MGMUNIVERSITY

Practical-1

Aim - Introduction to R-tool, Install R-programming environment and basic packages

Theory →

R-tool is a specialized software tool designed for various data analysis, statistical computing and graphical representation tasks. It is widely used in research, academia, and industry due to its flexibility and comprehensive range of functionalities

Key Features

i) Data Manipulation

R tool excels in handling large datasets, providing numerous functions for data cleaning, transformation and summarization

2) Statistical Analysis

It offers a vast array of statistical tests and models, making it suitable for both simple descriptive statistics and complex inferential analyses

3) Visualization

With powerful libraries like ggplot2, R tool allows for creating detailed and customizable visualizations including charts, graphs and plots.

4) Programming capabilities :

R tool uses the R programming lang, which is known for its simplicity and effectiveness in data manipulation & analysis

5) Extensibility

The tool supports numerous packages & libraries, enhancing numerous packages & libraries its capabilities to various domai

such as bioinformatics econometrics and social sciences

6) Integration

R tool can be integrated with other software and platforms, including databases, web services, and data visualization tools, facilitating seamless workflows.

Getting started with R

To start using R tool, one typically needs to

1) Install R - Download and install the R programming lang from the CRAN

2) Install RStudio - RStudio is a popular IDE for R that provides a user friendly interface

3 Learn the basic → R syntax, basic functions and data structures

4) Explore Packages - Install & explore various R packages that suit specific needs

Example workflow -

1] Data Import - Load data from CSV, Excel or databases [using data()]

2] Data cleaning - Handle missing values, filter and transform data

3] Statistical Analysis - Perform descriptive statistics and apply statistical models

4] Visualization - Creates plots and charts to visualize data insights

5] Reporting - Compile results into reports or dashboards

Difference b/w R & Python

Criteria	R	Python
Ease of learning	steeper learning curve	easier for beginners
Primary use	statistical analysis, data visualization	General purpose, data science, web dev
Libraries	ggplot2, dplyr tidyr	pandas, Numpy, scikit-learn TensorFlow
Community support	strong in academia	Large & diverse
Data Handling	Excellent with dplyr data.table	strong with pandas, Numpy
Visualization	Adv. with ggplot2 lattice	Robust with Matplotlib seaborn, Plotly
machine learning	Good for statistics	strong with scikit-learn TensorFlow

Criteria	R	Python
Integration	Good with statistical softwares	Excellent with various databases, services
Performance	Efficient for stats, slower for large scale	Fast with optimized libraries
IDEs	RStudio	Jupyter, Pycharm, Vs Code
Flexibility	specialized for stats	Versatile for many applications
Industry Popularity	Popular in academia	Widely adopted in Industry

Conclusion - In this practical we install R tool and Rstudio learnt about different basic packages, and key features of R and differentiating points of R and Python.

Shanmuk 19/8/2024

MGMUNIVERSITY

Practical - 2

Aim - Implement different data types & data structures in R (Vectors, Lists, Matrices, Array, factors, Data Frames)

Theory -

R offers a variety of data structures, each suited for different types of data manipulation and analysis tasks.

Here are the main data structure in R, along with their syntax and examples

1] Vectors

A seq. of elements of the same type Cnumeric, character, logical etc.
Vectors are one-dimensional data structures

Syntax -

Vector-name <- c (element1, 2, 3 ...)

Example -

```
num_vec <- c(1, 2, 3, 4)
char_vec <- c("apple", "banana", "cherry")
logical <- c(TRUE, FALSE, FALSE)
```

2] Matrices

A matrix is a rectangular arrangement of numbers in rows and colums. In a matrix as we know rows are the ones that run horizontally and columns vertically.
Matrices are 2D homogenous data structu

To create a matrix R, we need a function called matrix()

```
matrix_name <- matrix(data, nrow=3, ncol=3)
m <- matrix(1:9, nrow = 3, ncol = 3)
```

Data Types in R

A data types are used to specify the kind of data that can be stored in a variable

1] Numeric - (3, 6.7, 7, 121)

2] Integer - (2L, 42L ; where L declares this is an integer

3] logical - ("True")

