BellaBeat Case Study

Yanxi

2025-05-19

Statement of Business Task

Analyze FitBit Fitness Tracker Data to identify trends in smart device usage that can inform BellaBeat's marketing strategy and drive market growth.

Loading Packages

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

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
library(dplyr)
```

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

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

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

```
library(tidyr)
library(readr)
library(ggplot2)
library(lubridate)
```

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

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

Importing Datasets

```
setwd("/cloud/project/Fitabase Data 4.12.16-5.12.16")
sleep <- read.csv("sleepDay_merged.csv")
activity <- read.csv("dailyActivity_merged.csv")
hourly_steps <- read.csv("hourlySteps_merged.csv")
```

Inspecting and Cleaning Sleep Dataset

```
# Glance at the dataset
head(sleep)
```

```
##           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(sleep)
```

```
## '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 ...
```

```
glimpse(sleep)
```

```
## Rows: 413
## Columns: 5
## $ Id                 <dbl> 1503960366, 1503960366, 1503960366, 1503960366, 150~
## $ SleepDay           <chr> "4/12/2016 12:00:00 AM", "4/13/2016 12:00:00 AM", "~
## $ TotalSleepRecords  <int> 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ TotalMinutesAsleep <int> 327, 384, 412, 340, 700, 304, 360, 325, 361, 430, 2~
## $ TotalTimeInBed     <int> 346, 407, 442, 367, 712, 320, 377, 364, 384, 449, 3~
```

```
colnames(sleep)
```

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

```
sleep %>%
  select(TotalSleepRecords,
         TotalMinutesAsleep,
         TotalTimeInBed) %>%
  summary()
```

```
##  TotalSleepRecords TotalMinutesAsleep TotalTimeInBed
##  Min.   :1.000     Min.   : 58.0      Min.   : 61.0
##  1st Qu.:1.000     1st Qu.:361.0      1st Qu.:403.0
##  Median :1.000     Median :433.0      Median :463.0
##  Mean   :1.119     Mean   :419.5      Mean   :458.6
##  3rd Qu.:1.000     3rd Qu.:490.0      3rd Qu.:526.0
##  Max.   :3.000     Max.   :796.0      Max.   :961.0
```

```r
n_distinct(sleep$Id)
```

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

Now I had a general idea about this dataset - it tracks partcipants' number of sleep records (normally 1 but more if the individual takes naps during daytime), length of sleep and length of time in bed over a period of time, so each participant has multiple entries of record. There are records for 24 distinct users across different days.

```r
# Check for duplicates
duplicates <- sleep[duplicated(sleep), ]
sleep <- sleep[!duplicated(sleep), ]
```

I checked if there were any duplicates in this dataset. It turned out there were three duplicated records. I removed the duplicated records and updated the dataset.

```r
# Check for missing values
any_na <- any(is.na(sleep))
print(paste("Are there any missing values?", any_na))
```

```
## [1] "Are there any missing values? FALSE"
```

```r
na_by_column <- colSums(is.na(sleep))
print("Missing values per column:")
```

```
## [1] "Missing values per column:"
```

```r
print(na_by_column)
```

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

I tried two ways to detect missing values: one checks the dataset as a whole for missing values, the other checks by columns. is.na() function returns TRUE/FALSE for missing values. There were no missing values, and I was ready to proceed.

```r
# Convert to date format for tidyness
sleep$SleepDay <- as.Date(sleep$SleepDay, format = "%m/%d/%Y")
print(head(sleep$SleepDay))
```

```
## [1] "2016-04-12" "2016-04-13" "2016-04-15" "2016-04-16" "2016-04-17"
## [6] "2016-04-19"
```

```r
class(sleep$SleepDay)
```

```
## [1] "Date"
```

The SleepDay data was recorded in strings and involved unnecessary time component, so I converted the data to date format. print() and class() to double check if the conversion was successful.

