Vidyavardhini's
College of Engineering & Technology
Vasai Road (W)

Department of Artificial Intelligence & Data Science

Laboratory Manual
Student Copy

Semester	VI	Class	T.E
Course Code	CSL601		
Course Name	Data Analytics and Visualization Lab		

Vidyavardhini's College of Engineering & Technology

Vision

To be a premier institution of technical education; always aiming at becoming a valuable resource for industry and society.

Mission

- To provide technologically inspiring environment for learning.
- To promote creativity, innovation and professional activities.
- To inculcate ethical and moral values.
- To cater personal, professional and societal needs through quality education.

Department Vision:

To foster proficient artificial intelligence and data science professionals, making remarkable contributions to industry and society.

Department Mission:

- To encourage innovation and creativity with rational thinking for solving the challenges in emerging areas.
- To inculcate standard industrial practices and security norms while dealing with Data.
- To develop sustainable Artificial Intelligence systems for the benefit of various sectors.

Program Specific Outcomes (PSOs):

PSO1: Analyze the current trends in the field of Artificial Intelligence & Data Science and convey their finding by presenting / publishing at a national / international forums.

PSO2: Design and develop Artificial Intelligence & Data Science based solutions and applications for the problems in the different domains catering to industry and society.

Program Outcomes (POs):

Engineering Graduates will be able to:

- **PO1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

- **PO2. Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

- **PO3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

- **PO4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

- **PO5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

- **PO6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

- **PO7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

- **PO8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

- **PO9. Individual and teamwork:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

- **PO10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

- **PO11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

- **PO12. Life-long learning:** Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Course Objectives

1	To effectively use graph libraries such as matplotlib/seaborn/excel plots
2	To perform exploratory data analysis on a given data set
3	To fit a statistical model (Regression, ANOVA, ARIMA) on a given data set
4	To apply suitable visualization techniques for identifying patterns, trends and outliers in large data sets.

Course Outcomes

CO	Students will be able to	Action verbs	Bloom's Level
CSL601.1	Use graph libraries such as matplotlib/Seaborn/Excel plots.	Identify	Apply (Level 3)
CSL601.2	Perform exploratory data analysis and prepare the data for fitting a model	Apply	Apply (Level 3)
CSL601.3	Build a statistical model (Regression, ANOVA, ARIMA) on a given data set	Apply	Apply (Level 3)
CSL601.4	Apply suitable visualization techniques to get insights from a given data set	Implement	Apply (Level 3)

Mapping of Experiments with Course Outcomes

List of Experiments	CSL601.1	CSL601.2	CSL601.3	CSL601.4
Getting introduced to data analytics libraries in Python and R.	3	-	-	-
Implement R Program for Simple Linear Regression.	-	3	-	-
Implement R Program for Multiple Linear Regression	-	3	-	-
Time Series Analysis in Python/R.	-	-	3	-
Implementation of ARIMA model in python / R.	-	-	3	-
Text analytics: Implementation of Spam filter/Sentiment analysis in python/R.	-	-	3	-
Data Visualization: Use different R libraries for data visualization.	-	-	-	3
Data Visualization: Use different Python libraries for visualization.	-	-	-	3

List of Experiments

Sr. No.	Name of Experiment	DOP	DOC	Marks	Sign
	Basic Experiments				
1	Getting introduced to data analytics libraries in Python and R.				
2	Implement R Program for Simple Linear Regression.				
3	Implement R Program for Multiple Linear Regression				
4	Time Series Analysis in Python/R.				
5	Implementation of ARIMA model in python / R.				
6	Text analytics: Implementation of Spam filter/Sentiment analysis in python/R.				
7	Data Visualization: Use different R libraries for data visualization.				
8	Data Visualization: Use different Python libraries for visualization.				
	Project / Assignment				
9	Assignment 1 Introduction to data analytics				
10	Assignment 2 Regression				
11	Assignment 3 Time Series				
12	Assignment 4 Text Mining				
13	Assignment 5 Data visualization using R				
14	Assignment 6 Data visualization using Python				

