# Bellabeat Case Study Using R

Rafe Ng

2022-10-10

## PHASE 1: ASK

### 1.0 Introduction and background

Urška Sršen and Sando Mur founded Bellabeat, a high-tech company that manufactures health-focused smart products. Bellabeat has grown rapidly and quickly positioned itself as a tech-driven wellness company for women ever since it was founded in 2013. Bellabeat inspires and empowers women with knowledge about their own health and habits. Sršen knows that an analysis of Bellabeat's available consumer data would reveal more opportunities for growth. Hence, the marketing analytics team is being tasked to focus on a Bellabeat product and analyze smart device usage data in order to gain insight into how people are already using their smart devices.

### 1.1 Business Task

Analyze smart device usage data in order to gain insight into how consumers use non-Bellabeat smart devices and discover trends that can help influence Bellabeat marketing strategy.

### 1.2 Business Objectives

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?

### 1.3 Deliverables

1. A clear summary of the business task
2. A description of all data sources used
3. Documentation of any cleaning or manipulation of data
4. A summary of your analysis
5. Supporting visualizations and key findings
6. Your top high-level content recommendations based on your analysis

### 1.4 Key Stakeholders

1. **Urška Sršen**: Bellabeat's cofounder and Chief Creative Officer
2. **Sando Mur**: Mathematician, Bellabeat's cofounder and key member of the Bellabeat executive team
3. **Bellabeat marketing analytics team**: A team of data analysts guiding Bellabeat's marketing strategy

## PHASE 2: PREPARE

### 2.1 Data Source

1. The data is publicly available on Kaggle: FitBit Fitness Tracker Data and there are 18 CSV files.
2. Generated by respondents to a distributed survey via Amazon Mechanical Turk between 03-12-2016 and 05-12-2016.
3. 30 eligible Fitbit users consented to the submission of personal tracker data.
4. The personal data collected including minute-level output for physical activity, heart rate, and sleep monitoring.

### 2.2 Data Limitations

1. The data only contains information from 33 respondents, which is insufficient and unsuitable to represent the entire female population.
2. Data was collected in 2016, which is too old as the user's physical activity, heart rate, steps taken, and sleeping habits have likely changed throughout the 6 year gap.

### 2.3 Is Data ROCCC?

1. **Reliable**: **No** - It only contains 30 respondents.
2. **Original**: **_No** - Data was provided by a third party provider (Amazon Mechanical Turk).
3. **Comprehensive**: **Yes** - The data matches the majority of Bellabeat products.
4. **Current**: **No** - The data is 6 years old.
5. **Cited**: **No** - Data was collected by a third-party provider (Amazon Mechanical Turk).

## PHASE 3: PROCESS

### 3.1 Import Libraries

Libraries like ggplot2, tidyverse, dplyr, lubridate, and forcats are imported to use their functions.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggplot2)
library(dplyr)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
##
```

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

```
library(forcats)
```

### 3.2 Importing Datasets

In this analysis, we will be using 3 of the 18 datasets (in CSV files)

- dailyActivity_merged
- sleepDay_merged
- hourlySteps_merged

Each dataframe is assigned to a name:

- dailyactivity_df
- sleepday_df
- hourlysteps_df

```
dailyactivity_df <- read_csv("dailyActivity_merged.csv")
```

```
## Rows: 940 Columns: 15
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): ActivityDate
## dbl (14): Id, TotalSteps, TotalDistance, TrackerDistance, LoggedActivitiesDi...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
sleepday_df <- read.csv("sleepDay_merged.csv")
hourlysteps_df <- read.csv("hourlySteps_merged.csv")
```

We will preview the first few rows of the dataframes and their structures.

```
head(dailyactivity_df)
```

```
## # A tibble: 6 x 15
##       Id Activ~1 Total~2 Total~3 Track~4 Logge~5 VeryA~6 Moder~7 Light~8 Seden~9
##    <dbl> <chr>     <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 1.50e9 4/12/2~   13162    8.5     8.5        0    1.88   0.550    6.06       0
## 2 1.50e9 4/13/2~   10735    6.97    6.97       0    1.57   0.690    4.71       0
## 3 1.50e9 4/14/2~   10460    6.74    6.74       0    2.44   0.400    3.91       0
## 4 1.50e9 4/15/2~    9762    6.28    6.28       0    2.14   1.26     2.83       0
## 5 1.50e9 4/16/2~   12669    8.16    8.16       0    2.71   0.410    5.04       0
## 6 1.50e9 4/17/2~    9705    6.48    6.48       0    3.19   0.780    2.51       0
## # ... with 5 more variables: VeryActiveMinutes <dbl>,
## #   FairlyActiveMinutes <dbl>, LightlyActiveMinutes <dbl>,
## #   SedentaryMinutes <dbl>, Calories <dbl>, and abbreviated variable names
```

```
## #   1: ActivityDate, 2: TotalSteps, 3: TotalDistance, 4: TrackerDistance,
## #   5: LoggedActivitiesDistance, 6: VeryActiveDistance,
## #   7: ModeratelyActiveDistance, 8: LightActiveDistance,
## #   9: SedentaryActiveDistance
```

```
str(dailyactivity_df)
```

```
## spec_tbl_df [940 x 15] (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 ...
##  $ ActivityDate             : chr [1:940] "4/12/2016" "4/13/2016" "4/14/2016" "4/15/2016" ...
##  $ TotalSteps               : num [1:940] 13162 10735 10460 9762 12669 ...
##  $ TotalDistance            : num [1:940] 8.5 6.97 6.74 6.28 8.16 ...
##  $ TrackerDistance          : num [1:940] 8.5 6.97 6.74 6.28 8.16 ...
##  $ LoggedActivitiesDistance: num [1:940] 0 0 0 0 0 0 0 0 0 0 ...
##  $ VeryActiveDistance       : num [1:940] 1.88 1.57 2.44 2.14 2.71 ...
##  $ ModeratelyActiveDistance: num [1:940] 0.55 0.69 0.4 1.26 0.41 ...
##  $ LightActiveDistance      : num [1:940] 6.06 4.71 3.91 2.83 5.04 ...
##  $ SedentaryActiveDistance : num [1:940] 0 0 0 0 0 0 0 0 0 0 ...
##  $ VeryActiveMinutes        : num [1:940] 25 21 30 29 36 38 42 50 28 19 ...
##  $ FairlyActiveMinutes      : num [1:940] 13 19 11 34 10 20 16 31 12 8 ...
##  $ LightlyActiveMinutes     : num [1:940] 328 217 181 209 221 164 233 264 205 211 ...
##  $ SedentaryMinutes         : num [1:940] 728 776 1218 726 773 ...
##  $ Calories                 : num [1:940] 1985 1797 1776 1745 1863 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Id = col_double(),
##   ..   ActivityDate = col_character(),
##   ..   TotalSteps = col_double(),
##   ..   TotalDistance = col_double(),
##   ..   TrackerDistance = col_double(),
##   ..   LoggedActivitiesDistance = col_double(),
##   ..   VeryActiveDistance = col_double(),
##   ..   ModeratelyActiveDistance = col_double(),
##   ..   LightActiveDistance = col_double(),
##   ..   SedentaryActiveDistance = col_double(),
##   ..   VeryActiveMinutes = col_double(),
##   ..   FairlyActiveMinutes = col_double(),
##   ..   LightlyActiveMinutes = col_double(),
##   ..   SedentaryMinutes = col_double(),
##   ..   Calories = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
head(sleepday_df)
```

