How Can a Wellness Technology Company Play It Smart?

Chinazom Jennifer Okoli

2023-12-09

Introduction

Bellabeat is a high-tech company that manufactures health-focused smart products for women. Collecting data on activity, sleep, stress, and reproductive health has allowed Bellabeat to empower women with knowledge about their own health and habits. Bellabeat products comprises of the Bellabeat app, Leaf (which is worn as a bracelet, necklace, or clip), Time (wellness watch) and Spring (water bottle). All the other products are connected to the Bellabeat App.

Urška Sršen, co-founder and Chief Creative Officer of Bellabeat, believes that analyzing smart device fitness data could help unlock new growth opportunities for the company.Sršen asks to analyze smart device usage data in order to gain insight into how consumers use non-Bellabeat smart devices. We're to focus on Bellabeat's product, **Time**, and examine data from **fitness tracker**—a smart device similar to Bellabeat's, made by a different company called **Fitbit**. This analysis aims to understand how consumers are utilizing these smart devices.

A dataset which has a total of 18 data files, named (CC0: Public Domain, dataset made available through Mobius) was given which contains personal fitness tracker from thirty eligible Fitbit users who consented to the submission of personal tracker data. This includes information about daily activity, steps, tracker distance, calories, heart rate, sedentary time and sleep monitoring, that can be used to explore users' habits.

The tool we're using for this study is RMarkdown within RStudo. This will enable us to conduct analysis, create visualizations, document our work, and facilitate seamless sharing of results.

Buisness Goal

- To find some trends in smart device usage.
- How these trends could apply to Bellabeat customers.
- How these trends could help influence Bellabeat marketing strategy.

Installing and loading common packages and libraries for our analysis.

First off, we set a CRAN mirror;this specifies the location from which R will download packages when we install them. CRAN (Comprehensive R Archive Network) is a network of servers worldwide that mirror the same collection of R packages. By setting a CRAN mirror, we are telling R where to fetch the packages from.

The code below is a combination of R code and Markdown syntax and is typically used in R Markdown (Rmd) documents.

```
knitr::opts_chunk$set(echo = TRUE)
# Set a CRAN mirror
options(repos = c(CRAN = "https://cloud.r-project.org"))
```

```r
# Install packages
install.packages("tidyverse")
```

```
## Installing package into 'C:/Users/ADMIN/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)

## package 'tidyverse' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\ADMIN\AppData\Local\Temp\RtmpqgC2Aw\downloaded_packages
```

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

```
## Installing package into 'C:/Users/ADMIN/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)

## package 'dplyr' successfully unpacked and MD5 sums checked

## Warning: cannot remove prior installation of package 'dplyr'

## Warning in file.copy(savedcopy, lib, recursive = TRUE): problem copying
## C:\Users\ADMIN\AppData\Local\R\win-library\4.3\00LOCK\dplyr\libs\x64\dplyr.dll
## to C:\Users\ADMIN\AppData\Local\R\win-library\4.3\dplyr\libs\x64\dplyr.dll:
## Permission denied

## Warning: restored 'dplyr'

##
## The downloaded binary packages are in
##  C:\Users\ADMIN\AppData\Local\Temp\RtmpqgC2Aw\downloaded_packages
```

```r
library("tidyverse")
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2

## -- 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
```

```r
library(dplyr)
library(magrittr)
```

```
##
## Attaching package: 'magrittr'
##
## The following object is masked from 'package:purrr':
##
##     set_names
##
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(purrr)
```

Loading and exploring our CSV files

Here we created data frames with the same name as our CSV files from the dataset and import them using the file path; Typed in the data frame name, the assignment operator(<-), then the read.csv() which is the function for importing CSV files in R finally the file path which was gotten by clicking on the data file then **copy as path**. Paste in the parenthesis of the read.csv.

```
dailyActivity_merged <- read.csv("C:/Users/ADMIN/OneDrive/Documents/Fitabase Data 4.12.16-5.12.16/daily

View(dailyActivity_merged)

sleepDay_merged <- read.csv("C:/Users/ADMIN/OneDrive/Documents/Fitabase Data 4.12.16-5.12.16/sleepDay_m

View(sleepDay_merged)
```

We will concentrate on specific CSV files for this analysis among the provided 18, as many share identical data, and a few contain incomplete data which won't be relevant for this study.

Understanding some summary statistics

In this analysis, we are consolidating our data by computing the average of each variable. Data is aggregated and averaged in analysis to provide a summary measure that represents the central tendency of a set of values. This simplifies complex datasets, highlights overall trends, and facilitates easier interpretation of the information.

Averaging helps smooth out variations, making it a commonly used method for obtaining a representative value from a larger dataset.

