic indicators, visualise public health data, and communicate policy initiatives to the public. It helps policymakers make evidence-based decisions and promote transparency in government.

11. **Politics:** A common use of data visualisation in politics is a geographic map that displays the party each state or district voted for.

3.1.6 Presentation of Quantitative Data

In the initial stages of your research, you pose questions such as, "What is the number of students passing the school leaving examination, categorised by classes/divisions?", "How do the passing rates of female students compare to those of male students?", "Has there been an increase in educational attainment among the Scheduled Castes since the last census?", "Why does one village lack factions while another does not?". These questions are then translated into hypotheses involving variables. For instance, a student's gender is considered a variable, with male and female as its respective values. Similarly, marks obtained in an examination constitute a variable, with each student's marks serving as a value. The unity within a village is another variable, with the presence or absence of factions being its values. These variables are operationalized, meaning they are defined in observable and measurable terms. Subsequently, data collection takes place, employing suitable research designs such as surveys, observations, or the analysis of aggregate data. This may involve the creation of interview schedules, data recording sheets, or the adoption of observational or social anthropological approaches. After processing the collected data, which includes tasks like data entry and coding, the analysis phase begins.

Quantitative data is gathered through various means such as surveys, censuses, and field experimental designs, focusing on consistent units of analysis. These units can take the form of voters, students, districts, or similar entities, each identified with a sequential numbering system (e.g., voter 1, voter 2... up to voter n). The data collected relates to characteristics that are either continuous or discrete variables. a continuous variable encompasses an infinite range of potential values spanning from the lowest to the highest. For instance, age can be expressed in years, years and months, or even years, months, and days. Similarly, time spent studying per day can be quantified in terms of hours or hours and minutes. Examples of continuous variables include weight, temperature, building height, road length, student grades, and land holdings in acres. For example, gender consists of discrete categories such as male and female, with no intermediate values between them. Likewise, the type of school attended private, aided, or municipal is discrete, as there are no gradations between these distinct categories. Although discrete variables are countable (e.g., student 1, student 2... up to student n), they are just as quantitative as continuous variables.

Quantitative analysis refers to the type of analysis that includes quantitative data, which is, data expressed numerically. It is essential to keep in mind the levels of measurement when plotting variables; this is important because the type of data influences how the data should be displayed. Levels of measurement are clearly explained in the table (Table: 3.1.2.) below.

Table 3.1.2 Levels of measurement

Levels of measurement	Definition	Example
Nominal	There are differences, but there is no order to them, and we can't measure how much each differs.	Participants eye colour
Ordinal	Distinguishes differences, which have order, but we can't measure how much each differs.	Response on a Likert scale
Interval	There is an order, and the differences between figures are measurable.	Temperature
Ratio	The ratio is the same as the interval, with the difference that there is an absolute 0, meaning the values of the variable cannot go below 0.	Height

Presentation of quantitative data involves various techniques to effectively communicate numerical information. Here are some common methods for quantitative data.

3.1.6.1 Tables

Tables present quantitative data in rows and columns, providing a structured format for organising and comparing numerical values. They are useful for presenting precise numerical information and facilitating easy comparison between different categories or groups.

Tabulation is the first step before the data is used for analysis or interpretation. A table can be simple or complex, depending upon the number or measurement of a single set or multiple sets of items. Whether simple or complex, there are certain general principles which should be borne in mind in designing tables:

- The tables should be numbered e.g. Table 1, Table 2 etc.
- A title must be given to each table. The title must be brief and self-explanatory.
- The headings of columns and rows should be clear and concise.
- The data must be presented in an order.

  e.g.: size, importance, chronologically, alphabetically or geographically
- If percentages or averages are to be compared, they should be placed as close as possible.
- No table should be too large.
- Most people find a vertical arrangement better than a horizontal one because it is easier to scan the data from top to bottom than from left to right.
- Foot notes may be given where necessary, providing explanatory notes or additional information.

Tabulation of quantitative data is usually more cumbersome since the characteristic itself has a magnitude or size which is measurable as well as has the frequency i.e. the number of persons having each measure e.g. height, weight pulse rate etc. In the case of a quantitative variable, the variable can be divided into class intervals with the frequency mentioned against each class interval (or group interval).

Frequency will include all persons having that value of the attribute which falls within the range of the class interval. Table 3.1.3. shows a Height-Weight table based on age to monitor your child's growth.

Table 3.1.3 Height-Weight table based on age to monitor your child's growth

Age	Boys		Girls	
	Height	Weight	Height	Weight
At the time of birth	33	50.5	33	49.9
3 months	6	61	54	60.2
6 months	7.8	67.8	7.2	66.6
9 months	9.2	72.3	8.6	71.1
1 year	10.2	76.1	9.5	75

3.1.6.2 Charts and Graphs

The purpose of data presentation through charts and graphs is to have a quick visual impression. Readers can easily understand visual presentations since these have a striking visual impact. Tables are sometimes difficult to perceive. Graphical presentations convey the essence of the statistical data, circumventing the details. The data presented through a chart/graph must be simple. Complicated charts fail to convey the message and beat the purpose of these drawings. Hence a lot of details of the original data have to be foregone for preparing charts and diagrams.

It is of utmost importance that the visual presentations are produced correctly, using appropriate scales. Otherwise, distortion of data may occur and the resulting visualisations of statistical information can be misleading. Remember, computers will do what you tell them to do and not what should be done. It is better to try out the charts and graphs on colleagues first to detect any mis- interpretations. Graphs, charts and diagrams should be self-explanatory and the reader should be able to understand them without detailed reference to the text. The title should be informative and the axes of graphs should be clearly labelled. The key should be placed where required.

Charts in data visualization are essential tools that help translate complex numerical data into visually appealing and easily interpretable graphics. They simplify the process of identifying patterns, trends, and outliers, enabling individuals to make data-driven decisions efficiently. With various types of charts available, each serves a distinct purpose depending on the nature of the data being analyzed. Graphs are essential tools in data visualization that transform raw data into meaningful, visual representations, making complex information more accessible and easier to interpret. These visual tools

enable analysts, decision-makers, and audiences to quickly grasp insights from data, whether it's understanding fluctuations over time, comparing different categories, or identifying correlations. In essence, graphs not only simplify the interpretation of data but also enhance the ability to communicate insights clearly and effectively.

There are different kinds of graphs and charts available based on our application, they are Bar Charts, Line Charts, Pie Charts, Scatter Plots, Histograms, Box Plots, etc. Each type is very clearly defined below.

3.1.7 Types of Charts and Graphs

Charts and graphs are essential tools in data visualization that help transform raw data into meaningful visual representations. They enable viewers to quickly grasp complex information by highlighting patterns, trends, and comparisons that may be difficult to detect in tables or text. Different types of charts and graphs are designed to suit specific kinds of data and analytical purposes. Choosing the appropriate chart type is critical for effectively communicating the message and ensuring accurate interpretation. This section explores the most commonly used charts and graphs, their characteristics, benefits, drawbacks and the situations in which they are most effective.

3.1.7.1 Pictograph

A pictograph, also known as a pictogram, is a type of data visualization that uses pictures or symbols to represent data values. Instead of relying solely on bars or lines, pictographs convey information using images that represent specific quantities. Each image or symbol typically represents a certain number of items or units, making it a visually engaging way to show data, especially for audiences who may not be familiar with more traditional chart types.

Advantages

- **Engaging and Simple:** Pictographs are visually appealing and can help capture the attention of the audience, making the data more relatable and easier to understand, especially for non-experts.
- **Effective for Small Data Sets:** Pictographs are great for displaying small datasets where the use of images enhances comprehension.
- **Intuitive:** The use of familiar symbols or images can make the data more intuitive, especially in contexts where abstract numbers may be harder to grasp (e.g., showing the number of people using icons of people or animals).

Disadvantages

- **Limited Precision:** Pictographs may lack precision, as symbols represent rounded values, and exact numbers are sometimes difficult to derive just by looking at the image.
- **Not Ideal for Complex or Large Datasets:** For large datasets or datasets with many categories, pictographs can become cluttered or hard to interpret.

- Misleading Representation: If not designed properly (e.g., using unequal symbols or inconsistent scaling), pictographs can mislead the viewer about the significance of the data.

As an illustration, let's take the example of a class comprising 20 students who utilize different modes of transportation, such as buses, cars, autos, and bicycles. The Table 3.1.4 below displays the number of students using each mode of transportation.

Table 3.1.4 Number of students using each mode of transportation

Mode of transportation	Number of students
Bus	8
Car	2
Auto	4
Bicycle	6

We can represent the above Table 3.1.4 information using the pictograph (Figure 3.1.1) shown below.

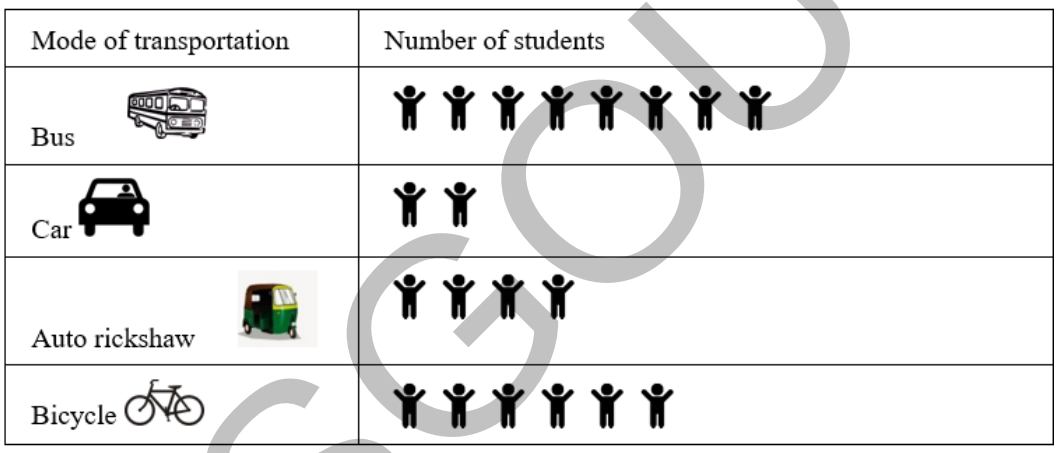


Fig. 3.1.1 Illustration of Pictograph

3.1.7.2 Bar Graph

A bar graph (also known as a bar chart) is one of the most common types of graphs used to display and compare data across different categories. In a bar graph, rectangular bars (either horizontal or vertical) are used to represent data, with the length of each bar being proportional to the value it represents. Bar graphs are particularly useful for comparing quantities, frequencies, or other measures across distinct groups or categories.

Advantages

- **Easy Comparison**: Bar graphs allow for straightforward comparisons between categories. It's clear to see which categories are larger or smaller based on the length of the bars.

- **Versatile**: They can display both small and large datasets, and they work well for both nominal (categorical) and ordinal data.

- **Clear Representation**: Bar graphs make it easy to interpret data visually, particularly when comparing values across categories.

Disadvantages

- **Cluttered with Too Many Categories:** If too many categories are displayed on a bar graph, it can become cluttered and difficult to interpret, especially if the bars are too close together.
- **Limited for Trend Analysis:** While bar graphs are great for comparisons, they are not suitable for showing changes over time or continuous data, where a line graph might be more appropriate.
- **Less Precise:** Bar graphs give a visual estimate of the data, so if you need to derive exact values, you may have to refer back to the data table.

We can represent the above Table 3.1.4 information using the bar graph (Figure 3.1.2) shown below.

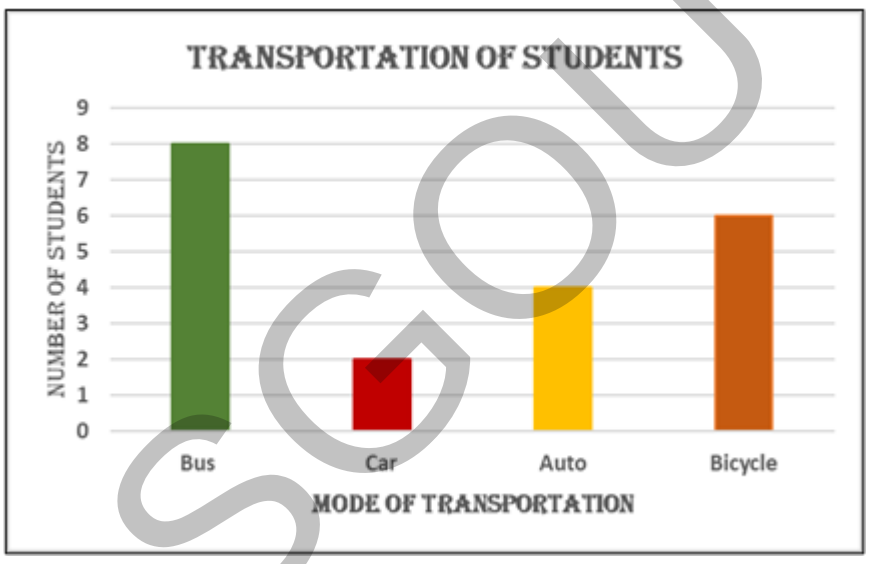


Fig. 3.1.2 Illustration of Bar graph
See the e-content for a color version of this image

3.1.7.3 Line Graphs

A line graph (or line chart) is a type of graph that uses points connected by straight lines to display data over a continuous range, typically over time. Line graphs are commonly used to visualize trends, changes, or patterns within a dataset, particularly when data points are collected at regular intervals. The x-axis usually represents time or another continuous variable, while the y-axis represents the data values or measurements.

Line graphs are ideal for showing how a particular variable changes in relation to time or another variable, making them one of the most widely used graph types in various fields such as finance, economics, science, and business.

Advantages

- **Trend Analysis:** Line graphs excel at displaying trends over time or continuous data, making it easy to spot patterns, increases, or decreases in the data.
- **Comparing Multiple Data Series:** When multiple lines are plotted, line graphs provide a clear way to compare trends or relationships between different datasets.
- **Clarity and Simplicity:** Line graphs are easy to read and understand, especially when displaying a smaller number of data points or variables.
- **Precision:** Line graphs offer more precision than bar graphs when it comes to showing gradual changes or specific data points.

Disadvantages

- **Not Ideal for Categorical Data:** Line graphs are best suited for continuous data, so they are not appropriate for categorical data (e.g., comparing distinct groups like regions or products).
- **Clutter with Multiple Series:** When too many lines are plotted on the same graph, it can become cluttered and difficult to interpret, especially if the lines overlap.
- **Limited in Representing Volatility:** Line graphs are effective for visualizing trends, but they may not be the best for showing extreme variations or irregular changes in data that need a more detailed examination.

We can represent the above Table 3.1.4 information using the line graph (Figure 3.1.3) shown below.

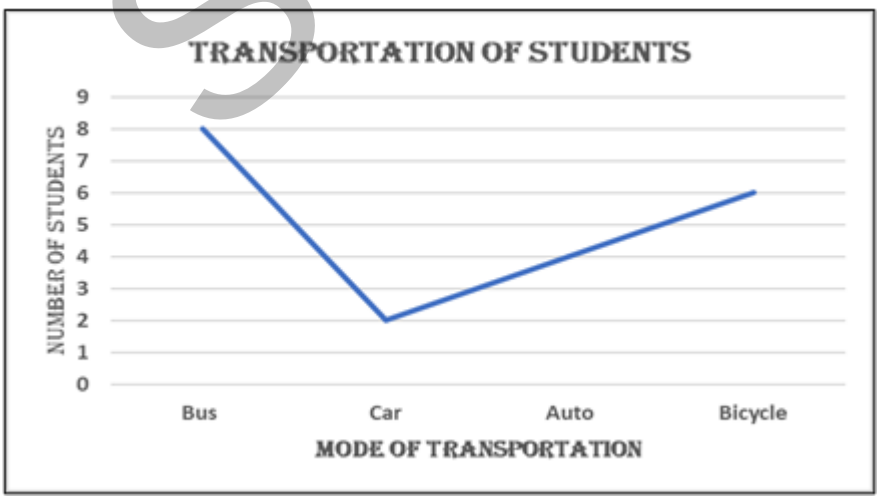


Fig. 3.1.3 Illustration of Line graph

3.1.7.4 Pie Chart

A pie chart is a circular chart divided into segments, each representing a part of the whole. Each slice of the pie is proportional to the quantity or percentage that it represents, making it an excellent tool for visualizing the composition of a dataset. Pie charts are most effective when showing how individual parts contribute to a total, and they are particularly useful when working with a small number of categories. The visual nature of pie charts makes them intuitive and easy for audiences to understand at a glance.

Advantages

- **Easy to Understand:** Pie charts are visually simple and can be easily interpreted, even by individuals with little to no background in data analysis.
- **Proportional Comparison:** They are great for showing the relative proportions of categories within a whole, making it easy to understand which category contributes the most or least.
- **Clear Representation of Percentages:** Pie charts are perfect for displaying percentages, making them a popular choice for survey results, market share, or demographic breakdowns.
- **Visually Appealing:** The use of different colors for each slice makes pie charts visually engaging, which can help draw attention to key insights.

Disadvantages

- **Limited to Few Categories:** Pie charts are most effective when there are only a small number of categories (usually 5 or fewer). With more categories, the slices become too small and difficult to differentiate.
- **Difficult to Compare Similar Slices:** If two slices are nearly the same size, it can be difficult to compare them accurately. Unlike bar charts or line graphs, pie charts don't provide precise measurements.
- **Not Ideal for Showing Trends:** Pie charts are static and are not suitable for showing changes over time or continuous data.
- **Can Be Misleading:** Pie charts can sometimes be misleading if the total is not 100%, or if they are designed with poorly chosen colors or incorrect scaling.

We can represent the above Table 3.1.4 information using the pie chart (Figure 3.1.4) shown below.

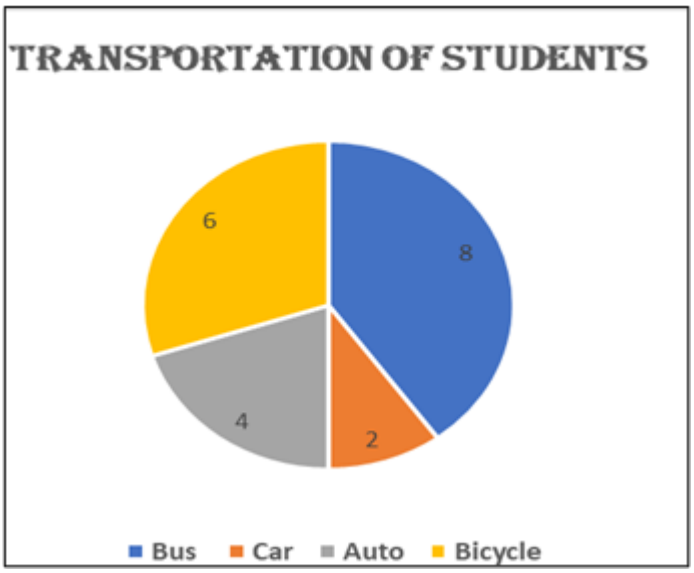


Fig. 3.1.4 Illustration of Pie chart
See the e-content for a color version of this image

3.1.7.5 Scatter Plot

A scatter plot (or scatter diagram) is a type of data visualization that displays individual data points as dots on a two-dimensional graph. The horizontal (x) and vertical (y) axes represent two different quantitative variables, and each point corresponds to a pair of values for these variables. Scatter plots are powerful tools for visualizing relationships, correlations, and patterns between two variables, and they are widely used in fields like statistics, economics, science, and engineering.

Advantages

- **Visualizing Relationships:** Scatter plots are excellent for identifying relationships between two variables. They make it easy to see if there is a positive, negative, or no correlation between the variables.
- **Identifying Patterns and Trends:** Scatter plots can reveal patterns, clusters, or trends within data. For example, you might notice that as one variable increases, so does the other (positive correlation).
- **Detecting Outliers:** Outliers are data points that deviate significantly from the rest of the data—are easy to spot on a scatter plot as they will appear as points distant from the main cluster of data.
- **Exploring Data:** Scatter plots allow for an intuitive exploration of data, making them great for initial data analysis and for discovering unexpected relationships.

Disadvantages

- Limited to Two Variables: Scatter plots are designed to represent the relationship between only two quantitative variables. They can't represent

more complex datasets with many variables (for that, you might need more advanced methods like multiple regression or 3D scatter plots).

- Clutter with Large Datasets: With large datasets, scatter plots can become cluttered, making it hard to differentiate individual points. Overlapping points (especially in dense areas) can lead to visual clutter.
- No Clear Explanation of Causality: While scatter plots show relationships, they do not imply causality. Just because two variables are correlated doesn't mean one causes the other. Further analysis is needed to determine causality.

We can represent the above Table 3.1.4 information using the scatter plot (Figure 3.1.5) shown below.

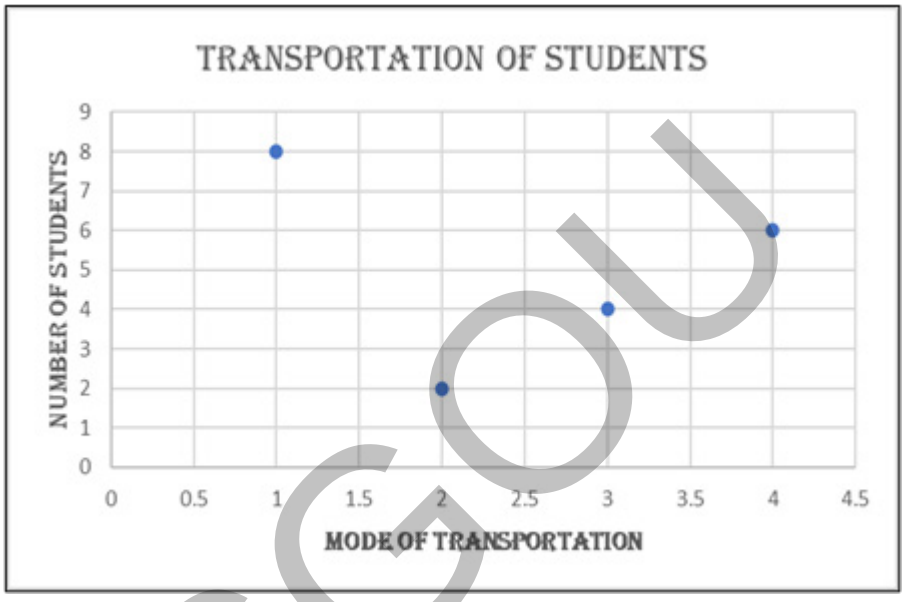


Fig. 3.1.5 Illustration of Scatter Plot
See the e-content for a color version of this image

3.1.7.6 Histogram

A histogram is a type of graph that represents the distribution of a dataset by grouping data into bins (or intervals) and plotting the frequency (count) of data points within each bin. Unlike bar charts, which display categorical data, histograms are specifically designed for continuous numerical data. They give a visual representation of the distribution, central tendency, and spread of the data, making them invaluable for understanding the underlying patterns or characteristics of a dataset.

Advantages

- **Distribution Visualization:** Histograms provide a clear visualization of the distribution of data, making it easy to see where data points are concentrated (i.e., peaks) and where there are gaps.
- **Identifying Skewness and Outliers:** Histograms help identify if data is skewed to the left or right and whether there are outliers (values that fall far outside the typical range).

- **Simple and Clear:** Histograms are straightforward to understand, and they are effective at showing the spread, central tendency, and variability of a dataset.
- **Analyzing Data Spread:** By grouping data into intervals, histograms provide insights into the spread of the data and can help detect patterns such as normal distribution or bimodal distributions.

Disadvantages

- **Choice of Bin Size:** The bin size (or width) significantly impacts the shape of the histogram. If the bins are too wide, important details of the data distribution may be lost. Conversely, if the bins are too narrow, the histogram might be overly fragmented and noisy.
- **Loss of Individual Data Points:** Since histograms group data into bins, individual data points are not visible. You lose the ability to see exact values, which may be important in some situations.
- **Not Ideal for Small Datasets:** For small datasets, a histogram may not provide enough meaningful insight, as there may not be enough data points to form clear, consistent patterns.

We can represent the above Table 3.1.4 information using the histogram (Figure 3.1.6) shown below.

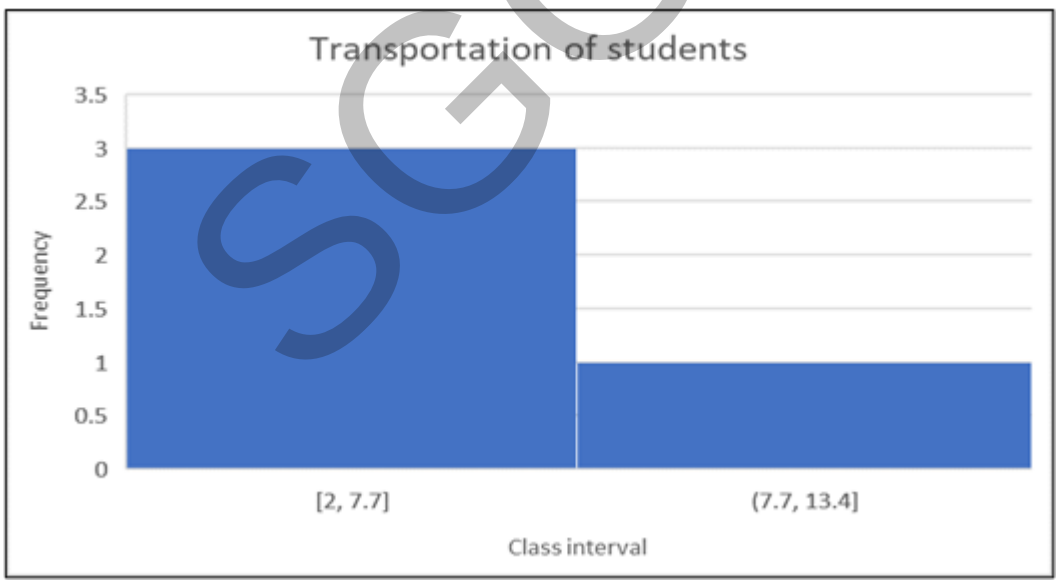


Fig. 3.1.6 Illustration of Histogram

3.1.7.7 Box Plots (Box-and-Whisker Plots)

A box plot (also called a box-and-whisker plot) is a standardized way of displaying the distribution of data based on five summary statistics: the minimum, first quartile (Q1), median, third quartile (Q3), and maximum. Box plots provide a clear visual representation

of a dataset's central tendency, spread, and outliers, making them particularly useful for identifying data distribution characteristics and comparing multiple datasets.

Advantages

- **Clear Distribution Overview:** Box plots provide a quick and clear summary of a dataset's distribution, showing the central tendency, spread, and variability.
- **Outlier Detection:** Box plots make it easy to identify outliers, which can be important for data cleaning and anomaly detection.
- **Comparison Between Multiple Datasets:** When plotting multiple box plots side by side, they allow for easy comparison of the distributions of different groups or datasets.
- **Visualizing Skewness:** The symmetry (or lack thereof) of the box and whiskers helps in identifying whether the data is skewed (left or right) or symmetric.

Disadvantages

- **Limited Detail:** While box plots are great for summarizing a dataset, they do not provide detailed information about the individual data points, such as exact values.
- **Less Intuitive for Small Datasets:** For very small datasets (fewer than 5 data points), box plots might not provide enough meaningful information to make comparisons.
- **No Information on Multimodal Distributions:** Box plots are not suitable for identifying multimodal distributions (i.e., distributions with multiple peaks). In such cases, histograms or kernel density plots might be more appropriate.

We can represent the above Table 3.1.4 information using the Box Plots (Figure 3.1.7) shown below.

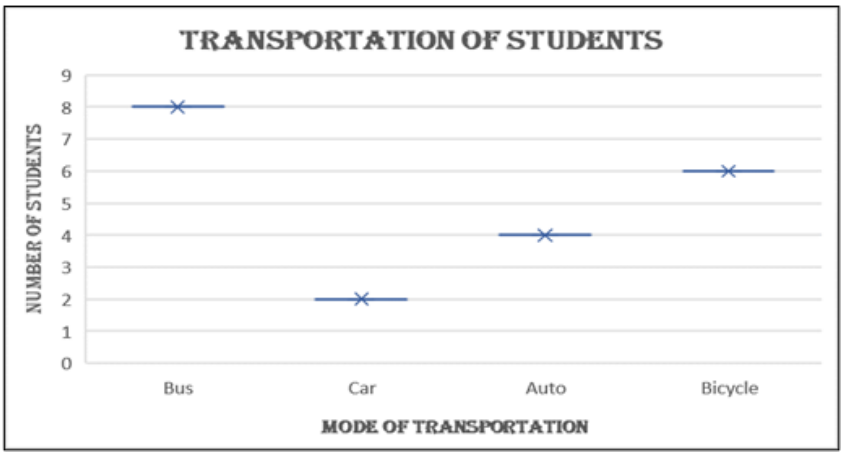


Fig. 3.1.7 Illustration of Box Plot

3.1.7.8 Heatmaps

A heatmap is a data visualization technique that uses color gradients to represent the values in a matrix or two-dimensional dataset. The colors in a heatmap correspond to data values, with different colors representing different magnitudes or categories of data. Heatmaps are particularly useful for visualizing complex data where relationships between variables are important, and they are widely used in fields such as statistics, machine learning, finance, biology, and web analytics.

Advantages

- **Quick Insights:** Heatmaps provide an immediate visual understanding of complex datasets by using color, allowing patterns, trends, and outliers to be quickly identified without needing to analyze raw numbers.
- **Clear Comparison:** Heatmaps make it easy to compare multiple variables or data points in the same visual. By using color to represent values, the relative significance of each data point becomes apparent.
- **Data Density:** Heatmaps are effective for representing large datasets in a compact space, showing multiple variables and their relationships in a single view.
- **Effective for Continuous Data:** Heatmaps work particularly well for continuous or large-scale datasets, where the color gradient can highlight the range and distribution of values.

Disadvantages

- **Color Interpretation:** The effectiveness of a heatmap depends heavily on the choice of colors. Poor color choices or a lack of contrast can make the data harder to interpret. Viewers might also misinterpret the data if the color gradient is not properly calibrated.
- **Over-Simplification:** While heat maps provide an overview, they may oversimplify complex data relationships. Important details might be missed if the heatmap fails to capture nuances in the data.
- **Data Density Issues:** For extremely large datasets, heatmaps may become overwhelming or difficult to read. Too many data points can make the color patterns indistinct, especially when there is a lack of differentiation between cells.
- **Not Ideal for Small Datasets:** For small datasets, heatmaps may not be the best choice. The richness of color variation is better suited for larger datasets where multiple values need to be compared at once.

We can represent the above Table 3.1.4 information using heat maps (Figure 3.1.8) shown below.

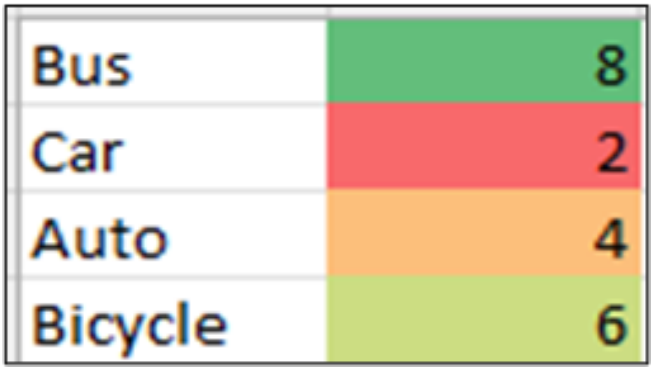


Fig. 3.1.8 Illustration of Heat maps

3.1.7.9 Area Charts

An area chart is a type of data visualization that combines elements of a line chart with the concept of filling the area beneath the line with color or shading. Area charts are primarily used to display the magnitude of a dataset over time or across categories, emphasizing the volume of data. This makes them useful for visualizing cumulative data and comparing multiple variables or datasets across different points.

Advantages

- **Shows Trends and Changes:** Area charts are effective at visualizing how a variable changes over time, providing insight into trends and variations.
- **Emphasizes Volume:** The filled area beneath the line makes it easy to understand the magnitude of values, which is particularly useful for displaying cumulative data.
- **Comparing Multiple Variables:** Stacked area charts allow for the comparison of multiple datasets, highlighting both individual trends and how different categories contribute to the total.
- **Intuitive and Visual:** The filled area under the line makes it easy to visually perceive data patterns and relative proportions.

Disadvantages

- **Can Be Hard to Interpret with Too Many Variables:** When there are too many series in a stacked area chart, it can become cluttered, making it difficult to distinguish between individual data points or trends.
- **Overlapping Areas:** In stacked area charts, the areas can overlap, which can obscure the values or make the interpretation of individual series difficult, especially if they are small in magnitude.
- **Less Precise for Small Variations:** Area charts are often less precise than line charts when it comes to showing small differences or exact values

because the emphasis is on the overall shape and magnitude rather than specific data points.

- **Can Be Misleading:** In some cases, if the data is not carefully represented (especially in stacked area charts), it may lead to a misleading perception of the magnitude of individual series, especially when the scales differ significantly.

We can represent the above Table 3.1.4 information using area charts (Figure 3.1.9) shown below.

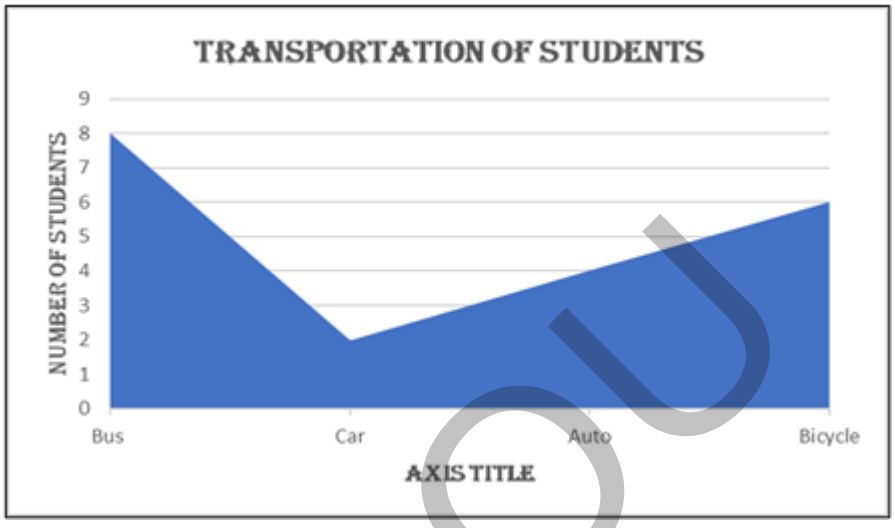


Fig. 3.1.9 Illustration of Area charts
See the e-content for a color version of this image

3.1.7.10 Bubble Charts

A bubble chart is a type of chart that displays data points as bubbles on a two-dimensional plot. Each bubble represents a data point, and its position on the chart is determined by two variables (usually the x-axis and y-axis). The size of the bubble typically represents a third variable, allowing for the visualization of three different dimensions of data in one plot. Bubble charts are often used to show relationships between numerical values, detect patterns, and identify clusters or outliers within a dataset.

Advantages

- **Multivariable Visualization:** Bubble charts allow for the display of three variables at once, enabling more complex relationships to be analyzed and understood in a single chart.
- **Visualizing Relationships and Patterns:** By using size and position to represent different data points, bubble charts help visualize trends, clusters, correlations, and outliers, making it easier to detect patterns in large datasets.
- **Identifying Outliers:** Larger or smaller bubbles stand out, making it easier to identify anomalies or outliers in the data.