4] Complex - (7+5i ; where 'i' is an imaginary numbers)

5] character - ("a" , "B" , 'c' is third, "69")

6] Raw - (as. raw (55); raw creates a raw vector of a specified length)

3] Arrays

Arrays are the R data objects which store the data in more than 2 dimensions Arrays are n-d data structures.
For ex, if we create an array of dimentions (2, 3, 3) then it creates 3 rectangular matrices with 2 rows & 3 columns Homogeneous data structures

array_name ← array(data, dim = c(dim1, dim2))

a ← array (1:12, dim = c(2,3,2))

4] Data Frames

Generic data objects of R which are used to store the tabular data.
Data frames are the foremost popular data objects in R programming because we are comfortable in seeing the data within the tabular form. They are 2D hetero-geneous data structures, These are lists of vectors of equal lengths

- A dataframe must have column names and every row should have a unique names ~~and~~
- Each column must have the identical number of items
- Each item in a single column must be of the same data type
- Different columns may have diff data types

df <— data.frame (col1 = data1, col2 = data2)

```
df <- data.frame (
      id = 1:3,
      name = c ("John", "Yashi"),
      age = c (28, 34, 23)
```

5] Lists

A lists is a generic object consisting of a ordered collection of object. lists are heterogeneous data structures. There are ID. A list can be a list of vectors, matrices, characters and functions

list_name <- list (element 1, 2, 3 ...)

l <- list (vector = num ,
matrix = mat ,
dataframe = df ,
)

6) Factors

Used to categorise the data and store it as levels. They are useful for storing categorical data. They can store both strings and integers.
They are useful to categorize unique values in columns like "TRUE" or "False",

factor_name <- factor (vector)

factor_example <- factor (c ("low", "high"))

Conclusion - In this practical we learnt about different data types and data structures in R and implemented them in R Studio.

Shanmuka 19/7/2024

MGMUNIVERSITY

Practical - 3

Aim - Program for basic statistics in R (mean, median, mode, variance, standard deviation, range summary)

Theory -

- Mean

The mean is the average of a set of numbers. It is calculated by adding up all the numbers in the dataset and then dividing by the number of values

$$\text{Mean}(\mu) = \frac{\sum_{i=1}^{n} x_i}{n}$$

```
x <- c(1,2,3. . . .) # dataset
mean.result = mean(x)
print(mean.result)
```

O/P → 2.8

Median

The median is the middle value of a dataset when it is ordered in ascending or descending order. If the data set has an even number of observations, the median is the avg of the 2 middle numbers.

```
median.result = median(x)
print(median.result)
```

o/p 2.5

Mode

The mode is the value that appears most frequently in a dataset. A dataset may have one mode, more than one mode, or no mode at all

```
mode <- function(x)
    ux <- unique(x)
ux[which.max(tabulate(match(x, ux)))]
mode.result = mode(x)
print(mode.result)
```

Variance

Variance measures the spread of a set of numbers. It is the avg of the squared differences from mean

$$\text{Variance } (\sigma^2) = \frac{\sum_{i=1}^{n} (x_i - \mu)^2}{n-1}$$

```
variance.result = var(x)
print(variance.result)
2.484211
```

Standard Deviation

The standard deviation is the squared root of the variance. It provides a measure of the average distance of each point from the mean.

$$\text{Standard deviation } (\sigma) = \sqrt{\text{variance}}$$

```
std_deviation_value <- sd(data)
sd.result = sqrt(var(x))
print(sd.result)
[1] 1.576138
```

Range

The difference between the max & min value in a data set

```
range.result <- range(x)
range.diff <- diff(range.result)
print(range.diff)
```

Summary

Provides a quick overview of the dataset, including the minimum, 1st quartile (25th percentile) median (50th percentile), mean 3rd quartile (75th percentile) and maximum values. This helps understand the distribution of data

```
summary.result <- summary(x)
print(summary.result)
```

