Detailed Project Plan - R Package for Pearson Diagram

Renswick Delvar, 300656093

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 - Background and Components

Skewness (β_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.

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

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 (β_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.

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

- $\beta_2 > 3$ - 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 third and fourth order of moments(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[1]. In particular, if the skewness values are very high, the square root transformation can be helpful in normalizing the skewness. Especially if the skewness values change greatly, this can help to simplify and make the plot easier to read.

The point of origin represents the Normal Distribution[3] (skewness = 0, kurtosis = 3). Gamma and exponential distributions can be seen to the right of the origin (with positive skewness along with substantial kurtosis). Beta distributions can have positive or negative skewness and varying kurtosis, depending on the parameters used[3]. This visualization makes it simple to categorize data distributions and compare unknown samples to known distributions.

In the x-axis of the diagram is skewness (β_1) and y-axis is Kurtosis (β_2). The square root of skewness value ($\sqrt{\beta_1}$) is used to capture the extreme

Various regions in the diagram correspond to different types of distributions, such as normal, exponential, uniform, and others. [2]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. And the developed package can be installed from GitHub using devtools.

- **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. (Additional/optional efforts to display the mean and variance of the input sample distribution on hover)

  - **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 This package checks for the presence of required packages and installs them as dependencies 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.

3.3.1 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. Computations in R are carried out by the R evaluator by evaluating function call objects[4]. 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 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.

3.3.2 Rcpp Integration:

Rcpp enables the integration of high-performance C++ code within R, making it particularly useful for jobs involving large datasets. External resources are utilised makes a big difference in R package development[5]. In this project, we'll utilize Rcpp to calculate skewness and kurtosis for larger datasets, taking advantage of C++'s speed and performance. Using Rcpp for computationally intensive tasks like skewness and kurtosis computations to increase efficiency while assuring seamless 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:

3.4.1 Input Validation

The input data must be numeric and have a minimum of three data points, which is ensured by the `validate_input()` method. An informative error message that stops further processing is returned if the data does not meet these requirements.

3.4.2 Data Range Checking

The package makes sure that the results of skewness and kurtosis computations stay within the expected ranges. If not, the user can review any anomalies or issues with the data by viewing the alert that is provided.

3.4.3 Error Handling for Plotting

The package makes sure that the skewness and kurtosis values are inside the bounds of the Pearson diagram when plotting (e.g., `plot_pearson_diagram()`). The function warns the user and adjusts the plot limits if the numbers are too extreme.

3.4.4 Rcpp Error Handling

Accurate error handling is used to catch and manage exceptions within the C++ code when integrating with C++ using Rcpp. This reduces package crashes when handling large 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.

3.5.1 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 `plot_pearson_diagram()`, `calculate_skewness_kurtosis()` and `compare_distributions()`. Users will be guided through typical use scenarios via the examples.

3.5.2 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.

3.5.3 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.

3.6 License

The package will be licensed under MIT as an open-source R package and installable using `devtools::install_github()`

4.1 Testing

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

4.1.1 Unit Testing

- **Correctness testing:** Unit tests are used to make sure that, given a set of inputs, each function returns the expected outcomes. 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:** To make sure that functions handle extreme values (such as datasets with outliers or insufficient points) correctly, they will be tested with extreme values or small datasets. Additionally, the tests will guarantee that when erroneous input is handled, the appropriate error messages are produced.

4.1.2 Evaluation of performance

- **Benchmarking:** Large datasets will be used to evaluate the performance of the package using functions like `cpp_calculate_skewness_kurtosis()` especially for the C++ functionsconnected to Rcpp.
- **Stress Testing:** To simulate extreme loads and identify any possible performance bottlenecks, the functions will be placed through stress testing. We will leverage the package's behavior in challenging scenarios to optimize it for practical uses.

4.1.3 Continuous Integration (automated testing)

- To ensure code quality and avoid regressions, a continuous integration pipeline will be set up to run tests automatically with every commit or pull request. Unit tests will be automated using tools such as testthat.

5.1 Milestones

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

S.No	Deliverable	Expected Date	Content
1	OOP & Rcpp integration - A3	17/09/2024	Initial implementation of OOP for Diagram class and Rcpp integration for Skewness/Kurtosis calculation
2	Draft Package and Test Plan	TBC	Package skeleton and the comprehensive test plan
3	Package in Development	TBC	Current state of the package in development phase with working condition
4	Peer Testing	TBC	Testing will be done by the peers based on the test plan
5	Package and Report Submission	TBC	Final version of the package with complete documentation and test reports for the submission

References

[1] Pearson Distribution, https://en.wikipedia.org/wiki/Pearson_distribution

[2] Systems of Frequency Curves, William Palin Elderton, Norman Lloyd Johnson

[3] An R Package for Fitting Distributions. Journal of Statistical Software, 64(4), 1–34. Delignette-Muller, M. L., & Dutang, C. (2015). fitdistrplus: https://doi.org/10.18637/jss.v064.i04

[4] Object-Oriented Programming, Functional Programming and R, John M. Chambers

[5] Thirteen Simple Steps for Creating An R Package with an External C++ Library, Dirk Eddelbuettel, 18 Nov, 2019.