- **Effective for Comparing Data Points:** With bubbles representing different groups or categories, it's easier to compare individual data points and how they vary across different dimensions.

Disadvantages

- **Overlapping Bubbles:** In cases where data points are too close together, the bubbles may overlap, making it difficult to interpret the exact values or distinguish individual points.
- **Difficult to Interpret with Too Many Bubbles:** When too many data points are plotted, the chart can become cluttered and harder to read, especially if there are many similar-sized bubbles.
- **Misleading Size Representation:** If the bubble sizes are not scaled properly or if viewers misinterpret bubble size, it can lead to confusion or misinterpretation of the data. Bubble size can visually exaggerate or minimize certain values.
- **Requires Accurate Scaling:** Proper scaling of bubble size is crucial for clear interpretation. If the sizes are not proportional or are scaled inconsistently, it can lead to inaccurate or misleading insights.
- **Lack of Precision:** Unlike bar or scatter plots, bubble charts do not provide precise values. The exact value of each bubble can be difficult to extract just by looking at the chart, especially when many bubbles overlap.

We can represent the above Table 3.1.4 information using bubble charts (Figure 3.1.10) shown below.

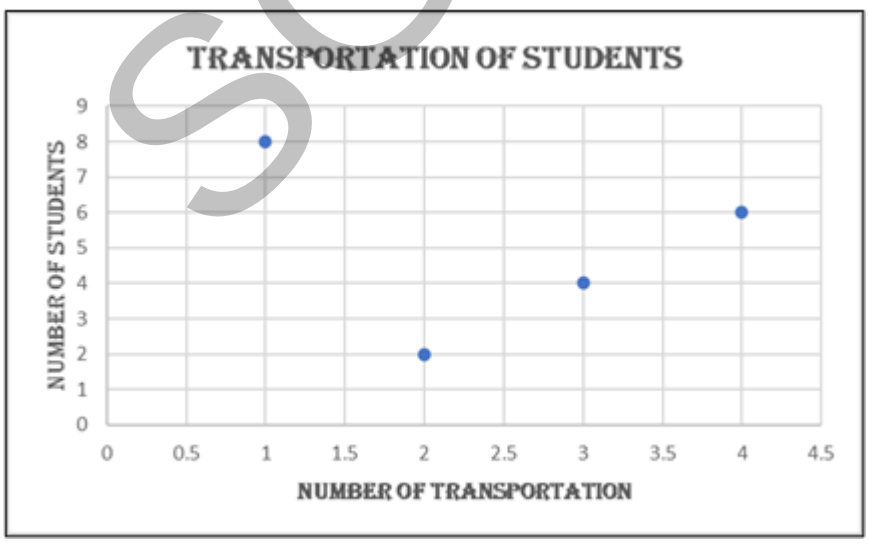


Fig. 3.1.10 Illustration of Bubble charts

3.1.7.11 Radar Charts (Spider Charts)

A radar chart (also known as a spider chart, web chart, or star plot) is a type of data visualization used to display multivariate data in a way that is easy to understand. It

is particularly effective for showing the relative strengths and weaknesses of different variables in a dataset, allowing for comparisons between multiple groups or entities across several dimensions. A radar chart consists of a central point with multiple axes radiating outward, each representing a different variable. Data values are plotted on these axes, and lines or areas are used to connect the data points, creating a web-like shape.

Advantages

- **Easy Comparison Across Multiple Variables:** Radar charts make it easy to compare multiple variables in a single view. You can see how one dataset performs across several dimensions and how different datasets compare in terms of relative strengths and weaknesses.
- **Effective for Visualizing Multidimensional Data:** They are ideal for visualizing data that has several dimensions (e.g., customer satisfaction across multiple categories, performance metrics across several departments).
- **Clear Trend and Pattern Identification:** Radar charts can help identify patterns or trends that might be difficult to spot with other chart types, especially when comparing multiple variables at once.
- **Visualizing Performance:** Radar charts are commonly used in performance analysis, such as comparing the performance of employees, products, or companies across different criteria.

Disadvantages

- **Limited Data Representation:** Radar charts are best used when there are a small number of variables (typically 3-7). With too many axes, the chart can become cluttered and difficult to read, reducing its effectiveness.
- **Difficult for Large Data Sets:** Radar charts can become overwhelming if there are too many categories or data series, as the chart becomes cluttered and harder to interpret.
- **Lack of Precision:** The positioning of data points on a radar chart does not always allow for easy reading of precise values. Unlike bar or line charts, where exact values are displayed, radar charts only give relative visual information.
- **Not Ideal for Showing Large Differences:** While radar charts are great for showing relative performance, they are not the best choice for emphasizing small differences between data points. Subtle variations in values may not be as apparent.
- **No Standardized Interpretation:** The appearance of a radar chart can be misleading if the axes are not properly scaled or if the data is not normalized, making interpretation more subjective.

We can represent the above Table 3.1.4 information using radar charts (Figure 3.1.11) shown below.

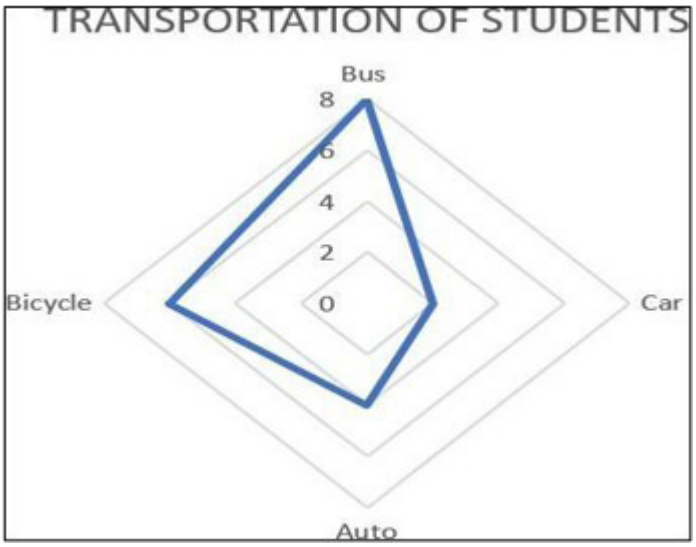


Fig. 3.1.11 Illustration of Radar charts
See the e-content for a color version of this image

3.1.7.12 Treemaps

A treemap is a hierarchical data visualization technique that displays data in a nested rectangular layout. It's used to represent hierarchical structures through the size and color of rectangles. The area of each rectangle is proportional to a specific value, and nested rectangles represent subcategories or child nodes of a parent category. Treemaps are particularly effective for visualizing proportions within hierarchical data, helping users to quickly identify patterns, trends, and outliers.

Treemaps are often used to show the composition of data across different levels of a hierarchy, with the most significant categories (or highest values) shown as the largest sections, making them easy to compare at a glance.

Advantages

- **Efficient Use of Space:** Treemaps maximize the use of available space by packing as many data points as possible into the area. They are excellent for visualizing large datasets with many categories or subcategories.
- **Easy to Compare Relative Sizes:** The size of the rectangles makes it easy to compare the values of different categories and subcategories at a glance.
- **Hierarchical Relationships:** Treemaps are ideal for displaying hierarchical data and understanding how individual elements contribute to the whole. They visually represent parent-child relationships, making it clear how subcategories relate to larger categories.
- **Visual Appeal:** Treemaps are visually appealing, especially when using colors effectively. The design makes it easier to spot patterns, trends, and outliers.
- **Identifying Proportions:** Treemaps help viewers quickly identify which categories or subcategories dominate a dataset, highlighting major contributors or areas of concern.

Disadvantages

- **Hard to Compare Precise Values:** Treemaps excel in showing relative proportions, but they are not ideal for representing precise values. For example, comparing the exact value of two rectangles can be difficult, especially when their sizes are similar.
- **Overcrowding:** When there are too many categories or levels in the hierarchy, the treemap can become overcrowded, leading to small rectangles that are hard to read and interpret. This can reduce the clarity of the visualization.
- **Complex Hierarchies:** When used to represent very deep or complex hierarchies, treemaps can become difficult to interpret, as multiple levels of nesting may make it hard to follow the data structure.
- **Limited to Hierarchical Data:** Treemaps are best suited for hierarchical datasets. If your data is not naturally hierarchical, using a treemap might not provide much insight and could be misleading.
- **Dependence on Effective Color Coding:** While color can be an advantage, it can also be a drawback if not used properly. Poor color choices or too many colors can lead to confusion or make it harder to discern patterns in the data.

We can represent the above Table 3.1.4 information using treemaps (Figure 3.1.12) shown below.

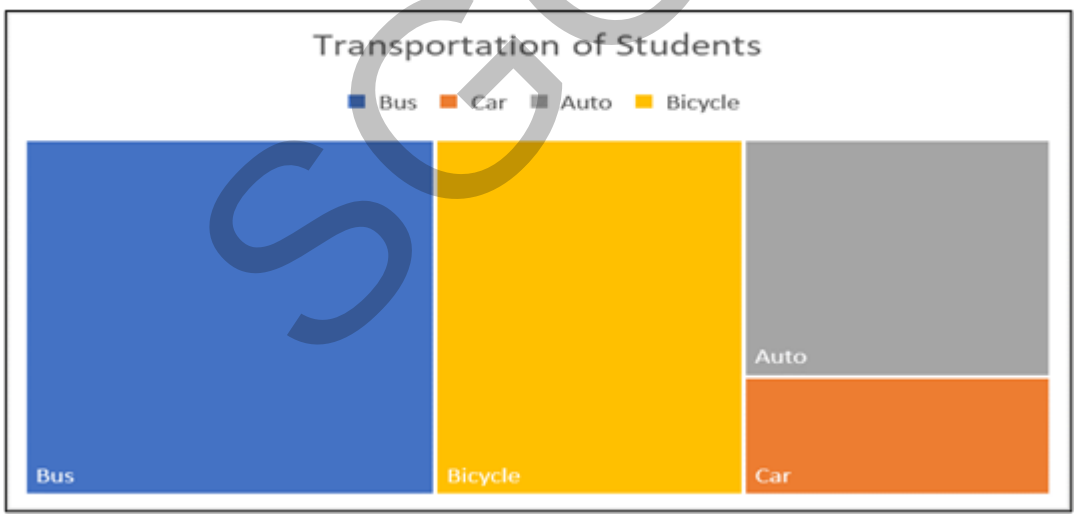


Fig. 3.1.12 Illustration of Treemaps
See the e-content for a color version of this image

3.1.7.13 Waterfall Charts

A waterfall chart is a type of data visualization used to display the cumulative effect of sequentially occurring positive and negative values. It is particularly useful for understanding the gradual impact of each individual item on a total, showing how an initial value is affected by a series of intermediate positive or negative values. Waterfall charts help in tracking the flow of data and are most commonly used for financial

analysis, showing how profits or losses accumulate over a period, but they can be applied in many contexts where understanding changes over time or categories is important.

Advantages

- **Clear Visualization of Cumulative Effects:** Waterfall charts are excellent for showing how a starting value is influenced by multiple factors, whether positive or negative. This makes them particularly useful in understanding how different components contribute to a final result.
- **Easy to Identify Drivers:** The sequential flow allows you to see what specific factors (positive or negative) have the largest impact on the final value, helping you quickly identify key drivers.
- **Simplicity and Clarity:** The chart is visually simple and easy to understand, as each step in the process is clearly marked. It's easy to follow the progression from one value to the next.
- **Useful for Financial Analysis:** Waterfall charts are widely used in finance to break down complex metrics like profit and loss, operating costs, and revenue changes over time. It helps stakeholders see how different elements of a business are contributing to the overall financial result.
- **Effective for Variance Analysis:** Waterfall charts can be used to analyze variances in budgets, forecasts, or other performance metrics, breaking down the reasons for deviations in a structured way.

Disadvantages

- **Limited to Sequential Data:** Waterfall charts are best suited for displaying data where changes occur sequentially (i.e., where the value is affected by a series of events or factors). They are not ideal for representing relationships between multiple variables simultaneously.
- **Difficult with Too Many Categories:** If there are too many intermediate changes, the chart can become cluttered, reducing clarity and making it difficult to interpret. In these cases, summarizing the changes into broader categories might help.
- **Lack of Exact Value Representation:** The chart shows relative changes between values, but it can be hard to pinpoint the exact numerical value of any given step unless the chart is very well-labeled. Exact figures may need to be cross-referenced with data tables or labels.
- **No Correlation Between Variables:** Waterfall charts do not show correlations or relationships between variables, making them unsuitable for certain types of analysis (e.g., comparing how multiple variables interact).
- **Not Ideal for Complex Data:** While great for simple, sequential changes, waterfall charts can become less effective when trying to display complex or large datasets with numerous variables and intricate relationships.

We can represent the above Table 3.1.4 information using waterfall charts (Figure 3.1.13) shown below.

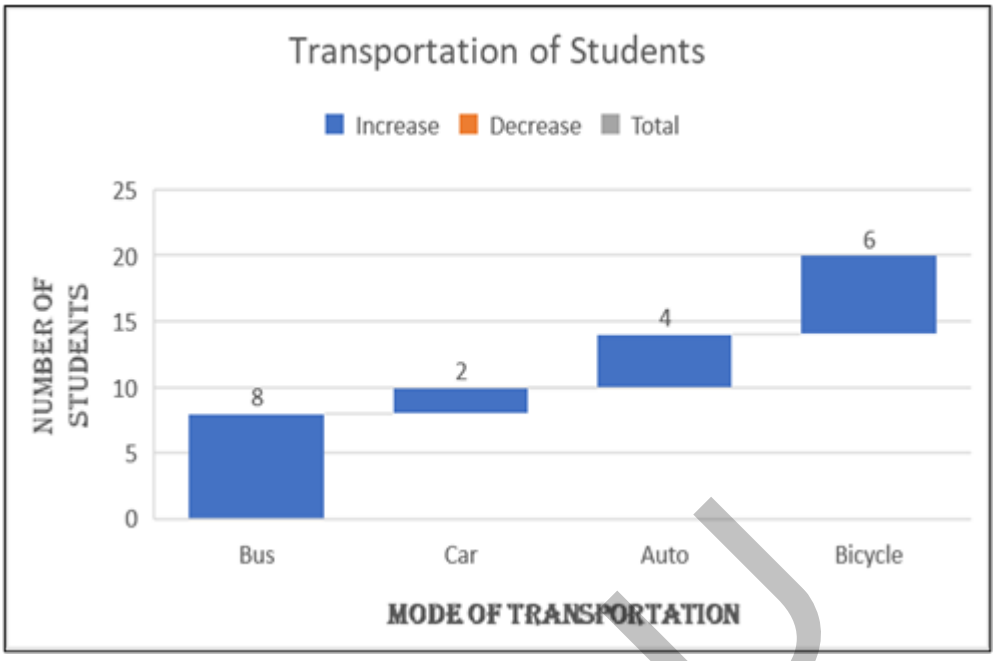


Fig. 3.1.13 Illustration of Waterfall charts
See the e-content for a color version of this image

3.1.7.14 Funnel Chart

A funnel chart is a type of data visualization that visually represents a sequential process with stages, showing how data flows from one stage to the next. It is particularly useful for representing the progression of a process and for highlighting drop-off rates between stages. The chart gets its name from the shape it creates: the widest part at the top represents the starting point, while the progressively narrower sections represent each subsequent stage, culminating in the final value at the bottom.

Funnel charts are commonly used in business, marketing, and sales to represent processes like sales pipelines, user conversion rates, or customer journeys, where the number of items or people typically decreases as they move through each stage.

Advantages

- **Clear Process Visualization:** Funnel charts are excellent for visualizing a process and clearly demonstrating how many items or opportunities make it from one stage to the next.
- **Effective for Highlighting Drop-off:** Funnel charts make it easy to spot where the largest drop-offs occur in a process, helping businesses identify areas for improvement.
- **Intuitive Layout:** The visual flow of the funnel chart is easy to understand, with the largest values at the top and the final outcomes at the bottom, making it intuitive for viewers.

- **Good for Conversion Analysis:** Funnel charts are widely used in sales, marketing, and e-commerce to track conversion rates and to analyze how potential customers move through different stages of the sales pipeline (e.g., from a lead to a sale).

- **Compact Visualization:** Funnel charts are space-efficient compared to other methods of tracking sequential processes, like stacked bar charts, and they provide a lot of insight in a simple, easy-to-read layout.

Disadvantages

- **Limited to Sequential Data:** Funnel charts are most effective when the data involves a clear sequence of stages. If your data doesn't follow a specific order or process, a funnel chart may not be an appropriate visualization tool.

- **Potential Over-Simplification:** While funnel charts provide a quick, high-level view of a process, they may oversimplify complex processes or hide important nuances, such as the reasons behind drop-offs between stages.

- **Not Ideal for Multiple Variables:** Funnel charts focus on a single flow or series of stages, so they are not effective for showing relationships between multiple variables or comparing different processes.

- **Not for Exact Analysis:** Funnel charts provide a visual representation of flow but are less useful when precise data analysis is needed. For example, users may not easily be able to read exact figures without additional context or labeling.

We can represent the above table 3.1.4 information using a funnel graph (Figure 3.1.14) shown below.

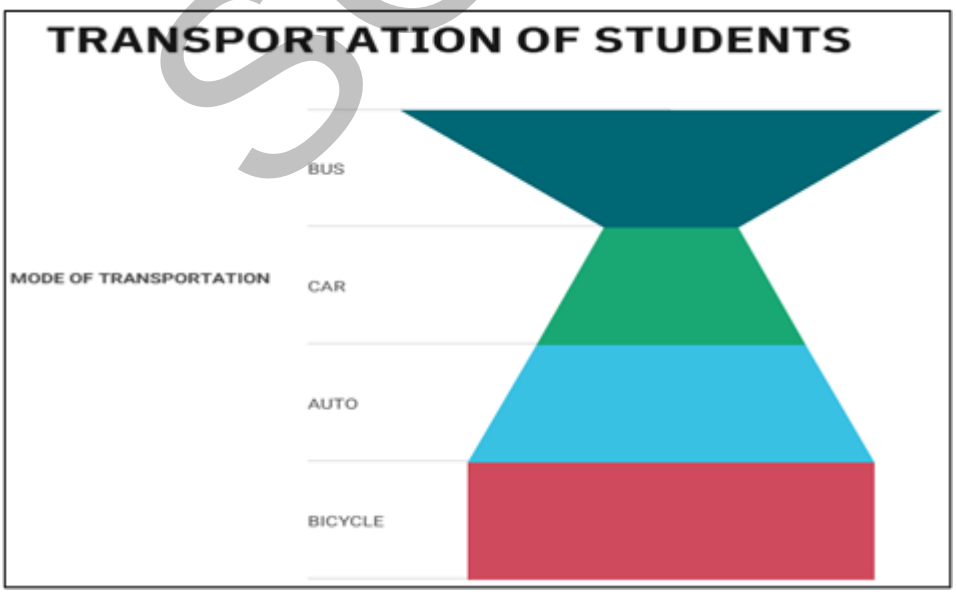


Fig. 3.1.14 Illustration of Funnel graph
See the e-content for a color version of this image

3.1.8 Visualization of Geospatial Data

Have you used Google maps while travelling? Mapmaking, or cartography, is the visualisation of geospatial data. It combines artistic elements to render data in a comprehensible manner for those without technical expertise while adhering to scientific principles to ensure the accuracy of the visual representation in relation to the underlying data. With Google Maps-based visualisation, you can visualise and interact with your geographic data just as you do with Google Maps: pan around, zoom in, even pop into Street View. Google Maps exemplifies the application of visualisation methods for geospatial data. It utilised diverse visualisation techniques to effectively represent geographic information.

Visualisation of geospatial data involves representing geographical information in a visual format, often using maps and spatially referenced data. It is a fascinating and powerful way to understand patterns and relationships related to locations on Earth. It essentially involves overlaying data variables onto a map, using latitude and longitude as the foundation. This helps us see information in a context that's intuitive and often easier to grasp than raw numbers and tables. Below is an overview of how Google Maps employs geospatial data visualisation.

Maps are the central element, acting as a container for data. They can range from displaying specific streets or parks to entire continents or the whole planet. These visualisations focus on the relationship between data and its physical location to create insight. Any positional data works for spatial analysis. What makes geospatial visualisations unique is the scale. A diagram of circuits on a microchip explores position, but it is not geospatial. It does not map to Earth or another planetary body. A map of the stars is also not considered geospatial, but a map of the surface of Mars is. Geovisualization overlays variables on a map using latitude and longitude to foster insight.

Maps are the primary focus of geospatial visualisations. They range from depicting a street, town, or park or subdivisions to showing the boundaries of a country, continent, or the whole planet. They act as a container for extra data. This allows you to create context using shapes and colour to change the visual focus. They help identify problems, track change, understand trends, and perform forecasting related to specific places and times. Geospatial visualisations highlight the physical connection between data points. This makes them susceptible to a few common pitfalls that may introduce error:

- Scaling: Changes in the size of the map can affect how the viewer interprets the data
- Auto-correlation: A view may create an association between data points appearing close on a map, even for unrelated data.

Geospatial visualisations can tell stories about human existence. Historically, doctors and scientists have used this kind of presentation to map illness, resources, and even simple navigation. In recent years, the most prevalent use of geospatial visualisations is likely through Google Maps and similar apps. They allow us to find the fastest way to travel from point A to point B or to identify where something is on Earth.

Geospatial visualisations are an essential tool in today's data-driven world, offering a dynamic way to represent information through maps and spatial formats. By combining geographic data with visual elements, these visualisations enable users to better understand complex patterns, relationships, and trends tied to specific locations. Whether in urban planning, environmental management, public health, or business analytics, geospatial visualisations enhance decision-making by making data more accessible, interactive, and meaningful. The key benefits of geospatial visualisations are explained below.

- Identifying patterns and trends: You can easily see clusters, hot spots, and geographic disparities by visually representing data on a map.
- Understanding relationships: Visualisation helps reveal how geographic factors might influence or be influenced by other variables.
- Effective communication: Visuals are often more engaging and accessible than raw data, allowing you to communicate insights to a wider audience.

Some common techniques for visualising geospatial data are briefly defined below with suitable images.

3.1.8.1 Choropleth Maps

A choropleth map is a type of thematic map in which areas are shaded or patterned in proportion to the value of a particular variable being represented. It is used to visualize geographic data and display variations in data across different regions, such as countries, states, counties, or even smaller units like neighborhoods or zip codes. The key feature of choropleth maps is their ability to represent statistical data in a spatial format, helping viewers to identify regional patterns, disparities, or trends in data.

Choropleth maps (Figure 3.1.15) are commonly used in fields like public health, economics, political science, and geography, where the aim is to show how a certain value or metric varies across geographical locations.

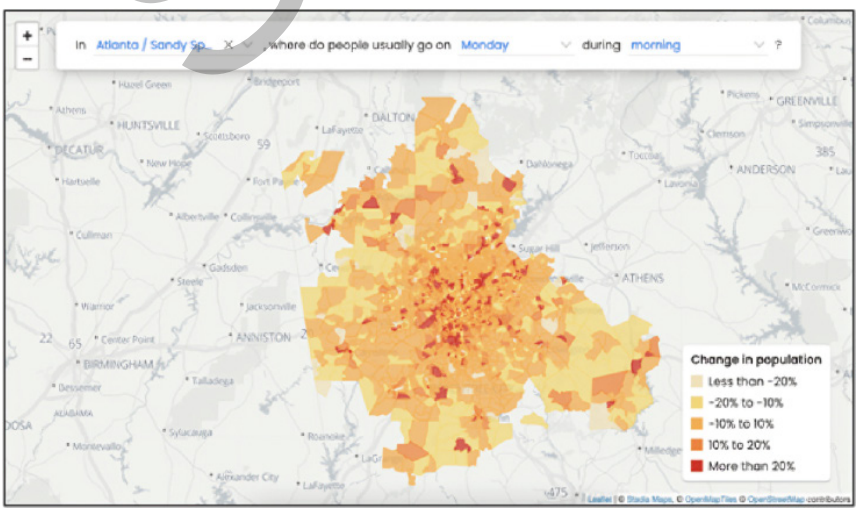


Fig. 3.1.15 Image of Choropleth Maps
See the e-content for a color version of this image

3.1.8.2 Point Maps

A point map is a type of geospatial visualization used to represent data points on a map based on specific geographic coordinates (latitude and longitude). These maps are highly effective for visualizing the location of discrete data items, such as individual events, objects, or occurrences, within a defined geographic space. Each point on a point map represents a single instance of data, and the position of the point on the map corresponds to its geographic location.

Point maps (Figure 3.1.16) are commonly used in a variety of fields, such as urban planning, crime analysis, logistics, environmental studies, and marketing, to identify patterns and trends in spatial distributions. By plotting the locations of data points on a map, point maps help analysts and decision-makers gain insights into the geographic aspects of the data.

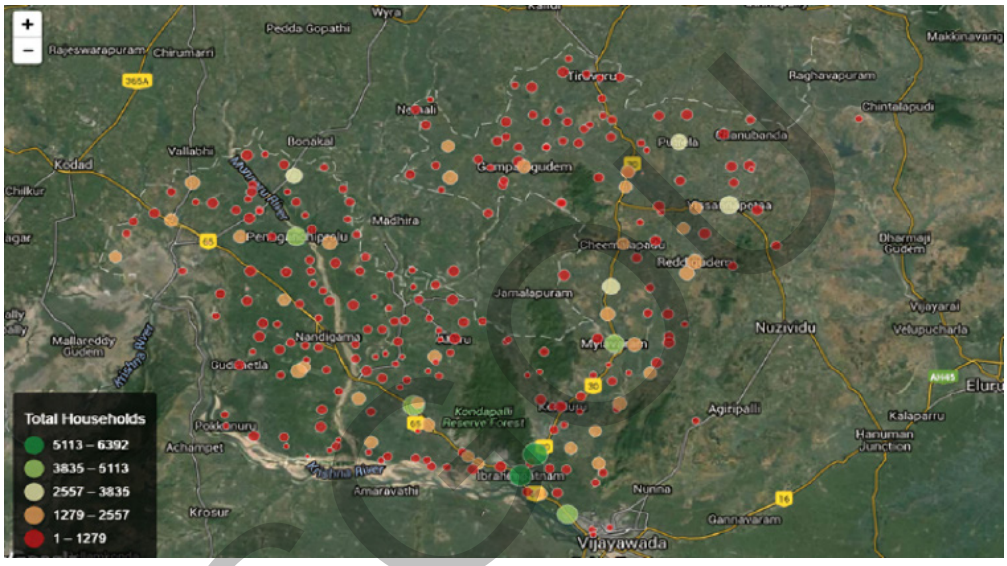


Fig. 3.1.16 Image of Point Maps
See the e-content for a color version of this image

3.1.8.3 Heatmaps

A heat map is a data visualization tool used to represent data in a matrix or grid format where individual values are represented by varying shades of color. The colors typically represent different ranges of values, with a color gradient often used to show intensity or magnitude, such as warmer colors (e.g., red or orange) indicating higher values and cooler colors (e.g., blue or green) indicating lower values. Heat maps are commonly used to visualize complex datasets where relationships between variables are not immediately apparent, helping to identify patterns, trends, or correlations at a glance.

Heat maps (Figure 3.1.17) are highly effective in showing the intensity of data points in a spatial or time-based context. They are widely used across various industries, including marketing, finance, healthcare, environmental science, and web analytics, to convey large amounts of information in a compact and easily interpretable format.

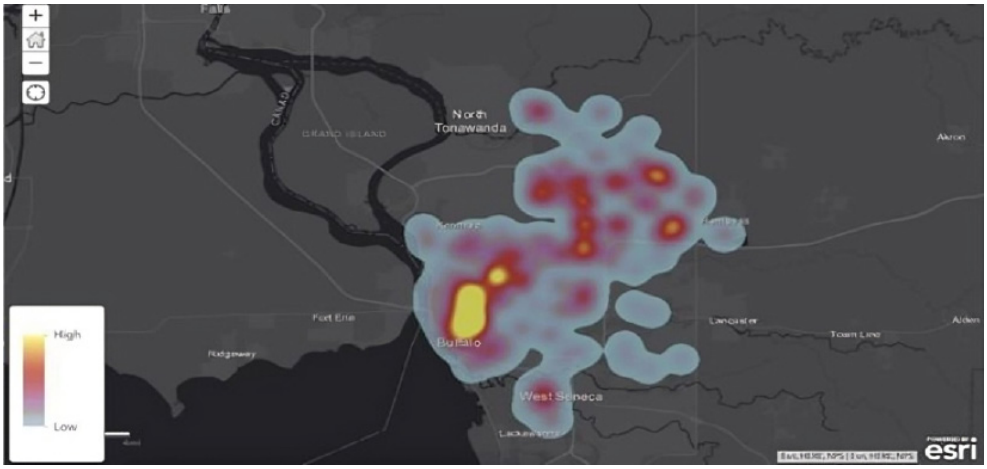


Fig. 3.1.17 Image of HeatMaps
See the e-content for a color version of this image

3.1.8.4 Proportional Symbol Maps

A proportional symbol map is a type of geospatial data visualization where symbols, usually circles, are placed over geographic locations to represent data values. The size of the symbol is directly proportional to the value of the data point it represents. This allows for the visualization of quantitative data across different geographic areas or locations, making it easier to compare values spatially. Unlike point maps, where each data point is represented by a fixed symbol, proportional symbol maps adjust the size of the symbols to convey information about the magnitude of the data.

Proportional symbol maps (Figure 3.1.18) are widely used in a variety of fields, including population studies, economics, environmental science, and urban planning, where it's important to show variations in a given variable across different regions. These maps help to visualize the distribution and magnitude of variables such as sales numbers, population density, or energy consumption, offering insights into spatial patterns and trends.

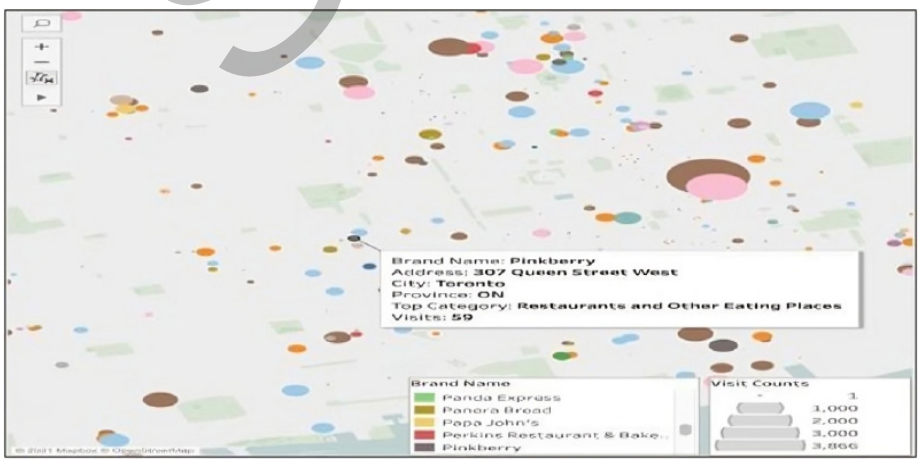


Fig. 3.1.18 Image of Proportional Symbol Maps
See the e-content for a color version of this image

3.1.8.5 Flow Maps

A flow map is a type of data visualization used to depict the movement of objects, people, goods, or information between different locations. It combines spatial geography with flowlines to illustrate the direction, volume, and magnitude of the flow between two or more points on a map. Unlike traditional maps that simply show locations or distributions, flow maps help visualize the dynamic movement across a region, offering insights into the connections and relationships between different geographic areas.

Flow maps (Figure 3.1.19) are particularly useful in understanding and analyzing patterns of migration, trade routes, traffic, energy flows, and information transfer. These maps help convey not just where things are happening, but how they are moving, which can provide valuable insights into the strength, direction, and volume of those movements.

Fig. 3.1.19 Image of Flow Maps
See the e-content for a color version of this image

3.1.8.6 Raster Maps

A raster map is a type of geospatial map where the data is represented in a grid format, with each cell or pixel in the grid holding a value that corresponds to a specific geographic location. This form of data representation is commonly used in Geographic Information Systems (GIS) and remote sensing, where continuous data (such as temperature, elevation, rainfall, or land cover) is captured and displayed over a particular geographic area.

Unlike vector maps, which represent geographic features as discrete objects (points, lines, or polygons), raster maps represent the earth's surface as a collection of individual cells or pixels arranged in a matrix. Each pixel in a raster map holds a data value, such as an image's color or an environmental measurement, making raster maps particularly effective for representing large-scale datasets that vary continuously over a region.

Raster maps (Figure 3.1.20) are commonly used in various fields, including environmental monitoring, urban planning, agriculture, climate studies, and remote sensing, where continuous geographic data needs to be represented in a visual format. Various types of raster maps are:

1. **Interactive Web Maps:** Interactive web maps leverage web-based mapping platforms and technologies, such as Google Maps or Leaflet.js, to create dynamic and customizable maps that allow users to explore and interact with geospatial data in real- time.

2. **3D Maps:** 3D maps add an additional dimension to traditional 2D maps, allowing for the visualisation of terrain, elevation, and building heights. They are useful for visualising landscapes, urban environments, and infrastructure projects.

3. **Time-Series Maps:** Time-series maps visualise changes in geospatial data over time. They allow users to observe trends, patterns, and spatial dynamics unfolding over different time periods.

Fig. 3.1.20 Image of Raster Maps
See the e-content for a color version of this image

3.1.8.7 Overlay Analysis

Overlay analysis is a critical spatial analysis technique used in Geographic Information Systems (GIS) to combine multiple layers of geospatial data to examine relationships, patterns, and interactions between different types of information. By overlaying different layers, analysts can identify areas of overlap, assess the combined effects of multiple variables, and make informed decisions based on integrated geographic data.

Overlay analysis (Figure 3.1.21) allows for the evaluation of how different spatial datasets intersect and how the attributes of each layer interact with each other. This technique is commonly used in fields such as environmental planning, urban development, disaster

management, land use planning, and resource management. The process generally involves three key steps:

1. **Data Collection:** Multiple geospatial datasets are gathered that represent different attributes of the geographic area of interest.

2. **Layer Alignment:** These datasets (layers) are aligned in a common coordinate system, ensuring that they correspond spatially.

3. **Analysis:** Overlay operations are carried out to examine how different layers interact with each other, revealing new spatial patterns, correlations, or potential issues.

Also, some key values are to be considered for this process. They are:

- **Clarity and accuracy:** Choose map types and colours that effectively communicate your message without misleading viewers.
- **Scale and aggregation:** Depending on the data and zoom level, consider aggregating points or using appropriate area units.
- **Context:** Add meaningful titles, legends, and annotations to provide clear understanding.

Applications of Overlay Analysis

1. **Environmental Impact Assessment:** Overlay analysis helps assess the impact of various human activities (e.g., construction, agriculture) on the environment. By overlaying environmental sensitivity maps with land-use maps, planners can identify areas that are vulnerable to degradation.

2. **Site Selection:** Overlay analysis can be used to identify the best locations for specific projects, such as locating new schools, hospitals, or factories. By considering factors like proximity to roads, population density, and environmental suitability, analysts can find optimal locations.

3. **Natural Disaster Management:** In disaster management, overlay analysis helps in assessing areas vulnerable to hazards like floods, earthquakes, or wildfires. By combining layers of risk (e.g., flood-prone areas, soil type, and population density), emergency response plans can be better prepared.

