Detailed Project Plan - R Package for Pearson Diagram

1.1 Objective:

The project's goal is to produce a R tool that plots data points from unknown distributions on a Pearson distributional diagram. This package allows users to analyze and visualize samples by plotting them against Skewness and Kurtosis.

1.2 Target Audience: The program is designed for statisticians, data scientists, and academics who have to understand and analyze complex data distributions.

2.1 Pearson Diagram: Backround and Components

Skewness ($\beta 1$) refers to the asymmetry of a probability distribution. If the skewness is positive, the distribution has a long right tail; if it is negative, the tail is to the left.

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

where,

$$S = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

and

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

- ($\beta 1$)>0 - Distribution is right skewed
- ($\beta 1$)<0 - Distribution is left skewed
- ($\beta 1$)=0 - Symmetric distribution

Kurtosis ($\beta 2$) measures the "tailedness" or sharpness of the distribution's peak. A kurtosis of 3 is characteristic of a normal distribution; values more than 3 suggest heavier tails and values less than 3 indicate lighter tails.

$$Kurtosis = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^4}{(n-1) \cdot S^4}$$

- ($\beta 2$)>3 - Distribution has sharper peak than normal distribution
- ($\beta 2$)<3 - Distribution is left skewed
- ($\beta 2$)=3 - Symmetric distribution

2.2 Pearson Family of Distributions The Pearson family has a variety of probability distributions that can be shown on the Pearson diagram which are grouped according to their skewness and kurtosis values.

The Pearson system classifies distributions into seven different families, each of which is determined by the differential equation that regulates their shape. The approach helps in classifying real-world data distributions based on their moments (mean, variance, skewness, and kurtosis).

2.3 Plotting using the Pearson Diagram The Pearson diagram represents skewness on the x-axis and kurtosis on the y-axis. Points matching to various distributions can be plotted using the determined skewness and kurtosis.

The origin corresponds to the Normal Distribution (skewness = 0, kurtosis = 3). Gamma and exponential distributions are shown to the right of the origin (with positive skewness and high kurtosis). Depending on the parameters, beta distributions can have either positive or negative skewness and varied kurtosis. This visualization makes it easy to classify data distributions and compare unknown data samples to known distributions.

In the X axis of the diagram is skewness(($\beta 1$)) and Y axis is Kurtosis(($\beta 2$)).

Various regions in the diagram correspond to different types of distributions, such as normal, exponential, uniform, and others. Based on the characteristics of each distribution it will be displayed in the diagram as dot, line, or area. Each representations were listed below with example distributions.

2.4 Different Components of the Pearson Diagram

- **Area Representation:** For distributions with a range of skewness and kurtosis (e.g., Beta Distribution, F-Distribution).
- **Dot Representation:** For distributions with specific skewness and kurtosis (e.g., Normal, Exponential).
- **Line Representation:** For distributions with a continuous range of skewness and kurtosis (e.g., Inverse Gamma, Log Normal).

3.1 Package Functionalities

Below are the functions intended to be developed as the completion of the project.

- **Core Functions:**

  1. **calculate_skewness_kurtosis(data):** Calculates and returns the skewness and kurtosis of the input data.

  – **Input:** A univariate or multivariate numerical data, where each observation represents a value from the dataset. Data should be validated for missing values and outliers before calculation.
  – **Output:** A named list containing two numeric values representing skewness and kurtosis.

  2. **plot_pearson_diagram(skewness, kurtosis):** Plots the calculated skewness and kurtosis on the Pearson diagram.

  – **Input:** Calculated skewness and kurtosis values
  – **Output:** A pearson plot built on ggplot2 with skewness and kurtosis plotted on the diagram.

  3. **compare_distributions(data_list):** This is an optional function which compares multiple distributions by plotting them on the Pearson diagram.

  – **Input:** A list of numeric vectors representing different distributions to be plotted.
  – **Output:** A Pearson diagram plot with multiple distributions.

  4. **export_diagram(file_path):** This function exports the Pearson diagram to a specified file format (e.g., PNG, PDF).

  – **Input:** File name (including extension) with specified path (string object).
  – **Output:** Output plot will be saved in the specified location.

- **Helper Functions:**

  1. **validate_input(data):** This function validates that the input data follows to the specified format and standards for further processing.

  - **Input:** Numeric vector to be validated.
  - **Output:** Returns TRUE if the input matches the required standards.

  2. **highlight_point_on_hover(point):** This feature makes the Pearson diagram more interactive by displaying more information when the user moves their cursor over a plotted point.

  - **Input:** Pearson plot object.
  - **Output:** Adds a hover tooltip that displays extra information, such as skewness, kurtosis (optionally mean and variance).

- **Performance-Optimized Functions:**

  1. **cpp_calculate_skewness_kurtosis(data):** A high-performance C++ function integrated using Rcpp to handle skewness and kurtosis calculations for large datasets.

  - **Input:** A univariate or multivariate numerical data, where each observation represents a value from the dataset
  - **Output:** A named list containing two numeric values representing skewness and kurtosis.

3.2 Integration with Existing Packages Make sure the package checks for the presence of required packages and installs them if necessary.

- **Required Packages:**

  - **Rcpp:** For integrating C++ code for performance optimization.
  - **ggplot2:** For creating and customizing the Pearson diagram and comparison charts.
  - **moments:** For calculating skewness and kurtosis if not implementing from scratch.

3.3 Object-Oriented Programming (OOP) and Rcpp Integration For this project, Rcpp and OOP are critical for structuring and optimizing the package's functionality. We'll write classes for data handling, statistical measures computations (such skewness and kurtosis), and charting and interactive features for Pearson diagrams. Rcpp will also include C++ code to improve efficiency with larger datasets.

