Analyzing Data from an Activity Monitoring Device

Saurabh Kelkar

2024-09-29

Contents

Introduction **1**

Data **1**

R Packages 1

Result and Discussion **2**

Part 1: Data Analysis without Imputation 2

Part 2: Data Analysis with Imputation 5

Conclusion **8**

Introduction

Devices that track physical activity generate substantial amounts of data, but much of this data is often not fully leveraged due to difficulties in accessing the raw data and limitations in the statistical tools and software available for analysis.

This project utilizes data from a personal activity tracker that logs activity every 5 minutes throughout the day. The dataset comprises two months of activity records from an anonymous individual, collected during October and November 2012, detailing the number of steps taken in 5-minute intervals each day.

Data

R Packages

```r
suppressPackageStartupMessages({
  library(dplyr)
  library(ggplot2)
  library(VIM)
  library(lattice)
})
```

```
## Warning: package 'VIM' was built under R version 4.4.1
```

```
## Warning: package 'colorspace' was built under R version 4.4.1
```

Functions from the `dplyr` package are used for data manipulation and analysis, while `lattice` and `ggplot2` are utilized for data visualization. The `VIM` package is employed to handle missing data. Startup messages from are suppressed using `suppressPackageStartupMessages()` to maintain clean code output.

The data is in CSV format, and we use the built-in `read.csv` function to load it:

```r
data <- read.csv("D:/Mini_projects/Project_1/activity.csv")
```

Before starting the analysis, it's important to inspect the data structure to understand its characteristics:

```r
glimpse(data)
```

```
## Rows: 17,568
## Columns: 3
## $ steps    <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ date     <chr> "2012-10-01", "2012-10-01", "2012-10-01", "2012-10-01", "2012~
## $ interval <int> 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 100, 105, 110, ~
```

In this case, `glimpse()` is used to explore the dataset. Alternatively, the built-in `str()` function can be used for a similar overview.

Result and Discussion

Part 1: Data Analysis without Imputation

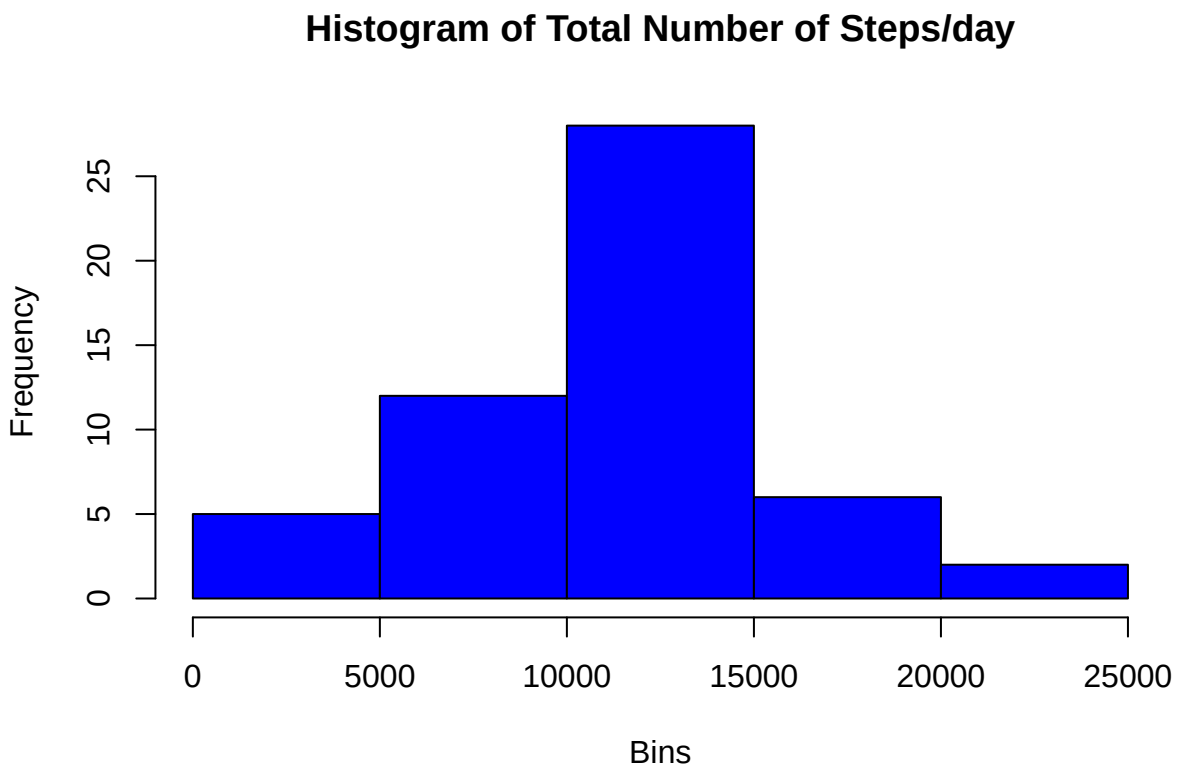
After examining the data structure, the next steps involve applying basic statistics and visualization to extract meaningful insights from the activity data. The first task is to determine the total number of steps taken per day.

