

# Bellabeat Wellness Device Case Study:

## Analyzing User Activity and Wellness

**Project Title:** Bellabeat Wellness Device Case Study: Analyzing User Activity and Wellness Patterns

**Introduction:** The Bellabeat Wellness Device Case Study explores user activity patterns, sleep behavior, and calorie expenditure to gain insights into wellness trends. This analysis was conducted to help Bellabeat, a wellness technology company, understand user behavior and enhance their products' engagement and impact on health. By leveraging data from Bellabeat devices, this study aims to identify actionable insights to inform product design and marketing strategies.

The study follows a structured 6-step framework, encompassing asking key questions, preparing and processing data, performing analyses, sharing findings through visualizations, and making data-driven recommendations. Each step is documented in detail to demonstrate the analytical process and outcomes.

[1. Ask](#)

[2. Prepare](#)

[3. Process](#)

[4. Analyze](#)

[5. Share](#)

[6. Act](#)

### 1. Ask

- **Objective:** Define the business task and set clear objectives for the analysis.

- **Business Task:** Analyze activity and wellness data from Bellabeat devices to understand user behavior and trends. The goal is to identify actionable insights that can guide product improvements and marketing strategies for Bellabeat's wellness products.
- **Questions:**
  - What are the daily activity patterns of Bellabeat users?
  - How do steps, calories, and sleep patterns vary across the week?
  - Are there any correlations between steps taken and calories burned?

## 2. Prepare

- **Data Sources:** [Bellabeat wellness data](#), which includes metrics such as daily steps, calories burned, activity levels, and sleep duration.
- **Setup and Data Import**

# Install necessary packages and load libraries

```
install.packages("tidyverse")
```

```
library(tidyverse)
```

```
library(lubridate)
```

# Load datasets

```
daily_activity <- read_csv("dailyActivity_all.csv")
```

```
daily_calories <- read_csv("daily_calories_aggr.csv")
```

```
daily_sleep <- read_csv("daily_total_sleep_aggr.csv")
```

```
daily_steps <- read_csv("distinct_daily_steps.csv")
```

- **Initial Data Inspection**

```
# Check column names and sample data
```

```
colnames(daily_activity)
```

```
colnames(daily_calories)
```

```
colnames(daily_sleep)
```

```
colnames(daily_steps)
```

```
head(daily_activity)
```

```
head(daily_calories)
```

```
head(daily_sleep)
```

```
head(daily_steps)
```

```
> colnames(daily_activity)
[1] "Id" "ActivityDate" "TotalSteps"
[4] "TotalDistance" "TrackerDistance" "LoggedActivitiesDistance"
[7] "VeryActiveDistance" "ModeratelyActiveDistance" "LightActiveDistance"
[10] "SedentaryActiveDistance" "VeryActiveMinutes" "FairlyActiveMinutes"
[13] "LightlyActiveMinutes" "SedentaryMinutes" "Calories"
[16] "DailySteps" "Daily_sleep" "TotalSleep"
> colnames(daily_calories)
[1] "Id" "ActivityDate" "TotalCalories"
> colnames(daily_sleep)
[1] "Id" "ActivityDate" "TotalSleep"
> colnames(daily_steps)
[1] "Id" "ActivityDate" "DailySteps"
```

```
      Id ActivityDate TotalSteps TotalDistance TrackerDistance LoggedActivitiesDist...1
      <dbl> <date>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
1 1.50e9 2016-04-12      224        0.140        0.140          0
2 1.50e9 2016-04-12    13162         8.5          8.5          0
3 1.50e9 2016-04-13    10735         6.97         6.97          0
4 1.50e9 2016-04-14    10460         6.74         6.74          0
5 1.50e9 2016-04-15     9762         6.28         6.28          0
6 1.50e9 2016-04-16    12669         8.16         8.16          0
# i abbreviated name: 1LoggedActivitiesDistance
# i 12 more variables: VeryActiveDistance <dbl>, ModeratelyActiveDistance <dbl>,
#   LightActiveDistance <dbl>, SedentaryActiveDistance <dbl>,
#   VeryActiveMinutes <dbl>, FairlyActiveMinutes <dbl>, LightlyActiveMinutes <dbl>,
#   SedentaryMinutes <dbl>, Calories <dbl>, DailySteps <dbl>, Daily_sleep <chr>,
#   TotalSleep <chr>
```

```

> head(daily_calories)
# A tibble: 6 × 3
      Id ActivityDate TotalCalories
  <dbl> <date>         <dbl>
1 1503960366 2016-04-12      2035
2 1503960366 2016-04-13      1797
3 1503960366 2016-04-14      1776
4 1503960366 2016-04-15      1745
5 1503960366 2016-04-16      1863
6 1503960366 2016-04-17      1728
> head(daily_sleep)
# A tibble: 6 × 3
      Id ActivityDate TotalSleep
  <dbl> <date>         <dbl>
1 1503960366 2016-04-12      10.9
2 1503960366 2016-04-13       6.4
3 1503960366 2016-04-15       6.87
4 1503960366 2016-04-16       5.67
5 1503960366 2016-04-17      11.7
6 1503960366 2016-04-19       5.07
> head(daily_steps)
# A tibble: 6 × 3
      id ActivityDate DailySteps
  <dbl> <chr>         <dbl>
1 1503960366 4/12/2016      13162
2 1503960366 4/13/2016      10735
3 1503960366 4/14/2016      10460
4 1503960366 4/15/2016       9762
5 1503960366 4/16/2016      12669
6 1503960366 4/17/2016       9705