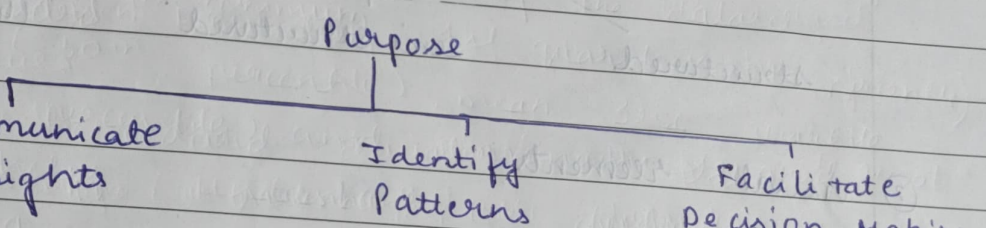
```
Min.  1st Qu.  Median  Mean  3rd Qu.  Max.
1.000  1.000   2.500   2.300  3.500   6.000
```

Practical - 4

Aim - Program for data visualization in R

Theory -

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs and maps, data visualization tools provide an accessible way to see and understand trends, outliers and patterns in data



Types of data visualization

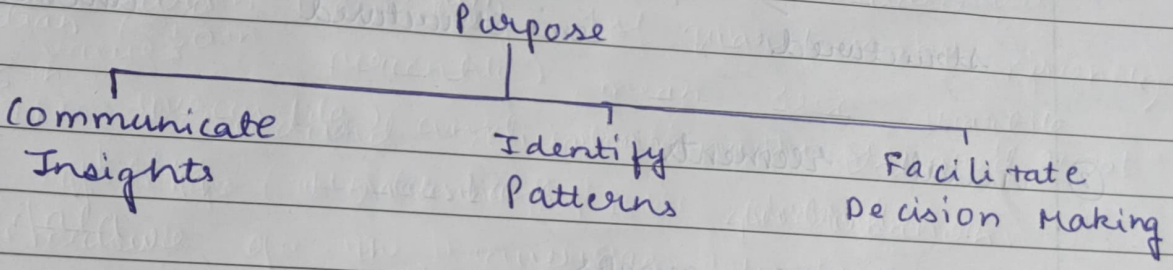
- ~~Dia~~ Charts
- Maps
- Graphs
- Plots

Practical - 4

Aim - Program for data visualization in R

Theory -

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs and maps, data visualization tools provide an accessible way to see and understand trends, outliers and patterns in data



Types of data visualization

- ~~Dia~~ Charts
- Maps
- Graphs
- Plots

Components

Data

Visual Elements

Axes & Scales

Legends and Labels

① Data

- Source : Origin of data, such as databases, CSV files, API or real time streams

Structured

Semi Structured

Unstructured

② Visual Elements

Charts and graphs

③ Axes & Scales

Define the coordinate system and scale for scale for representing data values

(4) Lables & Titles

Text elements that describe and provide context for the visual elements, including axis, lables, chart titles and data lables

1) Histogram -

Display ~~distribution~~ distribution of a single continuous variable. It divides the data into bines (intervals) and shows the frequency (count) of data points in each bin

2) Scatter plot

shows the relationship b/w 2 variables (continuous). Each point represents an observation in the dataset with its position determined by the values of 2 variables

3) Line Chart

A line chart is used to display data points connected by straight lines. It is useful for visualizing data trends over time

or a sequence

4) Pie Chart

A pie chart represents the proportion of diff categories as slices a circle. It is useful for displaying the relative size of parts to a whole

5) Box Plot

A box plot display the distribution of data based on a five number summary: min, first quartile median, 3rd quartile & max.
It is useful for identifying outliners and comparing distributions.

6) Density Plot

A density plot shows the distribution of a continuous variable, smoothing out the histogram
It estimates the probability density func of the variable.

7) Normal QQ Plot

A normal QQ Plot (quantile - quantile) plot compares the distribution of a dataset to a normal distribution.
It helps in assessing whether the data follows a normal distribution.

Conclusion

Data visualization is an essential tool in data analysis, enabling the clear & effective communication of data insights. By using different types of plots, we can uncover patterns, trends, and outliers in the data facilitating better decision-making and understanding of the data.