The syntax used in the first analysis was explained below as a guide for the subsequent analysis.

To create a new data frame 'calculated__average__activity' to know the average activity of each participant for this period of time.

```
calculated_average_activity <- dailyActivity_merged %>%
  group_by(Id) %>%
  summarise(
    AVGVeryActive = mean(as.integer(VeryActiveMinutes), na.rm = TRUE),
```

```r
    AVGFairlyActive = mean(as.integer(FairlyActiveMinutes), na.rm = TRUE),
    AVGLightActive = mean(as.integer(LightlyActiveMinutes), na.rm = TRUE),
    AVG_Activity = mean(
      as.integer(VeryActiveMinutes) +
      as.integer(FairlyActiveMinutes) +
      as.integer(LightlyActiveMinutes),
      na.rm = TRUE
    )
  )
```

Let's break down the syntax and understand each part:

Calculation of Average Activity:

- AVGVeryActive: Average value of VeryActiveMinutes for each ID.
- AVGFairlyActive: Average value of FairlyActiveMinutes for each ID.
- AVGLightActive: Average value of LightlyActiveMinutes for each ID.

Calculation of Total Activity:

- AVG_Activity: Average total activity, which is the sum of VeryActiveMinutes, FairlyActiveMinutes, and LightlyActiveMinutes for each ID.

Integer Conversion: The **as.integer function** is applied to the calculated averages to ensure they are integers.

Data Frame Creation:

- The results are stored in a new data frame named 'calculated_average_activity'.
- Each row of this data frame corresponds to a unique ID, and the columns represent the calculated average values.

```r
# Print the result
print(calculated_average_activity)
```

```
## # A tibble: 33 x 5
##            Id AVGVeryActive AVGFairlyActive AVGLightActive AVG_Activity
##         <dbl>         <dbl>           <dbl>          <dbl>        <dbl>
##  1 1503960366       38.7            19.2            220.         278.
##  2 1624580081        8.68            5.81           153.         168.
##  3 1644430081        9.57           21.4            178.         209.
##  4 1844505072        0.129           1.29           115.         117.
##  5 1927972279        1.32            0.774           38.6         40.7
##  6 2022484408       36.3            19.4            257.         313.
##  7 2026352035        0.0968          0.258          257.         257
##  8 2320127002        1.35            2.58           198.         202.
##  9 2347167796       13.5            20.6            252.         287.
## 10 2873212765       14.1             6.13           308          328.
## # i 23 more rows
```

To create a table 'calculated_average_inactivity' using SedentaryTime from the dailyActive_merged data frame, TotalMinutesAsleep and TotalTimeInBed from the sleepDay_merged data frame to know the average inactivity time of the participants.

Calculating the AVGSedentaryTime

```
averageSedentary_time <- dailyActivity_merged %>%
  group_by(Id) %>%
  summarise(
    AVG_SedentaryMinutes = as.integer(mean(as.integer(SedentaryMinutes), na.rm = TRUE))
  )
```

Calculating the average of TotalMinutesAsleep and TotalTimeInBed as 'average_BedTime'

```
average_BedTime <- sleepDay_merged %>%
  group_by(Id) %>%
  summarise(
    AVG_TotalMinutesAsleep = as.integer(mean(as.integer(TotalMinutesAsleep), na.rm = TRUE)),
    AVG_TotalTimeInBed = as.integer(mean(as.integer(TotalTimeInBed), na.rm = TRUE))
)
```

To merge the both data frame as average_Inactivity

```
average_Inactivity <- inner_join(average_BedTime, averageSedentary_time, by = "Id")
```

```
# Print the result
print(average_Inactivity)
```

```
## # A tibble: 24 x 4
##            Id AVG_TotalMinutesAsleep AVG_TotalTimeInBed AVG_SedentaryMinutes
##         <dbl>                  <int>              <int>                <int>
##  1 1503960366                    360                383                  848
##  2 1644430081                    294                346                 1161
##  3 1844505072                    652                961                 1206
##  4 1927972279                    417                437                 1317
##  5 2026352035                    506                537                  689
##  6 2320127002                     61                 69                 1220
##  7 2347167796                    446                491                  687
##  8 3977333714                    293                461                  707
##  9 4020332650                    349                379                 1237
## 10 4319703577                    476                501                  735
## # i 14 more rows
```