```r
# Detect (and Removing) Outliers
std_dev <- sd(sleep$TotalTimeInBed)
mean <- mean(sleep$TotalTimeInBed)
upper_bound <- mean + 3 * std_dev
lower_bound <- mean - 3 * std_dev
outliers <- sleep %>%
  filter(TotalTimeInBed > upper_bound | TotalTimeInBed < lower_bound)
print(outliers)
```

```
##            Id   SleepDay TotalSleepRecords TotalMinutesAsleep TotalTimeInBed
## 1  1644430081 2016-05-02                 1                796            961
## 2  1844505072 2016-04-15                 1                644            961
## 3  1844505072 2016-04-30                 1                722            961
## 4  1844505072 2016-05-01                 1                590            961
## 5  2320127002 2016-04-23                 1                 61             69
## 6  4319703577 2016-04-21                 1                 59             65
## 7  4388161847 2016-05-09                 1                 62             65
## 8  5553957443 2016-04-30                 2                775            843
## 9  7007744171 2016-05-01                 1                 58             61
## 10 8053475328 2016-05-07                 1                 74             75
```

Here, an outlier was defined as having a value below or over 3 standard deviations away from the mean. I first computed the mean and the standard deviation, and then calculated the lower and upper bounds. Values outside this domain were classified as an outlier.

10 outliers were detected under this definition, but I took a closer look at the data and found that these observations still display a positive correlation between TotalMinutesAsleep and TotalTimeInBed, which is a very relevant correlation. So I decided to keep these datapoints.

Now this data set is fully prepared for further analysis. I then performed similar procedures with other datasets.

Inspecting and Cleaning Activity Dataset

```r
# Glance at the dataset
head(activity)
```

```
##           Id ActivityDate TotalSteps TotalDistance TrackerDistance
## 1 1503960366    4/12/2016      13162          8.50            8.50
## 2 1503960366    4/13/2016      10735          6.97            6.97
## 3 1503960366    4/14/2016      10460          6.74            6.74
## 4 1503960366    4/15/2016       9762          6.28            6.28
## 5 1503960366    4/16/2016      12669          8.16            8.16
## 6 1503960366    4/17/2016       9705          6.48            6.48
##   LoggedActivitiesDistance VeryActiveDistance ModeratelyActiveDistance
## 1                        0               1.88                     0.55
## 2                        0               1.57                     0.69
## 3                        0               2.44                     0.40
## 4                        0               2.14                     1.26
## 5                        0               2.71                     0.41
## 6                        0               3.19                     0.78
##   LightActiveDistance SedentaryActiveDistance VeryActiveMinutes
## 1                6.06                       0                25
## 2                4.71                       0                21
## 3                3.91                       0                30
## 4                2.83                       0                29
## 5                5.04                       0                36
## 6                2.51                       0                38
##   FairlyActiveMinutes LightlyActiveMinutes SedentaryMinutes Calories
## 1                  13                  328              728     1985
## 2                  19                  217              776     1797
## 3                  11                  181             1218     1776
## 4                  34                  209              726     1745
## 5                  10                  221              773     1863
```

```
## 6                  20                   164              539     1728
```

```r
glimpse(activity)
```

```
## Rows: 940
## Columns: 15
## $ Id                       <dbl> 1503960366, 1503960366, 1503960366, 150396036~
## $ ActivityDate             <chr> "4/12/2016", "4/13/2016", "4/14/2016", "4/15/~
## $ TotalSteps               <int> 13162, 10735, 10460, 9762, 12669, 9705, 13019~
## $ TotalDistance            <dbl> 8.50, 6.97, 6.74, 6.28, 8.16, 6.48, 8.59, 9.8~
## $ TrackerDistance          <dbl> 8.50, 6.97, 6.74, 6.28, 8.16, 6.48, 8.59, 9.8~
## $ LoggedActivitiesDistance <dbl> 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, 3.5~
## $ ModeratelyActiveDistance <dbl> 0.55, 0.69, 0.40, 1.26, 0.41, 0.78, 0.64, 1.3~
## $ LightActiveDistance      <dbl> 6.06, 4.71, 3.91, 2.83, 5.04, 2.51, 4.71, 5.0~
## $ SedentaryActiveDistance  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ VeryActiveMinutes        <int> 25, 21, 30, 29, 36, 38, 42, 50, 28, 19, 66, 4~
## $ FairlyActiveMinutes      <int> 13, 19, 11, 34, 10, 20, 16, 31, 12, 8, 27, 21~
## $ LightlyActiveMinutes     <int> 328, 217, 181, 209, 221, 164, 233, 264, 205, ~
## $ SedentaryMinutes         <int> 728, 776, 1218, 726, 773, 539, 1149, 775, 818~
## $ Calories                 <int> 1985, 1797, 1776, 1745, 1863, 1728, 1921, 203~
```