- OOP Implementation:

R implements OOP using S3, S4, or R6 class systems. For this package, we'll use S3 classes because they provide an adaptable and user-friendly interface. We will define a class called PearsonDiagram to handle digram functionalities. It helps to manage the Pearson diagram and its components (e.g., areas, lines, dots).

- Attributes:

  - points: A dataframe containing the skewness and kurtosis values of plotted distributions.
  - diagram: The ggplot2 object representing the Pearson diagram.

- Methods:

  1. initialize(): Constructor to initialize an empty Pearson diagram.
  2. add_point(skewness, kurtosis): Adds a point to the Pearson diagram based on the skewness and kurtosis values.

3. highlight_point_on_hover(): Adds tooltip interactivity to display details when hovering over a plotted point.
4. plot_diagram(): Renders and displays the Pearson diagram.
5. export_diagram(file_path): Exports the Pearson diagram to a specified file format.

- Rcpp Integration:

Rcpp enables the integration of high-performance C++ code within R, making it especially effective for tasks involving huge datasets. In this project, we'll use Rcpp to calculate skewness and kurtosis for larger datasets while taking use of C++'s speed and performance. Using Rcpp for computationally complex activities such as skewness and kurtosis calculations to improve efficiency while ensuring smooth integration with the R package.

- Steps for Rcpp Integration

1. Install Rcpp: Verify the installation of Rcpp package.
2. Define the C++ function in Rcpp: Develop a C++ code for calculating skewness and kurtosis using Rcpp.
3. Compile and Use the C++ Code in R: Call the function from R for the optimized performance.

3.4 Exception Handling Exception handling in the package development ensures that the functions are reliable and handle incorrect inputs and outputs appropriately. The following are the important elements:

1. Input Validation

The validate_input() function ensures that the input data is numeric and has at least three data points. If the data does not fulfill these conditions, an informative error message is returned, which prevents further processing.

2. Data Range Checking

When computing skewness and kurtosis, the package ensures that the results fall within predicted ranges. If not, a notice is displayed, allowing the user to examine any outliers or data problems.

3. Error Handling for Plotting:

During plotting (for example, plot_pearson_diagram()), the package ensures that the skewness and kurtosis values are inside the Pearson diagram's boundaries. If the values are too extreme, the function issues a warning and modifies the plot bounds accordingly.

4. Rcpp Error Handling

When integrating with C++ using Rcpp, accurate error handling is used to catch and manage exceptions within the C++ code. This keeps the software from crashing while processing huge datasets.

3.5 Documentation This comprehensive documentation will make the package simple to use, maintain, and enhance, enabling accessibility for statisticians, data scientists, and developers alike.

1. User Guide

Detailed installation instructions for the package, taking into account prerequisites like ggplot2 and Rcpp. This will guarantee that users can easily set up the product. Examples of Use: Simple examples showing how to utilize key methods such as compare_distributions(), plot_pearson_diagram(), and calculate_skewness_kurtosis(). Users will be guided through typical use scenarios via the examples.

2. Documentation of Function:

The documentation for every function in the package will include: Enter the input parameters (numerical vectors for calculate_skewness_kurtosis()) to provide a comprehensive explanation of the inputs. Output: The structure of the outputs (such as a numerical vector with skewness and kurtosis). Samples: code snippets that show you how to call the function and read the output.

Roxygen2 will be used to provide this documentation, guaranteeing that users can obtain function information directly through the R help system.

3. Vignettes:

Step-by-Step Tutorials: These comprehensive guides will provide real-world examples and demonstrate how to use the package to analyze datasets using the Pearson diagram. The tutorials will include:

- Loading the dataset
- Calculating skewness and kurtosis.
- Plotting on the Pearson Diagram
- Comparing distributions.
- Vignettes will be interactive and available immediately from R.

4.1 Testing

The testing process is critical to ensuring that the package is reliable, correct, and efficient. The following are crucial components:

1. Unit Testing

- Correctness testing: Unit tests are used to ensure that each function achieves the expected results for a given set of inputs. For example, the `calculate_skewness_kurtosis()` function will be tested using known distributions (such as the normal distribution) to ensure that skewness and kurtosis computations are accurate.
- Edge Case Testing: Functions will be evaluated with extreme values or tiny datasets to ensure that they handle edge cases correctly (for example, datasets with outliers or insufficient points). The tests will also ensure that the correct error messages are generated during invalid input handling.

2. Performance testing

- Benchmarking: The package's performance will be assessed by running functions such as cpp_calculate_skewness_kurtosis() on big datasets. This is especially critical for the C++-based functions linked with Rcpp, which must handle huge datasets effectively.
- Stress Testing: The functions will be put through stress testing to mimic severe loads and discover potential performance bottlenecks. The package's behavior under harsh situations will be used to optimize it for real-world applications.

3. Automated testing (continuous integration)

- A continuous integration pipeline will be configured to execute tests automatically with each commit or pull request, assuring code quality and preventing regressions. Tools like testthat will be used to automate unit tests, while services like GitHub Actions and Travis CI will be utilized for continuous integration.

5.1 Milestones

Below are the planned deliverables till the completion of the Pearson Diagram R-Package development.

i. Initial implementation of OOP for Diagram class and Rcpp integration for Skewness/Kurtosis calculation - 17/09/2024
ii. Draft Package and Test Plan - TBC
iii. Initial Implementation of the package - TBC
iv. Peer Testing - TBC
v. Package and Report Submission - TBC