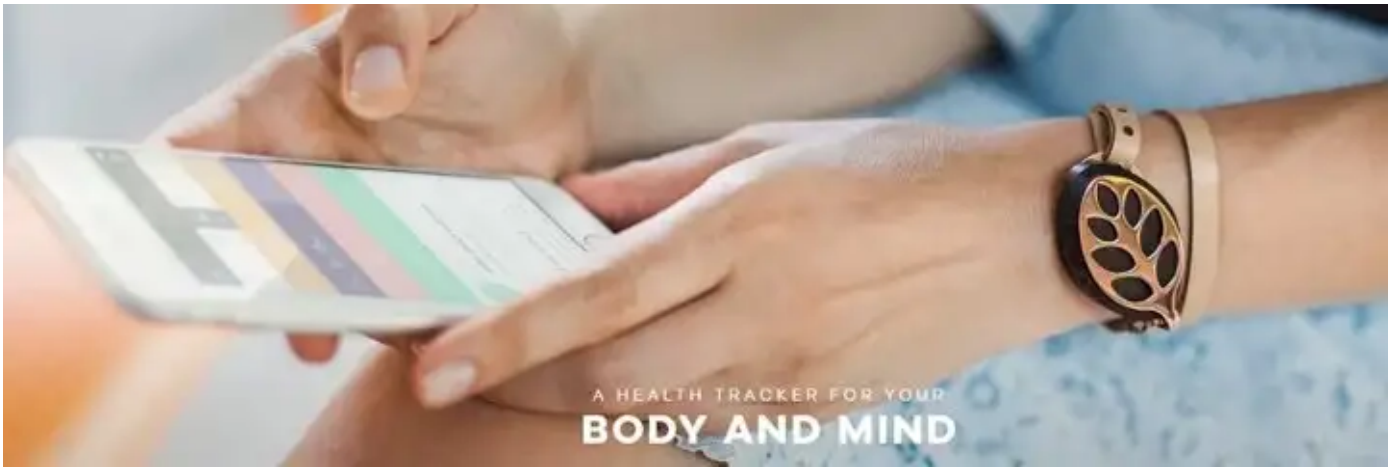
Ahmed Anees Zaveri

Dec 17 · 10 min read · Listen

Bellabeat — Google Data Analytics Capstone Project



Introduction

Bellabeat is a high-tech manufacturer of health-focused products for women. Collecting data on activity, sleep, stress, and reproductive health has allowed Bellabeat to empower women with knowledge about their own health and habits. It is currently a small successful business but possesses the potential to become a greater player in the global smart device market in the future. The company was founded in 2013 by Urška Sršen and Sando Mur, the former also serving as the company's Chief Creative Officer. Urška believes that analyzing smart device fitness data could help unlock new growth opportunities for the company.

Scenario

I am a junior data analyst working on the marketing analyst team at Bellabeat. I have been asked to focus on one of Bellabeat's products and analyze smart device data to gain insight into how consumers are using their smart devices. The insights I discover will then help guide marketing strategy for the company. Finally, I will present my analysis to the Bellabeat executive team along with my high-level recommendations for Bellabeat's marketing strategy. In order to answer these key business questions, the six steps of the data analyst process will be followed: **Ask**, **Prepare**, **Process**, **Analyze**, **Share** and **Act**.

Step 1: Ask

In the ask phase, the main idea is to simply ask the right questions which would help me in making sense of the data as well as the problem I have been tasked to solve. A few questions which need to be answered are:

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?

Business Task:

Analyze smart device usage data in order to gain insight into how consumers use **non-Bellabeat** smart devices.

Key Stakeholders:

- **Urška Sršen** — Bellabeat's cofounder and Chief Creative Officer
- **Sando Mur** — Mathematician and Bellabeat's cofounder; key member of the Bellabeat executive team
- **Bellabeat marketing analytics team** — A team of data analysts responsible for collecting, analyzing, and reporting data that helps guide Bellabeat's marketing strategy.

Step 2: Prepare

Data Source:

In order to explore smart device users' daily habits, public data available at Kaggle has

been used. This ***dataset*** belongs to a non-Bellabeat smart device called the **Fitbit Fitness Tracker**.

The dataset was generated by respondents to a distributed survey via Amazon Mechanical Turk between 03.12.2016–05.12.2016. It contains personal fitness track from thirty fitbit users. Thirty eligible Fitbit users consented to the submission of personal tracker data, including minute-level output for physical activity, heart rate, and sleep monitoring. It includes information about daily activity, steps, and heart rate that can be used to explore users' habits. Individual reports can be parsed by export session ID (column A) or timestamp (column B). Variation between output represents use of different types of Fitbit trackers and individual tracking behaviors / preferences.

Quality of Data:
This data has its fair share of limitations. Firstly, the sample size is very small to make any conclusive analysis. Moreover, many key characteristics of the participants such as age, gender, location are absent. Concisely put, the data does not entirely conform to the ROCCC standards. Even though it is **R**eliable, **O**riginial and **C**omprehensive but it is neither **C**urrent not is **C**ited.

Datasets used:
For this analysis, the datasets for daily activity, daily calories, daily intensities, daily steps, heartrate by seconds, hourly calories, hourly steps, sleep day and weight log information have been used.

Step 3: Process

Tool Employed:
R Studio has been used in order to complete this analysis. R allows the full spectrum of tasks one needs to perform in Data Analysis under one umbrella. From Data Cleaning to Analyzing the Data to Data Visualization, R can accomplish each of these tasks efficiently.

Installing and Loading Packages:

```
#Installing Packages
install.packages("tidyverse")
install.packages("lubridate")
install.packages("here")
install.packages("skimr")
install.packages("janitor")
install.packages("cowplot")
install.packages("hms")
install.packages("ggrepel")

#Loading Packages
library(tidyverse)
library(lubridate)
library(dplyr)
library(ggplot2)
library(tidyr)
library(here)
library(skimr)
library(janitor)
library(cowplot)
library(plotly)
library(hms)
library(ggrepel)
library(knitr)
```

Importing the Datasets and Viewing the Dataframes:

```
#Reading CSV Files
daily_activity <- read_csv("dailyActivity_merged.csv")
head(daily_activity)
colnames(daily_activity)
str(daily_activity)
glimpse(daily_activity)

daily_calories <- read_csv("dailyCalories_merged.csv")
head(daily_calories)
colnames(daily_calories)
str(daily_calories)
glimpse(daily_activity)

daily_intensities <- read_csv("dailyIntensities_merged.csv")
head(daily_intensities)
colnames(daily_intensities)
str(daily_intensities)
glimpse(daily_intensities)

heartrate <- read_csv("heartrate_seconds_merged.csv")
head(heartrate)
colnames(heartrate)
str(heartrate)
glimpse(heartrate)
```

```
hourly_calories <- read_csv("hourlyCalories_merged.csv")
head(hourly_calories)
colnames(hourly_calories)
str(hourly_calories)
glimpse(hourly_calories)

hourly_steps <- read_csv("hourlySteps_merged.csv")
head(hourly_steps)
colnames(hourly_steps)
str(hourly_steps)
glimpse(hourly_steps)

sleep <- read_csv("sleepDay_merged.csv")
head(sleep)
colnames(sleep)
str(sleep)
glimpse(sleep)

weight <- read_csv("weightLogInfo_merged.csv")
head(weight)
colnames(weight)
str(weight)
glimpse(weight)
```

