# Google Data Analytics - Case Study

How Can a Wellness Technology Company Play It Smart?

Lisa Krapp

### Introduction

This case study was completed as part of the **Google Data Analytics Certificate** on Coursera. The project follows the data analysis process—**ask, prepare, process, analyze, share, and act**—to explore smart device usage trends and provide insights for **Bellabeat**, a high-tech manufacturer of health-focused products for women.

As a junior data analyst in this scenario, my goal is to analyze fitness tracker data to uncover patterns in user behavior. The insights gained will help guide **Bellabeat's marketing strategy**, supporting the company's growth in the global smart device market.

## Business Task

Bellabeat aims to expand its presence in the **global smart device market** by leveraging **data-driven marketing strategies**. To achieve this, the company seeks to understand **how consumers use smart devices** and how these insights can be applied to Bellabeat's products.

As a junior data analyst on the **marketing analytics team**, my task is to:

1. **Analyze smart device usage trends** from publicly available fitness tracker data.
2. **Identify key patterns in user behavior**, including activity levels, sleep habits, and other wellness-related metrics.
3. **Apply these insights to Bellabeat's products** to guide marketing strategy.

### Key Business Questions

To address this task, the following **guiding questions** will be explored:

- **What are some trends in smart device usage?**
- **How could these trends apply to Bellabeat customers?**
- **How could these trends influence Bellabeat's marketing strategy?**

### Expected Outcomes

The findings from this analysis will help **optimize Bellabeat's marketing efforts** by:

- Identifying key user behaviors and preferences.
- Understanding how Bellabeat products align with consumer needs.
- Providing **data-backed recommendations** for future marketing campaigns.

This analysis will contribute to **Bellabeat's growth strategy** by ensuring its marketing approach is aligned with **real-world smart device usage trends**.

## Prepare & Process

These are all the imported packages:

```
library(tidyverse)
library(ggcorrplot)
library(slider)
```

The basis of my analysis will be the FitBit Fitness Tracker Data, which is publicly available here. Metadata about the tables like column descriptions are found here.

Upon inspection of the tables after downloading the dataset, the tables are split into **two time periods** (1: 12.03.2016 - 11.04.2026, 2: 12.04.2016 - 12.05.2016), which I will merge for a more structured analysis. Furthermore, some tables are available in a long and wide format, which is redundant information. I will only use the long format for my analysis. To be as accurate as possible, I will always work on the smallest time frame available for the different attributes. All steps needed for the analysis will be done in this file.

The merged tables will be the following:

- weight
- METs_minutes
- Sleep_minutes
- Intensities_minutes
- Heartrate_seconds
- Calories_minutes
- Steps_minutes
- Activity_days

The processed data will be stored in a seperate folder from the original data:

```
output_folder <- "./processed_data"

if (!dir.exists(output_folder)) {
  dir.create(output_folder)
  message("Folder created: ", output_folder)
} else {
  message("Folder already exists: ", output_folder)
}
```

In the next step I will always load the corresponding tables from both time frames, check what steps are needed to merge them and then merge them. Before merging, I will always change the date format (which is character in the tables) to a date time format. Because I want to analyze usage trends for different day times and will aggregate the data across the different dates, I will split this column into Date and Time. This function is handy for a first inspection of both tables

```
compare_tables <- function(table_name_1, table_name_2) {
  time_frame_1 <- "./fitbit_download/mturkfitbit_export_3.12.16-4.11.16/Fitabase Data 3.12.16-4.11.16"
  time_frame_2 <- "./fitbit_download/mturkfitbit_export_4.12.16-5.12.16/Fitabase Data 4.12.16-5.12.16"

  table_path_1 <- file.path(time_frame_1, table_name_1)
  table_path_2 <- file.path(time_frame_2, table_name_2)

  table_1 <- read_csv(table_path_1)
  table_2 <- read_csv(table_path_2)
```

```r
  head(table_1)
  head(table_2)

  glimpse(table_1)
  glimpse(table_2)
  print("Summary Table 1:")
  print(summary(table_1))
  print("Summary Table 2:")
  print(summary(table_2))

  # Return tables as a list
  return(list(table_1 = table_1, table_2 = table_2))
}
```

## Weight info

```r
weight_table_name <- "weightLogInfo_merged.csv"

weight_tables <- compare_tables(weight_table_name, weight_table_name)
weight_1 <- weight_tables$table_1
weight_2 <- weight_tables$table_2
```