```r
str(activity)
```

```
## 'data.frame':    940 obs. of  15 variables:
##  $ Id                      : num  1.5e+09 1.5e+09 1.5e+09 1.5e+09 1.5e+09 ...
##  $ ActivityDate            : chr  "4/12/2016" "4/13/2016" "4/14/2016" "4/15/2016" ...
##  $ TotalSteps              : int  13162 10735 10460 9762 12669 9705 13019 15506 10544 9819 ...
##  $ TotalDistance           : num  8.5 6.97 6.74 6.28 8.16 ...
##  $ TrackerDistance         : num  8.5 6.97 6.74 6.28 8.16 ...
##  $ LoggedActivitiesDistance: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ VeryActiveDistance      : num  1.88 1.57 2.44 2.14 2.71 ...
##  $ ModeratelyActiveDistance: num  0.55 0.69 0.4 1.26 0.41 ...
##  $ LightActiveDistance     : num  6.06 4.71 3.91 2.83 5.04 ...
##  $ SedentaryActiveDistance : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ VeryActiveMinutes       : int  25 21 30 29 36 38 42 50 28 19 ...
##  $ FairlyActiveMinutes     : int  13 19 11 34 10 20 16 31 12 8 ...
##  $ LightlyActiveMinutes    : int  328 217 181 209 221 164 233 264 205 211 ...
##  $ SedentaryMinutes        : int  728 776 1218 726 773 539 1149 775 818 838 ...
##  $ Calories                : int  1985 1797 1776 1745 1863 1728 1921 2035 1786 1775 ...
```

```r
colnames(activity)
```

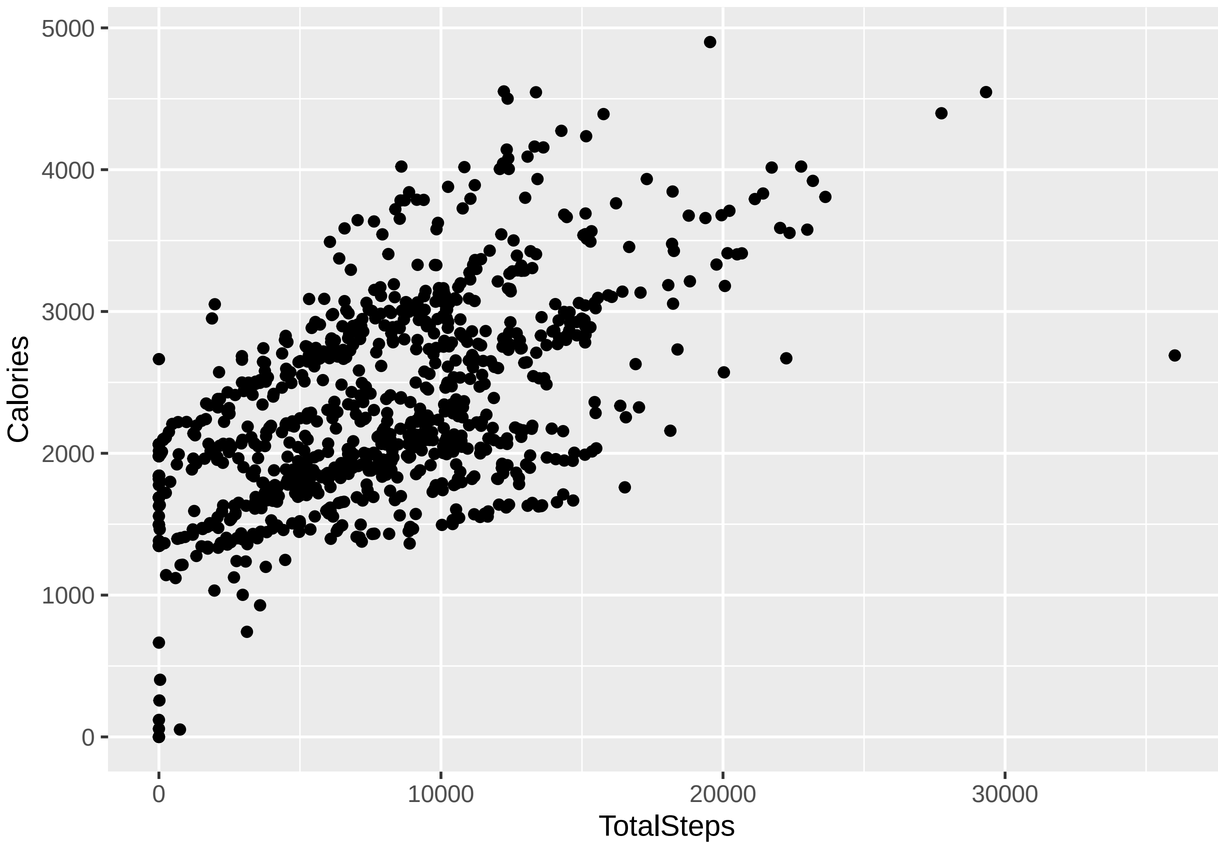
```
##  [1] "Id"                       "ActivityDate"
##  [3] "TotalSteps"               "TotalDistance"
##  [5] "TrackerDistance"          "LoggedActivitiesDistance"
##  [7] "VeryActiveDistance"       "ModeratelyActiveDistance"
##  [9] "LightActiveDistance"      "SedentaryActiveDistance"
## [11] "VeryActiveMinutes"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

```r
n_distinct(activity$Id)
```