To calculate the average TotalSteps and TrackerDistance as 'averageSteps_Distance'

```
averageSteps_Distance <- dailyActivity_merged %>%
  group_by(Id) %>%
  summarise(
    AVG_TotalSteps = as.integer(mean(as.integer(TotalSteps), na.rm = TRUE)),
    AVG_TrackerDistance = mean(TrackerDistance, na.rm = TRUE),
  )
```

```r
# Print the result
print(averageSteps_Distance)
```

```
## # A tibble: 33 x 3
##             Id AVG_TotalSteps AVG_TrackerDistance
##          <dbl>          <int>               <dbl>
##   1 1503960366          12116               7.81
##   2 1624580081           5743               3.91
##   3 1644430081           7282               5.30
##   4 1844505072           2580               1.71
##   5 1927972279            916               0.635
##   6 2022484408          11370               8.08
##   7 2026352035           5566               3.45
##   8 2320127002           4716               3.19
##   9 2347167796           9519               6.36
##  10 2873212765           7555               5.10
## # i 23 more rows
```

To calculate average Calories

```r
average_Calories <- dailyActivity_merged %>%
  group_by(Id) %>%
  summarise(
    AVG_Calories = as.integer(mean(as.integer(Calories), na.rm = TRUE))
  )
```

```r
print(average_Calories)
```

```
## # A tibble: 33 x 2
##             Id AVG_Calories
##          <dbl>        <int>
##   1 1503960366         1816
##   2 1624580081         1483
##   3 1644430081         2811
##   4 1844505072         1573
##   5 1927972279         2172
##   6 2022484408         2509
##   7 2026352035         1540
##   8 2320127002         1724
##   9 2347167796         2043
##  10 2873212765         1916
## # i 23 more rows
```