```
##           Id              SleepDay TotalSleepRecords TotalMinutesAsleep
## 1 1503960366 4/12/2016 12:00:00 AM                 1                327
## 2 1503960366 4/13/2016 12:00:00 AM                 2                384
## 3 1503960366 4/15/2016 12:00:00 AM                 1                412
## 4 1503960366 4/16/2016 12:00:00 AM                 2                340
## 5 1503960366 4/17/2016 12:00:00 AM                 1                700
## 6 1503960366 4/19/2016 12:00:00 AM                 1                304
##   TotalTimeInBed
```

```
## 1            346
## 2            407
## 3            442
## 4            367
## 5            712
## 6            320
```

```
str(sleepday_df)
```

```
## 'data.frame':    413 obs. of  5 variables:
##  $ Id                : num  1.5e+09 1.5e+09 1.5e+09 1.5e+09 1.5e+09 ...
##  $ SleepDay          : chr  "4/12/2016 12:00:00 AM" "4/13/2016 12:00:00 AM" "4/15/2016 12:00:00 AM" "
##  $ TotalSleepRecords : int  1 2 1 2 1 1 1 1 1 1 ...
##  $ TotalMinutesAsleep: int  327 384 412 340 700 304 360 325 361 430 ...
##  $ TotalTimeInBed    : int  346 407 442 367 712 320 377 364 384 449 ...
```

```
head(hourlysteps_df)
```

```
##           Id          ActivityHour StepTotal
## 1 1503960366 4/12/2016 12:00:00 AM       373
## 2 1503960366  4/12/2016 1:00:00 AM       160
## 3 1503960366  4/12/2016 2:00:00 AM       151
## 4 1503960366  4/12/2016 3:00:00 AM         0
## 5 1503960366  4/12/2016 4:00:00 AM         0
## 6 1503960366  4/12/2016 5:00:00 AM         0
```

```
str(hourlysteps_df)
```

```
## 'data.frame':    22099 obs. of  3 variables:
##  $ Id          : num  1.5e+09 1.5e+09 1.5e+09 1.5e+09 1.5e+09 ...
##  $ ActivityHour: chr  "4/12/2016 12:00:00 AM" "4/12/2016 1:00:00 AM" "4/12/2016 2:00:00 AM" "4/12/20
##  $ StepTotal   : int  373 160 151 0 0 0 0 0 250 1864 ...
```

Identify the column names for each dataframe.

```
colnames(dailyactivity_df)
```

```
##  [1] "Id"                       "ActivityDate"
##  [3] "TotalSteps"               "TotalDistance"
##  [5] "TrackerDistance"          "LoggedActivitiesDistance"
##  [7] "VeryActiveDistance"       "ModeratelyActiveDistance"
##  [9] "LightActiveDistance"      "SedentaryActiveDistance"
## [11] "VeryActiveMinutes"        "FairlyActiveMinutes"
## [13] "LightlyActiveMinutes"     "SedentaryMinutes"
## [15] "Calories"
```

```
colnames(sleepday_df)
```

```
## [1] "Id"                 "SleepDay"           "TotalSleepRecords"
## [4] "TotalMinutesAsleep" "TotalTimeInBed"
```

```
colnames(hourlysteps_df)
```

```
## [1] "Id"           "ActivityHour" "StepTotal"
```

Check the summary for each dataframe.

```
summary(dailyactivity_df)
```

```
##        Id            ActivityDate         TotalSteps    TotalDistance
##  Min.   :1.504e+09   Length:940         Min.   :    0   Min.   : 0.000
##  1st Qu.:2.320e+09   Class :character   1st Qu.: 3790   1st Qu.: 2.620
##  Median :4.445e+09   Mode  :character   Median : 7406   Median : 5.245
##  Mean   :4.855e+09                      Mean   : 7638   Mean   : 5.490
##  3rd Qu.:6.962e+09                      3rd Qu.:10727   3rd Qu.: 7.713
##  Max.   :8.878e+09                      Max.   :36019   Max.   :28.030
##  TrackerDistance  LoggedActivitiesDistance VeryActiveDistance
##  Min.   : 0.000   Min.   :0.0000           Min.   : 0.000
##  1st Qu.: 2.620   1st Qu.:0.0000           1st Qu.: 0.000
##  Median : 5.245   Median :0.0000           Median : 0.210
##  Mean   : 5.475   Mean   :0.1082           Mean   : 1.503
##  3rd Qu.: 7.710   3rd Qu.:0.0000           3rd Qu.: 2.053
##  Max.   :28.030   Max.   :4.9421           Max.   :21.920
##  ModeratelyActiveDistance LightActiveDistance SedentaryActiveDistance
##  Min.   :0.0000           Min.   : 0.000      Min.   :0.000000
##  1st Qu.:0.0000           1st Qu.: 1.945      1st Qu.:0.000000
##  Median :0.2400           Median : 3.365      Median :0.000000
##  Mean   :0.5675           Mean   : 3.341      Mean   :0.001606
##  3rd Qu.:0.8000           3rd Qu.: 4.782      3rd Qu.:0.000000
##  Max.   :6.4800           Max.   :10.710      Max.   :0.110000
##  VeryActiveMinutes FairlyActiveMinutes LightlyActiveMinutes SedentaryMinutes
##  Min.   :  0.00    Min.   :  0.00      Min.   :  0.0        Min.   :   0.0
##  1st Qu.:  0.00    1st Qu.:  0.00      1st Qu.:127.0        1st Qu.: 729.8
##  Median :  4.00    Median :  6.00      Median :199.0        Median :1057.5
##  Mean   : 21.16    Mean   : 13.56      Mean   :192.8        Mean   : 991.2
##  3rd Qu.: 32.00    3rd Qu.: 19.00      3rd Qu.:264.0        3rd Qu.:1229.5
##  Max.   :210.00    Max.   :143.00      Max.   :518.0        Max.   :1440.0
##     Calories
##  Min.   :   0
##  1st Qu.:1828
##  Median :2134
##  Mean   :2304
##  3rd Qu.:2793
##  Max.   :4900
```

```
summary(sleepday_df)
```

```
##        Id              SleepDay         TotalSleepRecords TotalMinutesAsleep
##  Min.   :1.504e+09   Length:413         Min.   :1.000     Min.   : 58.0
##  1st Qu.:3.977e+09   Class :character   1st Qu.:1.000     1st Qu.:361.0
##  Median :4.703e+09   Mode  :character   Median :1.000     Median :433.0
##  Mean   :5.001e+09                      Mean   :1.119     Mean   :419.5
```

```
##  3rd Qu.:6.962e+09                    3rd Qu.:1.000     3rd Qu.:490.0
##  Max.   :8.792e+09                    Max.   :3.000     Max.   :796.0
##  TotalTimeInBed
##  Min.   : 61.0
##  1st Qu.:403.0
##  Median :463.0
##  Mean   :458.6
##  3rd Qu.:526.0
##  Max.   :961.0
```

```
summary(hourlysteps_df)
```

```
##        Id            ActivityHour         StepTotal
##  Min.   :1.504e+09   Length:22099       Min.   :    0.0
##  1st Qu.:2.320e+09   Class :character   1st Qu.:    0.0
##  Median :4.445e+09   Mode  :character   Median :   40.0
##  Mean   :4.848e+09                      Mean   :  320.2
##  3rd Qu.:6.962e+09                      3rd Qu.:  357.0
##  Max.   :8.878e+09                      Max.   :10554.0
```

### 3.3 Cleaning and Formatting

Next, we will check for the unique IDs each dataframe contain.