In any dataset, `NA` values can be either removed or imputed using statistical techniques to ensure data consistency. Initially, the missing values will be ignored by filtering them out with `filter()`, and the total number of steps per day will be calculated using `summarise()`.

```r
total_steps <- data |>
  filter(steps != "NA") |>
  group_by(date) |>
  summarise(Total_Steps=sum(steps))
```

With the total number of steps calculated, the next step is to examine its distribution using a histogram:

```
hist(total_steps$Total_Steps,
     main = "Histogram of Total Number of Steps/day",
     xlab = "Bins", ylab = "Frequency", col = "blue")
```



The bins represent the number of steps taken each day, while the frequency shows how many days fall into each bin. The data appears to follow a bell curve, suggesting a normal distribution. The highest frequency is around the 10k bin, which likely indicates that the mean will be in this range. The histogram shows some gaps on the x-axis due to the filtering out of `NA` values. These `NA`s have not been fully removed or imputed, but were simply ignored. In a later section, the impact of filling in `NA` values on the histogram and central tendencies of the data will be explored.

Now, let's have a look at statistics of total number of steps taken per day with `summary()` function.

```
summary(total_steps$Total_Steps)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      41    8841   10765   10766   13294   21194
```

The summarized statistics shows the number of steps range from 41 to 21,194 per day. The median step count is 10,765, which is very close to the mean of 10,766, indicating a fairly symmetric distribution (also, seen in the Histogram). The lower quartile (1st Qu.) is 8,841 steps, and the upper quartile (3rd Qu.) is 13,294 steps, suggesting that 50% of the daily step counts fall between these two values. The data includes a wide spread, with some days having significantly fewer or more steps than the central tendency, as shown by the minimum (41) and maximum (21,194) values.

