Bellabeat Case Study

Ilma Umair

2023-08-06

Objective

Analyze smart device usage data in order to gain insight into how consumers use non-Bellabeat smart devices. Help the stakeholders of this project, Urška Sršen, Sando Mur and the Bellabeat marketing analytics team, to select one Bellabeat product to apply these insights to.

Dataset

The data for this project is an original dataset generated by respondents to a distributed survey via Amazon Mechanical Turk between 03.12.2016-05.12.2016. I obtained it from Kaggle where it's made available through the user Mobius. This dataset contains information from thirty Fitbit users who consented to the submission of their personal tracker data, including minute-level output for physical activity, heart rate, sleep monitoring, etc. This dataset is under the Public Domain license which entails that the person who associated a work with this deed has dedicated the work to the public domain by waiving all of his or her rights to the work worldwide under copyright law, including all related and neighboring rights, to the extent allowed by law.

The link to this dataset is https://www.kaggle.com/datasets/arashnic/fitbit?resource=download&select=Fitabase+Data+4.12.16-5.12.16

There are, however, some limitations to this datset. It has a small sample size of only 30 users and does not have all the parameters from each user like gender, occupation, etc.

By analyzing the data of a similar product, we can gain useful insights into user habits and come up with actionable suggestions that our stakeholders can use on their own products, giving them an edge in their particular niche of the market.

Tools used for data analysis

First, an Excel worksheet was created showcasing all the columns from each table. This gives a quick overview of all the parameters that the data was collected for and help decide which ones to choose for the analysis.

Next, RCloud was employed to analyze data and create visualizations. Finally, using it's Knit function the report was put together in a PDF document.

```
# Install and load all the required packages

install.packages("tidyverse")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.2'
## (as 'lib' is unspecified)
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
```

```
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
```

```
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(readxl)
library(readr)
install.packages("here")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.2'
## (as 'lib' is unspecified)
```

```
library("here")
```

```
## here() starts at /cloud/project
```

```
install.packages("skimr")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.2'
## (as 'lib' is unspecified)
```

```
library("skimr")
install.packages("janitor")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.2'
## (as 'lib' is unspecified)
```

```
library("janitor")
```

```
##
## Attaching package: 'janitor'
##
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
install.packages("dplyr")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.2'
## (as 'lib' is unspecified)
```

```
library("dplyr")
```

Reading files using the readr package and saving them in a shorter, easy to reference variables.

```
# Reading our files and saving them in a shorter variable

library(readr)
dailyActivity <- 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.
```

```
dailyIntensities<- read_csv("dailyIntensities_merged.csv")
```

```
## Rows: 940 Columns: 10
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): ActivityDay
## dbl (9): Id, SedentaryMinutes, LightlyActiveMinutes, FairlyActiveMinutes, Ve...
##
## 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.
```

The skim_without-charts() function was used to get a high level view of the dataset and quickly see if there are any missing characters in it.

```
# Get a summary of dailyActivity dataset. Shows if there are any missing characters too.
skim_without_charts(dailyActivity)
```

Table 1: Data summary

Name	dailyActivity
Number of rows	940
Number of columns	15
Column type frequency:	
character	1
numeric	14
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
ActivityDate	0	1	8	9	0	31	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
Id	0	1	4.855407e+09	2.924805e+09	1503960366	2320127e+09	4.445115e+09	6.962181e+09	8.877689e+09
TotalSteps	0	1	7.637910e+03	5.087150e+03	0	3.789750e+03	7.405500e+03	1.0727000e+04	3.6601900e+04
TotalDistance	0	1	5.490000e+00	3.920000e+00	0	2.620000e+00	5.240000e+00	7.710000e+00	2.803000e+01
TrackerDistance	0	1	5.480000e+00	3.910000e+00	0	2.620000e+00	5.240000e+00	7.710000e+00	2.803000e+01
LoggedActivitiesDistance	0	1	1.100000e-01	6.200000e-01	0	0.000000e+00	0.000000e+00	0.000000e+00	4.940000e+00
VeryActiveDistance	0	1	1.500000e+00	2.060000e+00	0	0.000000e+00	2.050000e-01	2.092000e+00	2.1992000e+01
ModeratelyActiveDistance	0	1	5.700000e-01	8.800000e-01	0	0.000000e+00	8.000000e-01	6.480000e-01	6.480000e+00
LightActiveDistance	0	1	3.340000e+00	2.040000e+00	0	1.950000e+00	3.060000e+00	4.780000e+00	1.0071000e+01

