Bellabeat Case Study Report

Albert Hunduza

2023-08-28

R Markdown

In this notebook, I outline how I approached the Bellabeat case study as part of my Google Data Analytics Capstone Project. The full description of the case study is attached in the file “Bellabeat Case Study”. A brief outline of the case study scenario is that I am working on the marketing analyst team at Bellabeat, a high-tech manufacturer of health-focused products for women. The goal is to analyze smart device usage data in order to gain insight into how people are already using their smart devices. Using these insights, I have to give high-level recommendations for how these trends can inform Bellabeat marketing strategy, particularly for one of the Bellabeat products. Key questions include, but are not limited to:

1. What are some trends in smart device usage?
2. How could these trends apply to Bellabeat customers?
3. How could these trends help influence Bellabeat marketing strategy?

This study is one of the key deliverables, mostly relevant to my colleageues on the marketing data analytics team at Bellabeat. I started by importing the required libraries.

```
# Load the required libraries
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

The next step was to import the relevant datasets from the dataset collection. I limited my choices to datasets that recorded daily and hourly data as these are more informative to a higher level marketing strategy.

```
# Importing relevant datasets
data_folder_path <- "Fitabase Data 4.12.16-5.12.16/" # sets path of folder containing files

daily_activity <- read.csv(paste(data_folder_path,"dailyActivity_merged.csv",sep=""))
sleep_day <- read.csv(paste(data_folder_path,"sleepDay_merged.csv",sep=""))
weight_log <- read.csv(paste(data_folder_path,"weightLogInfo_merged.csv",sep=""))
hourly_intensities <- read.csv(paste(data_folder_path,"hourlyIntensities_merged.csv",sep=""))
hourly_steps <- read.csv(paste(data_folder_path,"hourlySteps_merged.csv",sep=""))
heart_rate <- read.csv(paste(data_folder_path,"heartrate_seconds_merged.csv",sep=""))
```

I started by cleaning the data by checking for NaNs and duplicates:

```
# Checking for NaNs in each dataset
n_nans_daily_activity <- colSums(is.na(daily_activity))
n_nans_sleep_day <- colSums(is.na(sleep_day))
n_nans_weight_log <- colSums(is.na(weight_log))
n_nans_hourly_intensities <- colSums(is.na(hourly_intensities))
n_nans_hourly_steps <- colSums(is.na(hourly_steps))
n_nans_heart_rate <- colSums(is.na(heart_rate))

# Print the count of NaNs for each dataset
n_nans_daily_activity
```

```
##                       Id             ActivityDate               TotalSteps
##                        0                        0                        0
##            TotalDistance          TrackerDistance LoggedActivitiesDistance
##                        0                        0                        0
##       VeryActiveDistance ModeratelyActiveDistance      LightActiveDistance
##                        0                        0                        0
##  SedentaryActiveDistance        VeryActiveMinutes      FairlyActiveMinutes
##                        0                        0                        0
##     LightlyActiveMinutes         SedentaryMinutes                 Calories
##                        0                        0                        0
```

```
n_nans_sleep_day
```

```
##                Id           SleepDay  TotalSleepRecords TotalMinutesAsleep
##                 0                  0                  0                  0
##     TotalTimeInBed
##                 0
```

```
n_nans_weight_log
```

```
##             Id           Date       WeightKg   WeightPounds            Fat
##              0              0              0              0             65
##            BMI IsManualReport          LogId
##              0              0              0
```

```
n_nans_hourly_intensities
```

```
##               Id     ActivityHour   TotalIntensity AverageIntensity
##                0                0                0                0
```

```
n_nans_hourly_steps
```

```
##           Id ActivityHour    StepTotal
##            0            0            0
```

```
n_nans_heart_rate
```

```
##    Id  Time Value
##     0     0     0
```

Only the weight_log dataframe contained null values, and these were in the “Fat” column where some people in the study did not report their measurements for fat. Since an overwhelming majority (65/67) of records had null values in this column, I could not drop it. I therefore replaced all the NaNs in this column with “unreported”

```
# Replace NaNs in the "Fat" column with "unreported"
weight_log$Fat <- ifelse(is.na(weight_log$Fat), "unreported", weight_log$Fat)

# checking if NaNs have been dropped
n_nans_weight_log <- colSums(is.na(weight_log))
n_nans_weight_log
```

```
##             Id           Date       WeightKg   WeightPounds            Fat
##              0              0              0              0              0
##            BMI IsManualReport          LogId
##              0              0              0
```

Having eliminated the NaNs, I checked for duplicates

```
# Checking for duplicate rows in each dataset
n_duplicates_daily_activity <- sum(duplicated(daily_activity))
n_duplicates_sleep_day <- sum(duplicated(sleep_day))
n_duplicates_weight_log <- sum(duplicated(weight_log))
n_duplicates_hourly_intensities <- sum(duplicated(hourly_intensities))
n_duplicates_hourly_steps <- sum(duplicated(hourly_steps))
n_duplicates_heart_rate <- sum(duplicated(heart_rate))

# Print the count of duplicate rows for each dataset
n_duplicates_daily_activity
```

```
## [1] 0
```

```
n_duplicates_sleep_day
```

```
## [1] 3
```

```
n_duplicates_weight_log
```

```
## [1] 0
```

```
n_duplicates_hourly_intensities
```

```
## [1] 0
```

```
n_duplicates_hourly_steps
```

```
## [1] 0
```

```
n_duplicates_heart_rate
```

```
## [1] 0
```

The sleep_day dataframe had 3 duplicates. I eliminated them as follows:

```
# Remove duplicate rows from the sleep_day dataframe
sleep_day_cleaned <- sleep_day[!duplicated(sleep_day), ]

# check if duplicates have been removed
n_duplicates_sleep_day_cleaned <- sum(duplicated(sleep_day_cleaned))
n_duplicates_sleep_day_cleaned
```

```
## [1] 0
```

The hourly_intensities and hourly_steps dataframes are narrow and contain information that is related to each other. I therefore joined them on the Id and date columns into a new dataframe.

```
# merging the hourly_intensities and hourly_steps dataframes
hourly_exercises <- merge(hourly_intensities, hourly_steps, by = c("Id", "ActivityHour"))
```

My approach was to analyze each dataframe separately and etablish trends and behaviors from each before finally looking for insights across the dataframes

1) daily_activity dataframe

```
# Get the number of unique IDs/people in the "Id" column
unique_id_count <- daily_activity %>%
  distinct(Id) %>%
  nrow()

# Print the number of unique IDs/people
print(unique_id_count)
```

```
## [1] 33
```

There are 33 people who recorded their daily activity. Next we will find how many days each person recorded their activity. This will help us establish how useful our data can be.

```
# Convert the "ActivityDate" column to Date format
daily_activity %>%
  mutate(ActivityDate = as.Date(ActivityDate, format = "%m/%d/%Y")) %>%
  distinct(ActivityDate) %>%
  summarise(unique_date_count = n())
```

```
##   unique_date_count
## 1                31
```

```
# Convert the "ActivityDate" column to Date format and group by Id
result <- daily_activity %>%
  mutate(ActivityDate = as.Date(ActivityDate, format = "%m/%d/%Y")) %>%
  group_by(Id) %>%
  summarise(unique_date_count = n_distinct(ActivityDate))

# Set the option to display all rows
options(dplyr.print_max = Inf)

# Print the result
print(result)
```

```
## # A tibble: 33 × 2
##            Id unique_date_count
##         <dbl>             <int>
##  1 1503960366                31
##  2 1624580081                31
##  3 1644430081                30
##  4 1844505072                31
##  5 1927972279                31
##  6 2022484408                31
##  7 2026352035                31
##  8 2320127002                31
##  9 2347167796                18
## 10 2873212765                31
## 11 3372868164                20
## 12 3977333714                30
## 13 4020332650                31
## 14 4057192912                 4
## 15 4319703577                31
## 16 4388161847                31
## 17 4445114986                31
## 18 4558609924                31
## 19 4702921684                31
## 20 5553957443                31
## 21 5577150313                30
## 22 6117666160                28
## 23 6290855005                29
## 24 6775888955                26
## 25 6962181067                31
## 26 7007744171                26
## 27 7086361926                31
## 28 8053475328                31
## 29 8253242879                19
## 30 8378563200                31
## 31 8583815059                31
## 32 8792009665                29
## 33 8877689391                31
```

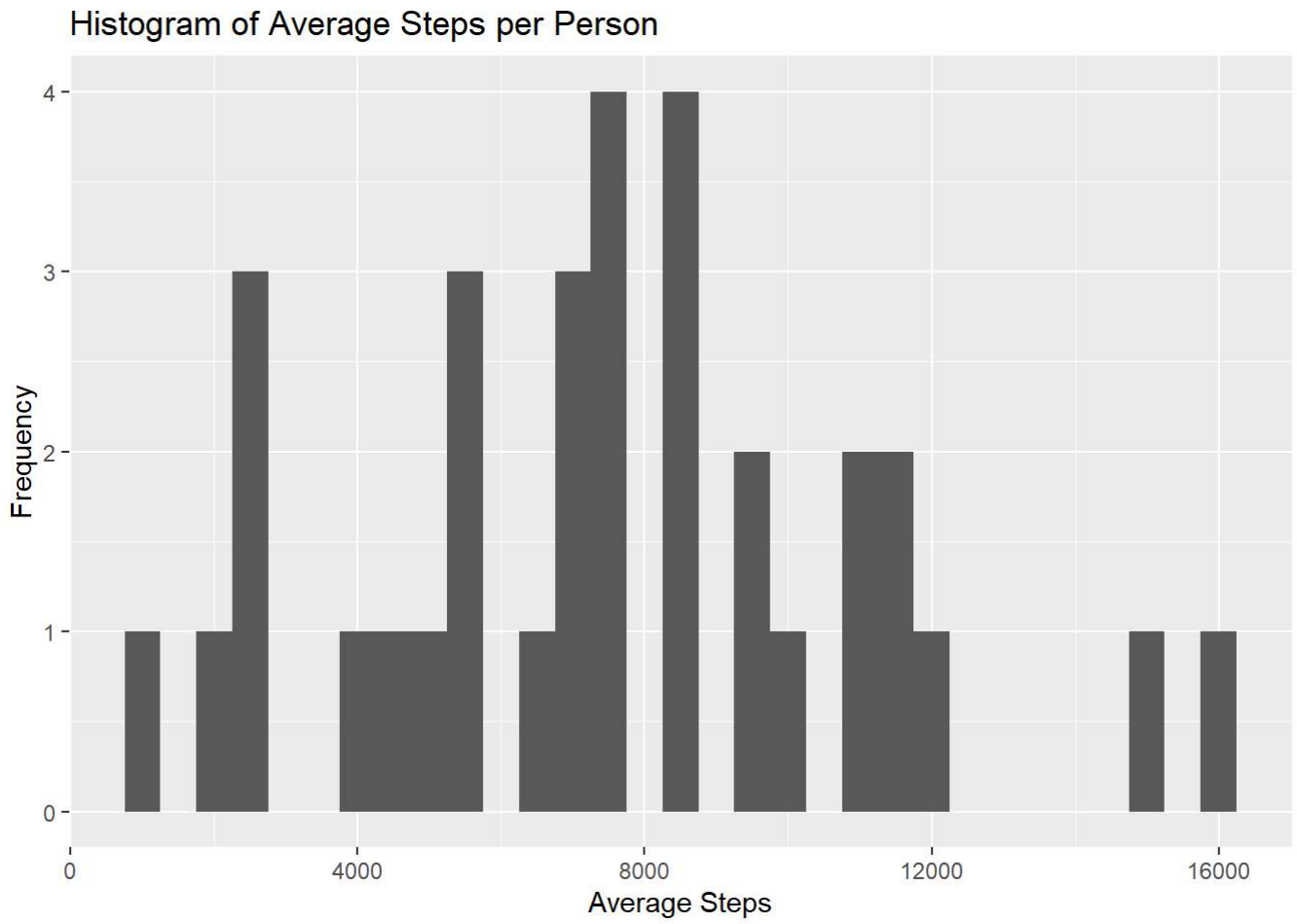
Not all participants recorded their daily activities on all 31 days of the study. One of the participants only recorded their data on 4 days. Next, I calculated the average number of steps and distance that each person walked over this study period.

