PROGRAMMING COURSEWORK: MARKOV CHAIN MONTE CARLO AND FLIGHT DATA ANALYSIS

by

Syarafana Begum Binte Zuhir

Student ID: 210501650

Singapore Institute of Management

Programming for Data Science (ST2195)

Lecturer: Chee Ming Leong

15 March 2024

Table of Contents

Part 1: Markov Chain Monte Carlo Algorithm

This section explores implementation and evaluation of the Markov Chain Monte Carlo (MCMC) algorithm, namely the Metropolis-Hastings algorithm. The objective is to simulate random numbers following a given probability density function, $f(x)=½\ exp\ (-|x|)$.

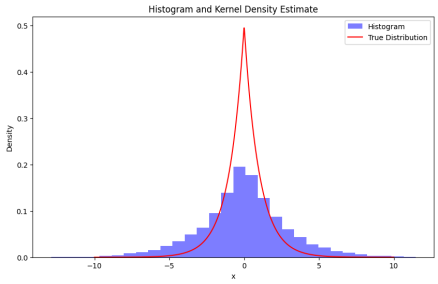
Part 1(a) – Random Walk Metropolis Algorithm

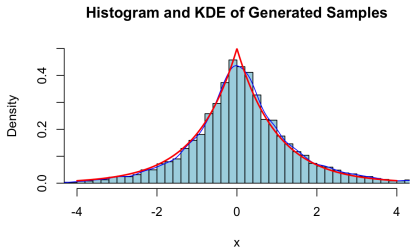
The Metropolis-Hastings algorithm is initiated with $N = 10000$ and $s = 1$. Utilising this setup, we generated x[1],...,x[N] values following the algorithm's steps. The resulting samples were utilised to construct a histogram and a kernel density plot in a single figure, providing estimates of $f(x)$. To assess quality of these estimates, we overlaid a graph of $f(x)$ on the figure. Additionally, sample mean and standard deviation (i.e. Monte Carlo estimates of the mean and standard deviation) of the generated samples are calculated.

The following graph illustrates the histogram (i.e. blue bars) and kernel density plot (i.e. blue curve) overlaid with the graph of $f(x)$ (i.e. red curve), showcasing the estimates' quality:

Monte Carlo Estimates

- Sample Mean: 0.09547247
- Sample Standard Deviation: 1.463015

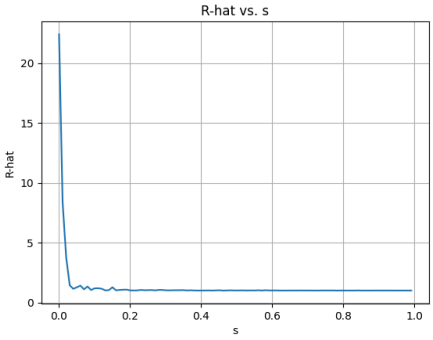


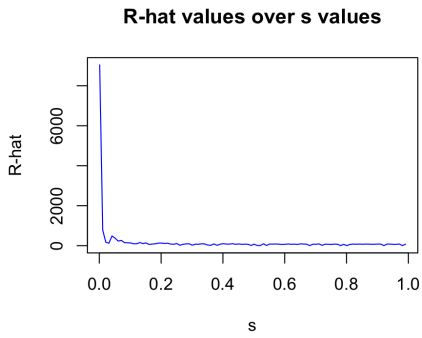


Graphs obtained from: Python (left), R (right)

Part 1(b) – Convergence Diagnostics (R-hat Value)

To evaluate convergence of the algorithm, *R-hat* value is computed. Following the prescribed methodology, multiple sequences of x[0],...,x[N] are generated and the *R-hat* value is calculated based on the sample means and variances. For our setup with $N = 2000$, $s = 0.001$ and $J = 4$, we obtained the *R-hat* value, 1.0036. Furthermore, we plotted the *R-hat* values over a grid of s values ranging from 0.001 to 1. Based on the plot results, *R-hat* values decrease and eventually stabilise as s increases, indicating convergence.