To merge all calculated data frames into a data frame 'FitbaseData_Calculated' for easy visualization

```r
# List of data frames
df_list <- list(average_BedTime, average_Calories, average_Inactivity, averageSedentary_time, averageSt
```

```r
# Merge data frames in the list based on the common column (Id)
FitbaseData_Calculated <- reduce(df_list, merge, by = "Id",all = TRUE)

# Resulting data frame (FitbaseData_Calculated)
print(FitbaseData_Calculated)
```

```
##            Id AVG_TotalMinutesAsleep.x AVG_TotalTimeInBed.x AVG_Calories
## 1  1503960366                      360                  383         1816
## 2  1624580081                       NA                   NA         1483
## 3  1644430081                      294                  346         2811
## 4  1844505072                      652                  961         1573
## 5  1927972279                      417                  437         2172
## 6  2022484408                       NA                   NA         2509
## 7  2026352035                      506                  537         1540
## 8  2320127002                       61                   69         1724
## 9  2347167796                      446                  491         2043
## 10 2873212765                       NA                   NA         1916
## 11 3372868164                       NA                   NA         1933
## 12 3977333714                      293                  461         1513
## 13 4020332650                      349                  379         2385
## 14 4057192912                       NA                   NA         1973
## 15 4319703577                      476                  501         2037
## 16 4388161847                      403                  426         3093
## 17 4445114986                      385                  416         2186
## 18 4558609924                      127                  140         2033
## 19 4702921684                      421                  441         2965
## 20 5553957443                      463                  505         1875
## 21 5577150313                      432                  460         3359
## 22 6117666160                      478                  510         2261
## 23 6290855005                       NA                   NA         2599
## 24 6775888955                      349                  369         2131
## 25 6962181067                      448                  466         1982
## 26 7007744171                       68                   71         2544
## 27 7086361926                      453                  466         2566
## 28 8053475328                      297                  301         2945
## 29 8253242879                       NA                   NA         1788
## 30 8378563200                      443                  483         3436
## 31 8583815059                       NA                   NA         2732
## 32 8792009665                      435                  453         1962
## 33 8877689391                       NA                   NA         3420
##    AVG_TotalMinutesAsleep.y AVG_TotalTimeInBed.y AVG_SedentaryMinutes.x
## 1                       360                  383                    848
## 2                        NA                   NA                     NA
## 3                       294                  346                   1161
## 4                       652                  961                   1206
## 5                       417                  437                   1317
## 6                        NA                   NA                     NA
## 7                       506                  537                    689
## 8                        61                   69                   1220
## 9                       446                  491                    687
## 10                       NA                   NA                     NA
## 11                       NA                   NA                     NA
## 12                      293                  461                    707
```

```
## 13                      349                 379                   1237
## 14                       NA                  NA                     NA
## 15                      476                 501                    735
## 16                      403                 426                    836
## 17                      385                 416                    829
## 18                      127                 140                   1093
## 19                      421                 441                    766
## 20                      463                 505                    668
## 21                      432                 460                    754
## 22                      478                 510                    796
## 23                       NA                  NA                     NA
## 24                      349                 369                   1299
## 25                      448                 466                    662
## 26                       68                  71                   1055
## 27                      453                 466                    850
## 28                      297                 301                   1148
## 29                       NA                  NA                     NA
## 30                      443                 483                    716
## 31                       NA                  NA                     NA
## 32                      435                 453                   1060
## 33                       NA                  NA                     NA
##    AVG_SedentaryMinutes.y AVG_TotalSteps AVG_TrackerDistance AVGVeryActive
## 1                     848          12116           7.8096774   38.70967742
## 2                    1257           5743           3.9148387    8.67741935
## 3                    1161           7282           5.2953334    9.56666667
## 4                    1206           2580           1.7061290    0.12903226
## 5                    1317            916           0.6345161    1.32258065
## 6                    1112          11370           8.0841935   36.29032258
## 7                     689           5566           3.4548387    0.09677419
## 8                    1220           4716           3.1877419    1.35483871
## 9                     687           9519           6.3555555   13.50000000
## 10                   1097           7555           5.1016129   14.09677419
## 11                   1077           6861           4.7070000    9.15000000
## 12                    707          10984           7.5169999   18.90000000
## 13                   1237           2267           1.6261290    5.19354839
## 14                   1217           3838           2.8625000    0.75000000
## 15                    735           7268           4.8922580    3.58064516
## 16                    836          10813           8.3932259   23.16129032
## 17                    829           4796           3.2458064    6.61290323
## 18                   1093           7685           5.0806452   10.38709677
## 19                    766           8572           6.9551613    5.12903226
## 20                    668           8612           5.6396774   23.41935484
## 21                    754           8304           6.2133333   87.33333333
## 22                    796           7046           5.3421429    1.57142857
## 23                   1193           5649           4.2724138    2.75862069
## 24                   1299           2519           1.8134615   11.00000000
## 25                    662           9794           6.5193549   22.80645161
## 26                   1055          11323           7.5757692   31.03846154
## 27                    850           9371           6.3880645   42.58064516
## 28                   1148          14763          11.4751612   85.16129032
## 29                   1287           6482           4.6673685   20.52631579
## 30                    716           8717           6.9135485   58.67741935
## 31                   1267           7198           5.6154838    9.67741935
## 32                   1060           1853           1.1865517    0.96551724
```

```
## 33                   1112          16040          13.2129031   66.06451613
##    AVGFairlyActive AVGLightActive AVG_Activity
## 1       19.1612903      219.93548    277.80645
## 2        5.8064516      153.48387    167.96774
## 3       21.3666667      178.46667    209.40000
## 4        1.2903226      115.45161    116.87097
## 5        0.7741935       38.58065     40.67742
## 6       19.3548387      257.45161    313.09677
## 7        0.2580645      256.64516    257.00000
## 8        2.5806452      198.19355    202.12903
## 9       20.5555556      252.50000    286.55556
## 10       6.1290323      308.00000    328.22581
## 11       4.1000000      327.90000    341.15000
## 12      61.2666667      174.76667    254.93333
## 13       5.3548387       76.93548     87.48387
## 14       1.5000000      103.00000    105.25000
## 15      12.3225806      228.77419    244.67742
## 16      20.3548387      229.35484    272.87097
## 17       1.7419355      209.09677    217.45161
## 18      13.7096774      284.96774    309.06452
## 19      26.0322581      237.48387    268.64516
## 20      13.0000000      206.19355    242.61290
## 21      29.8333333      147.93333    265.10000
## 22       2.0357143      288.35714    291.96429
## 23       3.7931034      227.44828    234.00000
## 24      14.8076923       40.15385     65.96154
## 25      18.5161290      245.80645    287.12903
## 26      16.2692308      280.73077    328.03846
## 27      25.3548387      143.83871    211.77419
## 28       9.5806452      150.96774    245.70968
## 29      14.3157895      116.89474    151.73684
## 30      10.2580645      156.09677    225.03226
## 31      22.1935484      138.29032    170.16129
## 32       4.0344828       91.79310     96.79310
## 33       9.9354839      234.70968    310.70968
```

Note: We use the **all = TRUE** to return all records from all the data frames, no data was filtered out. This is because the data frames do not have the same number of records, All except 'average_BedTime which has 24, has 33 records. Also as a result of this merging, columns that exist in more than one data frame are labeled with ".x" and ".y" for differentiation.

Plotting some explorations using Scatter plot and Smoothing.

The relationship between two variables in a scatter plot provides valuable insights into overall activity levels. A negative correlation suggests that as one variable increases, the other decreases, indicating a potential link between them. Conversely, a positive correlation implies that higher values in one variable are associated with higher values in the other. A zero correlation suggests the absence of a linear relationship between two variables. It indicates that changes in one variable are not systematically associated with changes in the other, meaning that as values of one variable vary, there is no consistent pattern or trend in the values of the other variable.

