Bellabeat: Google Data Analytics Case Study

Rebecca Gordon

2024-06-22

Bellabeat Case Study

Introduction

Purpose: This case study is an analysis of smart device data in order to identify current trends and insights that can be applied to Bellabeat's products for the purpose of marketing and increased customer satisfaction.

Company Summary: Bellabeat is a women's wellness company that creates innovative health and wellness products for women. Their high-tech and stylish devices collect bio-metric and lifestyle data to help women track their cycles, as well as their health and fitness goals.

Product Summary: The Bellabeat product that will be the focus of this case study is the Leaf. The Leaf has a simple, stylish design that is customizable to be worn as a bracelet, necklace, or clip. It allows women to achieve their wellness goals by tracking their activity, menstrual cycle, mindful practices, and sleep habits.

Tools: I will be using r in Rstudio to clean, analyze, and visualize the data in this case study.

ASK

Guiding Questions

- What is the problem you are trying to solve?
- How can your insights drive business decisions?

Business Task

The business task at hand is to identify winning marketing strategies for Bellabeat products, based on trends in current smart device data.

Shareholders

- Urška Sršen: Bellabeat's co-founder and Chief Creative Officer
- Sando Mur: Mathematician and Bellabeat's co-founder
- Bellabeat marketing analytics team

PREPARE

About the Data

This study will utilize personal tracker data gathered on Fitbit users as represented in the dataset Fitbit Fitness Tracker data (CC0:Public domain, source:https://zenodo.org/record/53894#.X9oeh3Uzaao, Owner: Mobius).

Because of concerns that the Fitbit Fitness Tracker data is not as complete or current as I would like, this study will also utilize a subset of PMData sports logging data (Attribution 4.0 International (CC BY 4.0), https://dl.acm.org/doi/10.1145/3339825.3394926) prepared by Kaggle user Salah as noted here. This data has demographic information including gender which may be especially interesting to this case study.

In both cases data was downloaded from Kaggle and stored securely on personal hard drive.

Installing and loading packages

Packages used in course of cleaning and analysis:

```
install.packages("tidyverse", repos = "https://mirror.las.iastate.edu/CRAN/")
```

```
## Installing package into 'C:/Users/Rebecca/AppData/Local/R/win-library/4.4'
## (as 'lib' is unspecified)
```

```
## package 'tidyverse' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Rebecca\AppData\Local\Temp\Rtmpiuq4Et\downloaded_packages
```

```
install.packages("here", repos = "https://mirror.las.iastate.edu/CRAN/")
```

```
## Installing package into 'C:/Users/Rebecca/AppData/Local/R/win-library/4.4'
## (as 'lib' is unspecified)
```

```
## package 'here' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Rebecca\AppData\Local\Temp\Rtmpiuq4Et\downloaded_packages
```

```
install.packages("skimr", repos = "https://mirror.las.iastate.edu/CRAN/")
```

```
## Installing package into 'C:/Users/Rebecca/AppData/Local/R/win-library/4.4'
## (as 'lib' is unspecified)
```

```
## package 'skimr' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Rebecca\AppData\Local\Temp\Rtmpiuq4Et\downloaded_packages
```

```
install.packages("janitor", repos = "https://mirror.las.iastate.edu/CRAN/")
```

```
## Installing package into 'C:/Users/Rebecca/AppData/Local/R/win-library/4.4'
## (as 'lib' is unspecified)

## package 'janitor' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Rebecca\AppData\Local\Temp\Rtmpiuq4Et\downloaded_packages
```

Loading packages:

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.4.1

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## 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
```

```
library(here)
```

```
## Warning: package 'here' was built under R version 4.4.1

## here() starts at C:/Users/Rebecca/Documents/Google Course/Project_names/Bellabeat_case_study
```

```
library(skimr)
```

```
## Warning: package 'skimr' was built under R version 4.4.1
```

```
library(janitor)
```

```
## Warning: package 'janitor' was built under R version 4.4.1

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

```r
library(dplyr)
```

Import data

Importing daily activty and heartrate data from Fitbit fitness tracker data:

```r
directory <-"C:/Users/Rebecca/Documents/Google Course/Project_names/Bellabeat_case_study/Primary data/Fi
file_path <- file.path(directory, "dailyActivity_merged.csv")
daily_activity <- read_csv(paste0(file_path))
```

```
## Rows: 457 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.
```

```r
directory <-"C:/Users/Rebecca/Documents/Google Course/Project_names/Bellabeat_case_study/Primary data/Fi
file_path <- file.path(directory, "dailyActivity_merged_2.csv")
daily_activity_2 <- read_csv(paste0(file_path))
```

```
## 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.
```

```r
directory <-"C:/Users/Rebecca/Documents/Google Course/Project_names/Bellabeat_case_study/Primary data/Fi
file_path <- file.path(directory, "heartrate_seconds_merged.csv")
heartrate_sec <- read_csv(paste0(file_path))
```

```
## Rows: 1154681 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): Time
## dbl (2): Id, Value
##
## 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.
```

```r
directory <-"C:/Users/Rebecca/Documents/Google Course/Project_names/Bellabeat_case_study/Primary data/Fi
file_path <- file.path(directory, "heartrate_seconds_merged_2.csv")
heartrate_sec_2 <- read_csv(paste0(file_path))
```

```
## Rows: 2483658 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): Time
## dbl (2): Id, Value
##
## 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.
```

Importing daily activity data from PMData:

```
directory <-"C:/Users/Rebecca/Documents/Google Course/Project_names/Bellabeat_case_study/Supplimentary d
file_path <- file.path(directory, "pmdata_daily_activity_merged.csv")
pm_daily_activity <- read_csv(paste0(file_path))
```

```
## Rows: 2194 Columns: 17
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (4): date_time, day, participant_id, gender
## dbl (13): age, height, steps, distance_km, calories, very_active_minutes, mo...
##
## 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.
```

PROCESS

I will be using r and Rmarkdown files in RStudio to clean, process, analyze, and visualize the data for this case study.

Data cleaning process:

I have detailed the main steps of my cleaning process separated by data source in this document.

Fitbit Fitness Tracker Data Take a quick look at data using the head function:

```
head(daily_activity)
```

```
## # A tibble: 6 x 15
##           Id ActivityDate TotalSteps TotalDistance TrackerDistance
##        <dbl> <chr>             <dbl>         <dbl>           <dbl>
## 1 1503960366 3/25/2016         11004          7.11            7.11
## 2 1503960366 3/26/2016         17609         11.6            11.6
## 3 1503960366 3/27/2016         12736          8.53            8.53
## 4 1503960366 3/28/2016         13231          8.93            8.93
## 5 1503960366 3/29/2016         12041          7.85            7.85
## 6 1503960366 3/30/2016         10970          7.16            7.16
## # i 10 more variables: LoggedActivitiesDistance <dbl>,
## #   VeryActiveDistance <dbl>, ModeratelyActiveDistance <dbl>,
## #   LightActiveDistance <dbl>, SedentaryActiveDistance <dbl>,
## #   VeryActiveMinutes <dbl>, FairlyActiveMinutes <dbl>,
## #   LightlyActiveMinutes <dbl>, SedentaryMinutes <dbl>, Calories <dbl>
```

```
head(daily_activity_2)
```

```
## # 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>
```

```
head(heartrate_sec)
```

```
## # A tibble: 6 x 3
##           Id Time                 Value
##        <dbl> <chr>                <dbl>
## 1 2022484408 4/1/2016 7:54:00 AM    93
## 2 2022484408 4/1/2016 7:54:05 AM    91
## 3 2022484408 4/1/2016 7:54:10 AM    96
## 4 2022484408 4/1/2016 7:54:15 AM    98
## 5 2022484408 4/1/2016 7:54:20 AM   100
## 6 2022484408 4/1/2016 7:54:25 AM   101
```

```
head(heartrate_sec_2)
```

```
## # A tibble: 6 x 3
##           Id Time                  Value
##        <dbl> <chr>                 <dbl>
## 1 2022484408 4/12/2016 7:21:00 AM    97
## 2 2022484408 4/12/2016 7:21:05 AM   102
## 3 2022484408 4/12/2016 7:21:10 AM   105
## 4 2022484408 4/12/2016 7:21:20 AM   103
## 5 2022484408 4/12/2016 7:21:25 AM   101
## 6 2022484408 4/12/2016 7:22:05 AM    95
```

Check the column names for consistent naming using the colnames function:

```
colnames(daily_activity)
```

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

```
colnames(daily_activity_2)
```

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

```
colnames(heartrate_sec)
```

```
## [1] "Id"    "Time"  "Value"
```

```
colnames(heartrate_sec_2)
```

```
## [1] "Id"    "Time"  "Value"
```

Clean column names for ease and consistency of use using the clean_names function:

```
daily_activity <- clean_names(daily_activity)
daily_activity_2 <- clean_names(daily_activity_2)
heartrate_sec <- clean_names(heartrate_sec)
heartrate_sec_2 <- clean_names(heartrate_sec_2)
```

If there are nulls in the numeric columns, means can not be calculated. Use colMeans function to verify that there are no nulls in all numeric columns:

```
colMeans(daily_activity[sapply(daily_activity, is.numeric)])
```

```
##                          id                 total_steps
##                4.628595e+09                6.546562e+03
##              total_distance             tracker_distance
##                4.663523e+00                4.609847e+00
## logged_activities_distance       very_active_distance
##                1.794274e-01                1.180897e+00
## moderately_active_distance      light_active_distance
##                4.786433e-01                2.890197e+00
##  sedentary_active_distance        very_active_minutes
##                1.903720e-03                1.662363e+01
##      fairly_active_minutes     lightly_active_minutes
##                1.307002e+01                1.700700e+02
##          sedentary_minutes                    calories
##                9.952823e+02                2.189453e+03
```

```
colMeans(daily_activity_2[sapply(daily_activity_2, is.numeric)])
```

```
##                          id                 total_steps
##                4.855407e+09                7.637911e+03
##              total_distance             tracker_distance
##                5.489702e+00                5.475351e+00
## logged_activities_distance       very_active_distance
##                1.081709e-01                1.502681e+00
## moderately_active_distance      light_active_distance
##                5.675426e-01                3.340819e+00
##   sedentary_active_distance        very_active_minutes
##                1.606383e-03                2.116489e+01
##      fairly_active_minutes     lightly_active_minutes
##                1.356489e+01                1.928128e+02
##           sedentary_minutes                    calories
##                9.912106e+02                2.303610e+03
```

```
colMeans(heartrate_sec[sapply(heartrate_sec, is.numeric)])
```

```
##           id        value
## 5.352122e+09 7.975687e+01
```

```
colMeans(heartrate_sec_2[sapply(heartrate_sec_2, is.numeric)])
```

```
##           id        value
## 5.513765e+09 7.732842e+01
```

Id columns show as numeric rather than character. Convert data type:

```
daily_activity$id <- as.character(daily_activity$id)
daily_activity_2$id <- as.character(daily_activity_2$id)
heartrate_sec$id <- as.character(heartrate_sec$id)
heartrate_sec_2$id <- as.character(heartrate_sec_2$id)
```

Use n_distinct on relevant non numeric columns to check for misspellings or errors by verifying the expected number of options are returned for columns like id:

```
n_distinct(daily_activity$id)
```

```
## [1] 35
```

```
n_distinct(daily_activity$activity_date)
```

```
## [1] 32
```

```
n_distinct(daily_activity_2$id)
```

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

```
n_distinct(daily_activity_2$activity_date)
```

```
## [1] 31
```

Counts were consistent with expected values for each relevant category.

Check for duplicates: None found but results of code are not displayed here for ease of reading.

Assessment: After following these steps there were:

- No special characters noted.
- No incorrect data types noted.
- No misspellings noted under column names or when reviewing data.
- No nulls in numeric columns
- No white space or other issues noted.
- No duplicates noted.

There are a lot of zero values in the daily activity data files. It is unclear if all of them are true zero measurements or lack of participation. Overall, the data appears to now be clean and usable, but there are some concerns that it is not complete (due to lots of zero values) or as current as it could be.