Some Examples of Results:

```
>   head(daily_activity)
# A tibble: 6 × 15
        Id Activit…¹ Total…² Total…³ Track…⁴ Logge…⁵ VeryA…⁶ Moder…⁷ Light…⁸ Seden…⁹ VeryA…˟ Fairl…˟ Light…˟
     <dbl> <chr>       <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
1 1503960366 4/12/2016   13162    8.5     8.5       0    1.88   0.550    6.06       0      25      13      328
2 1503960366 4/13/2016   10735    6.97    6.97      0    1.57   0.690    4.71       0      21      19      217
3 1503960366 4/14/2016   10460    6.74    6.74      0    2.44   0.400    3.91       0      30      11      181
4 1503960366 4/15/2016    9762    6.28    6.28      0    2.14   1.26     2.83       0      29      34      209
5 1503960366 4/16/2016   12669    8.16    8.16      0    2.71   0.410    5.04       0      36      10      221
6 1503960366 4/17/2016    9705    6.48    6.48      0    3.19   0.780    2.51       0      38      20      164
# … with 2 more variables: SedentaryMinutes <dbl>, Calories <dbl>, and abbreviated variable names
#   ¹ActivityDate, ²TotalSteps, ³TotalDistance, ⁴TrackerDistance, ⁵LoggedActivitiesDistance,
#   ⁶VeryActiveDistance, ⁷ModeratelyActiveDistance, ⁸LightActiveDistance, ⁹SedentaryActiveDistance,
#   ˟VeryActiveMinutes, ˟FairlyActiveMinutes, ˟LightlyActiveMinutes
# ℹ Use `colnames()` to see all variable names
```

```
> glimpse(daily_activity)
Rows: 940
Columns: 15
$ Id                       <dbl> 1503960366, 1503960366, 1503960366, 1503960366, 1503960366, 1503960366, 1503960…
$ ActivityDate             <chr> "4/12/2016", "4/13/2016", "4/14/2016", "4/15/2016", "4/16/2016", "4/17/2016", "…
$ TotalSteps               <dbl> 13162, 10735, 10460, 9762, 12669, 9705, 13019, 15506, 10544, 9819, 12764, 14371…
$ TotalDistance            <dbl> 8.50, 6.97, 6.74, 6.28, 8.16, 6.48, 8.59, 9.88, 6.68, 6.34, 8.13, 9.04, 6.41, 9…
$ TrackerDistance          <dbl> 8.50, 6.97, 6.74, 6.28, 8.16, 6.48, 8.59, 9.88, 6.68, 6.34, 8.13, 9.04, 6.41, 9…
$ LoggedActivitiesDistance <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
$ VeryActiveDistance       <dbl> 1.88, 1.57, 2.44, 2.14, 2.71, 3.19, 3.25, 3.53, 1.96, 1.34, 4.76, 2.81, 2.92, 5…
$ ModeratelyActiveDistance <dbl> 0.55, 0.69, 0.40, 1.26, 0.41, 0.78, 0.64, 1.32, 0.48, 0.35, 1.12, 0.87, 0.21, 0…
$ LightActiveDistance      <dbl> 6.06, 4.71, 3.91, 2.83, 5.04, 2.51, 4.71, 5.03, 4.24, 4.65, 2.24, 5.36, 3.28, 3…
$ SedentaryActiveDistance  <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0…
$ VeryActiveMinutes        <dbl> 25, 21, 30, 29, 36, 38, 42, 50, 28, 19, 66, 41, 39, 73, 31, 78, 48, 16, 52, 33,…
$ FairlyActiveMinutes      <dbl> 13, 19, 11, 34, 10, 20, 16, 31, 12, 8, 27, 21, 5, 14, 23, 11, 28, 12, 34, 35, 1…
$ LightlyActiveMinutes     <dbl> 328, 217, 181, 209, 221, 164, 233, 264, 205, 211, 130, 262, 238, 216, 279, 243,…
$ SedentaryMinutes         <dbl> 728, 776, 1218, 726, 773, 539, 1149, 775, 818, 838, 1217, 732, 709, 814, 833, 1…
$ Calories                 <dbl> 1985, 1797, 1776, 1745, 1863, 1728, 1921, 2035, 1786, 1775, 1827, 1949, 1788, 2…
```

```
> str(daily_intensities)
spc_tbl_ [940 × 10] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ Id                       : num [1:940] 1.5e+09 1.5e+09 1.5e+09 1.5e+09 1.5e+09 ...
 $ ActivityDay              : chr [1:940] "4/12/2016" "4/13/2016" "4/14/2016" "4/15/2016" ...
 $ SedentaryMinutes         : num [1:940] 728 776 1218 726 773 ...
 $ LightlyActiveMinutes     : num [1:940] 328 217 181 209 221 164 233 264 205 211 ...
 $ FairlyActiveMinutes      : num [1:940] 13 19 11 34 10 20 16 31 12 8 ...
 $ VeryActiveMinutes        : num [1:940] 25 21 30 29 36 38 42 50 28 19 ...
 $ SedentaryActiveDistance : num [1:940] 0 0 0 0 0 0 0 0 0 0 ...
 $ LightActiveDistance      : num [1:940] 6.06 4.71 3.91 2.83 5.04 ...
 $ ModeratelyActiveDistance: num [1:940] 0.55 0.69 0.4 1.26 0.41 ...
 $ VeryActiveDistance       : num [1:940] 1.88 1.57 2.44 2.14 2.71 ...
 - attr(*, "spec")=
  .. cols(
  ..   Id = col_double(),
  ..   ActivityDay = col_character(),
  ..   SedentaryMinutes = col_double(),
  ..   LightlyActiveMinutes = col_double(),
  ..   FairlyActiveMinutes = col_double(),
  ..   VeryActiveMinutes = col_double(),
  ..   SedentaryActiveDistance = col_double(),
  ..   LightActiveDistance = col_double(),
  ..   ModeratelyActiveDistance = col_double(),
  ..   VeryActiveDistance = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

```
>   glimpse(hourly_calories)
Rows: 22,099
Columns: 3
$ Id           <dbl> 1503960366, 1503960366, 1503960366, 1503960366, 1503960366, 1503960366, 1503960366, 1503960…
$ ActivityHour <chr> "4/12/2016 12:00:00 AM", "4/12/2016 1:00:00 AM", "4/12/2016 2:00:00 AM", "4/12/2016 3:00:00…
$ Calories     <dbl> 81, 61, 59, 47, 48, 48, 48, 47, 68, 141, 99, 76, 73, 66, 110, 151, 76, 83, 124, 104, 132, 1…
>   glimpse(sleep)
Rows: 413
Columns: 5
$ Id                 <dbl> 1503960366, 1503960366, 1503960366, 1503960366, 1503960366, 1503960366, 1503960366, 1…
$ SleepDay           <chr> "4/12/2016 12:00:00 AM", "4/13/2016 12:00:00 AM", "4/15/2016 12:00:00 AM", "4/16/2016…
$ TotalSleepRecords  <dbl> 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…
$ TotalMinutesAsleep <dbl> 327, 384, 412, 340, 700, 304, 360, 325, 361, 430, 277, 245, 366, 341, 404, 369, 277, …
$ TotalTimeInBed     <dbl> 346, 407, 442, 367, 712, 320, 377, 364, 384, 449, 323, 274, 393, 354, 425, 396, 309, …
```

Data Cleaning:

For the Data Cleaning step, I begin with checking how many NA values are present in each dataset.

```
#Checking how many NA values are there in each Dataset
sum(is.na(daily_activity))
sum(is.na(daily_calories))
sum(is.na(daily_intensities))
sum(is.na(heartrate))
sum(is.na(hourly_calories))
sum(is.na(hourly_steps))
sum(is.na(sleep))
sum(is.na(weight))
```

Results:

```
> sum(is.na(daily_activity))
[1] 0
> sum(is.na(daily_calories))
[1] 0
> sum(is.na(daily_intensities))
[1] 0
> sum(is.na(heartrate))
[1] 0
> sum(is.na(hourly_calories))
[1] 0
> sum(is.na(hourly_steps))
[1] 0
> sum(is.na(sleep))
[1] 0
> sum(is.na(weight))
[1] 65
```

We can clearly see that the weight dataset has a lot of NA values. Upon diving into the weight dataset further, we realise that all these NA values belong to the “Fat” column which only has 65 out of 67 NA values. Hence, it is understandable to get rid of the “Fat” column entirely.

```
#Removing "Fat" column which had NA values
weight <- weight[,-5]
sum(is.na(weight))
```

Next, data duplication is checked.

```
#Checking duplication of data
sum(duplicated(daily_activity))
sum(duplicated(daily_calories))
sum(duplicated(daily_intensities))
sum(duplicated(heartrate))
sum(duplicated(hourly_calories))
sum(duplicated(hourly_steps))
sum(duplicated(sleep))
sum(duplicated(weight))
```

Results:

```
> sum(duplicated(daily_activity))
[1] 0
> sum(duplicated(daily_calories))
[1] 0
> sum(duplicated(daily_intensities))
[1] 0
> sum(duplicated(heartrate))
[1] 0
> sum(duplicated(hourly_calories))
[1] 0
> sum(duplicated(hourly_steps))
[1] 0
> sum(duplicated(sleep))
[1] 3
> sum(duplicated(weight))
[1] 0
```

The results indicate that the sleep dataset has duplicated values. Hence, we remove the duplicated data.

```
#Removing duplicated data in sleep dataset
sleep <- distinct(sleep)
sum(duplicated(sleep))
```

Next, we add “Weekday” column to sleep and daily_activity datasets. This will allow us to analyze the activity and sleep data in a much more meaningful way.

```
#Adding Weekday column to Sleep Dataset
library(dplyr)
sleep <- sleep %>% mutate(Weekday = weekdays(as.Date(SleepDay, "%m/%d/%Y")))

#Adding Weekday Column to daily_activity Dataset
daily_activity <- daily_activity %>%
  mutate(Weekday = weekdays(as.Date(ActivityDate, "%m/%d/%Y")))
```

A significant step during the data cleaning process is to make sure that the data is in the correct format. For instance, here our “Date” columns in 5 of our datasets were in Character format. We need to change them to Date format.

```
#Changing Date Column from Character to Date
daily_activity$ActivityDate <- as.Date(daily_activity$ActivityDate, "%m/%d/%Y")
daily_calories$ActivityDay <- as.Date(daily_calories$ActivityDay, "%m/%d/%Y")
daily_intensities$ActivityDay <- as.Date(daily_intensities$ActivityDay,
                                         "%m/%d/%Y")
sleep$Date <- as.Date(sleep$SleepDay, "%m/%d/%Y")
weight$Date_new <- as.Date(weight$Date, "%m/%d/%Y")
```

Next, we realise that the Datetime columns need to be split up into separate Date and Time columns.