The tables seem to have the same information structure, so I can easily merge them.

```r
merge_and_save <- function(table_1, table_2, output_folder, output_filename) {
  # Merge the tables (stacking them)
  merged_table <- bind_rows(table_1, table_2)

  # Ensure the output folder exists
  if (!dir.exists(output_folder)) {
    dir.create(output_folder, recursive = TRUE)
  }

  # Define the full output path
  output_path <- file.path(output_folder, output_filename)

  # Save as CSV
  write_csv(merged_table, output_path)

  # Print message and return the merged table
  message("Merged table saved to: ", output_path)
  return(merged_table)
}

weight_1 <- weight_1 %>%
  mutate(DateTime = mdy_hms(Date),  # Convert to datetime
         Date = as.Date(DateTime),  # Extract only the date
         Time = format(DateTime, "%H:%M:%S")) %>% # Extract only the time
  select(-DateTime)

weight_2 <- weight_2 %>%
  mutate(DateTime = mdy_hms(Date),
         Date = as.Date(DateTime),
         Time = format(DateTime, "%H:%M:%S"))  %>%
  select(-DateTime)

merge_and_save(weight_1, weight_2, output_folder, "Weight.csv")
```

### METs

```
METs_table_name_1 <- "minuteMETsNarrow_merged.csv"
METs_table_name_2 <- "minuteMETsNarrow_merged.csv"

tables <- compare_tables(METs_table_name_1, METs_table_name_2)
METs_table_1 <- tables$table_1
METs_table_2 <- tables$table_2
```

The columns in the tables are matching, so they are ready for merging. However, this is written in the metadata about these tables: "All MET values exported from Fitabase are multiplied by 10. Please divide by 10 to get accurate MET values". So before merging, all MET values have to be devided by 10.

```
METs_table_1 = rename(METs_table_1, Date = ActivityMinute)
METs_table_2 = rename(METs_table_2, Date = ActivityMinute)

METs_table_1 <- METs_table_1 %>%
  mutate(DateTime = mdy_hms(Date),   # Convert to datetime
         Date = as.Date(DateTime),   # Extract only the date
         Time = format(DateTime, "%H:%M:%S"),# Extract only the time
         across(contains("MET"), ~ . / 10)) %>%
  select(-DateTime)

METs_table_2 <- METs_table_2 %>%
  mutate(DateTime = mdy_hms(Date),
         Date = as.Date(DateTime),
         Time = format(DateTime, "%H:%M:%S"),
         across(contains("MET"), ~ . / 10)) %>%
  select(-DateTime)


merge_and_save(METs_table_1, METs_table_2, output_folder, "METs_minutes.csv")
```

### Sleep

```
sleep_table_name <- "minuteSleep_merged.csv"

sleep_tables <- compare_tables(sleep_table_name, sleep_table_name)
sleep_1 <- sleep_tables$table_1
sleep_2 <- sleep_tables$table_2
```

From the metadata: Note: sleep minute data is commonly exported with :30 sec. In this case, the "floor" of the time value can be used to convert to whole minutes. Value indicating the sleep state. 1 = asleep, 2 = restless, 3 = awake Furthermore, "date" will be renamed to "Date", as in all other tables.

```
sleep_1 <- sleep_1 %>%
  rename(Date = date,
         SleepState = value) %>%
  mutate(DateTime = mdy_hms(Date),
         DateTime = floor_date(DateTime, "minute"),
         Date = as.Date(DateTime),
         Time = format(DateTime, "%H:%M:%S")) %>%
  select(-DateTime)

sleep_2 <- sleep_2 %>%
  rename(Date = date,
         SleepState = value) %>%
  mutate(DateTime = mdy_hms(Date),
         DateTime = floor_date(DateTime, "minute"),
         Date = as.Date(DateTime),
         Time = format(DateTime, "%H:%M:%S")) %>%
  select(-DateTime)

merge_and_save(sleep_1, sleep_2, output_folder, "Sleep_minutes.csv")
```

### Intensities

```
Intensities_name <- "minuteIntensitiesNarrow_merged.csv"
tables <- compare_tables(Intensities_name, Intensities_name)
Intensities_1 <- tables$table_1
Intensities_2 <- tables$table_2
```