Formative Assessment					
15	Th - Quiz 1 Introduction to data analytics				
16	Th - Quiz 2 Regression				
17	Th - Quiz 3 Time Seies				
18	Th - Quiz 4 Text Mining				
19	Th - Quiz 5: Data visualization using R				
20	Th - Quiz 6: Data visualization using Python				
21	Pr - Quiz 1: Graph Libraries				
22	Pr - Quiz 2: Exploratory data analysis				
23	Pr - Quiz 3: Regression				
24	Pr - Quiz 4: Text Mining				
25	Pr - Quiz 5: Data visualization using R				
26	Pr - Quiz 6: Data visualization using Python				

D.O.P: Date of performance

D.O.C : Date of correction

Experiment No.1
Getting introduced to data analytics libraries in Python and R.
Date of Performance:
Date of Submission:

Experiment No. **1**

Aim: Introduction to Data analytics libraries in Python and R.

Objective- Understand the use of Python and R, To effectively use libraries for data science.

Description:

Why Choose Python?

Python is a general-purpose, open-source programming language used in various software domains, including data science, web development, and gaming.

Launched in 1991, Python is one of the most popular programming languages in the world, occupying the top position in several programming language popularity indices, such as the TIOBE Index and the PYPL Index.

One of the reasons for the worldwide popularity of Python is its community of users. Python is backed by a vast community of users and developers who ensure the smooth growth and improvement of the language, as well as the continuous release of new libraries designed for all kinds of purposes.

Python is an easy language to read and write due to its high similarity with human language. In fact, high readability and interpretability are at the heart of the design of Python. For these reasons, Python is often cited as a go-to programming language for newcomers with no coding experience.

Over time, Python has been gaining popularity in the field of data science thanks to its simplicity and the endless possibilities provided by the hundreds of specialized libraries and packages that support any kind of data science task, such as data visualization, machine learning, and deep learning.

Why Choose R?

R is an open-source programming language specifically created for statistical computing and graphics.

Since its first launch in 1992, R has been widely adopted in scientific research and academia. Today, it remains one of the most popular analytics tools used in both traditional data analytics and the

rapidly-evolving field of business analytics. It ranks 11th and 7th position in the TIOBE Index and the PYPL Index, respectively.

Designed with statisticians in mind, with R, you can use complex functions within a few lines of code. All kinds of statistical tests and models are readily available and easily used, such as linear modeling, non-linear modeling, classifications, and clustering.

The extensive possibilities **R** offers are mostly due to its huge community. It has developed one of the richest collections of data-science-related packages. All of them are available via the Comprehensive **R** Archive Network **(CRAN).**

Another feature that makes **R** particularly remarkable is the power to generate quality reports with support for data visualization and its available frameworks to create interactive web applications. In this sense, **R** is widely considered the best tool for making beautiful graphs and visualizations.

R vs Python: Key Differences

Purpose

While Python and **R** were created with different purposes -Python as a general-purpose programming language and **R** for statistical analysis-nowadays , both are suitable for any data science task. However, Python is considered a more versatile programming language than R, as it's also extremely popular in other software domains, such as software development, web development , and gaming.

Type of Users

As a general-purpose programming language , Python is the standard go-to choice for software developers breaking into data science. Plus, Python 's focus on productivity makes it a more suitable tool to build complex applications.

By contrast, **R** is widely used in academia and certain sectors, such as finance and pharmaceuticals. It is the perfect language for statisticians and researchers with limited programming skills.

Learning curve

Python 's intuitive syntax is considered one of the closest programming languages to English. This makes it a very good language for new programmers , with a smooth and linear learning curve. Although **R** is designed to run basic data analysis easily and within minutes, things get harder with complex tasks, and it takes more time for **R** users to master the language.

Overall, Python is considered a good language for beginner programmers. **R** is easier to learn when you start out, but the intricacies of advanced functionalities make it more difficult to develop expertise.

Popularity

Although new programming languages, like **Julia** **,** are recently gaining momentum in data science, Python and **R** remain the absolute kings in the discipline.

However, in terms of popularity -always a very slippery concept- the differences are striking. Python has consistently outranked R, especially in recent years. Python ranks first in several programming language.