```

- **Data Preparation: Rename Columns**

```
library(dplyr)
```

```
# Convert column names to CamelCase for consistency
```

```
daily_calories <- daily_calories %>% rename(Id = id , TotalCalories = total_calories)
```

```
daily_sleep <- daily_sleep %>% rename(Id = id , TotalSleep = total_sleep)
```

```
daily_steps <- daily_steps %>% rename(Id = id)
```

- **Data Preparation: Date Format Conversion**

```
# Standardize date formats
```

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

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

```
daily_sleep$ActivityDate <- ymd(daily_sleep$ActivityDate)
```

```
daily_calories$ActivityDate <- ymd(daily_calories$ActivityDate)
```

### **3. Process**

- **Duplicate Check and Handling**

```
# Check for duplicates
```

```
duplicates <- daily_activity %>%
```

```
  group_by(Id, ActivityDate) %>%
```

```
  filter(n() > 1)
```

```
print(duplicates)
```

```
View(duplicates)
```

- **Missing Values Handling**

```
# Replace N/A values in DailySleep with 0

daily_activity <- daily_activity %>%

mutate(TotalSleep = ifelse(Daily_sleep == "#N/A", 0 , Daily_sleep))
```

- **Data Aggregation**

```
# Summarize data by Id and ActivityDate

daily_activity_cleaned <- daily_activity %>%

group_by(Id, ActivityDate) %>%

summarise(

  TotalSteps = sum(TotalSteps, na.rm = TRUE),

  TotalDistance = sum(TotalDistance, na.rm = TRUE),

  TrackerDistance = sum(TrackerDistance, na.rm = TRUE),

  VeryActiveDistance = sum(VeryActiveDistance, na.rm = TRUE),

  ModeratelyActiveDistance = sum(ModeratelyActiveDistance, na.rm = TRUE),

  LightActiveDistance = sum(LightActiveDistance, na.rm = TRUE),

  VeryActiveMinutes = max(VeryActiveMinutes, na.rm = TRUE),

  FairlyActiveMinutes = max(FairlyActiveMinutes, na.rm = TRUE),

  LightlyActiveMinutes = max(LightlyActiveMinutes, na.rm = TRUE),

  SedentaryMinutes = max(SedentaryMinutes, na.rm = TRUE),

  Calories = max(Calories, na.rm = TRUE),

  DailySteps = max(DailySteps, na.rm = TRUE),

  DailySleep = max(TotalSleep, na.rm = TRUE),

  .groups = "drop"
```

)

```
# View the cleaned data
```

```
View(daily_activity_cleaned)
```

```
head(daily_activity_cleaned)
```

```
colnames(daily_activity_cleaned)
```

	Id	ActivityDate	TotalSteps	TotalDistance	TrackerDistance	VeryActiveDistance	ModeratelyActiveDistance	LightActiveDistance	VeryActiveMinutes	FairlyActiveMinutes	LightlyActiveMinutes
1	1503960366	2016-04-12	13386	8.64	8.64	1.88	0.55	6.19	25	13	
2	1503960366	2016-04-13	10735	6.97	6.97	1.57	0.69	4.71	21	19	
3	1503960366	2016-04-14	10460	6.74	6.74	2.44	0.40	3.91	30	11	
4	1503960366	2016-04-15	9762	6.28	6.28	2.14	1.26	2.83	29	34	
5	1503960366	2016-04-16	12669	8.16	8.16	2.71	0.41	5.04	36	10	
6	1503960366	2016-04-17	9705	6.48	6.48	3.19	0.78	2.51	38	20	
7	1503960366	2016-04-18	13019	8.59	8.59	3.25	0.64	4.71	42	16	
8	1503960366	2016-04-19	15506	9.88	9.88	3.53	1.32	5.03	50	31	
9	1503960366	2016-04-20	10544	6.68	6.68	1.96	0.48	4.24	28	12	
10	1503960366	2016-04-21	9819	6.34	6.34	1.34	0.35	4.65	19	8	
11	1503960366	2016-04-22	12764	8.13	8.13	4.76	1.12	2.24	66	27	
12	1503960366	2016-04-23	14371	9.04	9.04	2.81	0.87	5.36	41	21	
13	1503960366	2016-04-24	10039	6.41	6.41	2.92	0.21	3.28	39	5	
14	1503960366	2016-04-25	15355	9.80	9.80	5.29	0.57	3.94	73	14	
15	1503960366	2016-04-26	13755	8.79	8.79	2.33	0.92	5.54	31	23	
16	1503960366	2016-04-27	18134	12.21	12.21	6.40	0.41	5.41	78	11	
17	1503960366	2016-04-28	13154	8.53	8.53	3.54	1.16	3.79	48	28	
18	1503960366	2016-04-29	11181	7.15	7.15	1.06	0.50	5.58	16	12	
19	1503960366	2016-04-30	14673	9.25	9.25	3.56	1.42	4.27	52	34	
20	1503960366	2016-05-01	10602	6.81	6.81	2.29	1.60	2.92	33	35	