Rename ActivityMinute to Date, put in correct format and then merge:

```
Intensities_1 <- Intensities_1 %>%
  rename(Date = ActivityMinute) %>%
  mutate(DateTime = mdy_hms(Date),
         Date = as.Date(DateTime),
         Time = format(DateTime, "%H:%M:%S"))  %>%
  select(-DateTime)

Intensities_2 <- Intensities_2 %>%
  rename(Date = ActivityMinute) %>%
  mutate(DateTime = mdy_hms(Date),
         Date = as.Date(DateTime),
         Time = format(DateTime, "%H:%M:%S"))  %>%
  select(-DateTime)

merge_and_save(Intensities_1, Intensities_2, output_folder, "Intensities_minutes.csv")
```

### Heartrate

```
file_name <- "heartrate_seconds_merged.csv"
tables <- compare_tables(file_name, file_name)
heartrates_1 <- tables$table_1
heartrates_2 <- tables$table_2
```

Rename Time column to Date, put into correct format and then merge:

```
heartrates_1 <- heartrates_1  %>%
  rename(Heartrate = Value) %>%
  mutate(DateTime = mdy_hms(Time),
         Date = as.Date(DateTime),
         Time = format(DateTime, "%H:%M:%S"))  %>%
  select(-DateTime)
heartrates_2 <- heartrates_2  %>%
  rename(Heartrate = Value) %>%
  mutate(DateTime = mdy_hms(Time),
         Date = as.Date(DateTime),
         Time = format(DateTime, "%H:%M:%S"))  %>%
  select(-DateTime)


merge_and_save(heartrates_1, heartrates_2, output_folder, "Heartrates_seconds.csv")
```

## Calories

The metadata states, that the calories tables contain information about the estimated calories **burned**.

```
file_name <- "minuteCaloriesNarrow_merged.csv"
tables <- compare_tables(file_name, file_name)
calories_1 <- tables$table_1
calories_2 <- tables$table_2
```

```
calories_1 <- calories_1  %>%
  rename(Time = ActivityMinute) %>%
  mutate(DateTime = mdy_hms(Time),
         Date = as.Date(DateTime),
         Time = format(DateTime, "%H:%M:%S"))  %>%
  select(-DateTime)

calories_2 <- calories_2  %>%
  rename(Time = ActivityMinute) %>%
  mutate(DateTime = mdy_hms(Time),
         Date = as.Date(DateTime),
         Time = format(DateTime, "%H:%M:%S"))  %>%
  select(-DateTime)

merge_and_save(calories_1, calories_2, output_folder, "Calories_minutes.csv")
```

## Steps

```
file_name <- "minuteStepsNarrow_merged.csv"
tables <- compare_tables(file_name, file_name)
steps_1 <- tables$table_1
steps_2 <- tables$table_2
```

```
steps_1 <- steps_1  %>%
  rename(Time = ActivityMinute) %>%
  mutate(DateTime = mdy_hms(Time),
         Date = as.Date(DateTime),
         Time = format(DateTime, "%H:%M:%S"))  %>%
  select(-DateTime)

steps_2 <- steps_2  %>%
  rename(Time = ActivityMinute) %>%
  mutate(DateTime = mdy_hms(Time),
```

```r
         Date = as.Date(DateTime),
         Time = format(DateTime, "%H:%M:%S"))  %>%
  select(-DateTime)

merge_and_save(steps_1, steps_2, output_folder, "Steps_minutes.csv")
```

### Daily Activity

```r
file_name <- "dailyActivity_merged.csv"
tables <- compare_tables(file_name, file_name)
daily_1 <- tables$table_1
daily_2 <- tables$table_2
```

The ActivityDate column is only in MMDDYYYY and not in Date Time, this will be changed. Then, the tables will be merged.

```r
daily_1 <- daily_1  %>%
  rename(Date = ActivityDate) %>%
  mutate(Date = mdy(Date),
         Time = format(as.POSIXct(paste("23:59:59"), format = "%H:%M:%S"), "%H:%M:%S"))
daily_2 <- daily_2  %>%
  rename(Date = ActivityDate) %>%
  mutate(Date = mdy(Date),
         Time = format(as.POSIXct(paste("23:59:59"), format = "%H:%M:%S"), "%H:%M:%S"))

merge_and_save(daily_1, daily_2, output_folder, "Activities_days.csv")
```

The two time frames are now merged and all data is in the correct format!

## Analyze

In the Analyze step, I will aggregate my data. I want to look at the data available for each of the 30 FitBit Users, but also look at patterns between users. Furthermore, I want to look at correlations between the different attributes.