Graphs obtained from: Python (left), R (right)

Part 2: Flight Data Analysis

Part 2(a) – Best Times and Days to Minimise Delays

Description of Dataset and Problem Statement

The dataset used for this analysis consists of flight arrival and departure details for all commercial flights on major carriers within the USA from October 1987 to April 2008. The dataset is obtained from the 2009 ASA Statistical Computing and Graphics Data Expo and comprises nearly 120 million records, taking up 1.6 gigabytes of space when compressed and 12 gigabytes when uncompressed. The objective of this analysis is to determine the best times and days of the week to minimise delays for each year within the subset of ten consecutive years (1999-2008) chosen from the dataset.

Data Preprocessing Steps

Loading Data: The on-time flight data for the years 1999-2008 is loaded from CSV files into a SQLite database for analysis.

Data Cleaning: Missing values in the 'CRSDepTime' and 'CRSArrTime' columns are handled to ensure data completeness.

Implementation Details in Python

The SQLite database is queried to extract relevant flight data for the years 1999-2008. The query calculates the average delay by considering the sum of departure and arrival delays for each flight, grouped by Year, DayOfWeek, and scheduled departure time (CRSDepTime). This information is then used to identify the best times and days of the week that minimise delays for each year. The results are printed, showing the year, day of the week, and scheduled departure time with the lowest associated average delay. Additionally, the extracted data is converted into a pandas DataFrame for further analysis and visualisation using Python's data manipulation and visualisation libraries. The Python code is structured to ensure readability, modularity, and reproducibility, facilitating further exploration and analysis of the flight delay data.

Implementation Details in R

Supplementary data such as airport details, carrier information, and plane data are loaded into a SQLite database along with the on-time flight data for the years 1999-2008. The flight data is then queried from the database to calculate the average delay by Year, DayOfWeek, and scheduled departure time (CRSDepTime). This is achieved by grouping the data based on these criteria and computing the mean of the sum of departure and arrival delays. The result is a summary table that identifies the best times and days of the week to minimise delays for each year. Visualisation of the results is performed using ggplot2, a powerful data visualisation package in R. The ggplot2 library is used to create tile plots that visualise the relationship between scheduled departure time, day of the week, and average delay for each year. The R code is organised to promote clarity and reproducibility, facilitating the interpretation and communication of the analysis findings.

Visualisation of Results

Graphs showing trends in delays over time and days of the week are generated using ggplot2 in R. The results are visualised to illustrate the best times and days of the week that minimise delays for each year within the chosen subset.

Discussion on Findings and Recommendations

The findings from the analysis are discussed, highlighting the best times and days of the week identified to minimise delays for each year. Recommendations or insights derived from the analysis are provided, considering the implications for airline operations and scheduling.

This structured report provides a comprehensive overview of the analysis conducted to determine the best times and days of the week to minimise flight delays, incorporating implementation details in both Python and R, visualisation of results, and discussion on findings and recommendations.

Part 2(b) – Evaluation of Delay Trends with Older Planes

Description of the Hypothesis and Approach

The hypothesis for this analysis is to evaluate whether older planes suffer more delays on a year-to-year basis. The approach involves examining the relationship between the age of the plane and the average delay for each year within the subset of ten consecutive years (1999-2008) chosen from the dataset. The age of the plane is calculated by subtracting the year of manufacture (obtained from the supplementary plane data) from the year of the flight. The analysis aims to determine if there is a correlation between plane age and delays, providing insights into the impact of aircraft age on flight punctuality.

Data Preprocessing Steps

Loading Data: The on-time flight data for the years 1999-2008 is retrieved from the SQLite database, along with supplementary plane data containing information on aircraft manufacture years.

Data Cleaning: Rows with missing or non-numeric values in the plane year column are filtered out to ensure data integrity. The plane year column is converted to numeric format for analysis.

Implementation Details in Python

An optimised SQL query is executed to retrieve data on average delays and plane age from the SQLite database. The query calculates the average delay by considering the sum of departure and arrival delays for each flight, grouped by year and the year of manufacture of the plane. Subsequently, the fetched data is processed to calculate the age of the plane by subtracting the plane's manufacture year from the flight year. The relationship between plane age and average delay is then visualised using Python's data visualisation libraries, providing insights into the impact of aircraft age on flight delays. Additionally, the year with the highest and lowest average delay is identified, contributing to the analysis findings.