```
# Calculate the average number of steps and distance per person
average_per_person <- daily_activity %>%
  group_by(Id) %>%
  summarise(average_steps = mean(TotalSteps, na.rm = TRUE),
            average_distance = mean(TotalDistance, na.rm = TRUE))

# Print the result
print(average_per_person)
```

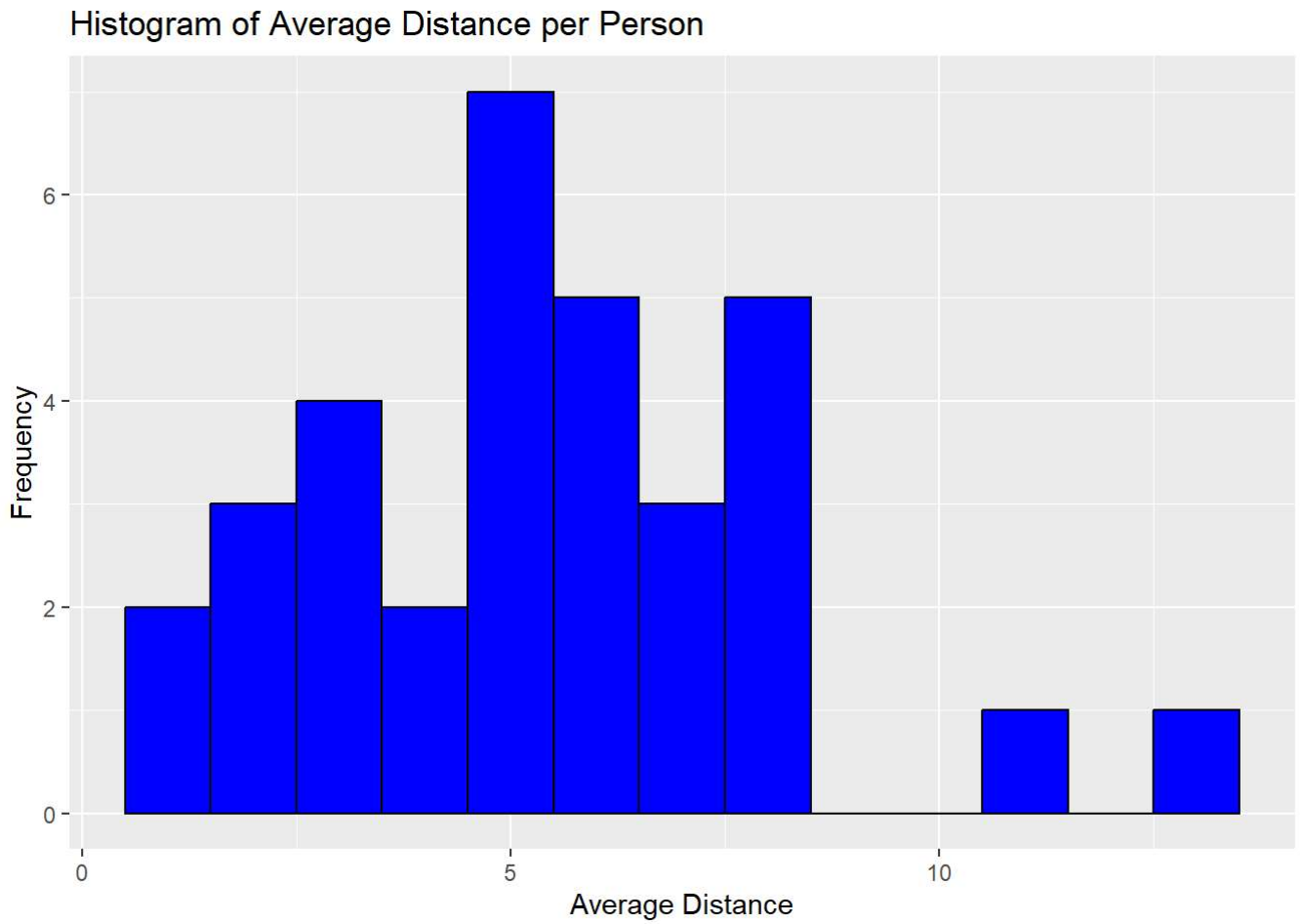
```
## # A tibble: 33 × 3
##            Id average_steps average_distance
##         <dbl>         <dbl>            <dbl>
##  1 1503960366        12117.            7.81
##  2 1624580081         5744.            3.91
##  3 1644430081         7283.            5.30
##  4 1844505072         2580.            1.71
##  5 1927972279          916.            0.635
##  6 2022484408        11371.            8.08
##  7 2026352035         5567.            3.45
##  8 2320127002         4717.            3.19
##  9 2347167796         9520.            6.36
## 10 2873212765         7556.            5.10
## 11 3372868164         6862.            4.71
## 12 3977333714        10985.            7.52
## 13 4020332650         2267.            1.63
## 14 4057192912         3838             2.86
## 15 4319703577         7269.            4.89
## 16 4388161847        10814.            8.39
## 17 4445114986         4797.            3.25
## 18 4558609924         7685.            5.08
## 19 4702921684         8572.            6.96
## 20 5553957443         8613.            5.64
## 21 5577150313         8304.            6.21
## 22 6117666160         7047.            5.34
## 23 6290855005         5650.            4.27
## 24 6775888955         2520.            1.81
## 25 6962181067         9795.            6.59
## 26 7007744171        11323.            8.02
## 27 7086361926         9372.            6.39
## 28 8053475328        14763.           11.5
## 29 8253242879         6482.            4.67
## 30 8378563200         8718.            6.91
## 31 8583815059         7199.            5.62
## 32 8792009665         1854.            1.19
## 33 8877689391        16040.           13.2
```

These results are best visualized in a histogram

```
# Calculate the average number of steps and distance per person
average_per_person <- daily_activity %>%
  group_by(Id) %>%
  summarise(average_steps = mean(TotalSteps, na.rm = TRUE),
            average_distance = mean(TotalDistance, na.rm = TRUE))

# Create a histogram for average steps
ggplot(average_per_person, aes(x = average_steps)) +
  geom_histogram(binwidth = 500) +
  labs(title = "Histogram of Average Steps per Person",
       x = "Average Steps",
       y = "Frequency")
```



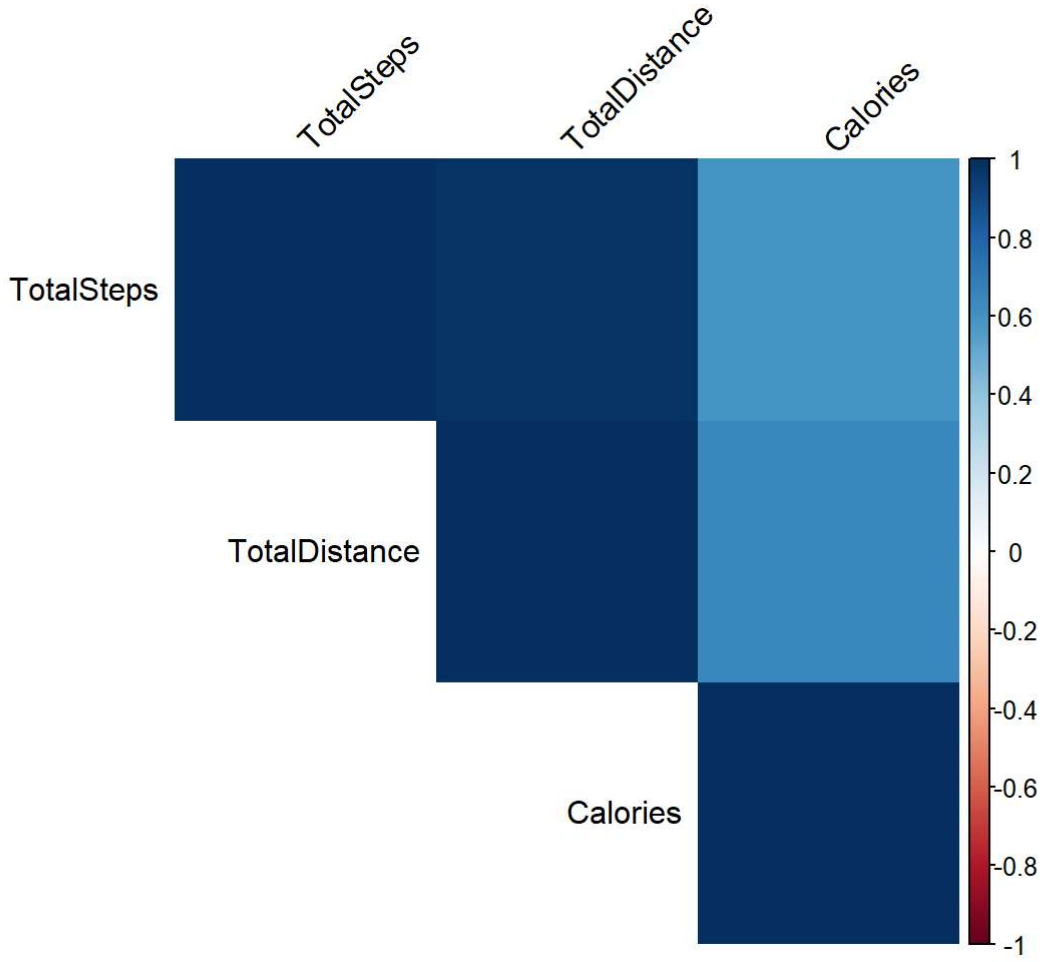

```
# Create a histogram for average distance
ggplot(average_per_person, aes(x = average_distance)) +
  geom_histogram(binwidth = 1, fill = "blue", color = "black") +
  labs(title = "Histogram of Average Distance per Person",
       x = "Average Distance",
       y = "Frequency")
```



On average, people in the study walked about 8000 steps or around 6km a day, with 16000 steps being the high end and 1000 the lower.

Next I created a correlation matrix to see which factor influences the amount of calories burned more between total steps and total distance.

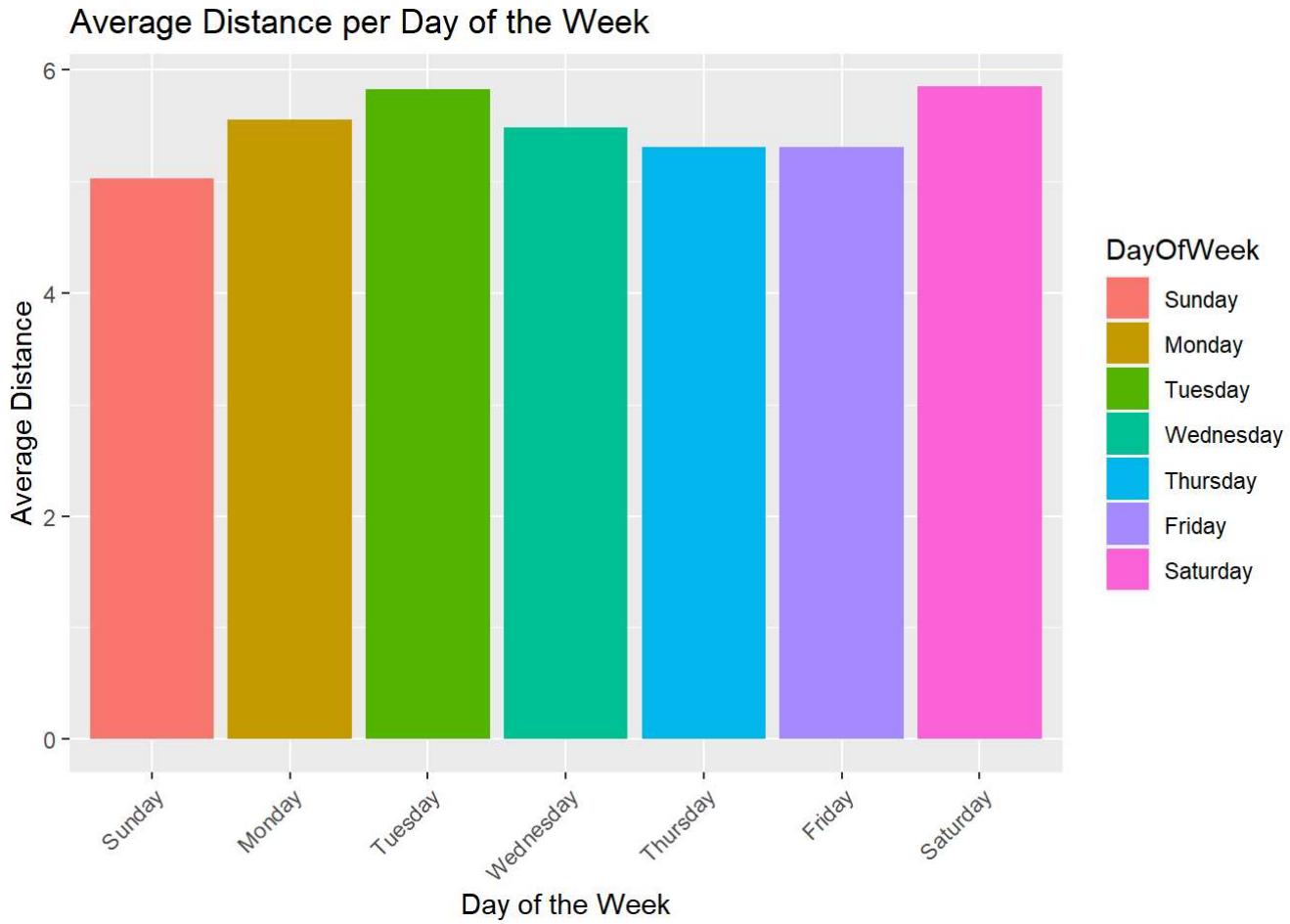
```
# Select the relevant columns
selected_columns <- daily_activity %>%
  select(TotalSteps, TotalDistance, Calories)