Showing 1 to 20 of 857 entries, 16 total columns

- **Check for Missing Values**

```
# Check for missing values in each dataset
```

```
sapply(daily_activity, function(x) sum(is.na(x)))
```

```
sapply(daily_calories, function(x) sum(is.na(x)))
```

```
sapply(daily_sleep, function(x) sum(is.na(x)))
```

```
sapply(daily_steps, function(x) sum(is.na(x)))
```

```

> sapply(daily_activity, function(x) sum(is.na(x)))
      Id ActivityDate TrackerDistance LoggedActivitiesDistance
      0              0              0
      0              0
      VeryActiveDistance ModeratelyActiveDistance
VeryActiveMinutes      FairlyActiveMinutes      LightActiveDistance SedentaryActiveDistance
      0              0              0              0
      0              0
      LightlyActiveMinutes      SedentaryMinutes
Daily_sleep      TotalSleep
      0              0
      0              0
> sapply(daily_calories, function(x) sum(is.na(x)))
      Id ActivityDate TotalCalories
      0              0              0
> sapply(daily_sleep, function(x) sum(is.na(x)))
      Id ActivityDate TotalSleep
      0              0              0
> sapply(daily_steps, function(x) sum(is.na(x)))
      Id ActivityDate DailySteps
      0              0              0
> |

```

## 4. Analyze

### ▪ Summary Statistics

# Calculate summary statistics

```














summary_data <- daily_activity_cleaned %>%
  summarise(
    min_steps = min(DailySteps, na.rm = TRUE),
    max_steps = max(DailySteps, na.rm = TRUE),
    avg_steps = mean(DailySteps, na.rm = TRUE),
    min_calories = min(Calories, na.rm = TRUE),
    max_calories = max(Calories, na.rm = TRUE),
    avg_calories = mean(Calories, na.rm = TRUE),
    min_sleep = min(DailySleep , na.rm = TRUE),
    max_sleep = max(DailySleep , na.rm = TRUE),
    avg_sleep = mean(DailySleep , na.rm = TRUE)

```



)

[View\(summary\\_data\)](#)

	 min_steps 	 max_steps 	 avg_steps 	 min_calories 	 max_calories 	 avg_calories 	 min_sleep	max_sleep	avg_sleep
1	17	36019	8377.525	52	4900	2366.592	0	11.67	3.233652

- **Step and Calorie Distribution**

```
# Create step and calorie ranges
```

```
step_distribution <- daily_activity_cleaned %>%
```

```
  mutate(
```

```
    step_range = case_when(
```

```
      DailySteps < 5000 ~ "(1) <5000",
```

```
      DailySteps >= 5000 & DailySteps <= 10000 ~ "(2) 5000 - 10000",
```

```
      DailySteps >= 10001 & DailySteps <= 15000 ~ "(3) 10001 - 15000",
```

```
      DailySteps > 15000 ~ "(4) >15000"
```

```
    )
```

```
  ) %>%
```

```
  group_by(step_range) %>%
```

```
  summarise(count_days = n()) %>%
```

```
  arrange(step_range)
```

```
calorie_distribution <- daily_activity_cleaned %>%
```

```
  mutate(
```

```
    calorie_range = case_when(
```

```
      Calories < 1000 ~ "(1) <1000",
```

```

    Calories >= 1000 & Calories <= 2000 ~ "(2) 1000-2000",
    Calories > 2000 ~ "(3) >2000"
  )
) %>%

group_by(calorie_range) %>%
summarise(count_days = n()) %>%
arrange(calorie_range)

# Plot the step distribution

library(ggplot2)

ggplot(step_distribution, aes(x = step_range, y = count_days, fill = step_range)) +
  geom_bar(stat = "identity") +
  labs(title = "Distribution of Daily Steps",
       x = "Step Range",
       y = "Count of Days") +
  theme_minimal() +
  theme(legend.position = "none")

# Plot the calorie distribution

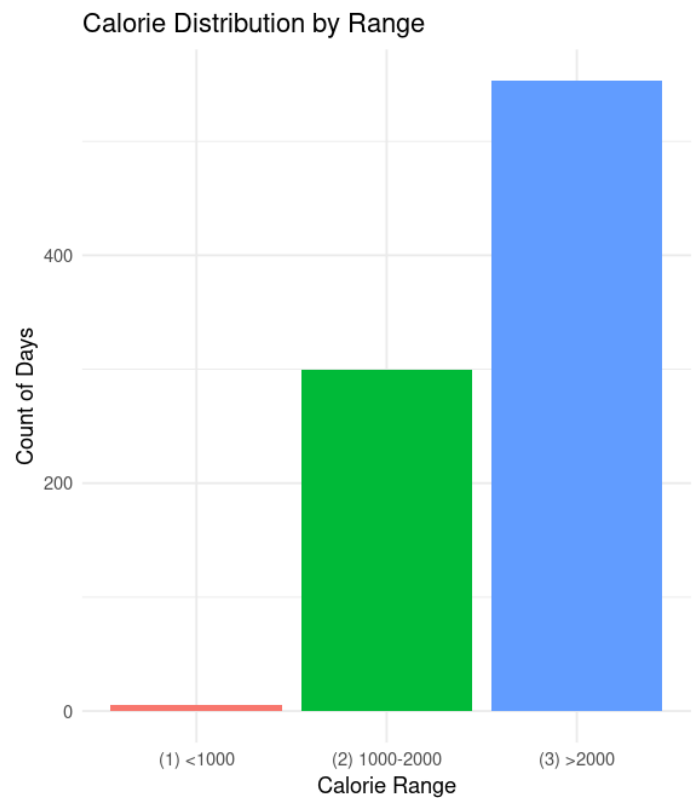
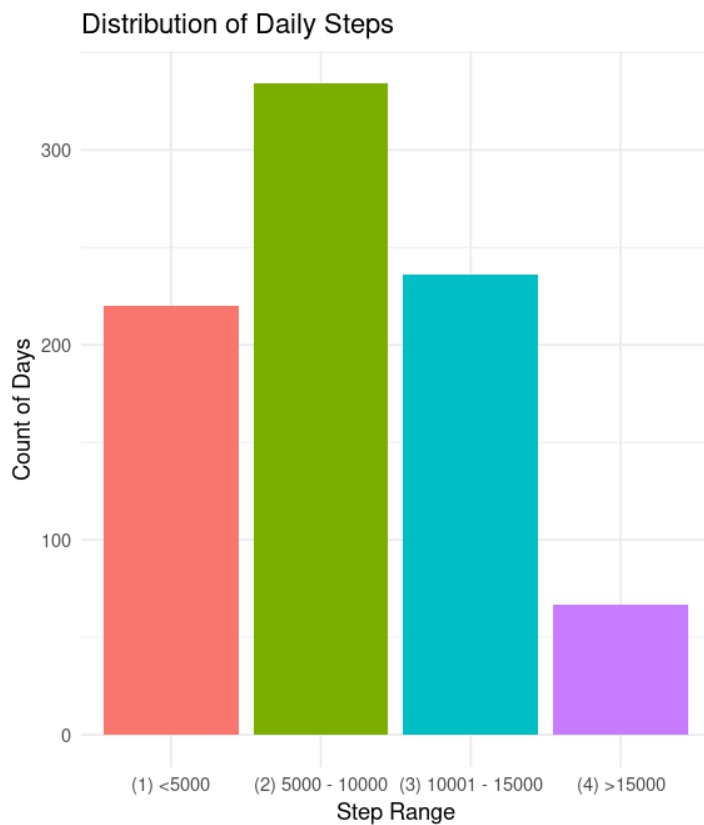
ggplot(calorie_distribution, aes(x = calorie_range, y = count_days, fill = calorie_range)) +
  geom_bar(stat = "identity") +
  labs(
    title = "Calorie Distribution by Range",
    x = "Calorie Range",