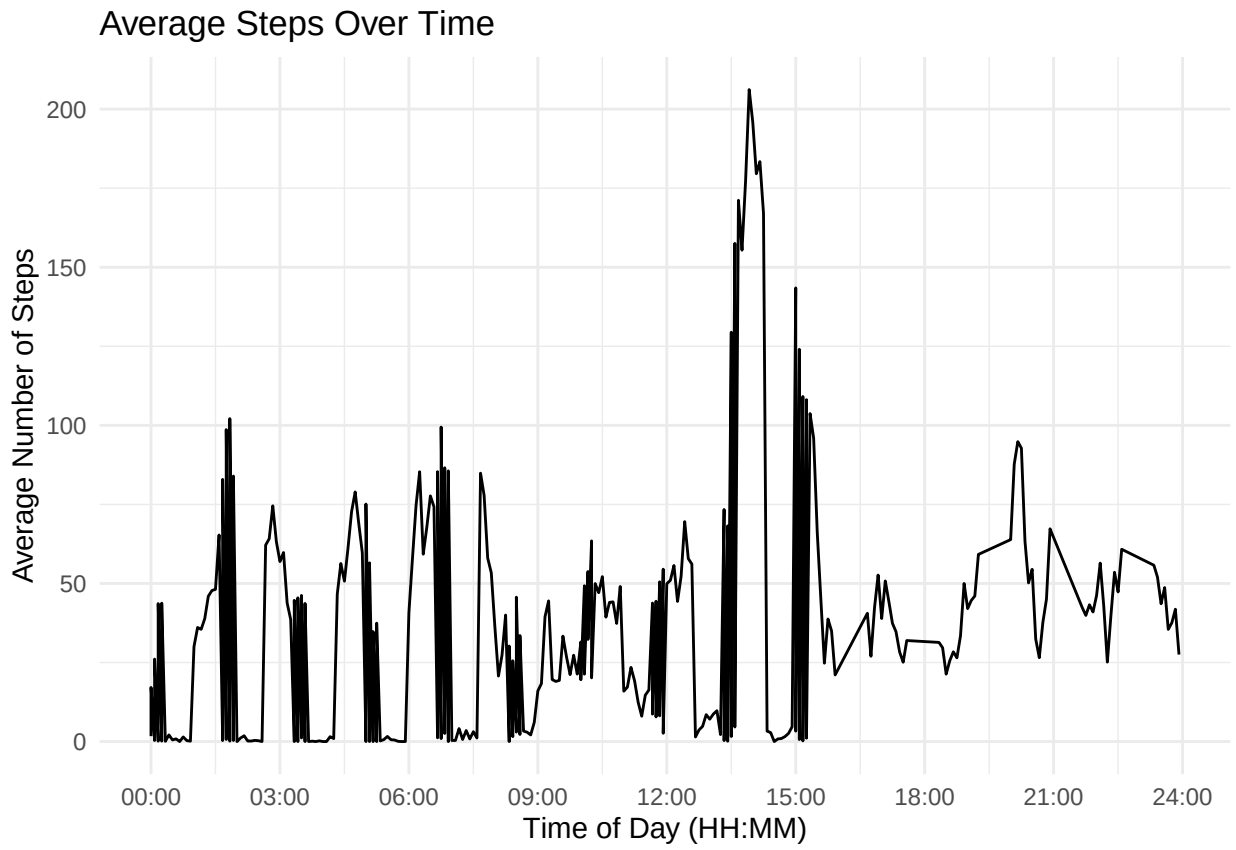
Another important aspect of the activity data is examining the average daily step pattern.

This can be explored using a time series plot of the 5-minute intervals (x-axis) and the average number of steps taken across all days (y-axis). Steps are aggregated by interval for each day.

```
# Aggregate steps over 5-minute intervals
avg_steps <- aggregate(steps ~ interval, data, mean, na.rm = TRUE)

# Fix the interval to be within a 24-hour clock
avg_steps$time_of_day <- avg_steps$interval %% 1440  # Wrap intervals within 1440 (24 hours * 60 minute

# Plot using ggplot2
ggplot(avg_steps, aes(x = time_of_day, y = steps)) +
  geom_line() +                    # Line plot for the time series
  labs(title = "Average Steps Over Time",
       x = "Time of Day (HH:MM)",  # Label for x-axis
       y = "Average Number of Steps") +    # Label for y-axis
  scale_x_continuous(breaks = seq(0, 1440, by = 180),  # Set x-axis breaks every 3 hours
       labels = function(x) sprintf("%02d:%02d", x %/% 60, x %% 60)) +  # Convert intervals to HH:MM fo
  theme_minimal()                  # Use a minimal theme for better aesthetics
```



Initially, a mistake was made by treating the interval variable as an integer, representing the number of intervals. This has been corrected by adjusting the interval to reflect the time of day.