```
#Splitting Datetime Column into separate Date and Time Columns
heartrate$datetime <- str_split_fixed(heartrate$Time, " ", n = 2)
heartrate$Date <- as.Date(heartrate$Time, "%m/%d/%Y")
heartrate$Time_new <- format(strptime(heartrate$datetime[,2], "%I:%M:%S %p"),
                             format="%H:%M:%S")
heartrate <- heartrate[-4]
heartrate$Time_new <- as_hms(heartrate$Time_new)

hourly_calories$datetime <- str_split_fixed(hourly_calories$ActivityHour, " ",
                                            n = 2)
hourly_calories$Date <- as.Date(hourly_calories$ActivityHour, "%m/%d/%Y")
hourly_calories$time_new <- format(strptime(hourly_calories$datetime[,2],
                                            "%I:%M:%S %p"), format="%H:%M:%S")
hourly_calories <- hourly_calories %>%  rename("Time" = "time_new")
hourly_calories <- hourly_calories[-4]
hourly_calories$Time <- as_hms(hourly_calories$Time)

hourly_steps$datetime <- str_split_fixed(hourly_steps$ActivityHour, " ", n = 2)
hourly_steps$Date <- as.Date(hourly_steps$ActivityHour, "%m/%d/%Y")
hourly_steps$Time <- format(strptime(hourly_steps$datetime[,2], "%I:%M:%S %p"),
                            format="%H:%M:%S")
hourly_steps <- hourly_steps[-4]
hourly_steps$Time <- as_hms(hourly_steps$Time)
```

Now that the data is clean, we move on to the **Analyze** phase.

Step 4: Analyze

We begin our analysis by counting the distinct IDs in each dataset:

```
#Counting Distinct IDs in each dataset
count(daily_activity %>% distinct(Id))
count(daily_calories %>% distinct(Id))
count(daily_intensities %>% distinct(Id))
count(heartrate %>% distinct(Id))
count(sleep %>% distinct(Id))
count(weight %>% distinct(Id))
count(hourly_steps %>% distinct(Id))
count(hourly_calories %>% distinct(Id))
```

Results:

```
> count(daily_activity %>% distinct(Id))
# A tibble: 1 × 1
      n
  <int>
1    33
> count(daily_calories %>% distinct(Id))
# A tibble: 1 × 1
      n
  <int>
1    33
> count(daily_intensities %>% distinct(Id))
# A tibble: 1 × 1
      n
  <int>
1    33
> count(heartrate %>% distinct(Id))
# A tibble: 1 × 1
      n
  <int>
1    14
> count(sleep %>% distinct(Id))
# A tibble: 1 × 1
      n
  <int>
1    24
> count(weight %>% distinct(Id))
# A tibble: 1 × 1
      n
  <int>
1     8
> count(hourly_steps %>% distinct(Id))
# A tibble: 1 × 1
      n
  <int>
1    33
> count(hourly_calories %>% distinct(Id))
# A tibble: 1 × 1
      n
  <int>
1    33
```

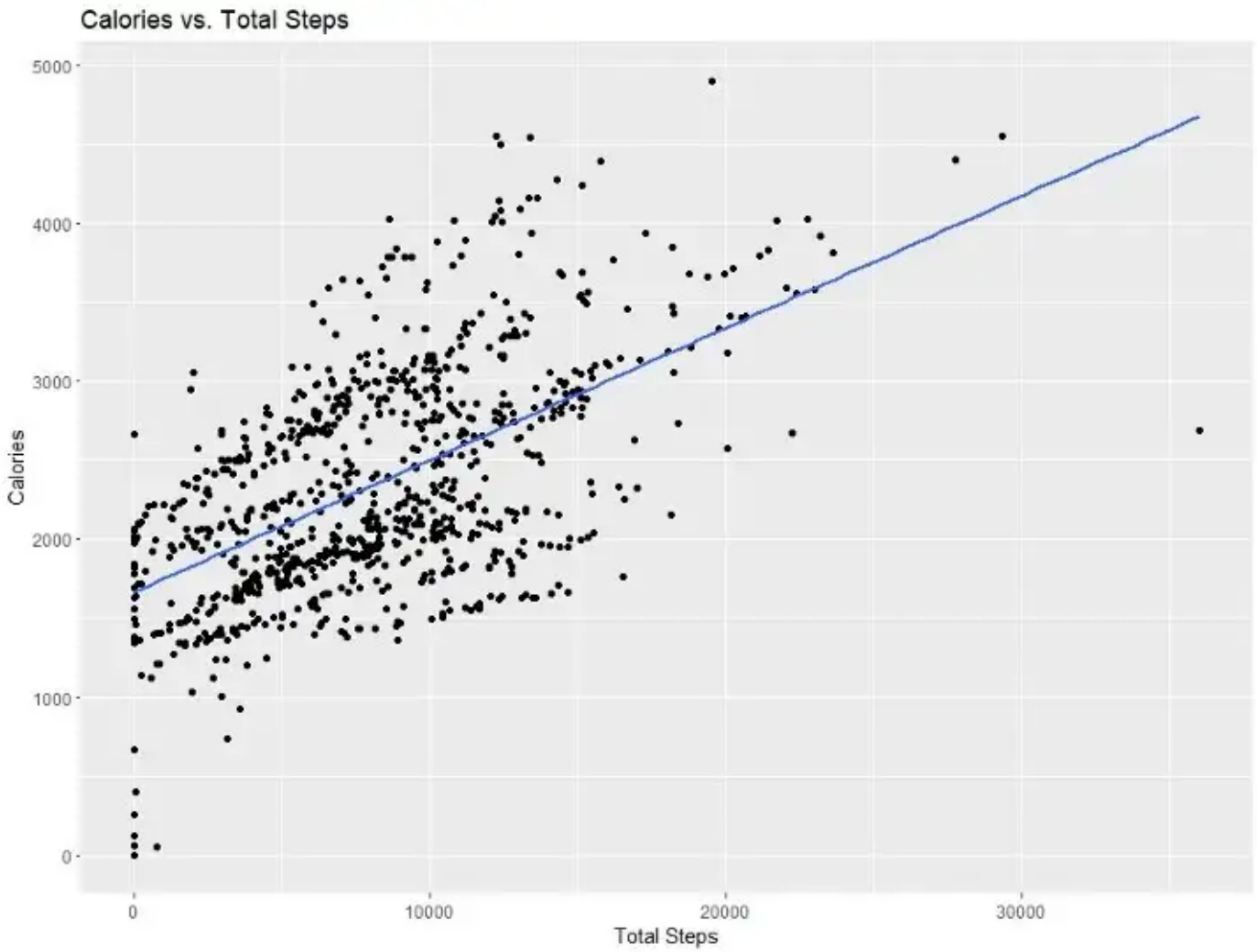
Plot showing Sedentary Minutes vs. Total Steps

```
#Sedentary Minutes vs. Total Steps
ggplot(data = daily_activity, aes(x = TotalSteps, y = SedentaryMinutes)) +
       geom_point() + geom_smooth(method = lm, se = FALSE) +
       labs(title="Sedentary Minutes vs. Total Steps") + xlab("Total Steps") +
       ylab("Sedentary Minutes")
```

The plot clearly shows a negative correlation between Sedentary Minutes and Total Steps which means that the more sedentary time an individual has, fewer are the number of steps taken by that individual.

Plot showing Calories vs. Total Steps