```

```

y = "Count of Days"
) +
theme_minimal() +
theme(legend.position = "none")

```



- **Daily Summary Over Time**

```

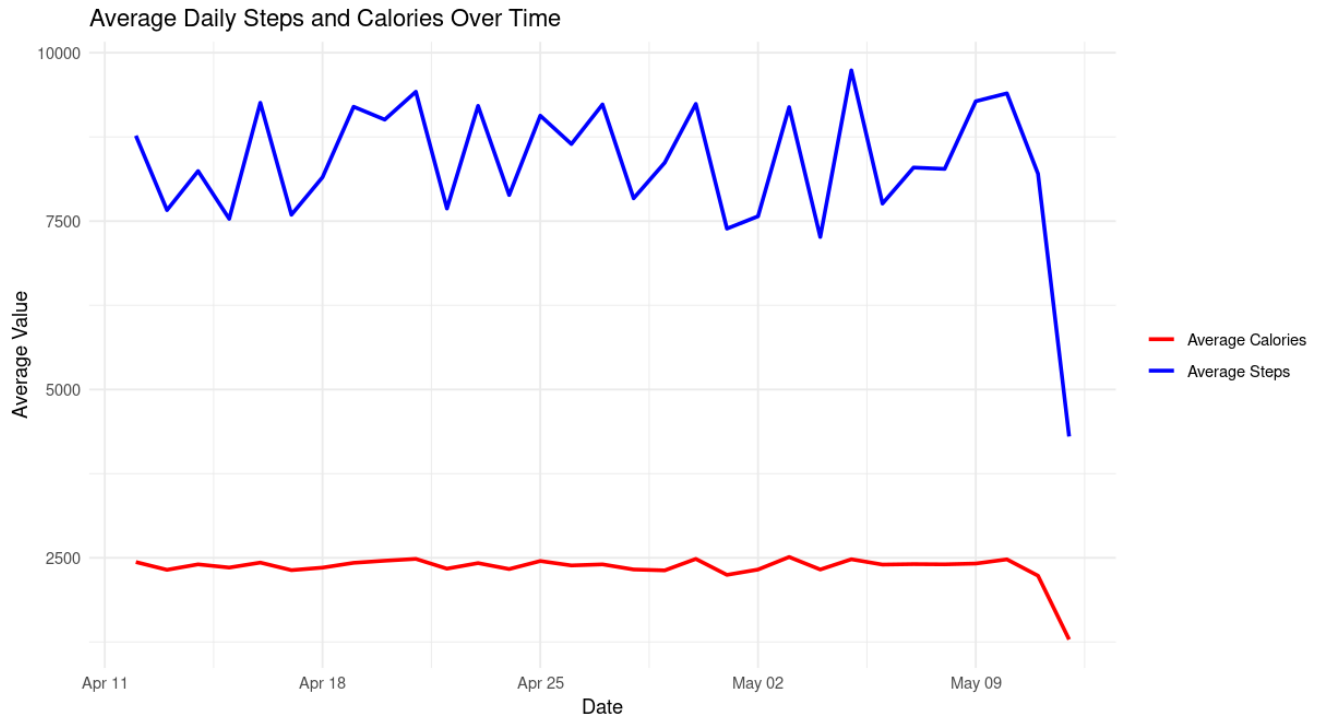
# Average daily steps and calories over time
daily_summary <- daily_activity_cleaned %>%
  group_by(ActivityDate) %>%
  summarise(
    avg_steps = mean(DailySteps, na.rm = TRUE),