PMData set Take a quick look at data using the head function:

```
head(pm_daily_activity)
```

```
## # A tibble: 6 x 17
##   date_time day    participant_id gender   age height steps distance_km calories
##   <chr>     <chr>  <chr>          <chr>  <dbl>  <dbl> <dbl>       <dbl>    <dbl>
## 1 11/1/2019 Friday p01            male      48    195 17873       14.4     4010.
## 2 11/2/2019 Satur~ p01            male      48    195 13118       10.6     3551.
## 3 11/3/2019 Sunday p01            male      48    195 14312       11.5     3732.
## 4 11/4/2019 Monday p01            male      48    195 10970        8.86    3391.
## 5 11/5/2019 Tuesd~ p01            male      48    195 16186       13.7     3896.
## 6 11/6/2019 Wedne~ p01            male      48    195  8189        6.52    3050.
## # i 8 more variables: very_active_minutes <dbl>,
## #   moderately_active_minutes <dbl>, lightly_active_minutes <dbl>,
## #   sedentary_minutes <dbl>, minutes_asleep <dbl>, minutes_awake <dbl>,
## #   minutes_after_wakeup <dbl>, time_in_bed_minutes <dbl>
```

Check the column names for consistent naming using the colnames function:

```
colnames(pm_daily_activity)
```

```
##  [1] "date_time"                "day"
##  [3] "participant_id"           "gender"
##  [5] "age"                      "height"
##  [7] "steps"                    "distance_km"
##  [9] "calories"                 "very_active_minutes"
## [11] "moderately_active_minutes" "lightly_active_minutes"
## [13] "sedentary_minutes"        "minutes_asleep"
## [15] "minutes_awake"            "minutes_after_wakeup"
## [17] "time_in_bed_minutes"
```

Change moderately_active_minutes to fairly_active_minutes for consistency across datasets:

```
pm_daily_activity <- rename(pm_daily_activity, fairly_active_minutes=moderately_active_minutes)
```

If there are nulls in the numeric columns, means can not be calculated. Use colMeans function to verify that there are no nulls in all numeric columns:

```
colMeans(pm_daily_activity[sapply(pm_daily_activity, is.numeric)])
```

```
##                    age                 height                  steps
##           3.511623e+01           1.796773e+02           1.028112e+04
##            distance_km               calories    very_active_minutes
##           7.996799e+00           2.887005e+03           4.245397e+01
##  fairly_active_minutes lightly_active_minutes      sedentary_minutes
##           2.079262e+01           1.929448e+02           7.889234e+02
##         minutes_asleep          minutes_awake   minutes_after_wakeup
##           3.441012e+02           4.997903e+01           6.750228e-01
##    time_in_bed_minutes
##           3.943345e+02
```

Use n_distinct on relevant non numeric columns to check for misspellings or errors by verifying the expected number of options are returned for columns like gender or day:

```
n_distinct(pm_daily_activity$day)
```

```
## [1] 7
```

```
n_distinct(pm_daily_activity$participant_id)
```

```
## [1] 15
```

```
n_distinct(pm_daily_activity$gender)
```

```
## [1] 2
```

Counts were consistent with appropriate number of options for each category.

Check for duplicates: None found but results of code are not displayed here for ease of reading.

Assessment: After following these steps there were:

- No special characters noted.
- No incorrect data types noted.
- No misspellings noted under column names or when reviewing data.
- No nulls in numeric columns
- No white space or other issues noted.
- No duplicates noted.

This data now appears to be clean and usable.

Overview

I have ensured that my data is clean and usable by checking to ensure it is:

- Free of duplicate rows/values
- Free of errors or misspellings
- Relevant to the business task
- Free of special characters
- Of appropriate data types
- Free of null values

I have also kept documentation of the cleaning process. I am combining the two Fitbit datasets so that analysis can be carried out on the entire two-month time period at once.

```
Combined_data_daily <- daily_activity %>%
  bind_rows(daily_activity_2)
```

```
Combined_heartrate <- heartrate_sec %>%
  bind_rows(heartrate_sec_2)
```

ANALYZE

In this section I will document the pertinent organization, aggregation, and analysis of the data performed for this case study. I have separated this analysis by data source, but some overlap and comparison will occur.

Identifying Trends in Fitbit data

Here is a summary of potentially relevant observations and trends noted through analysis of the Fitbit data.

Looking at data by ID: There are 35 distinct participants in the daily activity data:

```
Id_list <- Combined_data_daily %>%
           count(id) %>%
           group_by(id)
```

```
tibble(Id_list)
```

```
## # A tibble: 35 x 2
##    id             n
##    <chr>      <int>
##  1 1503960366    50
##  2 1624580081    50
##  3 1644430081    40
##  4 1844505072    43
##  5 1927972279    43
##  6 2022484408    43
##  7 2026352035    43
##  8 2320127002    43
##  9 2347167796    33
## 10 2873212765    43
## # i 25 more rows
```

There are 15 distinct participants in the heartrate data:

```
Id_list2 <- Combined_heartrate %>%
           count(id) %>%
           group_by(id)
```

```
tibble(Id_list2)
```

```
## # A tibble: 15 x 2
##    id              n
##    <chr>       <int>
##  1 2022484408 210587
##  2 2026352035   2929
##  3 2347167796 273487
##  4 4020332650 569255
##  5 4388161847 249748
##  6 4558609924 261507
##  7 5553957443 352971
##  8 5577150313 336209
##  9 6117666160 212565
## 10 6391747486   3747
## 11 6775888955  67871
## 12 6962181067 392201
## 13 7007744171 198378
## 14 8792009665 192928
## 15 8877689391 313956
```

Looking at mean values of interesting categories Provides the average heartrate values per participant:

```
mean_heartrate_id <- aggregate(value ~ id, data=Combined_heartrate, FUN=mean)

tibble(mean_heartrate_id)
```

```
## # A tibble: 15 x 2
##    id         value
##    <chr>      <dbl>
##  1 2022484408  80.6
##  2 2026352035  89.6
##  3 2347167796  76.4
##  4 4020332650  82.1
##  5 4388161847  66.1
##  6 4558609924  81.1
##  7 5553957443  68.9
##  8 5577150313  68.4
##  9 6117666160  83.7
## 10 6391747486  84.1
## 11 6775888955  95.0
## 12 6962181067  78.7
## 13 7007744171  90.7
## 14 8792009665  73.9
## 15 8877689391  84.7
```

Provides the average sedentary minute values per participant:

```
mean_sedmin_id <- aggregate(sedentary_minutes ~ id, data=Combined_data_daily, FUN=mean)

tibble(mean_sedmin_id)
```

```
## # A tibble: 35 x 2
##    id         sedentary_minutes
##    <chr>                  <dbl>
##  1 1503960366              834.
##  2 1624580081             1266.
##  3 1644430081             1130.
##  4 1844505072             1159.
##  5 1927972279             1216.
##  6 2022484408             1098.
##  7 2026352035              681.
##  8 2320127002             1228.
##  9 2347167796              686.
## 10 2873212765             1108.
## # i 25 more rows
```

Provides the average very active minute values per participant:

```
mean_actmin_id <- aggregate(very_active_minutes ~ id, data=Combined_data_daily, FUN=mean)

tibble(mean_actmin_id)
```

```
## # A tibble: 35 x 2
##    id         very_active_minutes
##    <chr>                    <dbl>
##  1 1503960366               37.6
##  2 1624580081                5.66
##  3 1644430081               10.9
##  4 1844505072                0.302
##  5 1927972279                0.953
##  6 2022484408               37.3
##  7 2026352035                0.0698
##  8 2320127002                1.23
##  9 2347167796               12.7
## 10 2873212765               11.6
## # i 25 more rows
```

This filtered the datasets so that they all show the same 15 participants about whom we have data for all three variables:

```
mean_sedmin_id_15 <- subset(mean_sedmin_id,id %in% Id_list2$id)
mean_actmin_id_15 <- subset(mean_actmin_id, id %in% Id_list2$id)
```

Now we have a combined table for the averages of all three variables per participant:

```
combined_data_15 <- mutate(mean_actmin_id_15, mean_heartrate = mean_heartrate_id$value)
combined_data_15 <- mutate(combined_data_15, mean_sedentary_min = mean_sedmin_id_15$sedentary_minutes)
combined_data_15 <- rename(combined_data_15, mean_very_active_min = very_active_minutes)
```

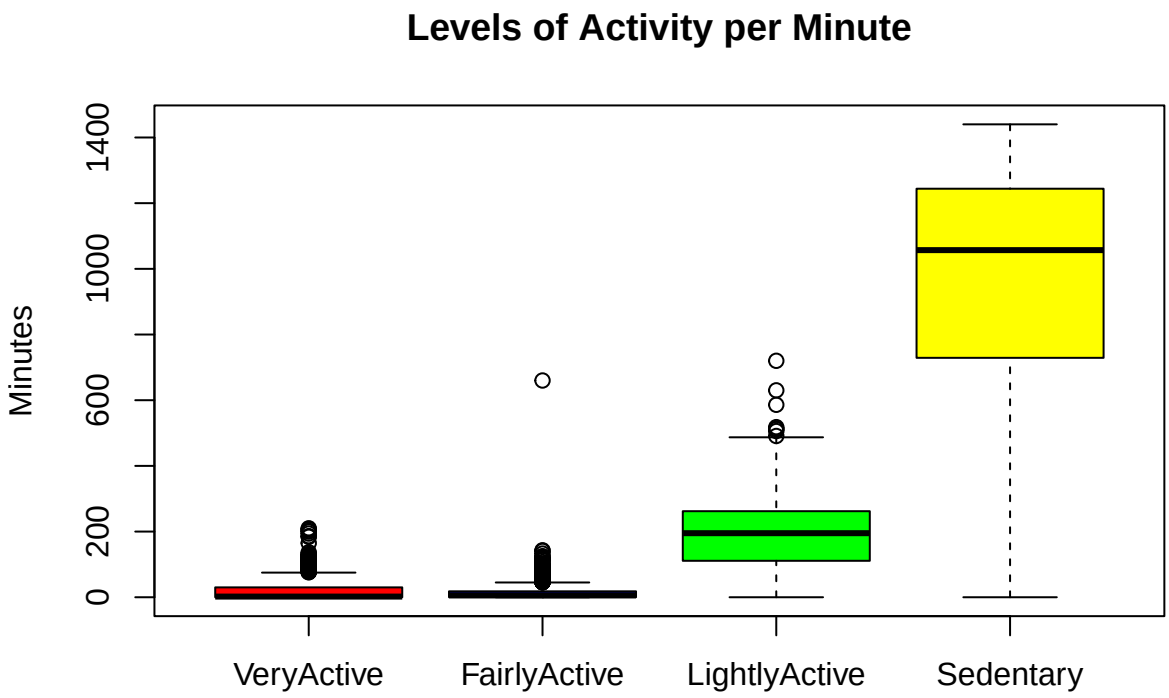
```
tibble(combined_data_15)
```

```
## # A tibble: 15 x 4
##    id         mean_very_active_min mean_heartrate mean_sedentary_min
##    <chr>                     <dbl>          <dbl>              <dbl>
##  1 2022484408              37.3             80.6              1098.
##  2 2026352035               0.0698          89.6               681.
##  3 2347167796              12.7             76.4               686.
##  4 4020332650               4.62            82.1              1159.
##  5 4388161847              18.4             66.1               949
##  6 4558609924               8.70            81.1              1091.
##  7 5553957443              23.5             68.9               651.
##  8 5577150313              85.9             68.4               731.
##  9 6117666160               1.29            83.7               814.
## 10 6391747486               5.11            84.1              1262.
## 11 6775888955              13.3             95.0              1227.
## 12 6962181067              26.7             78.7               645.
## 13 7007744171              35.3             90.7              1038.
## 14 8792009665               1.12            73.9              1012.
## 15 8877689391              66.3             84.7              1094.
```

Levels of Activity per minute Now that we have some summary data, I am interested in a comparison between the levels of activity per minute.

Here is a representation of that comparison:

```
boxplot(Combined_data_daily$very_active_minutes,
        Combined_data_daily$fairly_active_minutes,
        Combined_data_daily$lightly_active_minutes,
        Combined_data_daily$sedentary_minutes,
        main= "Levels of Activity per Minute",
        names= c("VeryActive", "FairlyActive", "LightlyActive", "Sedentary"),
        ylab= "Minutes",
        col = c("red", "blue", "green", "yellow"))
```



And the summary:

```
Combined_data_daily %>%
  select(very_active_minutes,
         fairly_active_minutes,
         lightly_active_minutes,
         sedentary_minutes) %>%
  summary()
```

```
##  very_active_minutes fairly_active_minutes lightly_active_minutes
##  Min.   :  0.00      Min.   :  0.0         Min.   :  0.0
##  1st Qu.:  0.00      1st Qu.:  0.0         1st Qu.:111.0
##  Median :  2.00      Median :  6.0         Median :195.0
##  Mean   : 19.68      Mean   : 13.4         Mean   :185.4
##  3rd Qu.: 30.00      3rd Qu.: 18.0         3rd Qu.:262.0
##  Max.   :210.00      Max.   :660.0         Max.   :720.0
##  sedentary_minutes
##  Min.   :   0.0
##  1st Qu.: 729.0
##  Median :1057.0
##  Mean   : 992.5
##  3rd Qu.:1244.0
##  Max.   :1440.0
```