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

This is a dataset about activity data, including total steps, distances travelled categorized by activity intensity (very active, moderately active, light active, and sedentary), the time spent at each intensity level, and calories (burned). There are records for 33 distinct users across different days, which is the maximum distinct users we have for this case study. The size of this sample is considered sufficient.

```
ggplot(data = activity) +
  geom_point(mapping = aes(x = TotalSteps, y = Calories))
```



Here I generated a quick plot to confirm if "Calories" IS calories burned. From the plot we can easily spot a positive correlation between total steps and calories, so it's reasonable to assume that Calories refer to calories burned (instead of consumed). However, I think that in a working environment, it's wiser to check with people who have more information about this dataset if possible, so as to avoid misinterpreting the data and coming to a skewed conclusion.

```
# Converting Date Format
activity$ActivityDate <- as.Date(activity$ActivityDate, format = "%m/%d/%Y")
```

I converted the date format in this dataset to Year-Month-day for consistency.

```
# Check for duplicates
duplicates <- activity[duplicated(activity), ]
```

There is no duplicate in this dataset.

```
# Check for missing values
any_na <- any(is.na(activity))
print(any_na)
```

```
## [1] FALSE
```

It returned FALSE, so there isn't any missing value. Now this dataset is also clean.

Inspecting and Cleaning Hourly Steps Dataset

```r
# Get a general idea about this dataset
head(hourly_steps)
```

```
##           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
```

```r
glimpse(hourly_steps)
```

```
## Rows: 22,099
## Columns: 3
## $ Id           <dbl> 1503960366, 1503960366, 1503960366, 1503960366, 150396036~
## $ ActivityHour <chr> "4/12/2016 12:00:00 AM", "4/12/2016 1:00:00 AM", "4/12/20~
## $ StepTotal    <int> 373, 160, 151, 0, 0, 0, 0, 0, 250, 1864, 676, 360, 253, 2~
```

```r
str(hourly_steps)
```

```
## '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 ...
```

```r
colnames(hourly_steps)
```

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

```r
n_distinct(hourly_steps$Id)
```

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

This is a dataset about number of steps taken measured by hours, which can givesus insights about when users are most and least active. This dataset contains data for 33 distinct users across different days, representing the full cohort available for this case study. The size of this sample is considered sufficient.