```
n_distinct(dailyactivity_df$Id)
```

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

```
n_distinct(sleepday_df$Id)
```

```
## [1] 24
```

```
n_distinct(hourlysteps_df$Id)
```

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

There are 33 distinct participants in dailyactivity_df and hourlysteps_df, whereas there are 24 distinct participants in sleepday_df. We then check for any missing values in each dataframe.

```
sum(is.na(dailyactivity_df))
```

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

```
sum(is.na(sleepday_df))
```

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

```
sum(is.na(hourlysteps_df))
```

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

Seems like there are no missing values, which is great. Then, we will fix the format of datetime for two of the dataframes, dailyactivity_df and hourlysteps_df. We will be changing from string variable to datetime variable.

```
dailyactivity_df$ActivityDate <- mdy(dailyactivity_df$ActivityDate)

sleepday_df$SleepDay <- mdy_hms(sleepday_df$SleepDay)

sum(is.na(hourlysteps_df))
```

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

```
hourlysteps_df <- hourlysteps_df %>%
  rename(date_time = ActivityHour) %>%
  mutate(date_time = as.POSIXct(date_time, format ="%m/%d/%Y %I:%M:%S %p",
                                tz=Sys.timezone()))
```

# PHASE 4: ANALYZE

## 4.1 Merging Data

We will merge two of the dataframes, dailyactivity_df and sleepday_df by their IDs and Dates. But first, we must rename their date columns to 'Date'.

```
dailyactivity_df <- rename(dailyactivity_df, Date = ActivityDate)
sleepday_df <- rename(sleepday_df, Date = SleepDay)

merged_df <- merge(sleepday_df, dailyactivity_df, by = c('Id', 'Date'))
View(merged_df)
```

We will add another column called 'ActivityDay' in the merged_df dataframe which contains the days from Monday to Sunday and we will order them accordingly.

```
merged_df <- add_column(merged_df, ActivityDay = weekdays(merged_df$Date),
                        .after = "Id")

merged_df$ActivityDay <-ordered(merged_df$ActivityDay,
                                levels = c("Monday", "Tuesday",
                                           "Wednesday", "Thursday",
                                           "Friday", "Saturday", "Sunday"))

View(merged_df)
```

### 4.2 Summary Statistics

Let's have a look at the summary statistics of the dataframes. We will first find the average calories, total steps taken, total minutes asleep, and total distance traveled grouped by days.

```
weekday_activity <- merged_df %>%
  group_by(ActivityDay) %>%
  summarize (daily_steps = mean(TotalSteps), daily_sleep = mean(TotalMinutesAsleep),
             daily_calories = mean(Calories), daily_distance = mean(TotalDistance))

View(weekday_activity)
```

We will also find the average calories, total steps taken, total minutes asleep, and total distance traveled but grouped by IDs.

```
user_activity <- merged_df %>%
  group_by(Id) %>%
  summarize (daily_steps = mean(TotalSteps), daily_sleep = mean(TotalMinutesAsleep),
             daily_calories = mean(Calories), daily_distance = mean(TotalDistance))

View(user_activity)
```

We then categorized the users to 5 categories based on their daily steps taken. These 5 categories are based on this link. The summary statistics is shown below:

```
user_type <- user_activity %>%
  mutate(user_type = case_when(
    daily_steps < 5000 ~ "Sedentary",
    daily_steps >= 5000 & daily_steps <= 7499 ~ "Low Active",
    daily_steps >= 7500 & daily_steps <= 9999 ~ "Somewhat Active",
    daily_steps >= 10000 & daily_steps <= 12499 ~ "Active",
    daily_steps >= 12500 ~ "Highly Active"
  ))

View(user_type)
```

The user activity type is then being calculated in percentage to plot the pie chart later in the Share Phase. The type of user activity is also being ordered accordingly, based on how active they are.

```
user_type_percent <- user_type %>%
  group_by(user_type) %>%
  summarise(Total = n(), Total_Percent = Total/24) %>%
  mutate(labels = scales::percent(Total_Percent))

user_type_percent$user_type <- ordered(user_type_percent$user_type,
                                       levels = c("Highly Active", "Active",
                                                  "Somewhat Active",
                                                  "Low Active", "Sedentary"))

View(user_type_percent)
```

We will find the user's app usage based on the days they are active and the users can categorized into 3 categories. We then calculate the percentages of the 3 categories and ordered the usage type accordingly.

```
user_usage <- merged_df %>%
  group_by(Id) %>% summarise(app_usage_days = sum(n())) %>%
  mutate(usage = case_when(
    app_usage_days >= 1 & app_usage_days <= 10 ~ "Low Usage",
    app_usage_days >= 11 & app_usage_days <= 20 ~ "Moderate Usage",
    app_usage_days >= 21 ~ "High Usage"
  ))

user_usage_percent <- user_usage %>%
  group_by(usage) %>%
  summarise(Total = n(), Total_Percent = Total/24) %>%
  mutate(labels = scales::percent(Total_Percent))

user_usage_percent$usage <- ordered(user_usage_percent$usage,
                                    levels = c("High Usage", "Moderate Usage",
                                               "Low Usage"))

View(user_usage_percent)
```

For the hourlysteps_df dataframe, we split the datetime into 2 columns, 'Date' and 'Time' to better visualize it in the Share Phase.

```
hourlysteps_df <- hourlysteps_df %>%
  separate(date_time, into = c("Date", "Time"), sep = " ") %>%
  mutate(Date = ymd(Date))

View(hourlysteps_df)
```

We then find the average steps taken grouped by time so that we can find out how many steps taken at a specific time.

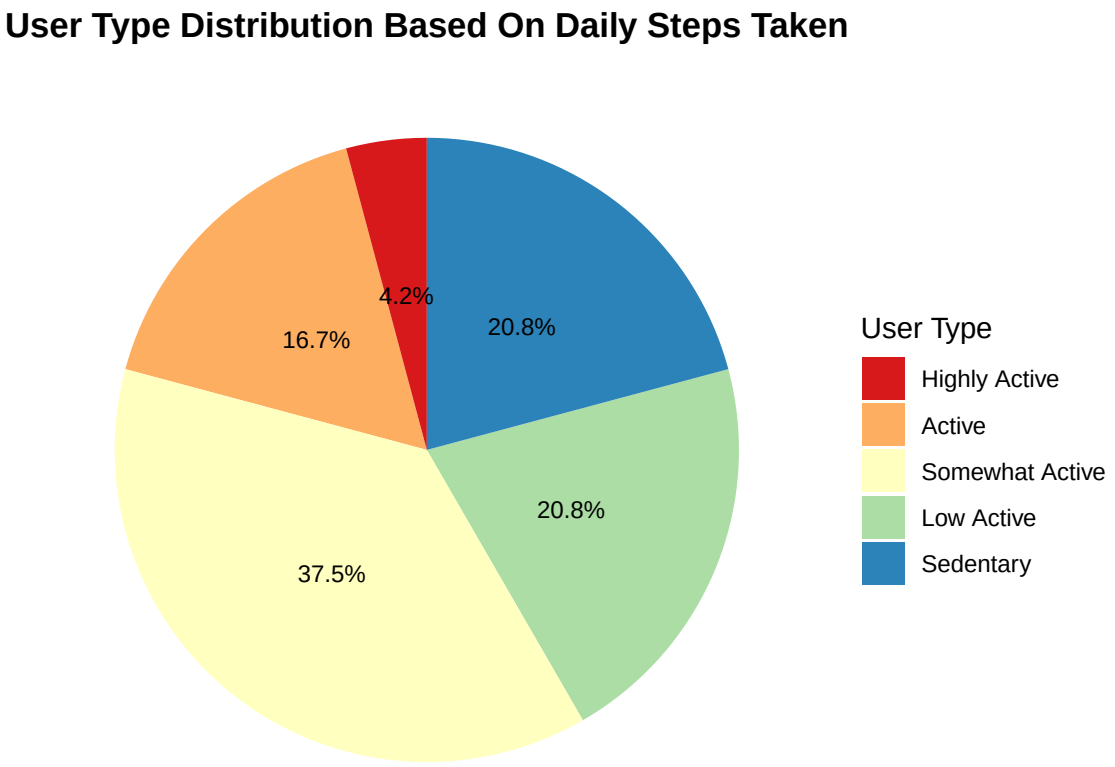
```
hourly_steps <- hourlysteps_df %>% group_by(Time) %>%
  summarise(mean_steps = mean(StepTotal))

View(hourly_steps)
```

## PHASE 5: SHARE PHASE

### 5.1 User Type Distribution Based On Daily Steps Taken

We used pie chart to visualize the user type distribution based on the steps they have taken daily to view which user type is the majority.