First, all tables will be read:

```r
data_folder <- "./processed_data"
activities <- read_csv(file.path(data_folder, "Activities_days.csv"))
calories <- read_csv(file.path(data_folder, "Calories_minutes.csv"))
heartrates <- read_csv(file.path(data_folder, "Heartrates_seconds.csv"))
intensities <- read_csv(file.path(data_folder, "Intensities_minutes.csv"))
METs <- read_csv(file.path(data_folder, "METs_minutes.csv"))
sleep <- read_csv(file.path(data_folder, "Sleep_minutes.csv"))
steps <- read_csv(file.path(data_folder, "Steps_minutes.csv"))
weight <- read_csv(file.path(data_folder, "weight.csv"))
```

All tables share a column called “Id”, which gives information about which user logged that information. To only include significant data, I want to see what information is available for how many persons. That way, I can draw informed conclusions from the data.

```r
# Create a summary of unique users per dataset
datasets <- list(
  Activities = activities,
  Calories = calories,
  Heartrates = heartrates,
  Intensities = intensities,
  METs = METs,
  Sleep = sleep,
  Steps = steps,
  Weight = weight
)

user_counts <- data.frame(
  Unique_Users = sapply(datasets, function(df) length(unique(df$Id)))
)

print(user_counts)
```

Heartrates and Weight Data is available for under 20 persons. Generalizing based on this data will lead to a lot of assumptions, so I won´t include these in further analysis.

```
datasets <- datasets[!(names(datasets) %in% c("Heartrates", "Weight"))]
print(names(datasets))

minutes_datasets <- datasets[!(names(datasets) %in% c("Activities"))]
```

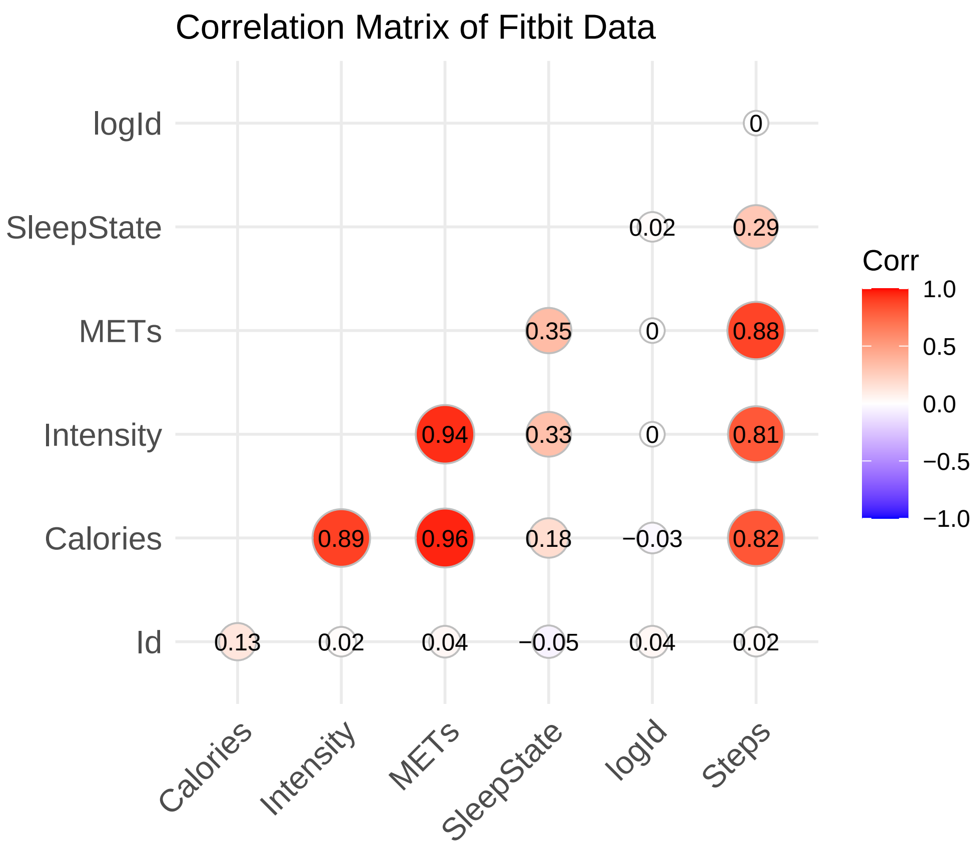
For a first impression, I want to check how the different variables are correlated to each other.

```
# Merge all datasets on Id, Date, and Time
merged_data <- Reduce(function(x, y) full_join(x, y, by = c("Id", "Date", "Time")), minutes_datasets)

# Remove non-numeric columns
numeric_data <- merged_data %>%
  select(where(is.numeric))

# Compute correlation matrix
cor_matrix <- cor(numeric_data, use = "pairwise.complete.obs")

# Plot correlation matrix
ggcorrplot(cor_matrix, lab = TRUE, lab_size = 3, method = "circle",
           type = "lower", colors = c("blue", "white", "red"),
           title = "Correlation Matrix of Fitbit Data")
```