```
#Calories vs. Total Steps
ggplot(data = daily_activity, aes(x = TotalSteps, y = Calories)) +
       geom_point() + geom_smooth(method = lm, se = FALSE) +
       labs(title = "Calories vs. Total Steps") + xlab("Total Steps") +
       ylab("Calories")
```



The plot shows a strong positive correlation between calories and steps i.e. the more steps a person takes, the more calories he/she burns.

Step 5: Share

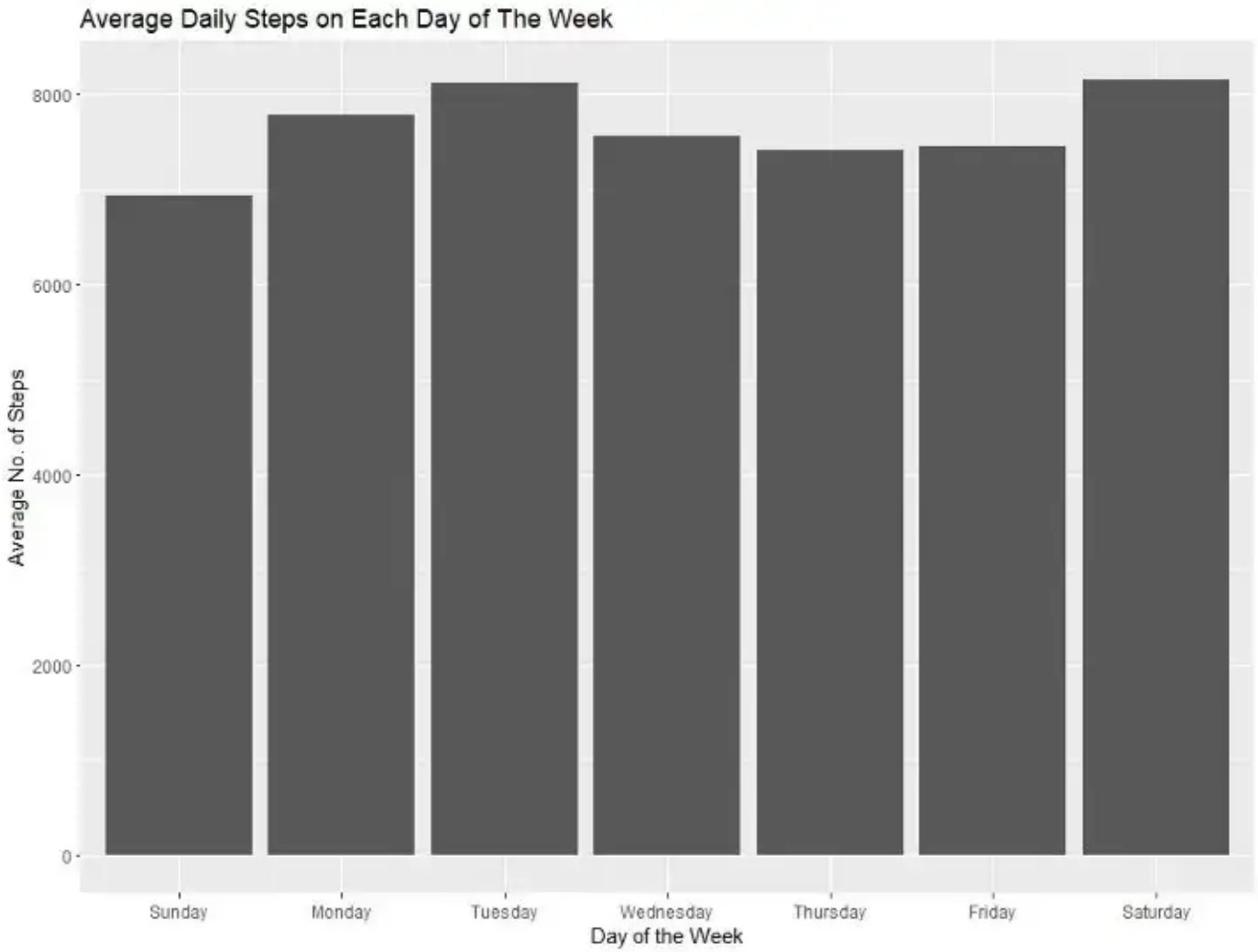
Once the basic visualizations are dealt with, we move ahead towards some more meaningful visualizations and the key findings pertaining to each visualization.

Weekday Array Creation for Ease in Plotting:

```
#Weekday order to to arrange bars of bar plot as per day of the week sequence
days_of_the_week <- c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday",
                      "Friday", "Saturday")
```

Bar Plot showing Average Daily Steps on Each Day of the Week

```
#Average Daily Steps on Each Day of The Week
ggplot(data = daily_activity, aes(x = factor(Weekday, days_of_the_week),
       y = TotalSteps)) + geom_bar(stat = "summary", fun = "mean") +
       labs(title = "Average Daily Steps on Each Day of The Week") +
       xlab("Day of the Week") + ylab("Average No. of Steps")
```



The bar plot shows that **Tuesday** and **Saturday** are the days of the week when the users have taken the most number of steps on average, while users take the least number of steps on **Sunday.**

Changing ID Column from Double to Character:

```
#Changing ID Column from double to character
daily_activity$Id <- as.character(daily_activity$Id)
sleep$Id <- as.character(sleep$Id)
```

Calculation of Average Steps of Each Individual:

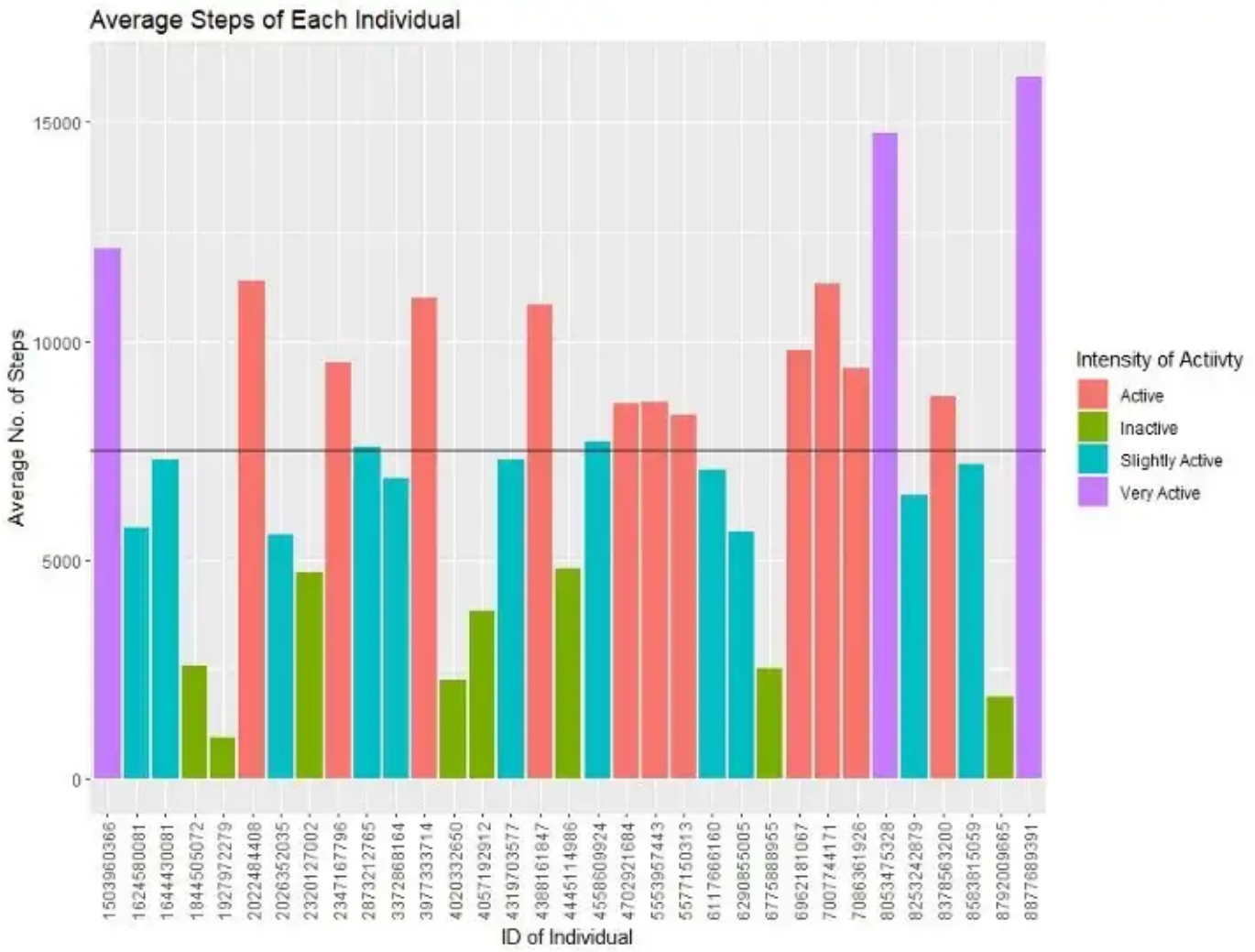
```
#Average Steps of Each Individual
AverageSteps <- daily_activity %>% select(Id, TotalSteps) %>% group_by(Id) %>%
  summarise(mean(TotalSteps))
```

Adding New Column to Distribute in Different Groups:

```
#Adding new column to distribute in different groups
AverageSteps <- AverageSteps %>% rename("AvgSteps" = "mean(TotalSteps)")
AverageSteps <- AverageSteps %>%
  mutate(
    ActivityIntensity = case_when(
      AvgSteps >= 12000 ~ "Very Active",
      AvgSteps >= 8000 & AvgSteps < 12000 ~ "Active",
      AvgSteps >= 5000 & AvgSteps < 8000  ~ "Slightly Active",
      AvgSteps < 5000 ~ "Inactive"
    )
  )
```

Plot showing Average Steps of Each Individual

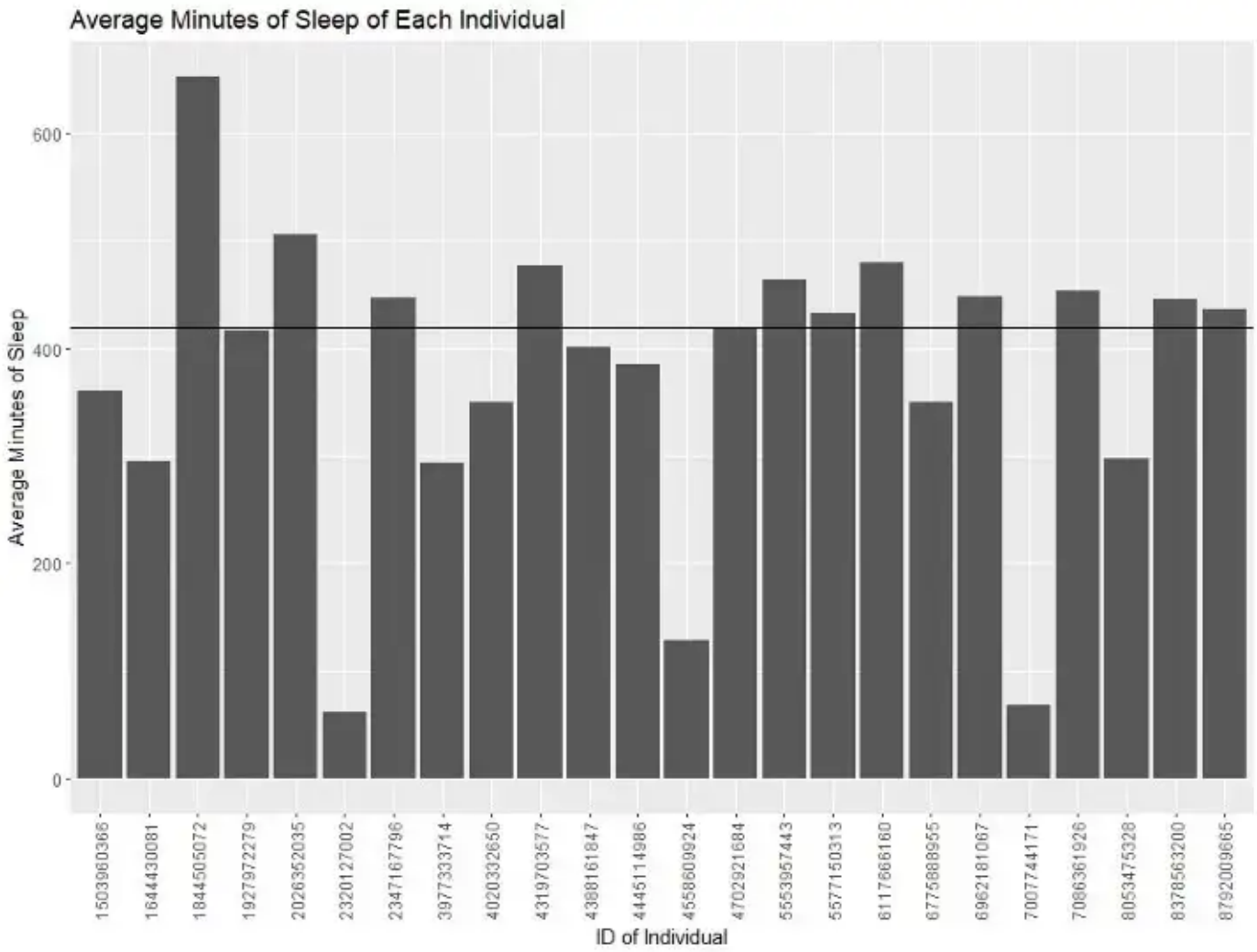
```
#Average Steps of Each Individual
ggplot(data = AverageSteps) + geom_col(mapping = aes(x = Id, y = AvgSteps,
       fill = ActivityIntensity)) + theme(axis.text.x = element_text(angle = 90,
       vjust = 0.5, hjust=1)) + labs(title = "Average Steps of Each Individual")
       + xlab("ID of Individual") + ylab("Average No. of Steps") +
       geom_hline(yintercept = mean(AverageSteps$AvgSteps)) +
       scale_fill_discrete(name = "Intensity of Actiivty")
```



The plot shows that 3 individuals stand above the rest when it comes to comparing their average daily steps. Majority of the individuals fall under the categories “Slightly Active” and “Active”.

Plot showing Average Minutes of Sleep of Each Individual

```
#Average Minutes of Sleep of Each Individual
ggplot(data = sleep, aes(x = Id, y = TotalMinutesAsleep)) +
       geom_bar(stat = "summary", fun = "mean") +
       theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
       labs(title = "Average Minutes of Sleep of Each Individual") +
       xlab("ID of Individual") + ylab("Average Minutes of Sleep") +
       geom_hline(yintercept = mean(sleep$TotalMinutesAsleep))
```



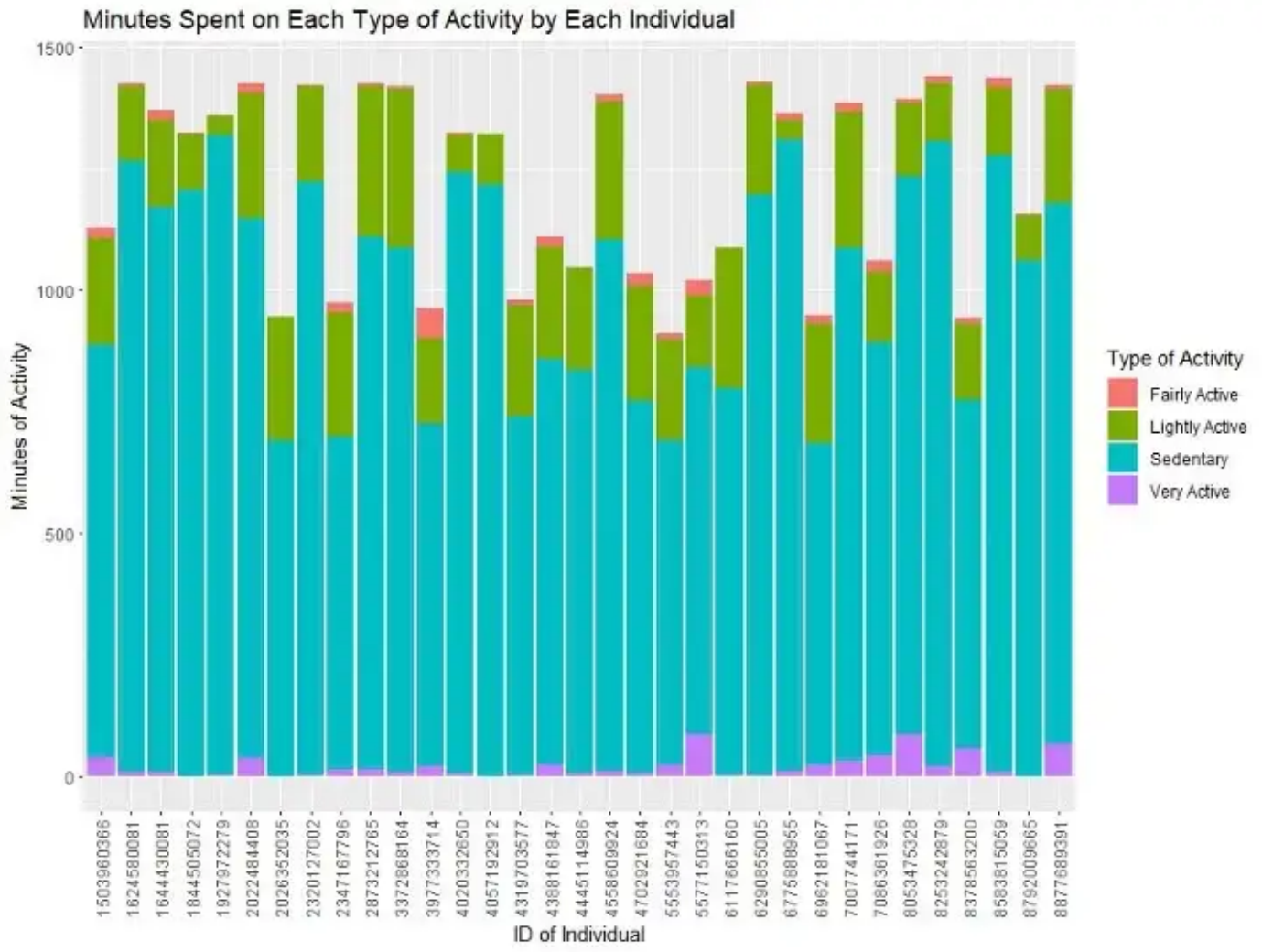
This plot gives an impression of how many minutes each individual sleeps on average. Out of the 24 individuals for whom the sleep data is available, half of them sleep less than the average sleep duration for all individuals.

Converting Daily_Activity Dataset from Wide to Long:

```
#Converting activity dataset from wide to long
daily_activity_long <- gather(daily_activity, key = "Activitytype",
                              value = "Minutes",11:14)
```

Stacked Bar Plot showing Minutes Spent on Each Type of Activity by Each Individual