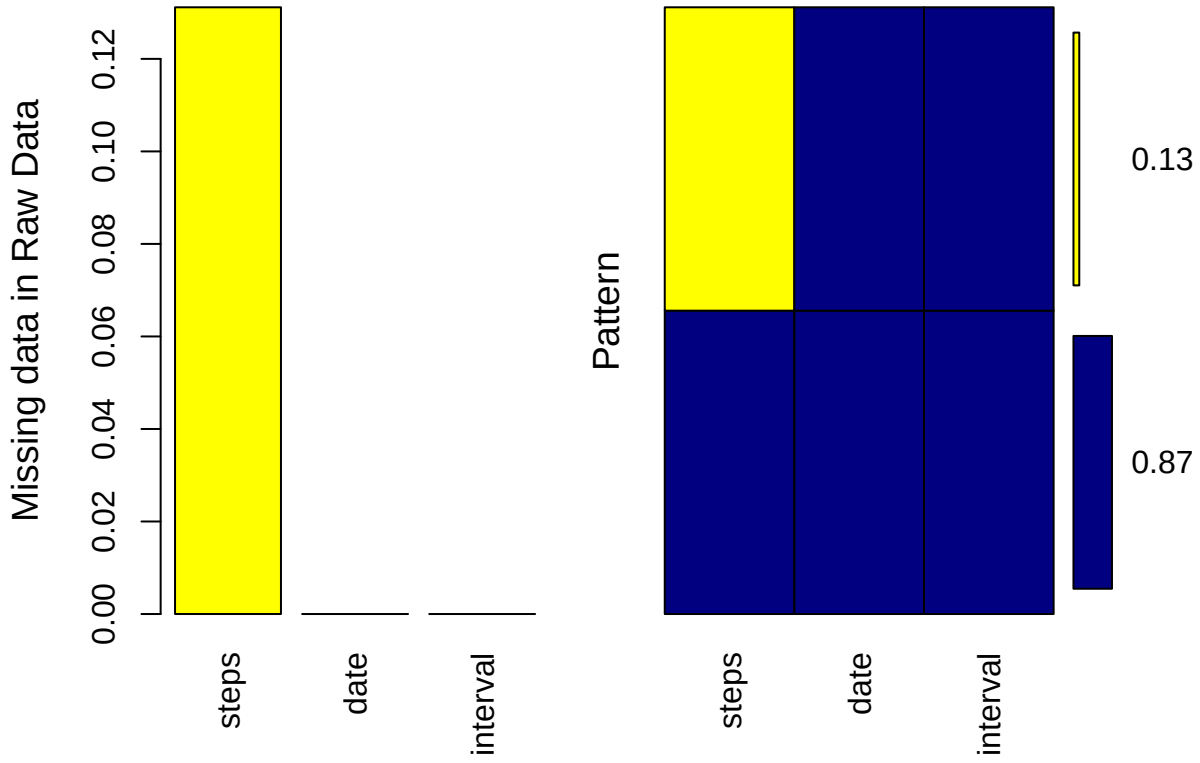
The time frame between 13:00 and 14:00 shows the highest average number of steps. This could indicate that the user might be walking more during lunchtime, suggesting a possible office worker routine where more walking occurs during breaks. Maybe the user is spending more time walking around than in the office. We have all been there! LOL!

Part 2: Data Analysis with Imputation

Recall that all previous calculations were performed without handling the `NA` values. However, missing data can significantly obscure data interpretation. To improve the accuracy of the analysis, the next step is to address the missing values.

First, let's calculate the total number of missing values in the dataset. `aggr()` function `VIM` package is useful in aggregating missing/imputed values. It calculates and plot the amount of missing/imputed values in each variable and the amount of missing/imputed values in certain combinations of variables.

```
aggr(data, col = c('navyblue', 'yellow'),
     numbers = TRUE, sortVars = TRUE, labels = names(data),
     cex.axis = 1.0, gap = 3,
     ylab = c("Missing data in Raw Data", "Pattern"))
```




```
##
##  Variables sorted by number of missings:
##  Variable     Count
##     steps 0.1311475
##      date 0.0000000
##  interval 0.0000000
```