```
ggplot(user_type_percent, aes(x = "", y = Total_Percent, fill = user_type)) +
  geom_bar(width = 1, stat = "identity") + coord_polar("y", start = 0) +
  theme_minimal() + theme(axis.title.x = element_blank(), axis.title.y = element_blank(), panel.border =
  labs(title = "User Type Distribution Based On Daily Steps Taken", fill = "User Type")
```

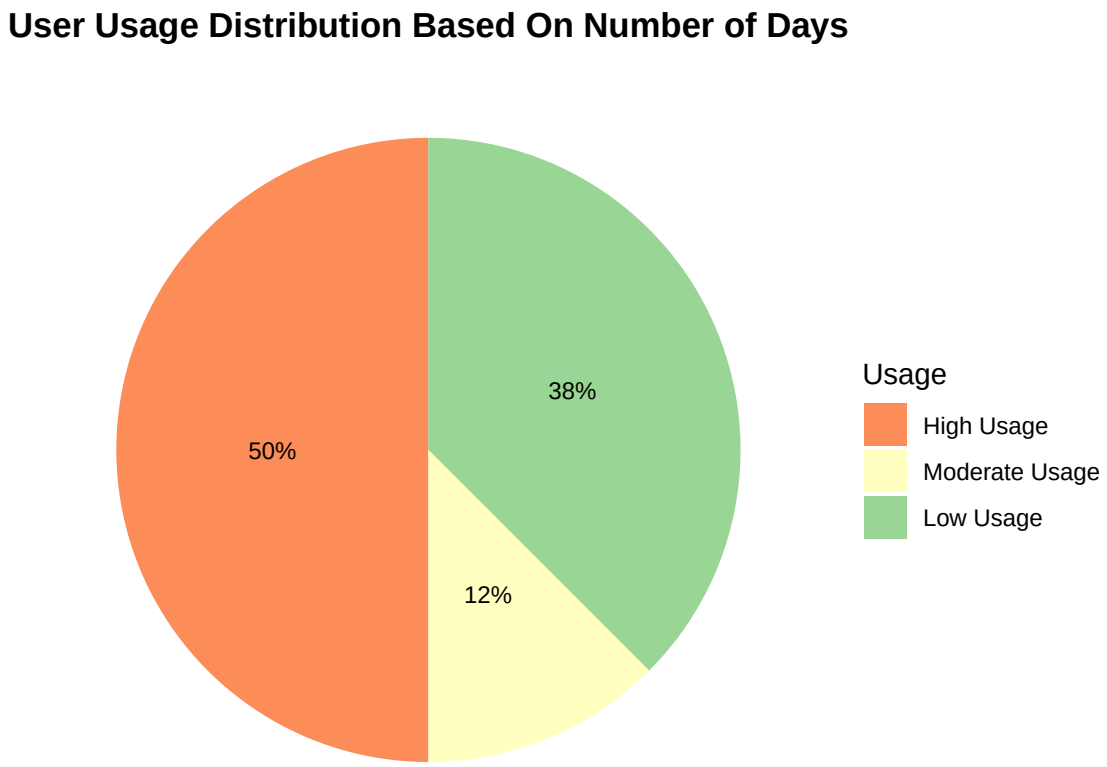



It is shown that majority of the users are type 'Somewhat Active'. User types 'Low Active' and 'Sedentary' share the same percentage, which is 20.8%. The 'Highly Active' user type is the least, only 4.2%.

### 5.2 User Usage Distribution Based On Number of Days

The user usage is also being visualized using a pie chart to view which usage type is the majority.

```
ggplot(user_usage_percent, aes(x = "", y = Total_Percent, fill = usage)) +
  geom_bar(width = 1, stat = "identity") + coord_polar("y", start = 0) +
  theme_minimal() + theme(axis.title.x = element_blank(), axis.title.y = element_blank(), panel.border =
  labs(title = "User Usage Distribution Based On Number of Days", fill = "Usage")
```

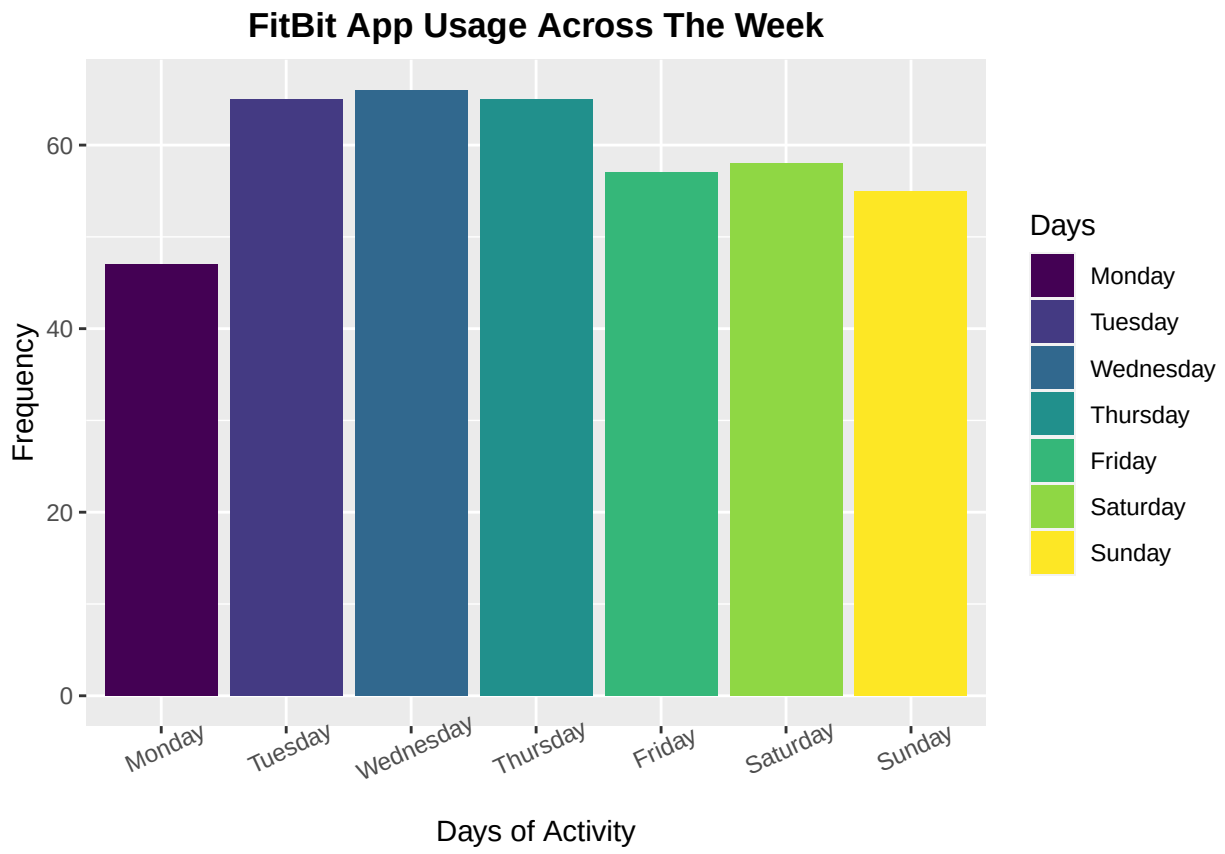



We can clearly see that half of the users belong to the 'High Usage' category where their app usage is more than 21 days. Only 12% of users belong to the 'Moderate Usage' which their app usage is between 11 and 20 days.

### 5.3 FitBit App Usage Across The Week

We will now check the FitBit app usage across the week.