Common Libraries

Both Python and R have robust and extensive ecosystems of packages and libraries specifically designed for data science. Most packages in Python are hosted in the Python Package Index **(PyPi)**, whereas **R** packages are normally stored in the Comprehensive R Archive Network (CRAN).

Below you can find a list of some of the most popular data science libraries in R and

Python. R packages: **dplyr:** It is a data manipulation library for R.

tidyr: a great package that will help you get your data clean and tidy.

22plot2: the perfect library for visualizing data.

Shiny: It is the ideal tool for creating interactive web apps directly from R.

Caret: one of the most important libraries for machine learning in R.

Python packages:

NumPy: provides a large collection of functions for scientific computing.

Pandas: perfect for data manipulation.

Matplotlib: the standard library for data visualization.

Scikit-learn: is a library in Python that provides many machine learning algorithms.

TensorFlow: a widely used framework for deep learning.

Common IDEs

An IDE, or Integrated Development Environment , enables programmers to consolidate the different aspects of writing a computer program. They are powerful interfaces with integrated capabilities that allow developers to write code more efficiently.

In Python, the most popular IDEs in data science are Jupyter Notebooks and its modem version, JupyterLab, as well as Spyder.

Attach Libraries you searched in Lab session-

Conclusion-

Experiment No.2
Implement R Program for Simple Linear Regression
Date of Performance:
Date of Submission:

EXPERIMENT NO 2

Aim: To write the implementation of linear regression.

Objective:- To understand the use of simple linear regression techniques by implementing user define dataset and importing dataset

Description:

Regression analysis is a very widely used statistical tool to establish a relationship model between two variables. One of these variables is called a predictor variable whose value is gathered through experiments. The other variable is called response variable whose value is derived from the predictor variable.

In Linear Regression these two variables are related through an equation, where the exponent (power) of both these variables is 1. Mathematically a linear relationship represents a straight line when plotted as a graph. A non-linear relationship where the exponent of any variable is not equal to 1 creates a curve.

The general mathematical equation f or a linear regression is

$y = ax + b$

Following is the description of the parameters used -

y is the response variable. x is the predictor variable.

a and b are constants which are called the coefficients.

Procedure:

The steps to create the relationship is

1. Carry out the experiment of gathering a sample of observed values of height and corresponding weights.
2. Create a relationship model using the **lm()** functions in R.
3. Find the coefficients from the model created and create the mathematical equation using these Get a summary of the relationship model to know the average error in prediction. Also called **residuals.**

To predict the weight of new persons, use the **predict()** function in R.

Program:

OUTPUT:

Conclusion:

1. Function used for linear regression in R is

2. Explain use of summary().

Experiment No.3
Implement R Program for Multiple Linear Regression.

Date of Performance:
Date of Submission:

Experiment No - 3

Aim :- Implement R Program for Multiple Linear Regression.

Objective:- To understand the use of Multiple linear regression techniques by implementing a predefined dataset of R Studio.

Description-

Multiple linear regression is the extension of linear regression in the relationship between more than two variables. In simple linear regression, we have one predictor and one response variable. But in multiple regressions, we have more than one predictor variable and one response variable.

There is the following general mathematical equation for multiple regression -

$$y=b_0+b_1*x_1+b_2*x_2+b_3*x_3+\cdots b_n*x_n$$

Here,

- y is a response variable.
- **bO, bl, b2...bn** are the coefficients.
- **xl, x2, ...xn** are the predictor variables.

In R, we create the regression model with the help of the **lm()** function. The model will determine the value of the coefficients with the help of the input data. We can predict the value of the response variable for the set of predictor variables using these coefficients.

There is the following syntax of lm() function in multiple regression

1. lm(y x l+x2+x3 ...., data)

Before proceeding further, we first create our data for multiple regression. We will use the "mtcars" dataset present in the R environment. The main task of the model is to create the relationship between the "mpg" as a response variable with "wt", "disp" and "hp" as predictor variables.