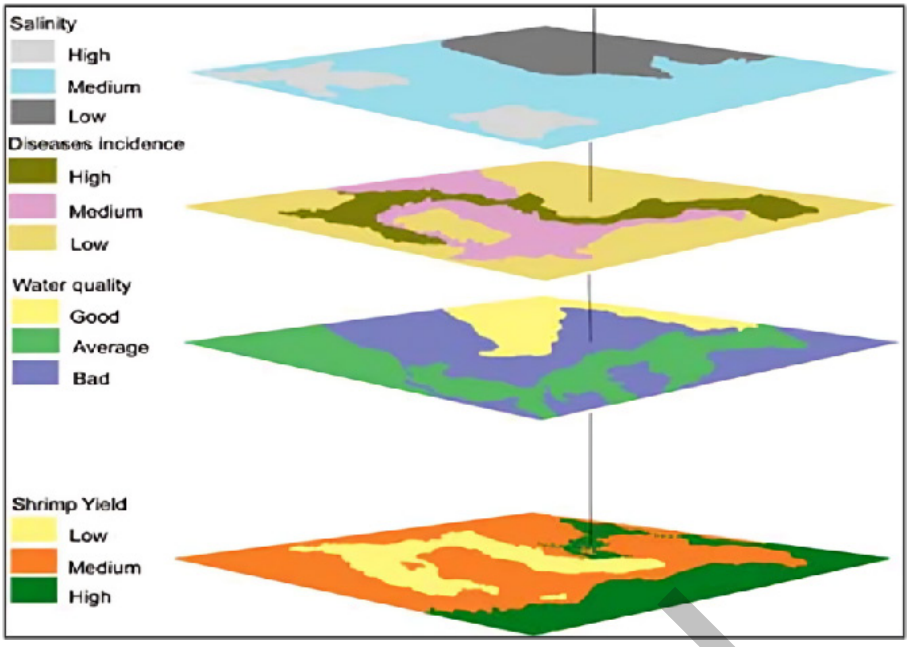


Fig. 3.1.21 Image of Overlay Analysis
See the e-content for a color version of this image

3.1.9 Numerical and Non-Numerical Data

Numerical data is commonly called quantitative data and non-numerical data is called qualitative data. This data is in the form of numbers. Any data in the form of numbers is numerical data. Data visualisation thrives on the interplay of numerical and non-numerical data, each playing a crucial role in painting a complete picture. Understanding their differences and how they can be combined effectively is key to creating impactful visualisations. By effectively wielding both numerical and non-numerical data, you can create data visualisations that are informative, engaging, and impactful.

3.1.9.1 Numerical Data

Numerical data, also known as quantitative data, refers to data that can be measured and expressed in numbers. It represents values that quantify characteristics or variables and can be subjected to mathematical operations such as addition, subtraction, averaging, and statistical analysis. Its key ideas are summarized as:

- **Quantitative variables:** Represents quantities and measurements: Age, income, temperature, sales figures, etc.
- **Analysis:** Numerical data visualisation enables quantitative analysis that is expressed as numbers, including statistical analysis, trend analysis, forecasting, and correlation analysis. It allows users to understand the numerical characteristics and behaviours of the data.
- **Visualisation Techniques:** Numerical data is often visualised using charts and graphs such as bar charts, line graphs, scatter plots, histograms, and box plots. These visualisations help to illustrate trends, patterns, distributions, and relationships within the data.

- **Strengths:**
    - Enables precise measurement and comparative analysis
    - Supports predictive modeling and statistical forecasting
    - Useful for identifying correlations and trends
    - Can be automatically processed by software and algorithms
- **Limitations:**
    - **Lacks context:** Numbers alone may not explain why something is happening.
    - **Can be misleading:** Poor graph design (like changing axis scales) can distort the message.
    - **Oversimplifies information:** It may ignore important details like emotions or reasons.
    - **Needs math skills:** Understanding charts and stats can be hard without basic knowledge.

3.1.9.2 Non-Numerical Data

Non-numerical data, also known as qualitative or categorical data, refers to information that cannot be measured or expressed using numbers. Instead, it describes characteristics, categories, or qualities and is used to label or group items based on attributes.

- **Categorical Variables:** Non-numerical data represents categories or labels and can include variables such as gender, ethnicity, marital status, and product categories.
- **Represents categories and qualitative attributes:** Colours, sizes, shapes, text, names, etc.
- **Expressed descriptively:** Categorizes information into groups.
- **Visualisation techniques:** Non-numerical data is visualised using techniques such as pie charts, bar charts, stacked bar charts, treemaps, and word clouds. These visualisations help to illustrate the distribution and proportions of different categories within the data.
- **Analysis:** Non-numerical data visualisation facilitates qualitative analysis, including comparison of categories, identification of patterns, and understanding of proportions and distributions. It allows users to explore the characteristics and relationships of categorical variables.
- **Strengths:**
    - Provides context and meaning
    - Captures subjective information

- Useful for categorization
- Supports detailed analysis

♦ **Limitations:**

- Hard to measure or compare
- Subjective and open to interpretation
- Difficult to visualize large datasets
- Requires coding or categorizing

3.1.9.3 Both numerical and non-numerical data

In many cases, data visualisation involves both numerical and non-numerical data together. For example, a bar chart may represent the sales figures (numerical data) of different product categories (non-numerical data). Similarly, a scatter plot may visualise the relationship between age (numerical data) and income (numerical data) across different demographic groups (non- numerical data).

♦ Enhanced storytelling: Use numerical data to provide factual evidence and non- numerical data to add context and humanise the story.

♦ Multi-dimensional insights: Reveal relationships between numerical and non- numerical variables, like income distribution across geographic regions.

♦ Engaging audience: Balance precision with relatability for broader appeal.

Examples:

♦ Sales data chart with colour-coded regions by product category: Combines sales figures (numerical) with product types (non-numerical) for deeper understanding.

♦ Network diagram of social connections: Uses nodes and edges to show relationships between individuals (non-numerical) with labels indicating age or profession (numerical).

♦ Choropleth map with income-based colour gradient and overlaid symbols for major cities: Combines income data (numerical) with city locations (non-numerical) for geographical analysis.

Recap

♦ Data visualization is the process of representing data visually to help people understand information quickly.

♦ The purpose of data visualization is to communicate data clearly, reveal insights, and support decision-making.

♦ Visual encoding refers to how data is represented using visual elements like position, size, color, and shape.

- Choosing the right visual format depends on the type of data and the message you want to convey.
- Design principles such as simplicity, clarity, and consistency help make visualizations effective and easy to interpret.
- Common challenges include avoiding misleading visuals, managing large datasets, and ensuring data accuracy.
- Quantitative data involves numerical values that can be measured and analyzed mathematically.
- Presenting quantitative data effectively often uses graphs like line charts, bar charts, histograms, and scatter plots.
- Different types of graphs and charts serve specific purposes, such as pie charts for proportions and scatter plots for relationships.
- Geospatial data visualization maps data tied to locations, showing spatial patterns using maps like choropleth and heat maps.
- Numerical data represents measurable quantities, while non-numerical data describes categories or qualities.
- Combining numerical and non-numerical data enhances the depth and meaning of visualizations.
- Common use cases for data visualization include business analytics, healthcare monitoring, environmental studies, and urban planning.
- Effective data visualization helps audiences grasp complex information quickly and supports data-driven decisions.
- Applying good design and choosing appropriate visual encodings are key to overcoming challenges and creating impactful visualizations.

Objective Type Questions

1. The practice of transforming data into a visual format such as a chart, graph, or map is called as ……
2. The mapping of data attributes to visual properties is called ……..
3. The representation of the information through pictures is called ……
4. Representation of hierarchical data as nested rectangles, where the size and color of each rectangle represent different attributes of the data is called ……

5. Distribution of quantitative data by grouping data into intervals, or bins, and plotting the frequency or density of data points within each interval is called …….

6. Data in the form of numbers is called…….

7. In which analysis combines multiple layers of geospatial data to identify spatial relationships and patterns?

8. Non-numerical data is also known as _____ data.

9. Heat maps are often used in _____ data visualization.

10. What type of data visualization helps identify outliers?

11. What is the first step in data visualization?

12. Which chart shows data as slices of a circle?

13. What is used to simplify complex data in visual form?

14. What is the key reason to choose the right chart type?

15. Which chart type compares values across categories?

Answers to Objective Type Questions

1. Data visualisation
2. Visual encoding
3. Pictograph
4. Treemaps
5. Histogram
6. Numerical data
7. Overlay analysis
8. Qualitative
9. Geospatial
10. Scatter plot

11. Data collection

12. Pie chart

13. Graph

14. Clarity

15. Bar graph

Assignments

1. Explain the purpose of data visualization. How does it support effective decision-making? Illustrate with suitable examples.

2. Discuss the key visual encoding techniques used in data visualization. How do these help in interpreting data accurately?

3. Describe the process of choosing the right visual format for a given dataset. What are the major design principles in data visualization?

4. Examine the challenges faced in data visualization. List and explain at least five common use cases of data visualization in different domains.

5. Compare and contrast different types of charts used for quantitative data presentation. When should one be preferred over another?

6. Discuss the visualization techniques for geospatial data. Include examples of tools or maps used.

7. How does the visualization of numerical data differ from non-numerical data? Provide examples of each.

Reference

1. Cairo, A. (2013). *The functional art: An introduction to information graphics and visualization*. Berkeley, CA: New Riders.

2. Tufte, E. R. (2001). *The visual display of quantitative information* (2nd ed.). Cheshire, CT: Graphics Press.

3. Kirk, A. (2016). *Data visualisation: A handbook for data driven design* (2nd ed.). London: SAGE Publications Ltd.

4. Ware, C. (2012). *Information visualization: Perception for design* (3rd ed.). Burlington, MA: Morgan Kaufmann.

5. Yau, N. (2013). *Data points: Visualization that means something*. Indianapolis, IN: Wiley.

6. Munzner, T. (2014). *Visualization analysis and design*. Boca Raton, FL: CRC Press.

7. Fry, B. (2008). *Visualizing data: Exploring and explaining data with the processing environment*. Sebastopol, CA: O'Reilly Media.

Suggested Reading

1. Data Visualization Catalogue https://datavizcatalogue.com
2. Information is Beautiful https://informationisbeautiful.net
3. FlowingData by Nathan Yau https://flowingdata.com
4. Observable https://observablehq.com
5. Datawrapper Blog https://blog.datawrapper.de
6. Kaggle https://www.kaggle.com/learn/data-visualization
7. Codeacademyhttps://www.codecademy.com/catalog/subject/data-visualization

Data Visualisation Methods

Learning Outcomes

At the end of this unit, the learner will be able to:

- define data visualization
- list types of data visualization analysis
- familiarize the steps in the data visualization life cycle
- narrate common visualization tools and elements

Prerequisites

Imagine you are part of a company's sales team. Every month, you receive huge spreadsheets full of numbers, rows and columns packed with data about sales, profits, customer regions, product categories, and more. It is your job to identify which products are doing well, which regions are underperforming, and what trends might affect your sales strategy.

At first, you try to make sense of it all by reading through the tables. But the more data you look at, the more confusing it gets. Important insights are buried in numbers, and small mistakes in interpretation can lead to big decisions being made for the wrong reasons.

Now imagine someone shows you a simple line graph. It instantly shows that sales dipped only in 2018, while every other year showed growth. In just seconds, you have gained an understanding that would have taken much longer with raw data alone.

Data visualization transforms complicated data into simple, visual formats that are easy to grasp. Whether you are a student, a business analyst, or a policymaker, having the ability to present data visually helps you identify patterns, make smarter decisions, and clearly share your insights.

By studying this unit, you will learn how to go beyond just numbers and create effective visuals like charts, graphs, maps, and dashboards. These tools will allow you to analyze single or multiple variables, helping you understand the full picture and present your findings in a clear and meaningful way.

Keywords

Exploratory Data Analysis, Slicer, Filter, Drill Down, Key Performance Indicators

Discussion

If you make a line graph showing the company's profits from 2013 to 2023, you can quickly see that profits went up every year except for a drop in 2018. This makes it easy to understand that the company did well most years, with just one year of loss. It would take more time to notice this by looking at a table of numbers, which shows why using visualization is helpful.

3.2.1 Data Visualization

Data visualization techniques involve using graphical tools such as charts, graphs, and maps to present information and data visually. These methods help simplify and clarify complex concepts for easier understanding. Various visualization approaches exist, each offering unique advantages and limitations. The most suitable method depends on factors like the nature of the data, the target audience, and the key message being communicated.

The main purpose of data visualization is to simplify data interpretation and improve accessibility, enabling users to quickly recognize patterns, trends, and anomalies. This becomes especially crucial in the context of big data, where the sheer amount of information can be overwhelming without clear and effective visual representation methods.

3.2.1.1 Types of Data Visualization Analysis

Data visualization is used to analyze visually the behavior of the different variables in a dataset, such as a relationship between data points in a variable or the distribution. Depending on the number of variables, you want to study at once, you can distinguish three types of data visualization analysis.

- **Univariate Analysis:** Used to summarize the behavior of only one variable at a time.
- **Bivariate Analysis:** Helps to study the relationship between two variables
- **Multivariate Analysis:** Allows data practitioners to analyze more than two variables at once.

3.2.2 Data Visualization Life Cycle

The data visualization cycle, also known as the Visual Analysis Cycle, refers to the different stages involved in creating an effective data visualization, from understanding the data to communicating insights to your audience as in Fig 3.2.1. It is an iterative

process that can be adapted to fit the specific needs of your project. It is not a linear process, but rather a continuous loop where each step informs and refines the others.

Fig. 3.2.1 Life Cycle of Data Visualization

1. Ask questions and define goals

Defining the question and identifying the audience is the initial phase of the data visualization life cycle. This involves clearly stating the specific objectives or questions that the visualization aims to address. It may include understanding what insights are needed, what decisions need to be supported, or what problems need to be solved. Equally important is determining who will be consuming the visualization and understanding their information needs and preferences. The audience may include stakeholders, decision-makers, analysts, or the general public. A clear understanding of the audience helps tailor the visualization to effectively communicate insights and engage the intended viewers.

2. Collect and clean data

Collecting relevant data from various sources such as databases, spreadsheets, or APIs (Application Programming Interfaces) is a critical step in the data visualization life cycle. The data can be structured or unstructured and may include numerical, categorical, or textual information. Once collected, the data often requires cleaning and preprocessing to address missing values, outliers, or inconsistencies. It is essential that the data is accurate, complete, and consistent. This stage involves tasks such as data imputation, normalization, and transformation to ensure the data is suitable for analysis.

3. Exploratory Data Analysis (EDA)

During EDA, analysts explore the dataset to understand its underlying patterns, trends, and outliers. Use statistical analysis techniques to identify relationships and draw conclusions. This stage often involves creating simple visualizations, such as histograms, scatter plots, and box plots, to gain insights into the data.

4. Visualization Design

Based on the insights gained from EDA, analysts design visualizations that effectively communicate the key findings and insights from the data. This stage involves selecting appropriate visualization types, colors, and layouts to convey the information clearly and accurately.

5. Visualisation Creation

In this stage, analysts use visualization tools and software to create the chosen visualizations, such as bar charts, line graphs, or heatmaps. Consider factors like the number of variables, data distribution, and desired level of detail. They may also customize the visualizations to meet specific requirements or audience preferences.

6. Interpretation and Analysis

Once the visualizations are created, analysts interpret the results and analyze the patterns, trends, and relationships depicted in the visualizations. This stage involves extracting actionable insights from the data to inform decision-making or further analysis.

7. Presentation and Communication

Finally, the insights derived from the visualizations are presented and communicated to stakeholders, decision-makers, or other relevant audiences. This stage may involve creating reports, dashboards, or presentations that effectively convey the key findings and recommendations derived from the data.

8. Feedback and Iteration

The data visualization cycle often involves an iterative process, where stakeholders provide feedback on the visualizations and insights presented. Based on this feedback, analysts may refine the visualizations, revisit the data analysis, or explore additional questions or hypotheses, leading to further iterations of the cycle.

3.2.3 Creating visualizations

Creating visualizations involves employing various tools and techniques to represent data in a meaningful and insightful manner.

3.2.3.1 Charts

Using charts for data visualization is a popular and powerful method to present data clearly. It turns complex information into simple visuals like graphs and charts, making it easier to spot patterns, trends, and connections that might not be obvious in raw data.

Table 3.2.1 Monthly Sales Data

Month	Sales (Rs)
January	5000
February	6000
March	7500
Aprill	8000
May	7000
June	8500

The Table 3.2.1 is a basic table chart showing monthly sales data. You could create this chart using spreadsheet software like Microsoft Excel or Google Sheets, or even with Markdown in a text editor.

3.2.3.2 Graphs

Graph-based data visualization, also known as Graph Visualization, Link Analysis, or Network Visualization, displays data in the form of a graph as in Fig 3.2.2. In this format, nodes stand for individual entities, while links or edges illustrate the connections or relationships between those entities.

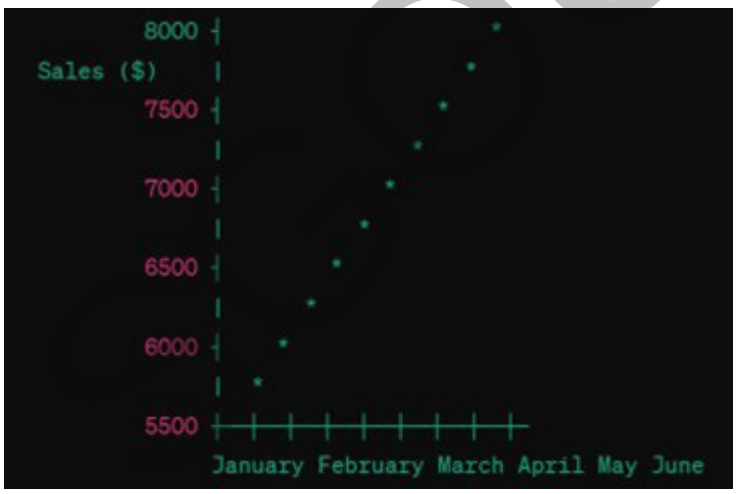


Fig. 3.2.2 Graphs
See the e-content for a color version of this image

In this line graph, the x-axis represents the months, and the y-axis represents the sales in dollars. Each point on the graph represents the sales for a particular month. The line connects these points, providing a visual representation of the trend in sales over the months.

3.2.3.3 Maps

A choropleth map is another common type of map. It is made by separating the area being mapped, such as by geographic or political boundaries, and then filling each resulting section with a different color or shade. Each color or shade represents a different variable, or a different value or range for a single variable. This makes choropleth maps

useful for visualising clusters of data across a geographic area while maintaining the context of regional boundaries as in Fig 3.2.3.

Let us represent the sales of electronics products in different districts of Kerala using a choropleth map, allowing stakeholders to quickly identify areas with higher and lower sales of electronics products.

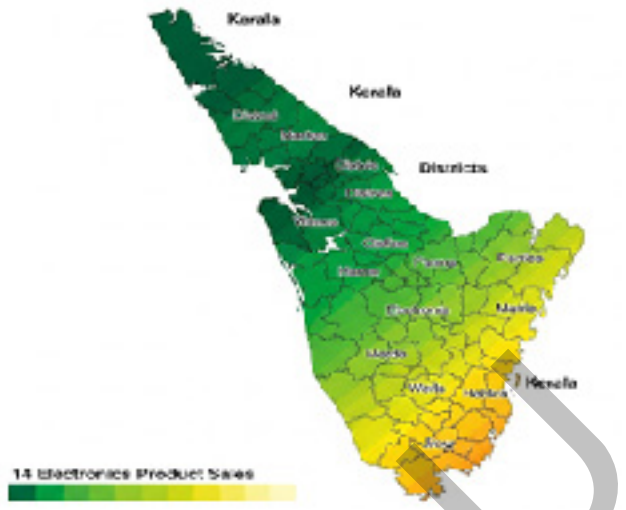


Fig. 3.2.3 Choropleth Map
See the e-content for a color version of this image

The color intensity (from dark green to yellow) represents the sales volume of electronics products in that district.

Districts with higher sales are shaded dark green, those with moderate sales are light green, and those with low sales are yellow.

3.2.3.4 Key Performance Indicators (KPIs)

A KPI, or Key Performance Indicator, is a quantifiable metric used to evaluate how effectively a specific objective is being met over time. Organizations use KPIs to establish goals and monitor progress, enabling departments such as finance and marketing to make informed decisions. Essentially, KPIs serve as tools to assess performance and drive improvement across different areas of a business as in Fig 3.2.4.

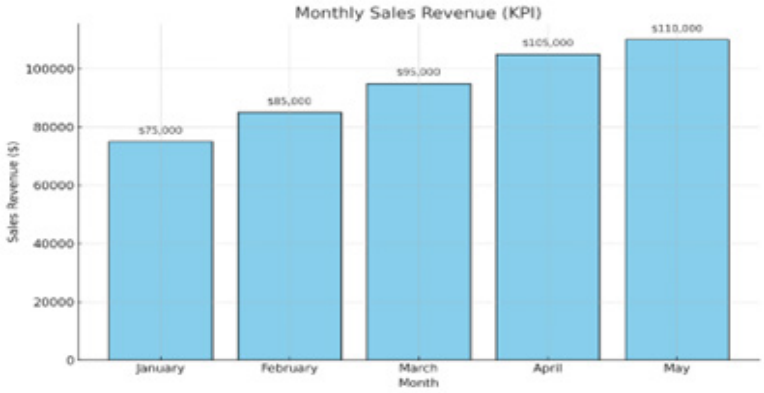


Fig. 3.2.4 KPI of Monthly Sales Revenue
See the e-content for a color version of this image

The x-axis represents each month and the y-axis represents the total sales revenue. Each bar represents the sales revenue for a particular month. By visually representing monthly sales revenue in a bar chart like this, stakeholders can quickly understand trends, identify patterns, and make data-driven decisions to improve sales performance. They can easily compare performance across different months and track progress towards revenue.

3.2.3.5 Slicers

Slicers are interactive, user-friendly buttons or panels commonly used in data visualization tools like Excel, Power BI, and Tableau. They allow users to visually filter data based on specific fields such as region, product category, or time. The primary purpose of slicers is to provide a quick and intuitive way to refine data views without needing to apply complex filters manually as in Fig 3.2.5. When a user clicks on a slicer item such as selecting "North" in a Region slicer, the dashboard instantly updates to display only the relevant data for that selection. Additionally, multiple slicers can be used simultaneously to filter data across different dimensions, like combining Region and Product Category, enabling more focused and dynamic analysis.

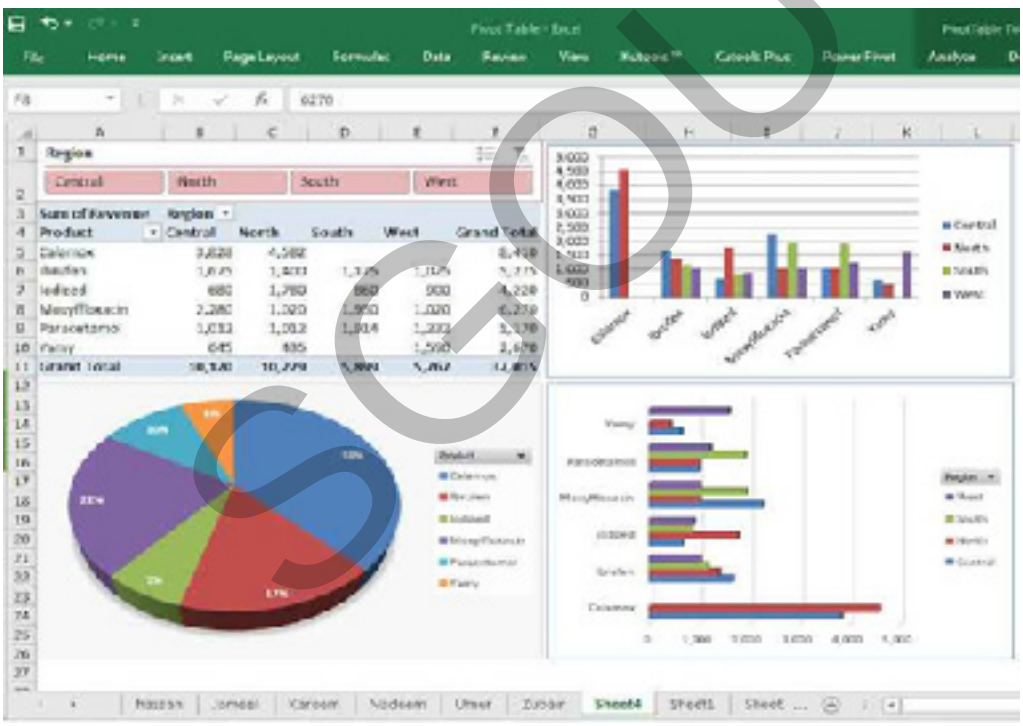


Fig. 3.2.5 Slicers
See the e-content for a color version of this image

3.2.3.6 Filters

Filters are tools used in data visualization to narrow down data based on specific conditions, allowing users to focus on relevant subsets of information as in Fig 3.2.6. They can be applied through manual input or dropdown selections, depending on the data visualization tool being used, such as Excel, Power BI, or Tableau. The main purpose of filters is to include or exclude particular data points based on defined criteria, which helps streamline analysis and uncover insights more efficiently.

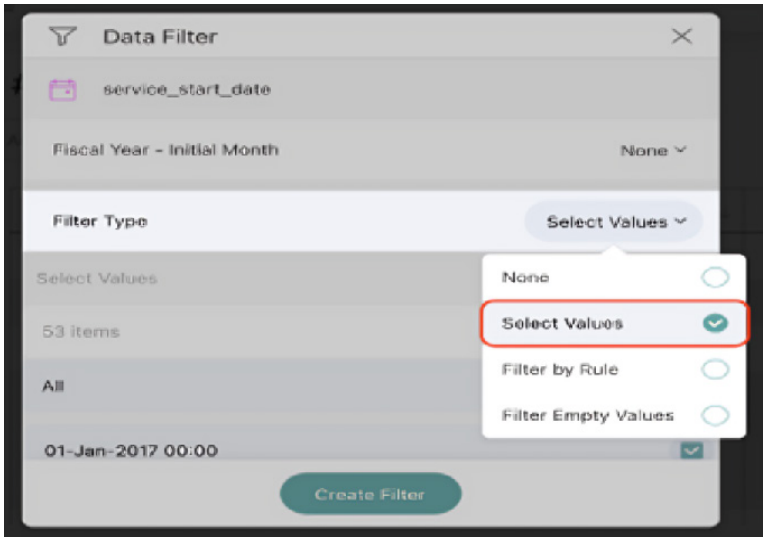


Fig. 3.2.6 Filter
See the e-content for a color version of this image

There are generally two main types of filters: Manual Filters and Conditional Filters.

Manual filters involve selecting specific items from a list, such as choosing "Product A" and "Product B" to analyze only those products. Conditional filters, on the other hand, are rule-based for example, displaying only records where sales are greater than $100,000 or filtering data for a specific date like "January 2025." By using filters, users can customize their view of the data to suit their specific analytical needs, leading to more targeted and meaningful insights.

3.2.3.7 Drill Through

Drill Through is a powerful feature in data visualization tools that allows users to interact with a summarized data point and navigate to a separate, more detailed report specific to that item. This functionality is especially useful for keeping the main dashboard clean and uncluttered while still enabling access to granular data when needed. The primary purpose of Drill Through is to provide a deeper level of insight into specific segments of the data without overwhelming users with too much information on the main view.

A user views a high-level summary such as total sales by region and decides to explore the “East Region” in more detail. By clicking on the data point for the East Region, the dashboard redirects to a new page or tab where detailed records are displayed. These might include individual transactions, a list of customers from that region, or product-level sales figures. This drill-through capability enhances data exploration and decision-making by linking summary and detailed views in a seamless and intuitive way. It empowers users to investigate anomalies, validate figures, or gain a clearer understanding of underlying patterns all without manual data digging or switching tools.

3.2.3.8 Drill Down

Data visualization using drill-down techniques involves presenting data in a hierarchical manner, allowing users to navigate through different levels of detail. This approach enables users to explore data progressively, starting from an overview and then diving into specific details as needed.

The following steps outline the process for implementing data visualization with drill-down functionality:

1. **Choose a Visualisation Tool:** Select a suitable data visualization tool that supports drill-down features. Popular options include Tableau, Power BI, and QlikView.

2. **Prepare Your Data:** Ensure your data is properly cleaned, structured, and organized. This might involve aggregating data at different levels or creating hierarchies within your dataset.

3. **Select an Initial Visualisation:** Start by choosing an initial visualization that provides a high- level overview of your data. This could be a bar chart, pie chart, treemap, or any other suitable visualization type depending on your data and objectives.

4. **Implement Drill-Down Interactivity:** Enable drill-down functionality in your chosen visualization tool. This typically involves configuring interactions or actions that allow users to navigate to more detailed levels of data. For example, clicking on a bar in a bar chart might drill down to a more detailed breakdown of that category.

5. **Define Drill-Down Paths:** Determine the hierarchical structure of your data and define the drill-down paths accordingly. This could involve drilling down from region to country to city, or from year to quarter to month, depending on your data dimensions.

6. **Create Visualizations for Detailed Levels:** Once users drill down to a specific level, present more detailed visualizations that provide deeper insights into the data. These visualizations should be tailored to the specific level of detail being explored.

7. **Provide Navigation Controls:** Include navigation controls such as back buttons or breadcrumbs to allow users to easily navigate between different levels of detail.

8. **Test and Iterate:** Test your drill-down visualizations with users to ensure they are intuitive and provide value. Gather feedback and iterate on your designs as needed to improve usability and effectiveness.

9. **Consider Performance:** Be mindful of performance considerations, especially when dealing with large datasets. Optimise your visualizations and data queries to ensure smooth and responsive drill-down experiences.

10. **Document and Train Users:** Document the drill-down paths and functionality and provide training to users so they can effectively utilise the drill-down features to explore and analyse the data.

3.2.3.9 Data visualization using Custom Visuals

Data visualization using custom visuals involves creating tailored visualizations that address specific data analysis requirements beyond what standard visualization tools offer. By the following steps, you can create custom data visualizations that effectively communicate insights, facilitate data exploration, and empower users to make informed decisions based on their data.

1. **Identify Requirements:** Understand the specific data analysis needs and objectives. Determine the key insights stakeholders are looking to derive from the data.

2. **Choose a Platform or Framework:** Select a platform or framework for creating custom visuals. Common options include D3.js, Chart.js, Highcharts, Plotly, and custom development using libraries like React or Angular.

3. **Prepare Your Data:** Ensure your data is clean, structured, and prepared for visualization. Perform any necessary data preprocessing, aggregation, or transformation.

4. **Design Custom Visualizations:** Design custom visualizations that effectively communicate the insights you want to convey. Consider factors such as chart type, color scheme, labelling, and interactivity.

5. **Implement Interactivity:** Add interactive features to your custom visuals to enhance user engagement and exploration. This could include tooltips, zooming, panning, filtering, and drill- down functionality.

6. **Optimise Performance:** Optimise your custom visuals for performance, especially when dealing with large datasets or complex visualizations. Minimise rendering times and optimise data processing operations.

7. **Test Across Platforms:** Test your custom visuals across different platforms and devices to ensure compatibility and responsiveness. Address any compatibility issues or rendering inconsistencies.

8. **Integrate with Existing Tools:** If applicable, integrate your custom visuals with existing data visualization tools or platforms such as Tableau, Power BI, or Excel. This allows users to leverage the benefits of custom visuals within familiar environments.

9. **Document and Share:** Document your custom visuals, including their design rationale, usage instructions, and any dependencies. Share your custom visuals with stakeholders and provide training or support as needed.

10. **Iterate Based on Feedback:** Gather feedback from users and stakeholders and iterate on your custom visuals based on their input. Continuously refine and improve your visuals to better meet the needs of your audience.

11. **Stay Updated:** Stay informed about advancements in data visualization techniques, technologies, and best practices. Incorporate new ideas and innovations into your custom visuals to keep them relevant and effective.

3.2.3.10 Data Visualization using Publishing a Report

Publishing a report involves creating visualizations and insights from data and then sharing them in a consumable format with stakeholders. By the following steps, you can effectively visualize data and publish reports that provide valuable insights and drive informed decision-making within your organisation or community.

1. **Define Objectives:** Understand the purpose of the report and the key questions it should answer. Determine the target audience and their information needs.

2. **Gather and Prepare Data:** Collect relevant data sources and clean, preprocess, and integrate them as needed. Ensure data quality and consistency.

3. **Select Visualization Tools:** Choose appropriate tools for creating visualizations and reports. Popular options include Tableau, Power BI, Google Data Studio, and Excel.

4. **Design Visualizations:** Create visualizations that effectively communicate insights and support the objectives of the report. Choose the appropriate chart types, color schemes, and labelling for clarity and impact.

5. **Organise Report Structure:** Define the structure of the report, including sections, chapters, or pages. Plan the flow of information and how visualizations will be presented.

6. **Create Dashboards and Reports:** Use the selected tool to design dashboards or reports. Arrange visualizations logically and include relevant annotations or commentary to provide context.

7. **Add Interactivity:** Incorporate interactive features such as filtering, drill-down, or tooltips to enhance engagement and exploration of the data.

8. **Validate and Test:** Review the report to ensure accuracy, completeness, and adherence to design principles. Test interactivity and responsiveness across different devices and screen sizes.

9. **Publish and Share:** Publish the report using the chosen platform or distribution method. Share it with stakeholders via email, presentations, or online portals.

10. **Promote Understanding:** Provide explanations and interpretations of the visualizations to aid understanding. Highlight key insights and trends to guide stakeholders in their analysis.

11. **Collect Feedback:** Encourage feedback from stakeholders and users to identify areas for improvement. Use feedback to refine future reports and visualizations.

12. **Monitor Usage and Impact:** Track the usage of the report and its impact on decision-making or actions. Use analytics to assess engagement and effectiveness.

13. **Iterate and Update:** Regularly update the report with new data or insights. Iterate on the design and content based on feedback and changing requirements.

Recap

- Data visualization transforms raw data into visuals like charts, graphs and maps.
- Makes complex data easier to understand and interpret.
- Helps identify patterns, trends and anomalies.
- Especially useful for analysing large datasets.
- Types of data visualization analysis:
    - Univariate: analyzes one variable.
    - Bivariate: explores the relationships between two variables.
    - Multivariate: examines more than two variables simultaneously
- Data visualization life cycle:
    - Ask questions and define goals.
    - Collect and clean data.
    - Perform Exploratory Data Analysis (EDA)
    - Create visualizations(Select chart types, colors, layout)
    - Create the visualizations using tools.
    - Interpret and analyze visuals.
    - Present findings to stakeholders.
    - Collect feedback and improve visuals.
- Common visualization elements:
- Charts (bar, pie, line) to show comparisons and trends.
- Graphs with nodes and edges to represent relationships.
- Maps(e.g., choropleth maps) to visualize geographic data.

- KPIs to track key metrics like revenue or performance.
- Slicers as interactive filters for dashboards.
- Filters to narrow down data based on conditions or values.
- Drill through to navigate from summary data to detailed data.
- Drill down to explore data Hierarchies.
- Custom visuals built for specific needs using tools like D3.js or plotly.
- Publishing reports involves designing, preparing data, creating visuals, and sharing with stakeholders using tools like Power BI or Tableau.