However, the date-time was stored as strings instead of proper date-time format, I needed to fix this.

```r
hourly_steps$ActivityHour = as.POSIXct(hourly_steps$ActivityHour, format = "%m/%d/%Y %I:%M:%S %p", tz=Sy
```

Now the data has been transformed into a POSIXct format and is ready to be used.

```r
# Duplicates
duplicates <- hourly_steps[duplicated(hourly_steps), ]
```

Again, no duplicates.

```r
# Missing values
missing_value <- any(is.na(hourly_steps))
print(missing_value)
```

```
## [1] FALSE
```

There are also no missing values, the quality of the datasets involved in this case study is good.

Evaluation of the data:

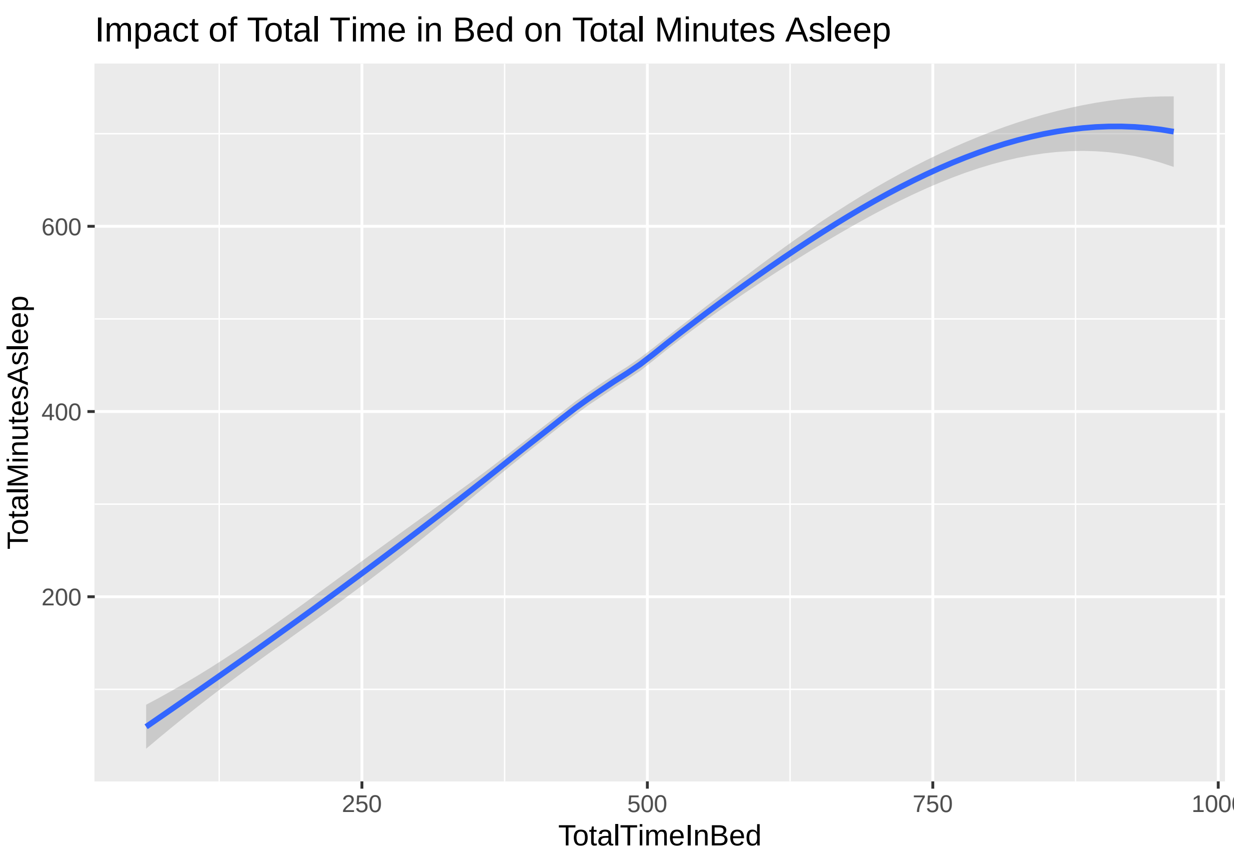
The collection of the datasets we use for this case comes from Kaggle, which has a usability score of 9.41, indicating that it's complete and credible. Upon inspection, there are only a couple of duplicates and no missing values, further confirming its credibility. However, this dataset collection does have limitations. The first limitation is the lack of column descriptions, which could potentially lead to misinterpretation and, subsequently, inaccurate conclusions. The second limitation is that although the important datasets encompass the maximum available distinct participants, the sample size of 30 is generally considered small and may therefore not be representative of the entire female population. Furthermore, the dataset description does not specify the gender of the participants, so we cannot be certain if we are investigating what we intend to investigate. If possible, we should request more data on female for a more accurate and representative reflection of the entire population. If not, we will have to be cautious when making conclusions and recommendations for actions, and always keep in mind that the whole analysis is conducted on a sample size of merely 30 people.