```
ggplot(merged_df, aes(x = ActivityDay, fill = ActivityDay)) +
  geom_bar() + theme(axis.text.x = element_text(angle = 25),
                     plot.title = element_text(hjust = 0.5, face = "bold")) + labs(title = "FitBit App
       x = "Days of Activity", y = "Frequency", fill = "Days")
```




We found out that the top 3 highest usage days are from Tuesday to Thursday, then the usage starts to drop from Friday till Monday.

### 5.4 Calories Burnt vs. Total Steps Taken

We will now visualize 'Calories Burnt vs. Total Steps Taken' using a scatterplot to observe the relationship between these two.

```
ggplot(merged_df, aes(x = TotalSteps, y = Calories)) +
  geom_point(color = "red") + geom_smooth() +
  geom_vline(xintercept = mean(merged_df$TotalSteps), color = "darkgoldenrod3") +
  geom_hline(yintercept = mean(merged_df$Calories), color = "purple") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold")) +
  labs(title = "Calories Burnt vs. Total Steps Taken", x = "Total Steps Taken", y = "Calories Burnt") +
  annotate("text", x = 11000, y = 500, label = "Average Steps Taken",
           color = "darkgoldenrod3", size = 2.5) +
  annotate("text", x = 19700, y = 2300, label = "Average Calories Burnt", color = "purple", size = 2.5)
```

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

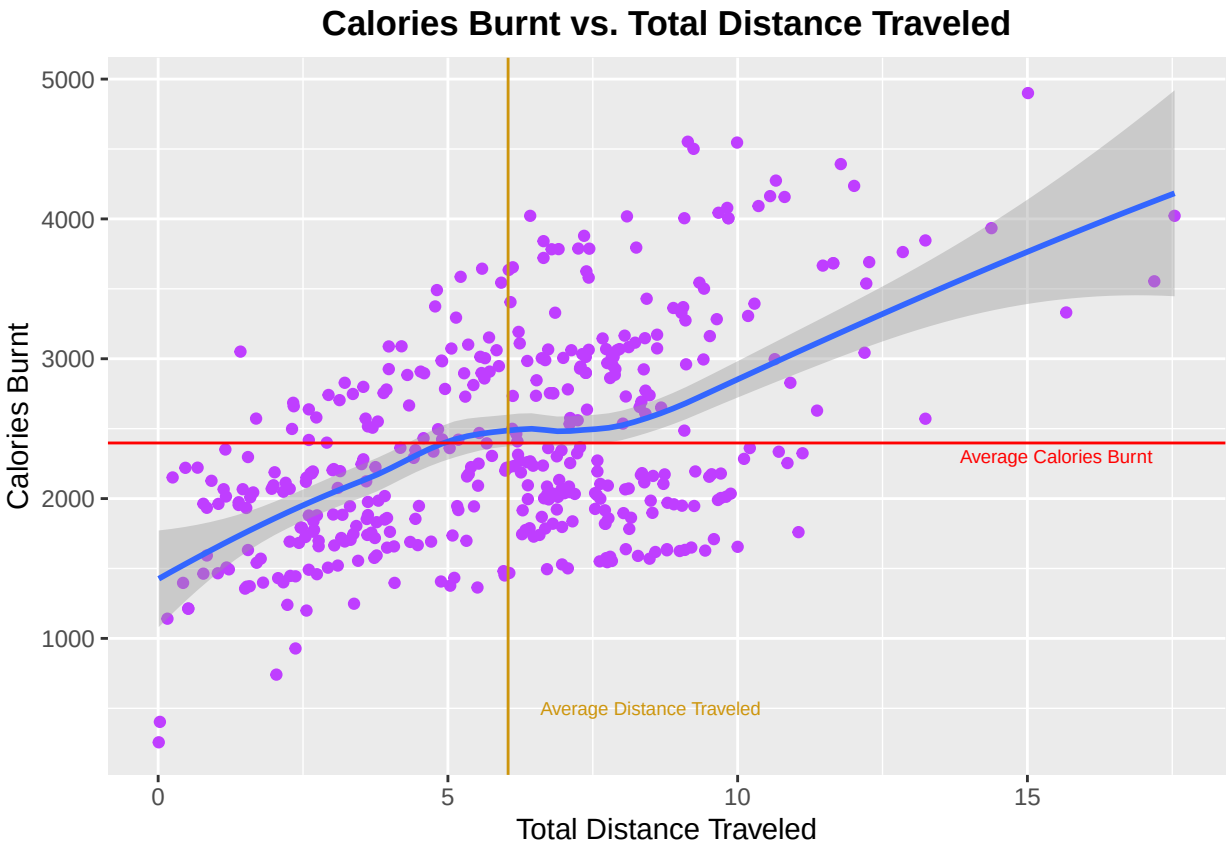

We can clearly see that there is a positive relationship between the total steps taken and the calories burnt. The more steps we take, the more calories we burn.

### 5.5 Calories Burnt vs. Total Distance Traveled

We will now visualize 'Calories Burnt vs. Total Distance Traveled' using a scatterplot to observe the relationship between these two.