Objective Type Questions

1. What type of analysis involves examining only one variable?
2. Which tool uses geographic boundaries with color intensity to show data?
3. Which KPI chart type is used to represent monthly sales revenue?
4. What is the initial phase of the data visualization life cycle?
5. Which interactive feature allows filtering by clicking visual elements like buttons?
6. Which technique allows users to explore data progressively from summary to detail?
7. Which type of filter displays records based on rules like “greater than 100.000”?
8. Which type of analysis examines the relationship between two variable?
9. Which step in publishing a report focuses on checking accuracy and interactivity?
10. What feature enables navigation from summary data to detailed reports?

Answers to Objective Type Questions

1. Univariate
2. Choropleth
3. Bar
4. Questions
5. Slicers
6. Drilldown
7. Conditional
8. Bivariate
9. Validate
10. Drillthrough

Assignments

1. Explain the Data Visualization Life Cycle and describe each stage involved in creating an effective data visualization.
2. Differentiate between Univariate, Bivariate, and Multivariate data visualization analyses with suitable examples.
3. Describe the importance of Exploratory Data Analysis (EDA) in the data visualization process. How does it help in understanding patterns and trends in data?
4. What are Slicers and Filters in data visualization tools? Compare their roles in refining and analyzing datasets effectively.
5. Define Drill Down and Drill Through techniques. How do these features enhance interactivity and depth in data visualization?
6. What are Key Performance Indicators (KPIs)? Explain their significance in monitoring business goals and decision-making.
7. Discuss the steps involved in Publishing a Data Visualization Report. How does effective report design support data-driven decision-making?

Reference

1. Murray, S. (2017). *Interactive data visualization for the web: An introduction to designing with D3* (2nd ed.). O'Reilly Media.

2. Ware, C. (2013). *Information visualization: Perception for design* (3rd ed.). Morgan Kaufmann.

3. Yau, N. (2013). *Data points: Visualization that means something*. Wiley.

4. Kirk, A. (2016). *Data visualization: A handbook for data driven design*. SAGE Publications.

5. Evergreen, S. D. H. (2016). *Effective data visualization: The right chart for the right data*. SAGE Publications.

Suggested Reading

1. Few, S. (2012). *Show me the numbers: Designing tables and graphs to enlighten* (2nd ed.). Analytics Press.

2. Knaflic, C. N. (2015). *Storytelling with data: A data visualization guide for business professionals*. Wiley.

3. Cairo, A. (2016). *The truthful art: Data, charts, and maps for communication*. New Riders.

4. Tufte, E. R. (2001). *The visual display of quantitative information* (2nd ed.). Graphics Press.

5. Healy, K. (2018). *Data visualization: A practical introduction*. Princeton University Press.

BLOCK 4
Familiarisation of Data Analysis Tools

Introduction to Data Analysis Using R

Learning Outcomes

At the conclusion of this unit, the learner will be able to:

- to know how to install and run R programming language
- to understand syntax and semantics of R programming language
- to understand different data types used in R
- to get proficiency in creating visualisations using R's graphics systems

Prerequisites

Imagine you are a data analyst working for a hospital. Every day, you receive thousands of patient records such as age, blood pressure, sugar level, and treatment details. To find useful insights such as which age group is most affected by diabetes or which treatment gives the best results, you need a tool to clean, analyze, and visualize this data.

R helps you do all these tasks efficiently through a few lines of code and beautiful graphs.There are several tools available in the market to perform data analysis. Learning new languages takes time. The data scientist can use two excellent tools, i.e., R and Python. We may not have time to learn them both at the time when we get started to learn data science. Learning statistical modelling and algorithms is more important than learning a programming language. A programming language is used to compute and communicate our discovery. R communicates with the other languages and possibly calls Python, Java, C++. The big data world is also accessible to R.

Keywords

R syntax, data types, operators, list, matrices, vectors, data frames

Discussion

4.1.1 Discussion

R draws heavily from the S language, originally developed during the 1960s and 1970s by researchers at Bell Laboratories in New Jersey. R developed by Ross Ihaka and Robert Gentleman from the University of Auckland in New Zealand, released in the early 1990s under the GNU public licence, functions as a platform for statistical analyses and visualisations.

The software was aptly named after the shared first initial of Ross and Robert. Since then, R has surged in popularity due to its unparalleled flexibility for data analysis and robust graphical tools, all available at no cost. Perhaps its most enticing aspect is that any researcher can contribute code in the form of packages (or libraries), ensuring rapid access to advancements in statistics and data science for the global community.

R plays a crucial role in numerous research and data analysis projects by providing access to a wide range of modern statistical methods, both simple and advanced, in a readily available and user-friendly manner. Nevertheless, newcomers to R often lack experience in programming overall. As a novice, you must not only acquire proficiency in using R for your particular data analysis objectives but also cultivate a programmer's mindset. This partially explains why R is sometimes perceived as "challenging" — however, rest assured, this reputation is not entirely justified.

Its ongoing development and dissemination are overseen by a group of statisticians recognized as the R Development Core Team. R is accessible in various versions, including the source code primarily written in C with some routines in Fortran, primarily tailored for Unix and Linux systems, as well as pre-compiled binaries compatible with Windows, Linux, and Macintosh. The necessary files for installing R, whether sourced from the original code or pre-compiled binaries, are distributed via the Comprehensive R Archive Network (CRAN) website.

R offers numerous functions for statistical analyses and graphics, with the latter instantly visualised in dedicated windows and capable of being saved in various formats such as jpg, png, bmp, ps, pdf, emf, pictex, and xfig (though format availability might vary based on the operating system). Results from statistical analyses are promptly displayed on-screen, with the option to save intermediate results like P-values, regression coefficients, and residuals either in files or for use in subsequent analyses.

Moreover, the R language empowers users to program loops for analysing multiple datasets sequentially. Additionally, it allows for the integration of diverse statistical functions within a single program to conduct more intricate analyses.

4.1.2 Key Features of R

- Open-source and free: Making it freely available for anyone to use, modify, and distribute. This open-source nature has contributed to its widespread adoption and continuous improvement by a global community of developers.

- Cross-platform: R is cross-platform compatible, running on major operating systems such as Windows, macOS, and Linux. This ensures flexibility and accessibility for users across different environments.
- Statistical Computing and Graphics: R offers a vast array of built-in functions and packages for statistical analysis, modelling, and visualisation. It provides comprehensive tools for exploratory data analysis, hypothesis testing, regression analysis, time-series analysis, clustering, and more.
- Powerful data visualisation: Users can create high-quality plots, charts, and interactive visualisations to explore and communicate insights effectively.
- Rich ecosystem of packages: boasts a rich ecosystem of packages contributed by users worldwide, covering various domains such as bioinformatics, finance, machine learning, and social sciences. These packages extend the functionality of R, enabling users to access advanced algorithms and techniques with ease.
- Object-oriented: Facilitates modular and reusable code.
- Interpretive language: Code executes line by line, allowing for interactive exploration and debugging.
- Interactive Environment: R provides an interactive environment through its command- line interface or integrated development environments (IDEs) like RStudio. Users can execute commands, run scripts, and visualise data in real-time, facilitating an iterative and exploratory workflow.
- Data Handling: R excels in data handling capabilities, allowing users to import data from various file formats (e.g., CSV, Excel, SQL databases) and manipulate data efficiently using built-in functions or packages like dplyr and tidyr. It supports handling large datasets and complex data structures like matrices, data frames, and lists.
- Reproducible Research: R facilitates reproducible research by integrating with tools like R Markdown and knitr, enabling users to create dynamic documents that combine code, results, and narrative text. This promotes transparency, collaboration, and the dissemination of research findings.

4.1.3 How to Install R

To install R,

1. Go to the official CRAN (Comprehensive R Archive Network) website at https://cran.r- project.org/
2. Download the latest version of R for Windows, Mac or Linux.
3. Choose a CRAN mirror: On the CRAN website, you'll find a list of mirrors. Choose the mirror closest to your location by clicking on its link.

4. Choose the link corresponding to your version of Windows (e.g., Windows 10). This will start the download process for the R installer file (e.g., "R-4.x.x-win.exe").

5. Follow the instructions in the installation wizard. You can generally accept the default settings unless you have specific preferences.

6. After the installation process is complete, you can launch R by double-clicking its icon on the desktop or finding it in the Start menu.

When you have downloaded and installed R, you can run R on your computer.

The screenshot below shows how it may look like when you run R on a Windows PC:

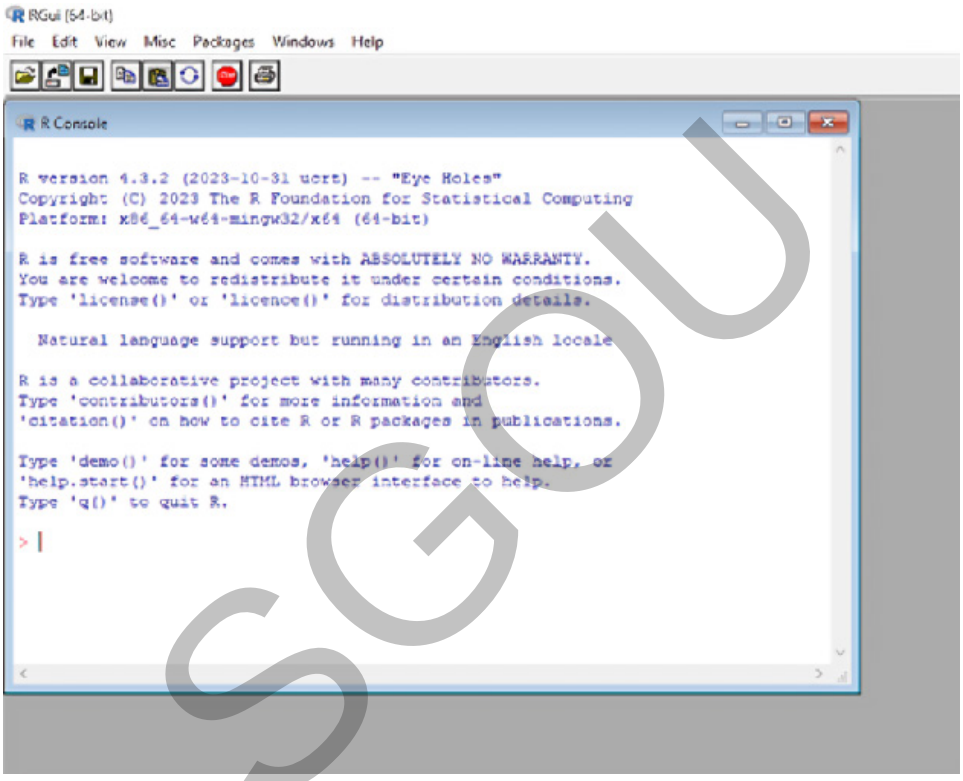


Fig. 4.1.1 Sample screen shot
See the e-content for a color version of this image

4.1.4 Fundamentals of R

4.1.4.1 Print "Hello world" using R

In R, we can simply print messages in two ways:

1. Simply print messages in double or single quotes without any function Syntax

   "Hello world"

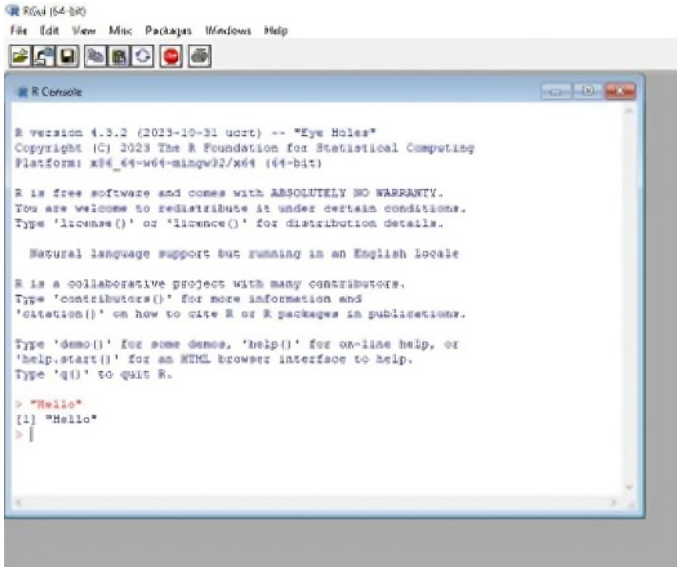


Fig. 4.1.2 Sample screen shot
See the e-content for a color version of this image

2. Also we can use print() function

   Syntax

   print("Hello world")

   It is up to you whether you want to use the print() function to output code. However, when your code is inside an R expression (e.g. inside curly braces {} like in the example above), use the print() function to output the result.

For Example:

```
for (x in 1:10) {
print(x)
  }
```

```
Output
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
```

Fig. 4.1.3 Output screen

4.1.4.2 To output numbers

To output a number just type the number in the console without quotes.

Syntax:

```
5
```

Output:

```
5
```

4.1.4.3 Simple calculations

We can do simple calculations by directly typing in the console without any coding with the help of operators.

Example: To add two numbers 6, 8, type 6+8 in the console

Syntax

```
6+8     # in console Output will be: 14
```

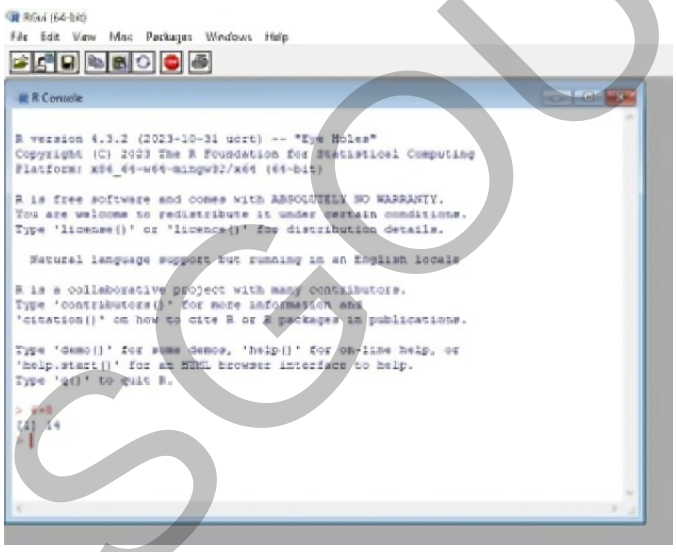


Fig. 4.1.4 Sample screen shot

See the e-content for a color version of this image

4.1.4.4 Comments in R

Comments can be used to explain R code, and to make it more readable. It can also be used to prevent execution when testing alternative code. Comments starts with a #. When executing code, R will ignore anything that starts with #.

Syntax for single line comment:

```
# This is a comment

"Hello world"

Or

"Hello world" # This is a comment
```

Syntax for single multiline comment:

```
# This is a comment
# written in
# more than just one line
"Hello World!"
```

4.1.4.5 Creating Variables in R

We utilize clock rooms to securely store our bags. Each storage unit is assigned a unique name or number, which we use to identify and locate our bag within the facility. In programming languages, this identifying number or name is referred to as a variable. Different variables are stored in different memory locations.

Variables are named memory locations. Variables are used to store values in memory. R does not have a command for declaring a variable. A variable is created the moment you first assign a value to it. To assign a value to a variable, use the <- sign. To output (or print) the variable value, just type the variable name. In other programming languages, it is common to use = as an assignment operator. In R, we can use both = and <- as assignment operators. However, <- is preferred in most cases because the = operator can be forbidden in some context in R.

Syntax

```
name <- "John"
age <- 40
name # output "John"
age  # output 40
```

From the example above, name and age are variables, while "John" and 40 are values.

Rules for naming a Variable

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume)

- A variable name must start with a letter and can be a combination of letters, digits, period(.) and underscore(_). If it starts with period(.), it cannot be followed by a digit.

  Example: name_1, a.b, .dir

- A variable name cannot start with a number or underscore (_)
- Variable names are case-sensitive

  Example: age, Age and AGE are three different variables

- Reserved words cannot be used as variables

  TRUE, FALSE, NULL, if, else, print, while, repeat, for, return, break,

- Remember that variable names are case-sensitive!

R Multiple Variables

R allows you to assign the same value to multiple variables in one line:

Example

```
# Assign the same value to multiple variables in one line
var1 <- var2 <- var3 <- "Orange"

# Print variable values
var1
var2
var3
```

Output:

```
Orange
Orange
Orange
```

4.1.4.6 R Concatenate Elements

We can concatenate or join two or more elements. The paste() function is used for joining two elements. To combine both text and a variable, R uses comma (,)

Example

```
text <- "Morning"
paste("Good", text)
```

Output

```
Good Morning
```

We also use, to add a variable to another variable:

Example

```
text1 <- "Goods"
text2 <- "Morning"
paste(text1, text2)
```

Output

```
Good Morning
```

4.1.4.7 R Data Types

In the student registration process, we are using different data such as name, date of birth, address, family monthly income, etc. Different types of data are used in the registration process. Date of birth is date type, monthly income is numeric data, a name is a group of letters.

In programming, data type is an important concept. Variables can store data of different types, and different types can do different things. In R, variables do not need to be declared with any particular type, and can even change type after they have been set. R has a variety of data types and object classes. We can use the class() function to check the data type of a variable.

Basic data types in R can be divided into the following types:

- **numeric** - Used to store numeric values, including integers and decimals. Numeric values are typically represented as double-precision floating-point numbers.

  Example: 10.5, 55, 787

  x <- 10.5 # numeric

  class(x)

Output

  "Numeric"

- **integer** - A special type of numeric data that represents whole numbers. In R, integers are denoted by adding the L suffix to the number.

  Example: 1L, 55L, 100L, where the letter "L" declares this as an integer

  x <- 1000L# integer

  class(x)

Output

  "Integer"

- **complex** - (9 + 3i, where "i" is the imaginary part)

  x <- 9i + 3 # complex

  class(x)

Output

  "complex"

- **character** - Used to store complex numbers with real and imaginary parts. Complex numbers are represented using the i suffix.

"k", "R is exciting", "FALSE", "11.5"

x <- "R is exciting" # character/string

class(x)

Output

"Character"

- **logical** - Used to store Boolean values, which can be either TRUE or FALSE. These are often used for conditions and logical operations.

  x <- TRUE# logical/boolean

  class(x)

Output

"Logical"

4.1.4.8 Type Conversion

Converting a variable from one type into another is called type conversion. You can convert from one type to another with the following functions:

1. **Character to Numeric:** To convert character data to numeric data, you can use the as.numeric() function.

   character_value <- "123"

   numeric_value <- as.numeric(character_value)

2. **Numeric to Character:** To convert numeric data to character data, you can use the as.character() function.

   numeric_value <- 123

   character_value <- as.character(numeric_value)

3. **Logical to Numeric:** Logical values TRUE and FALSE can be converted to numeric values 1 and 0 respectively using the as.numeric() function.

   logical_value <- TRUE

   numeric_value <- as.numeric(logical_value)

4. **Numeric to Logical:** Numeric values can be converted to logical values. Any non-zero numeric value becomes TRUE, while zero becomes FALSE.

   numeric_value <- 5

   logical_value <- as.logical(numeric_value)

5. **Factor to Character/Numeric:** Factors can be converted to character or numeric values using as.character() or as.numeric() functions respectively.

```
factor_value <- factor(c("A", "B", "C"))
character_value <- as.character(factor_value)
numeric_value <- as.numeric(factor_value)
```

6. **Date and Time Conversions:** Date and time data can be converted between different formats using functions like as.Date(), as.POSIXct(), and as.POSIXlt().

```
date_string <- "2024-02-19"
date_value <- as.Date(date_string)
posix_string <- "2024-02-19 12:00:00"
posix_value <- as.POSIXct(posix_string)
```

4.1.4.9 Built-in Math Functions

In R, there are many built-in mathematical functions that you can use for various numerical computations. Here are some commonly used mathematical functions in R:

1. Trigonometric Functions:

   sin(): Sine

   cos(): Cosine

   tan(): Tangent

   asin(): Arcsine (inverse sine)

   acos(): Arccosine (inverse cosine)

   atan(): Arctangent (inverse tangent)

For example, the sin() function is used to compute the sine of an angle, specified in radians. The sine function returns the sine of the input angle, which is a value between -1 and 1.

```
angle <- pi/4 # Angle of 45 degrees (pi/4 radians)
sin_value <- sin(angle)
print(sin_value)
```

Output

```
0.7071068
```

2. Exponential and Logarithmic Functions:

   exp(): Exponential function

   log(): Natural logarithm (base e)

log10(): Common logarithm (base 10)

log2(): Binary logarithm (base 2)

For example, the exp() function is used to compute the exponential function, which raises the mathematical constant e to the power of a given value. The constant e is approximately equal to 2.718281.

```
value <- 2
exp_value <-
exp(value)
print(exp_value)
```

Output

```
7.389056
```

3. Statistical Functions:

   mean(): Mean (average) of a numeric vector

   median(): Median (middle value) of a numeric vector

   sd(): Standard deviation of a numeric vector

   var(): Variance of a numeric vector

   min(): Minimum value in a numeric vector

   max(): Maximum value in a numeric vector

For example, the min() and max() functions can be used to find the lowest or highest number in a set:

```
max(5, 10, 15)
```

Output

```
15
```

```
min(5, 10, 15)
```

Output

```
5
```

4. Other Common Functions:

   abs(): Absolute value

   sqrt(): Square root

   round(): Round to the nearest integer

   ceiling(): Round up to the nearest integer

   floor(): Round down to the nearest integer

sign(): Sign function (returns -1 for negative values, 0 for zero, and 1 for positive values)

For example, the sqrt() function is used to compute the square root of a given value. It returns the non-negative square root of the input value.

value <- 16

sqrt_value <- sqrt(value) print(sqrt_value)

Output

4

4.1.4.10 Random Number Generation

runif(): Generate random numbers from a uniform distribution

rnorm(): Generate random numbers from a normal distribution

sample(): Randomly sample values from a vector or sequence

For example, the sample() function is used to generate random samples from a specified set of values. It can be used to randomly select elements from a vector or to shuffle the elements of a vector.

sample(x, size, replace = FALSE, prob = NULL)

- x: The vector from which to take the samples.
- size: The number of samples to take.
- replace: A logical value indicating whether sampling should be done with replacement.

  If replace = TRUE, elements in x can be selected more than once.
- prob: An optional probability vector giving the probability of selecting each element inx. If not specified, all elements have an equal chance of being selected.

Randomly select elements from a vector:

values <- c("A", "B", "C", "D", "E")

Sample three values from the vector without replacement

sample_values <- sample(values, size = 3)

Print the sampled values

print(sample_values)

4.1.4.11 R String Literals

Strings are used for storing text. A string is surrounded by either single quotation marks, or double quotation marks:

"hello" is the same as 'hello':

Assign a String to a Variable

Assigning a string to a variable is done with the variable followed by the <- operator and the string:

Example

```
str <- "Hello"
str # print the value of str
```

Multiline Strings

You can assign a multiline string to a variable like this:

Example

```
str <- "R draws heavily from the S language,
originally developed during the 1960s and 1970s
by researchers at Bell Laboratories in New Jersey.
R developed by Ross Ihaka and Robert Gentleman
from the University of Auckland in New Zealand,
released in the early 1990s under the GNU public licence,
functions as a platform for statistical analyses and visualisations."
str # print the value of str
```

Output

R draws heavily from the S language, originally developed during the 1960s and 1970s by researchers at Bell Laboratories in New Jersey. R developed by Ross Ihaka and Robert Gentleman from the University of Auckland in New Zealand, released in the early 1990s under the GNU public licence, functions as a platform for statistical analyses and visualisations.

However, note that R will add a "\n" at the end of each line break. This is called an escape character, and the n character indicates a new line.

If you want the line breaks to be inserted at the same position as in the code, use the cat() function:

Example

```
str <- "R draws heavily from the S language,
originally developed during the 1960s and 1970s
by researchers at Bell Laboratories in New Jersey.
R developed by Ross Ihaka and Robert Gentleman
```

```
from the University of Auckland in New Zealand,
released in the early 1990s under the GNU public licence,
functions as a platform for statistical analyses and visualisations."
cat(str)
```

Output

```
R draws heavily from the S language,
originally developed during the 1960s and 1970s
by researchers at Bell Laboratories in New Jersey.
R developed by Ross Ihaka and Robert Gentleman
from the University of Auckland in New Zealand,
released in the early 1990s under the GNU public licence,
functions as a platform for statistical analyses and visualizations.>
```

Escape Characters

To insert characters that are illegal in a string, you must use an escape character. An escape character is a backslash \ followed by the character you want to insert.

An example of an illegal character is a double quote inside a string that is surrounded by double quotes:

Example

```
str <- "We are the so-called "Vikings", from the north."
str
```

Result:

Error: unexpected symbol in "str <- "We are the so-called "Vikings"

To fix this problem, use the escape character \":

Example

The escape character allows you to use double quotes when you normally would not be allowed:

```
str <- "We are the so-called \"Vikings\", from the north."
str
cat(str)
```

4.1.4.12 R Booleans (Logical Values)

In programming, you often need to know if an expression is true or false. You can evaluate any expression in R, and get one of two answers, TRUE or FALSE. When you compare two values, the expression is evaluated and R returns the logical answer:

Example

```
10 > 9        # TRUE because 10 is greater than 9
10 == 9  # FALSE because 10 is not equal to 9
10 < 9        # FALSE because 10 is greater than 9
```

You can also compare two variables:

Example

```
a <- 10
b <- 9
a > b
```

You can also run a condition in an if statement, which you will learn much more about in the if..else chapter.

Example

```
a <- 200
b <- 33
if (b > a) {
print ("b is greater than a")
} else {
print("b is not greater than a")
}
```

4.1.4.13 R Operators

When creating an ATM program setup, we utilise the addition operation (+) when customers make deposits and the subtraction operation (-) when they withdraw funds. These operations involve the use of operators. Operators are used to perform operations on values and variables. These are standard symbols used for the purpose of logical and arithmetic operations.

> Variables on which operations are performed are called operands, and the operator applied to these operands is referred to as the opcode.

Here addition of 3+2, 3 and 2 are operands and '+' is opcode.

R divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Miscellaneous operators

1. Arithmetic operators

Arithmetic operators are used with numeric values to perform common mathematical operations:

Table 4.1.1 Arithmetic operators

Operator	Syntax	Description
+	x+y	Addition: adds two numbers
-	x-y	Subtraction : subtracts two numbers
*	x*y	Multiplication : Multiplies two numbers. It is also used to replicate strings.
/	x/y	Division (float): divides the first operand by the second.
^	x^y	Exponent: represents "x" raised to the power of "y." This means that "x" is multiplied by itself "y" times.
%%	x%y	Modulus: returns the remainder when the first operand is divided by the second.
%/%	x%/%y	Integer division : divides the first operand by the second and returns the integer value.

2. Assignment operators

Assignment operators are used to assign values to variables. <<- is a global assigner.

Example

```
my_var <- 3
my_var <<- 3
```

It is also possible to turn the direction of the assignment operator.

```
3 -> my_var
3 ->> my_var
my_var # print my_var
```

3. Comparison Operators

Comparison operators are used to compare two values and return the result as boolean value

'True' or 'False'. It is also called relational operators.

Table 4.1.2 Comparison operators

Operator	Syntax	Description
==	x==y	Equal to, it return 'True' if x equal to y, otherwise 'False'
<	x<y	Less than, it return 'True' if x less than y, otherwise 'False'
<=	x<=y	Less than or equal to, it return 'True' if x less than or equal to y, otherwise 'False'
>	x>y	Greater than, it return 'True' if x greater than y, otherwise 'False'
>=	x>=y	Greater than, it return 'True' if x greater than or equal to y, otherwise 'False'
!=	x!=y	Not equal to, it return 'True' if x not equal to y, otherwise 'False'

4. Logical Operators

Logical operators are used to combine conditional statements and return boolean value

'True' if both statements are true otherwise false.

Table 4.1.3 Logical operators

Operator	Syntax	Description
&	x>1 &x<10	Element-wise Logical AND operator. It returns TRUE if both elements are TRUE, that is x>1 and x<10, otherwise return 'False'
&&	x>1 && x<10	Return true if both statements are 'True', that is x>1 and x<10, otherwise return 'False'
|	x<10 | x>5	Elementwise- Logical OR operator. It returns TRUE if one of the statement is TRUE, that is x<10 or x>5 otherwise return false
||	x<10 || x>5	Return true if any one of the condition is true that is x<10 or x>5 otherwise return false
!	!(x<10 and x>5) !(x<10 or x>5)	Reverse the result, returns False if the result is true

5. Miscellaneous operators

Miscellaneous operators are used to manipulate data:

Table 4.1.4 Miscellaneous operators

Operato r	Syntax	Description	Example
:	x:y	Creates a series of numbers in a sequence	x <- 1:10 > x [1] 1 2 3 4 5 6 7 8 9 10
%in%	x%in%y	Find out if an element belongs to a vector	x %in% 3 [1] FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
%*%	x%*%y	Matrix Multiplication	x <- Matrix1 %*% Matrix2

4.1.5 Control structures

Control structures define the flow of a program's execution. Based on the results of relational or logical operations, the program's flow may deviate. These structures enable you to make decisions, repeat actions, and control the order in which code blocks are executed. Various control structures exist, including:

A. Conditional statements

B. Loop statements

C. Loop control statements

A. Conditional statements (Decision making statements)

Conditional statements enable a program to evaluate expressions and execute code only if certain conditions are true or false. Decision-making is a particular situation in a programming language in which we need to make some decisions. Based on these decisions we will execute the next block of code. It decides the direction of the flow of program execution. R uses curly brackets { } to define the scope in the code.

In R, there are four types of decision-making statements.

- if statements
- else if statements
- if else statements
- nested if statements

1. if Statement

An "if statement" is written with the if keyword, and it is used to specify a block of code to be executed if a condition is TRUE:

Example

```
a <- 33
b <- 200
if (b > a) {
print("b is greater than a")
}
```

In this example we use two variables, a and b, which are used as a part of the if statement to test whether b is greater than a. As a is 33, and b is 200, we know that 200 is greater than 33, and so we print to screen that "b is greater than a".

2. else if

The else if keyword is R's way of saying "if the previous conditions were not true, then try this condition". You can use as many else if statements as you want in R.

Example

```
a <- 33
b <- 33
if (b > a) {
print("b is greater than a")
}
else if (a == b) {
print ("a and b are equal")
}
```

In this example a is equal to b, so the first condition is not true,

but the else if condition is true, so we print to screen that "a and b are equal".

3. if else

The else keyword catches anything which isn't caught by the preceding conditions.

Example

```
a <- 200
b <- 33
if (b > a) {
print("b is greater than a")
} else if (a == b) {
```

```
print("a and b are equal")
} else {
print("a is greater than b")
}
```

In this example, a is greater than b, so the first condition is not true, also the else if condition is not true, so we go to the else condition and print to screen that "a is greater than b".

You can also use else without else if:

Example

```
a <- 200
b <- 33
if (b > a) {
print("b is greater than a")
} else {
print("b is not greater than a")
}
```

4. Nested If Statements

You can also have if statements inside if statements, this is called nested if statements.

Example

```
x <- 41
if (x > 10) {
print("Above ten")
if (x > 20) {
print("and also above 20!")
} else {
print("but not above 20.")
}
} else {
 print("below 10.")
}
```

4.1.6 R - AND OR Operators

AND

The & symbol (and) is a logical operator, and is used to combine conditional statements:

Example

Test if a is greater than b, AND if c is greater than a:

```
a <- 200
b <- 33
c <- 500
if (a > b & c > a) {
print("Both conditions are true")
}
```

OR

The | symbol (or) is a logical operator, and is used to combine conditional statements:

Example

Test if a is greater than b, or if c is greater than a:

```
a <- 200
b <- 33
c <- 500
if (a > b | a > c) {
print("At least one of the conditions is true")
}
```

4.1.7 R Loops

I would like to create a beaded chain with a consistent colour and pattern. To begin, I gather beads of the same colour, ensuring I have enough for the desired length; in this case, I'll need 50 beads. Next, I select a thread to string the beads onto. Starting with one bead, I thread it onto the string, followed by the second, and so on. This process is repeated 50 times until all beads are strung onto the thread. Finally, I tie a secure knot at each end to prevent the beads from slipping out.

Initially collect 50 beads

Then select thread

Repeat the following steps in 50 times:

Take one bead

Thread it onto the string

After that, put a tie on both ends.

> A loop is used to execute a statement or a group of statements several times. It iterates over a set of code a specified number of times.

Loops can execute a block of code as long as a specified condition is reached. Loops are handy because they save time, reduce errors, and they make code more readable.

We use a counter to know how many times the process is executed. The value of this counter decides whether to continue the execution or not. Since loops work on the basis of such conditions, a variable like the counter will be used to construct a loop. This variable is generally known as loop control variable because it actually controls the execution of the loop. Every loop has for parts:

1. Initialisation: Before entering a loop, its control variable must be initialized. During initialisation, the loop control variable gets its first value. The initialisation statement is executed only once, at the beginning of the loop.

2. Test expression: It is a relational or logical expression whose value is either True or False. It decides whether the loop-body will be executed or not. If the test expression evaluates to True, the loop-body gets executed, otherwise it will not be executed.

3. Update statement: The update statement modifies the loop control variable by changing its value. The update statement is executed before the next iteration.

4. Body of the loop: The statements that need to be executed repeatedly constitute the body of the loop. It may be a simple statement or a compound statement.

R has two loop commands:

- while loops
- for loops

5. R While Loops

With the while loop execute a set of statements as long as a condition is TRUE:

Example

Print i as long as i is less than 6:

```
i <- 1
while (i< 6)
 { print(i)
```

```
i <- i + 1
}
```

In the example above, the loop will continue to produce numbers ranging from 1 to 5. The loop will stop at 6 because 6 < 6 is FALSE. The while loop requires relevant variables to be ready, in this example we need to define an indexing variable, i, which we set to 1. Remember to increment i, or else the loop will continue forever.

1. R For Loop

A for loop is used for iterating over a sequence:

Example

```
for (x in 1:10) { print(x)
fruits <- list("apple", "banana", "cherry")
for (x in fruits) {
print(x)
}
```

Example

Print the number of dices:

```
dice <- c(1, 2, 3, 4, 5, 6)
for (x in dice) {
print(x)
}
```

2. If .. Else Combined with a While Loop

To demonstrate a practical example, let us say we play a game of Yahtzee!

Example

Print "Yahtzee!" If the dice number is 6:

```
dice <- 1
while (dice <= 6) {
if (dice < 6) {
print("No Yahtzee")
}
print("Yahtzee!")
}
dice <- dice + 1
}
```

If the loop passes the values ranging from 1 to 5, it prints "No Yahtzee". Whenever it passes the value 6, it prints "Yahtzee!".

3. R Nested Loops

It is also possible to place a loop inside another loop. This is called a nested loop:

Example

Print the adjective of each fruit in a list:

```
adj <- list("red", "big", "tasty")
fruits <- list("apple", "banana", "cherry")
 for (x in adj) {
for (y in fruits) {
print(paste(x, y))
}
}
```

4.1.8 Loop control statements

In Python, loop control statements are used to change the normal flow of loop execution. These can be used if you wish to skip an iteration or stop the execution. There are three main loop control statements: break, continue, and pass.

1. break

The break statement is used to exit the loop prematurely. When the break statement is encountered inside a loop, the loop is terminated immediately, and the program continues with the next statement after the loop.

Example

Exit the loop if i is equal to 4.

```
i <- 1
while (i < 6) {
print(i)
i <- i + 1
if (i == 4) {
break
}
}
```

The loop will stop at 3 because we have chosen to finish the loop by using the break statement when i is equal to 4 (i == 4).

2. Next

With the next statement, we can skip an iteration without terminating the loop:

Example

Skip the value of 3:

```
i <- 0
while (i < 6) {
i <- i + 1
if (i == 3) {
next
}
print(i)
}
```

When the loop passes the value 3, it will skip it and continue to loop.