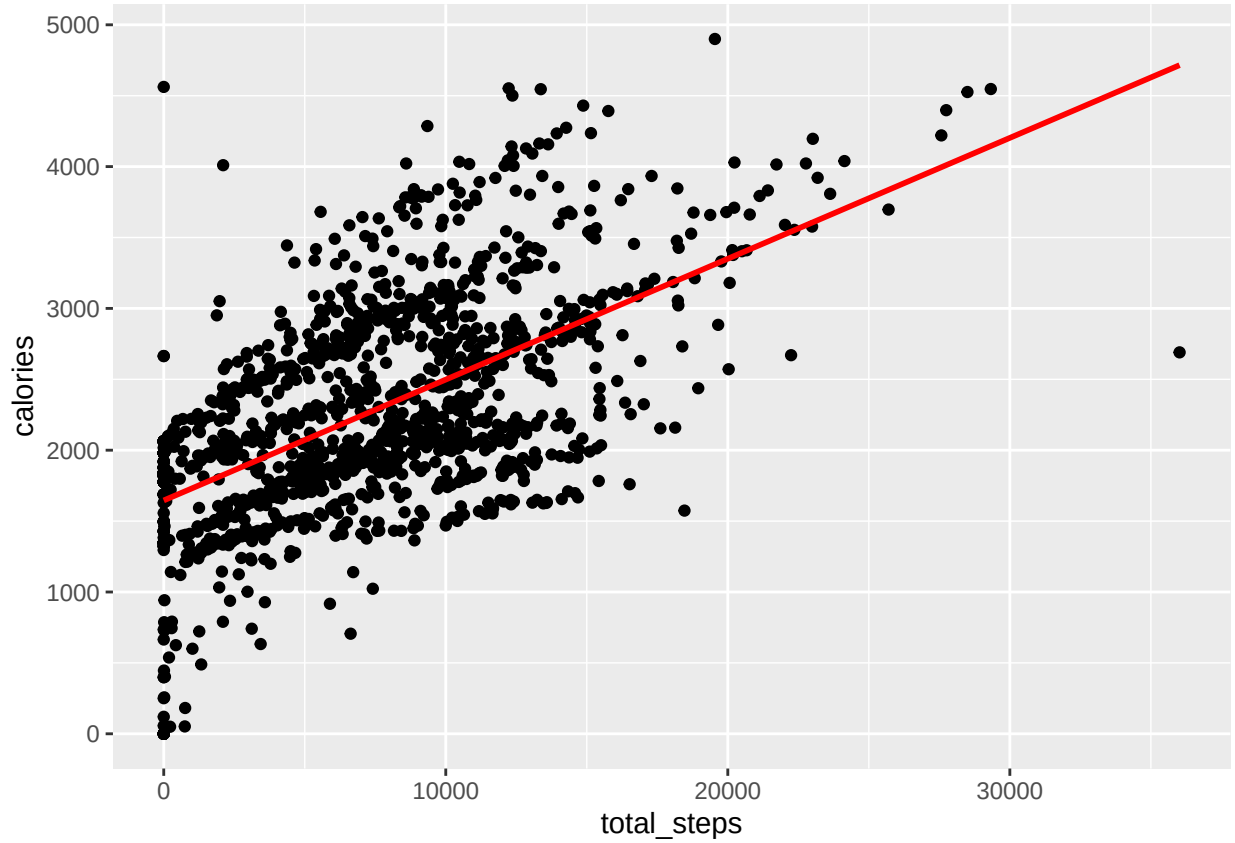
These clearly show that sedentary minutes has the highest values of all four level of activity per minute segments. The combined mean for sedentary minutes is 992.50 min (16.54 hr). The combined mean for

lightly active minutes is 185.4 (3.09 hr). The combined mean for fairly active minutes is 13.4 (.22 hr). The combined mean for very active minutes is 19.68 (.33 hr). This may indicate that there is a large opportunity gap in marketing for customers looking to evaluate and/or improve their level of activity.

Calories vs Total Steps This plot explores the relationship between calories and total steps:

```
ggplot(data=Combined_data_daily, aes(x=total_steps, y=calories)) + geom_point() +
  geom_smooth(method = lm, se=FALSE, col="red")
```

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



Here is the important evaluation information for this relationship:

```
model <- lm(calories ~ total_steps, data = Combined_data_daily)

summary(model)
```

```
##
## Call:
## lm(formula = calories ~ total_steps, data = Combined_data_daily)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2025.48  -385.14    -9.05   418.25  2916.25
```

```
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.646e+03  2.795e+01   58.88   <2e-16 ***
## total_steps 8.522e-02  3.121e-03   27.30   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 608.1 on 1395 degrees of freedom
## Multiple R-squared:  0.3483, Adjusted R-squared:  0.3478
## F-statistic: 745.5 on 1 and 1395 DF,  p-value: < 2.2e-16
```

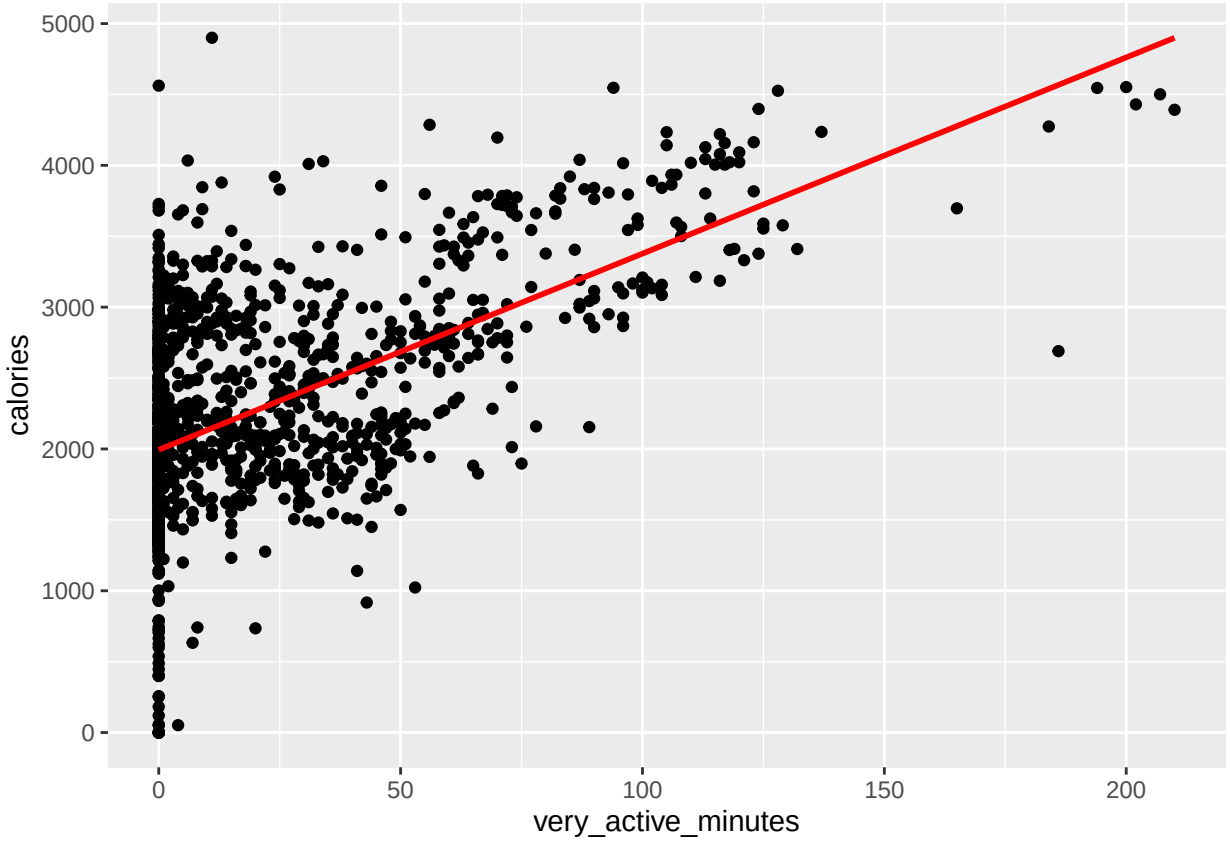
This gives us: Multiple R-squared: 0.3483, Adjusted R-squared: 0.3478, p-value: $< 2.2e\text{-}16$

This indicates that the correlation is weak to moderate and likely to be statistically significant.

Calories vs Very Active Minutes This plot explores the relationship between calories and very active minutes:

```
ggplot(data=Combined_data_daily, aes(x=very_active_minutes, y=calories)) + geom_point() + geom_smooth(m
```

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



Here is the important evaluation information for this relationship:

```r
model8 <- lm(calories ~ very_active_minutes, data = Combined_data_daily)

summary(model8)
```

```
##
## Call:
## lm(formula = calories ~ very_active_minutes, data = Combined_data_daily)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1997.33  -380.23   -48.98   389.02  2753.82
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         1993.9811    19.2956  103.34   <2e-16 ***
## very_active_minutes   13.8361     0.5176   26.73   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 612.5 on 1395 degrees of freedom
## Multiple R-squared:  0.3388, Adjusted R-squared:  0.3383
## F-statistic: 714.7 on 1 and 1395 DF,  p-value: < 2.2e-16
```

This gives us: Multiple R-squared: 0.3388, Adjusted R-squared: 0.3383, p-value: $<$ 2.2e-16

This indicates that the correlation is weak to moderate and likely to be statistically significant.

Identifying and comparing trends in PMdata

Looking at data by ID: There are 15 distinct participants in the PMData:

```r
Id_list3 <- pm_daily_activity %>%
           count(participant_id) %>%
           group_by(participant_id)
```

```r
tibble(Id_list3)
```

```
## # A tibble: 15 x 2
##    participant_id     n
##    <chr>          <int>
##  1 p01              152
##  2 p02              152
##  3 p03              152
##  4 p04              152
##  5 p05              152
##  6 p06              152
##  7 p07              148
##  8 p08              143
##  9 p09              152
## 10 p10              148
## 11 p11              152
## 12 p13              152
```

```
## 13 p14               90
## 14 p15              145
## 15 p16              152
```

There are 3 female participants and 12 male participants:

```
Id_list4 <- pm_daily_activity %>%
          select(participant_id, "gender") %>%
            filter(gender == "female") %>%
             count(participant_id) %>%
            group_by(participant_id)

Id_list5 <- pm_daily_activity %>%
          select(participant_id, "gender") %>%
          filter(gender == "male") %>%
           count(participant_id) %>%
          group_by(participant_id)
```

```
tibble(Id_list4)
```

```
## # A tibble: 3 x 2
##   participant_id     n
##   <chr>          <int>
## 1 p04              152
## 2 p10              148
## 3 p11              152
```

```
tibble(Id_list5)
```

```
## # A tibble: 12 x 2
##    participant_id     n
##    <chr>          <int>
##  1 p01              152
##  2 p02              152
##  3 p03              152
##  4 p05              152
##  5 p06              152
##  6 p07              148
##  7 p08              143
##  8 p09              152
##  9 p13              152
## 10 p14               90
## 11 p15              145
## 12 p16              152
```

Id_list4 shows female participants: 4, 10, 11

Id_list5 shows male participants: 1, 2, 3, 5, 6, 7, 8, 9, 13, 14, 15, 16

Looking at mean values of interesting categories Provides the average total step values per participant:

```r
pm_mean_totalstep_id <- aggregate(steps ~ participant_id, data=pm_daily_activity, FUN=mean)

tibble(pm_mean_totalstep_id)
```

```
## # A tibble: 15 x 2
##    participant_id  steps
##    <chr>           <dbl>
##  1 p01            12607.
##  2 p02            15505.
##  3 p03             4301.
##  4 p04             9978.
##  5 p05            10790.
##  6 p06            13870.
##  7 p07            15685.
##  8 p08            15759.
##  9 p09             5704.
## 10 p10             9144.
## 11 p11             9678.
## 12 p13             5374.
## 13 p14            11664.
## 14 p15            13454.
## 15 p16             1851.
```

Provides the average calorie values per participant:

```r
pm_mean_cal_id <- aggregate(calories ~ participant_id, data=pm_daily_activity, FUN=mean)

tibble(pm_mean_cal_id)
```

```
## # A tibble: 15 x 2
##    participant_id calories
##    <chr>             <dbl>
##  1 p01               3607.
##  2 p02               3442.
##  3 p03               2512.
##  4 p04               2028.
##  5 p05               3385.
##  6 p06               3574.
##  7 p07               3185.
##  8 p08               3811.
##  9 p09               2732.
## 10 p10               2255.
## 11 p11               2455.
## 12 p13               2462.
## 13 p14               2643.
## 14 p15               3166.
## 15 p16               2008.
```

Provides the average very active minute values per participant:

```
pm_mean_veryactmin_id <- aggregate(very_active_minutes ~ participant_id, data=pm_daily_activity, FUN=mea

tibble(pm_mean_veryactmin_id)
```

```
## # A tibble: 15 x 2
##    participant_id very_active_minutes
##    <chr>                        <dbl>
##  1 p01                           48.6
##  2 p02                          107.
##  3 p03                           13.7
##  4 p04                           28.8
##  5 p05                           33.6
##  6 p06                           39.6
##  7 p07                           64.0
##  8 p08                           87.6
##  9 p09                           16.3
## 10 p10                           21.6
## 11 p11                           28.4
## 12 p13                           12.4
## 13 p14                           54.6
## 14 p15                           88.3
## 15 p16                            2.10
```

Provides the average sedentary minute values per participant:

```
pm_mean_sedmin_id <- aggregate(sedentary_minutes ~ participant_id, data=pm_daily_activity, FUN=mean)

tibble(pm_mean_sedmin_id)
```

```
## # A tibble: 15 x 2
##    participant_id sedentary_minutes
##    <chr>                      <dbl>
##  1 p01                         719.
##  2 p02                         727.
##  3 p03                        1108.
##  4 p04                         702.
##  5 p05                         816.
##  6 p06                         621.
##  7 p07                         618.
##  8 p08                         633.
##  9 p09                         822.
## 10 p10                         855.
## 11 p11                         773.
## 12 p13                        1127.
## 13 p14                         747.
## 14 p15                         638.
## 15 p16                         893.
```