Adding Smoothing to a scatter plot enhances the interpretability of the plot, making it easier to identify general patterns and draw insights.

Understanding this correlation helps tailor health recommendations and customize product messaging and features to address the specific needs of different customer segments, enhancing overall user engagement and satisfaction.

Let's find trends using visualization, we first load our plotting package - ggplot2

```r
library("ggplot2")
```

How are the number of steps taken related to the duration of sedentary minutes?

```r
# Creating scatter plot
ggplot(data=FitbaseData_Calculated, aes(x=AVG_SedentaryMinutes.y, y=AVG_TotalSteps))+
geom_point() +
labs(title = "Scatter Plot", x = "AVG_SedentaryMinutes.y", y = "AVG_TotalSteps")+
  geom_smooth()
```

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

The trend in this scatter plot shows that as **Average Total Steps** increases, **average Sedentary Minutes** decreases. This suggests that there is a **negative relationship** between Average Total Steps and average Sedentary Minutes. In other words, the more steps a person takes, the less time they spend sitting. This could imply that physical activity is inversely related to sedentary behavior.

A summarized explanation of the above syntax which will guide as a reference for subsequent ones.

The syntax employs functions from the "ggplot2" package in R. The ggplot() function initializes the plot, aes() specifies the aesthetic mappings that is which variable takes either the x or y axis. The geom_point() for scattered plot, adds points to the plot, labs() sets the title and axis labels, and geom_smooth() incorporates a smoothed trend line. Additionally, drop_na() is used for handling missing values in the data.

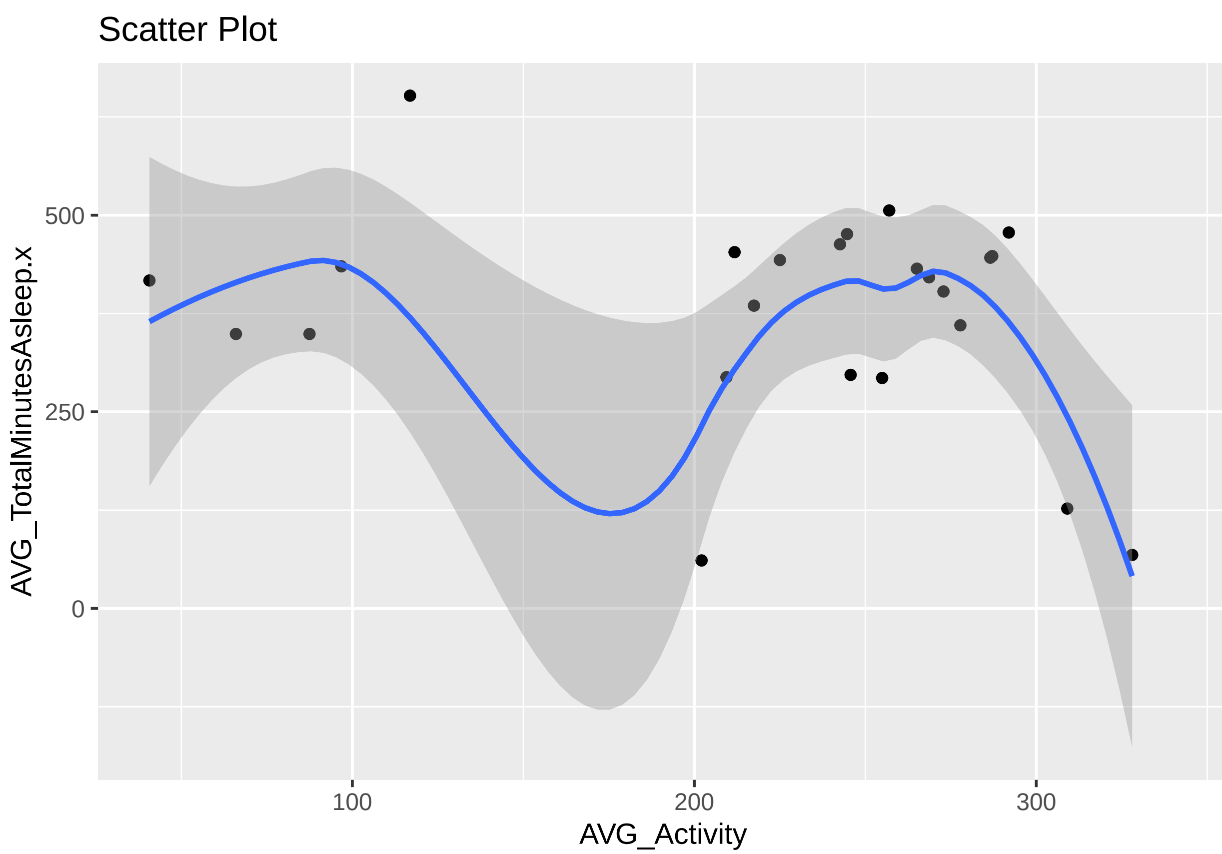
Relationship between activity and sleep

```
ggplot(data=FitbaseData_Calculated, aes(x=AVG_Activity, y=AVG_TotalMinutesAsleep.x,drop_na()))+
  geom_point() +
  labs(title = "Scatter Plot", x = "AVG_Activity", y = "AVG_TotalMinutesAsleep.x")+
  geom_smooth()
```