4.1.9 R Functions

Let us consider the case of a coffee making machine and discuss its functioning. Water, milk, sugar and coffee powder are supplied to the machine. The machine processes it according to a set of predefined instructions stored in it and returns the coffee which is collected in a cup. The instruction-set may be as follows:

1. Get 60 ml milk, 120 ml water, 5 gm coffee powder and 20 gm sugar from the storage of the machine.
2. Boil the mixture
3. Pass it to the outlet

Usually there will be a button in the machine to invoke this procedure. Let us name the button with the word coffeemaker. Symbolically we can represent the invocation as:

Cup = coffeemaker (Water, Milk, Sugar, Coffee Powder)

We can compare all these aspects with functions in programs. The word coffeemaker is the name of the function, water, milk, sugar and coffee powder are parameters for the function and coffee is the result returned. It is stored in a cup. Instead of a cup, we can use a glass, tumbler or any other container.

Function is a named unit of statements in a program to perform a specific task as part of the solution. A function is a block of statements that performs a particular task. The main idea behind the function is to put some commonly or repeatedly done tasks together and can use functions whenever we need to perform the same task multiple times without writing the same code again. As our program grows larger and larger, functions make it more organized and manageable. A function is a block of code which only runs when it is called. You can pass data, known as parameters, into a function. A function can return data as a result.

Creating a Function

To create a function, use the function() keyword:

Example

```
my_function <- function() { # create a function with the name my_function
print("Hello World!")
}
```

Call a Function

To call a function, use the function name followed by parenthesis, like my_function():

Example

```
my_function <- function() {
print("Hello World!")
}
my_function() # call the function named my_function
```

Arguments

Information can be passed into functions as arguments.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

The following example has a function with one argument (fname). When the function is called, we pass along a first name, which is used inside the function to print the full name:

Example

```
my_function <- function(fname) {
paste(fname, "Griffin")
}
my_function("Peter")
my_function("Lois")
```

my_function("Stewie")

Parameters or Arguments?

The terms "parameter" and "argument" can be used for the same thing: information that is passed into a function.

From a function's perspective:

A parameter is the variable listed inside the parentheses in the function definition. An argument is the value that is sent to the function when it is called.

4.1.10 R Nested Function

There are two ways to create a nested function:

Call a function within another function.

Write a function within a function.

Example

Call a function within another function:

```
Nested_function <- function(x, y) {
a <- x + y
return(a)
}
Nested_function(Nested_function(2,2), Nested_function(3,3))
```

Example:

The function tells x to add y.

The first input Nested_function(2,2) is "x" of the main function.

The second input Nested_function(3,3) is "y" of the main function.

The output is therefore (2+2) + (3+3) = 10.

Example

Write a function within a function:

```
Outer_func <- function(x) {
Inner_func <- function(y) {
a <- x + y
return(a)
}
return (Inner_func)
}
```

```
output <- Outer_func(3) # To call the Outer_func

output(5)
```

Example Explained

You cannot directly call the function because the Inner_func has been defined (nested) inside the Outer_func.

We need to call Outer_func first in order to call Inner_func as a second step. We need to create a new variable called output and give it a value, which is 3 here. We then print the output with the desired value of "y", which in this case is 5.

The output is therefore 8 (3 + 5).

4.1.11 R Function Recursion

R also accepts function recursion, which means a defined function can call itself.

Recursion is a common mathematical and programming concept. It means that a function calls itself. This has the benefit of meaning that you can loop through data to reach a result.

The developer should be very careful with recursion as it can be quite easy to slip into writing a function which never terminates, or one that uses excess amounts of memory or processor power. However, when written correctly, recursion can be a very efficient and mathematically- elegant approach to programming.

In this example, tri_recursion() is a function that we have defined to call itself ("recurse"). We use the k variable as the data, which decrements (-1) every time we recurse. The recursion ends when the condition is not greater than 0 (i.e. when it is 0).

To a new developer it can take some time to work out exactly how this works, the best way to find out is by testing and modifying it.

Example

```
tri_recursion <- function(k) {
if (k > 0) {
result <- k + tri_recursion(k - 1)
 print(result)
} else {
result = 0
return(result)
}
}

tri_recursion(6)
```

4.1.12 R Global Variables

Global Variables

Variables that are created outside of a function are known as global variables.

Global variables can be used by everyone, both inside of functions and outside.

Example

Create a variable outside of a function and use it inside the function:

```
txt <- "awesome"
my_function <- function() {
paste("R is", txt)
}
my_function()
```

If you create a variable with the same name inside a function, this variable will be local, and can only be used inside the function. The global variable with the same name will remain as it was, global and with the original value.

Example

Create a variable inside of a function with the same name as the global variable:

```
txt <- "global variable"
my_function <- function() {
txt = "fantastic"
paste("R is", txt)
}
my_function()
txt # print txt
```

If you try to print txt, it will return "global variable" because we are printing txt outside the function.

The Global Assignment Operator

Normally, when you create a variable inside a function, that variable is local, and can only be used inside that function.

To create a global variable inside a function, you can use the global assignment operator <<-

Example

If you use the assignment operator <<-, the variable belongs to the global scope:

```
my_function <- function() {
txt <<- "fantastic"
paste("R is", txt)
}
my_function()
print(txt)
```

Also, use the global assignment operator if you want to change a global variable inside a function:

Example

To change the value of a global variable inside a function, refer to the variable by using the global assignment operator <<-:

```
txt <- "awesome"
my_function <- function() {
txt <<- "fantastic"
paste("R is", txt)
}
my_function()
paste("R is", txt)
```

4.1.13 R Data structures

Have you stacked your food plate after eating? For our next lunch, we'll be taking the first plate from the top, which corresponds to the last one placed. This follows the principle of last in, first out for plate stacking. Likewise, computers are required to store, retrieve, and process large amounts of data, as they function primarily as data processing machines. Python offers well-defined data structures to efficiently handle data according to users' requirements.

In R, there are several built-in data structures that are commonly used to store and manipulate data.

1. R Vectors

A vector is simply a list of items that are of the same type. To combine the list of items to a vector, use the c() function and separate the items by a comma.

In the example below, we create a vector variable called fruits, that combine strings:

```
# Vector of strings
fruits <- c("banana", "apple", "orange")
# Print fruits
Fruits
```

Output

```
"banana" "apple" "orange"
```

In this example, we create a vector that combines numerical values:

```
# Vector of numerical values
numbers <- c(1, 2, 3)
# Print numbers
numbers
```

Output

```
1 2 3
```

To create a vector with numerical values in a sequence, use the : operator:

```
# Vector with numerical values in a sequence
numbers <- 1:10
numbers
```

Output

```
1 2 3 4 5 6 7 8 9 10
```

You can also create numerical values with decimals in a sequence, but note that if the last element does not belong to the sequence, it is not used:

```
# Vector with numerical decimals in a sequence
numbers 1 <- 1.5:6.5
 numbers1
```

Output

```
1.5 2.5 3.5 4.5 5.5 6.5
```

```
# Vector with numerical decimals in a sequence where the last element is not used
numbers 2 <- 1.5:6.3
 numbers2
```

Output

```
1.5 2.5 3.5 4.5 5.5 6.5
1.5 2.5 3.5 4.5 5.5
```

In the example below, we create a vector of logical values:

```
# Vector of logical values
log_values <- c(TRUE, FALSE, TRUE, FALSE)
 log_values
```

Output

```
TRUE FALSE TRUE FALSE
```

Vector Length

To find out how many items a vector has, use the length() function:

Example

```
fruits <- c("banana", "apple", "orange")
length(fruits)
```

Output

```
3
```

2. R Lists

A list in R can contain many different data types inside it. A list is a collection of data which is ordered and changeable.

To create a list, use the list() function:

Example

```
# List of strings
thislist <- list("apple", "banana", "cherry")
# Print the list
thislist
```

Output

```
[[1]]
[1] "apple"
[[2]]
[1] "banana"
[[3]]
[1] "cherry"
```

Access Lists

You can access the list items by referring to its index number, inside brackets. The first item has index 1, the second item has index 2, and so on:

Example

```
thislist <- list("apple", "banana", "cherry")
thislist[1]
```

Output

```
[[1]]
[1] "apple"
```

Change Item Value

To change the value of a specific item, refer to the index number:

Example

```
thislist <- list("apple", "banana", "cherry")
thislist[1] <- "blackcurrant"
# Print the updated list
thislist
```

Output

```
[[1]]
[1] "blackcurrant"
[[2]]
[1] "banana"
[[3]]
[1] "cherry"
```

List Length

To find out how many items a list has, use the length() function:

Example

```
thislist <- list("apple", "banana", "cherry")
length(thislist)
```

Output

```
3
```

Check if Item Exists

To find out if a specified item is present in a list, use the %in% operator:

Example

Check if "apple" is present in the list:

```
thislist <- list("apple", "banana", "cherry")
"apple" %in% thislist
```

Output

```
TRUE
```

Add List Items

To add an item to the end of the list, use the append() function:

Example

Add "orange" to the list:

```
thislist <- list("apple", "banana", "cherry")
append(thislist, "orange")
```

Output

```
[[1]]
[1] "apple"
[[2]]
[1] "banana"
[[3]]
[1] "cherry"
[[4]]
[1] "orange"
```

To add an item to the right of a specified index, add "after=index number" in the append() function:

Example

Add "orange" to the list after "banana" (index 2):

```
thislist <- list("apple", "banana", "cherry")
append(thislist, "orange", after = 2)
```

Output

[[1]]

[1] "apple"

[[2]]

[1] "banana"

[[3]]

[1] "orange"

[[4]]

[1] "cherry"

Remove List Items

You can also remove list items. The following example creates a new, updated list without an "apple" item:

Example

Remove "apple" from the list:

```
thislist <- list("apple", "banana", "cherry")
newlist <- thislist[-1]
# Print the new list
newlist
```

Output

[[1]]

[1] "banana"

[[2]]

[1] "cherry"

Range of Indexes

You can specify a range of indexes by specifying where to start and where to end the range, by using the : operator:

Example

Return the second, third, fourth and fifth item:

```
thislist <- list("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
(thislist)[2:5]
```

Output

```
[[1]]
[1] "banana"
[[2]]
[1] "cherry"
[[3]]
[1] "orange"
[[4]]
[1] "kiwi"
```

Note: The search will start at index 2 (included) and end at index 5 (included). Remember that the first item has index 1.

Loop Through a List

You can loop through the list items by using a for loop:

Example

Print all items in the list, one by one:

```
thislist <- list("apple", "banana", "cherry")
for (x in thislist) {
print(x)
}
```

Output

```
[1] "apple"
[1] "banana"
[1] "cherry"
```

Join Two Lists

There are several ways to join, or concatenate, two or more lists in R. The most common way is to use the c() function, which combines two elements together:

Example

```
list1 <- list("a", "b", "c")
list2 <- list(1,2,3)
list3 <- c(list1,list2)
list3
```

Output

```
[[1]]
[1] "a"
[[2]]
[1] "b"
[[3]]
[1] "c"
[[4]]
[1] 1
[[5]]
[1] 2
[[6]]
[1] 3
```

3. R Matrices

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 amount of rows and columns. Remember the c() function is used to concatenate items together.

Example

```
# Create a matrix
thismatrix <- matrix(c(1,2,3,4,5,6), nrow = 3, ncol = 2)
# Print the matrix
thismatrix
```

Output

```
[,1]  [,2]
[1,]1        4
[2,]2        5
[3,]3        6
```

You can also create a matrix with strings:

Example

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange"), nrow = 2, ncol = 2)
thismatrix
```

Output

[,1][,2]

[1,] "apple" "cherry"

[2,] "banana" "orange"

Access Matrix Items

You can access the items by using [ ] brackets. The first number "1" in the bracket specifies the row-position, while the second number "2" specifies the column-position:

Example

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

Output

[1] "cherry"

The whole row can be accessed if you specify a comma after the number in the bracket:

Example

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

Output

[1] "banana" "orange"

The whole column can be accessed if you specify a comma before the number in the bracket:

Example

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

Output

[1] "cherry" "orange"

Access More Than One Row

More than one row can be accessed if you use the c() function:

Example

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange","grape", "pineapple",
"pear", "melon", "fig"), nrow = 3, ncol = 3)

thismatrix[c(1,2),]
```

Output

```
[,1]  [,2]    [,3]
[1,] "apple" "orange" "pear"
[2,] "banana" "grape" "melon"
```

Access More Than One Column

More than one column can be accessed if you use the c() function:

Example

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange","grape", "pineapple",
"pear", "melon", "fig"), nrow = 3, ncol = 3)

thismatrix[, c(1,2)]
```

Output

```
[,1][,2]

[1,] "apple" "orange"

[2,] "banana" "grape"

[3,] "cherry" "pineapple"
```

Add Rows and Columns

Use the cbind() function to add additional columns in a Matrix. The cells in the new column must be of the same length as the existing matrix.

Example

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange","grape", "pineapple",
"pear", "melon", "fig"), nrow = 3, ncol = 3)

newmatrix <- cbind(thismatrix, c("strawberry", "blueberry", "raspberry"))

# Print the new matrix

newmatrix
```

Output

```
[,1][,2]     [,3]     [,4]

[1,] "apple" "orange" "pear" "strawberry" [2,]

"banana" "grape" "melon" "blueberry" [3,]

"cherry" "pineapple" "fig" "raspberry"
```

Use the rbind() function to add additional rows in a Matrix. The cells in the new row must be of the same length as the existing matrix.

Example

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange","grape", "pineapple",
"pear", "melon", "fig"), nrow = 3, ncol = 3)

newmatrix <- rbind(thismatrix, c("strawberry", "blueberry", "raspberry"))

# Print the new matrix

newmatrix
```

Output

```
[,1]         [,2]       [,3]
[1,] "apple"   "orange"    "pear"
[2,] "banana"  "grape"     "melon"
[3,] "cherry"  "pineapple" "fig"
[4,] "strawberry" "blueberry" "raspberry"
```

Remove Rows and Columns

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

Example

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange", "mango", "pineapple"),
nrow = 3, ncol =2)

#Remove the first row and the first column

thismatrix <- thismatrix[-c(1), -c(1)]

thismatrix
```

Output

```
[1] "mango"  "pineapple"
```

Check if an Item Exists

To find out if a specified item is present in a matrix, use the %in% operator:

Example

Check if "apple" is present in the matrix:

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange"), nrow = 2, ncol = 2)

"apple" %in% thismatrix
```

Output

```
[1] TRUE
```

Number of Rows and Columns

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

Example

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

Output

```
[1] 2 2
```

Matrix Length

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

Example

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange"), nrow = 2, ncol = 2)
length(thismatrix)
```

Output

```
[1] 4
```

Total cells in the matrix is the number of rows multiplied by number of columns.

In the example above: Dimension = 2*2 = 4

Loop Through a Matrix

You can loop through a Matrix using a for loop. The loop will start at the first row, moving right:

Example

Loop through the matrix items and print them:

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange"), nrow = 2, ncol = 2)
for (rows in 1:nrow(thismatrix)) {
for (columns in 1:ncol(thismatrix)) {
print(thismatrix[rows, columns])
}
}
```

Output

```
[1] "apple"
[1] "cherry"
[1] "banana"
[1] "orange"
```

Combine two Matrices

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

Example

```
# Combine matrices

Matrix1 <- matrix(c("apple", "banana", "cherry", "grape"), nrow = 2, ncol = 2)

Matrix2 <- matrix(c("orange", "mango", "pineapple", "watermelon"), nrow = 2, ncol = 2)

# Adding it as a rows

Matrix_Combined <- rbind(Matrix1, Matrix2)

Matrix_Combined
```

Output

```
     [,1]      [,2]
[1,] "apple" "cherry"
[2,] "banana" "grape"
[3,] "orange" "pineapple"
[4,] "mango" "watermelon"
```

```
# Adding it as a columns

Matrix_Combined <- cbind(Matrix1, Matrix2)

Matrix_Combined
```

```
     [,1]  [,2]  [,3]     [,4]
[1,] "apple" "cherry" "orange" "pineapple"
[2,] "banana" "grape" "mango" "watermelon"
```

4. R Arrays

Compared to matrices, arrays can have more than two dimensions. We can use the array() function to create an array, and the dim parameter to specify the dimensions:

Example

```
# An array with one dimension with values ranging from 1 to 24

thisarray <- c(1:24)

thisarray
```

Output

```
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
```

```
# An array with more than one dimension
multiarray <- array(thisarray, dim = c(4, 3, 2))
multiarray
```

Output

```
, , 1

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


, , 2


     [,1] [,2] [,3]
[1,]   13   17   21
[2,]   14   18   22
[3,]   15   19   23
[4,]   16   20   24
```

Example Explained

In the example above we create an array with the values 1 to 24.

How does dim=c(4,3,2) work?

The first and second number in the bracket specifies the amount of rows and columns. The last number in the bracket specifies how many dimensions we want.

Arrays can only have one data type.

Access Array Items

You can access the array elements by referring to the index position. You can use the [] brackets to access the desired elements from an array:

Example

```
thisarray <- c(1:24)
multiarray <- array(thisarray, dim = c(4, 3, 2))
multiarray[2, 3, 2]
```

Output

```
[1] 22
```

The syntax is as follow: array[row position, column position, matrix level]

You can also access the whole row or column from a matrix in an array, by using the c() function:

Example

```
thisarray <- c(1:24)
# Access all the items from the first row from matrix one
multiarray <- array(thisarray, dim = c(4, 3, 2))
multiarray[c(1),,1]
```

Output

```
[1] 1 5 9
```

```
# Access all the items from the first column from matrix one
multiarray <- array(thisarray, dim = c(4, 3, 2))
multiarray[,c(1),1]
```

Output

```
[1] 1 2 3 4
```

A comma (,) before c() means that we want to access the column.A comma (,) after c() means that we want to access the row.

Check if an Item Exists

To find out if a specified item is present in an array, use the %in% operator:

Example

Check if the value "2" is present in the array:

```
thisarray <- c(1:24)
multiarray <- array(thisarray, dim = c(4, 3, 2))
2 %in% multiarray
```

Output

```
[1] TRUE
```

Amount of Rows and Columns

Use the dim() function to find the amount of rows and columns in an array:

Example

```
thisarray <- c(1:24)
multiarray <- array(thisarray, dim = c(4, 3, 2))
dim(multiarray)
```

Output

[1] 4 3 2

Array Length

Use the length() function to find the dimension of an array:

Example

```
thisarray <- c(1:24)
multiarray <- array(thisarray, dim = c(4, 3, 2))
length(multiarray)
```

Output

[1] 24

Loop Through an Array

You can loop through the array items by using a for loop:

Example

```
thisarray <- c(1:24)
multiarray <- array(thisarray, dim = c(4, 3, 2))
for(x in multiarray){
print(x)
}
```

Output

[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
[1] 11
[1] 12
[1] 13
[1] 14
[1] 15
[1] 16

```
[1] 17
[1] 18
[1] 19
[1] 20
[1] 21
[1] 22
[1] 23
[1] 24
```

4.1.14 R Data Frames

Data Frames are data displayed in a format as a table. Data Frames can have different types of data inside it. While the first column can be character, the second and third can be numeric or logical. However, each column should have the same type of data.

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

Example

```
# 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
```

Output

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

Summarise the Data

Use the summary() function to summarise the data from a Data Frame:

Example

```
Data_Frame <- data.frame (
Training = c("Strength", "Stamina", "Other"),
Pulse = c(100, 150, 120),
Duration = c(60, 30, 45)
```

```
)
Data_Frame
summary(Data_Frame)
```

Output

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

You will learn more about the summary() function in the statistical part of the R tutorial.

Access Items

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

Example

```
Data_Frame <- data.frame (
Training = c("Strength", "Stamina", "Other"),
Pulse = c(100, 150, 120),
Duration = c(60, 30, 45)
)
Data_Frame[1]
Data_Frame[["Training"]]
Data_Frame$Training
```

Output

```
>    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:

Example

```
Data_Frame <- data.frame (
Training = c("Strength", "Stamina", "Other"),
Pulse = c(100, 150, 120),
Duration = c(60, 30, 45)
)
# Add a new row
New_row_DF <- rbind(Data_Frame, c("Strength", 110, 110))
# Print the new row
New_row_DF
```

Output

```
  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:

Example

```
Data_Frame <- data.frame (
Training = c("Strength", "Stamina", "Other"),
Pulse = c(100, 150, 120),
Duration = c(60, 30, 45)
)
# Add a new column
New_col_DF <- cbind(Data_Frame, Steps = c(1000, 6000, 2000))
```

```
# Print the new column
New_col_DF
```

Output

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

Remove Rows and Columns

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

Example

```
Data_Frame <- data.frame (
Training = c("Strength", "Stamina", "Other"),
Pulse = c(100, 150, 120),
Duration = c(60, 30, 45)
)
# Remove the first row and column
Data_Frame_New <- Data_Frame[-c(1), -c(1)]
# Print the new data frame
Data_Frame_New
```

Output

```
  Pulse Duration
2  150       30
3  120       45
```

Amount of Rows and Columns

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

Example

```
Data_Frame <- data.frame (
Training = c("Strength", "Stamina", "Other"),
Pulse = c(100, 150, 120),
Duration = c(60, 30, 45)
)
dim(Data_Frame)
```

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

Example

```
Data_Frame <- data.frame (
Training = c("Strength", "Stamina", "Other"),
Pulse = c(100, 150, 120),
Duration = c(60, 30, 45)
)
ncol(Data_Frame)
nrow(Data_Frame)
Data Frame Length
```

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

Example

```
Data_Frame <- data.frame (
Training = c("Strength", "Stamina", "Other"),
Pulse = c(100, 150, 120),
Duration = c(60, 30, 45)
)
length(Data_Frame)
Combining Data Frames
```

Use the rbind() function to combine two or more data frames in R vertically:

Example

```
Data_Frame1 <- data.frame (
Training = c("Strength", "Stamina", "Other"),
Pulse = c(100, 150, 120),
Duration = c(60, 30, 45)
)
Data_Frame2 <- data.frame (
Training = c("Stamina", "Stamina", "Strength"),
Pulse = c(140, 150, 160),
Duration = c(30, 30, 20)
)
New_Data_Frame <- rbind(Data_Frame1, Data_Frame2)
New_Data_Frame
```

And use the cbind() function to combine two or more data frames in R horizontally:

Example

```
Data_Frame3 <- data.frame (
Training = c("Strength", "Stamina", "Other"),
Pulse = c(100, 150, 120),
Duration = c(60, 30, 45)
)
Data_Frame4 <- data.frame (
Steps = c(3000, 6000, 2000),
Calories = c(300, 400, 300)
)
New_Data_Frame1 <- cbind(Data_Frame3, Data_Frame4)
New_Data_Frame1
```

4.1.15 R Factors

Factors are used to categorise data. Examples of factors are:

Demography: Male/Female

Music: Rock, Pop, Classic, Jazz

Training: Strength, Stamina

To create a factor, use the factor() function and add a vector as argument:

Example

```
# Create a factor
music_genre <- factor(c("Jazz", "Rock", "Classic", "Classic", "Pop", "Jazz", "Rock", "Jazz"))
# Print the factor
music_genre
```

Output

```
[1] Jazz Rock    Classic Classic Pop   Jazz    Rock    Jazz
```

Levels: Classic Jazz Pop Rock

You can see from the example above that the factor has four levels (categories): Classic, Jazz, Pop and Rock. To only print the levels, use the levels() function:

Example

```
music_genre <- factor(c("Jazz", "Rock", "Classic", "Classic", "Pop", "Jazz", "Rock", "Jazz"))
levels(music_genre)
```

Output

```
[1] "Classic" "Jazz"          "Pop"   "Rock"
```

You can also set the levels, by adding the levels argument inside the factor() function:

Example

```
music_genre <- factor(c("Jazz", "Rock", "Classic", "Classic", "Pop", "Jazz", "Rock", "Jazz"),
levels = c("Classic", "Jazz", "Pop", "Rock", "Other"))
levels(music_genre)
```

Output

```
[1] "Classic" "Jazz"          "Pop"   "Rock""Other"
```

Factor Length

Use the length() function to find out how many items there are in the factor:

Example

```
music_genre <- factor(c("Jazz", "Rock", "Classic", "Classic", "Pop", "Jazz", "Rock", "Jazz"))
length(music_genre)
```

Output

```
[1] 8
```

Access Factors

To access the items in a factor, refer to the index number, using [] brackets:

Example

Access the third item:

```
music_genre <- factor(c("Jazz", "Rock", "Classic", "Classic", "Pop", "Jazz", "Rock", "Jazz"))
music_genre[3]
```

Output

```
[1] Classic
```

Levels: Classic Jazz Pop Rock

Change Item Value

To change the value of a specific item, refer to the index number:

Example

Change the value of the third item:

```
music_genre <- factor(c("Jazz", "Rock", "Classic", "Classic", "Pop", "Jazz", "Rock", "Jazz"))

music_genre[3] <- "Pop"

music_genre[3]
```

Output

```
[1] Pop

Levels: Classic Jazz Pop Rock
```

Note that you cannot change the value of a specific item if it is not already specified in the factor. The following example will produce an error:

Example

Trying to change the value of the third item ("Classic") to an item that does not exist/ not predefined ("Opera"):

```
music_genre <- factor(c("Jazz", "Rock", "Classic", "Classic", "Pop", "Jazz", "Rock", "Jazz"))

music_genre[3] <- "Opera"

music_genre[3]
```

Output

```
Warning message:

In `[<-.factor`(`*tmp*`, 3, value = "Opera") :

invalid factor level, NA generated
```

However, if you have already specified it inside the levels argument, it will work:

Example

Change the value of the third item:

```
music_genre <- factor(c("Jazz", "Rock", "Classic", "Classic", "Pop", "Jazz", "Rock", "Jazz"),

levels = c("Classic", "Jazz", "Pop", "Rock", "Opera"))

music_genre[3] <- "Opera"

music_genre[3]
```

Output

[1] Opera

Levels: Classic Jazz Pop Rock Opera

4.1.16 Data Manipulation Using dplyr Package (Filtering, Grouping, Summarising)

The dplyr package in R provides a powerful set of tools for data manipulation, including filtering, grouping, and summarising data. When working with data you must:

Figure out what you want to do.

Describe those tasks in the form of a computer program.

Execute the program.

The dplyr package makes these steps fast and easy:

By constraining your options, it helps you think about your data manipulation challenges.

It provides simple "verbs", functions that correspond to the most common data manipulation tasks, to help you translate your thoughts into code.

It uses efficient backends, so you spend less time waiting for the computer.

This document introduces you to dplyr's basic set of tools, and shows you how to apply them to data frames. dplyr also supports databases via the dbplyr package, once you've installed, read vignette("dbplyr") to learn more.

Here's a brief overview of some of the key functions in the dplyr package:

- filter():

Used to filter rows based on specific conditions.

Example

```
library(dplyr)
filtered_data <- filter(data_frame, column_name > 10)
```

- arrange():

Used to reorder rows based on one or more variables.

Example

```
arranged_data <- arrange(data_frame, column_name)
```

- select():

Used to select columns of interest from a data frame.

Example

```
selected_data <- select(data_frame, column1, column2)
```

- mutate():

Used to create new columns or modify existing ones based on transformations.

Example

```
mutated_data <- mutate(data_frame, new_column = column1 * 2)
```

group_by(): Used to group data by one or more variables.

Example

```
grouped_data <- group_by(data_frame, column_name)
```

- summarize():

Used to compute summary statistics for groups of data.

Example

```
summarized_data <- summarize(grouped_data, mean_value = mean(column_name))
```

- join():

Used to combine multiple data frames based on common variables.

Example

```
merged_data <- left_join(data_frame1, data_frame2, by = "common_column")
```

4.1.17 Data Visualization in R (ggplot2)

Data visualisation with R and ggplot2 in R Programming Language also termed as Grammar of Graphics is a free, open-source, and easy-to-use visualisation package widely used in R Programming Language. It is the most powerful visualisation package written by Hadley Wickham. It includes several layers on which it is governed. The layers are as follows:

Building Blocks of layers with the grammar of graphics

- Data: The element is the data set itself
- Aesthetics: The data is to map onto the Aesthetics attributes such as x-axis, y-axis, colour, fill, size, labels, alpha, shape, line width, line type
- Geometrics: How our data being displayed using point, line, histogram, bar, boxplot
- Facets: It displays the subset of the data using Columns and rows

- Statistics: Binning, smoothing, descriptive, intermediate
- Coordinates: the space between data and display using Cartesian, fixed, polar, limits
- Themes: Non-data link

Loading the ggplot2 package:

Before using ggplot2, you need to install and load the package in your R session: install.packages("ggplot2")

library(ggplot2)

Creating a basic plot:

The basic syntax for creating a plot with ggplot2 involves specifying the data frame and mapping variables to aesthetics like x-axis, y-axis, colour, shape, etc. Here's an example of creating a scatter plot:

ggplot(data = my_data, aes(x = x_variable, y = y_variable)) +

geom_point()

Creating a basic plot:

The basic syntax for creating a plot with ggplot2 involves specifying the data frame and mapping variables to aesthetics like x-axis, y-axis, colour, shape, etc. Here's an example of creating a scatter plot:

ggplot(data = my_data, aes(x = x_variable, y = y_variable)) +

geom_point()

Customizing plots:

ggplot2 allows extensive customization of plots. You can modify aspects such as axis labels, plot titles, colours, shapes, themes, etc.

ggplot(data = my_data, aes(x = x_variable, y = y_variable, colour = category)) +

geom_point() +

labs(x = "X Axis Label", y = "Y Axis Label", title = "My Plot Title") +

theme_minimal()

Adding layers:

You can add different layers to your plot using geom_ functions. For instance, geom_point() for scatter plots, geom_line() for line plots, geom_bar() for bar plots, etc.

ggplot(data = my_data, aes(x = x_variable, y = y_variable)) +

geom_point() +

geom_smooth(method = "lm")

Faceting:

Faceting allows you to create small multiples, where subsets of data are displayed in separate panels. This is achieved using the facet_wrap() or facet_grid() functions.

ggplot(data = my_data, aes(x = x_variable, y = y_variable)) +

geom_point() +

facet_wrap(~category)

4.1.18 Qualitative and Quantitative Data in R

4.1.18.1 Qualitative data

Qualitative data is non-statistical and is typically unstructured or semi-structured. This data isn't necessarily measured using hard numbers you use to develop graphs and charts. Instead, it is categorised based on properties, attributes, labels, and other identifiers. Qualitative data can be used to ask the question, 'why'. It is investigative and asks open-ended questions to conduct the research. Generating this data from qualitative research is used for theorizations, interpretations, developing hypotheses, and initial understandings.

Qualitative data examples

Suppose we have a book, its qualitative features are paper quality, design cover page etc. When discussing qualitative data, we talk about a specific object's characteristics. Qualitative data is derived through qualitative analysis of detailed information about the matter. Thus, qualitative data identifiers can be subjective, making qualitative data analysis a complex process with numerous possibilities and structures.

4.1.18.2 Quantitative Data

Quantitative data is statistical and typically structured – meaning it is more rigid and defined. This data type is measured using numbers and values, making it a more suitable candidate for data analysis. Whereas qualitative is open for exploration, quantitative data is much more concise and close-ended. It can be used to ask 'how much' or 'how many,' followed by conclusive information. Quantitative data, also known as continuous data, consists of numeric data that support arithmetic operations. This is in contrast with qualitative data, whose values belong to predefined classes with no arithmetic operation allowed.

Quantitative data examples

From the above example of a book, quantitative data are size of book, number of pages, price of book etc.

Table 4.1.5 Comparative table of qualitative and quantitative data

Qualitative Data	Quantitative Data
1. Qualitative data uses methods like interviews, participant observation, and focus on a grouping to gain collective information.	1. Quantitative data uses methods as questionnaires, surveys, and structural observations to gain collective information.
2. Data format used in it is textual. Datasheets are contained of audio or video recordings and notes.	2. Data format used in it is numerical. Datasheets are obtained in the form of numerical values.
3. Qualitative data talks about the experience or quality and explains the questions like 'why' and 'how'.	3. Quantitative data talks about the quantity and explains the questions like 'how much', 'how many .
4. The data is analysed by grouping it into different categories.	4. The data is analysed by statistical methods.
5.Qualitative data are subjective and can be further open for interpretation.	5. Quantitative data are fixed and universal.

Recap

- R is a powerful open-source software used for statistical computing and data analysis. It provides a wide range of tools for data manipulation, visualization, and machine learning.
- The R environment allows users to create high-quality plots and reports for effective data presentation.
- Fundamentals of R
  - Print simple message
  - comments
  - Variables
  - Strings
  - Data types
  - Operators
  - Conditional statements
  - Loops
  - Functions
- Data structures in R
  - List
  - vectors
  - Frames

- Matix
- Arrays
- Factors

♦ Data Manipulation Using dplyr Package
- Filtering
- Grouping,
- Summarising

♦ ggplot2 is a popular R package used for creating elegant and complex data visualizations easily.

♦ With ggplot2, you can create various plots like bar charts, histograms, scatter plots, and line graphs. It also provides powerful options for customization, such as colors, themes, and labels. Overall, ggplot2 helps transform raw data into clear, informative, and visually appealing graphics.

♦ Qualitative data (also called categorical data) represent non-numeric information such as names, labels, or categories — for example, gender, color, or city.

♦ Quantitative data represent numeric values that can be measured or counted — for example, height, weight, or age. These are stored as numeric or integer types in R and can be used for statistical calculations and visualizations.

Objective Type Questions

1. In R execution code executes line by line, allowing for interactive exploration and debugging so it is called
2. Symbol used for single line comment
3. In R name of variable cannot starts with
4. Which object is used to find type of a variable
5. Which function rounds a number upwards to its nearest integer
6. Write output of ceiling(1.4) and floor(1.4)
7. The function is used to find the number of characters in a string is
8. Write output of the code

   i<-1