Provides the average sleep minute values per participant:

```r
pm_mean_sleepmin_id <- aggregate(minutes_asleep ~ participant_id, data=pm_daily_activity, FUN=mean)

tibble(pm_mean_sleepmin_id)
```

```
## # A tibble: 15 x 2
##    participant_id minutes_asleep
##    <chr>                   <dbl>
##  1 p01                      350.
##  2 p02                      351.
##  3 p03                      190.
##  4 p04                      422.
##  5 p05                      290.
##  6 p06                      373.
##  7 p07                      418.
##  8 p08                      413.
##  9 p09                      382.
## 10 p10                      301.
## 11 p11                      337.
## 12 p13                      139.
## 13 p14                      390.
## 14 p15                      437.
## 15 p16                      395.
```

This combines the datasets showing averages:

```r
combined_data_pm <- mutate(pm_mean_totalstep_id, total_step = steps)
combined_data_pm <- mutate(combined_data_pm, sedentary_min = pm_mean_sedmin_id$sedentary_minutes)
combined_data_pm <- mutate(combined_data_pm, very_active_minutes = pm_mean_veryactmin_id$very_active_mi
combined_data_pm <- mutate(combined_data_pm, calories = pm_mean_cal_id$calories)
combined_data_pm <- mutate(combined_data_pm, minutes_asleep = pm_mean_sleepmin_id$minutes_asleep)
combined_data_pm <- select(combined_data_pm, -steps)
```

```r
tibble(combined_data_pm)
```

```
## # A tibble: 15 x 6
##    participant_id total_step sedentary_min very_active_minutes calories
##    <chr>               <dbl>         <dbl>               <dbl>    <dbl>
##  1 p01                12607.          719.               48.6     3607.
##  2 p02                15505.          727.              107.      3442.
##  3 p03                 4301.         1108.               13.7     2512.
##  4 p04                 9978.          702.               28.8     2028.
##  5 p05                10790.          816.               33.6     3385.
##  6 p06                13870.          621.               39.6     3574.
##  7 p07                15685.          618.               64.0     3185.
##  8 p08                15759.          633.               87.6     3811.
##  9 p09                 5704.          822.               16.3     2732.
## 10 p10                 9144.          855.               21.6     2255.
## 11 p11                 9678.          773.               28.4     2455.
## 12 p13                 5374.         1127.               12.4     2462.
## 13 p14                11664.          747.               54.6     2643.
## 14 p15                13454.          638.               88.3     3166.
## 15 p16                 1851.          893.                2.10    2008.
## # i 1 more variable: minutes_asleep <dbl>
```

Since we already know which participants are male or female we can add that information back in for comparison:

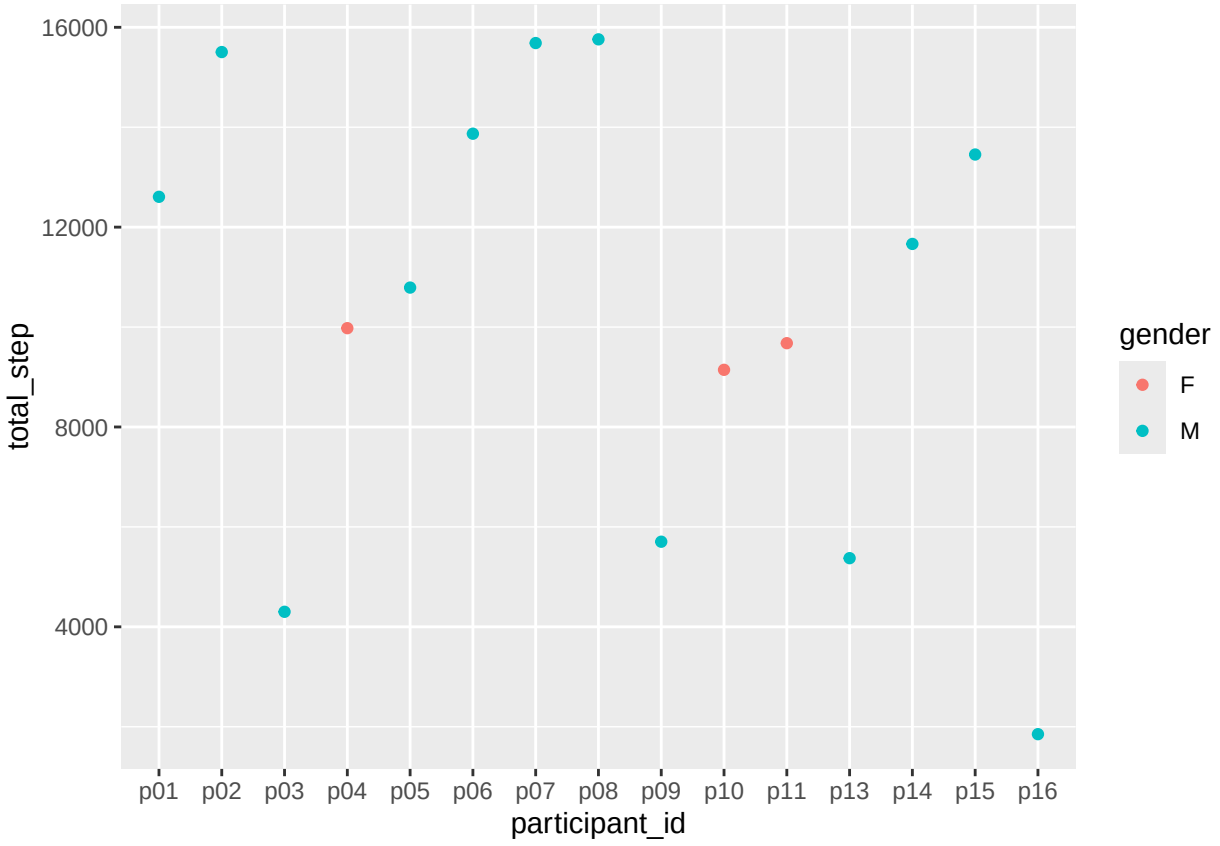
```
gender_pm <- c("M", "M", "M", "F", "M", "M", "M", "M", "M", "F", "F", "M", "M", "M", "M")
combined_data_pm <- mutate(combined_data_pm, "gender" = gender_pm)
```

```
tibble(combined_data_pm)
```

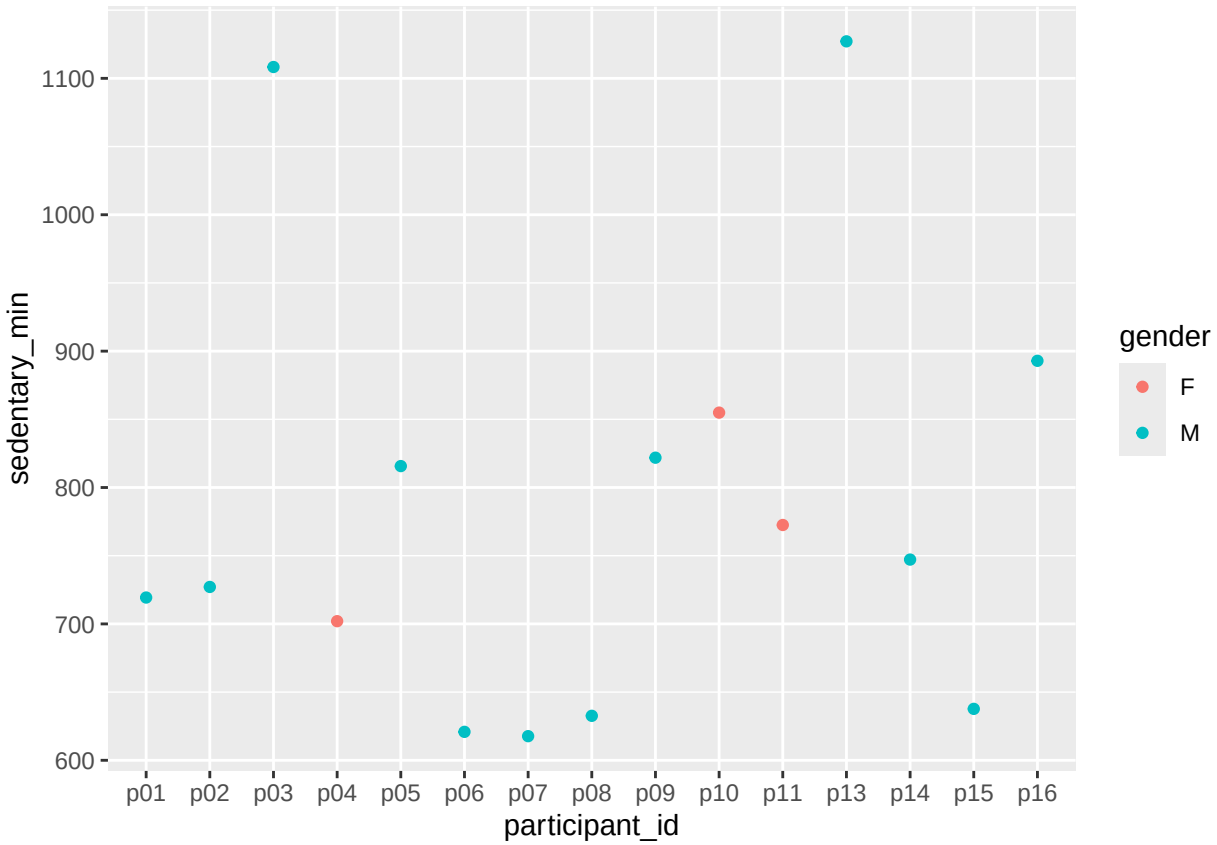
```
## # A tibble: 15 x 7
##    participant_id total_step sedentary_min very_active_minutes calories
##    <chr>               <dbl>         <dbl>                <dbl>    <dbl>
##  1 p01                12607.          719.                48.6     3607.
##  2 p02                15505.          727.               107.      3442.
##  3 p03                 4301.         1108.                13.7     2512.
##  4 p04                 9978.          702.                28.8     2028.
##  5 p05                10790.          816.                33.6     3385.
##  6 p06                13870.          621.                39.6     3574.
##  7 p07                15685.          618.                64.0     3185.
##  8 p08                15759.          633.                87.6     3811.
##  9 p09                 5704.          822.                16.3     2732.
## 10 p10                 9144.          855.                21.6     2255.
## 11 p11                 9678.          773.                28.4     2455.
## 12 p13                 5374.         1127.                12.4     2462.
## 13 p14                11664.          747.                54.6     2643.
## 14 p15                13454.          638.                88.3     3166.
## 15 p16                 1851.          893.                 2.10    2008.
## # i 2 more variables: minutes_asleep <dbl>, gender <chr>
```

Visualization of mean values comparing females to males:

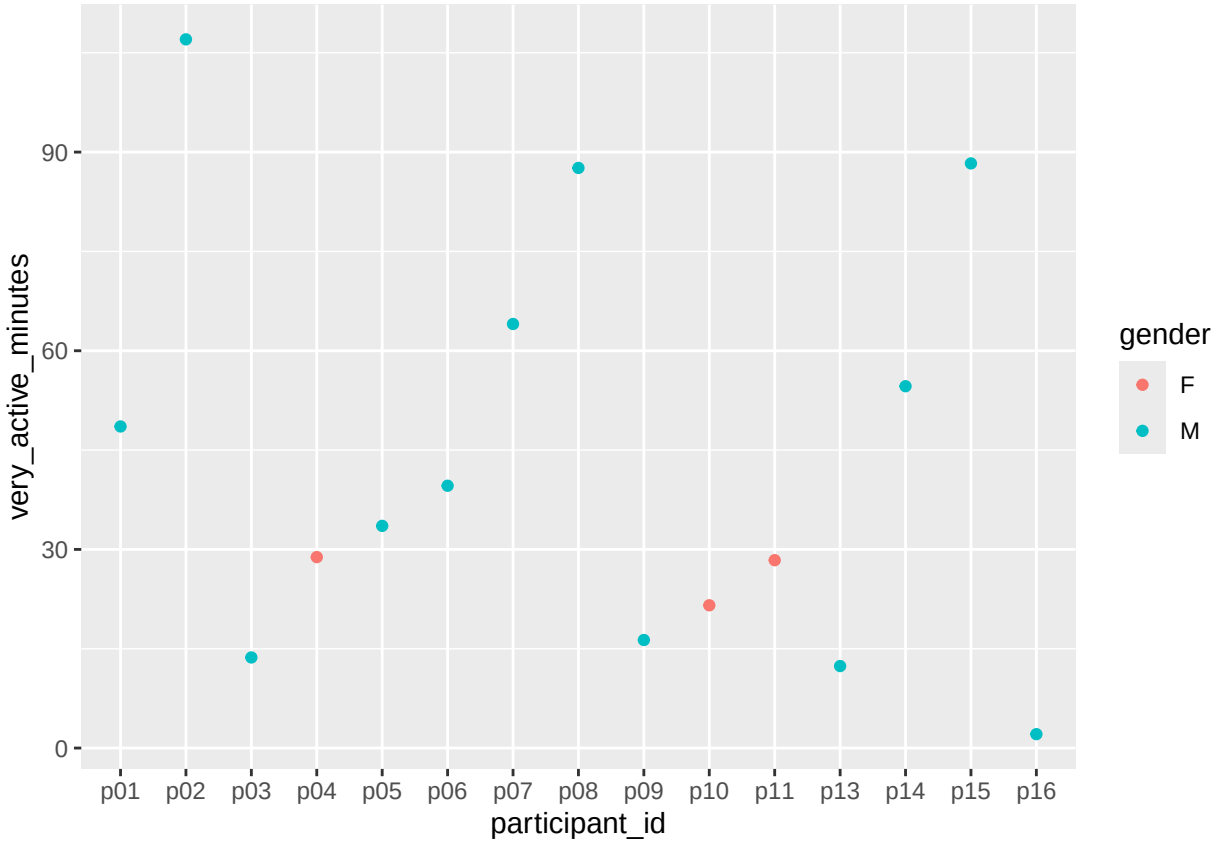
```
ggplot(data=combined_data_pm, aes(x=participant_id, y=total_step, colour = gender)) + geom_point()
```