# Calculate the correlation matrix
correlation_matrix <- cor(selected_columns, use = "complete.obs")

# Visualize the correlation matrix with a title
corrplot(correlation_matrix, method = "shade", type = "upper",
         tl.col = "black", tl.srt = 45)
```



From the correlation matrix, we can see that total distance moved has a slightly greater influence on calories burned than total steps. Both factors are positively correlated to calories burned.

Next, I sought to discover if there are any days of the week when people tend to be more active than others

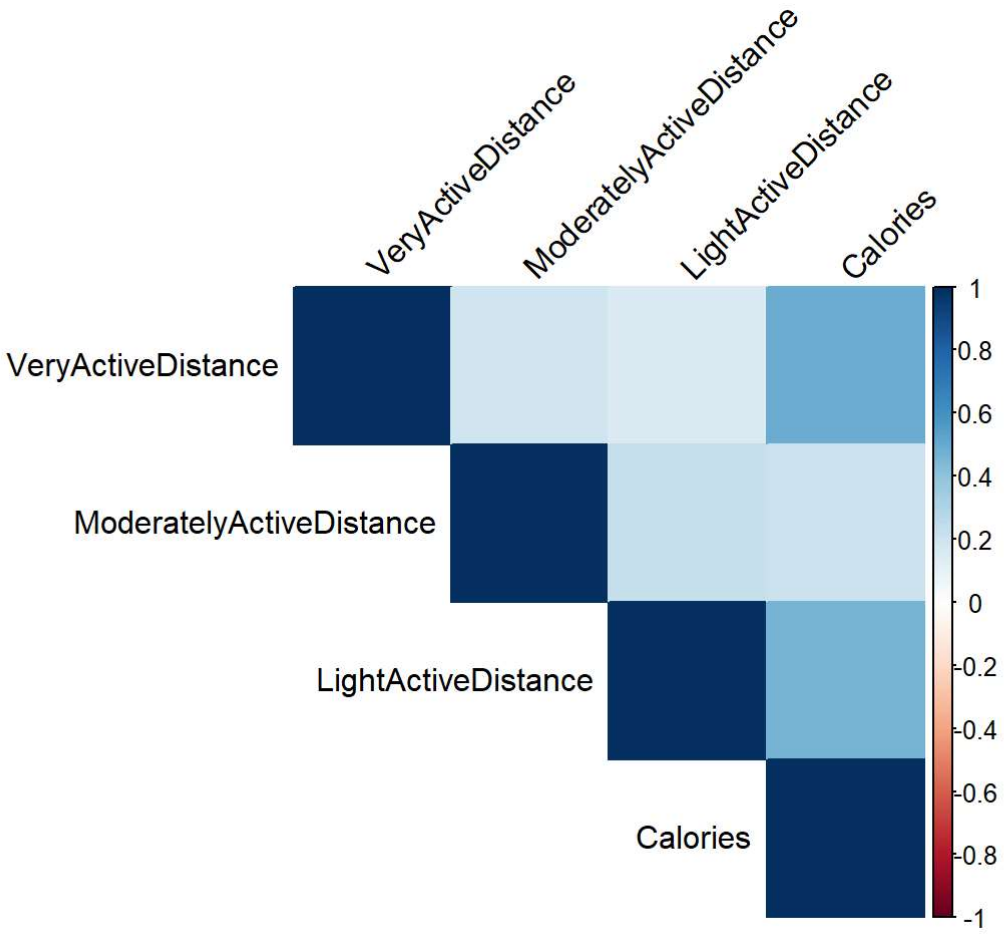
```
# Convert the "ActivityDate" column to Date format and add a new column for the day of the week
daily_activity_with_day <- daily_activity %>%
  mutate(ActivityDate = as.Date(ActivityDate, format = "%m/%d/%Y"),
         DayOfWeek = weekdays(ActivityDate))

# Group by day of the week and calculate the average distance
average_distance_per_day <- daily_activity_with_day %>%
  group_by(DayOfWeek) %>%
  summarise(average_distance = mean(TotalDistance, na.rm = TRUE))

# Order the days of the week
day_order <- c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")
average_distance_per_day$DayOfWeek <- factor(average_distance_per_day$DayOfWeek, levels = day_or
der)

# Create a bar plot
ggplot(average_distance_per_day, aes(x = DayOfWeek, y = average_distance, fill = DayOfWeek)) +
  geom_bar(stat = "identity") +
  labs(title = "Average Distance per Day of the Week",
       x = "Day of the Week",
       y = "Average Distance") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



The average distance moved seems to be constant for all days of the week. Saturday and Tuesday have slightly more distance than the other days and Sunday slightly less.

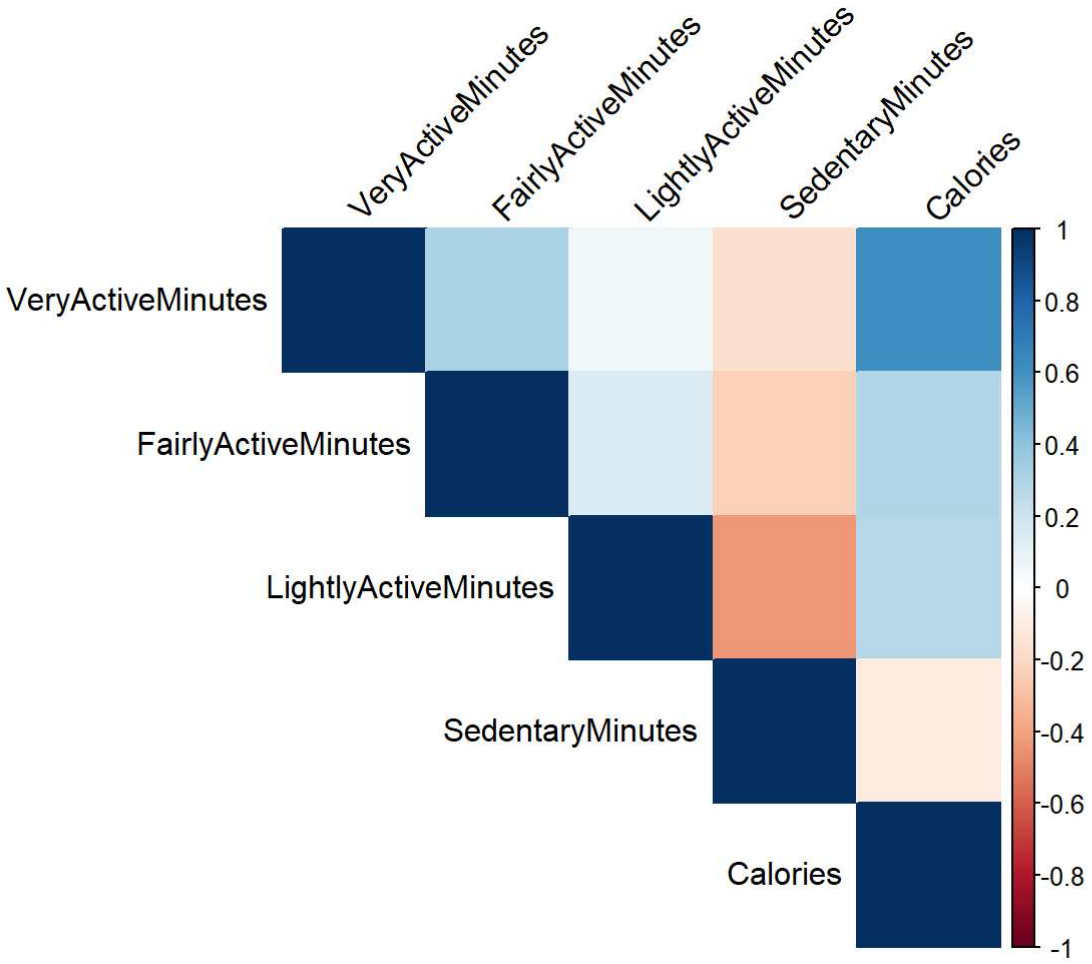
Another important thing was to see how the different intensities of activity influence calories burned. I used another correlation matrix to visualize this.

```
# Select the relevant columns
selected_columns <- daily_activity %>%
  select(VeryActiveDistance, ModeratelyActiveDistance, LightActiveDistance, Calories)

# Calculate the correlation matrix
correlation_matrix <- cor(selected_columns, use = "complete.obs")

# Visualize the correlation matrix with a title
corrplot(correlation_matrix, method = "color", type = "upper",
         tl.col = "black", tl.srt = 45)
```



I did the same thing for minutes

```
# Select the relevant columns
selected_columns <- daily_activity %>%
  select(VeryActiveMinutes, FairlyActiveMinutes, LightlyActiveMinutes, SedentaryMinutes, Calorie
s)

# Calculate the correlation matrix
correlation_matrix <- cor(selected_columns, use = "complete.obs")

# Visualize the correlation matrix with a title
corrplot(correlation_matrix, method = "color", type = "upper",
         tl.col = "black", tl.srt = 45)
```



From these correlation matrices, we can see that Very Intense activity has the greatest effect on calories burned. Moderate and light activity are also positively correlated to calories burned while sedentary minutes are negatively correlated to calories burned.

2) sleep_day dataframe

I started by counting the number of people in this dataset and how many contibutions they each had

```
# Number of unique people (Ids)
unique_people_count <- sleep_day_cleaned %>%
  summarise(unique_people = n_distinct(Id))

# Print the number of unique people
print(unique_people_count)
```

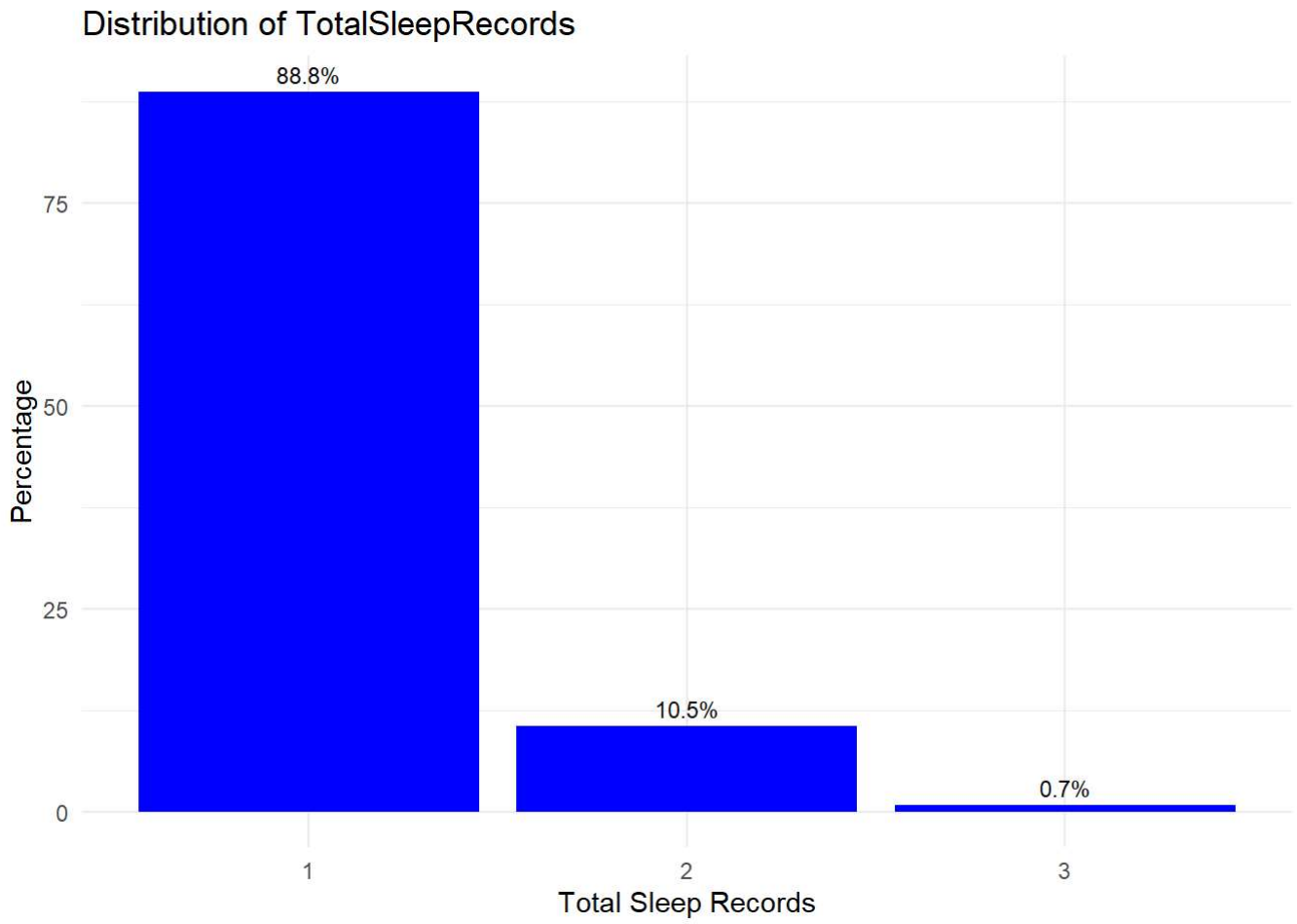
```
##   unique_people
## 1            24
```

```
# Number of unique days each person has a record for
unique_days_per_person <- sleep_day_cleaned %>%
  group_by(Id) %>%
  summarise(unique_days = n_distinct(as.Date(SleepDay, format = "%m/%d/%Y")))

# Print the number of unique days per person
print(unique_days_per_person)
```

```
## # A tibble: 24 × 2
##            Id unique_days
##         <dbl>       <int>
##  1 1503960366          25
##  2 1644430081           4
##  3 1844505072           3
##  4 1927972279           5
##  5 2026352035          28
##  6 2320127002           1
##  7 2347167796          15
##  8 3977333714          28
##  9 4020332650           8
## 10 4319703577          26
## 11 4388161847          23
## 12 4445114986          28
## 13 4558609924           5
## 14 4702921684          27
## 15 5553957443          31
## 16 5577150313          26
## 17 6117666160          18
## 18 6775888955           3
## 19 6962181067          31
## 20 7007744171           2
## 21 7086361926          24
## 22 8053475328           3
## 23 8378563200          31
## 24 8792009665          15
```

Next, I wanted to find out how many times people tend to sleep per day

```
# Calculate the percentages
percentage_data <- sleep_day_cleaned %>%
  group_by(TotalSleepRecords) %>%
  summarise(count = n()) %>%
  mutate(percentage = (count / sum(count)) * 100)

# Create a bar graph with percentages
ggplot(percentage_data, aes(x = factor(TotalSleepRecords), y = percentage)) +
  geom_bar(stat = "identity", fill = "blue") +
  geom_text(aes(label = sprintf("%.1f%%", percentage)), vjust = -0.5, size = 3, color = "black")
+
  labs(title = "Distribution of TotalSleepRecords",
       x = "Total Sleep Records",
       y = "Percentage") +
  theme_minimal() +
  scale_x_discrete(labels = function(x) sprintf("%d", as.integer(x)))
```