At this point, I was ready to dive deeper into the datasets, perform detailed analysis, create visualizations, and identify trends.

Plot from Sleep Dataset

```
ggplot(data = sleep, aes(x = TotalTimeInBed, y = TotalMinutesAsleep)) +
  geom_smooth() +
  labs(title = "Impact of Total Time in Bed on Total Minutes Asleep")
```

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



```r
sleep_less_than_300 <- sum(sleep$TotalMinutesAsleep < 300)
print(sleep_less_than_300 / nrow(sleep) * 100)
```

```
## [1] 12.19512
```

```r
sleep_less_than_360 <- sum(sleep$TotalMinutesAsleep < 360)
print(sleep_less_than_360 / nrow(sleep) * 100)
```

```
## [1] 24.39024
```

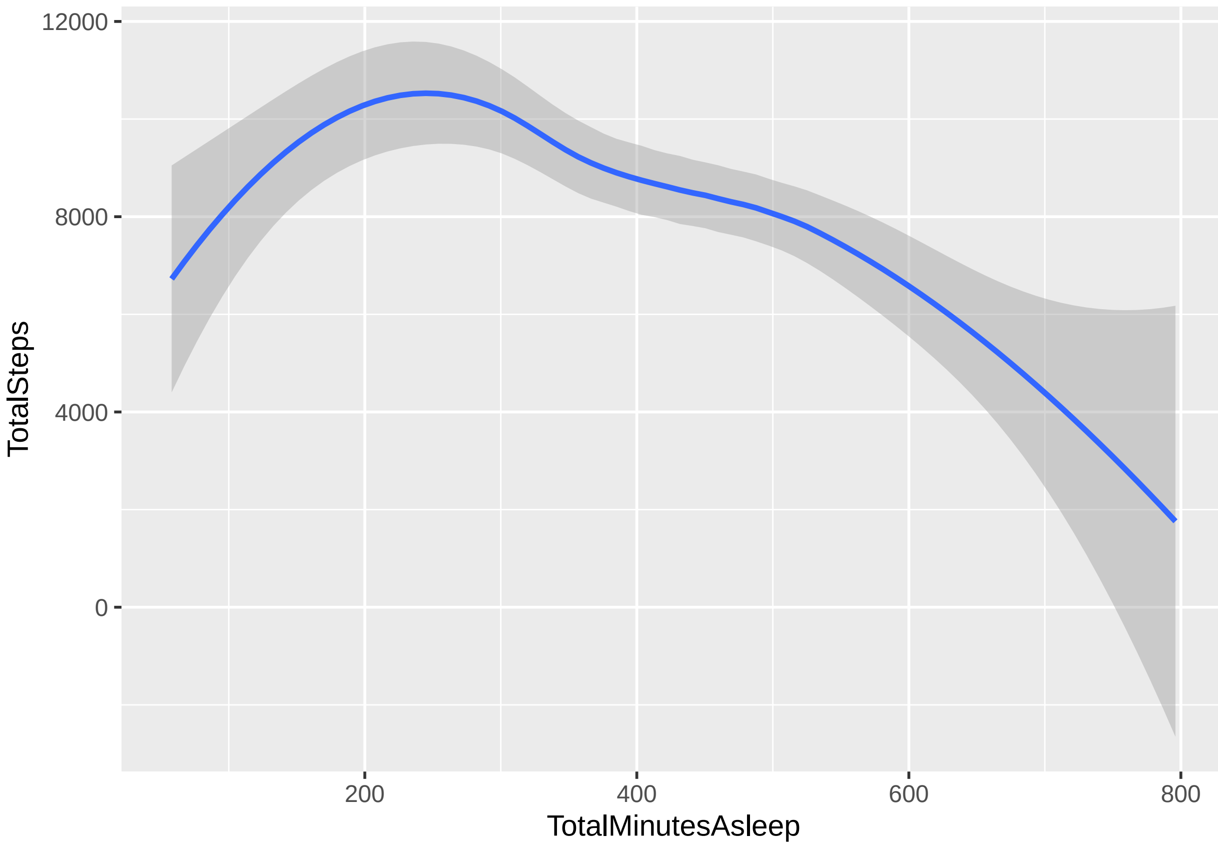
In general, there is a strong positive correlation between total time in bed and sleep duration. Therefore, we can infer from the chart that the longer someone spends in time, the longer they tend to sleep. An interesting point is that people who spend an extremely long time in bed don't sleep proportionally more. This suggests these individuals might be struggling to fall asleep.
Furthermore, I noticed that there are a considerable number of observations with total sleep time less than 300 minutes (5 hours). I computed the exact number and found that 12% of observations fall into this category. This proportion jumps to 24.4% if we raise the threshold to 6 hours, which is still below the recommended hours of sleep. This suggests that many people are sleeping less (or much less) than they should, which can be detrimental to their health.