```

```
avg_calories = round(mean(Calories,na.rm = TRUE),2)) %>%  
arrange(ActivityDate)
```

```
# Plot daily_summary data
```

```
ggplot(daily_summary, aes(x = ActivityDate)) +  
  geom_line(aes(y = avg_steps, color = "Average Steps"), size = 1) +  
  geom_line(aes(y = avg_calories, color = "Average Calories"), size = 1) +  
  labs(  
    title = "Average Daily Steps and Calories Over Time",  
    x = "Date",  
    y = "Average Value"  
  ) +  
  scale_color_manual(values = c("Average Steps" = "blue", "Average Calories" = "red")) +  
  theme_minimal() +  
  theme(legend.title = element_blank())
```



## ▪ Weekly Trends

```
# Add day of week column
```

```
daily_activity_cleaned <- daily_activity_cleaned %>%
```

```
  mutate(DayOfWeek = wday(ActivityDate, label = TRUE))
```

```
# Weekly summary
```

```
weekly_summary <- daily_activity_cleaned %>%
```

```
  group_by(DayOfWeek) %>%
```

```
  summarise(
```

```
    avg_steps = mean(DailySteps, na.rm = TRUE),
```

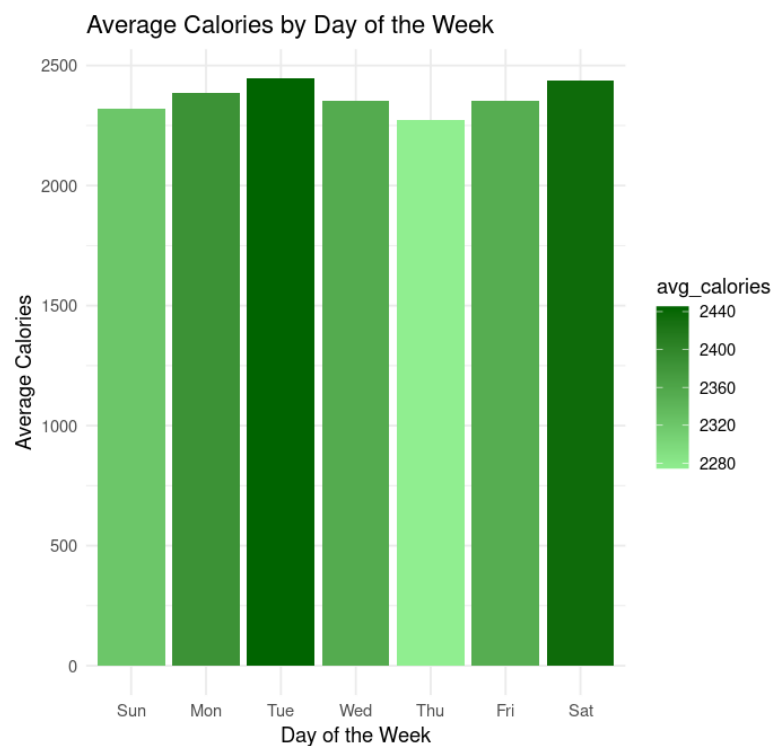
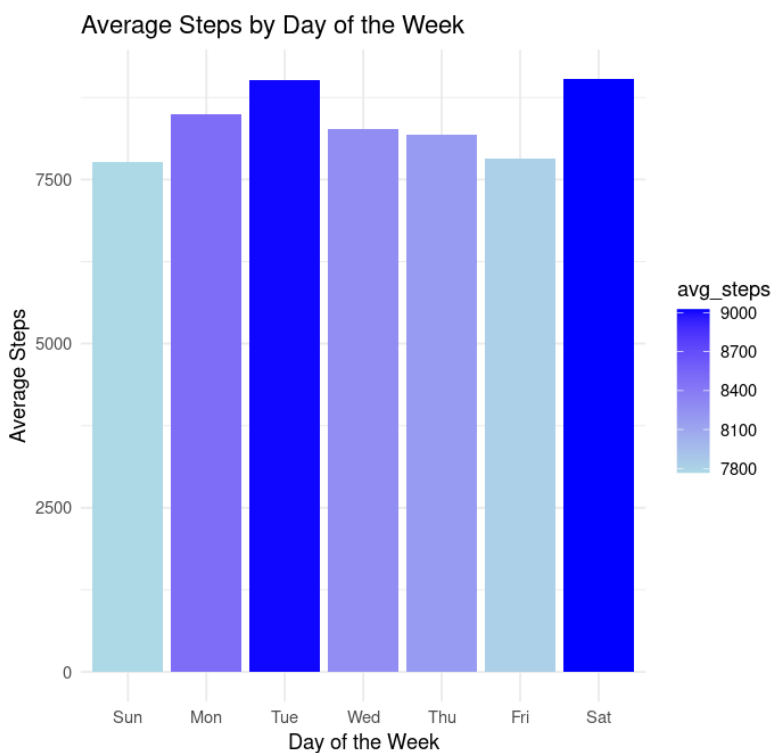
```
    avg_calories = mean(Calories, na.rm = TRUE)
```

```
)
```

```
# Plot for weekly trends
```

```
ggplot(weekly_summary, aes(x = DayOfWeek, y = avg_steps, fill = avg_steps)) +  
  geom_bar(stat = "identity") +  
  scale_fill_gradient(low = "lightblue", high = "blue") +  
  labs(title = "Average Steps by Day of the Week", x = "Day of the Week", y = "Average Steps") +  
  theme_minimal()
```

```
ggplot(weekly_summary, aes(x = DayOfWeek, y = avg_calories, fill = avg_calories)) +  
  geom_bar(stat = "identity") +  
  scale_fill_gradient(low = "lightgreen", high = "darkgreen") +  
  labs(title = "Average Calories by Day of the Week", x = "Day of the Week", y = "Average  
Calories") +  
  theme_minimal()
```