For this purpose, we will create a subset of these variables from the "mtcars" dataset.

```
1.    data<-mtcars[,c("mpg","wt","disp","hp")]
2.    print(head(input)
```

Program-

```
# Load required libraries
library(ggplot2)

# Sample dataset (for demonstration purposes)
# Let's create a dataset to predict 'y' based on two independent variables 'X1' and 'X2'
set.seed(42)  # For reproducibility
```

```r
data <- data.frame(
  X1 = rnorm(100, mean=5, sd=2),
  X2 = rnorm(100, mean=10, sd=3),
  y = rnorm(100, mean=50, sd=10)
)

# Create a multiple linear regression model
model <- lm(y ~ X1 + X2, data=data)

# Summary of the model
summary(model)

# Making predictions
data$predicted <- predict(model)

# Plotting the results
# Note: For visualizing 3D data we will use pairs plot for all variables
pairs(data, panel = panel.smooth)

# You can also visualize predictions vs actual values in a simpler 2D plot
ggplot(data, aes(x=predicted, y=y)) +
  geom_point(alpha=0.5, color='blue') +
  geom_abline(slope=1, intercept=0, color='red') +
  ggtitle('Actual vs Predicted Values') +
  xlab('Predicted Values') +
  ylab('Actual Values')
```

Output-

```
Call:

lm(formula = y ~ X1 + X2, data = data)


Residuals:

   Min     1Q Median     3Q    Max

-29.525 - 5.159  0.107  4.903  29.057


Coefficients:

            Estimate Std. Error t value Pr(>|t|)

(Intercept)    48.0514    1.5335  31.33   < 2e-16 ***

X1          -1.0470    0.7390  -1.42   0.157

X2           0.4730    0.4197   1.13   0.260

---

Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


Residual standard error: 10.88 on 97 degrees of freedom

Multiple R-squared:  0.04628,        Adjusted R-squared:  0.03392

F-statistic: 3.287 on 2 and 97 DF,  p-value: 0.06821
```

Conclusion-

The implementation of Multiple Linear Regression in R demonstrates how to model the relationship between a single dependent variable and multiple independent variables. By using the lm() function, we are able to estimate coefficients for each predictor, assess the overall model fit, and identify which variables significantly contribute to explaining variability in the dependent variable.

The summary output provides crucial statistics like the coefficients, significance levels, and R-squared values, which help in interpreting the model's effectiveness. Furthermore, visualization of actual versus predicted values aids in evaluating model performance and understanding prediction accuracy.

Multiple Linear Regression is a powerful technique that extends the capabilities of simple regression by allowing for the analysis of multiple factors simultaneously. It is widely used in fields such as economics, social sciences, and health research, making it an essential tool for data-driven decision-making and insight generation. Understanding how to implement and interpret multiple linear regression models is fundamental for researchers and analysts working with complex data sets.

Experiment No.4
Time Series Analysis in Python/R.
Date of Performance:
Date of Submission:

Experiment No- 4

Aim :- Implement Time Series Analysis for rainfall in R Programming. **Objective:-** To

understand the use of time series models for prediction.

Description:-

- Time series analysis is a specific way of analyzing a sequence of data points collected over an interval of time. In time series analysis, analysts record data points at consistent intervals over a set period of time rather than just recording the data points randomly.

The basic syntax for ts() function in time series analysis is -

- timeseries.object.name <- ts(data, start, end, frequency)

Following is the description of the parameters used -

- data is a vector or matrix containing the values used in the time series.
- start specifies the start time for the first observation in time series.
- end specifies the end time for the last observation in time series.
- frequency specifies the number of observations per unit time.

- Except the parameter "data" all other parameters are optional.

Different Time Intervals

The value of the frequency parameter in the ts() function decides the time intervals at which the data points are measured. A value of 12 indicates that the time series is for 12 months. Other values and its meaning is as below -

- frequency = 12 pegs the data points for every month of a year.
- frequency = 4 pegs the data points for every quarter of a year.
- frequency = 6 pegs the data points for every 10 minutes of an hour.
- frequency = 24*6 pegs the data points for every 10 minutes of a day.

Program :-

Output:

Conclusion :-

1. An orderly set of data arranged in accordance with their time of occurrence is called

2. The graph of time series is called

3. Use of Matrix()-

Experiment No.5
Implementation of ARIMA model in python / R.
Date of Performance:
Date of Submission:

Experiment No 5

Aim- Implementation of ARIMA model in **R** Programming.