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
SedentaryActiveDistance	0	1	0.000000e+00	0.000000e-02	0	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e-01
VeryActiveMinutes	0	1	2.116000e+01	2.284000e+01	0	0.000000e+00	4.000000e+00	3.200000e+01	2.100000e+02
FairlyActiveMinutes	0	1	1.356000e+01	1.999000e+01	0	0.000000e+00	6.000000e+00	1.900000e+01	1.430000e+02
LightlyActiveMinutes	0	1	1.928100e+02	1.091700e+02	0	1.270000e+02	1.990000e+02	2.640000e+02	5.180000e+02
SedentaryMinutes	0	1	9.912100e+02	3.012700e+02	0	7.297500e+02	1.057500e+03	1.329500e+03	1.440000e+03
Calories	0	1	2.303610e+03	7.181700e+02	0	1.828500e+03	2.134000e+03	2.793250e+03	4.900000e+03

The n_missing() function from the skimr package was used to show columns with missing values.

```
# Which of the columns have missing values?
dailyActivity %>%
  skim() %>%
  dplyr::filter(n_missing > 0)
```

```
## # A tibble: 0 x 17
## # i 17 variables: skim_type <chr>, skim_variable <chr>, n_missing <int>,
## #   complete_rate <dbl>, character.min <int>, character.max <int>,
## #   character.empty <int>, character.n_unique <int>,
## #   character.whitespace <int>, numeric.mean <dbl>, numeric.sd <dbl>,
## #   numeric.p0 <dbl>, numeric.p25 <dbl>, numeric.p50 <dbl>, numeric.p75 <dbl>,
## #   numeric.p100 <dbl>, numeric.hist <chr>
```

The sum(duplicated()) function was used to check for duplicates.

```
# Check if there are duplicate values in dailyAcitivity
sum(duplicated(dailyActivity))
```

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

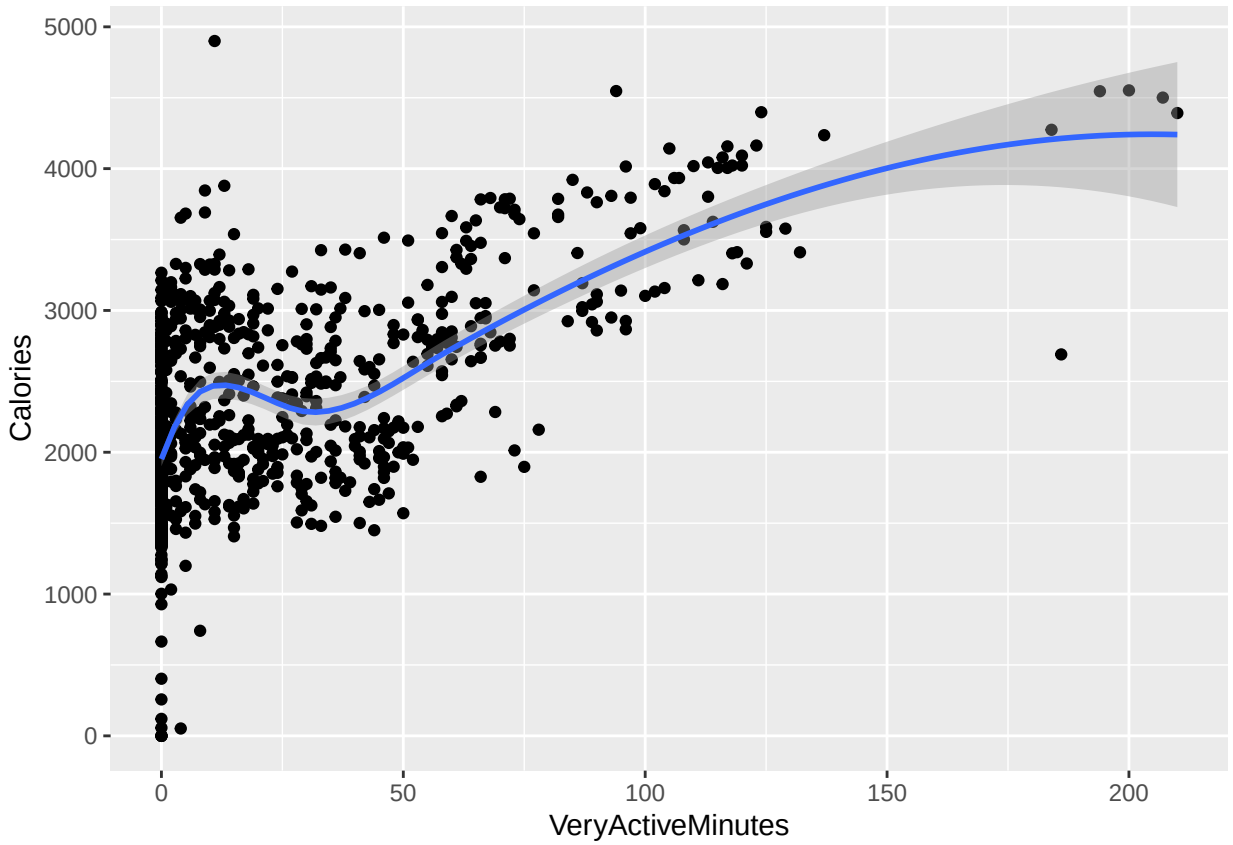
```
head(dailyActivity)
```

```
## # A tibble: 6 x 15
##           Id ActivityDate TotalSteps TotalDistance TrackerDistance
##        <dbl> <chr>             <dbl>         <dbl>           <dbl>
## 1 1503960366 4/12/2016         13162          8.5             8.5
## 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
## # i 10 more variables: LoggedActivitiesDistance <dbl>,
## #   VeryActiveDistance <dbl>, ModeratelyActiveDistance <dbl>,
## #   LightActiveDistance <dbl>, SedentaryActiveDistance <dbl>,
## #   VeryActiveMinutes <dbl>, FairlyActiveMinutes <dbl>,
## #   LightlyActiveMinutes <dbl>, SedentaryMinutes <dbl>, Calories <dbl>
```