```
#Minutes Spent on Each Type of Activity by Each Individual
ggplot(data = daily_activity_long, aes(fill = Activitytype, x = Id,
       y = Minutes)) + geom_bar(position = "stack", stat = "summary",
       fun = "mean") + theme(axis.text.x = element_text(angle = 90, vjust = 0.5,
       hjust=1)) + labs(title = "Minutes Spent on Each Type of Activity by Each
       Individual") + xlab("ID of Individual") + ylab("Minutes of Activity") +
       scale_fill_discrete(name = "Type of Activity",labels = c("Fairly Active",
       "Lightly Active", "Sedentary", "Very Active"))
```

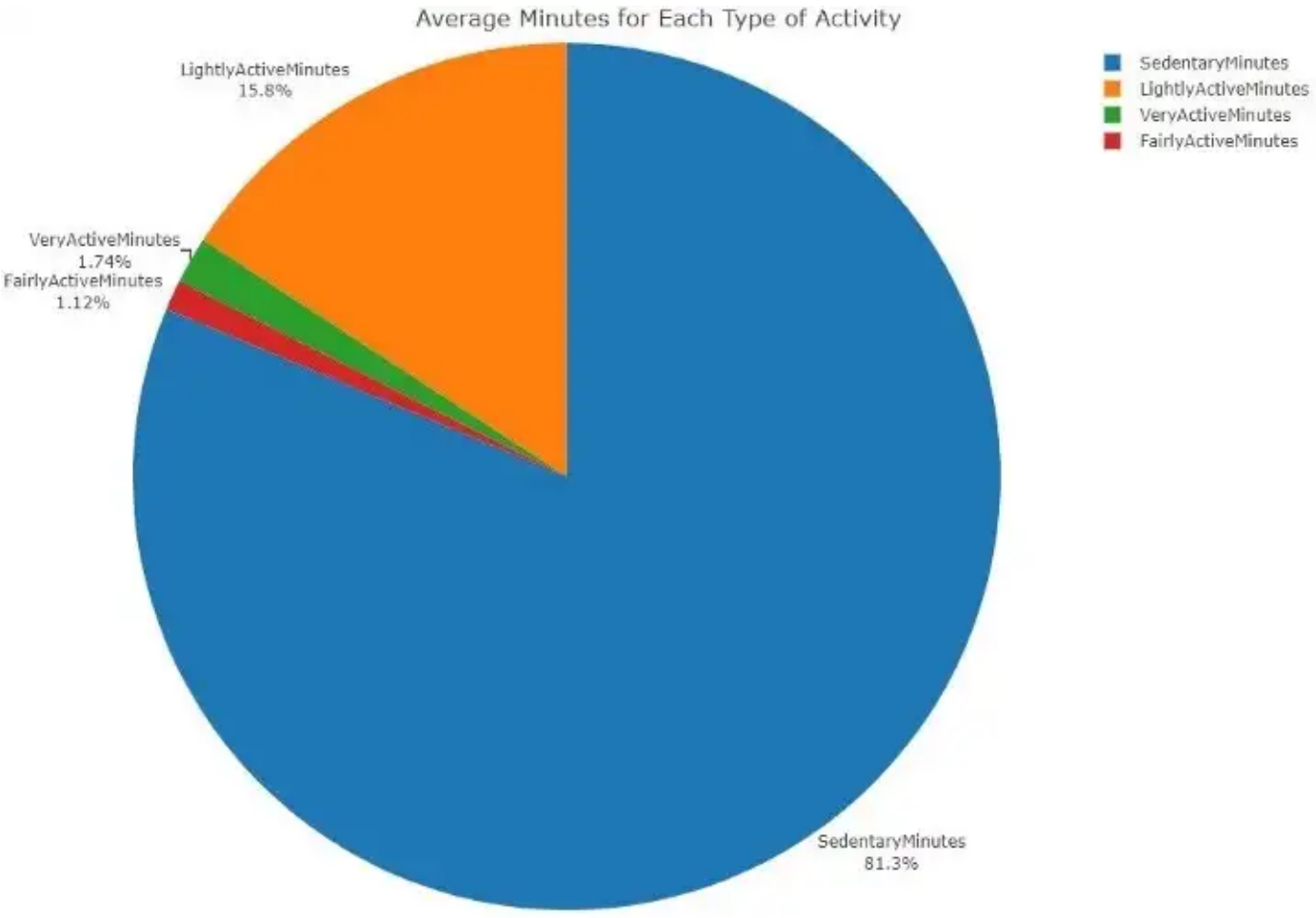


Calculating the Average Minutes for Each Type of Activity:

```
#Average Minutes for each type of Activity
AverageActivityMinutes <- daily_activity_long %>%
  select(Activitytype, Minutes) %>% group_by(Activitytype) %>%
  summarise(mean(Minutes))
AverageActivityMinutes <- AverageActivityMinutes %>%
  rename("AvgMins" = "mean(Minutes)")
AverageActivityMinutes$AvgMins <- round(AverageActivityMinutes$AvgMins, 1)
```

Pie Chart to show Average Minutes for Each Type of Activity

```
#Pie Chart to show Average Minutes for each type of Activity
plot_ly(AverageActivityMinutes, labels = ~Activitytype, values = ~AvgMins,
        type = 'pie',textposition = 'outside',textinfo = 'label+percent') %>%
        layout(title = 'Average Minutes for Each Type of Activity')
```



The chart clearly shows that major chunk (81.3%) of an individual's daily activity is comprised of Sedentary Minutes while only a minuscule portion is spent actively (1.74%).

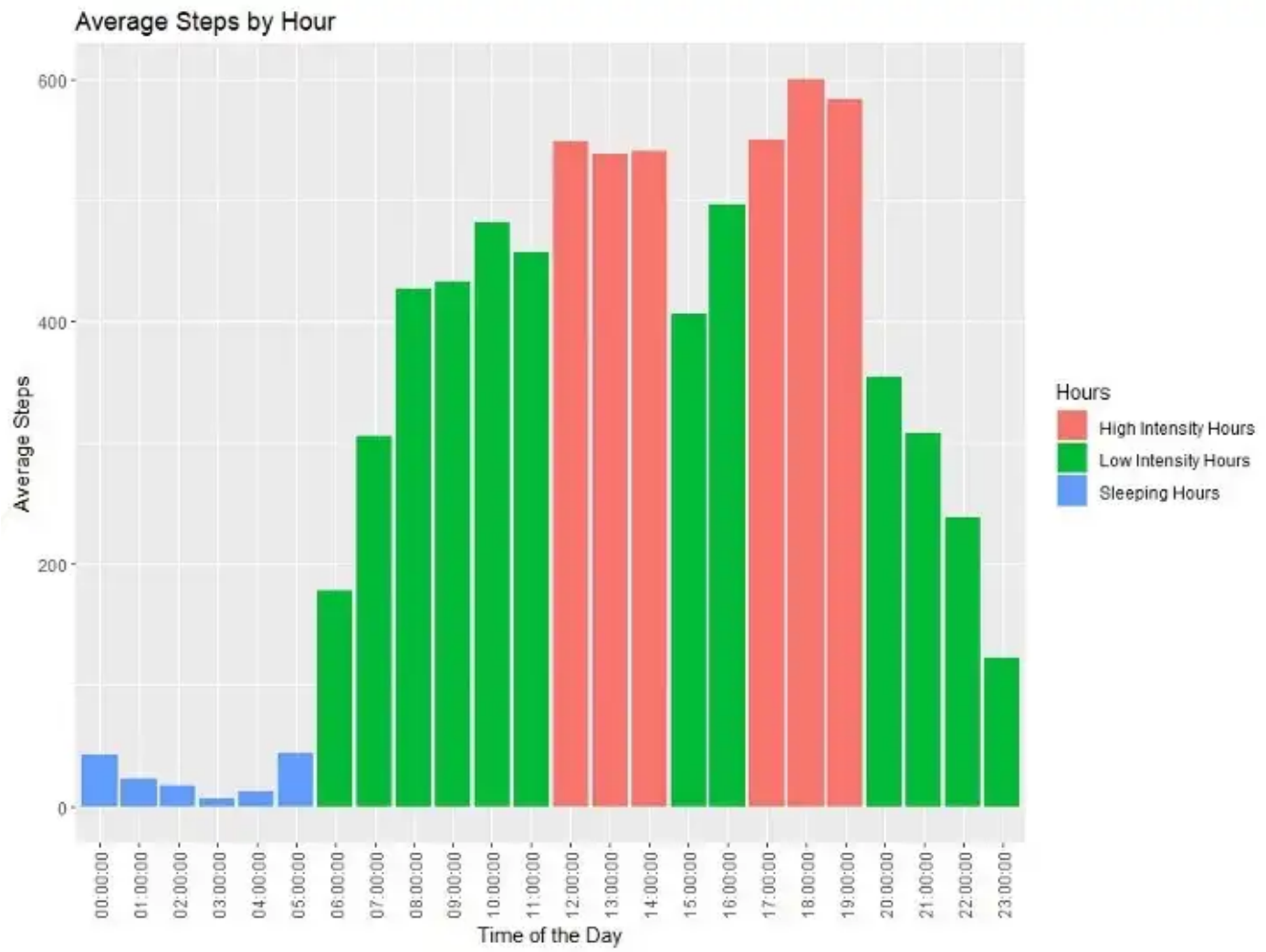
Calculation of Average Steps During Each Hour