Next, I want to look at when the FitBit Data is generated during the day. For this, I want to aggregate all days and all users and plot the mean values against the Time. This, I will do for Calories, Intensities, METs, SleepState and Steps.

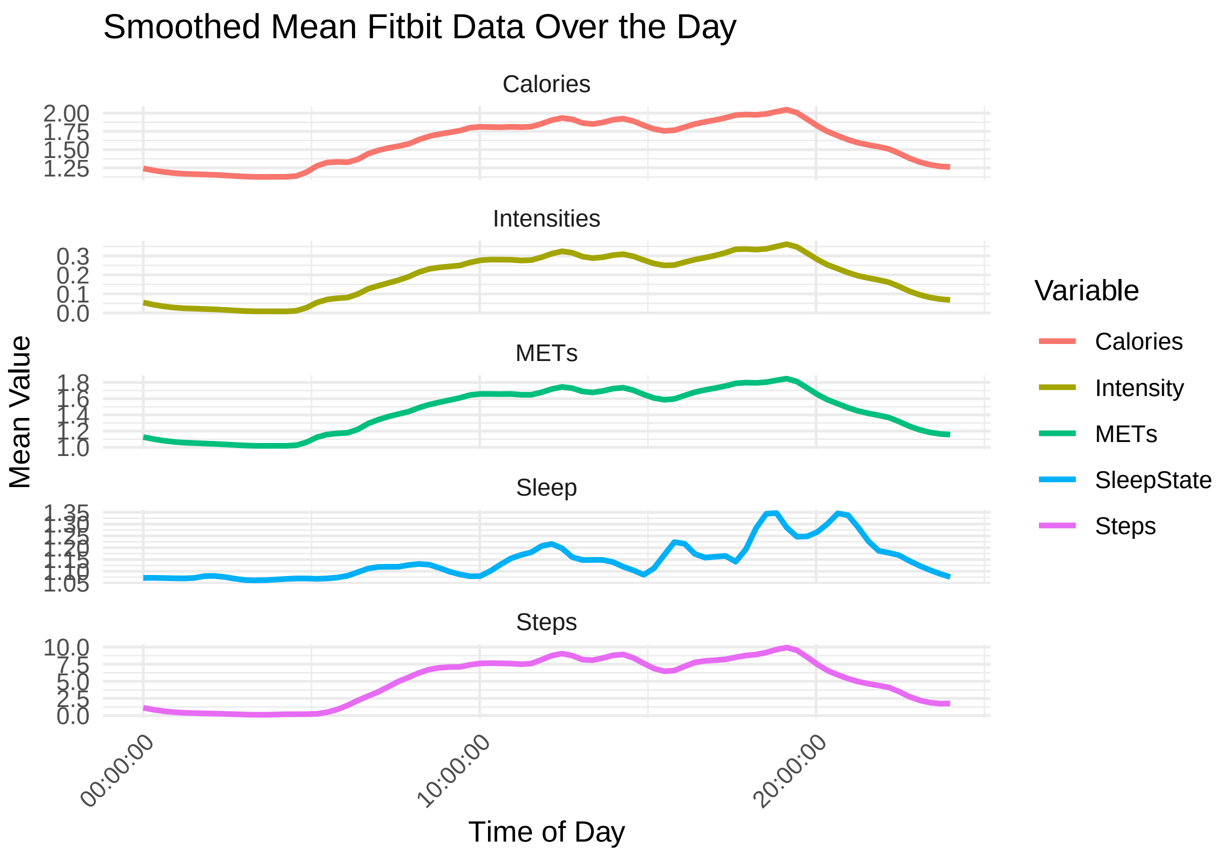
```
# Aggregate each dataset by Time
aggregated_data <- lapply(names(minutes_datasets), function(name) {
  df <- minutes_datasets[[name]]

  # Remove columns only if they exist
  columns_to_remove <- intersect(c("Id", "Date", "logId"), names(df))
  df <- df %>% select(-all_of(columns_to_remove))

  df %>%
    group_by(Time) %>%
    summarise(across(where(is.numeric), mean, na.rm = TRUE), .groups = "drop") %>%
    pivot_longer(-Time, names_to = "Variable", values_to = "Value") %>%
    mutate(Dataset = name) # Add dataset name
})

# Combine all datasets into one dataframe
plot_data <- bind_rows(aggregated_data)
```

```
ggplot(plot_data, aes(x = Time, y = Value, color = Variable)) +
  geom_smooth(method = "loess", span = 0.1, se = FALSE) +
  facet_wrap(~Dataset, scales = "free_y", nrow = 5) +
  labs(title = "Smoothed Mean Fitbit Data Over the Day",
       x = "Time of Day",
       y = "Mean Value") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