Scatter plots to compare the calories burnt based on how much time they spent being very active and sedentary. These graphs confirm the theory that more calories are burnt being very active compared to being sedentary.

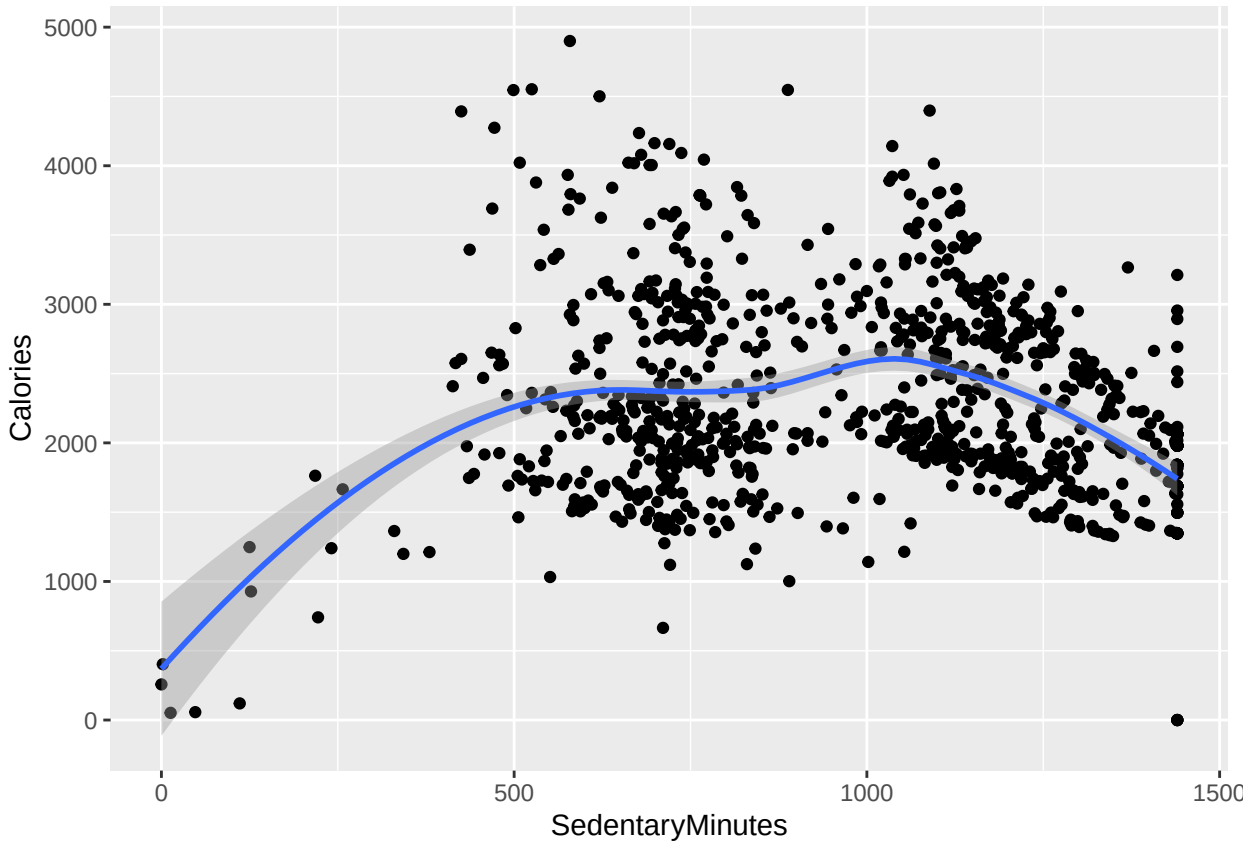
```
# Plotting calories with different categories of active minutes

ggplot(data = dailyActivity, aes(x=VeryActiveMinutes, y=Calories)) + geom_point() + geom_smooth()
```