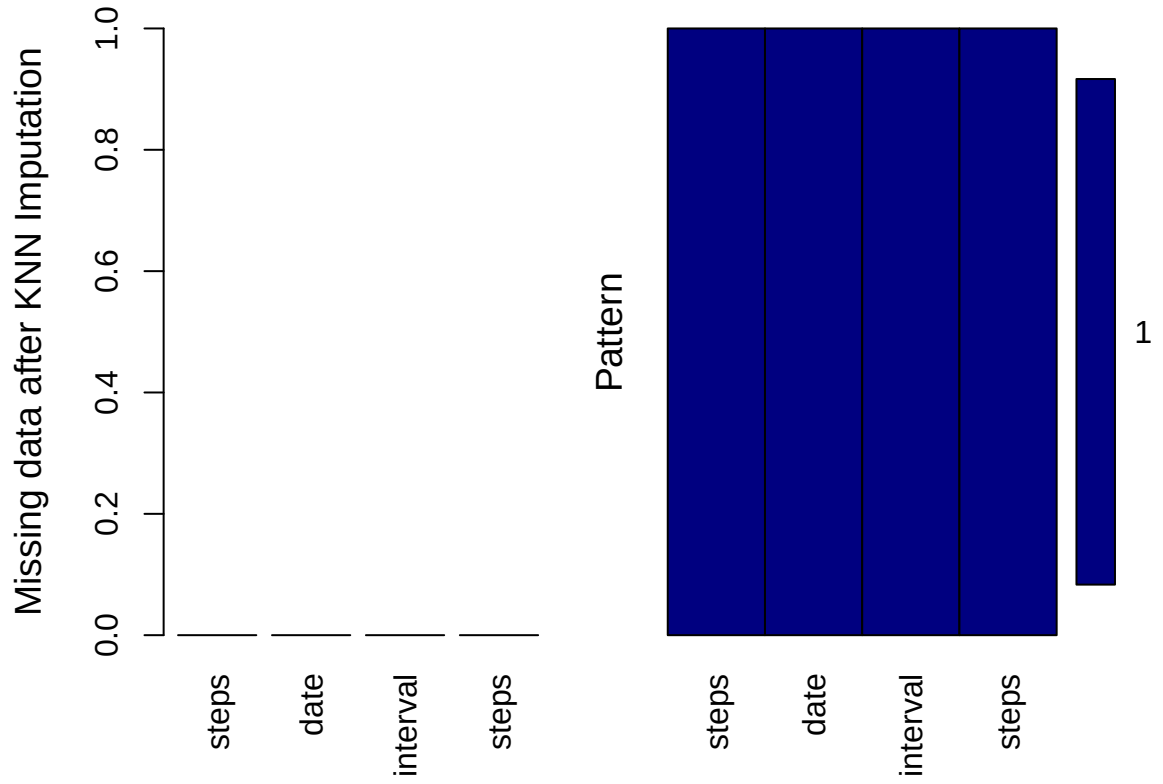
The activity data contains **2,304** missing values for the "steps" variable (highlighted in yellow on the bar plot), representing almost 13% of the total dataset. The bar plot also indicates that missing values occur only in the "steps" variable. Therefore, the focus will be on addressing the missing values in "steps."

There are several methods for filling in missing values, such as replacing them with the mean or median, forward or backward filling, or interpolating based on neighboring values. If the percentage of missing data is low (around 5%), simpler methods like mean or median imputation may suffice. However, when a larger portion of the data is missing, more advanced techniques like K-Nearest Neighbors (KNN) Imputation are more effective.

KNN Imputation works by identifying the closest observations (neighbors) within the dataset and imputing missing values based on the known values of these neighbors. This method is especially useful when the dataset contains multiple correlated variables. Here, we will use KNN Imputation to handle the missing data.

```
new_dat <- kNN(data, "steps", numFun = median)
```

```
aggr(new_dat, col = c('navyblue', 'yellow'),
     numbers = TRUE, sortVars = TRUE, labels = names(data),
     cex.axis = 1.0, gap = 3,
     ylab = c("Missing data after KNN Imputation", "Pattern"))
```

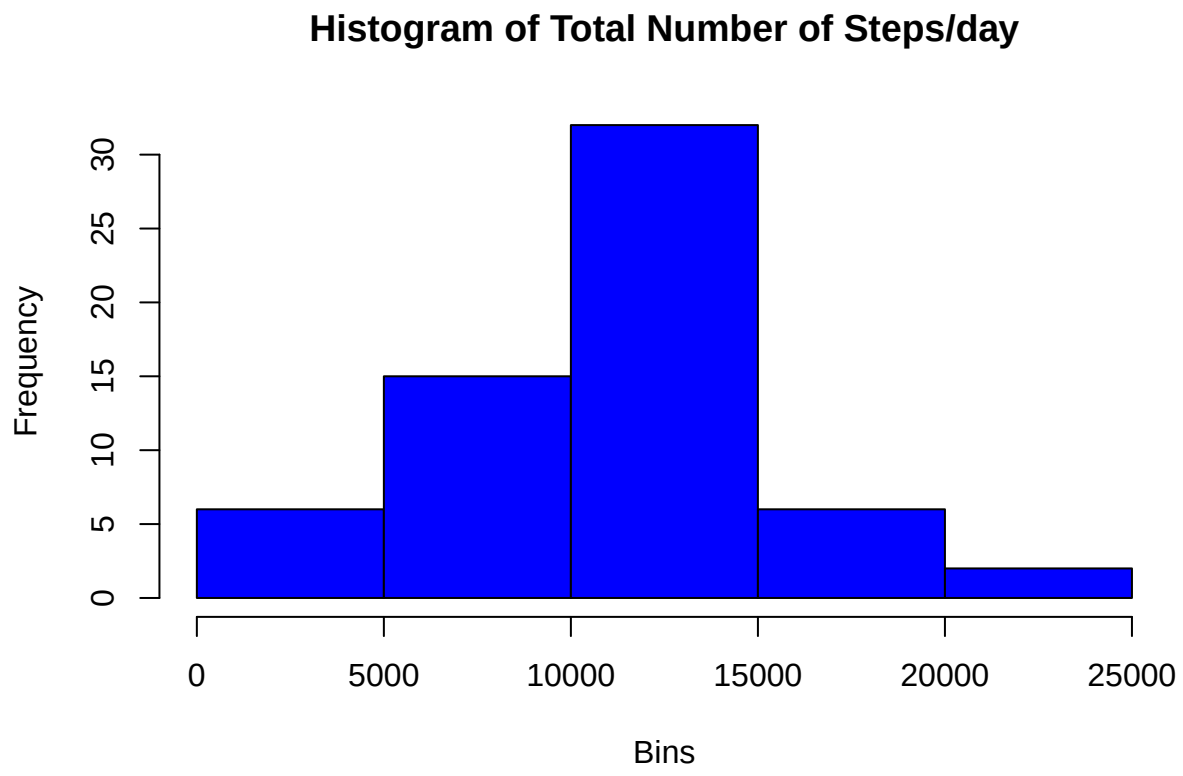



```
##
##  Variables sorted by number of missings:
##  Variable Count
##     steps     0
##      date     0
##  interval     0
##     steps     0
```