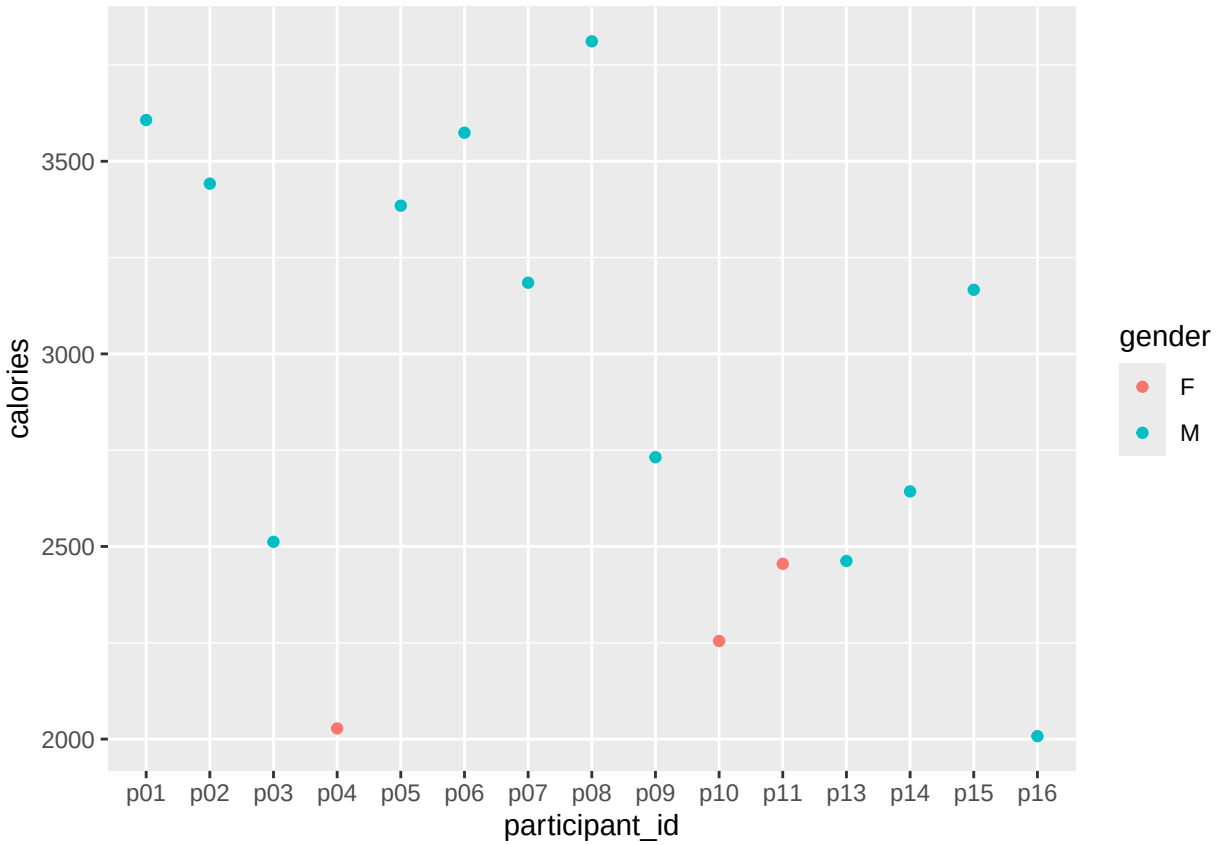

```
ggplot(data=combined_data_pm, aes(x=participant_id, y=sedentary_min, colour = gender)) + geom_point()
```



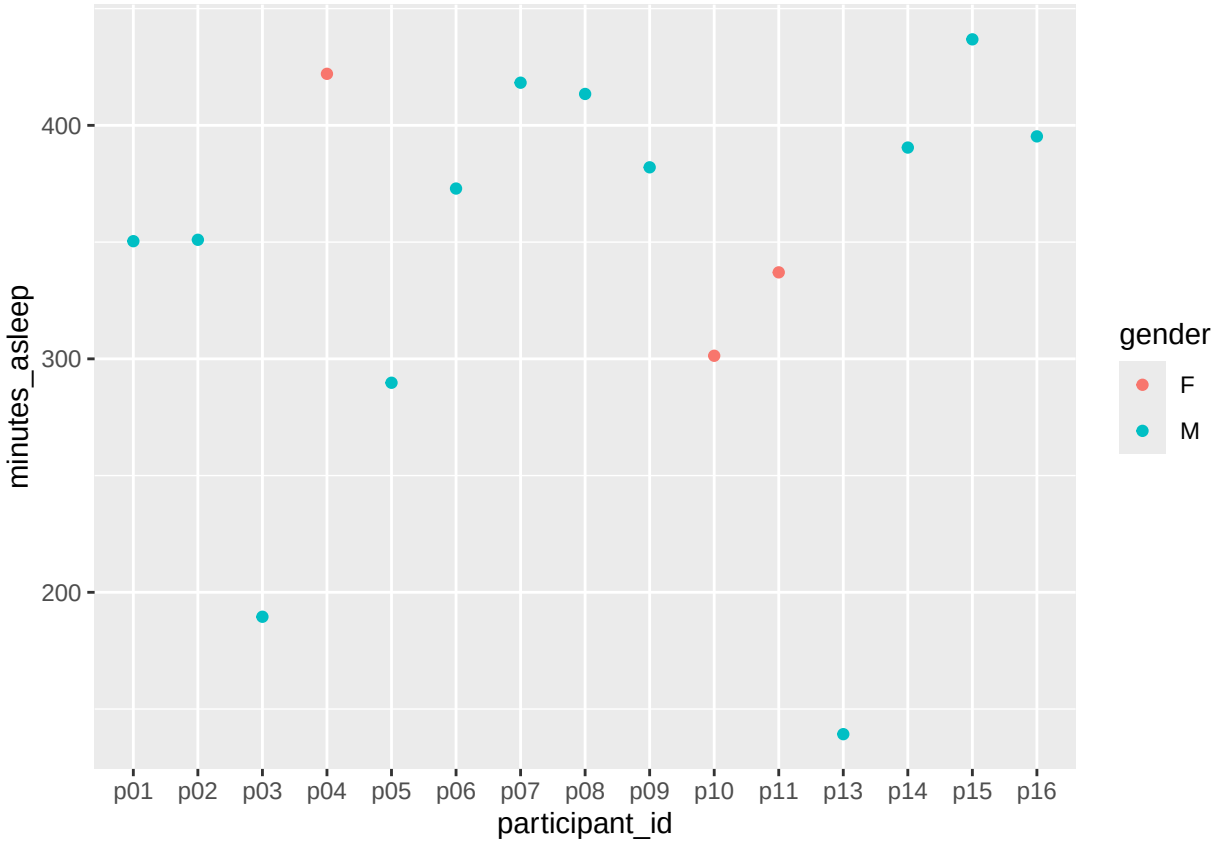

```
ggplot(data=combined_data_pm, aes(x=participant_id, y=very_active_minutes, colour = gender)) + geom_poi
```




```
ggplot(data=combined_data_pm, aes(x=participant_id, y=calories, colour = gender)) + geom_point()
```




```
ggplot(data=combined_data_pm, aes(x=participant_id, y=minutes_asleep, colour = gender)) + geom_point()
```

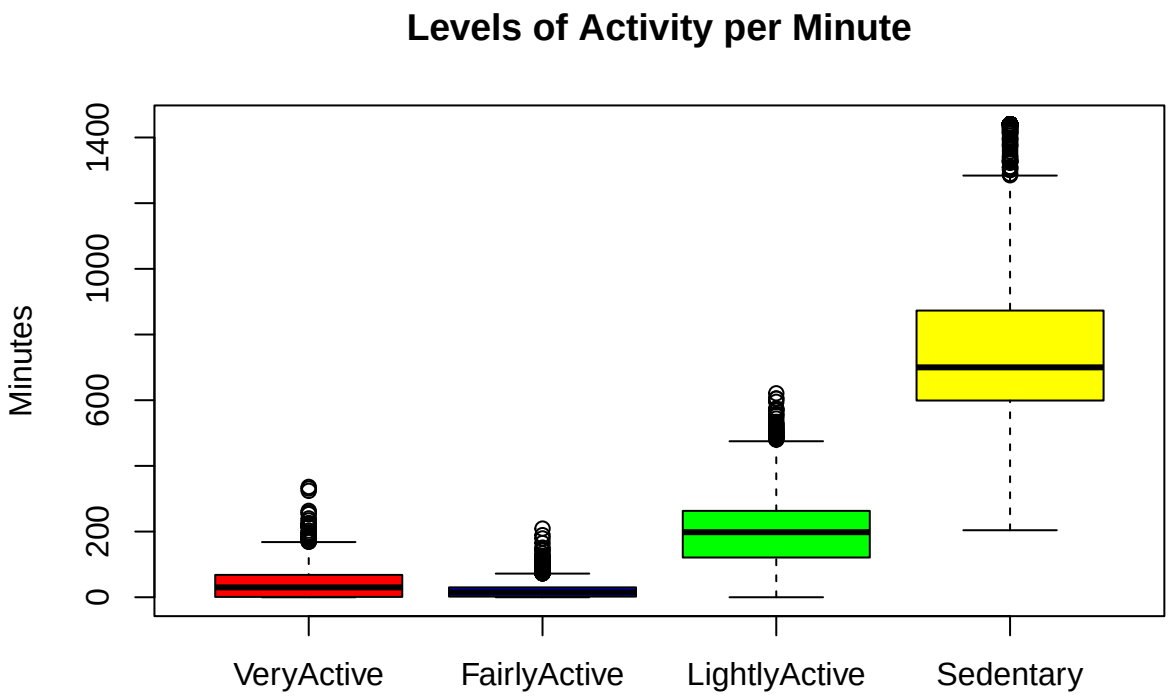


Are there any significant observations about the female participants. Although there is a small number of participants, doing a visual comparison of the mean values by gender does suggest some trends across gender. The plots above indicate for our sample that:

- Female average scores for calories consumed are on the bottom of the scale compared to males.
- Female average scores for total steps and very active minutes are at the low to middle end of the scale compared to males.
- Female average scores for sedentary minutes are at the middle to high end of the scale compared to males.
- Female average minutes of sleep scores are on par with male scores.

Levels of Activity per minute Here is a representation of the comparison between levels of activity per minute for the PMData:

```
boxplot(pm_daily_activity$very_active_minutes,
        pm_daily_activity$fairly_active_minutes,
        pm_daily_activity$lightly_active_minutes,
        pm_daily_activity$sedentary_minutes,
        main= "Levels of Activity per Minute",
        names= c("VeryActive", "FairlyActive", "LightlyActive", "Sedentary"),
        ylab= "Minutes",
        col = c("red", "blue", "green", "yellow"))
```




```
pm_daily_activity %>%
  select(very_active_minutes,
         fairly_active_minutes,
         lightly_active_minutes,
         sedentary_minutes) %>%
  summary()
```

```
##  very_active_minutes fairly_active_minutes lightly_active_minutes
##  Min.   :  0.00      Min.   :  0.00        Min.   :  0.0
##  1st Qu.:  1.25      1st Qu.:  2.00        1st Qu.:121.2
##  Median : 30.00      Median : 15.00        Median :198.0
##  Mean   : 42.45      Mean   : 20.79        Mean   :192.9
##  3rd Qu.: 68.00      3rd Qu.: 30.00        3rd Qu.:263.0
##  Max.   :336.00      Max.   :209.00        Max.   :621.0
##  sedentary_minutes
##  Min.   : 204.0
##  1st Qu.: 599.0
##  Median : 700.0
##  Mean   : 788.9
##  3rd Qu.: 872.5
##  Max.   :1440.0
```

We do see a similar pattern with regard to the levels of activity. Sedentary minutes still has the highest values and mean value of all four activity levels.