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

## Warning: Removed 9 rows containing non-finite values (`stat_smooth()`).

## Warning: Removed 9 rows containing missing values (`geom_point()`).
```



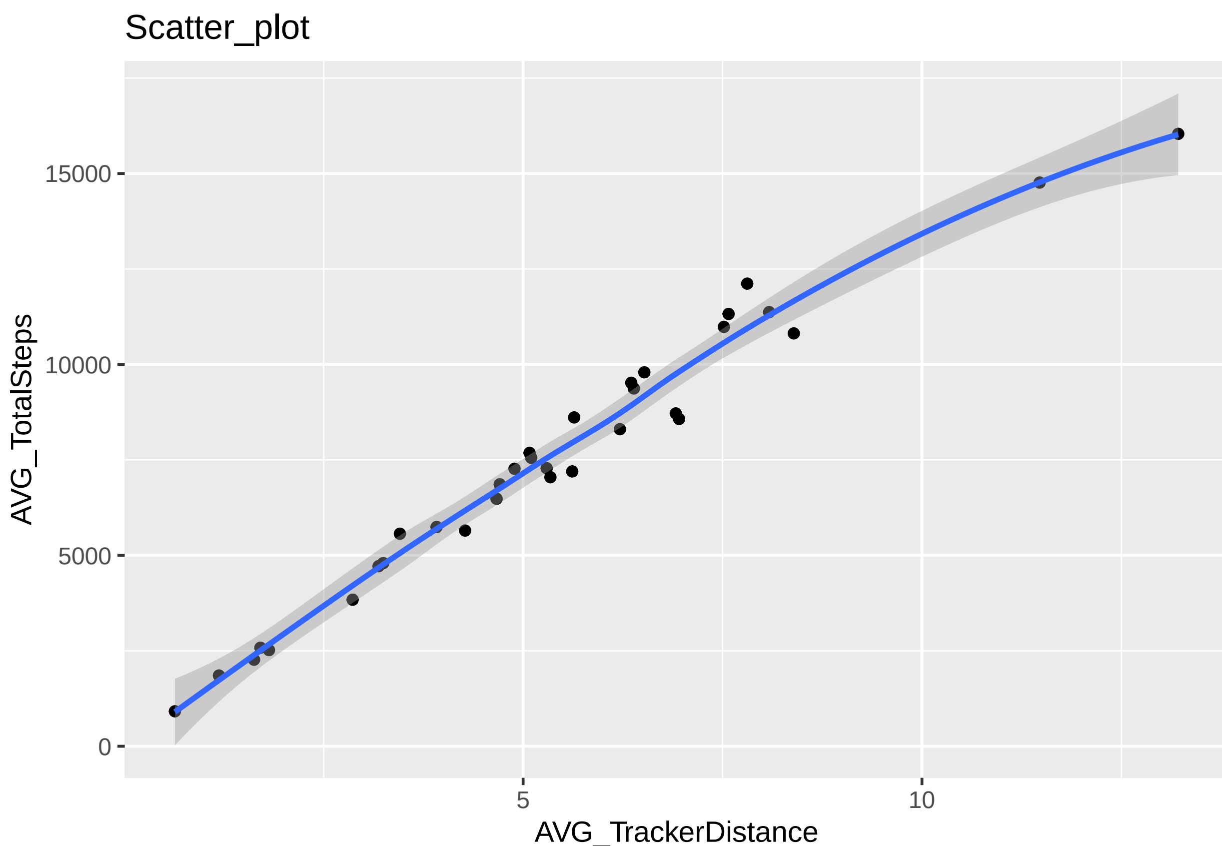
The trend in this scatter plot shows that as **Average Activity** increases, **average Minutes Asleep** decreases. This relationship is an **inverse relationship**, meaning that as one variable increases, the other decreases. The graph shows that people who have higher levels of physical activity tend to sleep less than those who have lower levels of physical activity. This could be because active people have more energy, need less rest, are more occupied or have different lifestyles than less active people.

However, this graph does not imply causation, only correlation. There could be other factors that affect both variables, such as age, health, or environment.

To check relationship between Tracker Distance and Total Steps