Objective- To Understand use of Auto-Regression Integrated Moving Average Time Series Model

Theory-

In R programming , data analysis and visualization is so easy to learn the behavior of the data. Moreover, the **R** language is used mostly in the data science field after Python. Time series analysis is a type of analysis of data used to check the behavior of data over a period of time. The data is collected over time sequentially by the **ts()** function along with some parameters. It helps in analyzing the pattern of the data over a graph. There are many techniques used to forecast the time series object over the plot graph but the **ARIMA model** is the most widely used approach out of them.

Time Series Forecasting

Time series forecasting is a process of predicting future values with the help of some statistical tools and methods used on a data set with historical data. Some of the applications of time series forecasting are:

- Predicting stock prices
- Forecast weather
- Forecast the sales of a product

ARIMA model

ARIMA stands for AutoRegressive Integrated Moving Average and is specified by three order parameters: *(p, d, q).*

- **AR(p) Autoregression:** A regression model that utilizes the dependent relationship between a current observation and observations over a previous period.An auto regressive *(AR(p))* component refers to the use of past values in the regression equation for the time series.
- **I(d) Integration:** Uses differencing of observations (subtracting an observation from observation at the previous time step) in order to make the time series stationary. Differencing involves the subtraction of the current values of a series with its previous values d number of times.

1. Load the data set after installing the package forecast.
2. The Steps of Pre-processing are done, which creates a separate time-series or timestamp.
3. Making Time-series stationary and check the required transformations.
4. The difference value 'd' will be performed.
5. The core important step in ARIMA is plotting ACF and PACF.
6. Determine the two parameters p and q from the plots.
7. The previously created value fits the Aroma model and predicts the future values. The Fitting Process is also named as Box-Jenkins Method.

8. Doing Validation.auto. Arima() function is used for automatic prediction and ARIMA Models. This function uses unit root tests, minimization of the AIC and MLE to obtain an ARIMA model.

To make the series stationary, we need to differentiate a previous value from the current value.

d= pval-cval, if the value is already stationary the d=O.

predict() - Used to predict the model based on the results of the various fitting model used.

Implementation of ARIMA model in R

Conclusion:-

1. The difference between the actual value of the time series and the forecasted value called-

2. Use of ARIMA ()-

3. Use of forecast()-

Experiment No.6
Text analytics: Implementation of Spam filter/Sentiment analysis in python/R.
Date of Performance:
Date of Submission:

EXPERIMENT NO 6

Aim: Implementation of Sentiment Analysis

Objective:- To understand the use of various sentiment Analysis techniques by implementing them

Description:

Sentiment Analysis: Sentiment Analysis is a text analysis technique that allows companies to make sense of qualitative data. By detecting positive and negative sentiment in text data, such as tweets, product reviews, and support tickets, you can understand how customers feel about your brand, product, or service, and gain insights that lead to data-driven decisions

Sentiment Analysis deals with analyzing emotions and the perspective of a speaker or an author from a given piece of text. "Sentiment analysis or opinion mining refers to the appliance of language process, linguistics, and text analytics to spot and extract subjective information in supply materials". This field of technology deals with analyzing and predicting the hidden information keep within the text. This hidden information gives valuable insights regarding user's intentions, style and odds. Sentiment Analysis specializes in categorizing the text at the extent of subjective and objective nature. Judgement indicates that the text bears opinion content where's perspicacity indicates that the text is while not opinion content

Some examples-

1. **Subjective-** This motion picture by tom cruise and Angelina jolie is great. (This sentence has an opinion, it talks regarding the motion picture and also the writer's emotions regarding same "great" and thence its subjective

2. **Objective-** This motion picture stars tom cruise and Angelina.

(This sentence may be a reality, general information instead of an opinion or a read of some individual and thence its objective)

The subjective text may be additional categorized into three broad classes supported the emotions expressed within the text.

1. Positive- I like to look at Star series.

2. Negative- The movie was awful.

3. Neutral- I typically get hungry by evening. (This sentence has users views, emotions hence it's subjective however because it doesn't have any positive or negative polarity therefore it's neutral).