```
ggplot(merged_df, aes(x = TotalDistance, y = Calories)) +
  geom_point(color = "darkorchid1") + geom_smooth() +
  geom_vline(xintercept = mean(merged_df$TotalDistance), color = "darkgoldenrod3") +
  geom_hline(yintercept = mean(merged_df$Calories), color = "red") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold")) +
  labs(title = "Calories Burnt vs. Total Distance Traveled",
       x = "Total Distance Traveled", y = "Calories Burnt") +
  annotate("text", x = 8.5, y = 500, label = "Average Distance Traveled", color = "darkgoldenrod3", size
  annotate("text", x = 15.5, y = 2300, label = "Average Calories Burnt", color = "red", size = 2.5)
```

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



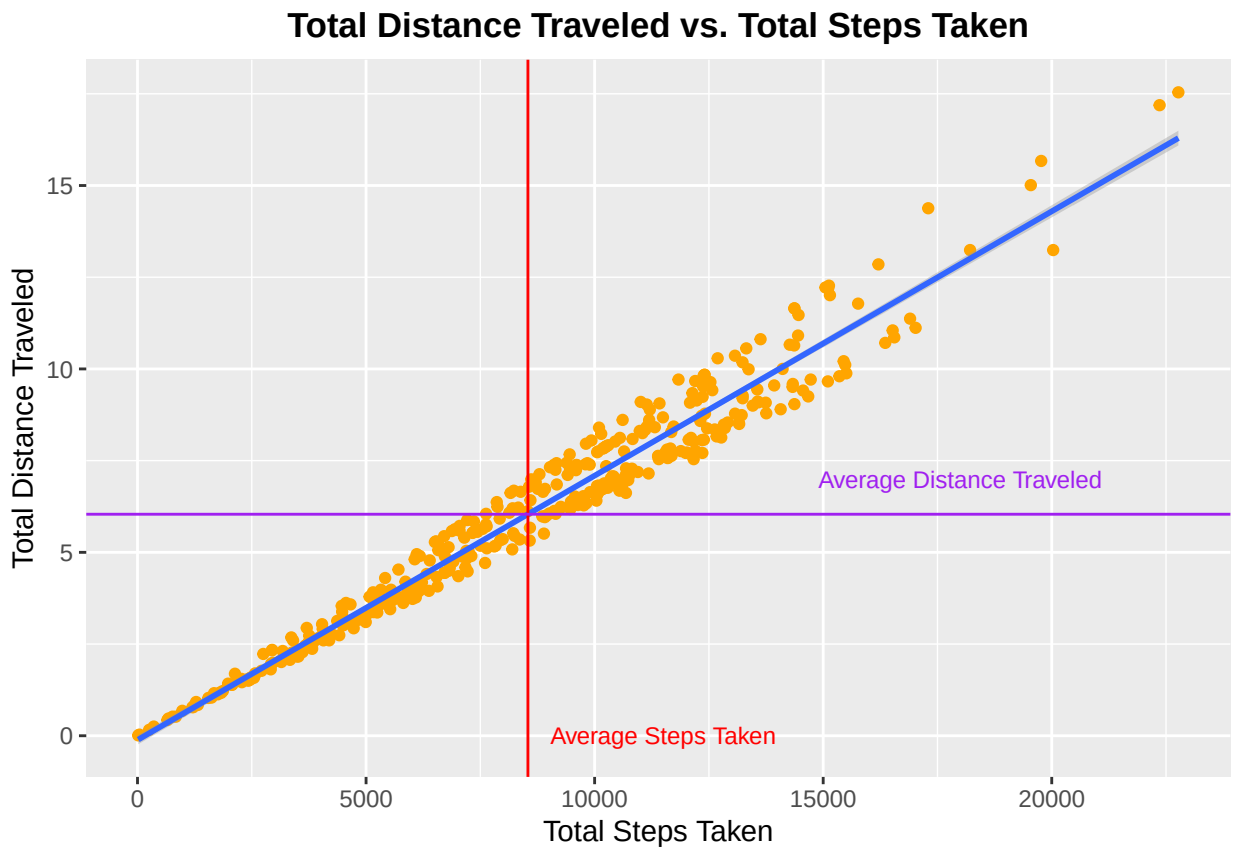

We can also clearly see that there is a positive correlation between the total distance traveled and the calories burnt. The further we traveled, the more calories we burn.

### 5.6 Total Distance Traveled vs. Total Steps Taken

We can observe that the total steps taken and total distance traveled have a positive relationship with calories burnt. So, let's visualize the relationship between the total steps taken and total distance traveled and see whether there is a linear relationship between these two.

```
ggplot(merged_df, aes(x = TotalSteps, y = TotalDistance)) +
  geom_point(color = "orange") + geom_smooth(method = "lm") +
  geom_vline(xintercept = mean(merged_df$TotalSteps), color = "red") +
  geom_hline(yintercept = mean(merged_df$TotalDistance), color = "purple") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold")) +
  labs(title = "Total Distance Traveled vs. Total Steps Taken",
       x = "Total Steps Taken", y = "Total Distance Traveled") +
  annotate("text", x = 18000, y = 7, label = "Average Distance Traveled", color = "purple", size = 3) +
  annotate("text", x = 11500, y = 0, label = "Average Steps Taken",
           color = "red", size = 3)
```

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

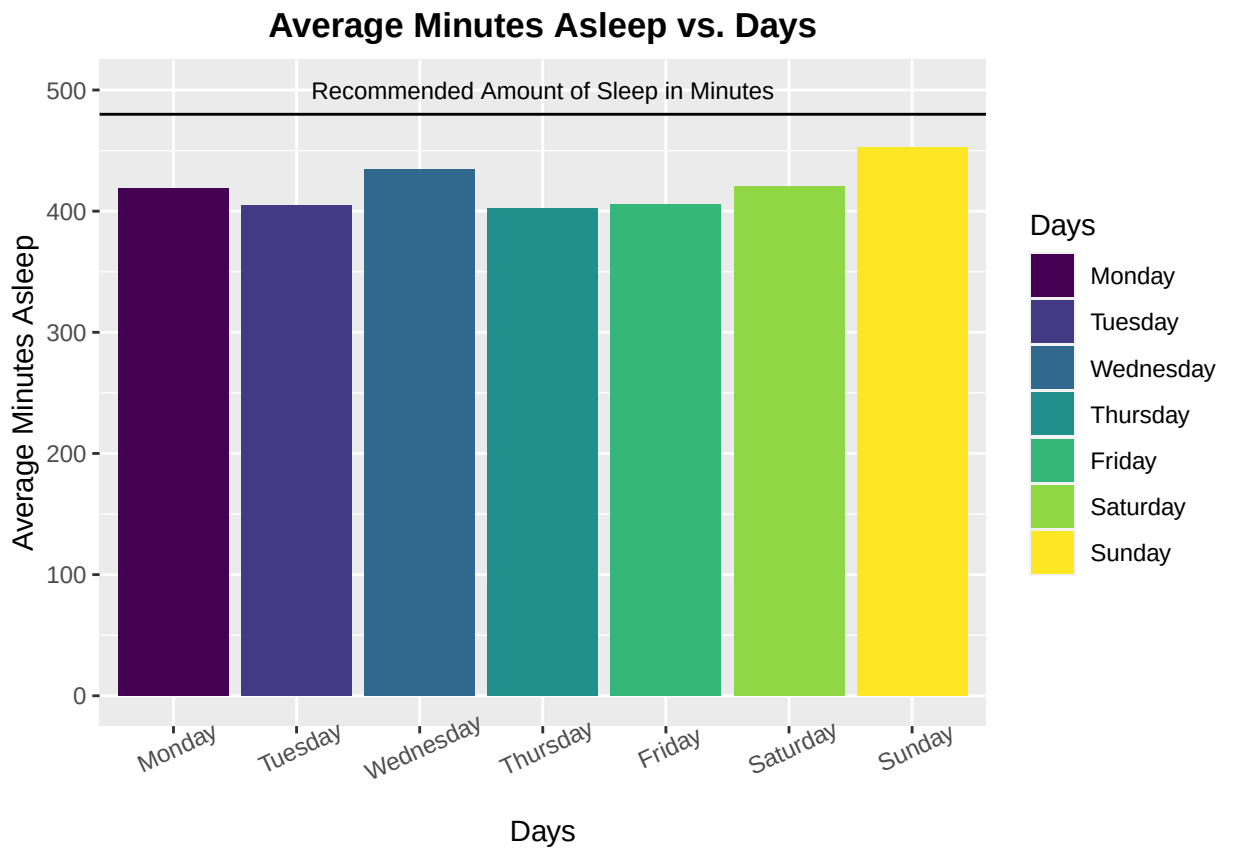



Surprisingly, there is a positive linear correlation between the total steps taken and total distance traveled. Hence, this is why there previous 2 scatterplots look similar.

### 5.7 Average Minutes Asleep vs. Days

Let's visualize the average minutes the respondents sleep on each day across the week.