```
#Average Steps by hour
AverageHourlySteps <- hourly_steps %>% select(Time, StepTotal) %>%
  group_by(Time) %>% summarise(mean(StepTotal))
AverageSteps <- daily_activity %>% select(Id, TotalSteps) %>%
  group_by(Id) %>% summarise(mean(TotalSteps))
AverageHourlySteps <- AverageHourlySteps %>% rename("AvgSteps" =
"mean(StepTotal)")
AverageHourlySteps$AvgSteps <- round(AverageHourlySteps$AvgSteps, 1)
AverageHourlySteps <- AverageHourlySteps %>%
  mutate(
    StepsIntensity = case_when(
      AvgSteps >= 500 ~ "High Intensity Hours",
      AvgSteps < 100 ~ "Sleeping Hours",
      TRUE ~ "Low Intensity Hours"
    )
  )
```

Plot showing Average Steps by Hour

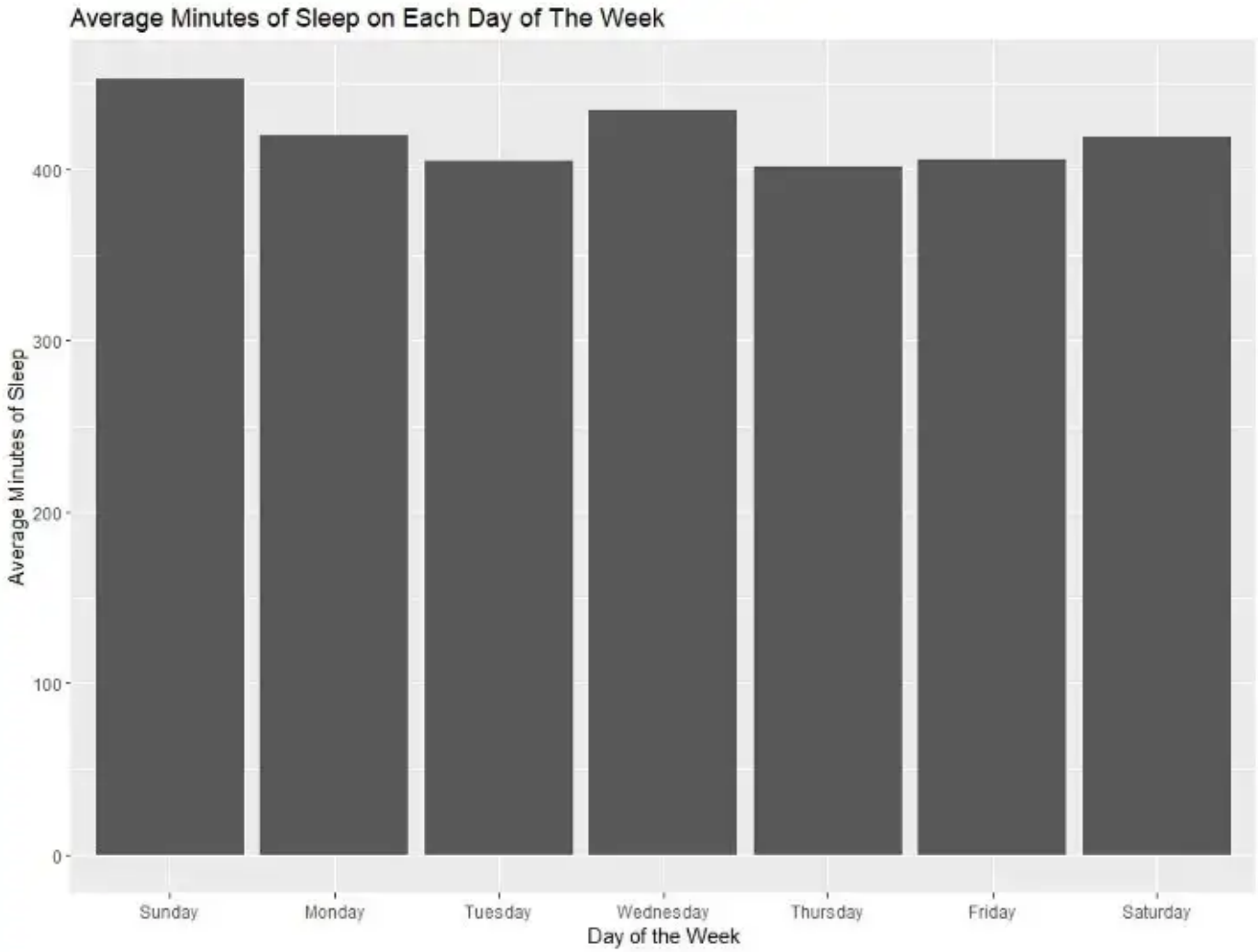
```
AverageHourlySteps$Time <- as.character(AverageHourlySteps$Time)

ggplot(data = AverageHourlySteps) + geom_col(aes(x = Time, y = AvgSteps,
       fill = StepsIntensity)) + theme(axis.text.x = element_text(angle = 90,
       vjust = 0.5, hjust=1)) + labs(title = "Average Steps by Hour") +
       xlab("Time of the Day") + ylab("Average Steps") +
       scale_fill_discrete(name = "Hours")
```



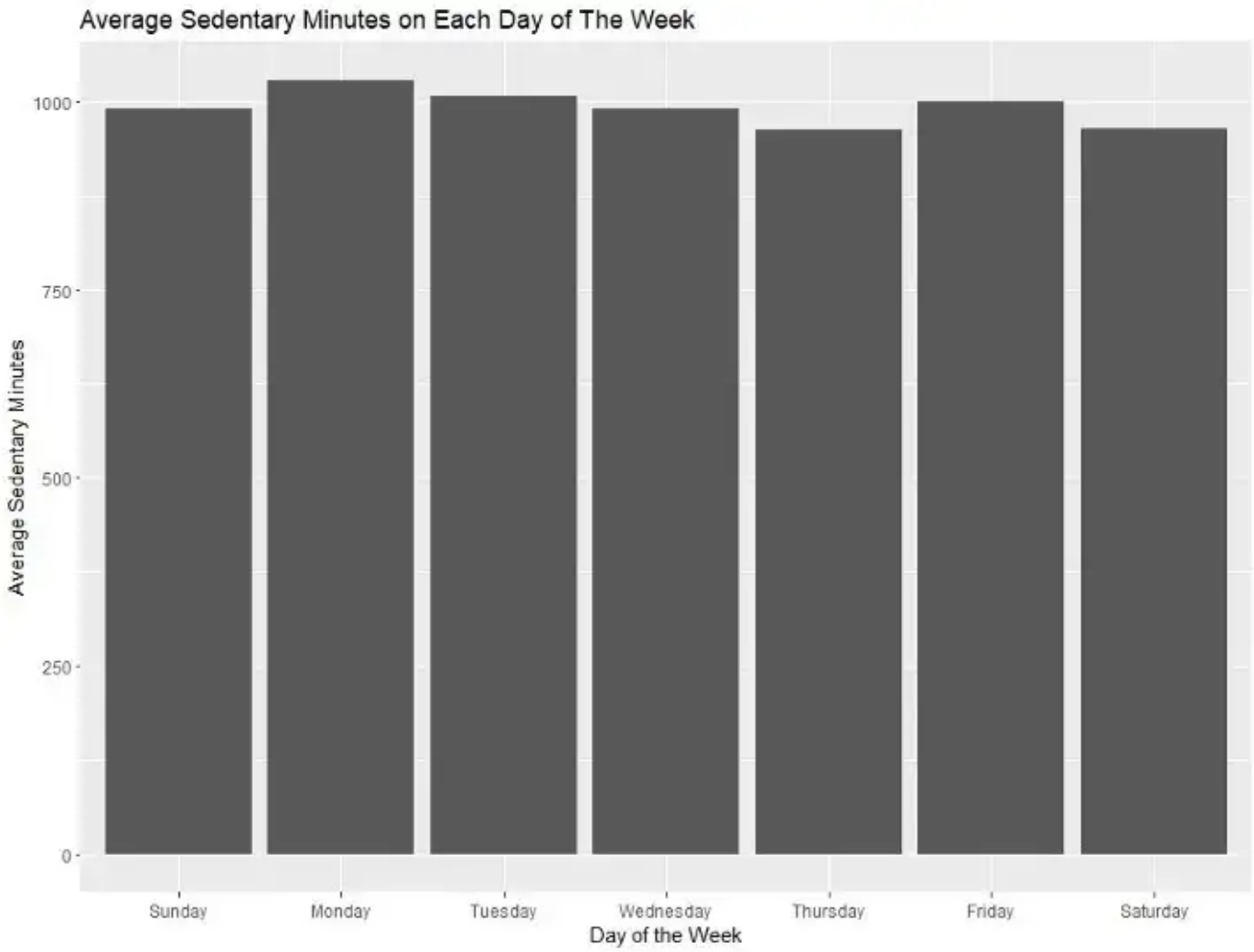
Plot showing Average Minutes of Sleep on Each Day of the Week