As expected, KNN Imputation has addressed the missing values in the dataset. Similar to the earlier analysis, the next step is to estimate the total number of steps per day and plot a histogram to observe the changes resulting from replacing the `NA` values.

```
new_total_steps <- new_dat |>
  group_by(date) |>                 # Group by the date variable
  summarise(Total_Steps=sum(steps)) # Use summarise to calculate the total number of steps in day

# Plot histogram to see the distribution of data
hist(new_total_steps$Total_Steps,
     main = "Histogram of Total Number of Steps/day",
     xlab = "Bins", ylab = "Frequency", col = "blue")
```




```
# Calculate summary statistics
summary(new_total_steps$Total_Steps)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0    8821   10600   10538   12883   21194
```

The summary statistics after imputation reveal several key insights: the minimum number of steps recorded is 0 (probably a result of KNN), indicating days with no recorded activity, while the maximum is 21,194 steps, suggesting a significant level of physical activity on certain days. The first quartile (1st Qu.) is 8,821 steps, and the third quartile (3rd Qu.) is 12,883 steps, indicating that 25% of the days had fewer than 8,821 steps and 75% had fewer than 12,883 steps. The median value of 10,600 steps represents the midpoint of the data, indicating that half of the days recorded fewer than 10,600 steps and half recorded more. The mean,

calculated at 10,538 steps, is very close to the median, suggesting a relatively symmetric distribution of step counts without significant outliers influencing the average.

Conclusion

Comparing the statistics before and after imputation reveals that the mean and median values remained largely unchanged. This stability suggests that the missing values had a limited effect on the overall dataset, and their absence did not significantly distort the central tendency measures. One reason for this could be the relatively small proportion of missing data—only 13% of the total dataset. In many cases, a dataset with such a small percentage of missing entries is not enough to noticeably shift key summary statistics like the mean and median, especially if the missing data points are distributed evenly or randomly. Furthermore, since imputation fills in missing values based on patterns or relationships within the available data, the filled values tend to align with the existing dataset structure, further minimizing any drastic changes in the statistical measures. This highlights that in cases with a moderate amount of missing data, especially when handled with effective imputation methods, the integrity of the dataset can often be preserved without significantly altering the insights drawn from it.

Note

1. This project does not focus on answering any specific research questions. Instead, its primary objective is to demonstrate the ability to thoroughly analyze, visualize, and interpret data, while addressing various complexities within the dataset. The aim is to showcase a range of data manipulation techniques, statistical methods, and visualization tools to extract meaningful information.

2. The topic and dataset are associated with an assignment from Coursera: Reproducible Research. Therefore, I cannot share the raw data used for analysis without permission. However, I have expanded upon the assignment to turn it into a small project in R, applying my skills in both R and reproducible research.

3. This project is written in `R Markdown` and kneaded into a document using `knitr` package.