```
ggplot(weekday_activity, aes(x = ActivityDay, y = daily_sleep, fill = ActivityDay)) +
  geom_col() + theme(axis.text.x = element_text(angle = 25),
                     plot.title = element_text(hjust = 0.5)) +
  geom_hline(yintercept = 480) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold")) +
  labs(title = "Average Minutes Asleep vs. Days",
       x = "Days", y = "Average Minutes Asleep", fill = "Days") +
  annotate("text", x = "Thursday", y = 500,
           label = "Recommended Amount of Sleep in Minutes", size = 3)
```

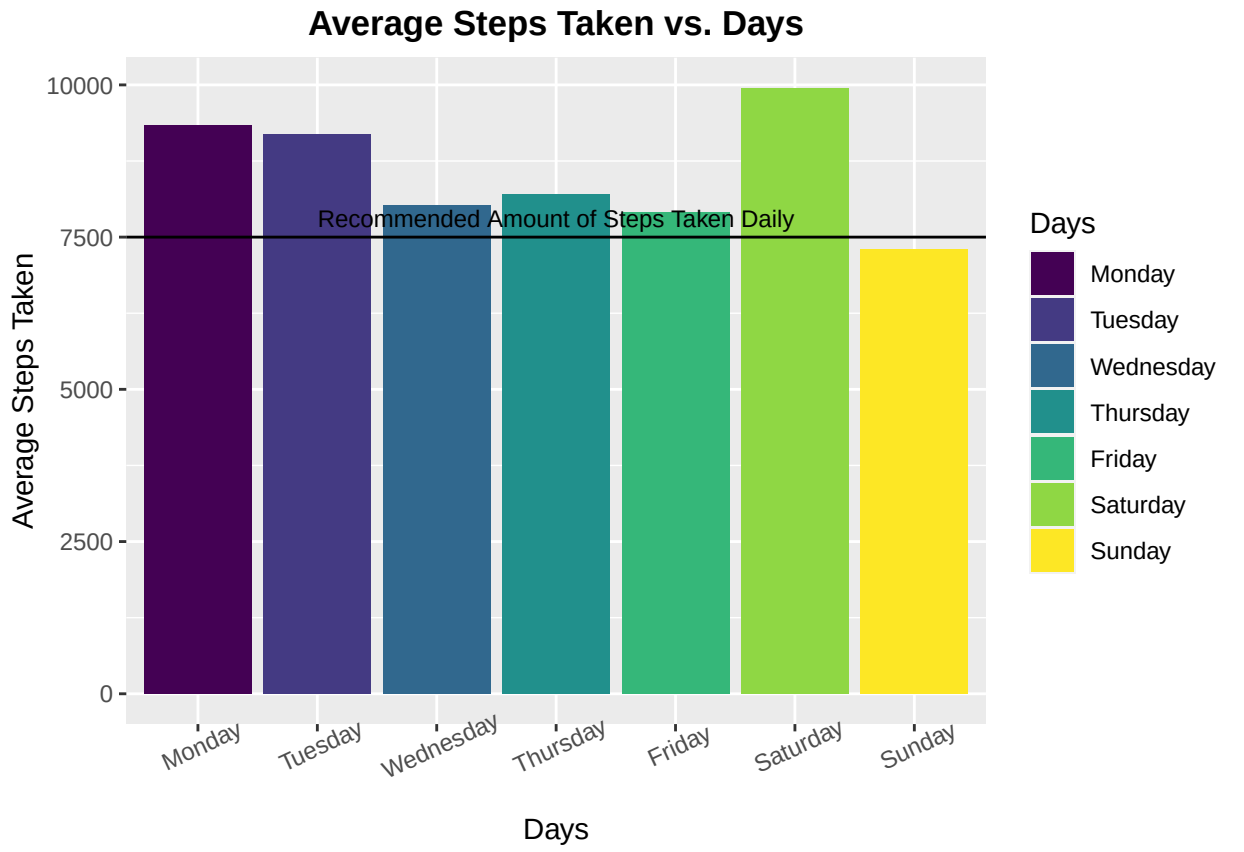



We can observe that none of the respondents get an average 8 hours or 480 minutes of sleep per day. The closet to that amount of sleeping time is Sunday.

### 5.8 Average Steps Taken vs. Days

According to this link, 7500 steps per day is considered somewhat active. Hence, 7500 steps per day is the recommended amount steps that should be taken.

```
ggplot(weekday_activity, aes(x = ActivityDay, y = daily_steps, fill = ActivityDay)) +
  geom_col() + theme(axis.text.x = element_text(angle = 25),
                     plot.title = element_text(hjust = 0.5)) +
  geom_hline(yintercept = 7500) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold")) +
  labs(title = "Average Steps Taken vs. Days",
       x = "Days", y = "Average Steps Taken", fill = "Days") +
  annotate("text", x = "Thursday", y = 7800,
           label = "Recommended Amount of Steps Taken Daily", size = 3)
```



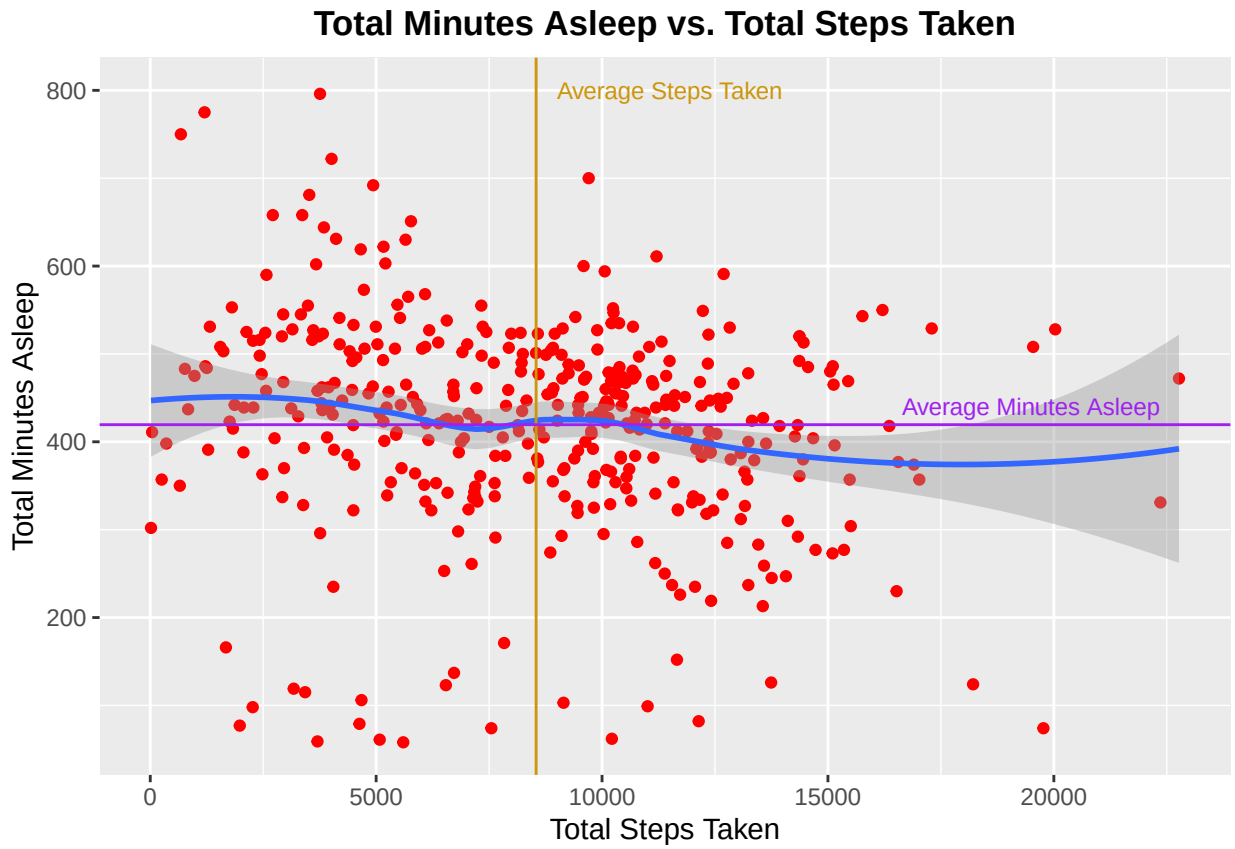

Based on the visualization, the respondents walk the recommended amount steps daily except for Sunday.

### 5.9 Total Minutes Asleep vs. Total Steps Taken

We would like to know whether there is a relationship between the total steps taken and the total minutes asleep. Hence, a scatterplot is plotted for these two variables. We assume that the more steps the respondents take, the more time they will spend sleeping to regain their energy.