Create a table with average values for the four levels of activity per minute for both Fitbit data and PMData:

```r
mean_fitbit_data <- c(992.50, 185.4, 13.4, 19.68)
mean_pm_data <- c(788.9, 192.9, 20.79, 42.45)
level_of_activity <- c("sedentary_minutes", "lightly_active_minutes", "fairly_active_minutes", "very_ac
```

```r
mean_activity_levels1 <- data.frame(fitbit_mean = mean_fitbit_data, pm_mean = mean_pm_data, level_of_ac
```

```r
tibble(mean_activity_levels1)
```

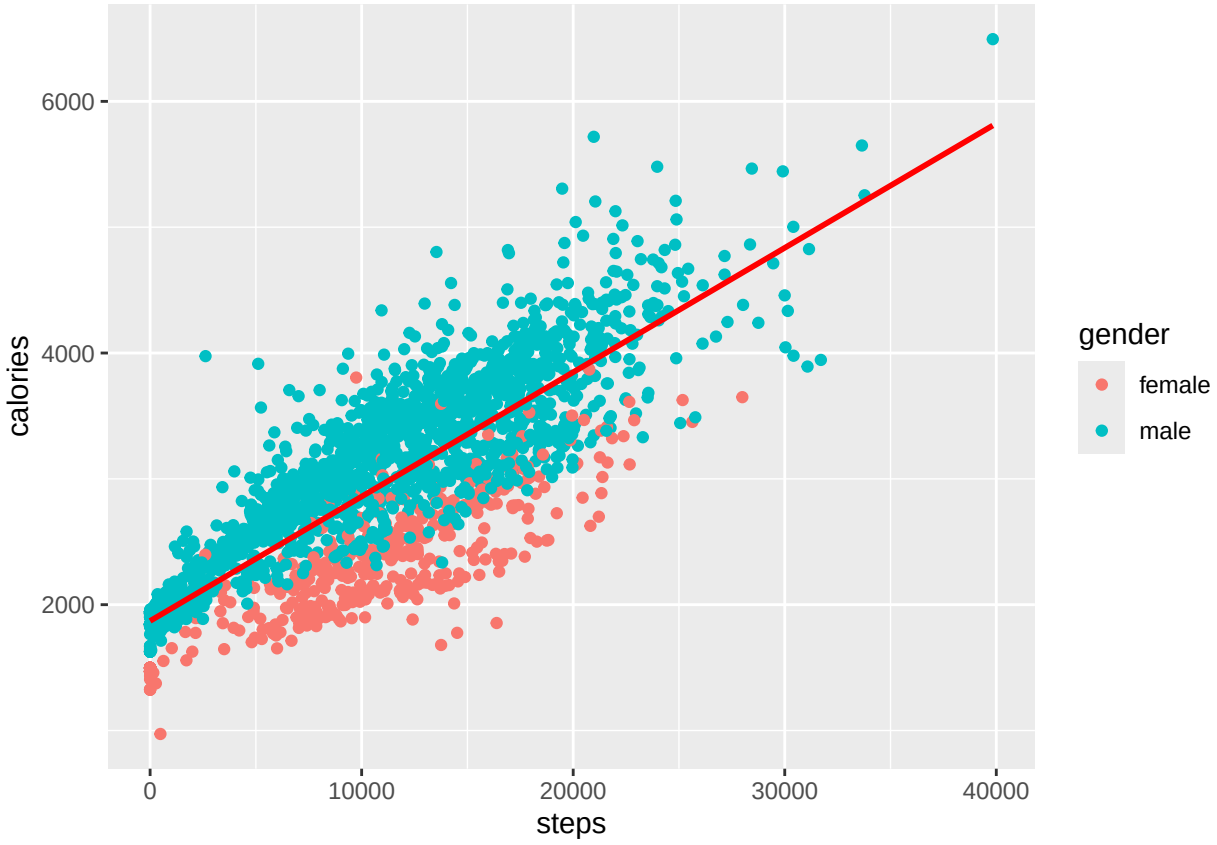
```
## # A tibble: 4 x 3
##   fitbit_mean pm_mean level_of_activity
##         <dbl>   <dbl> <chr>
## 1       992.    789.  sedentary_minutes
## 2       185.    193.  lightly_active_minutes
## 3        13.4    20.8 fairly_active_minutes
## 4        19.7    42.4 very_active_minutes
```

Calories vs Total Steps These plots explores the relationship between calories and total steps:

Highlighted for gender with color detail.

```r
ggplot(data=pm_daily_activity, aes(x=steps, y=calories, colour = gender)) + geom_point() + geom_smooth(
```

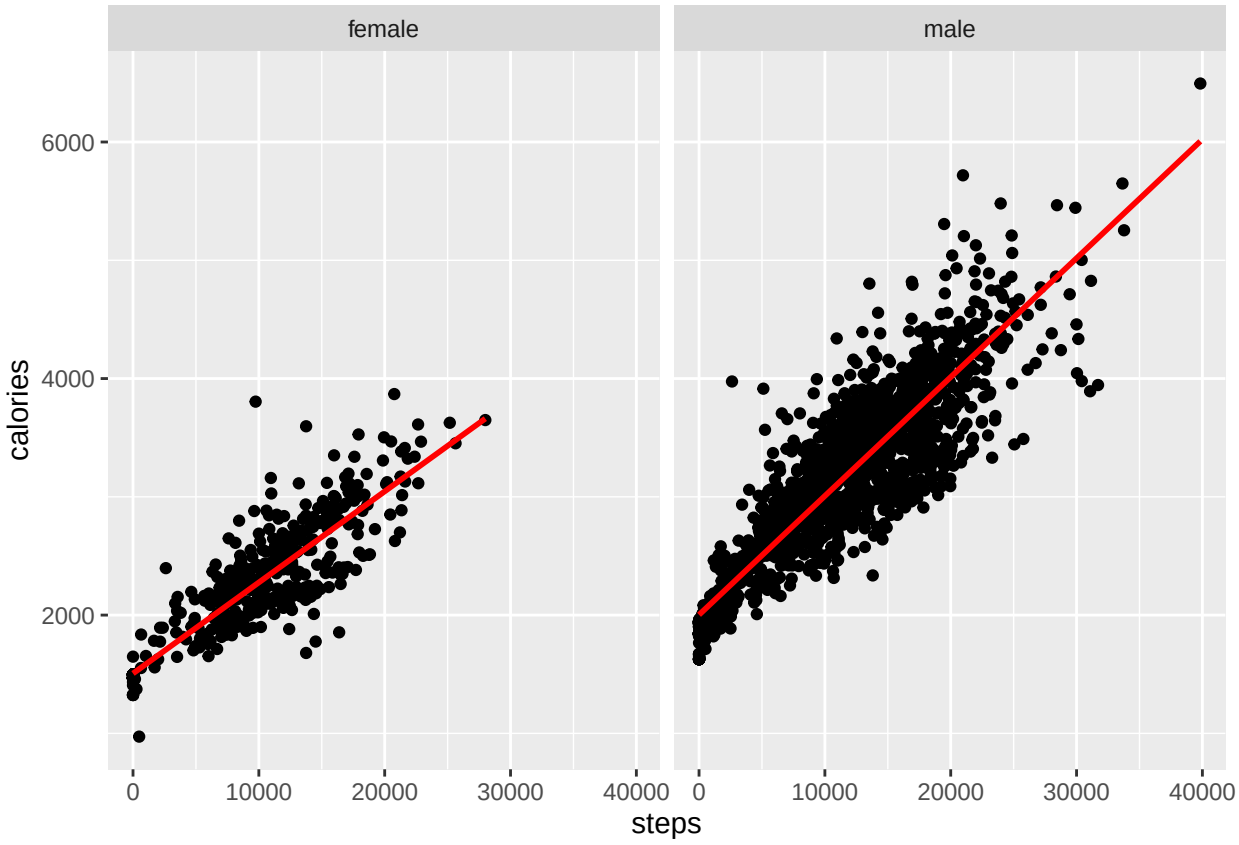
```
## `geom_smooth()` using formula = 'y ~ x'
```



Separated by gender.

```
ggplot(data=pm_daily_activity, aes(x=steps, y=calories)) + geom_point() + geom_smooth(method = lm, se=F
```

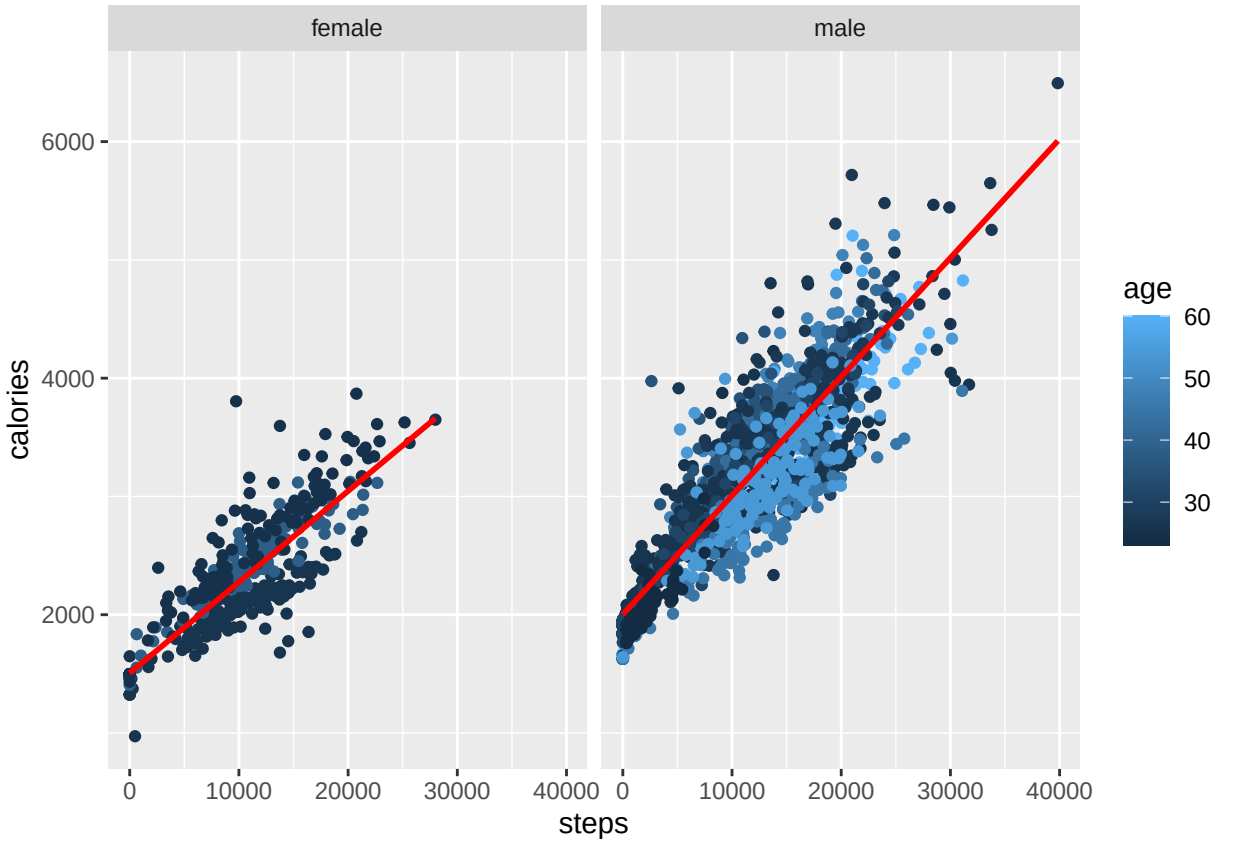
```
## `geom_smooth()` using formula = 'y ~ x'
```



Separated by gender with age detail noted in color.

```
ggplot(data=pm_daily_activity, aes(x=steps, y=calories, colour = age)) + geom_point() + geom_smooth(meth
```

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



Here is the important evaluation information for this relationship between calories and total steps without additional factors:

```
modelA <- lm(calories ~ steps, data = pm_daily_activity)

summary(modelA)
```

```
##
## Call:
## lm(formula = calories ~ steps, data = pm_daily_activity)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1635.7  -314.3    41.0   308.3  1845.5
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.871e+03  1.699e+01  110.13   <2e-16 ***
## steps       9.884e-02  1.372e-03   72.04   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 443.4 on 2192 degrees of freedom
## Multiple R-squared:  0.7031, Adjusted R-squared:  0.7029
## F-statistic:  5190 on 1 and 2192 DF,  p-value: < 2.2e-16
```