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




```
ggplot(data = dailyActivity, aes(x=SedentaryMinutes, y=Calories)) + geom_point() + geom_smooth()
```

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



A new column was created in 'dailyIntensities' dataframe to calculate the weekday based on the date in the column ActivityDay and saving the mutated table in the variant 'weekday_steps'.Then, 'weekday_steps' was grouped according to the days of the week and a table was plotted with the mean amount of time doing each of the different intensity of physical activity.

```
#Make a new column in dailyIntensities for weekday and save the output in a new dataframe

weekday_steps <- dailyIntensities %>%
  mutate(dailyIntensities, weekday = weekdays(as.Date(ActivityDay))) %>%
  group_by(weekday)
head(weekday_steps)
```

```
## # A tibble: 6 x 11
## # Groups:   weekday [2]
##         Id ActivityDay SedentaryMinutes LightlyActiveMinutes FairlyActiveMinutes
##      <dbl> <chr>                 <dbl>                <dbl>                <dbl>
## 1   1.50e9 4/12/2016               728                  328                   13
## 2   1.50e9 4/13/2016               776                  217                   19
## 3   1.50e9 4/14/2016              1218                  181                   11
## 4   1.50e9 4/15/2016               726                  209                   34
## 5   1.50e9 4/16/2016               773                  221                   10
## 6   1.50e9 4/17/2016               539                  164                   20
## # i 6 more variables: VeryActiveMinutes <dbl>, SedentaryActiveDistance <dbl>,
## #   LightActiveDistance <dbl>, ModeratelyActiveDistance <dbl>,
## #   VeryActiveDistance <dbl>, weekday <chr>
```