- **Weekly Patterns in Activity and Movement**

```
# Summarize the data by DayOfWeek
```

```
weekly_activity_summary <- daily_activity_cleaned %>%
```

```
  group_by(DayOfWeek) %>%
```

```
  summarise(
```

```
    avg_very_active = mean(VeryActiveMinutes, na.rm = TRUE),
```

```
    avg_light_active = mean(LightlyActiveMinutes, na.rm = TRUE),
```

```
    avg_sedentary = mean(SedentaryMinutes, na.rm = TRUE),
```

```
    avg_very_active_distance = mean(VeryActiveDistance, na.rm = TRUE),
```

```
    avg_moderately_active_distance = mean(ModeratelyActiveDistance, na.rm = TRUE),
```

```
    avg_light_active_distance = mean(LightActiveDistance, na.rm = TRUE)
```

```
  )
```

```
# Reshape the data into long format for plotting
```

```
weekly_activity_long <- weekly_activity_summary %>%
```

```
  pivot_longer(cols = starts_with("avg_"),
```

```
    names_to = "ActivityType",
```

```
    values_to = "Minutes")
```

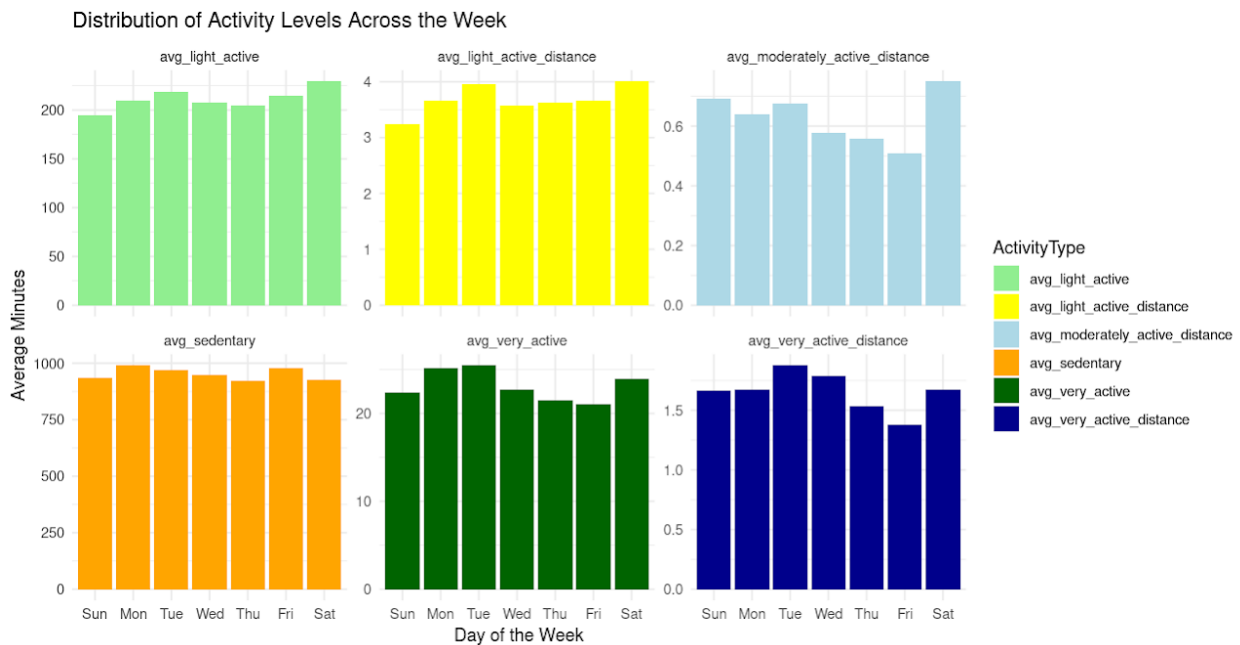
```
# Plot weekly activities
```

```
ggplot(weekly_activity_long, aes(x = DayOfWeek, y = Minutes, fill = ActivityType)) +
```

```
  geom_col(position = "dodge") +
```

```
  facet_wrap(~ ActivityType, scales = "free_y") +
```

```
labs(title = "Distribution of Activity Levels Across the Week",
     x = "Day of the Week",
     y = "Average Minutes") +
scale_fill_manual(values = c("avg_very_active" = "darkgreen", "avg_light_active" = "lightgreen", "avg_sedentary" = "orange", "avg_very_active_distance" = "darkblue", "avg_moderately_active_distance" = "lightblue", "avg_light_active_distance" = "yellow")) +
theme_minimal()
```