```r
ggplot(data=FitbaseData_Calculated, aes(x=AVG_TrackerDistance, y=AVG_TotalSteps,))+
  geom_point() +
  labs(title = "Scatter_plot", x = "AVG_TrackerDistance", y = "AVG_TotalSteps")+
  geom_smooth()
```

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

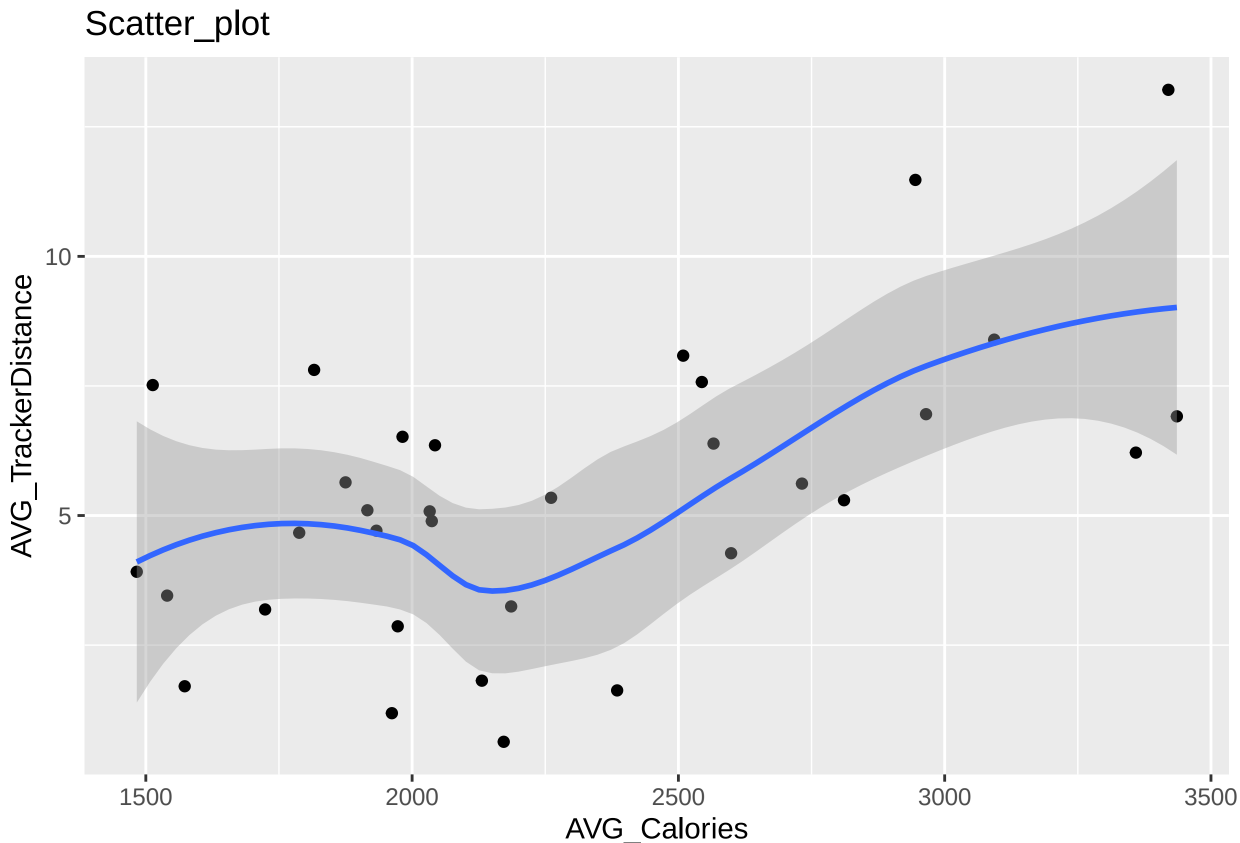


The trend in this scatter plot shows that as the **average tracker distance** increases, the **average total steps** also increases. This suggests that there is a **positive relationship** between the two variables. In other words, the more distance a person covers with their tracker, the more steps they are likely to take. This could imply that people who use trackers are more motivated to walk or exercise more.

To check relationship between Calories and Activity