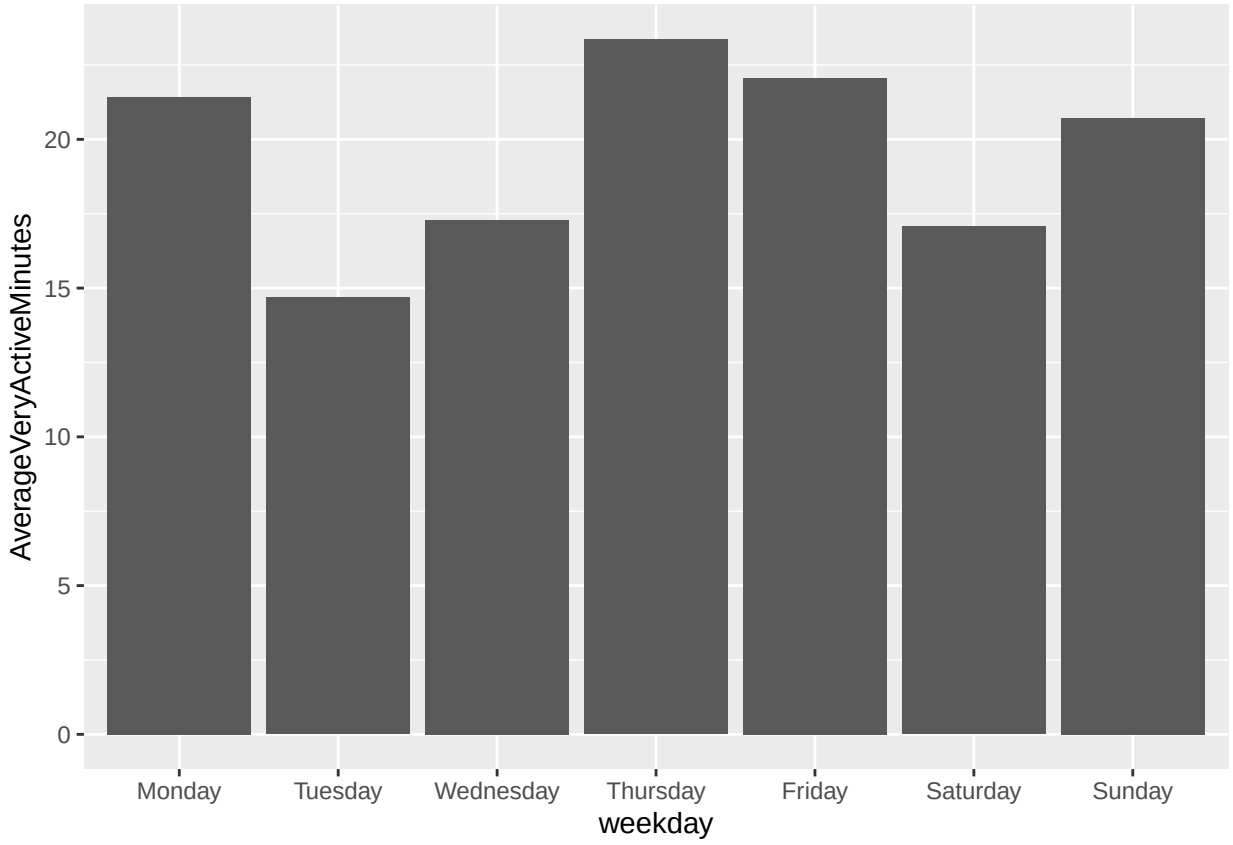
```
weekday_steps$weekday <-ordered(weekday_steps$weekday, levels=c("Monday", "Tuesday", "Wednesday", "Thurs
                                                                "Friday", "Saturday", "Sunda
```

```
weekday_steps <-weekday_steps%>%
  group_by(weekday) %>%
  summarize (AverageSedentaryMinutes = mean(SedentaryMinutes), AverageLightlyActiveMinutes = mean(Light
  drop_na()
head(weekday_steps)
```

```
## # A tibble: 6 x 5
##   weekday   AverageSedentaryMinu~1 AverageLightlyActive~2 AverageFairlyActiveM~3
##   <ord>                      <dbl>                  <dbl>                  <dbl>
## 1 Monday                     1024                    200.                   9.47
## 2 Tuesday                     834.                   156.                   9.65
## 3 Wednesday                   996.                   181.                  12.6
## 4 Thursday                    956.                   169.                  17.1
## 5 Friday                     1033.                   207.                  15.4
## 6 Saturday                    958.                   185.                  17.4
## # i abbreviated names: 1: AverageSedentaryMinutes,
## #   2: AverageLightlyActiveMinutes, 3: AverageFairlyActiveMinutes
## # i 1 more variable: AverageVeryActiveMinutes <dbl>
```

Next a bar chart was created to see the amount of time spent each day of the week doing the most amount of activity. The scale of the Y axis was adjusted to better see the differences between the days. The resulting bar chart shows a marked decrease in physical activity on Tuesdays, a key insight that Bellabeat users can tap into.

```
#Plot a bar chart for AverageSedentaryMinutes on each weekday

ggplot(data = weekday_steps) + geom_col(mapping = aes(x = weekday, y = AverageVeryActiveMinutes))
```



Conclusion

Bellabeat can use this data to market its fitness trackers like the Ivy or the Leaf. Surprising insights, like the Tuesday slump of very active minutes, into a user's daily or weekly activities can urge them to be more conscious of their wellness habits and take actions like buy a fitness tracker to track their daily activities. The marketing team can also showcase the calories burnt with increasing number of minutes spent being very active and suggest to the users to ascertain their sweet spot where the calories burnt are highest, so they can better plan their exercise routines. In conclusion, data analysis of a similar product can help Bellabeat market its fitness tracker better and show potential customers the benefits of having it.