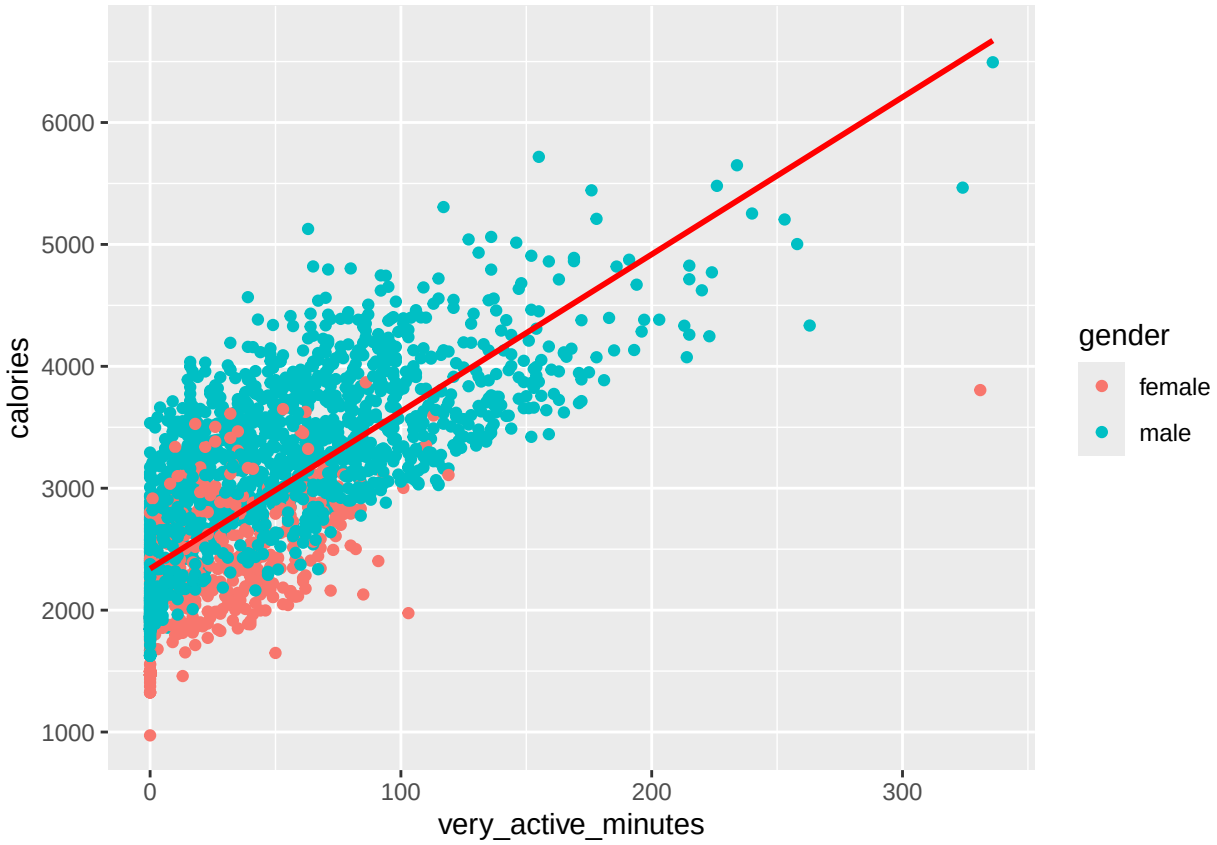
This gives us: Multiple R-squared: 0.7031, Adjusted R-squared: 0.7029, p-value: $< 2.2e\text{-}16$

This indicates the correlation is moderate to strong in this model and likely to be statistically significant.

Calories vs Very Active Minutes These plots explore the relationship between calories and very active minutes: Highlighted for gender with color detail.

```
ggplot(data=pm_daily_activity, aes(x=very_active_minutes, y=calories, colour = gender)) + geom_point() +
```

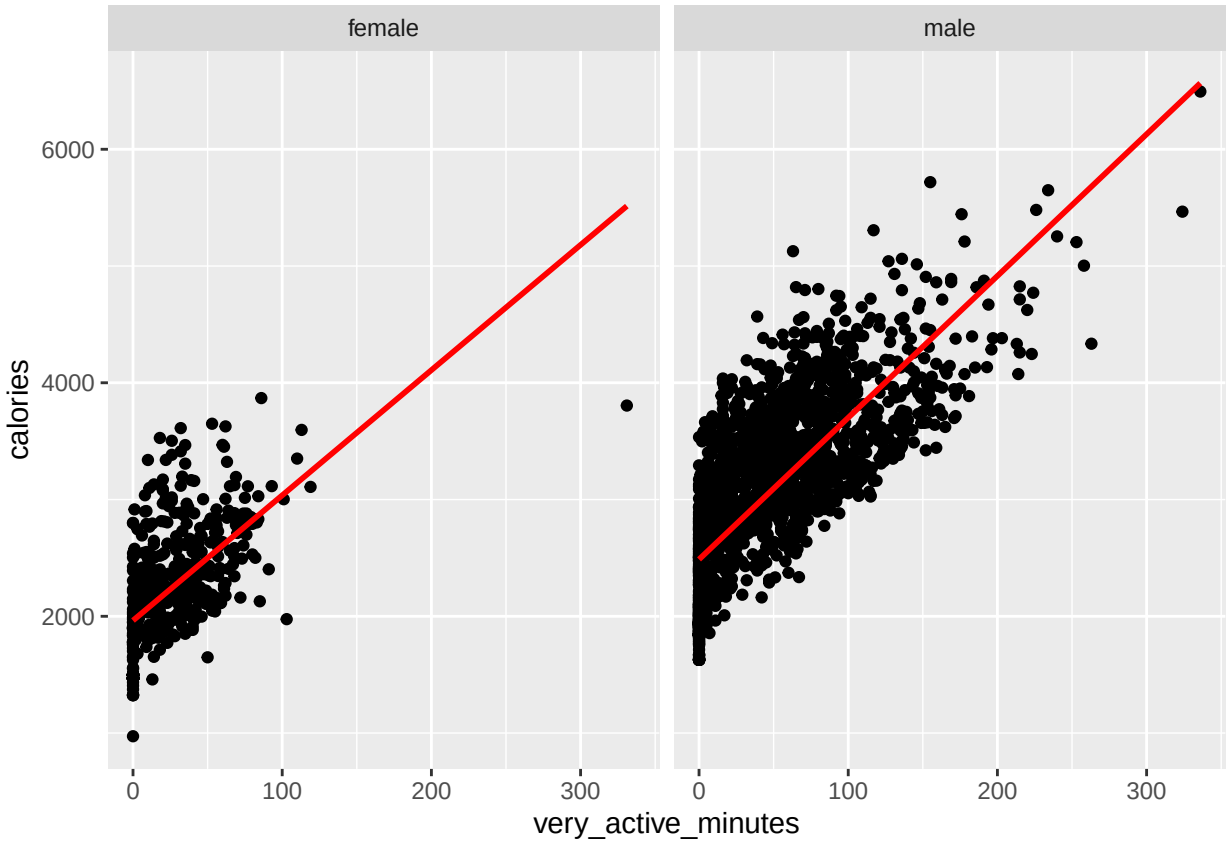
```
## `geom_smooth()` using formula = 'y ~ x'
```



Separated by gender.

```
ggplot(data=pm_daily_activity, aes(x=very_active_minutes, y=calories)) + geom_point() + geom_smooth(metl
```

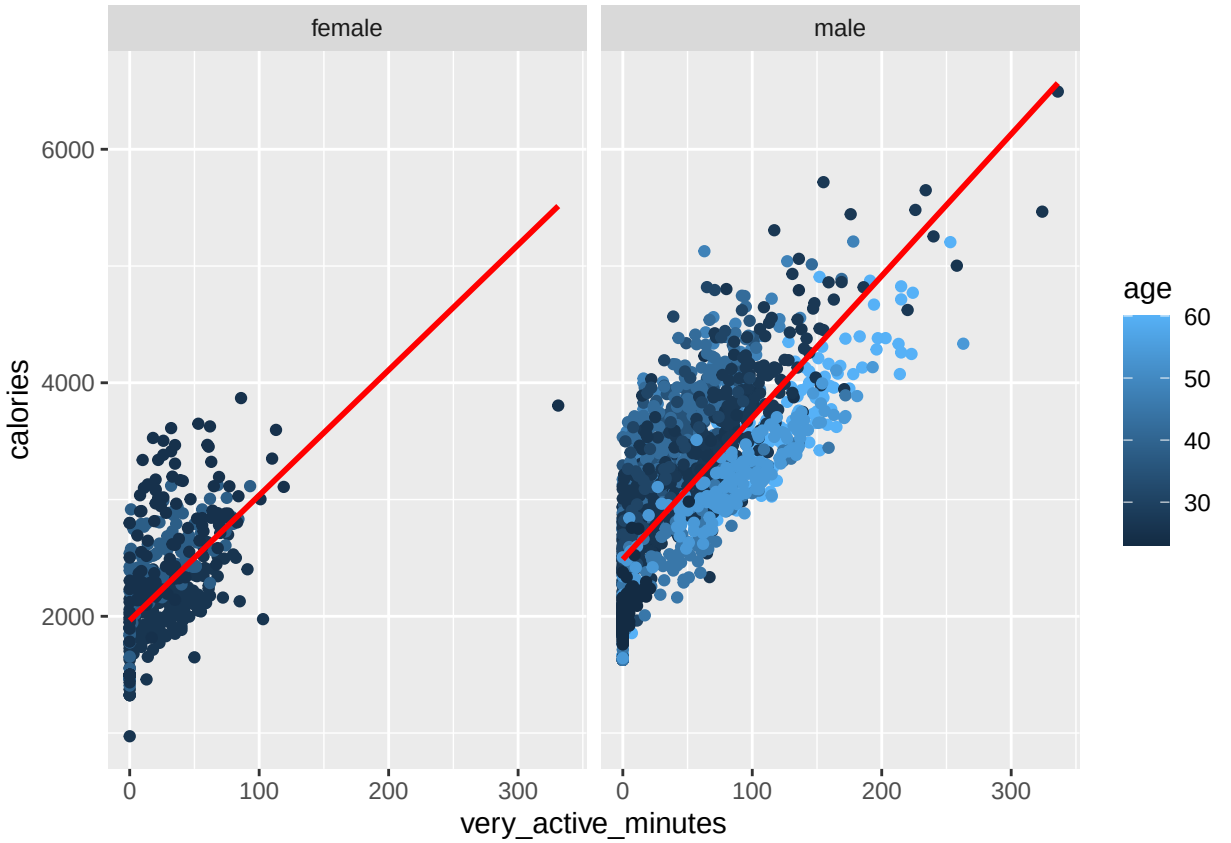
```
## `geom_smooth()` using formula = 'y ~ x'
```



Separated by gender with age detail noted in color.

```
ggplot(data=pm_daily_activity, aes(x=very_active_minutes, y=calories, colour = age)) + geom_point() + g
```

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



Here is the important evaluation information for this relationship between calories and very active minutes without additional factors:

```
modelC <- lm(calories ~ very_active_minutes, data = pm_daily_activity)

summary(modelC)
```

```
##
## Call:
## lm(formula = calories ~ very_active_minutes, data = pm_daily_activity)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2803.02  -421.52   -77.61   416.16  1975.06
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         2339.5422    16.0198  146.04   <2e-16 ***
## very_active_minutes   12.8954     0.2551   50.55   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 552.9 on 2192 degrees of freedom
## Multiple R-squared:  0.5383, Adjusted R-squared:  0.5381
## F-statistic:  2555 on 1 and 2192 DF,  p-value: < 2.2e-16
```

This gives Multiple R-squared: 0.5383, Adjusted R-squared: 0.5381, p-value: $< 2.2e\text{-}16$

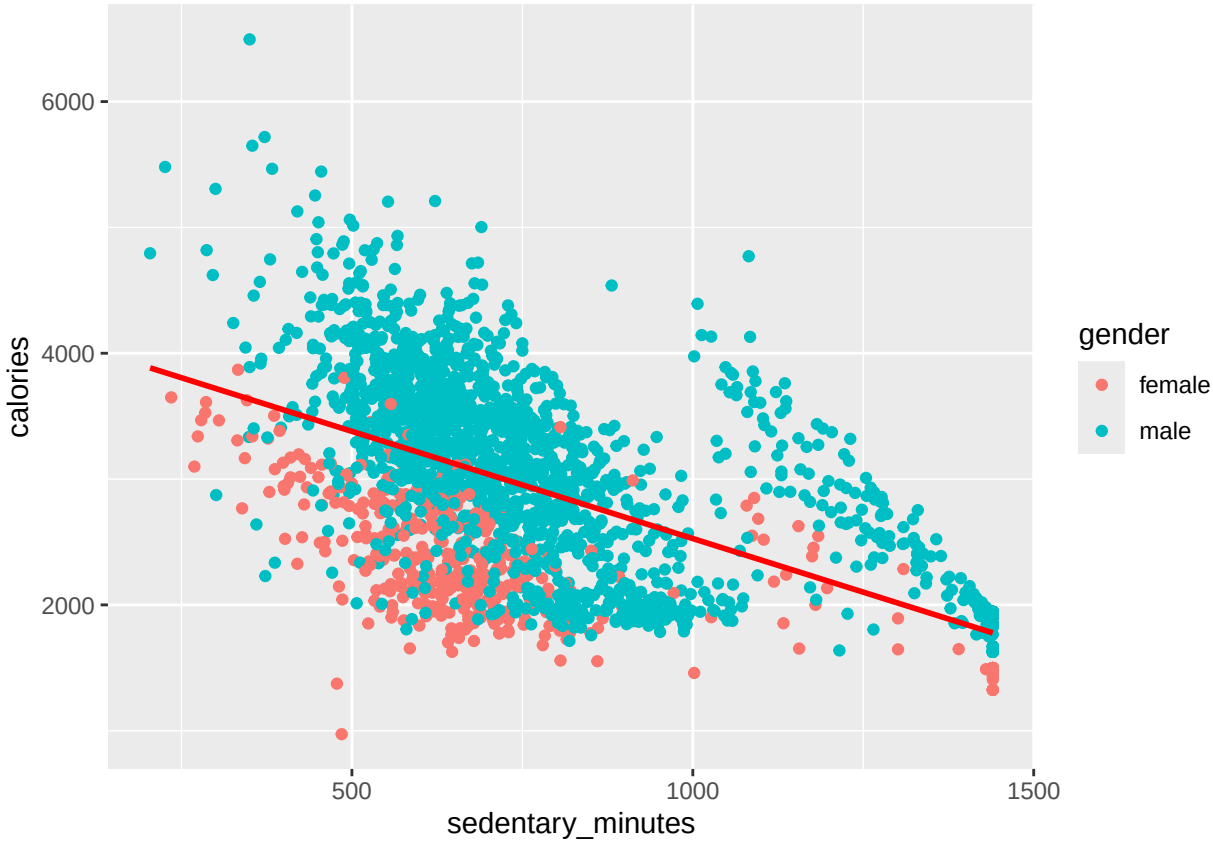
This indicates a moderate correlation that is likely to be statistically significant.