```
ggsave("Mean Data over the Day.png", plot = last_plot(), width = 12, height = 20)
```

Calories, Intensities, METs and Steps are highly connected to each other, which makes sense. The sleep during the day time is a bit more varied, than I would have expected.

```
col_means <- activities %>%
  summarise(across(where(is.numeric), mean, na.rm = TRUE))

col_means
```

The mean step count of around 7,300 steps per day is notably lower than the commonly recommended 10,000 steps per day, which is often cited as a benchmark for maintaining a healthy and active lifestyle. While this guideline is somewhat arbitrary and not necessarily backed by strong scientific evidence, studies have shown that higher step counts correlate with improved cardiovascular health and lower mortality risk.

One possible explanation for this relatively low average could be that many users have sedentary jobs or lifestyles, limiting their daily movement. Additionally, the dataset might include rest days or inactive users, which could pull the mean downward.

Potential Strategy for Encouraging More Activity: A fitness watch could implement behavioral nudges to encourage users to walk more, such as:

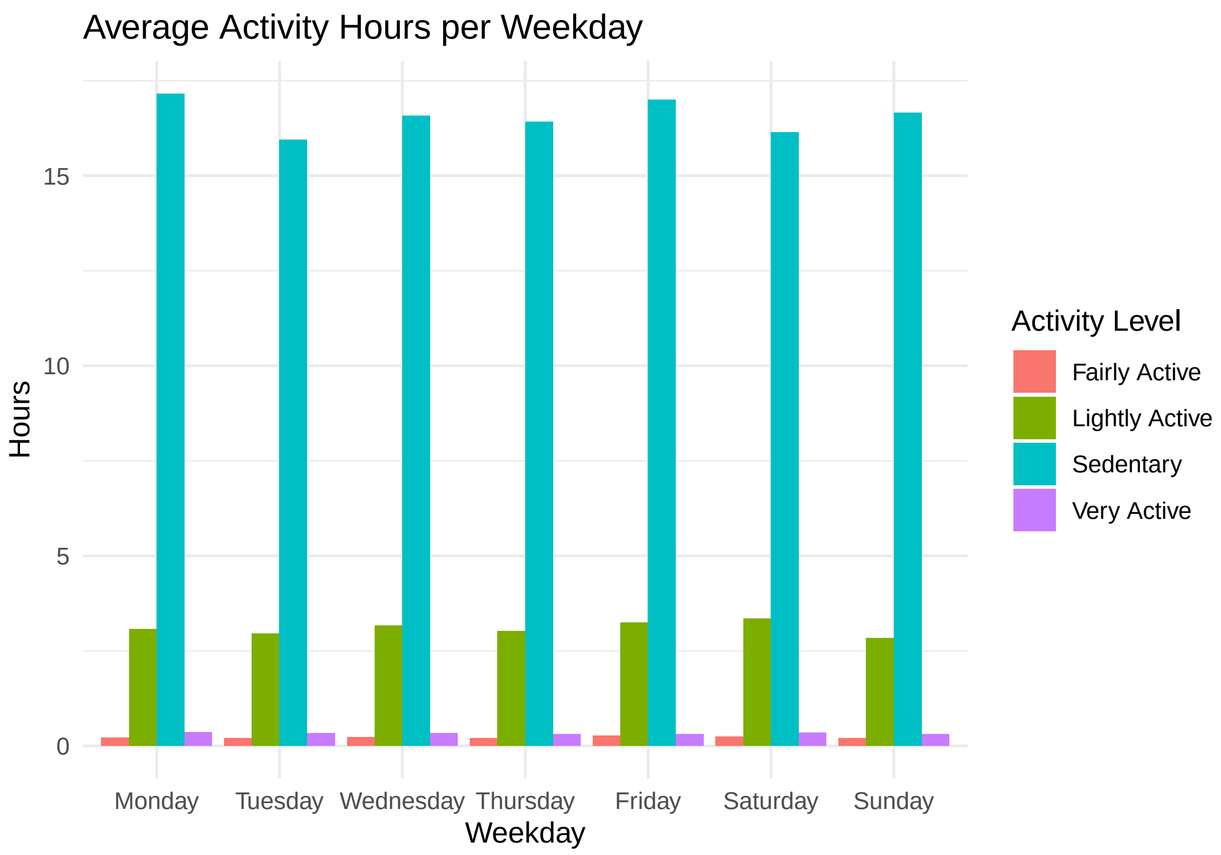
- Personalized step goals that adjust dynamically based on a user's baseline activity.
- Hourly movement reminders to prevent long sedentary periods.
- Gamification elements like badges, challenges, or step competitions with friends.
- Integration with public health recommendations, providing insights on how small changes (e.g., taking the stairs instead of the elevator) can contribute to long-term health benefits.

Next, I want to analyze how the different activity zones are spread across the different weekdays.