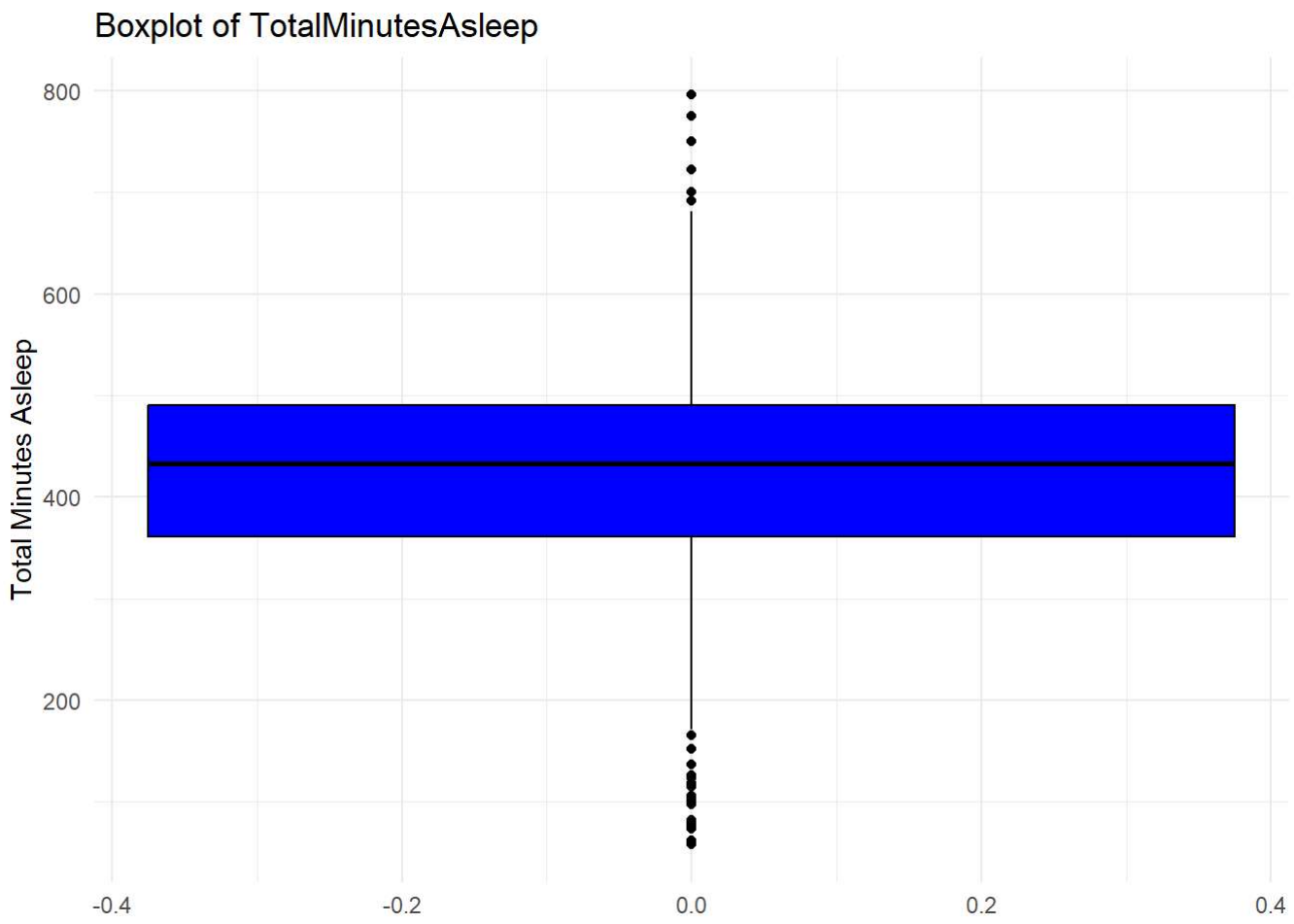


89% of the time, people in the study slept just once per day, 10% of the times they slept twice and very rarely slept thrice

```
# Calculate the average TotalSleepRecords per unique Id
average_sleep_records_per_id <- sleep_day_cleaned %>%
  group_by(Id) %>%
  summarise(average_sleep_records = mean(TotalSleepRecords, na.rm = TRUE))

# Print the result
print(average_sleep_records_per_id)
```

```
## # A tibble: 24 × 2
##            Id average_sleep_records
##         <dbl>                <dbl>
##  1 1503960366                 1.08
##  2 1644430081                 1
##  3 1844505072                 1
##  4 1927972279                 1.6
##  5 2026352035                 1
##  6 2320127002                 1
##  7 2347167796                 1
##  8 3977333714                 1.14
##  9 4020332650                 1
## 10 4319703577                 1.04
## 11 4388161847                 1.30
## 12 4445114986                 1.39
## 13 4558609924                 1
## 14 4702921684                 1.07
## 15 5553957443                 1.23
## 16 5577150313                 1.04
## 17 6117666160                 1.22
## 18 6775888955                 1
## 19 6962181067                 1.10
## 20 7007744171                 1
## 21 7086361926                 1
## 22 8053475328                 1
## 23 8378563200                 1.13
## 24 8792009665                 1
```

```
# Create a boxplot for TotalMinutesAsleep
ggplot(sleep_day_cleaned, aes(y = TotalMinutesAsleep)) +
  geom_boxplot(fill = "blue", color = "black") +
  labs(title = "Boxplot of TotalMinutesAsleep",
       y = "Total Minutes Asleep") +
  theme_minimal()
```



This is corroborated by these averages that shows that each person slept only once on average, with 1.23 being the maximum average number of sleep records per person.

```
# Calculate the average TotalMinutesAsleep
average_minutes_asleep <- mean(sleep_day_cleaned$TotalMinutesAsleep, na.rm = TRUE)

# Print the result
print(average_minutes_asleep)
```

```
## [1] 419.1732
```

```
# Calculate the average TotalMinutesAsleep per person
average_minutes_asleep_per_person <- sleep_day_cleaned %>%
  group_by(Id) %>%
  summarise(average_minutes_asleep = mean(TotalMinutesAsleep, na.rm = TRUE))

# Print the result
print(average_minutes_asleep_per_person)
```

```
## # A tibble: 24 × 2
##            Id average_minutes_asleep
##         <dbl>                  <dbl>
##  1 1503960366                  360.
##  2 1644430081                  294
##  3 1844505072                  652
##  4 1927972279                  417
##  5 2026352035                  506.
##  6 2320127002                   61
##  7 2347167796                  447.
##  8 3977333714                  294.
##  9 4020332650                  349.
## 10 4319703577                  477.
## 11 4388161847                  400.
## 12 4445114986                  385.
## 13 4558609924                  128.
## 14 4702921684                  417.
## 15 5553957443                  463.
## 16 5577150313                  432
## 17 6117666160                  479.
## 18 6775888955                  350.
## 19 6962181067                  448
## 20 7007744171                   68.5
## 21 7086361926                  453.
## 22 8053475328                  297
## 23 8378563200                  445.
## 24 8792009665                  436.
```

The average sleep time is 419 minutes which is about 7 hours.

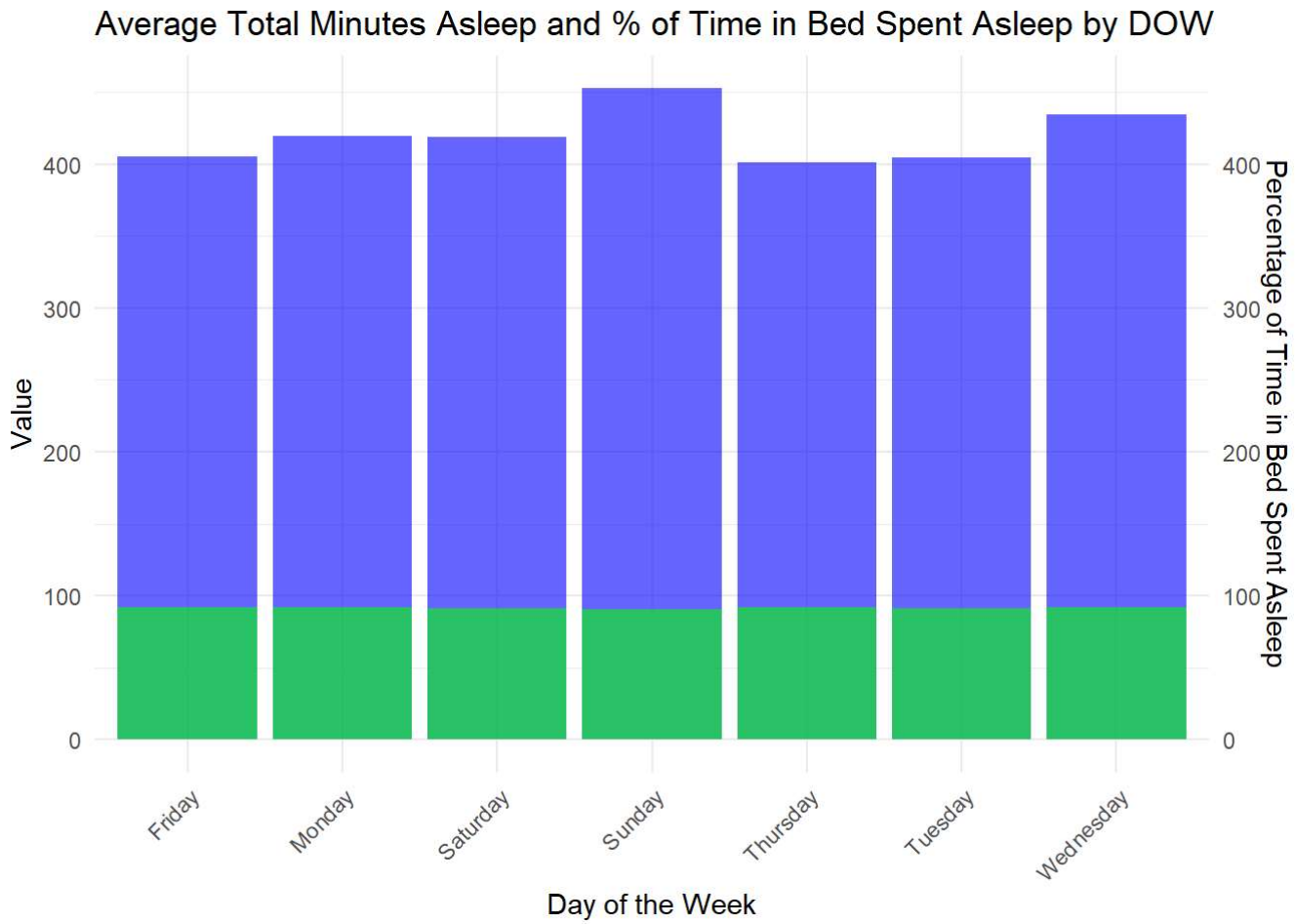
Next, I calculated the amount of time spent in bed actually sleeping.

```
# Calculate the percentage of time spent in bed that each person was actually asleep
percentage_time_asleep_per_person <- sleep_day_cleaned %>%
  mutate(PercentageAsleep = (TotalMinutesAsleep / TotalTimeInBed) * 100) %>%
  group_by(Id) %>%
  summarise(AveragePercentageAsleep = mean(PercentageAsleep, na.rm = TRUE))

# Print the result
print(percentage_time_asleep_per_person)
```

```
## # A tibble: 24 × 2
##            Id AveragePercentageAsleep
##         <dbl>                   <dbl>
##  1 1503960366                    93.6
##  2 1644430081                    88.2
##  3 1844505072                    67.8
##  4 1927972279                    94.7
##  5 2026352035                    94.1
##  6 2320127002                    88.4
##  7 2347167796                    91.0
##  8 3977333714                    63.4
##  9 4020332650                    93.0
## 10 4319703577                    94.7
## 11 4388161847                    94.7
## 12 4445114986                    92.5
## 13 4558609924                    90.7
## 14 4702921684                    95.2
## 15 5553957443                    91.5
## 16 5577150313                    93.9
## 17 6117666160                    94.0
## 18 6775888955                    94.2
## 19 6962181067                    96.1
## 20 7007744171                    95.7
## 21 7086361926                    97.1
## 22 8053475328                    98.5
## 23 8378563200                    91.8
## 24 8792009665                    96.0
```

This is important because it hints at sleep quality. Most of the participants were asleep more than 90% of the time they were in bed on average. The highest was 98.5% and the lowest was 67.8%

```
# Convert the "SleepDay" column to Date format and add a new column for the day of the week
sleep_day_cleaned <- sleep_day_cleaned %>%
  mutate(SleepDay = as.Date(SleepDay, format = "%m/%d/%Y"),
         DayOfWeek = weekdays(SleepDay))

# Calculate the average TotalMinutesAsleep and percentage of Time in Bed spent asleep for each d
ay of the week
average_sleep_per_day_of_week <- sleep_day_cleaned %>%
  group_by(DayOfWeek) %>%
  summarise(AverageTotalMinutesAsleep = mean(TotalMinutesAsleep, na.rm = TRUE),
            AveragePercentageAsleep = mean((TotalMinutesAsleep / TotalTimeInBed) * 100, na.rm =
TRUE))

# Create bar graphs
ggplot(average_sleep_per_day_of_week, aes(x = DayOfWeek)) +
  geom_bar(aes(y = AverageTotalMinutesAsleep), stat = "identity", fill = "blue", alpha = 0.6, po
sition = "dodge") +
  geom_bar(aes(y = AveragePercentageAsleep), stat = "identity", fill = "green", alpha = 0.6, pos
ition = "dodge") +
  labs(title = "Average Total Minutes Asleep and % of Time in Bed Spent Asleep by DOW",
       x = "Day of the Week",
       y = "Value") +
  scale_y_continuous(sec.axis = sec_axis(~ ., name = "Percentage of Time in Bed Spent Asleep"))
+
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_fill_manual(values = c("blue", "green")) +
  scale_alpha_manual(values = c(0.6, 0.6)) +
  guides(fill = guide_legend(title = "Legend", override.aes = list(alpha = 1)))
```



Again, there doesn't seem to be any significant differences between hours spent asleep on different days of the week.On average the people in the study slept slightly more on Sundays than on the other days of the week.