Example:

Sentiment Analysis

I I

POSITIVE

"Great service for an

afforda ble price.

We will definitel y be booki ng agian "

NEUTRAL

"Just booked two nights at the hotel
"

NEGATIVE

"Horrible services. The room

was dirty and u n pleasant . Not worth the money "

Method 1: Using Positive and Negative Word Count - With Normalization for Calculating Sentiment Score

In this method, we will calculate the Sentiment Scores by classifying and counting the Negative and Positive words from the given text and taking the ratio of the difference of Positive and Negative Word Counts and Total Word Count. We will be using the Amazon Cell Phone Reviews dataset from Ka le.

Code:

Output:

Method 2: Using Positive and Negative Word Counts - With Semi Normalization to calculate Sentiment Score

In this method, we calculate the sentiment score by evaluating the ratio of Count of Positive Words and Count of Negative Words + 1. Since there is no difference of values involved, the sentiment value will always be more than 0. Also, adding 1 in the denominator would save from Zero Division Error. Let's start with the implementation. The implementation code will remain the same till the Negative and Positive Word Count from the previous part with a difference that this time we don't need the total word count, thus will be omitting that part.

Code:

Output:

Conclusion-

Techniques used for sentimental analysis are

Experiment No.7

Data Visualization: Use different R libraries for data visualization
Date of Performance:
Date of Submission:

Experiment No - 7

Aim - Data Visualization: Data Visualization: Use different R libraries for data visualization.

Objective:- To understand and apply the R libraries for visualization using python.

Description:-

1. ggplot2 ggplot2 is an R data visualization library that is based on The Grammar of Graphics. ggplot2 can create data visualizations such as bar charts, pie charts, histograms, scatterplots, error charts, etc. using highlevel API. It also allows you to add different types of data visualization components or layers in a single visualization. Once ggplot2 has been told which variables to map to which aesthetics in the plot, it does the rest of the work so that the user can focus on interpreting the visualizations and take less time in creating them. But this also means that it is not possible to create highly customized graphics in ggplot2. But there are a lot of resources in the RStudio community and Stack Overflow which can provide help in ggplot2 when needed. Just like dplyr, if you want to install ggplot2, you can install the tidyverse or you can just install ggplot2 using install.packages("ggplot2")

2. Plotly

Plotly is a free open-source graphing library that can be used to form data visualizations. Plotly is an R package that is built on top of the Plotly JavaScript library (plotly.js) and can be used to create web-based data visualizations that can be displayed in Jupyter notebooks or web applications using Dash or saved as individual HTML files. Plotly provides more than 40 unique chart types like scatter plots, histograms, line charts, bar charts, pie charts, error bars, box plots, multiple axes, sparklines, dendrograms, 3-D charts, etc. Plotly also provides contour plots, which are not that common in other data visualization libraries. In addition to all this, Plotly can be used offiine with no internet connection. You can install Plotly from CRAN using install.packages('plotly') or install the latest development version from GitHub using devtools:: install_github("ropensci/plotly").

3. Esquisse

Esquisse is a data visualization tool in R that allows you to create detailed data visualizations using the ggplot2 package. You can create all sorts of scatter plots, histograms , line charts, bar charts, pie charts, error bars, box plots, multiple axes, sparklines, dendrograms, 3-D charts, etc. using Esquisse and also export these graphs or access the code for creating these graphs. Esquisse is such a famous and easily used data visualization tool because of its drag and drops ability that makes it popular even among beginners. You can install Esquisse from CRAN using install.packages("esquisse ") or install the development version from GitHub using remotes: :install_github("dreamRs/esquisse ").

4. Lattice

Lattice is a data visualization tool that is primarily used to implement Trellis graphs in R. These Trellis graphs are used to view many complicated and multi-variable data sets at the same time so they can be compared. Since all these different plots end up looking like a Trellis, this is called a Trellis graph. Since Lattice is a high-level data visualization library, it can handle many of the typical graphics without needing many customizations. In case you want to extend the capabilities of Lattice, they can download the LatticeExtra package which is an extended version. You can install Lattice from CRAN using install.packages(
"lattice ") or install the development version from GitHub usmg remotes: :install_github("deepayan/lattice ").