```
        while (i < 6) {
          print(i)
          i<-i+1
          if (i==4) {
            break
  }
}
```

9. To find out how many items a list has, use the function

10. Create a 3x2 matrices

Answers to Objective Type Questions

1. Interpretive language
2. #
3. A number or underscore ( )
4. class()
5. ceiling()
6. 2,1
7. nchar()
8. 1 2 3 5 6
9. length()
10. thismatrix <- matrix(c(1,2,3,4,5,6), nrow = 3, ncol = 2)
    thismatrix

Assignments

1. Write different steps to install R
2. Explain features of R
3. Explain different data types available in R
4. Explain different operators in R
5. What is data structure? Explain different data structures with examples.
6. Explain data manipulation using dplyr package
7. What are the differences between qualitative data and quantitative data

Reference

1. Dalgaard, P. (2008). *Introductory statistics with R* (2nd ed.). Springer.https://doi.org/10.1007/978-0-387-79054-1
2. Matloff, N. (2011). *The art of R programming: A tour of statistical software design.* No Starch Press.
3. Verzani, J. (2014). *Using R for introductory statistics* (2nd ed.). CRC Press.
4. Wickham, H., & Grolemund, G. (2017). *R for data science: Import, tidy, transform, visualize, and model data.* O'Reilly Media. https://r4ds.had.co.nz
5. Ismay, C., & Kim, A. Y. (2019). *ModernDive: An introduction to statistical and data sciences via R.* CRC Press.

Suggested Reading

1. Agarwal, B. L. (2013). *Basic statistics*. New Age International Publishers.
2. Bhat, B. R., Sri Venkata Ramana T, & Rao Madhava K. S. (1977). *Statistics: A Beginner's Text Vol. 2*. New Age International (P) Ltd., New Delhi.
3. Dekking, F. M., & others. (2005). *A Modern Introduction to Probability and Statistics*. Springer Verlag, New York.
4. Seema Acharya, Subhasini Chellappan (2015), *Big Data Analytics*, Wiley.
5. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). *An Introduction to Statistical Learning: with Applications in R*. Springer.

Familiarisation of Data Analytics tool – WEKA

Learning Outcomes

After completing this unit, the learner will be able to:

- define the basic features, interface, and functionalities of the WEKA data analytics tool
- explain the working principles of Decision Tree Classifier and Naïve Bayes Classifier in WEKA
- apply K-Means and Agglomerative Clustering algorithms to perform clustering using WEKA
- implement Linear Regression models to analyze relationships between data attributes
- compare the performance of different machine learning algorithms in WEKA based on accuracy and efficiency

Prerequisites

To gain the most from this unit, learners should already possess a fundamental understanding of data and basic statistical concepts. They are expected to know about different data types such as numerical, categorical, and textual data, and understand statistical terms like mean, median, mode, standard deviation, and correlation. For example, knowing how to find the average monthly income of employees or the correlation between temperature and electricity usage helps in understanding how relationships between variables are analyzed in WEKA.

It is also important for learners to have prior exposure to the basics of machine learning. They should be aware of the difference between supervised and unsupervised learning, as well as common algorithms such as Decision Trees, Naïve Bayes, and Linear Regression. For instance, understanding how a Decision Tree predicts whether a customer will buy a product or how Linear Regression estimates housing prices will help connect theory to practical WEKA applications in classification, clustering, and regression.

Additionally, learners should be comfortable using computer applications for data handling, such as Microsoft Excel, Google Sheets, or database tools. Experience in organizing and cleaning datasets like removing duplicate records or missing entries, will be useful when preparing data for WEKA. A basic familiarity with Java-based tools is also recommended, since WEKA is developed in Java and supports both a graphical interface and command-line operations for data analysis.

Keywords

Decision Tree Classifier, Naïve Bayes Classifier, K-Means Clustering, Agglomeration Clustering, Linear Regression

Discussion

4.2.1 Fundamentals of WEKA

WEKA (Waikato Environment for Knowledge Analysis) is an open-source software tool used for data analysis, machine learning, and data mining. Developed at the University of Waikato, New Zealand, it provides an easy-to-use graphical interface that allows users to build, test, and evaluate machine learning models. WEKA supports a wide variety of data mining tasks such as classification, clustering, regression, association rule mining, and visualization, making it a valuable tool for both beginners and researchers. Block diagram of WEKA shown in figure 4.2.1.

4.2.1.1 Purpose of WEKA

The main purpose of WEKA is to simplify the process of data analysis and model building. It helps users apply machine learning algorithms to datasets without requiring programming skills. WEKA is commonly used for:

- Exploring datasets and understanding data patterns.

- Building predictive models for decision-making.
- Evaluating and comparing machine learning algorithms.
- Teaching and research in data mining and artificial intelligence.

Example: A teacher can use WEKA to analyze students' past performance data and predict future academic results.

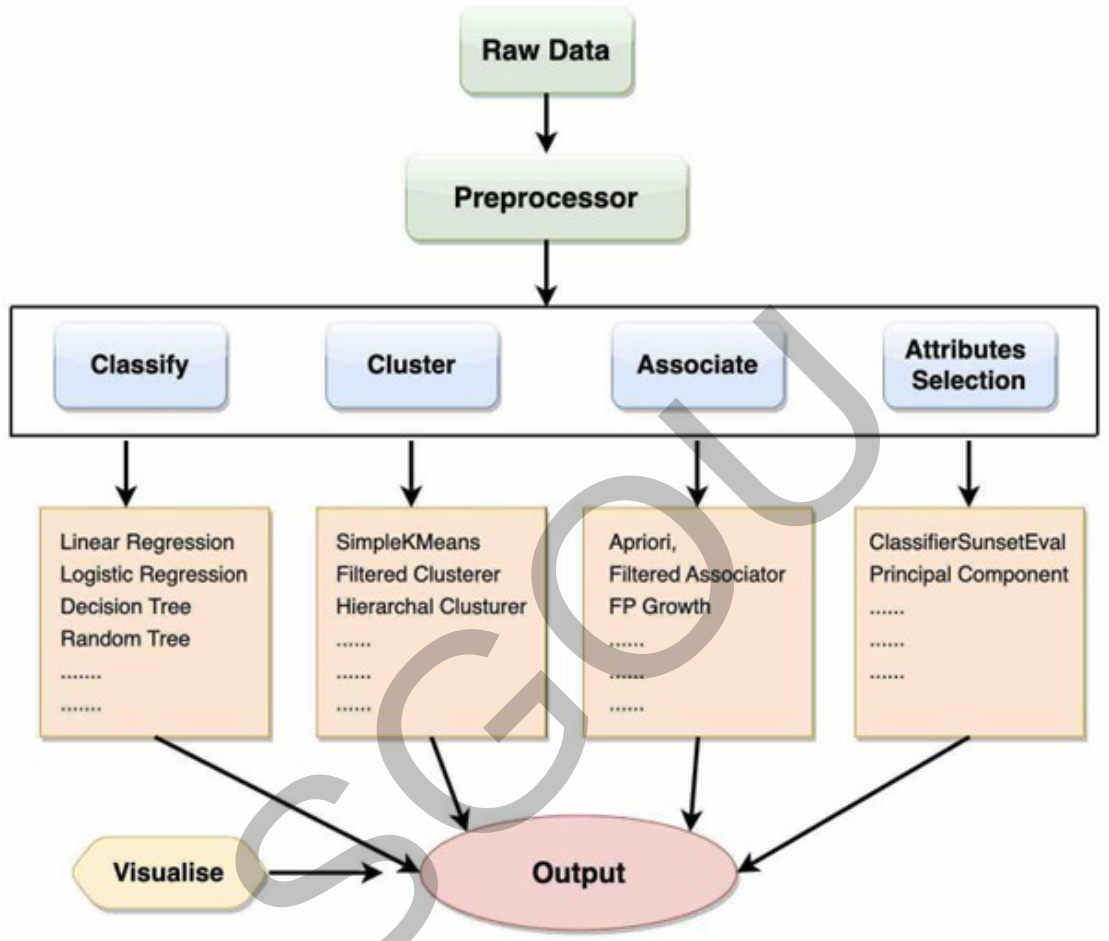


Fig. 4.2.1 Block diagram of WEKA

4.2.1.2 Features

Weka is widely recognized for its flexibility and user-friendly design, offering a range of features that make it a preferred tool for data scientists and researchers.

- **Graphical User Interface (GUI):** Weka's easy-to-navigate interface enables users to analyze datasets, apply machine learning techniques, and view results visually without requiring advanced programming skills.
- **Machine Learning Algorithms:** It includes an extensive library of algorithms for classification, regression, clustering, and association rule mining, along with options for feature selection and ensemble methods.

- **Data Preprocessing:** Weka provides multiple tools for preparing data, such as cleaning, normalization, and attribute selection, ensuring datasets are ready for analysis.
- **Scripting and Programming Support:** The platform offers a Java-based API for developers and allows integration with other programming languages like Python and R.
- **Visualization Capabilities:** Weka comes with various visualization features, including scatter plots, histograms, and decision tree views, to help users interpret data effectively.
- **Data Import and Export:** It supports several data file formats such as CSV, ARFF, and Excel, making it simple to import and export datasets.
- **Extensibility:** Being open source, Weka can be customized and extended to add new features or algorithms according to user needs.

4.2.1.3 Install Weka on Windows

WEKA is an open-source software developed in Java, making it compatible with any system that supports the Java platform. While the core tool is Java-based, can integrate WEKA's functionality with other popular languages using wrappers and external packages (Table 4.2.1).

Table 4.2.1 Popular languages using wrappers and external packages in WEKA

Language	Integration Method	Purpose
Java	Native API	The foundational language. You can call WEKA algorithms directly from your own Java code.
Python	python-weka-wrapper	A widely used wrapper that allows you to use WEKA's functionality from Python scripts.
R	RWeka Package	An interface that enables you to access WEKA algorithms and functions directly within the R environment.

WEKA can operate on various operating systems such as Windows, Linux, and macOS. It also offers a range of visualization tools that assist in data analysis, data cleaning, and predictive modeling. Follow the below steps to install Weka on Windows:

Step 1: To install WEKA on your computer, go to the official WEKA website and download the appropriate installation file.

Step 2: Weka is successfully installed on the system and an icon is created on the desktop (Figure 4.2.2).

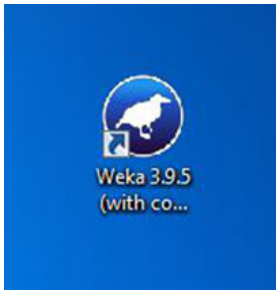


Fig. 4.2.2 WEKA icon is created on the desktop

Step 3: Run the software and see the interface (Figure 4.2.3).

Fig. 4.2.3 WEKA interface

WEKA provides several key interfaces to perform data analysis and machine learning:

- **Explorer:** A graphical environment for data preprocessing, classification, clustering, and visualization.
- **Experimenter:** Allows users to compare different algorithms and analyze performance results.
- **Knowledge Flow:** A visual workflow interface for designing and executing data mining processes.
- **Workbench:** The main interface that provides access to all the tools and features of the software.
- **Simple CLI (Command Line Interface):** For advanced users who prefer command-based operations.

4.2.1.4 Data Formats Supported

Weka supports a wide range of data file formats. Some of the supported formats as follows:

- arff
- arff.gz
- csv
- dat
- data
- json
- json.gz
- xrff
- xrff.gz

These formats provide flexibility in importing and exporting datasets for various machine learning tasks within Weka. The list of file types supported by Weka can be viewed in the drop-down menu located at the bottom of the interface, as illustrated in the figure 4.2.4 below.

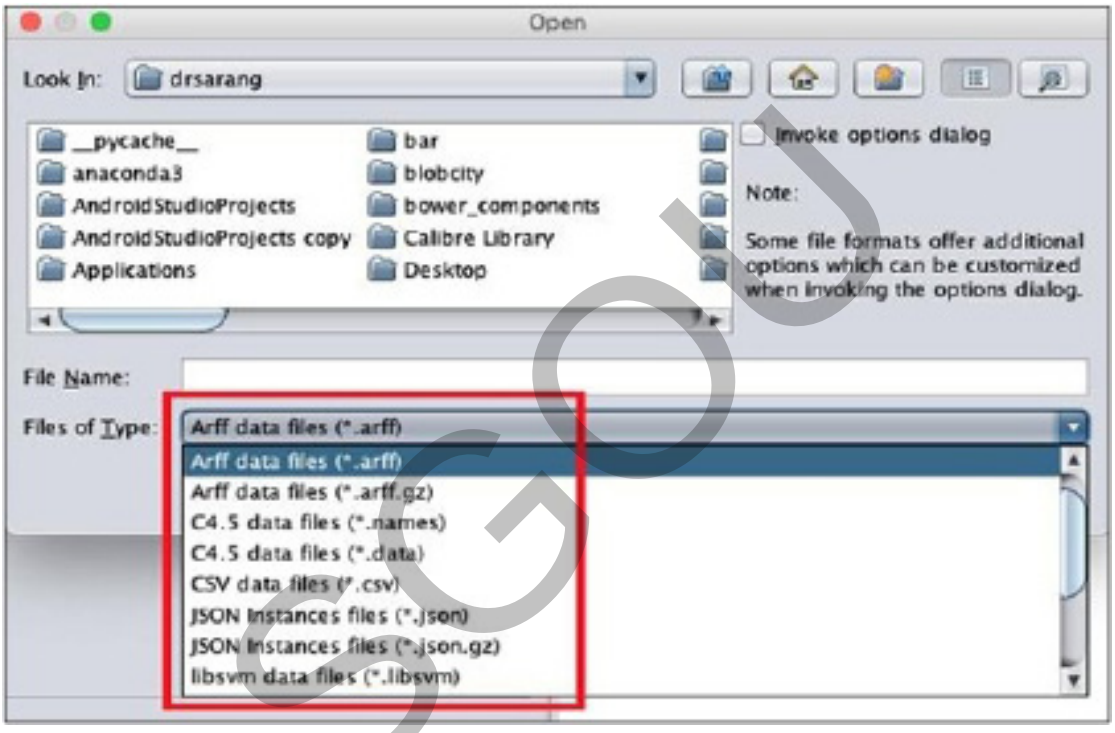


Fig.4.2.4 Supported Data Formats of WEKA

Weka mainly relies on the *Attribute-Relation File Format (ARFF),* a plain text format used to define data attributes and their corresponding values. An ARFF file is divided into two key sections: the *header*, which specifies attribute names, data types (such as numeric, nominal, string, or date), and their possible values; and the *data section*, which contains the actual dataset.

As an example for Arff format, the person data file named as *simple* loaded from the WEKA sample databases is shown in figure 4.2.5 below.

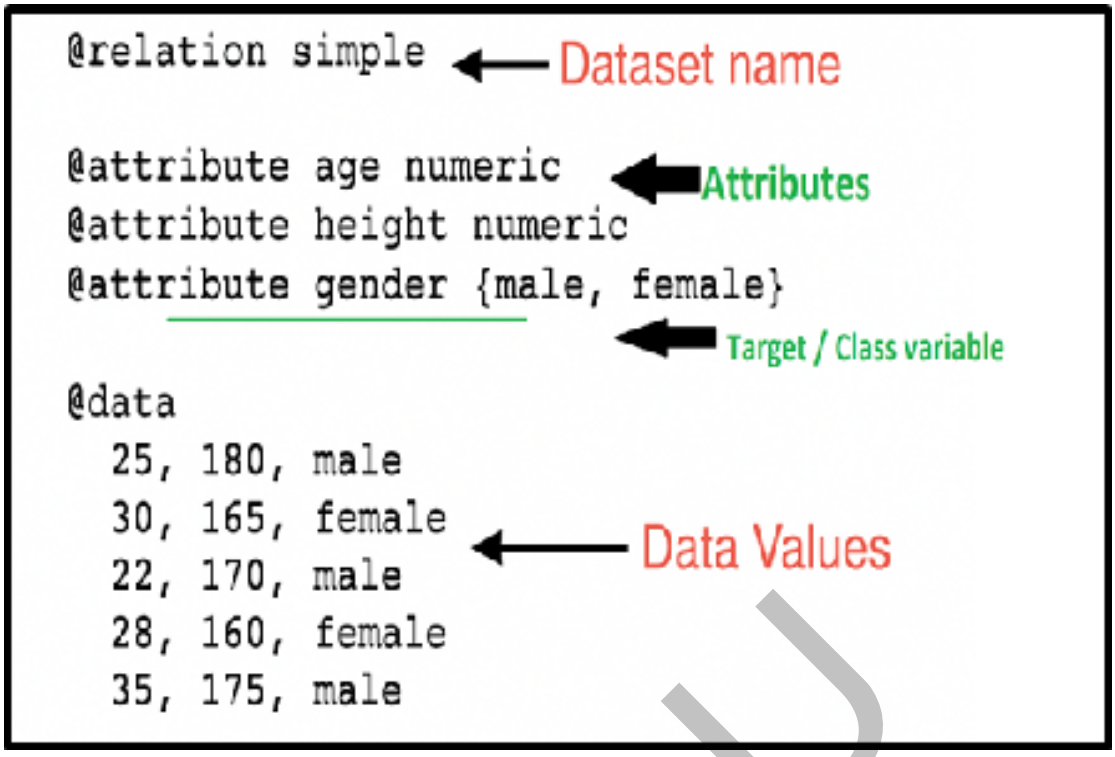


Fig. 4.2.5 Attribute-Relation File Format

- The **@relation** tag specifies *the name of the dataset* or database.
- The **@attribute** tag defines the *attributes (fields or columns)* present in the dataset.
- The **@data** tag indicates the *beginning of the data section*, where each row represents a record with comma-separated values.
- You can also designate one of the attributes as the *Target or Class variable*, which represents the *output or label* to be predicted.

Apart from ARFF, Weka is compatible with several other file formats, enhancing its flexibility and usability with different data sources:

- **CSV (Comma-Separated Values):** This common format stores tabular data as plain text, with values separated by commas. Although Weka can directly import CSV files, they do not include metadata like attribute descriptions found in ARFF files.
- **JSON (JavaScript Object Notation):** A lightweight format for data exchange, JSON is supported by Weka and is ideal for representing hierarchical or complex data structures.
- **XRFF (XML-based ARFF):** This is the XML variant of ARFF, offering a more structured and readable format for both data and metadata.

- **Other Supported Formats:** Weka also works with additional file types, including *LibSVM, Matlab ASCII,* and *binary serialized instances*, providing broad compatibility with various machine learning tools and environments.

4.2.1.5 Main Machine Learning Techniques in WEKA

WEKA supports several key machine learning approaches:

- **Classification:** Assigns data to predefined categories.

  Examples: Decision Tree, Naïve Bayes.

- **Clustering:** Groups similar data points into clusters.

  Examples: K-Means, Agglomerative Clustering.

- **Regression:** Predicts continuous values based on input data.

  Example: Linear Regression for predicting sales or prices.

- **Association:** Discovers interesting relationships among data items.

  Example: Market basket analysis to find product purchase patterns.

4.2.1.6 Applications of WEKA

WEKA is widely used in academic, industrial, and research settings for various real-world applications such as:

- **Predictive Modeling:** Forecasting student performance or stock prices.
- **Customer Segmentation:** Grouping customers based on purchasing behavior.
- **Spam and Text Classification:** Identifying spam emails or sentiment in text.
- **Healthcare Analytics:** Predicting diseases based on patient data.
- **Educational Data Mining:** Analyzing learning patterns to improve teaching strategies.

4.2.1.7 Advantages of Using WEKA

WEKA offers a variety of benefits that make it one of the most popular tools for learning and applying machine learning and data mining techniques. Its user-friendly interface and wide range of algorithms make it suitable for students, researchers, and professionals alike.

- **Free and Open Source:** WEKA is free to use and easily available for everyone.
- **Easy to Use:** It has a simple graphical interface that helps beginners use it without programming.

- **Many Algorithms:** WEKA includes several built-in algorithms for classification, clustering, and regression.
- **Data Preprocessing:** It allows cleaning and preparing data before analysis.
- **Visualization Tools:** WEKA provides graphs and charts to display results clearly.
- **Cross-Platform Support:** Since it is written in Java, it works on Windows, Linux, and macOS.
- **Model Comparison:** The Experimenter tool helps compare the performance of different algorithms.
- **Useful for Learning and Research:** It is widely used in colleges and research projects to study data mining and machine learning.

4.2.1.8 Limitations of Weka

Although WEKA offers numerous benefits, it also has a few limitations:

- **Scalability:** WEKA may not perform efficiently with very large datasets because it loads all data into memory at once.
- **Multi-Relational Data Mining:** WEKA lacks built-in support for multi-relational data mining, though external tools can be used to merge related database tables into a single table for analysis.
- **Sequence Modeling:** WEKA does not provide native support for sequence modeling, which restricts its application in some specialized domains.

4.2.2 Decision Tree Classifier

A Decision Tree Classifier is one of the most popular and interpretable machine learning algorithms used for both classification and prediction tasks. It works by splitting the dataset into branches based on attribute values, forming a tree-like structure where each internal node represents a test on an attribute, each branch corresponds to an outcome of the test, and each leaf node represents a class label or decision.

The main advantage of decision trees lies in their simplicity and interpretability. They mimic human decision-making by using a series of "if-then" rules that are easy to understand and visualize.

4.2.2.1 Structure of a Decision Tree

The structure of a Decision Tree forms the foundation of how the algorithm makes predictions and classifications. It represents data in a hierarchical and tree-like form, where decisions are made at different levels based on the values of attributes. Understanding the structure of a decision tree is essential for analyzing how the model learns patterns and classifies data efficiently. A Decision Tree consists of the following components:

- **Root Node:** Represents the entire dataset and is split into two or more homogeneous sets.
- **Internal Nodes:** Represent tests on attributes that further divide the dataset.
- **Branches:** Indicate the outcome of each test condition.
- **Leaf/Terminal Nodes:** Represent the final class label or decision outcome.

4.2.2.2 Working Principle

The Decision Tree algorithm follows these basic steps:

1. **Select the Best Attribute:** The algorithm identifies the attribute that best divides the data into distinct classes. Criteria such as *Information Gain*, *Gain Ratio*, or *Gini Index* are used for this purpose.

Note: *Information Gain, Gain Ratio*, and *Gini Index* are statistical measures used in Decision Tree algorithms to decide which attribute should be used to split the data at each step.

- **Information Gain** measures how much uncertainty in the dataset is reduced after splitting based on an attribute.
- **Gain Ratio** adjusts Information Gain by considering the number and size of branches, helping to avoid bias toward attributes with many values.
- **Gini Index** measures the impurity or diversity of data; a lower Gini value means the node is more pure.

These criteria help the algorithm choose the best attribute that separates the data most effectively.

2. **Split the Dataset:** The dataset is partitioned into subsets based on the values of the selected attribute.
3. **Recursive Partitioning:** The process continues recursively for each subset until one of the following conditions is met:
    - All records belong to the same class.
    - There are no remaining attributes to split.
    - The tree reaches a predefined depth limit.
4. **Assign Labels:** Once the tree is built, each leaf node is labeled with the majority class of data samples in that subset.

4.2.2.3 Common Decision Tree Algorithms

There are several algorithms used to build Decision Trees, each with its own method for choosing the best attribute and splitting the data. These algorithms help improve the

accuracy and efficiency of the tree. Some of the most commonly used Decision Tree algorithms include ID3, C4.5, CART, and J48. Each of these algorithms follows the same basic idea of dividing data but uses different rules or measures to decide how to make the splits.

- **ID3 (Iterative Dichotomiser 3):** Uses *Information Gain* as the splitting criterion.
- **C4.5 (Successor of ID3):** Uses *Gain Ratio* and handles continuous attributes and missing values.
- **CART (Classification and Regression Tree):** Uses the *Gini Index* and can perform both classification and regression.
- **J48 (WEKA Implementation of C4.5):** A widely used algorithm available in WEKA for constructing decision trees.

4.2.2.4 Advantages of Decision Tree Classifiers

Decision Tree Classifiers offer several benefits that make them popular in data analysis and machine learning. They are simple to understand, easy to visualize, and work well with both numerical and categorical data. Because of their clear structure and logical flow, Decision Trees help users easily see how decisions are made. Some advantages are:

- Easy to interpret and visualize.
- Handles both numerical and categorical data.
- Requires little data preprocessing (no need for feature scaling).
- Works well for small to medium-sized datasets.
- Can model non-linear relationships.

4.2.2.5 Limitations of Decision Tree Classifiers

Although Decision Tree Classifiers are easy to use and interpret, they also have some drawbacks. Small changes in the data can also affect the structure of the tree. Understanding the limitations helps in choosing the right method and improving the model's accuracy and reliability. Some limitations are:

- Prone to overfitting, especially with noisy data.
- Sensitive to small variations in the data.
- May produce complex trees that are hard to generalize.
- Biased toward attributes with more levels or categories.

4.2.2.6 Applications of Decision Tree Classifiers

Decision Tree Classifiers are widely used in various fields to support decision-making and prediction tasks. They help analyze patterns, classify data, and provide clear, rule-based results that are easy to interpret.

- **Medical Diagnosis:** Used to predict the presence or absence of a disease based on patient data.
- **Customer Churn Analysis:** Helps identify customers who are likely to stop using a service.
- **Credit Risk Assessment:** Evaluates whether a borrower is likely to repay a loan.
- **Weather Forecasting:** Predicts weather conditions such as rain or sunshine using past data.
- **Student Performance Prediction:** Estimates students' academic outcomes based on factors like attendance, grades, and study habits.

4.2.3 Decision Tree Classifier implementations using WEKA