```
ggplot(merged_df, aes(x = TotalSteps, y = TotalMinutesAsleep)) +
  geom_point(color = "red") + geom_smooth() +
  geom_vline(xintercept = mean(merged_df$TotalSteps), color = "darkgoldenrod3") +
  geom_hline(yintercept = mean(merged_df$TotalMinutesAsleep), color = "purple") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold")) +
  labs(title = "Total Minutes Asleep vs. Total Steps Taken", x = "Total Steps Taken", y = "Total Minutes
  annotate("text", x = 11500, y = 800, label = "Average Steps Taken",
           color = "darkgoldenrod3", size = 3) +
  annotate("text", x = 19500, y = 440, label = "Average Minutes Asleep", color = "purple", size = 3)
```

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

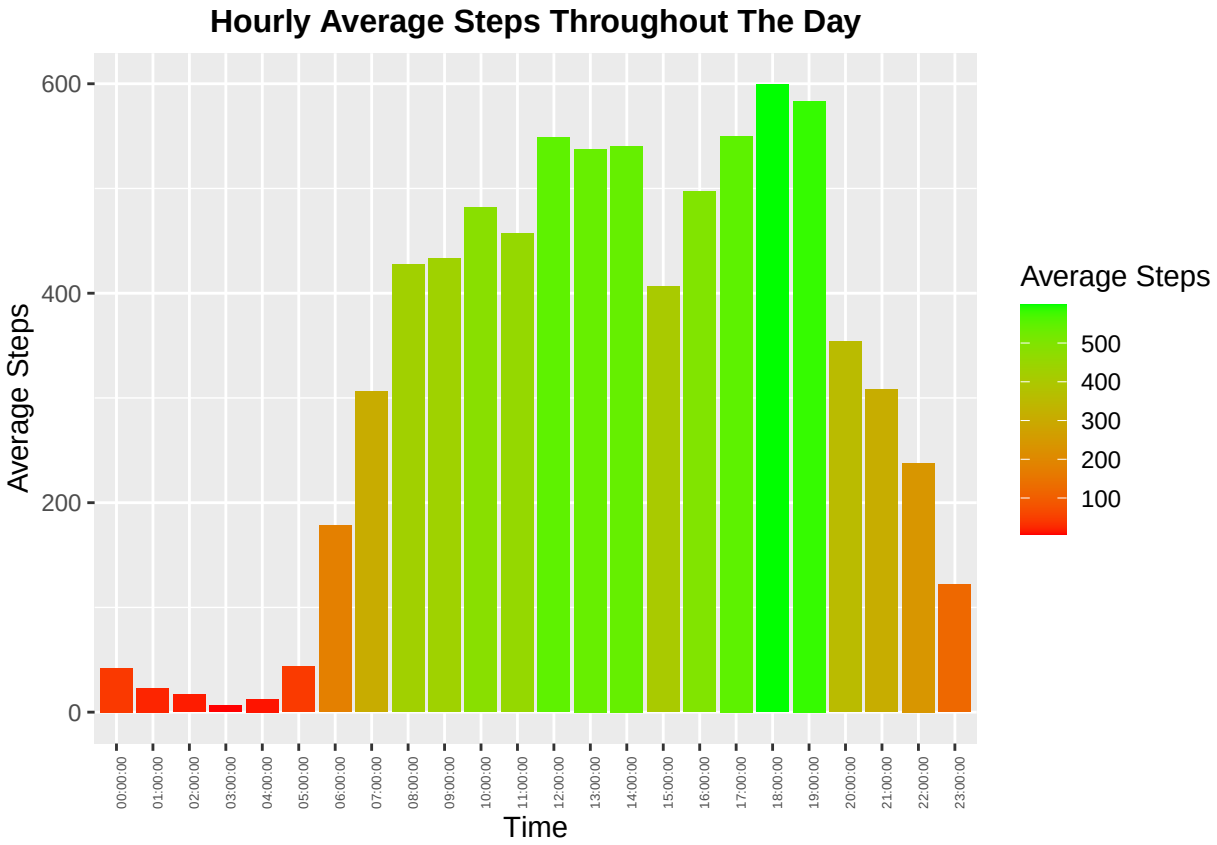



Apparently, there is no correlation between 2 variables at all.

### 5.10 Hourly Average Steps Throughout The Day

We now would like to visualize how many steps the respondents take during different hours in a day.

```
ggplot(hourly_steps, aes(x = Time, y = mean_steps, fill = mean_steps)) + geom_col() + scale_fill_gradie
  theme(axis.text.x = element_text(angle = 90, size = 5),
        plot.title = element_text(hjust = 0.5, face = "bold", size = 12)) + labs(title = "Hourly Averag
```




It is clear that the respondents are active between 8 am and 7 pm with an average of more than 400 steps. The respondents are the most active from 12 pm to 2 pm and 5 pm to 7 pm with an average of more than 500 steps.

## PAHSE 6: ACT

### 6.1 Conclusion

Based on my analysis, there are some interesting insights that I have found from these datasets. There is a positive linear correlation between total steps taken and total distance traveled. Due to this, the correlation between these 2 variables and calories burnt are similar, which is positive as visualized by scatterplots.

Majority of the users are type 'Somewhat Active' based on the the steps they have taken daily and half of them belong to the 'High Usage' category where their app usage is more than 21 days. The FitBit app usage across the week is peak on Wednesday.

All of the respondents do not get an average of 8 hours or 480 minutes of sleep daily. However, the respondents sleep more time on sunday as compared to other days. It is recommended to walk 7500 steps per day and the respondents meet the recommended amount on each day except for Sunday.

There is no correlation between the total steps taken and the total minutes asleep based on the scatterplot plotted, which is surprising. From the visualization, the respondents are the most active from 12 pm to 2 pm and 5 pm to 7 pm with an average of more than 500 steps.

### 6.2 Recommendations

There are some recommendations that can be made to help improve Bellabeat app, future analysis of their data, and the company's growth in general:

- Collect more responses from users to improve the analysis.
- Collect the data directly from the users instead of using a third-party provider.
- Notify the users if they do not reach the minimum number of steps taken per day, which is 7500 steps as a reminder to benefit their health.
- Send notifications about latest health updates to users who are type 'Low Usage' or users who are not active for more than 10 days as a reminder to use Bellabeat app.
- Send alert notifications to users who do not get the recommended amount of sleep, which is 8 hours or 480 minutes a day to remind them to get enough sleep and the app can also provide some sleep techniques to help users who potentially have insomnia.
- Create a reward system for the users based on their total steps taken daily. Points will be given to users for a certain number of steps they have reached daily and the users can accumulate these points to get a discount when buying other Bellabeat products.