5. RGL

The RGL package in R is created specifically for making 3-D data visualizations and data plots. It has many graphics commands that work in 3 dimensions but is modeled loosely after the classic 2-D graphics in R. RGL is also inspired by the grid package in R but it is incompatible with it. However, seasoned R coders can easily use RGL because of an existing familiarity with the grid. And RGL is very cool! It has a lot of options for 3-D shapes, various lighting effects, creating new shapes, and also animations. You can install
RGL from CRAN using install.packages("rgl ").

The dygraphs package is an **R** interface to the JavaScript charting library dygraphs that are used to provide various charts for visualizing data sets. This package can be used for creating various interactive visualizations with zooming, and panning options along with default mouse-over labels. dygraphs also provides support for various graph overlays such as point annotations, shaded regions, event lines, etc. You can also plot the xts time series objects automatically. However, all of these features do not come at the expense of speed in dygraph. Rather, it can provide maximal interactivity even with millions of data points in the visualization. You can install **RGL** from CRAN using install.packages(" dygraphs").

7. Leaflet

Just like dygraphs, the Leaflet package is an **R** interface to the JavaScript Leaflet library that is extremely popular. Leaflet is very useful in creating interactive but lightweight maps that are seen on various websites such as the Washington Post, the New York Times, etc. There are many useful features in this package such as interactive panning and zooming in the charts, the option to combine Polygons, Lines, Popups, etc. to create charts, embed maps in knitr, create maps in mercator projections that are nonspherical and so on. The Leaflet package can be used at the **R** console after installing it from CRAN using the command install.packages("leaflet").

All of these **R** Libraries for Data Visualization are excellent options if you want to create data visualizations. Each of these libraries has its strengths and you can choose the best one depending on the type of visualization or data science project you want to create. Now that you know these libraries, go on and create beautiful yet informative data visualizations using them!

Program

1. Write a R program to demonstrate use of the ggplot library.

2. Write a R program to demonstrate use of plotly library.

Conclusion-

Features of plotly and ggplot2 libraries are

What is Plotly?

Which programming languages can be used with Plotly?

Experiment No.8
Data Visualization: Use different Python libraries for visualization.
Date of Performance:
Date of Submission:

Experiment No - 8

Aim - Data Visualization: Use Matplotlib and seabom Python libraries for visualization.

Objective:- To understand and apply the Matplotlib and seabom python libraries for visualization using python.

Description-

Data visualization is an easier way of presenting the data, however complex it is, to analyze trends and relationships amongst variables with the help of pictorial representation.

The following are the advantages of Data Visualization

- Easier representation of compels data
- Highlights good and bad performing areas
- Explores relationship between data points
- Identifies data patterns even for larger data points

While building visualization, it is always a good practice to keep some below mentioned points in mind

- Ensure appropriate usage of shapes, colors, and size while building visualization
- Plots/graphs using a co-ordinate system are more pronounced
- Knowledge of suitable plot with respect to the data types brings more clarity to the information
- Usage oflabels, titles, legends and pointers passes seamless information the wider audience

Python Libraries

There are a lot of python libraries which could be used to build visualization like matplotlib, vispy, bokeh, seabom, pygal, folium, plotly, cufflinks, and networkx. Of the many, matplotlib and seabom seems to be very widely used for basic to intermediate level of visualizations.

Matplotlib

It is an amazing visualization library in Python for 2D plots of arrays, It is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002. Let's try to understand some of the benefits and features of matplotlib

- It's fast, efficient as it is based on numpy and also easier to build
- Has undergone a lot of improvements from the open source community since inception and hence a better library having advanced features as well
- Well maintained visualization output with high quality graphics draws a lot of users to it
- Basic as well as advanced charts could be very easily built
- From the users/developers point of view, since it has a large community support, resolving issues and debugging becomes much easier

Seaborn

Conceptualized and built originally at the Stanford University, this library sits on top of matplotlib. In a sense, it has some flavors of matplotlib while from the visualization point, it is much better than matplotlib and has added features as well. Below are its advantages

- Built-in themes aid better visualization
- Statistical functions aiding better data insights
- Better aesthetics and built-in plots
- Helpful documentation with effective examples