WEKA provides an intuitive interface for building and visualizing decision trees. The *J48 algorithm* (based on C4.5) is the most commonly used decision tree classifier in WEKA. Implementing a Decision Tree Classifier in WEKA is a simple and practical way to understand how decision trees work. WEKA provides a user-friendly interface that allows users to build, test, and visualize models without writing code. By following a few easy steps such as loading the dataset, selecting the classifier, setting parameters, and viewing results users can quickly create and analyze decision trees. These steps help in understanding both the working process and the performance of the model in real-world applications. Steps to implement Decision Tree classifier in WEKA are:

Step 1: Open WEKA Explorer

- Launch the WEKA GUI Chooser and select Explorer mode.

Step 2: Load Dataset

- Click Open File and choose a dataset (for example weather.nominal.arff or a CSV file).
- The dataset attributes and instances are displayed in the Preprocess tab.

Step 3: Select the Classifier

- Go to the Classify tab.
- Choose the Test Option: Select how the model should be evaluated. Here there are four testing options (Figure 4.2.6).

- Use Training Set
- Supplied Test Set
- Cross-Validation (commonly 10-fold)
- Percentage Split (e.g., 70% training, 30% testing)

In *cross-validation*, the dataset is divided into a specified number of folds, and each fold is used once for testing while the others are used for training. In the percentage split method, the data is divided into training and testing sets based on a chosen percentage, such as 70% for training and 30% for testing.

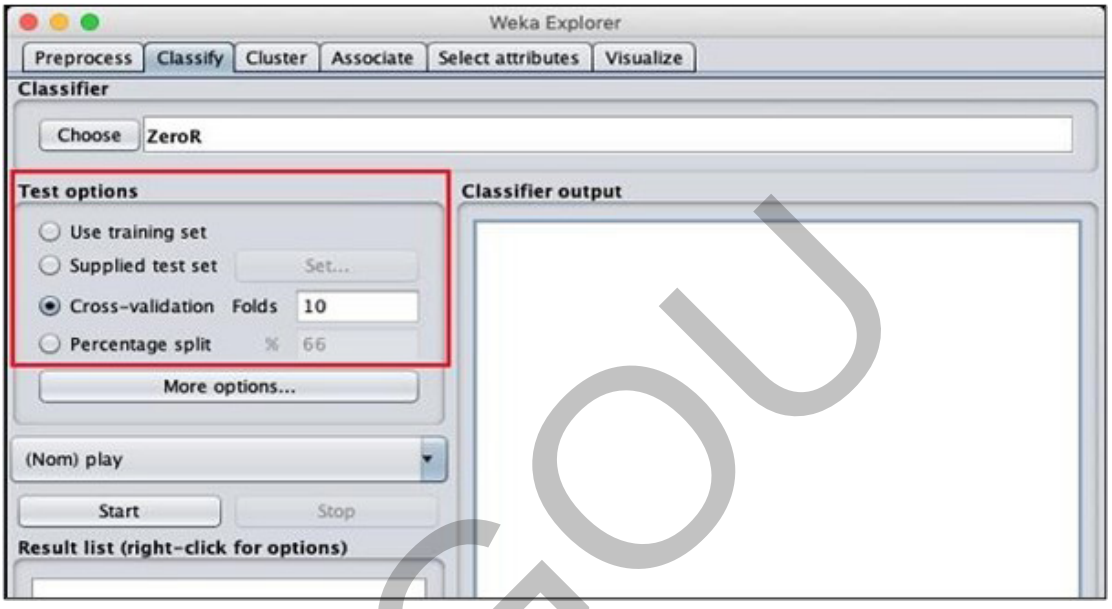


Fig. 4.2.6 Test options

- Next select the classifier.
- Click *trees* → *J48* to select the decision tree classifier (Figure 4.2.7).

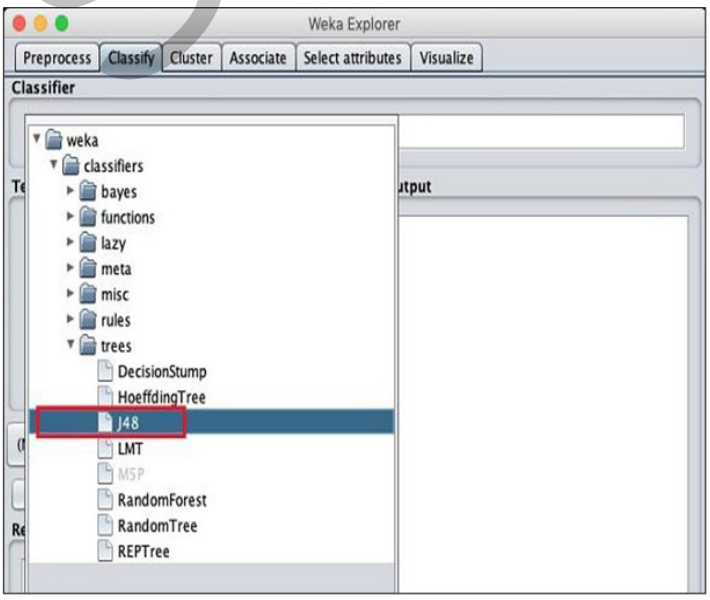


Fig. 4.2.7 Tree classifier

Step 4: Set Parameters (Optional)

- Click on *J48* to open the configuration dialog.
- Common parameters include:
  - **Confidence Factor (C):** Used for pruning the tree (default = 0.25).
  - **Minimum Number of Instances per Leaf (M):** Controls when to stop splitting.
  - **Use Unpruned Tree:** To disable pruning (unchecked by default).

Step 5: Start the Classification

- Click *Start* to build and evaluate the model.
- The *Result List* area shows the output, including accuracy, confusion matrix, and other performance metrics (Figure 4.2.8).

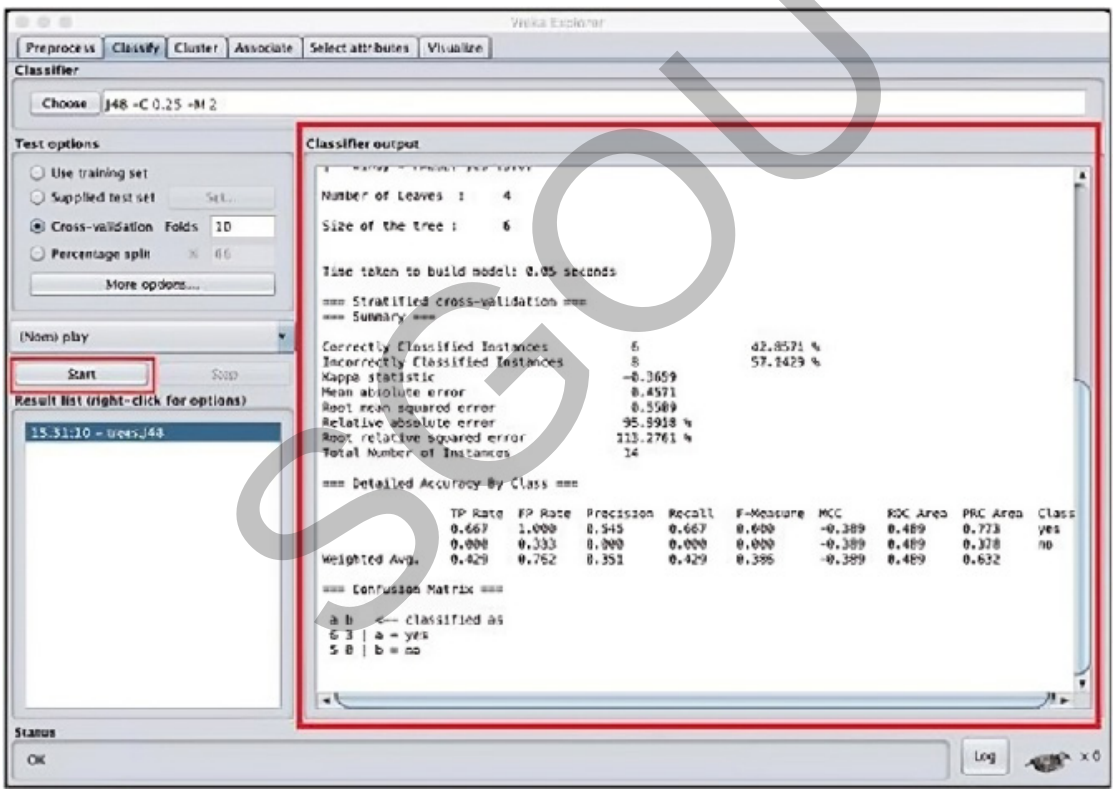


Fig. 4.2.8 Result List of Decision Ttree classifier
See the e-content for a color version of this image

Step 6: View the Decision Tree

- Right-click on the result entry and select *Visualize Tree (*Figure 4.2.9).

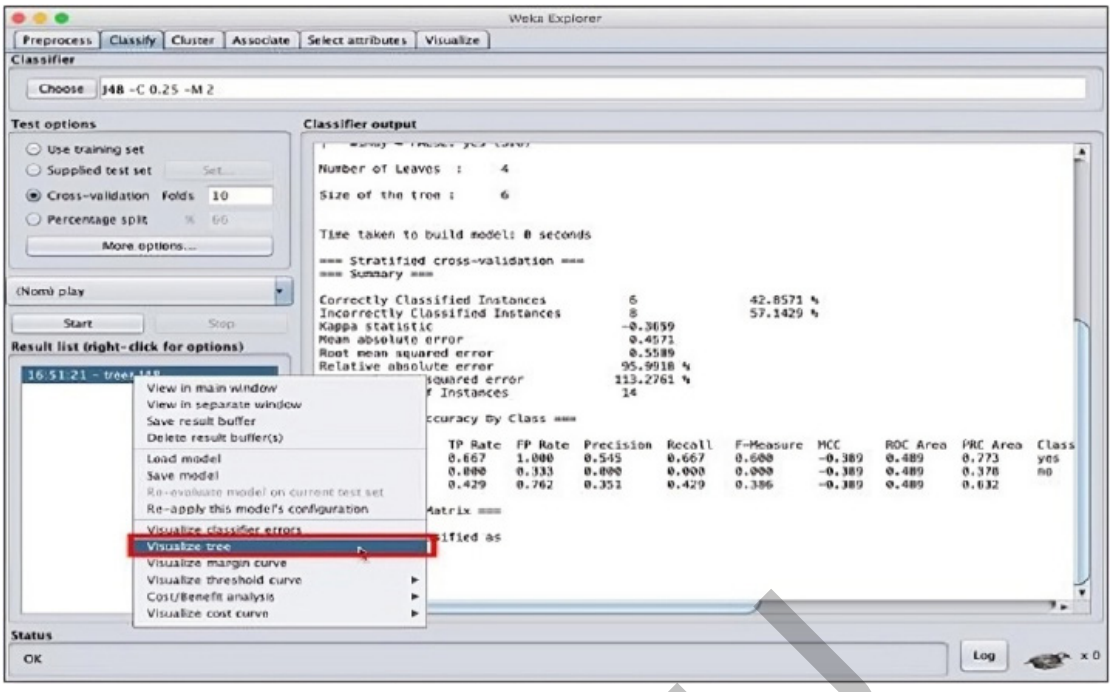


Fig. 4.2.9 Visualize Tree
See the e-content for a color version of this image

- The graphical tree diagram (Figure 4.2.10) displays nodes (attribute tests) and leaves (class decisions).

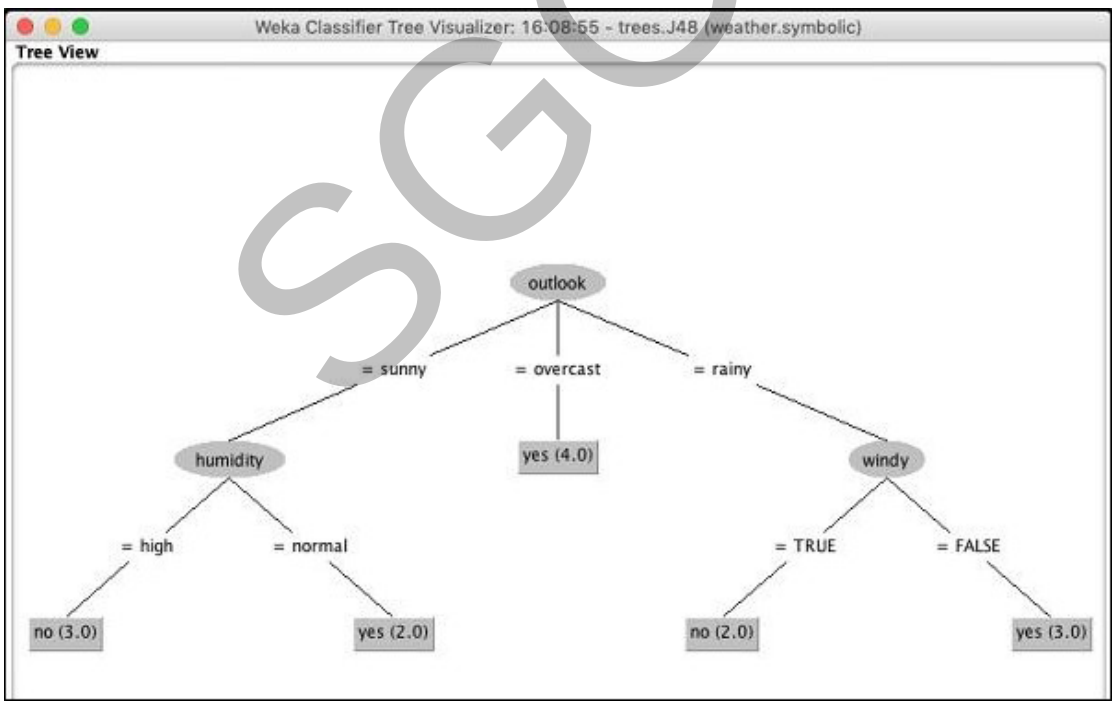


Fig. 4.2.10 Graphical tree diagram

Example: Weather Dataset

Table 4.2.2 Weather Dataset

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Overcast	Mild	High	True	Yes
Rain	Cool	Normal	False	Yes

After applying J48 in WEKA, the resulting tree might look like this:

Outlook = Sunny: No

Outlook = Overcast: Yes

Outlook = Rain:

Windy = True: No

Windy = False: Yes

This simple tree clearly shows the decision rules based on the "Outlook" and "Windy" attributes.

Interpreting the Output

WEKA provides the following results after training:

- Correctly Classified Instances (% Accuracy)
- Confusion Matrix (shows predicted vs. actual classes)
- Precision, Recall, F-measure for each class
- Decision Tree Visualization (for better understanding of classification logic)

The Decision Tree Classifier is a powerful and interpretable algorithm for classification tasks. Through WEKA's J48 implementation, users can easily construct, visualize, and evaluate decision trees without programming knowledge. However, proper parameter tuning and pruning are essential to prevent overfitting and ensure accurate, generalizable models.

4.2.4 Naïve Bayes Classifier

The Naïve Bayes Classifier is a simple yet powerful probabilistic machine learning algorithm used mainly for classification tasks. It is based on Bayes' Theorem, which describes the probability of an event based on prior knowledge of related conditions. The term "naïve" comes from the assumption that all attributes in the dataset are independent of each other, which simplifies computation.

Despite this strong assumption, Naïve Bayes performs remarkably well in many real-world applications, especially those involving text classification, spam filtering, and sentiment analysis. Bayes' Theorem provides a mathematical way to calculate conditional probabilities. The formula is,

$$P(H|E) = \frac{P(E|H) \times P(H)}{P(E)}$$

where:

- P(H|E) = Probability of hypothesis *H* being true given the evidence *E* (posterior probability).
- P(E|H) = Probability of evidence *E* occurring when *H* is true (likelihood).
- P(H)= Probability of hypothesis *H* being true (prior probability).
- P(E)= Probability of evidence *E* (marginal probability).

In classification, the hypothesis *H* represents the class, and the evidence *E* represents the features of the data instance.

4.2.4.1 Working Principle

The Naïve Bayes Classifier works by calculating the probability of each class for a given instance and then selecting the class with the highest probability. Steps are:

1. Calculate the prior probability for each class.
2. For each attribute, calculate the conditional probability given each class.
3. Multiply all probabilities for each class.
4. Choose the class with the highest resulting probability.

Although it assumes independence among attributes, the method is computationally efficient and gives good results in practice.

4.2.4.2 Advantages of Naïve Bayes Classifier

The Naïve Bayes Classifier offers several advantages that make it one of the most popular and practical algorithms for classification tasks. It is simple to understand, quickly to build, and highly efficient even with large datasets. The strengths make Naïve Bayes a preferred choice for applications such as spam detection, sentiment analysis, and document classification. Some advantages are:

- Simple and easy to implement.
- Fast in both training and prediction.
- Performs well with large datasets.
- Works effectively with text and categorical data.
- Handles missing values efficiently.

4.2.4.3 Limitations of Naïve Bayes Classifier

Although the Naïve Bayes Classifier is simple and efficient, it has some important limitations. Understanding the limitations helps in applying the algorithm effectively and interpreting its results correctly. Some limitations are**:**

- Assumes all features are independent, which is rarely true in real data.
- Performs poorly when features are strongly correlated.
- Zero probability problem: if a feature value does not appear in the training set for a class, the probability becomes zero (can be handled using Laplace smoothing).

4.2.4.4 Applications of Naïve Bayes Classifier

The Naïve Bayes Classifier is widely used in many real-world applications where quick and reliable classification is needed. Its ability to handle large amounts of data and produce accurate results makes it suitable for tasks such as,

- **Email Spam Detection:** Identifies whether an incoming email is spam or legitimate based on its content and keywords.
- **Text and Document Classification:** Categorizes documents or articles into topics such as sports, education, or technology.
- **Sentiment Analysis:** Determines whether a review or comment expresses a positive or negative opinion.
- **Medical Diagnosis:** Helps predict diseases by analyzing patient symptoms and medical data.
- **Weather Prediction:** Estimates weather conditions like rain or sunshine using historical data patterns.

4.2.5 Naïve Bayes Classifier Implementation using WEKA

Implementing a Naïve Bayes Classifier in WEKA is a straightforward process that helps users understand how probabilistic classification works. WEKA provides an easy-to-use graphical interface where users can load datasets, select the Naïve Bayes algorithm, train the model, and analyze results without any programming. Following these steps allows learners to see how the algorithm classifies data based on probabilities and how well it performs on different datasets. Steps to implement Naïve Bayes Classifier in WEKA are:

Step 1: Open WEKA Explorer

- Start WEKA and choose the *Explorer* interface from the main menu.

Step 2: Load Dataset

- Click *Open File* and select your dataset (for example, *weather.nominal.arff*).
- The attributes and number of instances will appear in the *Preprocess* tab.

Step 3: Select the Classifier

- Go to the Classify tab.
- Click Choose → bayes → NaiveBayes. (Figure 4.2.11).

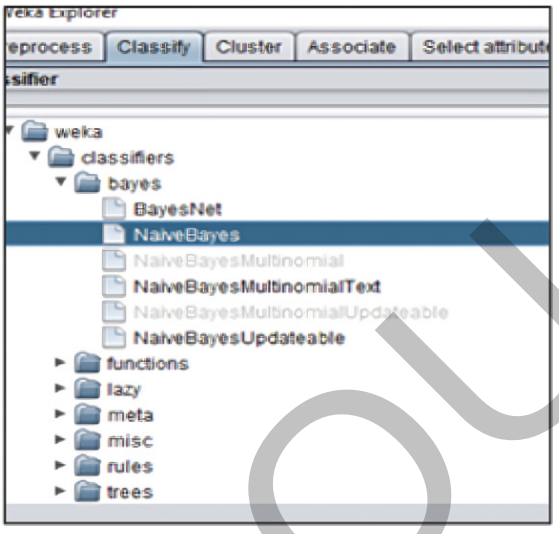


Fig. 4.2.11 Select the Classifier

To adjust the parameters, select the 'choose' button positioned on the right side, and in this instance, we'll retain the default values (Figure 4.2.12).

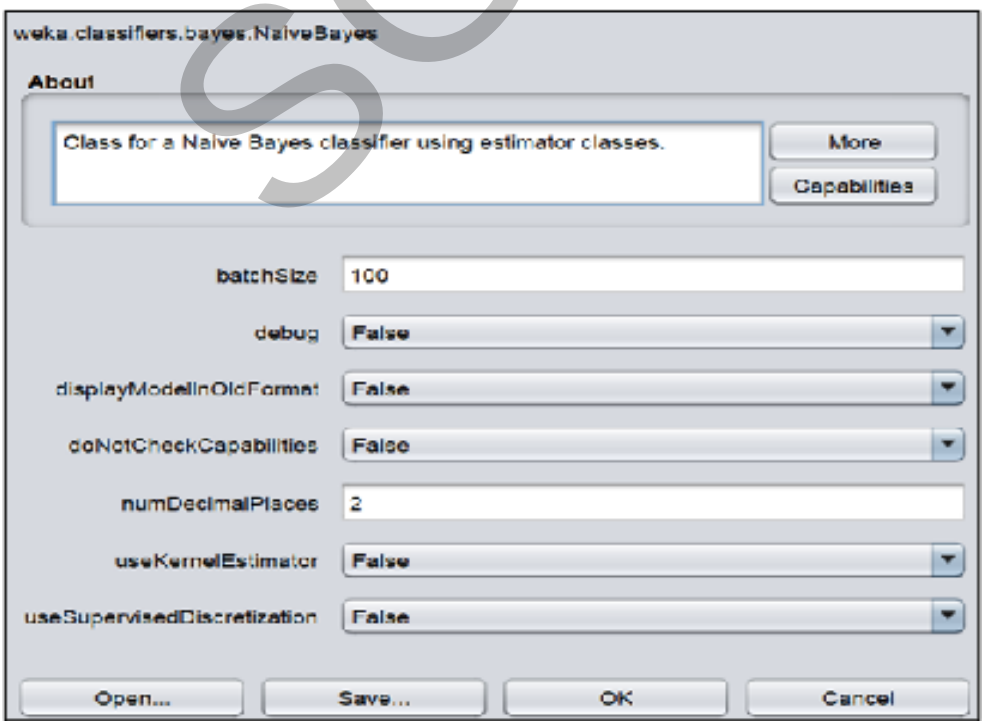


Fig. 4.2.12 Adjust the parameters

Step 4: Choose the Test Option

Select how you want to test your model:

- Use training set
- Supplied test set
- Cross-validation (commonly 10-fold)
- Percentage split (e.g., 70% training and 30% testing)

We opt for the *Percentage split* as our evaluation method from the available options under "Test" in the main panel. Since we lack a distinct test dataset, we'll employ a 66 percent split to obtain a reliable assessment of the model's accuracy (Figure 4.2.13).

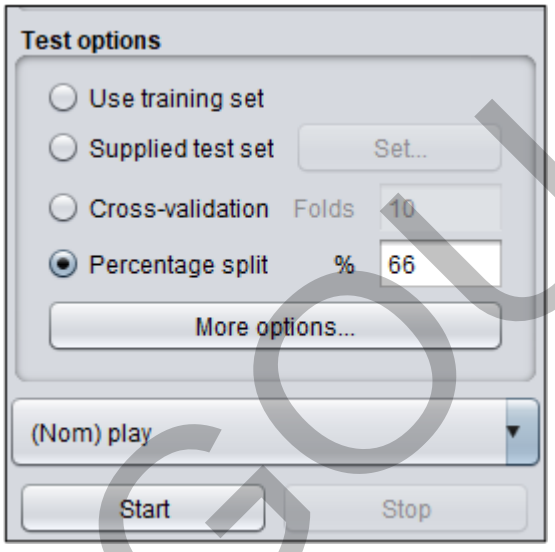


Fig. 4.2.13 Test options

Step 5: Start the Classification

- Click *Start* to train and test the model.
- The *Result List* section displays the classification results, accuracy, and confusion matrix.

Once the model generation process is finished, the evaluation statistics will appear in the right panel (Figure 4.2.14).

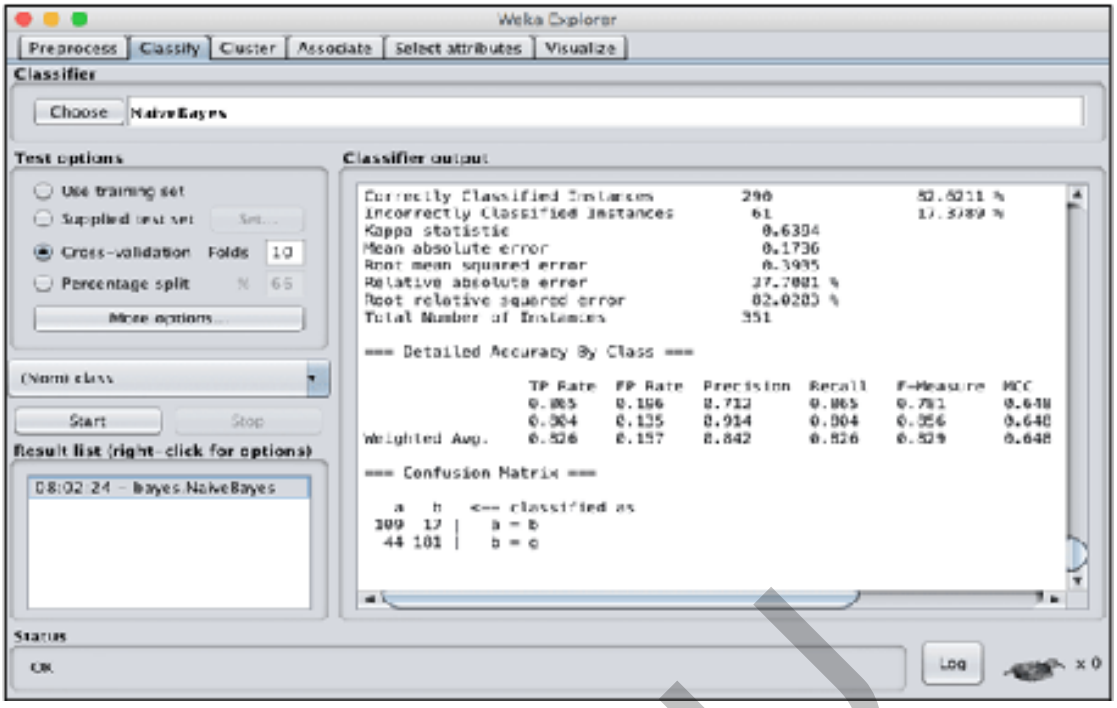


Fig. 4.2.14 Evaluation statistics

Step 6: Analyze the Output

After training, WEKA shows:

- Correctly classified instances (accuracy)
- Confusion matrix (shows predicted vs. actual values)
- Precision, Recall, and F-measure for each class
- Summary statistics, such as mean absolute error and root mean squared error.

To visualize, click Visualize Threshold Curve to see how the data points are grouped in 2D space (Figure 4.2.15).

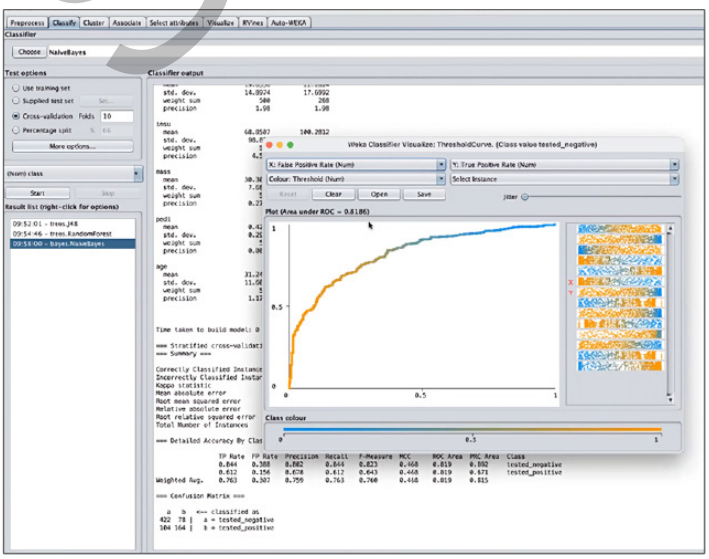


Fig. 4.2.15 Visualize Threshold Curve
See the e-content for a color version of this image

The *Naïve Bayes Classifier* is a fast, simple, and effective algorithm for classification tasks. It works well for large datasets and problems involving categorical data, such as text or email classification. WEKA makes it easy to build, test, and understand Naïve Bayes models without programming, helping learners grasp the concept of probabilistic classification.

4.2.6 K-Means Clustering

K-Means Clustering is one of the most widely used unsupervised learning algorithms in data mining and machine learning. Unlike classification algorithms, which use labeled data, clustering algorithms group similar data points together based on their features without using predefined labels**.**

The goal of K-Means is to divide a dataset into *K clusters*, where each cluster contains data points that are more similar to each other than to those in other clusters. It is efficient, easy to understand, and works well for large datasets.

4.2.6.1 Working Principle

K-Means works by minimizing the distance between data points and their cluster centers (centroids)**.** Step-by-step process are:

1. **Choose the number of clusters (K):** Decide how many groups (clusters) you want to form in the dataset.

2. **Initialize centroids:** Randomly select K data points as the initial cluster centers.

3. **Assign points to nearest centroid:** Each data point is assigned to the cluster with the nearest centroid using a distance measure (usually *Euclidean distance*).

4. **Recalculate centroids:** Compute the mean of all data points in each cluster to find the new centroids.

5. **Repeat until convergence:** Steps 3 and 4 are repeated until the centroids do not change significantly or the maximum number of iterations is reached.

The result is a set of clusters where data points within a cluster are similar and those in different clusters are dissimilar.

Mathematical Representation

The objective of K-Means is to minimize the *sum of squared distances* between data points and their assigned cluster centers.

$$J = \sum_{i=1}^{K} \sum_{x \in C_i} ||x - \mu_i||^2$$

where:

- K = number of clusters
- Ci= set of points in cluster *i*
- $\grave{i}_i$= mean (centroid) of cluster *i*

4.2.6.2 Advantages of K-Means Clustering

The K-Means Clustering algorithm offers several advantages that make it one of the most popular clustering methods in data analysis. Some advantages are:

- Simple and easy to implement.
- Efficient for large datasets.
- Produces well-separated and distinct clusters.
- Converges quickly in most cases.

4.2.6.3 Limitations of K-Means Clustering

Although K-Means Clustering is efficient and easy to use, it has several limitations. Some imitations are:

- Requires the number of clusters (K) to be specified in advance.
- Sensitive to the initial placement of centroids.
- Struggles with non-spherical or overlapping clusters.
- Affected by outliers and noisy data.

4.2.6.4 Applications of K-Means Clustering

The K-Means Clustering algorithm is widely applied in various real-world domains where grouping or pattern discovery is required. Its ability to efficiently organize large datasets into meaningful clusters makes it useful in areas such as:

- **Customer Segmentation in Marketing:** Groups customers based on purchasing behavior, preferences, or demographics.
- **Document or Text Grouping:** Organizes similar documents or articles into categories for easier retrieval and analysis.
- **Image Compression and Segmentation:** Reduces image size by clustering pixels with similar colors or features.
- **Pattern Recognition:** Identifies patterns and similarities in data for applications like handwriting or speech recognition.
- **Market Basket Analysis:** Groups products frequently bought together to improve cross-selling and marketing strategies.

4.2.7 Implementing K-Means Clustering in WEKA

Implementing K-Means Clustering in WEKA allows users to explore unsupervised learning visually and practically. WEKA's Simple K-Means algorithm provides a straightforward way to create and analyze clusters without writing code. The process involves loading a dataset, selecting the clustering algorithm, setting the number of clusters, and viewing the results. Steps to implement K-Means clustering in WEKA are:

Step 1: Open WEKA Explorer

- Launch the WEKA interface and select Explorer mode.

Step 2: Load Dataset

- Click Open File and choose your dataset.
- The dataset attributes and instances will appear under the Preprocess tab.

Step 3: Select the Clustering Algorithm

- Go to the Cluster tab.
- Click Choose → SimpleKMeans from the list of clustering algorithms (Figure 4.2.16).

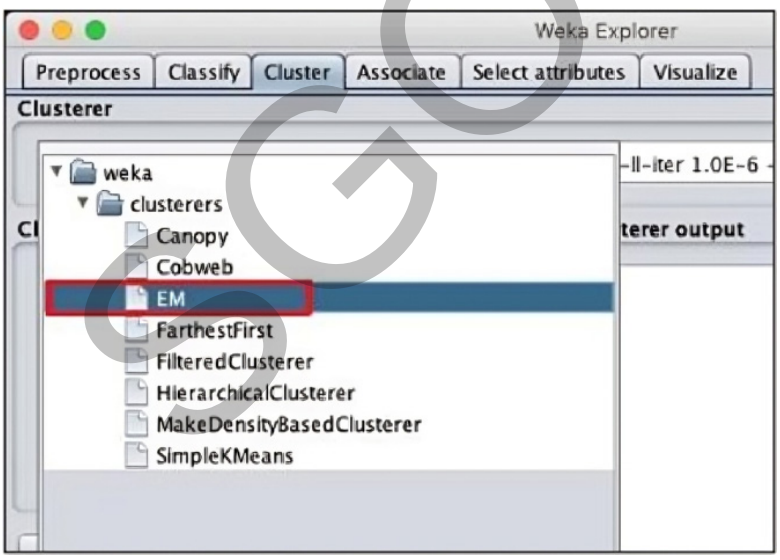


Fig. 4.2.16 Select SimpleKMeans

Step 4: Set Parameters (Optional)

Click on SimpleKMeans to open the settings window and adjust parameters (Figure 4.2.17) such as:

- **NumClusters (K):** The number of clusters to form (e.g., 3 for the Iris dataset).
- **DistanceFunction:** Usually set to *EuclideanDistance*.

- **MaxIterations:** Defines how many times the algorithm will refine centroids.
- **PreserveInstancesOrder:** Keeps data in the same order in the output.

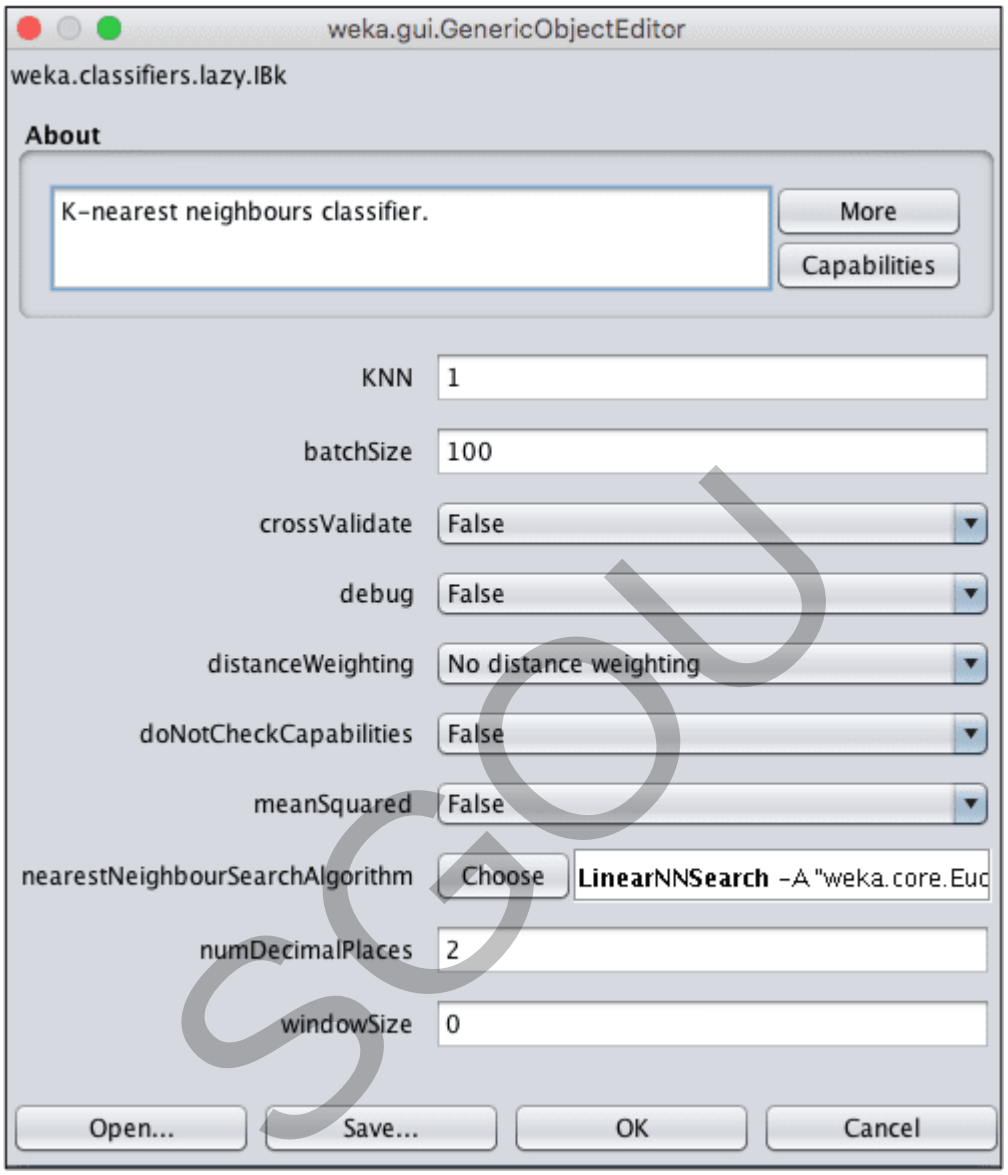


Fig. 4.2.17 Set Parameters

Step 5: Choose the Mode of Evaluation

Under Cluster mode, select how you want WEKA to evaluate the results:

- **Use training set**: clusters the entire dataset.
- **Classes to clusters evaluation**: compares clusters to actual class labels (if available).

Step 6: Run the Clustering Process

- Click Start to execute the algorithm.

- WEKA will display the cluster summary in the Result list window (Figure 4.2.18).

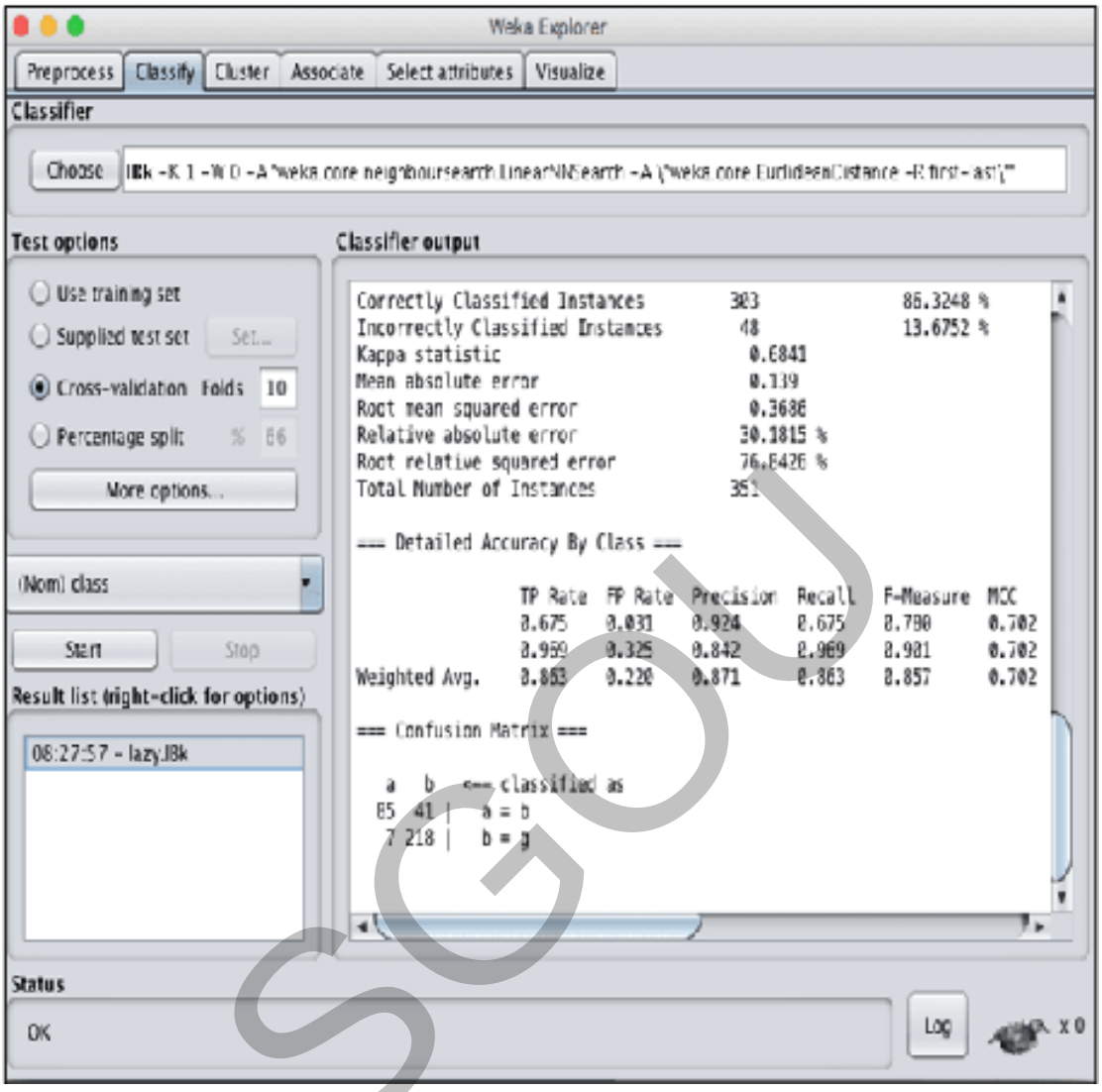


Fig. 4.2.18 Result list window

Step 7: View and Interpret Results

- The output includes information such as:
  - Cluster centroids (mean values of attributes for each cluster).
  - Number of instances in each cluster.
  - Clustered Instances by ID.
- To visualize, click Visualize Cluster Assignments to see how the data points are grouped in 2D space (Figure 4.2.19).

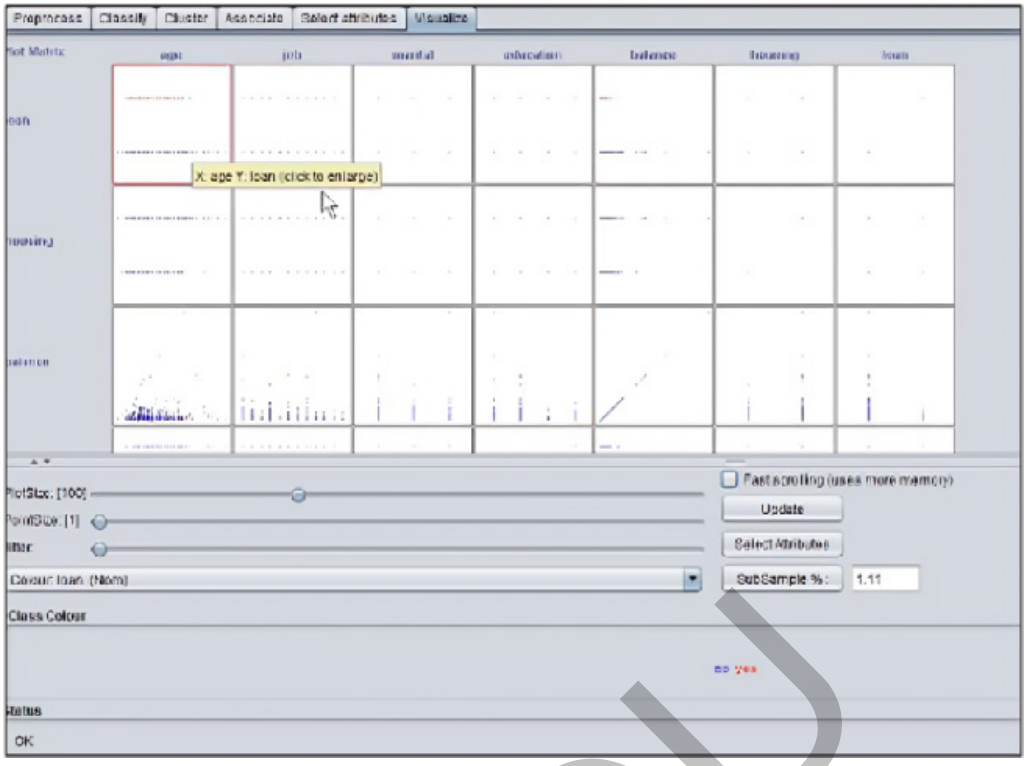


Fig. 4.2.19 Visualize Cluster Assignments
See the e-content for a color version of this image

K-Means Clustering is an efficient and practical algorithm for finding natural groupings in data. Using WEKA's *SimpleKMeans*, users can easily perform clustering, visualize group structures, and interpret results without programming. This hands-on approach helps learners understand how unsupervised learning identifies patterns and similarities within datasets.

4.2.8 Agglomeration Clustering

Agglomerative Clustering is a type of Hierarchical Clustering that builds clusters step by step by merging smaller clusters into larger ones. It is called *bottom-up* clustering because it starts with each data point as a single cluster and gradually combines them based on their similarity until only one cluster remains or the desired number of clusters is reached.

This method is particularly useful for discovering nested group structures in data and visualizing relationships using a *dendrogram*, which shows how clusters are merged at each step.

4.2.8.1 Working Principle

Agglomerative clustering group's data points based on distance or similarity measures such as Euclidean distance**.** The process continues until all data points belong to one cluster or a stopping condition is met. Step-by-Step Process are:

1. **Start with individual clusters:** Treat each data point as its own cluster.

2. **Calculate the distance matrix:** Compute distances between all clusters (using metrics such as Euclidean, Manhattan, or cosine similarity).

3. **Merge the closest clusters:** Combine the two clusters that are most similar or have the smallest distance between them.

4. **Update the distance matrix:** Recalculate the distances between the new cluster and all other clusters.

5. **Repeat steps 3 and 4:** Continue merging clusters until only one cluster remains or the desired number of clusters is formed.

4.2.8.2 Advantages of Agglomerative Clustering

The Agglomerative Clustering technique offers several advantages that make it useful for exploring and understanding data structures.

- Does not require specifying the number of clusters in advance.
- Produces a *dendrogram* that shows the full hierarchy of cluster merging.
- Can handle small datasets effectively.
- Works with different distance and linkage methods.

4.2.8.3 Limitations of Agglomerative Clustering

Although Agglomerative Clustering is an effective method for discovering hierarchical relationships in data, it has several limitations.

- Computationally expensive for large datasets.
- Once clusters are merged, they cannot be split again.
- Sensitive to noise and outliers.
- May create imbalanced cluster sizes depending on linkage choice.

4.2.8.4 Applications of Agglomerative Clustering

The Agglomerative Clustering method is widely used in various fields where understanding relationships and group structures within data is important.

- **Gene Expression Data Analysis in Bioinformatics:** Groups genes with similar expression patterns to understand biological functions.
- **Market Segmentation and Customer Profiling:** Identifies customer groups based on purchasing habits and preferences.
- **Document or Text Grouping:** Clusters similar documents or articles to simplify organization and retrieval.

- **Image Segmentation:** Divides an image into meaningful regions for analysis or object recognition.