## ▪ Outlier Detection

```
# Detect outliers
```

```
mean_steps <- mean(daily_activity_cleaned$DailySteps, na.rm = TRUE)
```

```
sd_steps <- sd(daily_activity_cleaned$DailySteps, na.rm = TRUE)
```



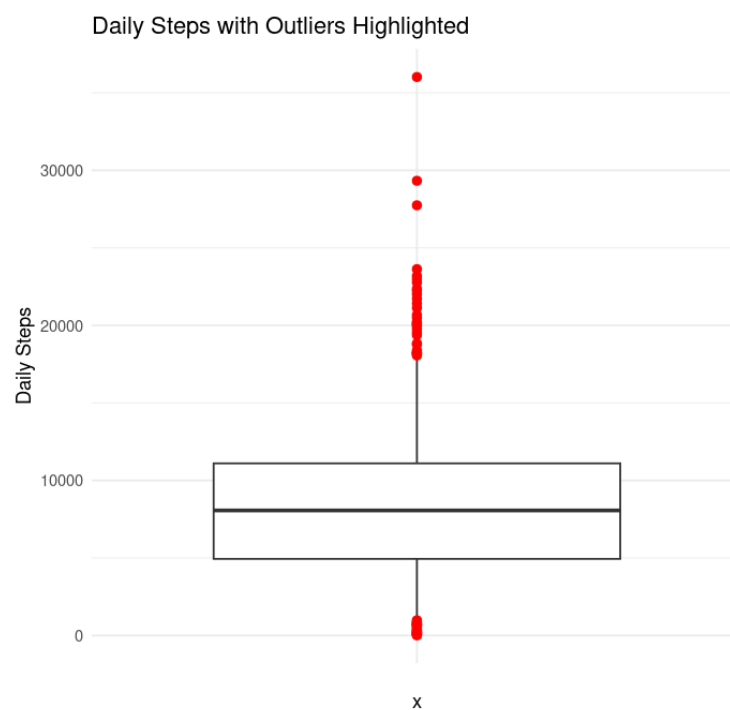
```

outliers <- daily_activity_cleaned %>%
  filter(DailySteps > (mean_steps + 2 * sd_steps) | DailySteps < 1000)

outliers <- outliers %>%
  mutate(avg_daily_steps = mean_steps,
         standard_deviation = sd_steps)

# Plot outliers
ggplot(daily_activity_cleaned, aes(x = "", y = DailySteps)) +
  geom_boxplot(outlier.shape = NA) +
  geom_point(data = outliers, aes(y = DailySteps), color = "red", size = 2) +
  labs(title = "Daily Steps with Outliers Highlighted", y = "Daily Steps") +
  theme_minimal()