Nature of Visualization

Depending on the number of variables used for plotting the visualization and the type of variables , there could be different types of charts which we could use to understand the relationship. Based on the count of variables, we could have

- Univariate plot(involves only one variable)
- Bivariate plot(more than one variable in required)

A Univariate plot could be for a continuous variable to understand the spread and distribution of the variable while for a discrete variable it could tell us the count

Similarly, a Bivariate plot for continuous variable could display essential statistic like correlation , for a continuous versus discrete variable could lead us to very important conclusions like understanding data distribution across different levels of a categorical variable. A bivariate plot between two discrete variables could also be developed.

Scatter Plot

Scatter plots or scatter graphs is a bivariate plot having greater resemblance to line graphs in the way they are built. A line graph uses a line on an X-Y axis to plot a continuous function, while a scatter plot relies on dots to represent individual pieces of data. These plots are very useful to see if two variables are correlated. Scatter plot could be 2 dimensional or 3 dimensional.

Syntax: seaborn.scatterplot(x=None , y=None, hue=None , style=None, size=None, data=None, palette=None , hue_order=None , hue_norm=None , sizes=None , size_order=None , size_norm=None , markers=True , style_order=None , x_bins=None , y_bins=None , units=None , estimator=None , ci=95, n_boot= lOOO, alpha='auto', xjitter=None , yjitter=None , legend= 'brief ', ax=None , **kwargs) Parameters:

x, y: Input data variables that should be numeric. data: Dataframe where each column is a variable and each row is an observation. size: Grouping variable that will produce points with different sizes. style: Grouping variable that will produce points with different markers. palette: Grouping variable that will produce points with different markers. markers: Object determining how to draw the markers for different levels. alpha: Proportional opacity of the points.

Returns: This method returns the Axes object with the plot drawn onto it.

Histograms display counts of data and are hence similar to a bar chart. A histogram plot can also tell us how close a data distribution is to a normal curve. While working out statistical method, it is very important that we have a data which is normally or close to a normal distribution. However, histograms are univariate in nature and bar charts bivariate.

A bar graph charts actual counts against categories e.g. height of the bar indicates the number of items in that category whereas a histogram displays the same categorical variables in bins.

Bins are integral part while building a histogram they control the data points which are within a range. As a widely accepted choice we usually limit bin to a size of 5-20, however this is totally governed by the data points which is present.

Countplot

A countplot is a plot between a categorical and a continuous variable. The continuous variable in this case being the number of times the categorical is present or simply the frequency. In a sense, count plot can be said to be closely linked to a histogram or a bar graph.

Syntax : seabom.countplot(x=None , y=None , hue=None , data=None, order=None, hue_order=None , orient=None, color=None, palette=None , saturation=0.75 , dodge=True , ax=None, **kwargs)

Parameters : This method is accepting the following parameters that are described below:

- x, y: This parameter take names of variables in data or vector data, optional, Inputs for plotting long-form data.
- hue : (optional) This parameter take column name for colour encoding.
- data : (optional) This parameter take DataFrame , array, or list of arrays, Dataset for plotting.

  Ifx and y are absent, this is interpreted as wide-form. Otherwise it is expected to be long-form.
- order, hue_order : (optional) This parameter take lists of strings. Order to plot the categorical levels in, otherwise the levels are inferred from the data objects.
- orient : (optional)This parameter take "v" I "h", Orientation of the plot (vertical or horizontal). This is usually inferred from the dtype of the input variables but can be used to specify when the "categorical " variable is a numeric or when plotting wide-form data.
- color : (optional) This parameter take matplotlib color, Color for all of the elements, or seed for a gradient palette.
- palette : (optional) This parameter take palette name, list, or diet, Colors to use for the different levels of the hue variable. Should be something that can be interpreted by color_palette() , or a dictionary mapping hue levels to matplotlib colors.

Program 1- Scatter plot (create or upload any suitable data set)

Code :

Output:

Program - pie chart to show cars data

Code:

Output:

Conclusion-

What is the difference between matplotlib and seaborn?

Which library is used to create statistical graphics in Python?

Which function is used to create a histogram in Seaborn?