```r
ggplot(data=FitbaseData_Calculated, aes(x=AVG_Calories, y=AVG_TrackerDistance,))+
  geom_point() +
  labs(title = "Scatter_plot", x = "AVG_Calories", y = "AVG_TrackerDistance")+
  geom_smooth()
```

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



The above scatter plot shows the relationship between average calories burned and average tracker distance. The trend in the scatter plot is **positive**, meaning that as the average tracker distance increases, the average calories burned also increases. The relationship between the two variables is **linear**, meaning that they change at a constant rate. The points are **fairly close** to the line of best fit, indicating a **strong** correlation between the two variables. This suggests that there is a **direct** and **proportional** relationship between average calories burned and average tracker distance, and that the fitness tracker app is **accurate** in measuring both variables.

Conclusion

Analyzing trends in Fitbit fitness tracker data offers valuable insights applicable to Bellabeat's customers:

The data suggests a correlation between increased steps and reduced sedentary time, indicating that Bellabeat's products could encourage a more active lifestyle by minimizing sedentary behavior. Additionally, the inverse relationship between physical activity and sleep duration implies that Bellabeat's offerings may cater to individuals with active lifestyles, guiding potential adjustments in sleep-related features or recommendations.

Furthermore, the positive correlation between tracker usage and physical activity emphasizes the motivational aspect of Bellabeat's products, suggesting they inspire users to be more active consistently. The positive relationship between tracker distance and calories burned indicates that Bellabeat's products accurately measures and supports users in achieving fitness goals, providing a comprehensive tool for managing and monitoring calorie expenditure.

In summary, these insights will enable Bellabeat to customize communication with users, improve product features, and offer recommendations thereby enhancing user engagement and satisfaction. By promoting an active lifestyle, addressing sleep patterns, emphasizing tracker usage, and highlighting accurate fitness metric measurements, Bellabeat can effectively support its customers' holistic approach to health and well-being.

Recommendation

Bellabeat, as a company, can leverage these insights to tailor its product features and marketing strategies. Based on the analysis of Fitbit fitness tracker data, the following recommendations are made for Bellabeat:

1. **Promote Active Lifestyle:** Emphasize products as catalysts for an active lifestyle, focusing on features that encourage increased steps and reduced sedentary time.
2. **Tailor Sleep Support:** Acknowledge the inverse relationship between physical activity and sleep duration. Consider tailoring sleep-related features or recommendations for users with active lifestyles.
3. **Highlight Tracker Motivation:** Leverage the positive correlation between tracker usage and physical activity. Emphasize how Bellabeat's products motivate users to maintain consistent physical activity through regular tracker use.
4. **Emphasize Calorie Management:** Capitalize on the positive relationship between tracker distance and calories burned. Promote Bellabeat's fitness tracker app as an accurate tool for managing and monitoring calorie expenditure.
5. **Promote Stress Management:** Integrate stress-related notifications in Bellabeat's Wellness watch, guiding users on optimal break times for enhanced well-being and productivity. Encourage a healthy work-life balance through personalized stress-aware notifications. Incorporating gamification elements, such as challenges and rewards, can further motivate users to manage stress effectively and maintain a balanced lifestyle.
6. **Encourage Hydration:** Implement hydration-related reminders in the Time product using Bellabeat's Spring product for drinking water. Notifying users to maintain adequate water intake for improved health and vitality most especially during intense activities. This supports a healthier lifestyle, with timely hydration prompts tailored to individual needs.
7. **Optimizing Female Health Features: Proactive Notifications** Bellabeat being a female-focused company can enhance female health features with proactive notifications for menstruation, ovulation, and mood fluctuations using the **Time wellness watch** in addition to the Bellabeat app by;
   - Emphasizing on real-time, personalized notifications for menstrual cycles, ovulation, and mood. Position the product as a comprehensive health companion that actively supports and informs users about their unique female health journey.
   - Communicate how Bellabeat empowers women with informed decisions about their health.Share user testimonials or stories that showcase how the inclusion of proactive notifications has positively impacted their wellness journeys.
   - Provide content educating users on the significance of notifications for proactive health.
   - Foster a supportive community for women to share experiences and tips. Facilitate discussions on social platforms or within the Bellabeat community to create a supportive space for women to exchange insights and tips, illustrating instances where timely notifications contributed to their improved health management and overall well-being.

By incorporating these recommendations, Bellabeat can align its marketing strategy and product features with the observed trends, enhancing user engagement and satisfaction among its customers.This will position Bellabeat as a brand that not only provides fitness tracking but also actively supports users in adopting a healthier and more balanced lifestyle. Tailoring messages to resonate with these trends can enhance user engagement and attract a wider audience seeking holistic health solutions.

While these trends offer valuable insights, it's crucial to acknowledge that correlation does not imply causation. Bellabeat may consider further research to understand the nuanced factors influencing these trends and refine its offerings accordingly.