3) weight_log dataframe

```
# Number of unique people (Ids)
unique_people_count <- weight_log %>%
  summarise(unique_people = n_distinct(Id))

# Print the number of unique people
print(unique_people_count)
```

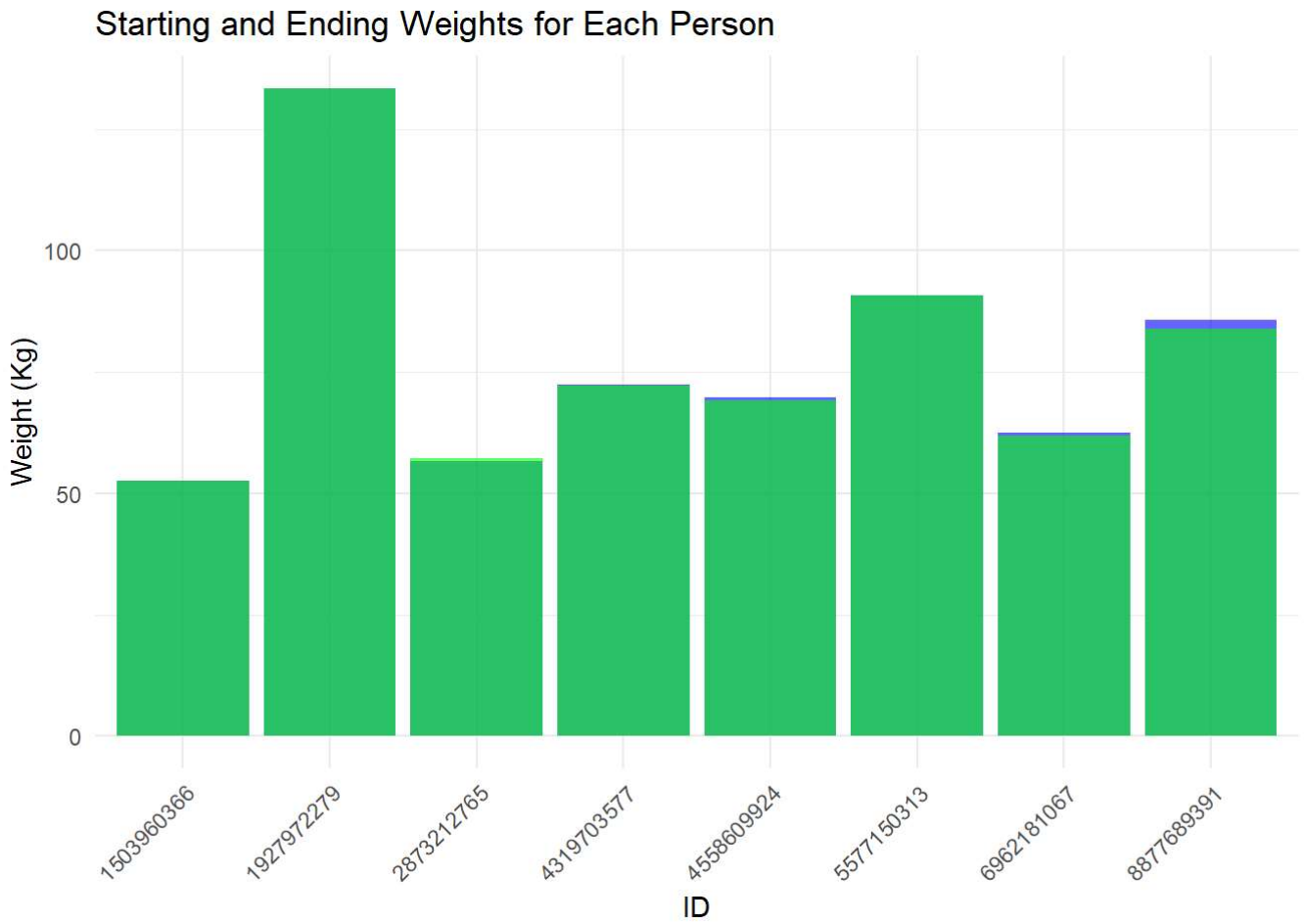
```
##   unique_people
## 1             8
```

```
# Total days with records for each person
total_days_per_person <- weight_log %>%
  group_by(Id) %>%
  summarise(total_days = n_distinct(Date))

# Print the total days with records per person
print(total_days_per_person)
```

```
## # A tibble: 8 × 2
##           Id total_days
##        <dbl>      <int>
## 1 1503960366          2
## 2 1927972279          1
## 3 2873212765          2
## 4 4319703577          2
## 5 4558609924          5
## 6 5577150313          1
## 7 6962181067         30
## 8 8877689391         24
```

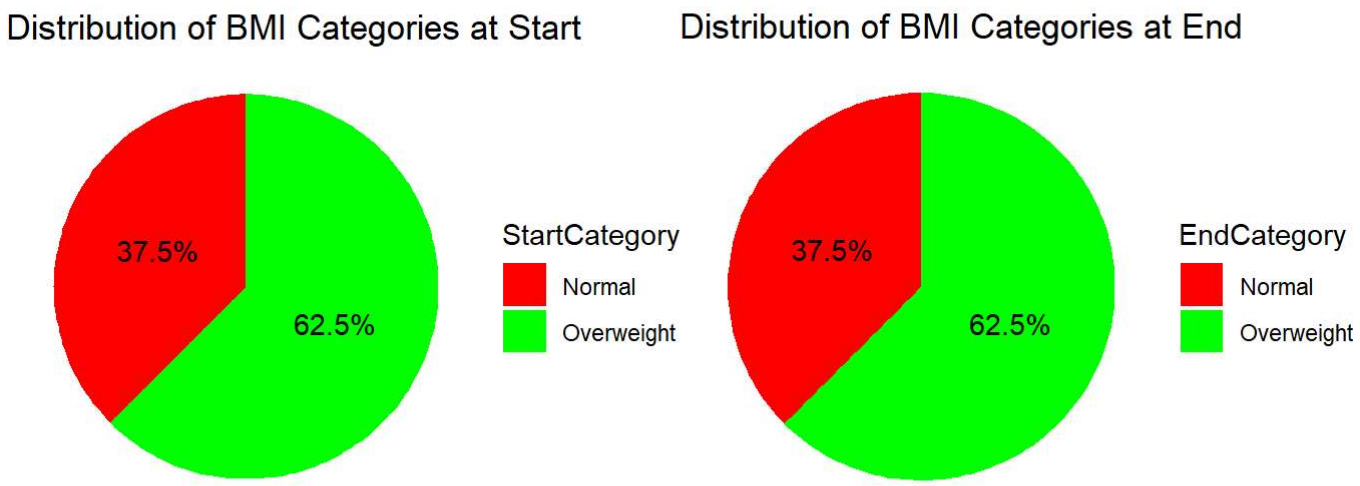
We can see from the results that much fewer people recorded their weight during the study and even fewer did it consistently.

```
# Convert the "Date" column to a proper date format
weight_log$Date <- as.POSIXct(weight_log$Date, format = "%m/%d/%Y %I:%M:%S %p")

# Find the starting weight for each person
starting_weight <- weight_log %>%
  group_by(Id) %>%
  summarise(starting_weight = first(WeightKg))

# Find the ending weight for each person
ending_weight <- weight_log %>%
  group_by(Id) %>%
  summarise(ending_weight = last(WeightKg))

# Combine the starting and ending weights
weight_summary <- left_join(starting_weight, ending_weight, by = "Id")

# Create a bar graph
ggplot(weight_summary, aes(x = factor(Id))) +
  geom_bar(aes(y = starting_weight), stat = "identity", fill = "blue", alpha = 0.6, position = p
osition_dodge(width = 0.8)) +
  geom_bar(aes(y = ending_weight), stat = "identity", fill = "green", alpha = 0.6, position = po
sition_dodge(width = 0.8)) +
  labs(title = "Starting and Ending Weights for Each Person",
       x = "ID",
       y = "Weight (Kg)") +
  scale_fill_manual(values = c("blue", "green")) +
  scale_alpha_manual(values = c(0.6, 0.6)) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Over the month of the study, some of the participants lost a small amount of weight.However, the sample size is too small to try to infer a trend from.

```
# Convert the "Date" column to a proper date format
weight_log$Date <- as.POSIXct(weight_log$Date, format = "%m/%d/%Y %I:%M:%S %p")

# Create a function to categorize BMI
categorize_bmi <- function(bmi) {
  if (bmi < 18.5) {
    return("Underweight")
  } else if (bmi >= 18.5 && bmi <= 25) {
    return("Normal")
  } else {
    return("Overweight")
  }
}

# Categorize BMI for the first and last records
weight_log_summary <- weight_log %>%
  group_by(Id) %>%
  arrange(Date) %>%
  summarise(StartCategory = categorize_bmi(first(BMI)),
            EndCategory = categorize_bmi(last(BMI)))

# Create pie charts for the start and end categories
pie_chart_start <- weight_log_summary %>%
  count(StartCategory) %>%
  mutate(percentage = n / sum(n) * 100) %>%
  ggplot(aes(x = "", y = n, fill = StartCategory, label = paste0(percentage, "%"))) +
  geom_bar(stat = "identity", width = 1) +
  geom_text(aes(label = paste0(percentage, "%")), position = position_stack(vjust = 0.5)) +
  coord_polar(theta = "y") +
  scale_fill_manual(values = c("red", "green", "blue")) +  # Red for "Overweight," green for "No
rmal"
  labs(title = "Distribution of BMI Categories at Start") +
  theme_void() +
  theme(legend.position = "right")

pie_chart_end <- weight_log_summary %>%
  count(EndCategory) %>%
  mutate(percentage = n / sum(n) * 100) %>%
  ggplot(aes(x = "", y = n, fill = EndCategory, label = paste0(percentage, "%"))) +
  geom_bar(stat = "identity", width = 1) +
  geom_text(aes(label = paste0(percentage, "%")), position = position_stack(vjust = 0.5)) +
  coord_polar(theta = "y") +
  scale_fill_manual(values = c("red", "green", "blue")) +  # Red for "Overweight," green for "No
rmal"
  labs(title = "Distribution of BMI Categories at End") +
  theme_void() +
  theme(legend.position = "right")

# Display the pie charts
gridExtra::grid.arrange(pie_chart_start, pie_chart_end, ncol = 2)
```



We can see that 37.5% of the participants were overweight and the rest were normal weight.

4) hourly_exercises dataframe

```
# Number of unique people (Ids)
unique_people_count <- hourly_exercises %>%
  summarise(unique_people = n_distinct(Id))

# Print the number of unique people
print(unique_people_count)
```

```
##   unique_people
## 1            33
```

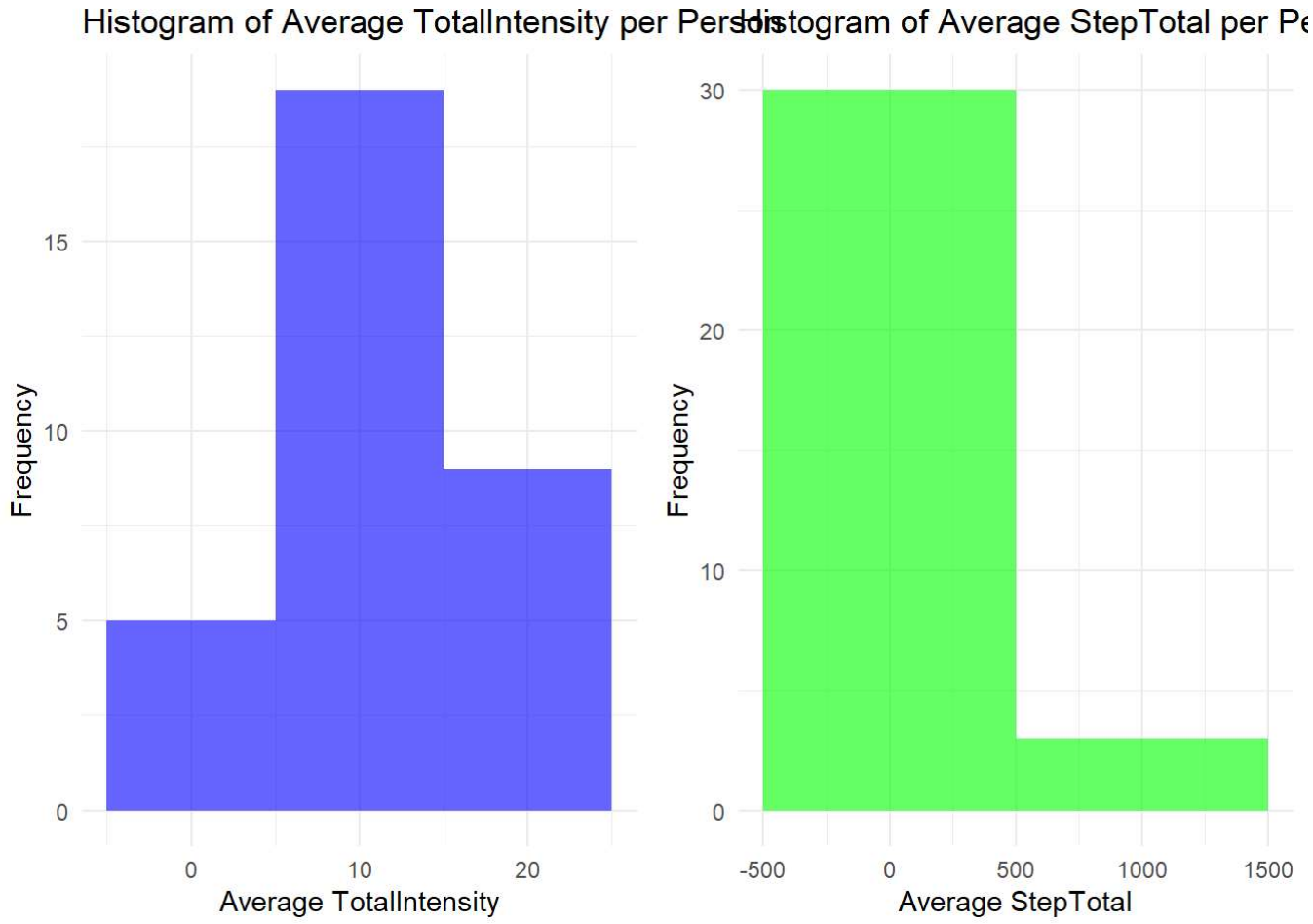
```
# Number of days and unique hours per person
days_and_hours_per_person <- hourly_exercises %>%
  group_by(Id) %>%
  summarise(total_days = n_distinct(as.Date(ActivityHour)),
            unique_hours = n_distinct(ActivityHour))