```r
# Convert Date column to weekdays
activities <- activities %>%
  mutate(Weekday = factor(weekdays(Date),
                          levels = c("Monday", "Tuesday", "Wednesday", "Thursday",
                                     "Friday", "Saturday", "Sunday")))  # Ensure correct order

# Compute mean values per weekday and convert to hours
activity_summary <- activities %>%
  group_by(Weekday) %>%
  summarise(across(c(VeryActiveMinutes, FairlyActiveMinutes, LightlyActiveMinutes, SedentaryMinutes),
                   ~ mean(.x, na.rm = TRUE) / 60, .names = "{.col}_Hours")) %>%
  pivot_longer(cols = -Weekday, names_to = "ActivityLevel", values_to = "Hours") %>%
  mutate(ActivityLevel = recode(ActivityLevel,
                                "VeryActiveMinutes_Hours" = "Very Active",
                                "FairlyActiveMinutes_Hours" = "Fairly Active",
                                "LightlyActiveMinutes_Hours" = "Lightly Active",
                                "SedentaryMinutes_Hours" = "Sedentary"))


ggplot(activity_summary, aes(x = Weekday, y = Hours, fill = ActivityLevel)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Average Activity Hours per Weekday",
       x = "Weekday",
       y = "Hours",
       fill = "Activity Level") +
  theme_minimal()
```



There is no significant difference between the weekdays. Sundays seem to be a little less active, than the other days. All in all, the sedentary time is very high across all days. It might be beneficial for the fitness tracker to send out reminders to move, after sitting for some time.

## Conclusion

### Summary of Key Findings

Our analysis of Fitbit activity data provides valuable insights into users' daily movement patterns. The average step count of approximately 7000 steps per day falls below the commonly recommended target of 10,000 steps, indicating room for increased physical activity. The majority of time is spent in sedentary or lightly active states, emphasizing the need for strategies to encourage more movement.

### Implications for Bellabeat

These findings suggest several opportunities for Bellabeat to enhance user engagement: - **Encouraging More Activity:** Personalized nudges, step challenges, or reminders could help users gradually increase their daily step count. - **Weekday Trends:** If activity levels show consistent dips on specific days, Bellabeat can introduce targeted interventions, such as themed workouts or motivational messages. - **Addressing Sedentary Behavior:** Implementing movement reminders or gamified active breaks could help users maintain a more active lifestyle.

### Limitations of the Analysis

While the dataset provides useful insights, several limitations should be acknowledged: - The dataset does not include demographic information, making it difficult to analyze trends based on age, gender, or other factors. - The sample may not be fully representative of the broader population. - The analysis does not consider external factors such as weather, work schedules, or individual fitness goals.

## Recommendations for Future Research

Further investigations could build on this analysis by: - Examining correlations between sleep patterns and activity levels. - Segmenting data by user demographics to uncover tailored fitness recommendations. - Analyzing long-term fitness trends to evaluate the impact of fitness tracker engagement over time.

By leveraging these insights, Bellabeat can refine its marketing strategy and develop new features to support users in leading more active and healthier lifestyles.