Plot from Activity Combined with Sleep

```r
combined_activity_and_sleep <- activity %>%
  inner_join(sleep, by = c("Id" = "Id", "ActivityDate" = "SleepDay"))

ggplot(data = combined_activity_and_sleep, aes(x = TotalMinutesAsleep, y = TotalSteps)) +
  geom_smooth()
```

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

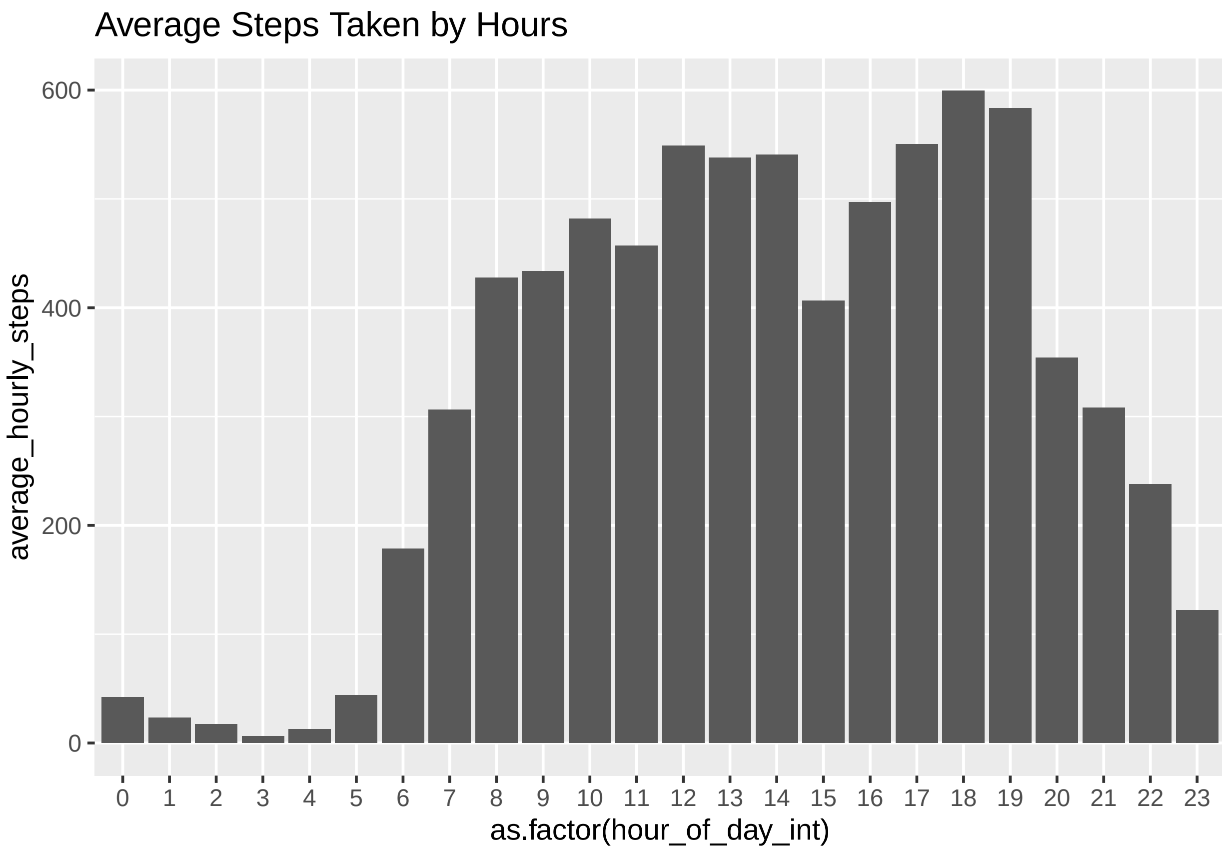


To gain deeper insights, I merged the Activity dataset with the Sleep dataset to investigate the correlation bewteen sleep duration and steps taken. The visualization of this combined data reveals a largely negative correlation, indicating that people with longer sleep durations tend to take fewer steps. This finding suggests an opportunity to target our product at individuals with higher sleep durations.

Plot from Hourly Steps

```
average_hourly_steps <- hourly_steps %>%
  mutate(hour_of_day_int = hour(ActivityHour)) %>%
  group_by(hour_of_day_int) %>%
  summarise(average_hourly_steps = mean(StepTotal))

ggplot(data = average_hourly_steps, aes(x = as.factor(hour_of_day_int), y = average_hourly_steps)) +
  geom_bar(stat = "identity") +
  labs(title = "Average Steps Taken by Hours")
```



The dataset was originally recorded in date-time format, but I was more interested in the hourly data on average. So, I extracted the hour and performed an mean calculation for each hour of the day across different individuals and days.

From the bar chart, we can see that on average, activity starts at 7:00 or 8:00, presumably when most people get up. The most steps occur around 17:00 to 19:00, which makes sense because people normally take more steps after work, either for commuting or for exercising. However, a point worth noticing is that the average step remains high during the daytime, It's reasonable to deduce that, in general, our participants do not engage in prolonged desk-bound work. But if we were to look at the entire female population, this may not hold true. Many females (as well as males) do work a corporate job in which they have little chance to move around during daytime. Hence, we would expect the steps taken during the daytime by the entire population to be fewer than what we observe in this dataset. This reminds us again that we are working with a fairly small sample and need to be cautious about the conclusions we draw.