# Print the number of days and unique hours per person
print(days_and_hours_per_person)
```

```
## # A tibble: 33 × 3
##            Id total_days unique_hours
##         <dbl>      <int>        <int>
##  1 1503960366         13          717
##  2 1624580081         14          736
##  3 1644430081         13          708
##  4 1844505072         14          731
##  5 1927972279         14          736
##  6 2022484408         14          736
##  7 2026352035         14          736
##  8 2320127002         14          735
##  9 2347167796          2          414
## 10 2873212765         14          736
## 11 3372868164          3          472
## 12 3977333714         12          696
## 13 4020332650         14          732
## 14 4057192912          2           88
## 15 4319703577         14          724
## 16 4388161847         14          735
## 17 4445114986         14          735
## 18 4558609924         14          736
## 19 4702921684         14          731
## 20 5553957443         14          730
## 21 5577150313         13          708
## 22 6117666160         11          660
## 23 6290855005         11          665
## 24 6775888955          9          610
## 25 6962181067         14          732
## 26 7007744171          9          601
## 27 7086361926         14          733
## 28 8053475328         14          735
## 29 8253242879          2          431
## 30 8378563200         14          735
## 31 8583815059         13          718
## 32 8792009665         11          672
## 33 8877689391         14          735
```

```
# Calculate the average TotalIntensity and average StepTotal per person
average_intensity_and_steps_per_person <- hourly_exercises %>%
  group_by(Id) %>%
  summarise(average_total_intensity = mean(TotalIntensity, na.rm = TRUE),
            average_step_total = mean(StepTotal, na.rm = TRUE))

# Print the result
print(average_intensity_and_steps_per_person)
```

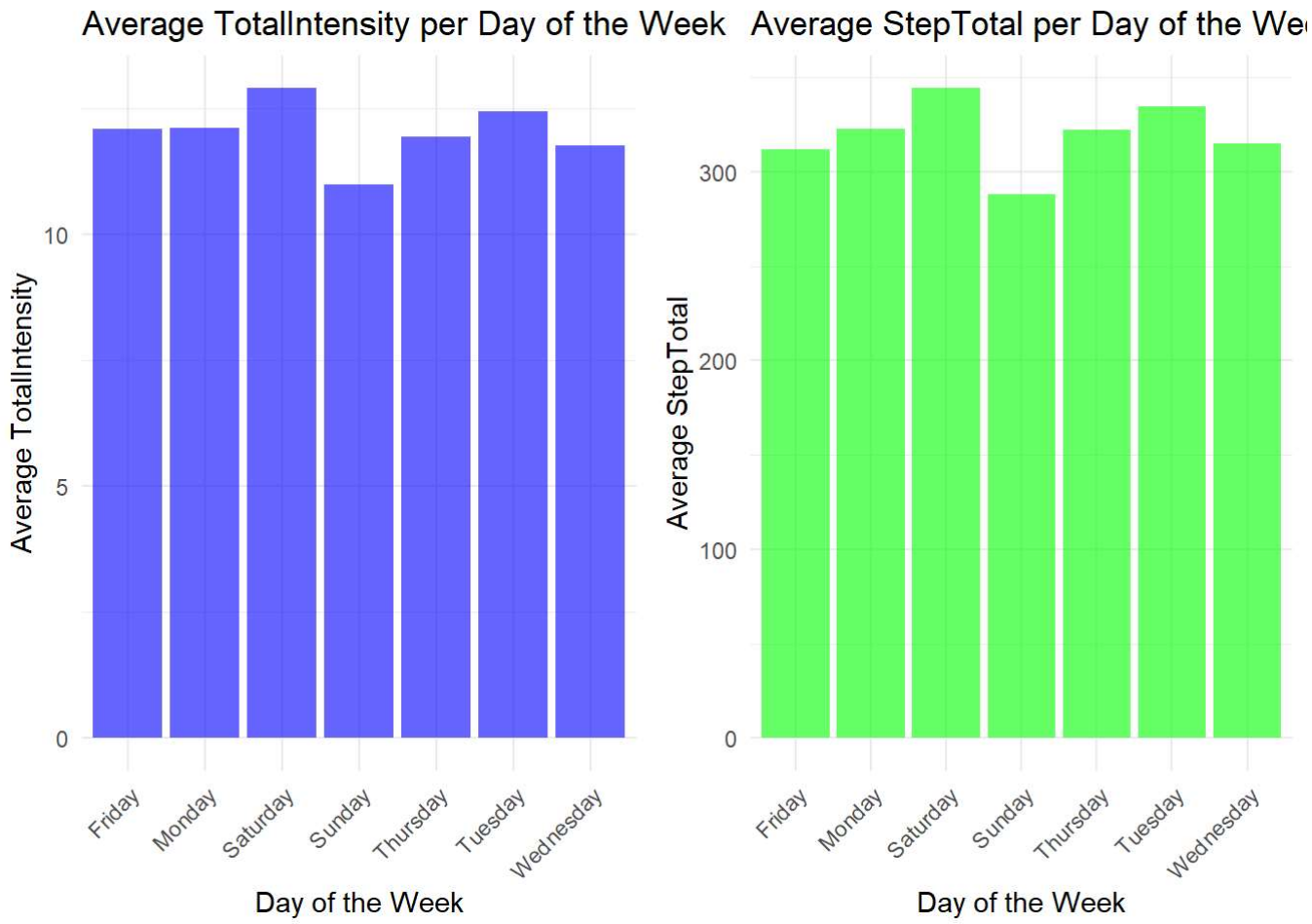
```
## # A tibble: 33 × 3
##            Id average_total_intensity average_step_total
##         <dbl>                   <dbl>              <dbl>
##  1 1503960366                   16.2               522.
##  2 1624580081                    8.04              242.
##  3 1644430081                   10.5               308.
##  4 1844505072                    5.02              109.
##  5 1927972279                    1.86               38.6
##  6 2022484408                   17.0               478.
##  7 2026352035                   10.8               234.
##  8 2320127002                    8.74              199.
##  9 2347167796                   14.5               414.
## 10 2873212765                   15.1               318.
## 11 3372868164                   15.4               290.
## 12 3977333714                   15.2               472.
## 13 4020332650                    4.36               93.4
## 14 4057192912                    4.90              174.
## 15 4319703577                   11.3               289.
## 16 4388161847                   14.3               435.
## 17 4445114986                    9.79              202.
## 18 4558609924                   14.4               323.
## 19 4702921684                   12.9               363.
## 20 5553957443                   12.8               366.
## 21 5577150313                   19.9               351.
## 22 6117666160                   12.5               281.
## 23 6290855005                   10.6               246.
## 24 6775888955                    4.37              107.
## 25 6962181067                   14.9               415.
## 26 7007744171                   17.6               490.
## 27 7086361926                   13.6               392.
## 28 8053475328                   17.9               622.
## 29 8253242879                    9.11              285.
## 30 8378563200                   14.9               367.
## 31 8583815059                    9.13              245.
## 32 8792009665                    4.43               80.0
## 33 8877689391                   19.1               674.
```

```
# Calculate the average TotalIntensity and average StepTotal per person
average_intensity_and_steps_per_person <- hourly_exercises %>%
  group_by(Id) %>%
  summarise(average_total_intensity = mean(TotalIntensity, na.rm = TRUE),
            average_step_total = mean(StepTotal, na.rm = TRUE))

# Create a histogram for average TotalIntensity
histogram_intensity <- ggplot(average_intensity_and_steps_per_person, aes(x = average_total_inte
nsity)) +
  geom_histogram(binwidth = 10, fill = "blue", alpha = 0.6) +
  labs(title = "Histogram of Average TotalIntensity per Person",
       x = "Average TotalIntensity",
       y = "Frequency") +
  theme_minimal()

# Create a histogram for average StepTotal
histogram_steps <- ggplot(average_intensity_and_steps_per_person, aes(x = average_step_total)) +
  geom_histogram(binwidth = 1000, fill = "green", alpha = 0.6) +
  labs(title = "Histogram of Average StepTotal per Person",
       x = "Average StepTotal",
       y = "Frequency") +
  theme_minimal()

# Display the histograms
gridExtra::grid.arrange(histogram_intensity, histogram_steps, ncol = 2)
```



From the histograms, we can see that the majority of people took 500 steps or less at high intensity during exercise.

```
# Convert the "ActivityHour" column to a proper date format and extract the day of the week
hourly_exercises <- hourly_exercises %>%
  mutate(ActivityHour = as.POSIXct(ActivityHour, format = "%m/%d/%Y %I:%M:%S %p"),
         DayOfWeek = weekdays(ActivityHour))

# Calculate the average TotalIntensity and average StepTotal per day of the week
average_intensity_per_day <- hourly_exercises %>%
  group_by(DayOfWeek) %>%
  summarise(average_total_intensity = mean(TotalIntensity, na.rm = TRUE))

average_steps_per_day <- hourly_exercises %>%
  group_by(DayOfWeek) %>%
  summarise(average_step_total = mean(StepTotal, na.rm = TRUE))

# Create a bar plot for average TotalIntensity per day of the week
bar_plot_intensity <- ggplot(average_intensity_per_day, aes(x = DayOfWeek, y = average_total_int
ensity)) +
  geom_bar(stat = "identity", fill = "blue", alpha = 0.6) +
  labs(title = "Average TotalIntensity per Day of the Week",
       x = "Day of the Week",
       y = "Average TotalIntensity") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Create a bar plot for average StepTotal per day of the week
bar_plot_steps <- ggplot(average_steps_per_day, aes(x = DayOfWeek, y = average_step_total)) +
  geom_bar(stat = "identity", fill = "green", alpha = 0.6) +
  labs(title = "Average StepTotal per Day of the Week",
       x = "Day of the Week",
       y = "Average StepTotal") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Display the bar plots
gridExtra::grid.arrange(bar_plot_intensity, bar_plot_steps, ncol = 2)
```



Again, there seems to be constant exercise over the week, except for Sunday which has slightly less than the rest and Saturday which has slightly more.

5) heart_rate dataframe

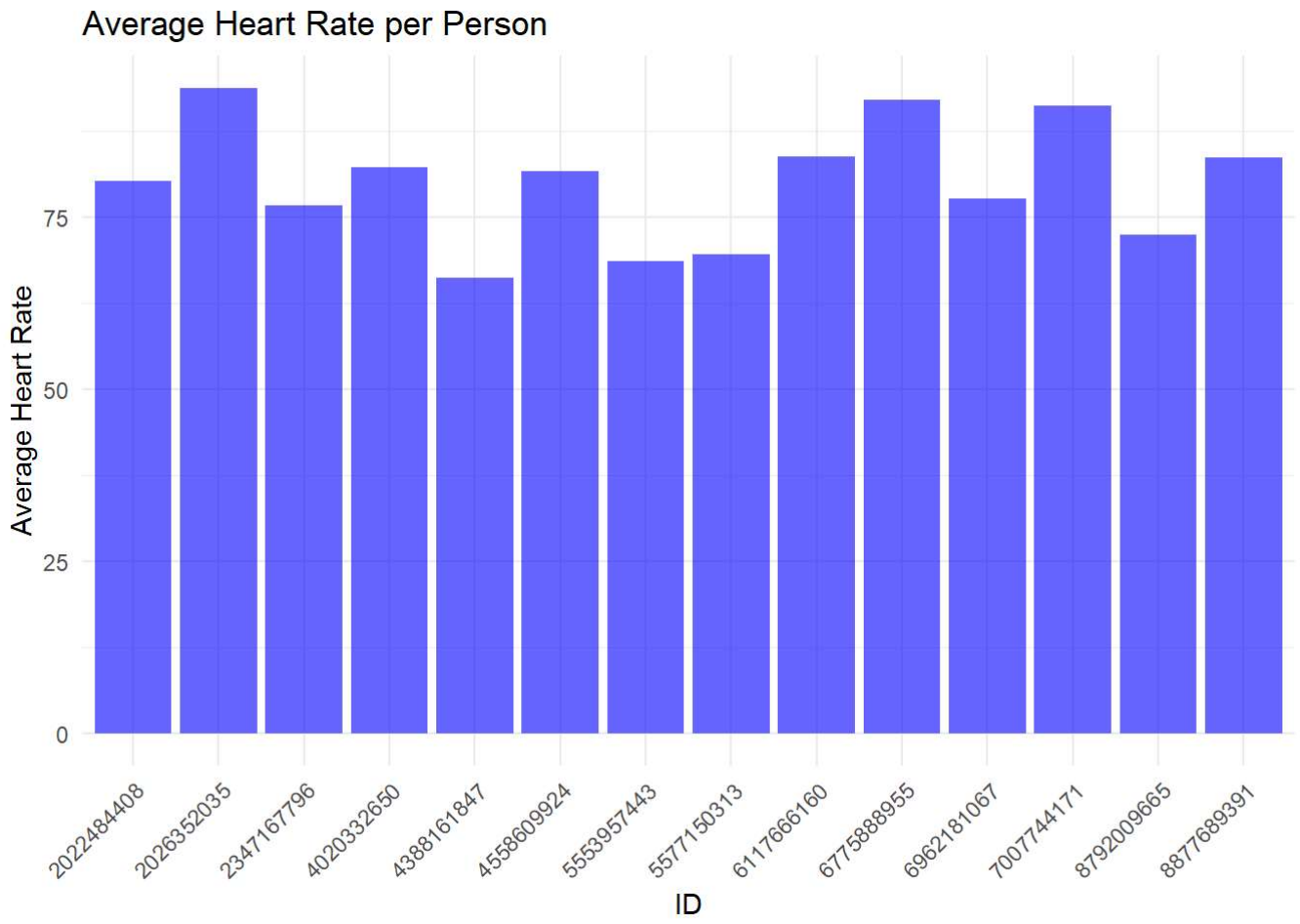
```
# Number of unique people (Ids)
unique_people_count <- heart_rate %>%
  summarise(unique_people = n_distinct(Id))