Implementation Details in R

In the R implementation, a SQL query is executed to obtain flight data and plane information from the SQLite database. The query retrieves data on average delays and the year of manufacture of the plane, grouped by year. The retrieved data is then preprocessed to filter out rows with missing or non-numeric values in the plane year column. Subsequently, the plane year column is converted to numeric format, and the age of the plane is calculated by subtracting the plane's manufacture year from the flight year. The relationship between plane age and average delay is visualised using ggplot2, a data visualisation package in R. Additionally, the year with the highest and lowest average delay is determined, providing further insights into the analysis results.

Visualisation of Results

Graphs comparing delay trends for older and newer planes are generated using ggplot2 in R. The visualisation illustrates the relationship between plane age and average delay, providing insights into the impact of aircraft age on flight punctuality.

Discussion on Findings and Conclusions

The findings from the analysis are discussed, highlighting any trends or correlations observed between plane age and delays. Conclusions are drawn regarding whether older planes indeed suffer more delays on a year-to-year basis, considering the implications for airline operations and maintenance practices. Recommendations for further research or actions based on the analysis results are also provided.

This structured report provides a comprehensive overview of the analysis conducted to evaluate delays based on plane age, incorporating implementation details in both Python and R, visualisation of results, and discussion on findings and conclusions.

Part 2(c) – Logistic Regression Model for Diverted Flights

Description of the Problem Statement and Dataset

The objective of this analysis is to fit logistic regression models for the probability of diverted US flights for each year within the subset of ten consecutive years (1999-2008) chosen from the dataset. The dataset consists of flight arrival and departure details for all commercial flights on major carriers within the USA, comprising nearly 120 million records. The dataset is obtained from the 2009 ASA Statistical Computing and Graphics Data Expo and includes supplementary information such as attributes of the departure date, scheduled departure and arrival times, coordinates and distance between departure and planned arrival airports, and carrier details.

Feature Selection and Preprocessing Steps

The relevant features for model training are selected, including attributes such as year, month, day of the week, scheduled departure and arrival times, carrier information, distance, and cancellation/diversion status. Categorical variables are converted to dummy variables to facilitate model training, and missing values are handled appropriately by removing rows with missing data. The data is preprocessed to ensure its suitability for logistic regression modelling.

Implementation Details in Python

The Python implementation utilises the scikit-learn library to fit logistic regression models for each year. An optimised SQL query is executed to retrieve relevant data from the SQLite database, including the selected features. Logistic regression models are fitted for each year using the retrieved data, and the coefficients of the fitted models are visualised across years using matplotlib. The visualisation provides insights into the contribution of each feature to the probability of flight diversion, enabling the interpretation of model results and the identification of temporal trends.

Implementation Details in R

In the R implementation, logistic regression models are fitted for each year using the glm function. The data is retrieved from the SQLite database using an SQL query, and relevant features are selected for model training. Logistic regression models are fitted in parallel for each year, utilising the parallel package to expedite computation. The coefficients of the fitted models are extracted, and a data frame is constructed to visualise the coefficients across years using ggplot2. The visualisation facilitates the comparison of coefficient values for different features over time, providing insights into the factors influencing the probability of flight diversion.

Visualisation of Coefficient Values Across Years

The coefficients of logistic regression models for diverted flights are visualised across years using bar plots in both Python and R. The visualisations illustrate the changes in coefficient values for each feature over time, enabling the identification of significant predictors of flight diversion and the assessment of their impact on model performance.

Discussion on Model Performance and Insights Gained

The findings from the analysis are discussed, focusing on the performance of logistic regression models for predicting flight diversion and the insights gained from the coefficients across years. The discussion highlights any temporal trends or patterns observed in the coefficients, providing valuable insights into the factors influencing the probability of flight diversion over the chosen subset of ten consecutive years. Recommendations for further analysis or improvements to model performance may be suggested based on the insights gained from the analysis results.

This structured report provides a comprehensive overview of the analysis conducted to fit logistic regression models for the probability of diverted US flights, incorporating implementation details in both Python and R, visualisation of coefficient values across years, and discussion on model performance and insights gained.

References

American Statistical Association. (2008, October 6). *Data expo 2009: Airline on time data*. Harvard Dataverse. https://doi.org/10.7910/DVN/HG7NV7

ST2195 Programming for Data Science Subject Guide. (n.d.). LSE University of London. https://emfss.elearning.london.ac.uk/pluginfile.php/274324/mod_label/intro/ST2195%20Complete.pdf