- **Social Network Analysis:** Detects communities or groups of users with similar connections or interactions.

4.2.9 Implementing Agglomeration Clustering in WEKA

WEKA provides an easy way to perform Agglomerative (Hierarchical) Clustering through the Hierarchical Clusterer algorithm. This method helps users visualize how clusters are formed step by step and understand relationships among data points. The graphical interface allows easy adjustment of parameters such as linkage type and distance measure. Steps to implement Agglomerative Clustering in WEKA are:

Step 1: Open WEKA Explorer

- Launch the WEKA interface and select Explorer mode.

Step 2: Load Dataset

- Click Open File and select a dataset (e.g., iris.arff or weather.numeric.arff).
- The dataset attributes and instances will appear in the Preprocess tab.

Step 3: Select the Clustering Algorithm

- Go to the Cluster tab.
- Click Choose → hierarchical → HierarchicalClusterer to select the Agglomerative Clustering algorithm (Figure 4.2.20).

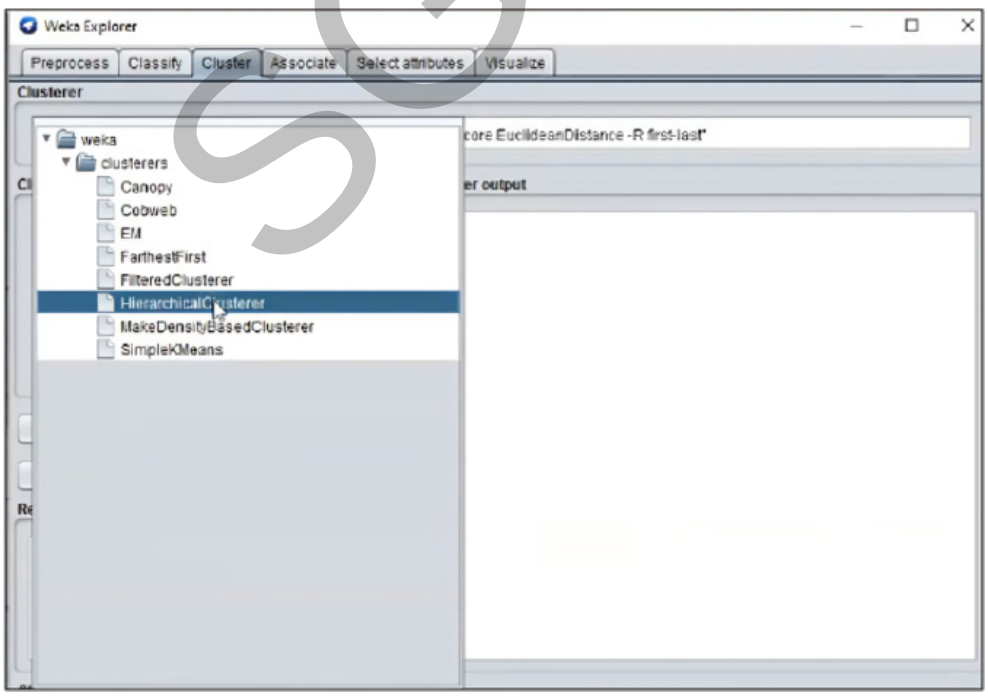


Fig. 4.2.20 HierarchicalClusterer option
See the e-content for a color version of this image

Step 4: Set Parameters (Optional)

Click on *HierarchicalClusterer* to open the configuration window and set parameters (Figure 4.2.21) such as:

- **NumClusters:** Desired number of final clusters (optional; WEKA can determine it automatically).
- **LinkType:** Choose from *SINGLE*, *COMPLETE*, *AVERAGE*, or *CENTROID* linkage.
- **DistanceFunction:** Select the distance measure (default is *EuclideanDistance*).
- **PrintNewick:** Option to print the cluster tree in Newick format (for dendrograms).

Note: Linkage Criteria

The way distances between clusters are measured after merging depends on the *linkage method* used:

a. **Single Linkage:** Distance between the closest points of two clusters.

b. **Complete Linkage:** Distance between the farthest points of two clusters.

c. **Average Linkage:** Average distance between all pairs of points in the two clusters.

d. **Centroid Linkage:** Distance between the centroids (mean points) of the clusters.

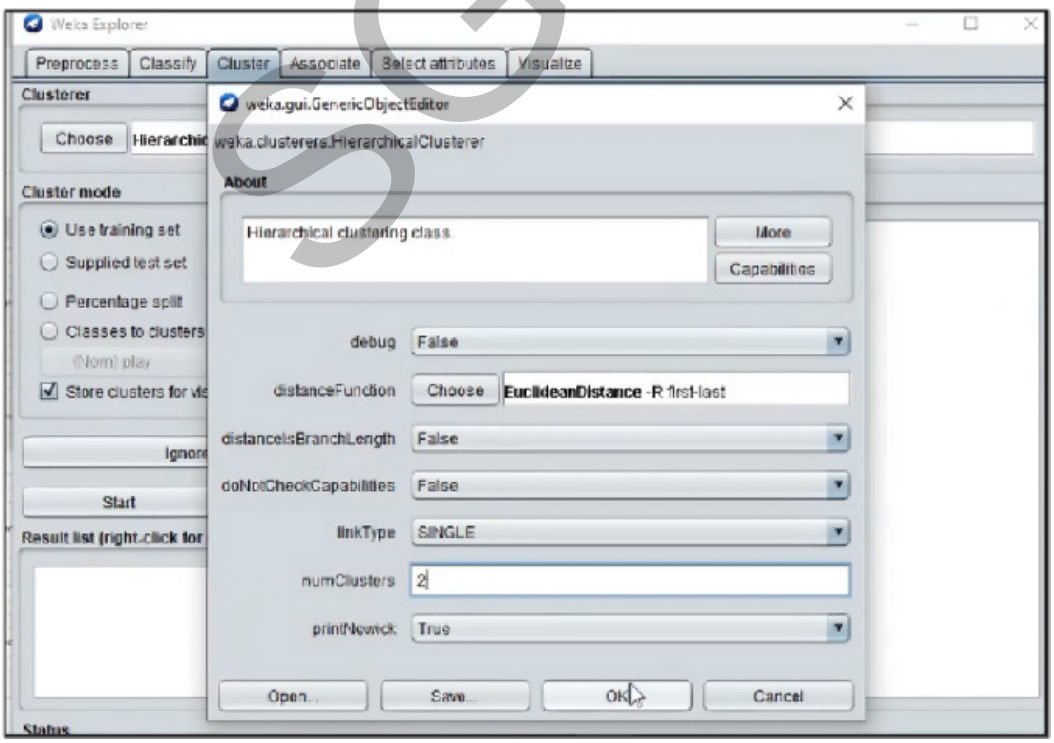


Fig. 4.2.21 Set parameters
See the e-content for a color version of this image

Step 5: Choose the Mode of Evaluation

Under Cluster mode, choose one of the following options:

- **Use training set:** Performs clustering on the entire dataset.
- **Classes to clusters evaluation:** Compares generated clusters with actual class labels (if available).

Step 6: Run the Clustering Process

- Click Start to execute the algorithm.
- WEKA will build the hierarchical cluster structure and display the results in the Result List panel (Figure 4.2.22).

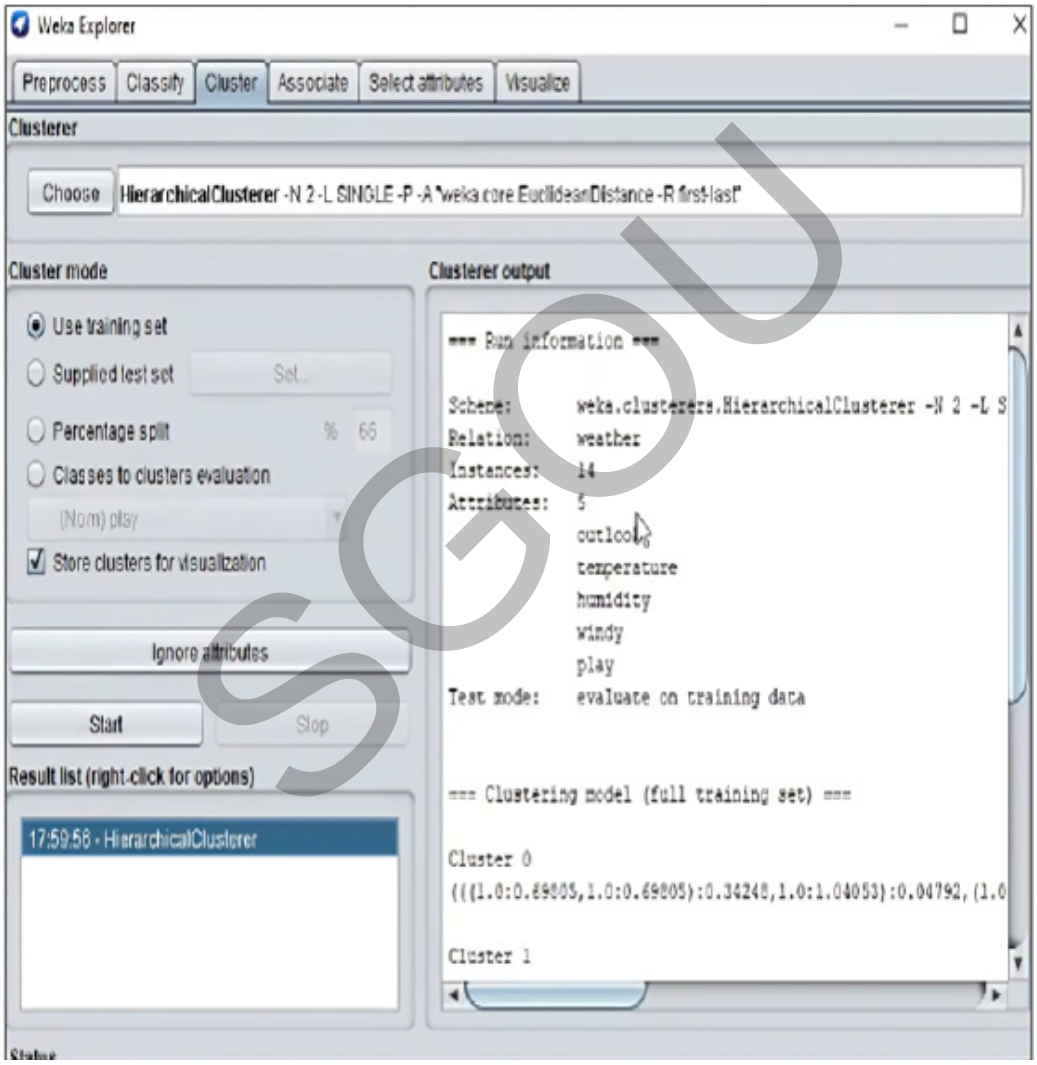


Fig. 4.2.22 Result List
See the e-content for a color version of this image

Step 7: View and Interpret Results

The output includes:

- Cluster assignments for each instance.

- Number of clusters formed.
- Linkage and distance function used.
- Dendrogram visualization in figure 4.2.23 (available through "Visualize Cluster Assignments" or exported in Newick format).

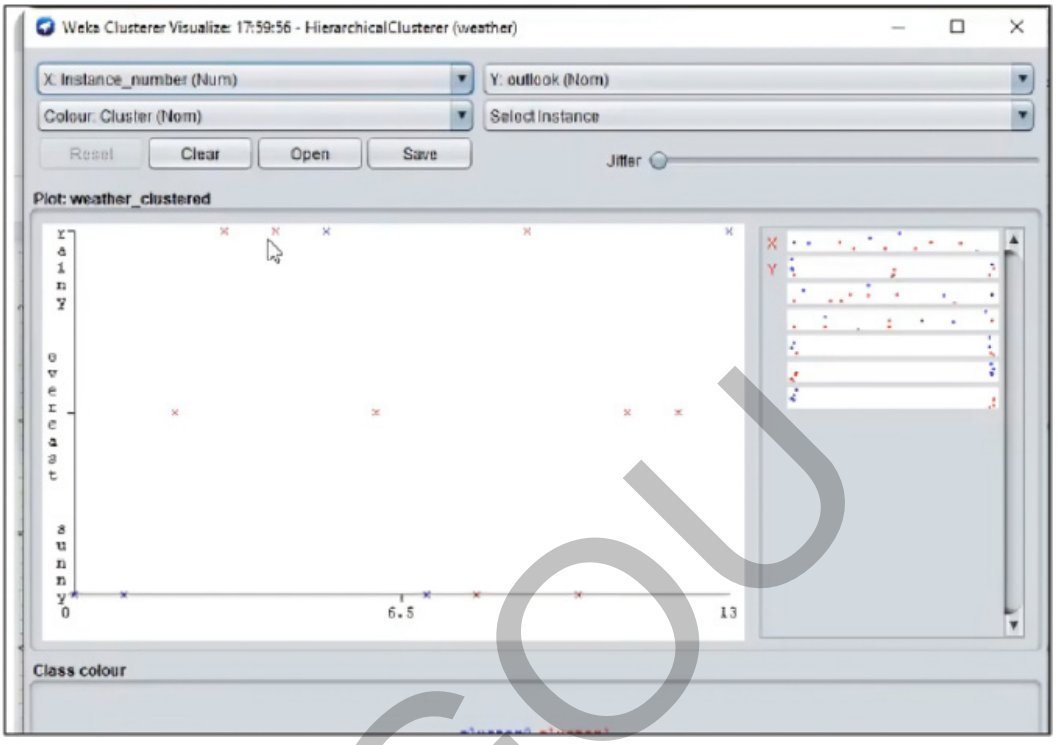


Fig. 4.2.23 Visualize Cluster Assignments
See the e-content for a color version of this image

Agglomerative (Hierarchical) Clustering is a powerful technique for identifying natural group structures and visualizing data relationships. Using WEKA's Hierarchical Clusterer, users can easily experiment with different linkage methods and distance measures to understand how clusters form. This practical approach helps learners grasp how hierarchical clustering builds clusters from the bottom up, making it valuable for both analysis and visualization tasks.

4.2.10 Linear Regression

WEKA provides a simple and interactive way to implement Linear Regression for both simple and multiple variable models. The Linear Regression algorithm in WEKA automatically calculates coefficients, predicts output values, and evaluates model accuracy using standard statistical measures. This makes it ideal for beginners and researchers to understand regression concepts visually.

4.2.10.1 Working Principle

The working principle of Linear Regression is to find a straight-line relationship between the input variables (independent variables) and the output variable (dependent variable).

It assumes that the change in the dependent variable is directly proportional to the change in the independent variable.

The line of best fit is represented by the equation:

$$Y = a + bX$$

where:

- **Y** = predicted value (dependent variable)
- **X** = input value (independent variable)
- **a** = intercept (value of Y when X = 0)
- **b** = slope (how much Y changes for a unit change in X)

The algorithm uses the least squares method to minimize the difference between the actual and predicted values, ensuring the best possible fit for the data.

4.2.10.2 Advantages of Linear Regression

The Linear Regression algorithm offers several advantages that make it a popular choice for predictive modeling. Because of its efficiency and interpretability, Linear Regression is widely used in various fields such as economics, business, and science for forecasting and data analysis. Some advantages are:

- Simple and easy to understand.
- Quick to train and interpret results.
- Works well for small datasets.
- Helps identify the strength of relationships between variables.
- Provides a mathematical model for prediction and forecasting.

4.2.10.3 Limitations of Linear Regression

Although Linear Regression is a widely used and effective predictive technique, it has certain limitations. Some limitations are:

- Assumes a linear relationship between variables, which may not always exist.
- Sensitive to outliers, which can affect the accuracy.
- Not suitable for non-linear data.
- Assumes that the residuals (errors) are normally distributed and independent.
- Performance decreases if predictors are highly correlated (multicollinearity).

4.2.10.4 Applications of Linear Regression

The Linear Regression technique is widely used in various real-world applications where predicting a continuous outcome is required.

- Predicting sales based on advertising expenditure.
- Forecasting trends such as temperature, population, or revenue.
- Risk analysis in finance and insurance.
- Medical data analysis, such as predicting blood pressure based on age and weight.
- Student performance prediction using attendance and past scores.

4.2.11 Implement Linear Regression in WEKA

Implementing Linear Regression in WEKA is a simple and interactive process that allows users to build and analyze predictive models without programming. By following a few easy steps such as loading a dataset, selecting the Linear Regression algorithm, training the model, and evaluating its performance users can understand how regression works and how variables influence the predicted output. Steps to Implement Linear Regression in WEKA are:

Step 1: Open WEKA Explorer

- Launch WEKA and click on Explorer mode.

Step 2: Load Dataset

- Click Open File and select a dataset (e.g., weather.nominal.arff or housing.arff).
- The dataset attributes and instances will be displayed under the Preprocess tab.

Step 3: Select the Algorithm

- Go to the Classify tab.
- Click Choose → functions → LinearRegression to select the algorithm. Click on the name of the algorithm to review the algorithm configuration (Figure 4.2.24).

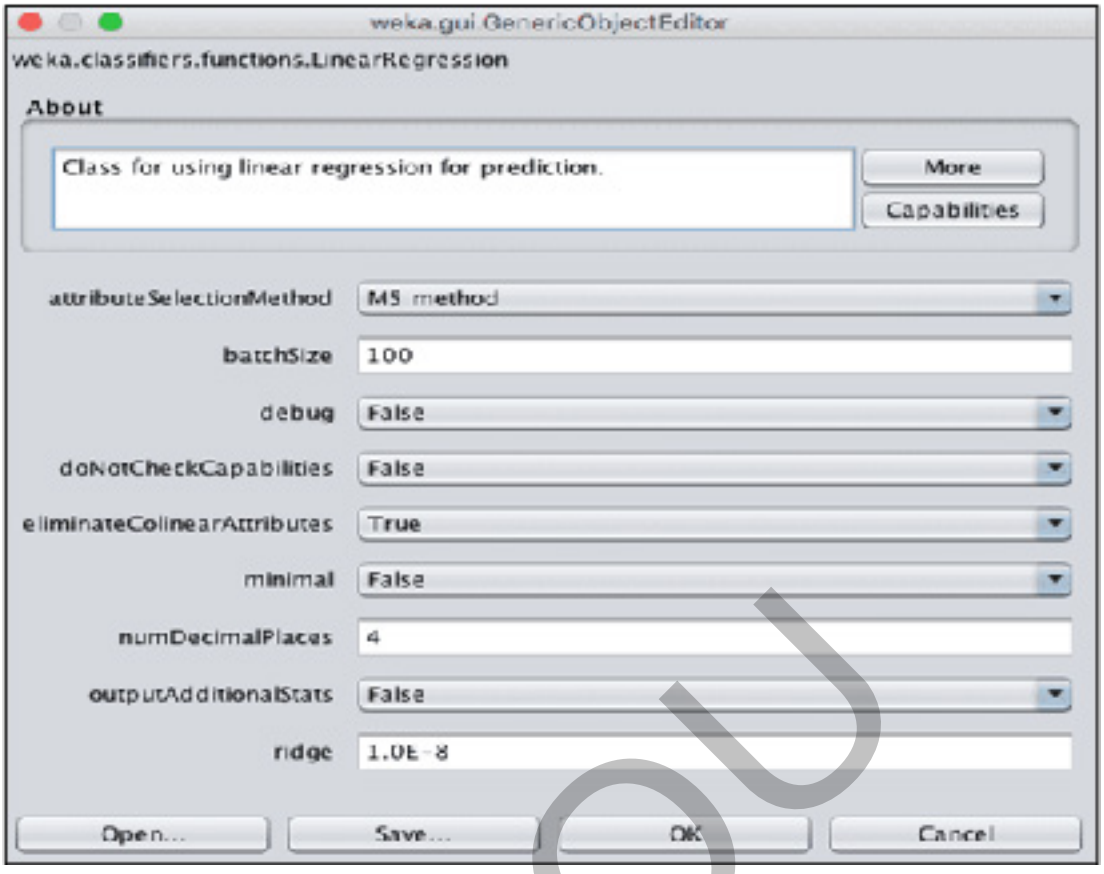


Fig. 4.2.24 Weka Configuration of Linear Regression

Step 4: Choose the Target Variable

- In the Classify tab, select the attribute to be predicted (the dependent variable) from the "Choose class attribute" dropdown list.

Step 5: Set Parameters (Optional)

Click on LinearRegression to open its settings. You can:

- Enable or disable attribute selection methods.
- Use ridge regression to handle multicollinearity.
- Set debugging and output options if needed.

Step 6: Select Evaluation Mode

Choose how to test the model:

- **Use training set:** Trains and tests on the same data.
- **Percentage split:** Splits data into training and testing sets (e.g., 70% training, 30% testing).
- **Cross-validation:** Divides data into folds (e.g., 10-fold) for better evaluation.

Step 7: Run the Algorithm

- Click Start to execute the Linear Regression model.
- WEKA will display the regression equation, coefficient values, and error measures (Figure 4.2.25).

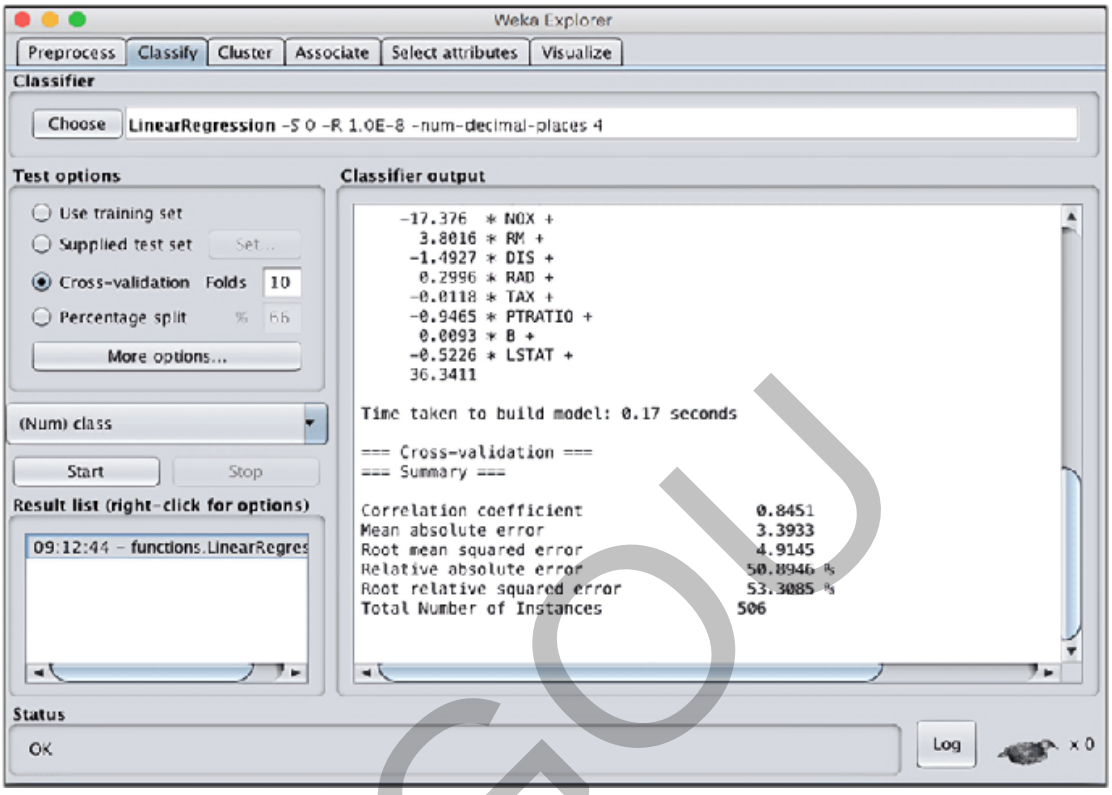


Fig. 4.2.25 Weka Results for Linear Regression

Step 8: View and Interpret Results

The output includes:

- **Regression Equation** showing how attributes relate to the target variable.
- **Correlation Coefficient (R):** Indicates the strength of the relationship.
- **Mean Absolute Error (MAE)** and **Root Mean Squared Error (RMSE):** Indicate prediction accuracy.

Linear Regression in WEKA provides a practical and visual approach to understanding how continuous values can be predicted using linear relationships. It helps learners observe the importance of variables, interpret coefficients, and evaluate model accuracy through built-in performance measures. This makes it a fundamental technique for predictive modeling and trend forecasting.

Recap

- WEKA is a powerful open-source data mining tool that supports various machine learning algorithms for classification, clustering, and regression.
- The Decision Tree Classifier uses a tree-like structure to make decisions based on attribute values.
- Information Gain, Gain Ratio, and Gini Index are commonly used criteria for splitting nodes in a Decision Tree.
- Common Decision Tree algorithms include ID3, C4.5, and CART.
- Naïve Bayes Classifier is a probabilistic model based on Bayes' Theorem, assuming independence among features.
- It is widely used for text classification, spam filtering, and sentiment analysis.
- K-Means Clustering is an unsupervised learning algorithm that groups data into K clusters based on similarity.
- It works by assigning data points to the nearest cluster centroid and recalculating centroids until stability.
- Agglomerative Clustering is a hierarchical clustering method that merges smaller clusters into larger ones in a bottom-up approach.
- The clustering process can be visualized using a dendrogram, showing how clusters merge step by step.
- Linear Regression is a supervised learning algorithm used to predict continuous numerical values.
- It finds the best-fit line that represents the relationship between dependent and independent variables.
- WEKA provides easy implementation of all these algorithms through its Explorer interface, without the need for coding.
- Each algorithm can be evaluated in WEKA using modes like Use Training Set, Percentage Split, or Cross-Validation.
- Understanding and applying these models help in solving real-world problems such as prediction, classification, and pattern discovery across domains like business, healthcare, and education.

Objective Type Questions

1. WEKA stands for _____.
2. WEKA is primarily used for _____.

3. Which type of learning does the Decision Tree Classifier belong to?
4. In a Decision Tree, the topmost node is called the _____.
5. The ID3 algorithm uses which criterion to split the data?
6. The formula for calculating entropy is used in which classifier?
7. The Naïve Bayes Classifier is based on _____ theorem.
8. K-Means is an example of a _____ learning algorithm.
9. In K-Means, the value of K represents _____.
10. Which tab in WEKA allows users to preprocess data before analysis?
11. The process of updating centroids in K-Means continues until _____.
12. Agglomerative Clustering follows a _____ approach.
13. Which file format does WEKA used?
14. The visual representation of hierarchical clustering is called a _____.
15. In Agglomerative Clustering, merging is based on _____ measures.
16. Linear Regression is used to predict _____ variables.
17. The equation of simple linear regression is _____.
18. Which evaluation method in WEKA divides data into training and testing sets?
19. Which WEKA tab is used to apply machine learning algorithms?
20. Which algorithm in WEKA can be used for both simple and multiple regression models?

Answers to Objective Type Questions

1. Waikato Environment for Knowledge Analysis
2. Data mining and machine learning tasks
3. Supervised learning
4. Root node

5. Information Gain
6. Decision Tree Classifier
7. Bayes' Theorem
8. Unsupervised
9. Number of clusters
10. Preprocess Tab
11. Cluster centroids no longer change (convergence)
12. Bottom-up (hierarchical)
13. ARFF
14. Dendrogram
15. Distance or similarity
16. Continuous (numerical)
17. Y = a + bX
18. Percentage Split
19. Classify tab
20. Linear Regression

Assignments

1. Explain the working principle of the Decision Tree Classifier. Demonstrate the implementation steps in WEKA using a sample dataset.
2. Implement a Naïve Bayes Classifier using WEKA. Explain its advantages and limitations with examples.
3. Perform K-Means Clustering in WEKA using a real-world dataset. Discuss the advantages and challenges of K-Means.
4. Describe the working principle and applications of the Agglomerative Clustering. Demonstrate the implementation steps in WEKA using a sample dataset.

5. Evaluate a Linear Regression model in WEKA. Discuss its advantages and limitations with examples.

Reference

1. Bhatia, P. (2019). *Beginning with WEKA and R Language: Data Mining and Data Warehousing* (1st ed.). Cambridge University Press.

2. Brownlee, J. (2016). *Machine Learning Mastery With Weka: Analyze Data, Implement Algorithms, and Understand Machine Learning* (1st ed.). Machine Learning Mastery.

3. Holmes, G., Donkin, A., & Witten, I. H. (2011). *WEKA: A Machine Learning Workbench* (2nd ed.). Springer.

4. Zaki, M. J., & Meira, W., Jr. (2020). *Data Mining and Machine Learning: Fundamental Concepts and Algorithms* (2nd ed.). Cambridge University Press.

5. Han, J., Pei, J., & Kamber, M. (2022). *Data Mining: Concepts and Techniques* (4th ed.). Morgan Kaufmann.

Suggested Reading

1. Official WEKA Documentation https://www.cs.waikato.ac.nz/ml/weka/

2. MachineLearningMastery–WEKATutorials https://machinelearningmastery.com/start-here/#weka

3. GeeksforGeeks – Data Mining and Machine Learning Concepts https://www.geeksforgeeks.org/machine-learning/

4. Towards Data Science (Medium) https://towardsdatascience.com/

5. KDnuggets – Data Science and Machine Learning Guide https://www.kdnuggets.com/

MODEL QUESTION PAPER SETS

SREENARAYANAGURU OPEN UNIVERSITY

MODEL QUESTION PAPER
SET - 01

QP CODE: ……

Reg. No: …….....
Name: …….....

THIRD SEMESTER FYUGP EXAMINATION

SKILL ENHANCEMENT COURSE

SGB24CA201SE - DATA ANALYTICS

Time: 2 Hours | Max Marks: 45

Section A

Answer any 5 questions. Each carries one mark

(5x1=5)

1. Which measure of central tendency is most affected by extreme values?
2. Which type of analytics is used to predict future trends?
3. What is the probability of drawing an ace from a standard deck of cards?
4. In which type of learning do clustering algorithms operate?
5. Which chart shows data as slices of a circle?
6. The function is used to find the number of characters in a string is _____
7. Which technique allows users to explore data progressively from summary to detail?
8. Which type of learning does the Decision Tree Classifier belong to?

Section B

Answer any 5 questions. Each carries two marks

(5x2=10)

9. Write a note on the measures of central tendency.
10. Explain the various phases of the Data Analytics Life Cycle?
11. Write about Venn diagram with the help of example.

12. Write about disjoint and non-disjoint events.

13. What is meant by quantitative data in visualization?

14. Define slicers in the context of dashboards.

15. What is univariate analysis? Give an example.

16. Give any 2 features of Weka.

Section C

Answer any 4 questions. Each carries five marks

(4x5=20)

17. Explain the concepts of Joint Probability and Marginal Probability.

18. Differentiate between structured and unstructured data with examples.

19. How does the visualization of numerical data differ from non-numerical data? Provide examples of each.

20. Explain about operators in R.

21. Describe the steps involved in building and evaluating a Decision Tree Classifier using WEKA.

22. Describe the stages of the Data Visualization Life Cycle.

Section D

Answer any 1 question. Each carries 10 mark

(1x10=10)

23. Explain the common methods used for the presentation of qualitative data?

24. Provide a detailed account of the K-Means Clustering and explain its implementation using the WEKA tool.

SREENARAYANAGURU OPEN UNIVERSITY

MODEL QUESTION PAPER
SET - 02

QP CODE: ……

Reg. No: …….....
Name: …….....

THIRD SEMESTER FYUGP EXAMINATION

SKILL ENHANCEMENT COURSE

SGB24CA201SE - DATA ANALYTICS

Time: 2 Hours Max Marks: 45

Section A

Answer any 5 questions. Each carries one mark

(5x1=5)

1. What is the difference between the maximum and minimum values in a data set called?
2. Which measure of central tendency represents the value that occurs most frequently in a dataset?
3. What is the range of possible values for a probability?
4. What does DBSCAN stand for?
5. Which chart type compares values across categories?
6. Which R package is widely used for data visualization?
7. Which KPI chart type is used to represent monthly sales revenue?
8. In K-Means, the value of K represents _____.

Section B

Answer any 5 questions. Each carries two marks

(5x2=10)

9. What are the concepts of shifting and scaling in data preprocessing?
10. Differentiate between structured and unstructured data with examples.
11. Write about dependent and independent events.

12. Write about the normal distribution and formula for it.

13. State two advantages of using data visualization in data analysis.

14. What is R programming mainly used for in data analysis?

15. What is the use of slicers in data visualization tools?

16. What is Agglomeration Clustering?

Section C

Answer any 4 questions. Each carries five marks

(4x5=20)

17. Explain the measures of central tendency.

18. Compare and contrast linear regression, Gaussian regression, and polynomial regression. Discuss the strengths and limitations of each type of regression.

19. Explain the purpose of data visualization?

20. Implement a Naïve Bayes Classifier using WEKA. Explain its advantages and limitations.

21. Describe how mathematical operations can be performed on vectors and matrices in R with examples.

22. Define Drill Down and Drill Through techniques. How do these features enhance interactivity and depth in data visualization?

Section D

Answer any 1 question. Each carries 10 mark

(1x10=10)

23. Explain about
    a. Joint probability
    b. Bayes Theorem

24. Discuss the concept of Density-Based Clustering using DBSCAN. Explain how it identifies clusters and handles noise points with an example.

സർവ്വകലാശാലാഗീതം

വിദ്യയാൽ സ്വതന്ത്രരാകണം
വിശ്വപൗരരായി മാറണം
ഗ്രഹപ്രസാദമായ് വിളങ്ങണം
ഗുരുപ്രകാശമേ നയിക്കണേ

കൂരിരുട്ടിൽ നിന്നു ഞങ്ങളെ
സൂര്യവീഥിയിൽ തെളിക്കണം
സ്നേഹദീപ്തിയായ് വിളങ്ങണം
നീതിവൈജയന്തി പാറണം

ശാസ്ത്രവ്യാപ്തിയെന്നുമേകണം
ജാതിഭേദമാകെ മാറണം
ബോധരശ്മിയിൽ തിളങ്ങുവാൻ
ജ്ഞാനകേന്ദ്രമേ ജ്വലിക്കണേ

കുരീപ്പുഴ ശ്രീകുമാർ

SREENARAYANAGURU OPEN UNIVERSITY

Regional Centres

Kozhikode
Govt. Arts and Science College
Meenchantha, Kozhikode,
Kerala, Pin: 673002
Ph: 04952920228
email: rckdirector@sgou.ac.in

Thalassery
Govt. Brennen College
Dharmadam, Thalassery,
Kannur, Pin: 670106
Ph: 04902990494
email: rctdirector@sgou.ac.in

Tripunithura
Govt. College
Tripunithura, Ernakulam,
Kerala, Pin: 682301
Ph: 04842927436
email: rcedirector@sgou.ac.in

Pattambi
Sree Neelakanta Govt. Sanskrit College
Pattambi, Palakkad,
Kerala, Pin: 679303
Ph: 04662912009
email: rcpdirector@sgou.ac.in

DON'T LET IT
BE TOO LATE

SAY NO TO DRUGS

LOVE YOURSELF
AND ALWAYS BE
HEALTHY

SREENARAYANAGURU
OPEN UNIVERSITY

SREENARAYANAGURU OPEN UNIVERSITY

The State University for Education, Training and Research in Blended Format, Kerala

Data Analytics

COURSE CODE: SGB24CA201SE

SGOU

Sreenarayanaguru Open University
Kollam, Kerala Pin- 691601, email: info@sgou.ac.in, www.sgou.ac.in Ph: +91 474 2966841

ISBN 978-81-987966-4-6