```
#Average Minutes of Sleep on Each Day of the Week
ggplot(data = sleep, aes(x = factor(Weekday, days_of_the_week),
       y = TotalMinutesAsleep)) + geom_bar(stat = "summary", fun = "mean") +
       labs(title = "Average Minutes of Sleep on Each Day of The Week") +
       xlab("Day of the Week") + ylab("Average Minutes of Sleep")
```



Plot showing Average Sedentary Minutes on Each Day of the Week

```
#Average Sedentary minutes on Each Day of the Week
ggplot(data = daily_activity, aes(x = factor(Weekday, days_of_the_week),
       y = SedentaryMinutes)) + geom_bar(stat = "summary", fun = "mean") +
       labs(title = "Average Sedentary Minutes on Each Day of The Week") +
       xlab("Day of the Week") + ylab("Average Sedentary Minutes")
```



The previous two plots shows that most individuals sleep the most on Sundays. However, sedentary minutes are mostly the same on each day of the week with Monday and Friday having marginally greater numbers.

Plotting Sleeping Minutes vs. Sedentary Minutes relative to Total Minutes

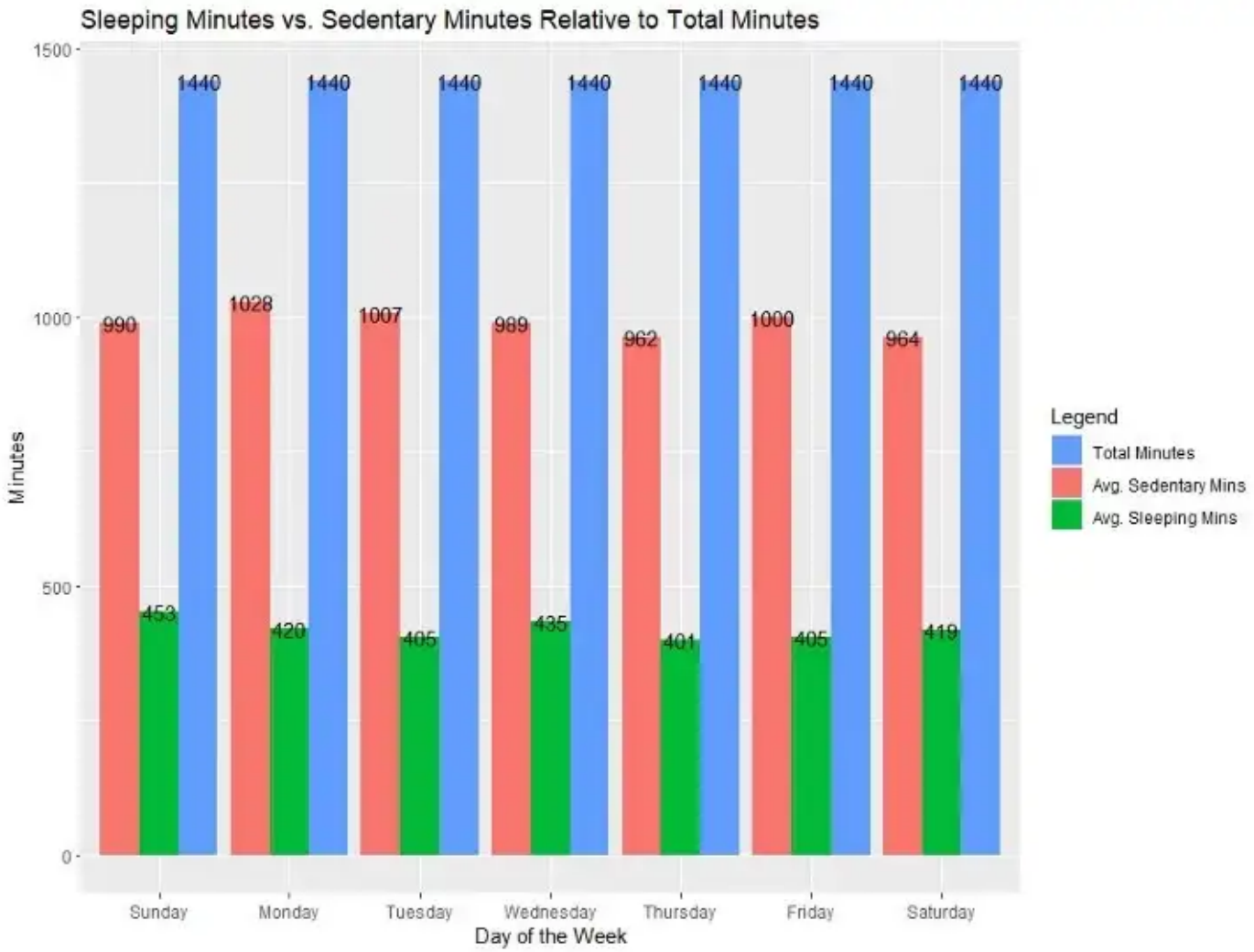
```
SedentaryMinutes <- daily_activity %>% select(Weekday, SedentaryMinutes) %>%
  group_by(Weekday) %>% summarise(mean(SedentaryMinutes))
SedentaryMinutes <- SedentaryMinutes %>%
  rename("AvgSedentaryMins" = "mean(SedentaryMinutes)")
SedentaryMinutes$AvgSedentaryMins <- round(SedentaryMinutes$AvgSedentaryMins, 0)

SleepingMinutes <- sleep %>% select(Weekday, TotalMinutesAsleep) %>%
  group_by(Weekday) %>% summarise(mean(TotalMinutesAsleep))
SleepingMinutes <- SleepingMinutes %>%
  rename("AvgSleepingMins" = "mean(TotalMinutesAsleep)")
SleepingMinutes$AvgSleepingMins <- round(SleepingMinutes$AvgSleepingMins, 0)

merge_sleep_sedentary <- merge(SleepingMinutes, SedentaryMinutes,
                               by = "Weekday")
merge_sleep_sedentary <- merge_sleep_sedentary %>% mutate(Total_Mins = 1440)
merge_sleep_sedentary_long <- gather(merge_sleep_sedentary, key = "SleepSed",
                                     value = "Minutes", 2:4)
merge_sleep_sedentary_long <- merge_sleep_sedentary_long %>%
  mutate(Percentage = Minutes * 100 / 1440)
merge_sleep_sedentary_long$Percentage <-
  round(merge_sleep_sedentary_long$Percentage, 1)



ggplot(data = merge_sleep_sedentary_long, aes(x = factor(Weekday,
       days_of_the_week), y = Minutes, fill = SleepSed)) +
       geom_bar(position="dodge", stat="identity") + geom_text(aes(label =
       Minutes), position = position_dodge(0.9)) + xlab("Day of the Week") +
       scale_fill_discrete(name = "Legend", breaks = c("Total_Mins",
       "AvgSedentaryMins", "AvgSleepingMins"), labels = c("Total Minutes",
       "Avg. Sedentary Mins", "Avg. Sleeping Mins")) + labs(title = "Sleeping
       Minutes vs. Sedentary Minutes Relative to Total Minutes") +
       theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```



Step 6: Act

Now that our visualizations are done with, it's time to act on those key findings. Here are a few of my recommendations for the Bellabeat Marketing Team:

- The sedentary time is too high i.e. more than 16 hours every day, as per the data gathered by the app. This number needs to be reduced with a suitable marketing strategy. Customers need to be informed of the great health risks associated with inactivity and lethargy.
- The Bellabeat app needs to undergo a thorough transformation. Merely providing data on user's health would not make it stand out in the global market. Rather, the app should serve as a means to re-invigorate the users so that they meet their fitness goals.
- On average the users are not sleeping enough. In order to help them improve their sleep, Bellabeat app can send out sleep notifications to remind the user to go to bed

early. Moreover, the app can also be allowed to shift the phone to “Do-Not-Disturb” (DND) mode after a certain time at night so that phone notifications do not forcefully keep the user awake.

- Bellabeat app needs to enable users to add friends and view each other’s activity so that there is a sense of competition which drives the users to be more active.
- Recommend users to get atleast 8,000 steps a day and enable alert notifications through the app to motivate users to meet their daily goal. Moreover, direct the users’ attention to the amazing health benefits associated with accomplishing the 8,000 daily steps feat.
- Bellabeat app can suggest ideas for low calorie food which can serve as a reminder for obese or overweight users to alter their eating habits.
- On average, the users were active for a very small amount of time even though the recommended amount of time one must indulge in vigorous activity is 75 minutes per week. Bellabeat app needs to recommend multiple activities (hiking trips, marathons etc.)happening in the vicinity of the user’s residence to encourage her to take part in those activities in order to meet the 75 minutes per week goal.
- Enable the app to notify the user to engage in an activity if she has been sedentary for an extended period of time.
- Enable the app to notify the user to get up from the bed if she spends a considerable amount of time lying in bed despite being awake.

Bellabeat Fitness Tracker Watch Google Data Analytics