# Print the number of unique people
print(unique_people_count)
```

```
##   unique_people
## 1            14
```

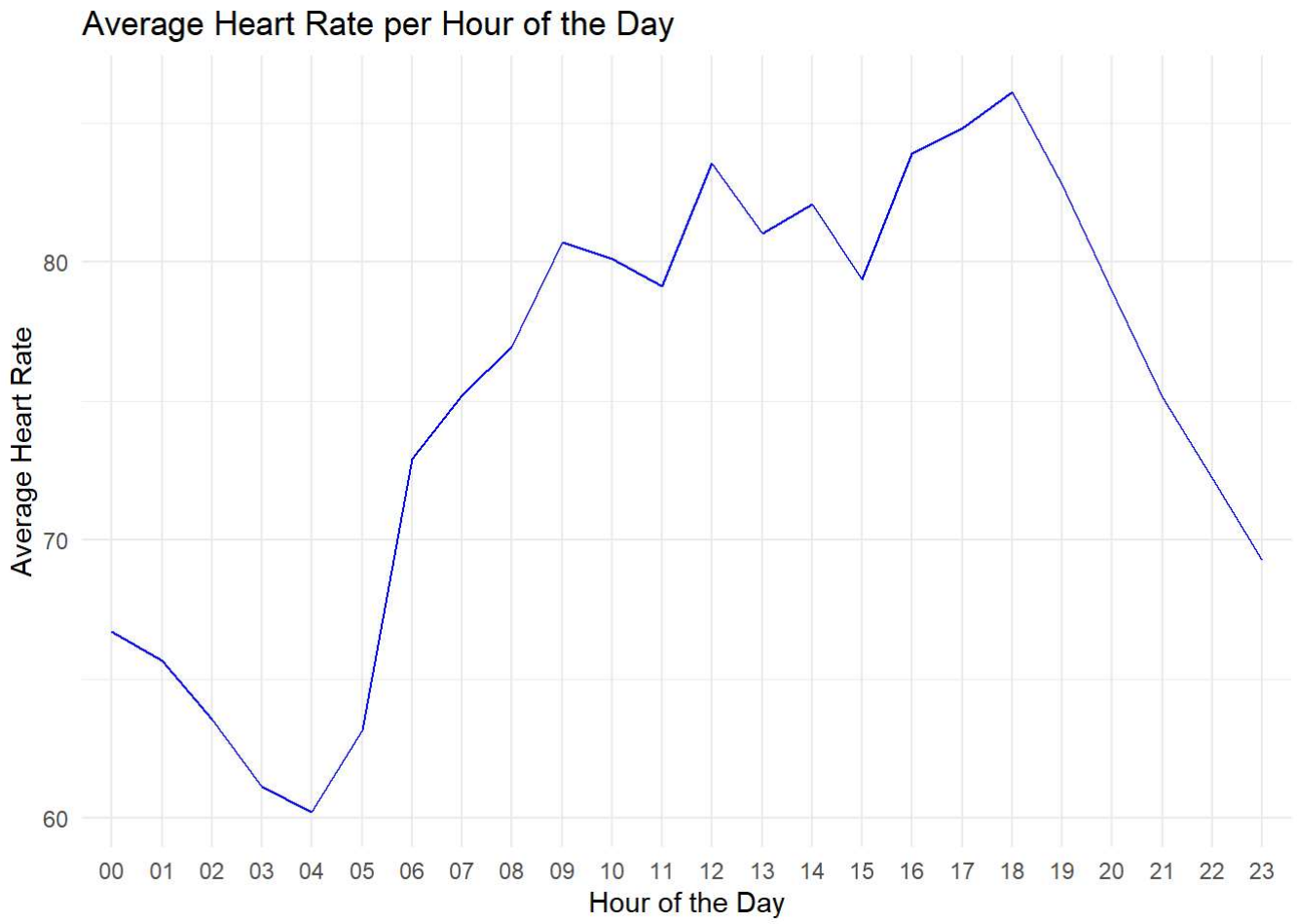
```
# Number of unique records per person
unique_records_per_person <- heart_rate %>%
  group_by(Id) %>%
  summarise(unique_records = n_distinct(Time))

# Print the number of unique records per person
print(unique_records_per_person)
```

```
## # A tibble: 14 × 2
##            Id unique_records
##         <dbl>          <int>
##  1 2022484408         154104
##  2 2026352035           2490
##  3 2347167796         152683
##  4 4020332650         285461
##  5 4388161847         249748
##  6 4558609924         192168
##  7 5553957443         255174
##  8 5577150313         248560
##  9 6117666160         158899
## 10 6775888955          32771
## 11 6962181067         266326
## 12 7007744171         133592
## 13 8792009665         122841
## 14 8877689391         228841
```

```
# Calculate the average heart rate per person
average_heart_rate_per_person <- heart_rate %>%
  group_by(Id) %>%
  summarise(average_heart_rate = mean(Value, na.rm = TRUE))

# Create a bar plot for average heart rate per person
bar_plot_heart_rate <- ggplot(average_heart_rate_per_person, aes(x = factor(Id), y = average_hea
rt_rate)) +
  geom_bar(stat = "identity", fill = "blue", alpha = 0.6) +
  labs(title = "Average Heart Rate per Person",
       x = "ID",
       y = "Average Heart Rate") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Display the bar plot
print(bar_plot_heart_rate)
```

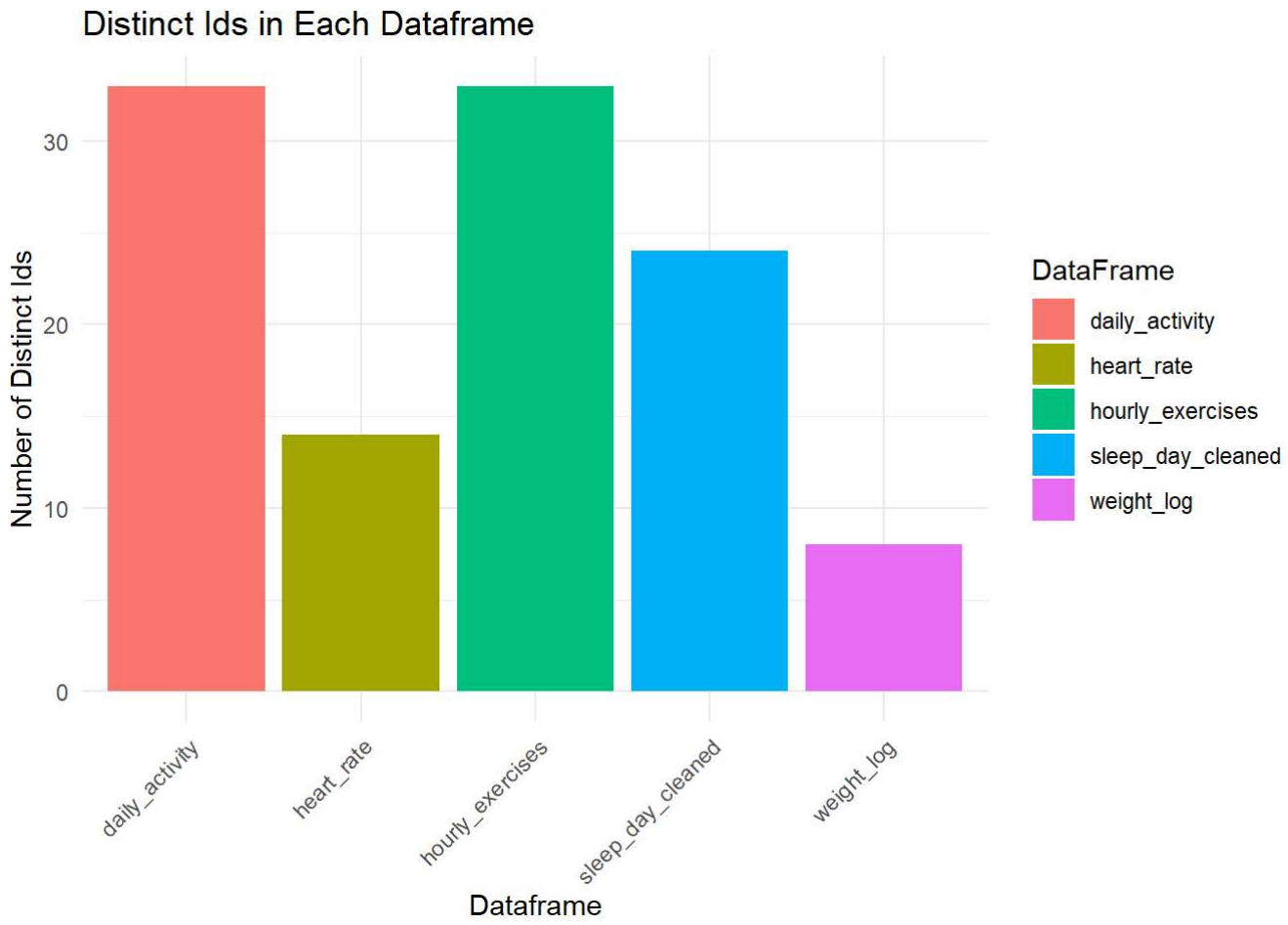



```
# Convert the "Time" column to a proper datetime format
heart_rate$Time <- as.POSIXct(heart_rate$Time, format = "%m/%d/%Y %I:%M:%S %p")

# Extract the hour of the day from the "Time" column
heart_rate$HourOfDay <- format(heart_rate$Time, format = "%H")

# Calculate the average heart rate for each hour of the day
average_heart_rate_per_hour <- heart_rate %>%
  group_by(HourOfDay) %>%
  summarise(average_heart_rate = mean(Value, na.rm = TRUE))

# Create a line plot for average heart rate per hour of the day
line_plot_heart_rate <- ggplot(average_heart_rate_per_hour, aes(x = HourOfDay, y = average_heart
_rate, group = 1)) +
  geom_line(color = "blue") +
  labs(title = "Average Heart Rate per Hour of the Day",
       x = "Hour of the Day",
       y = "Average Heart Rate") +
  theme_minimal()

# Display the line plot
print(line_plot_heart_rate)
```


Average Heart Rate per Hour of the Day
Average Heart Rate
80
70
60
00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour of the Day

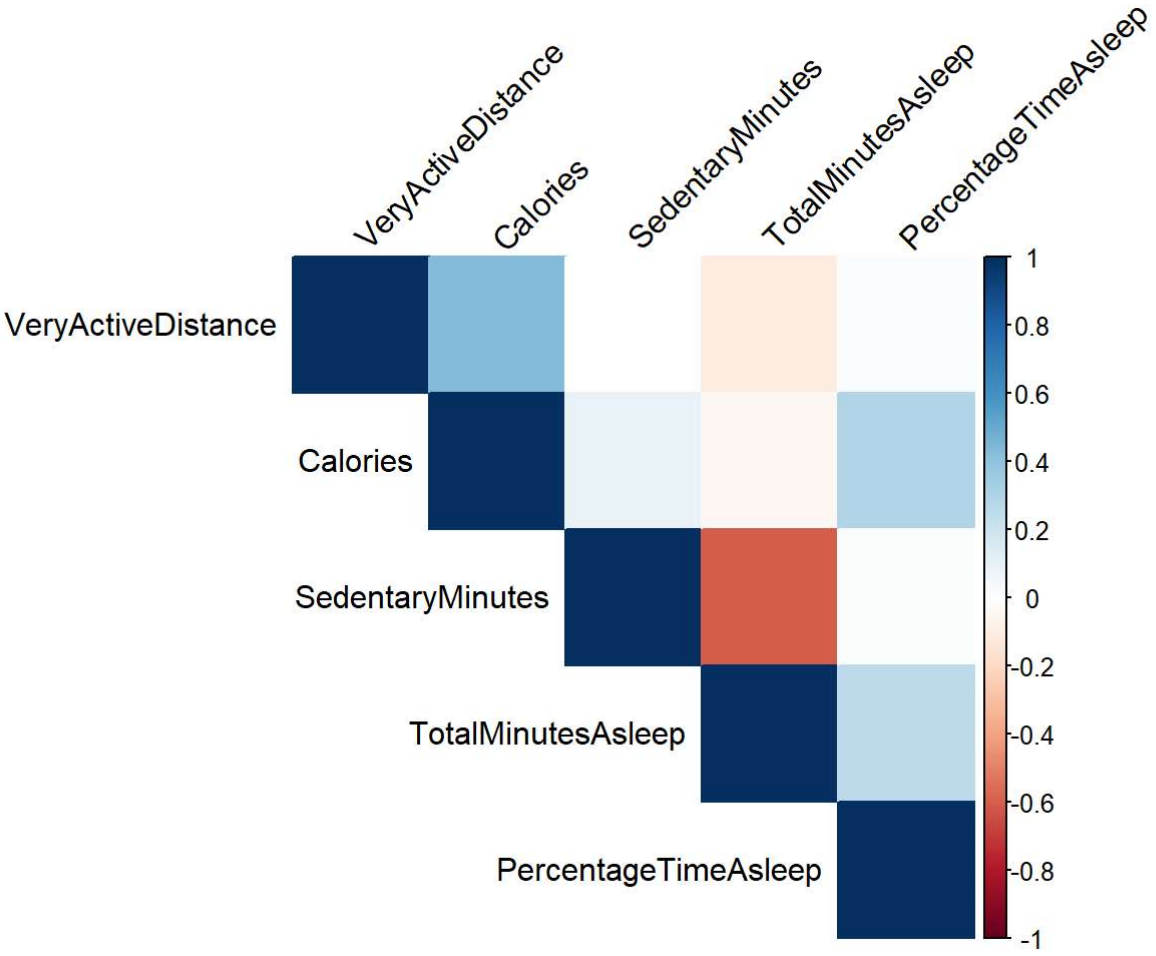
6) Cross Dataframe analysis