```

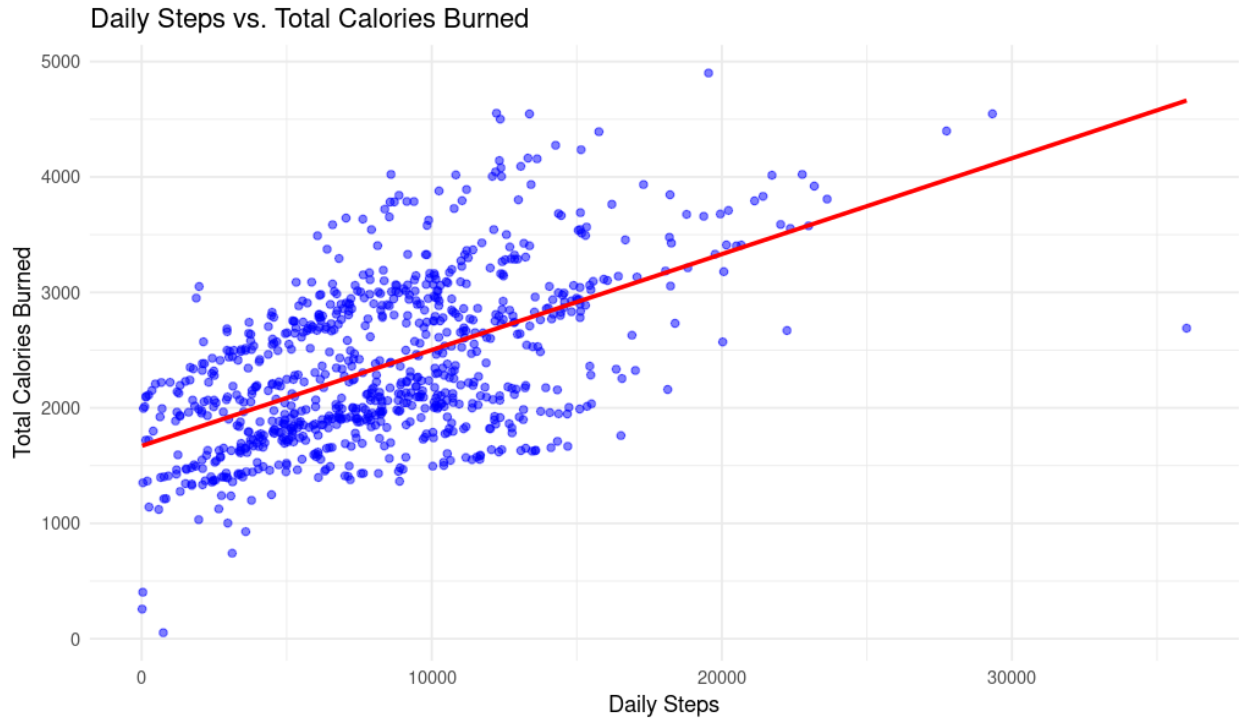


- **Correlation Analysis**

```
# Correlation
```

```
correlation <- cor(daily_activity_cleaned$DailySteps, daily_activity_cleaned$Calories, use  
= 'complete.obs')
```

```
ggplot(daily_activity_cleaned, aes(x = DailySteps, y = Calories)) +  
  geom_point(color = 'blue', alpha = 0.5) +  
  geom_smooth(method = 'lm', se = FALSE, color = 'red') +  
  labs(  
    title = 'Daily Steps vs. Total Calories Burned',  
    x = 'Daily Steps',  
    y = 'Total Calories Burned'  
  ) +  
  theme_minimal()
```



## 5. Share

### Key Analysis Steps and Findings:

#### 1. Summary Statistics:

- Calculated key statistics, including minimum, maximum, and average values for daily steps, calories burned, and sleep duration.
- These summary metrics help provide a quick overview of the data and set a foundation for deeper analysis.

#### 2. Activity and Calorie Distribution:

- Categorized daily steps into ranges (<5000, 5000-10000, 10001-15000, >15000) and visualized the distribution to understand user activity levels.
- A similar approach was used for calorie ranges, providing insight into users' daily energy expenditure.

#### 3. Trends Over Time:

- Analyzed daily steps and calories over time to observe any patterns or fluctuations in activity.
- Created visualizations to show weekly trends, which indicated patterns in user activity that vary by day of the week.

#### **4. Correlation Analysis:**

- Calculated the correlation between daily steps and calories to assess the relationship between physical activity and calorie expenditure.
- A moderate positive correlation was found, suggesting that as steps increase, calorie burn generally increases as well.

#### **5. Outlier Detection:**

- Identified days with unusually high or low activity levels using statistical thresholds.
- Highlighted these outliers in visualizations to help Bellabeat identify opportunities for user engagement.

### **6. Act**

#### **Recommendations for Bellabeat**

Based on the insights gathered from the analysis, the following recommendations can help Bellabeat enhance user engagement and promote healthier lifestyles:

##### **1. Introduce Activity Challenges and Reminders on Weekends:**

Analysis showed that users tend to be more sedentary during the weekends. Bellabeat can introduce weekend challenges or send reminders to encourage users to increase their activity levels.

##### **2. Promote Notifications for Low Activity or Calorie Burn Days**

Bellabeat could implement notifications for users who have low daily steps or calorie burn, encouraging them to stay active. Personalized messages could remind users of their health

goals and suggest achievable activities to help them stay on track. This would create a more supportive experience and help users develop consistent wellness habits.

## Next Steps

To further refine Bellabeat's offerings, consider these next steps:

- **User Segmentation Analysis:** Identify distinct user segments based on activity, calorie burn, and sleep patterns. This will enable Bellabeat to offer more personalized recommendations and tailor marketing strategies to different user needs.
- **Enhanced Sleep Insights:** Explore the relationship between sleep patterns and activity levels to offer advice on sleep quality improvements, aligning with holistic wellness goals.

## Tools and Skills Demonstrated

**R Programming:** Used extensively for data cleaning, transformation, and statistical analysis. Leveraged libraries such as dplyr and tidyverse for efficient data manipulation, and ggplot2 for visualization.

**Data Cleaning:** Ensured data consistency and accuracy by handling missing values, removing duplicates, and converting data types. Utilized conditional transformations to prepare datasets for analysis.

**Data Analysis:** Conducted Exploratory Data Analysis (EDA) to examine distributions, detect outliers, and calculate correlations between key metrics (daily steps and calorie burn).

**Data Visualization:** Created informative and visually engaging plots using ggplot2 to illustrate weekly activity patterns, distribution of steps and calories, and correlations. Faceted and gradient-filled charts enabled clear communication of insights.

## Conclusion

This analysis of Bellabeat's wellness data highlights the potential for enhancing user engagement through data-driven insights. By identifying trends in user activity, sleep, and calorie expenditure, Bellabeat can introduce personalized features that align with users' wellness goals.