However, this bar plot reveals a pattern that holds true for most people: activity levels reduce drastically from 20:00, with average steps falling sharply.

Key Findings and Recommendations for Marketing Strategy

The key takeaways from this analysis are:

- Total time in bed is positively correlated with sleep duration, meaning that people tend to sleep more if they spend more time in bed. So a recommendation based on this finding is that BellaBeat can highlight the app's notification features that helps users to get in bed early and sleep more.
- Sleep durations for users with extremely long time in bed are not proportionally long. This insight provides a foundation for targeting mindfulness content at people who have sleep problems.

- This study reveals a shorter-than-six-hour sleep duration for over 20% of the sample observations. This suggests that BellaBeat can emphasize the app's ability to lead to better health decisions.
- Individuals who sleep more tend to take fewer steps. This suggests that people with longer sleep durations may be a potentially promising market segment. It's recommended that Bellabeat target their products at these people and help them exercise more and enhance their health.
- The finding that people are most active around 17:00 to 19:00 suggests that BellaBeat can schedule marketing emails or messages just before they start to get active, as this is the time when they are most likely to check their phones, making the marketing most effective.
- The analysis also finds that the activity level plummet from 20:00, which suggests that this is the time when people transition to a post-activity and pre-sleep phase. Thus, this is also the perfect time for BellaBeat to promote stress management and mindfulness features of the app. BellaBeat can send messages to help users unwind and prepare for the next day.

These recommendations provide preliminary guidance on the marketing strategy. However, it has to be kept in mind that we are working with a small sample here, and that more data is required to make these conclusions more reliable.