```
# Create a data frame with the number of distinct Ids for each dataframe
distinct_ids <- data.frame(
  DataFrame = c("daily_activity", "sleep_day_cleaned", "weight_log", "hourly_exercises", "heart_
rate"),
  DistinctIds = c(
    n_distinct(daily_activity$Id),
    n_distinct(sleep_day_cleaned$Id),
    n_distinct(weight_log$Id),
    n_distinct(hourly_exercises$Id),
    n_distinct(heart_rate$Id)
  )
)

# Create a bar graph
bar_graph <- ggplot(distinct_ids, aes(x = DataFrame, y = DistinctIds, fill = DataFrame)) +
  geom_bar(stat = "identity") +
  labs(title = "Distinct Ids in Each Dataframe",
       x = "Dataframe",
       y = "Number of Distinct Ids") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Display the bar graph
print(bar_graph)
```



This graph gives us the clearest hint of what people are more likely to keep track of. The majority keep track of their daily activity and exercises. Slightly fewer keep track of their sleep, and even less than that monitor their heart rate. Very few measured their weight and recorded their fat etc.

```
# Convert the date columns to a common format
daily_activity$ActivityDate <- as.Date(daily_activity$ActivityDate, format = "%m/%d/%Y")
sleep_day_cleaned$SleepDay <- as.Date(sleep_day_cleaned$SleepDay)

# Merge the two dataframes based on 'Id' and matching dates
merged_data <- inner_join(daily_activity, sleep_day_cleaned, by = c("Id", "ActivityDate" = "Slee
pDay"))

# Calculate the percentage of time spent in bed actually asleep
merged_data <- merged_data %>%
  mutate(PercentageTimeAsleep = (TotalMinutesAsleep / TotalTimeInBed) * 100)

# Select the relevant columns for the correlation matrix
selected_columns <- merged_data %>%
  select(
    VeryActiveDistance,
    Calories,
    SedentaryMinutes,
    TotalMinutesAsleep,
    PercentageTimeAsleep
  )

# Calculate the correlation matrix
correlation_matrix <- cor(selected_columns, use = "complete.obs")

corrplot(correlation_matrix, method = "color", type = "upper", tl.col = "black", tl.srt = 45)
```




```
# Print the correlation matrix
print(correlation_matrix)
```

```
##                       VeryActiveDistance    Calories SedentaryMinutes
## VeryActiveDistance          1.000000000  0.43930152      0.007169507
## Calories                    0.439301520  1.00000000      0.098655706
## SedentaryMinutes            0.007169507  0.09865571      1.000000000
## TotalMinutesAsleep         -0.102781443 -0.03169899     -0.601073140
## PercentageTimeAsleep        0.020414005  0.29486181      0.019756470
##                       TotalMinutesAsleep PercentageTimeAsleep
## VeryActiveDistance          -0.10278144           0.02041401
## Calories                    -0.03169899           0.29486181
## SedentaryMinutes            -0.60107314           0.01975647
## TotalMinutesAsleep           1.00000000           0.26452684
## PercentageTimeAsleep         0.26452684           1.00000000
```

From the correlation matrix above, we can see that burning a lot of calories seems to improve the “quality of sleep” and increasing sedentary minutes during the day decreases the total sleep time.

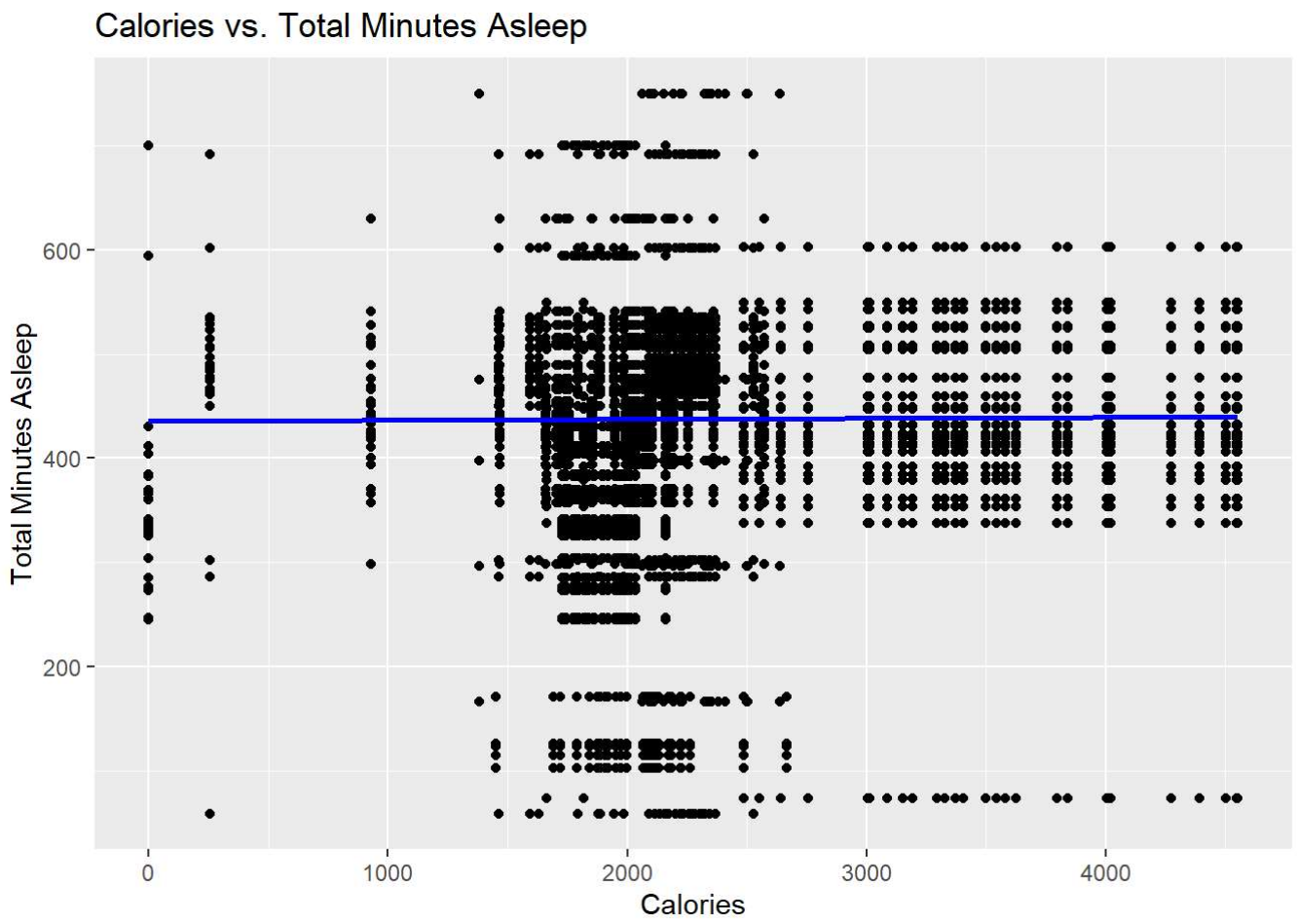
```
# Convert date columns to a common format
daily_activity$ActivityDate <- as.Date(daily_activity$ActivityDate, format = "%m/%d/%Y")
sleep_day_cleaned$SleepDay <- as.Date(sleep_day_cleaned$SleepDay)
weight_log$Date <- as.POSIXct(weight_log$Date, format = "%Y-%m-%d %H:%M:%S")

# Join the dataframes based on 'Id' and matching dates
merged_data <- daily_activity %>%
  inner_join(sleep_day_cleaned, by = c("Id")) %>%
  inner_join(weight_log, by = c("Id"))
```

```
## Warning in inner_join(., sleep_day_cleaned, by = c("Id")): Detected an unexpected many-to-man
y relationship between `x` and `y`.
## ℹ Row 1 of `x` matches multiple rows in `y`.
## ℹ Row 1 of `y` matches multiple rows in `x`.
## ℹ If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```
## Warning in inner_join(., weight_log, by = c("Id")): Detected an unexpected many-to-many relat
ionship between `x` and `y`.
## ℹ Row 1 of `x` matches multiple rows in `y`.
## ℹ Row 1 of `y` matches multiple rows in `x`.
## ℹ If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

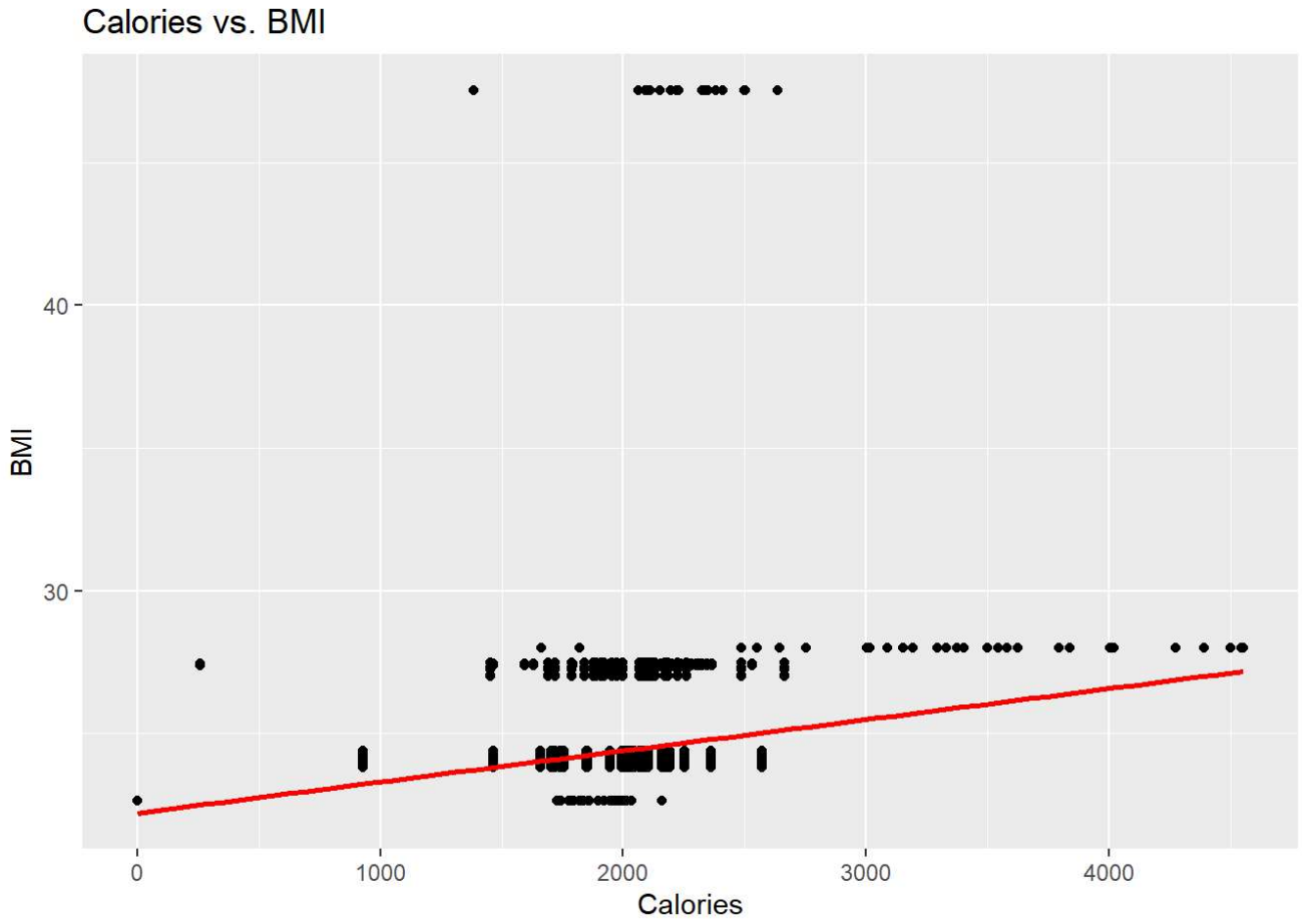
```
# Select the relevant columns
selected_columns <- merged_data %>%
  select(Calories, TotalMinutesAsleep, WeightKg, BMI)

# Create scatterplot with regression lines
scatterplot_calories_vs_minutes_asleep <- ggplot(selected_columns, aes(x = Calories, y = TotalMi
nutesAsleep)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +  # Add linear regression line
  labs(title = "Calories vs. Total Minutes Asleep",
       x = "Calories",
       y = "Total Minutes Asleep")

scatterplot_calories_vs_bmi <- ggplot(selected_columns, aes(x = Calories, y = BMI)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "red") +  # Add linear regression line
  labs(title = "Calories vs. BMI",
       x = "Calories",
       y = "BMI")

# Display the scatterplots with regression lines
print(scatterplot_calories_vs_minutes_asleep)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
print(scatterplot_calories_vs_bmi)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



End