Calories vs Sedentary minutes These plots explore the relationship between calories and sedentary minutes:

Highlighted for gender with color detail.

```
ggplot(data=pm_daily_activity, aes(x=sedentary_minutes, y=calories, colour = gender)) + geom_point() + g
```

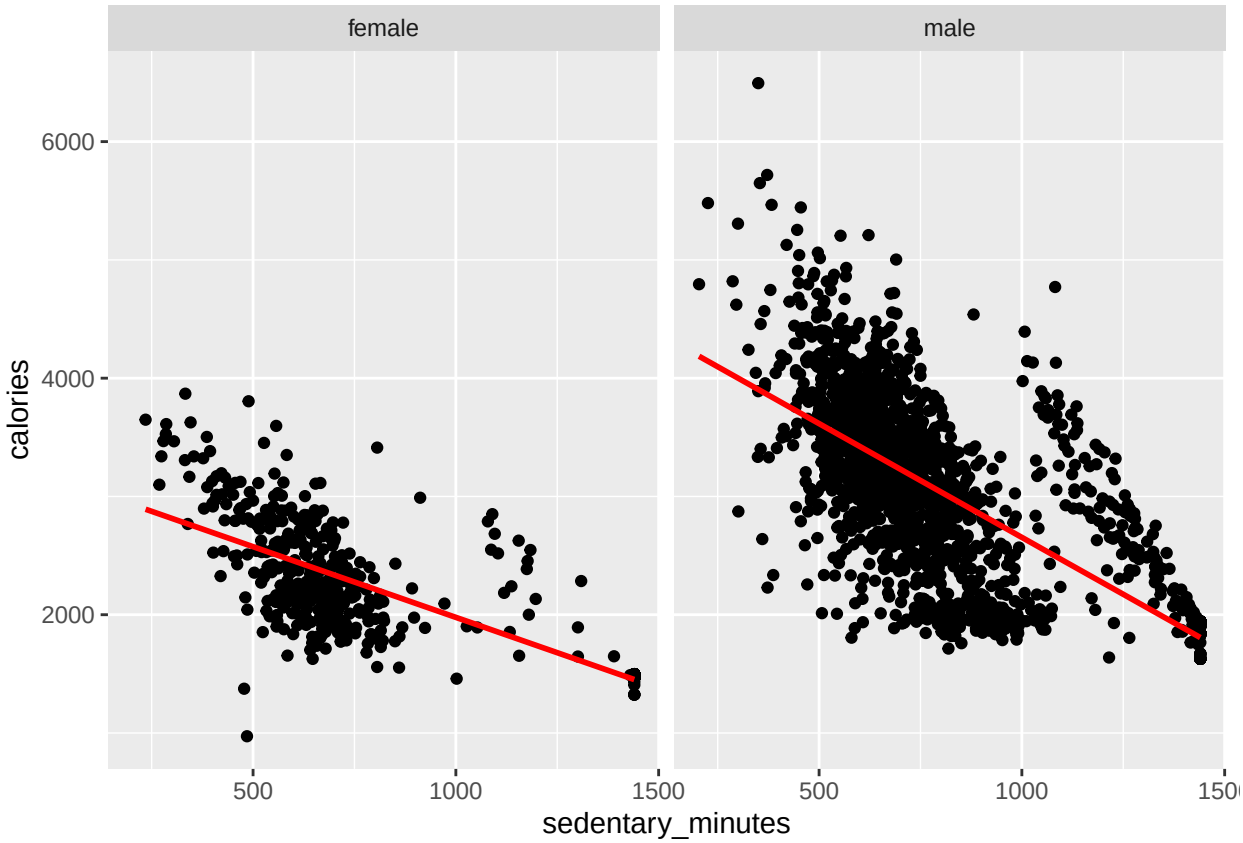
```
## `geom_smooth()` using formula = 'y ~ x'
```



Separated by gender.

```
ggplot(data=pm_daily_activity, aes(x=sedentary_minutes, y=calories)) + geom_point() + geom_smooth(metho
```

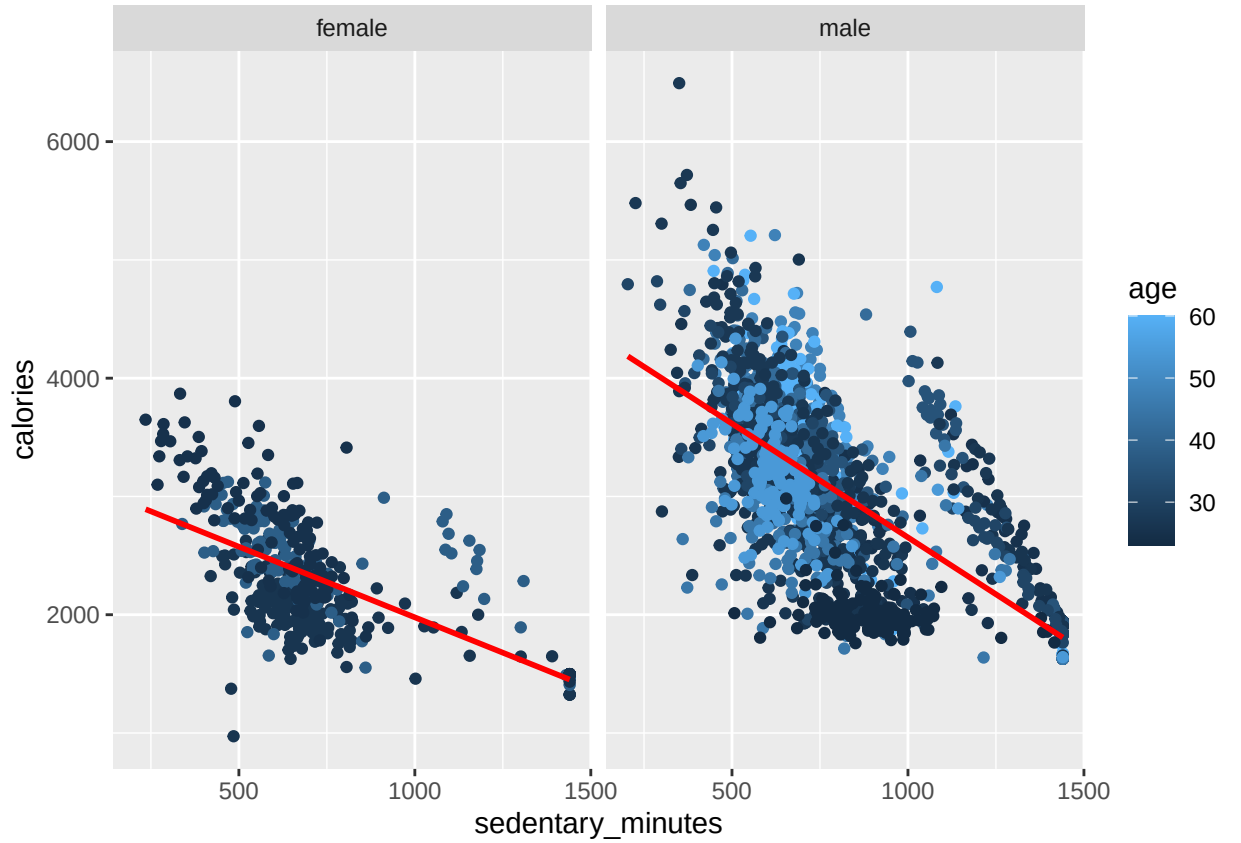
```
## `geom_smooth()` using formula = 'y ~ x'
```



Separated by gender with age detail noted in color.

```
ggplot(data=pm_daily_activity, aes(x=sedentary_minutes, y=calories, colour = age)) + geom_point() + geo
```

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



Here is the important evaluation information for this relationship between calories and sedentary minutes without additional factors:

```
modelF <- lm(calories ~ sedentary_minutes, data = pm_daily_activity)

summary(modelF)
```

```
##
## Call:
## lm(formula = calories ~ sedentary_minutes, data = pm_daily_activity)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2432.34  -467.72    51.26   432.91  2859.49
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       4231.36842   40.04277  105.67   <2e-16 ***
## sedentary_minutes   -1.70405    0.04764  -35.77   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 646.6 on 2192 degrees of freedom
## Multiple R-squared:  0.3685, Adjusted R-squared:  0.3682
## F-statistic:  1279 on 1 and 2192 DF,  p-value: < 2.2e-16
```

This gives Multiple R-squared: 0.3685, Adjusted R-squared: 0.3682, p-value: < 2.2e-16

This indicates a mild to moderate negative correlation that is likely to be statistically significant.

SHARE

Observations

Analysis of both datasets indicate that even among users of smart tracking devices, the level of sedentary activity per minute is very high compared to other activity levels.

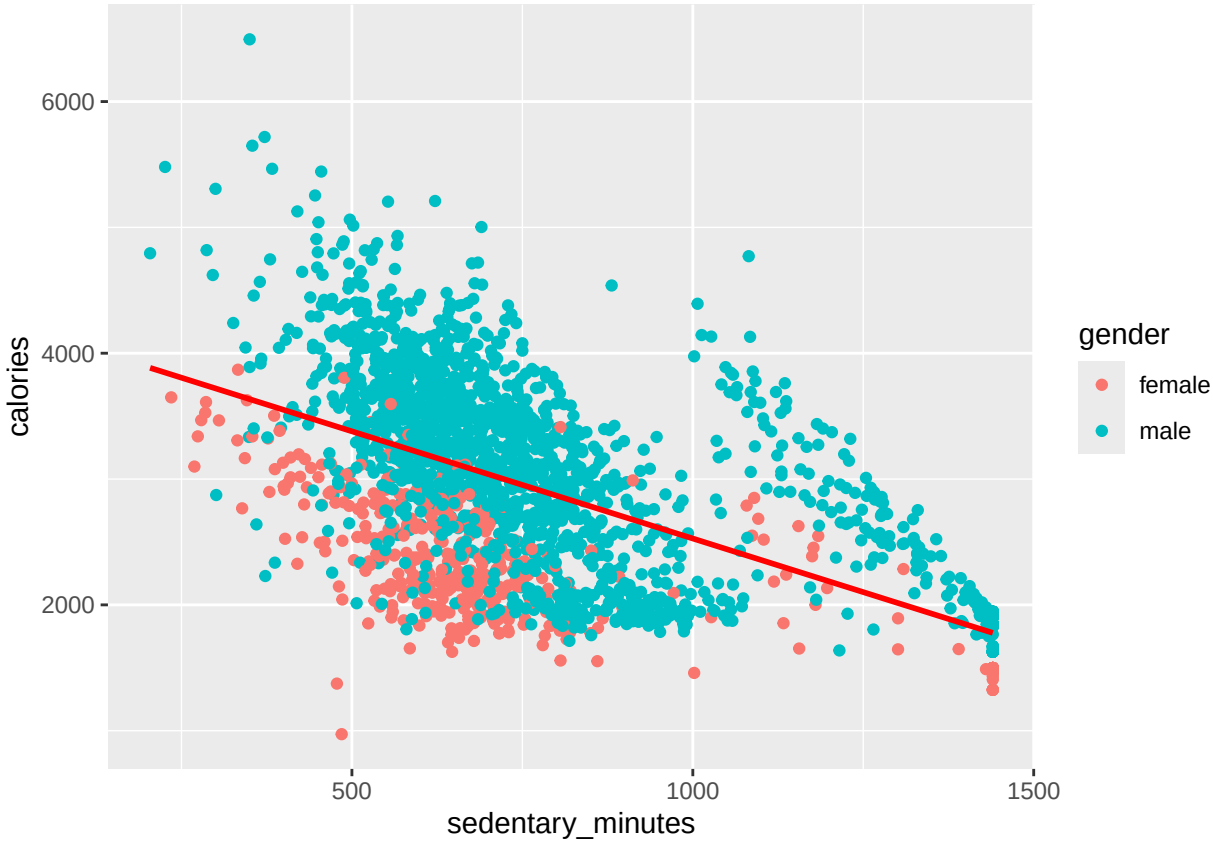
```
tibble(mean_activity_levels1)
```

```
## # A tibble: 4 x 3
##   fitbit_mean pm_mean level_of_activity
##         <dbl>   <dbl> <chr>
## 1       992.    789.  sedentary_minutes
## 2       185.    193.  lightly_active_minutes
## 3        13.4    20.8 fairly_active_minutes
## 4        19.7    42.4 very_active_minutes
```

The data also indicates that sedentary minutes are negatively correlated with calorie consumption while very active minutes and total steps are positively correlated with calorie consumption.

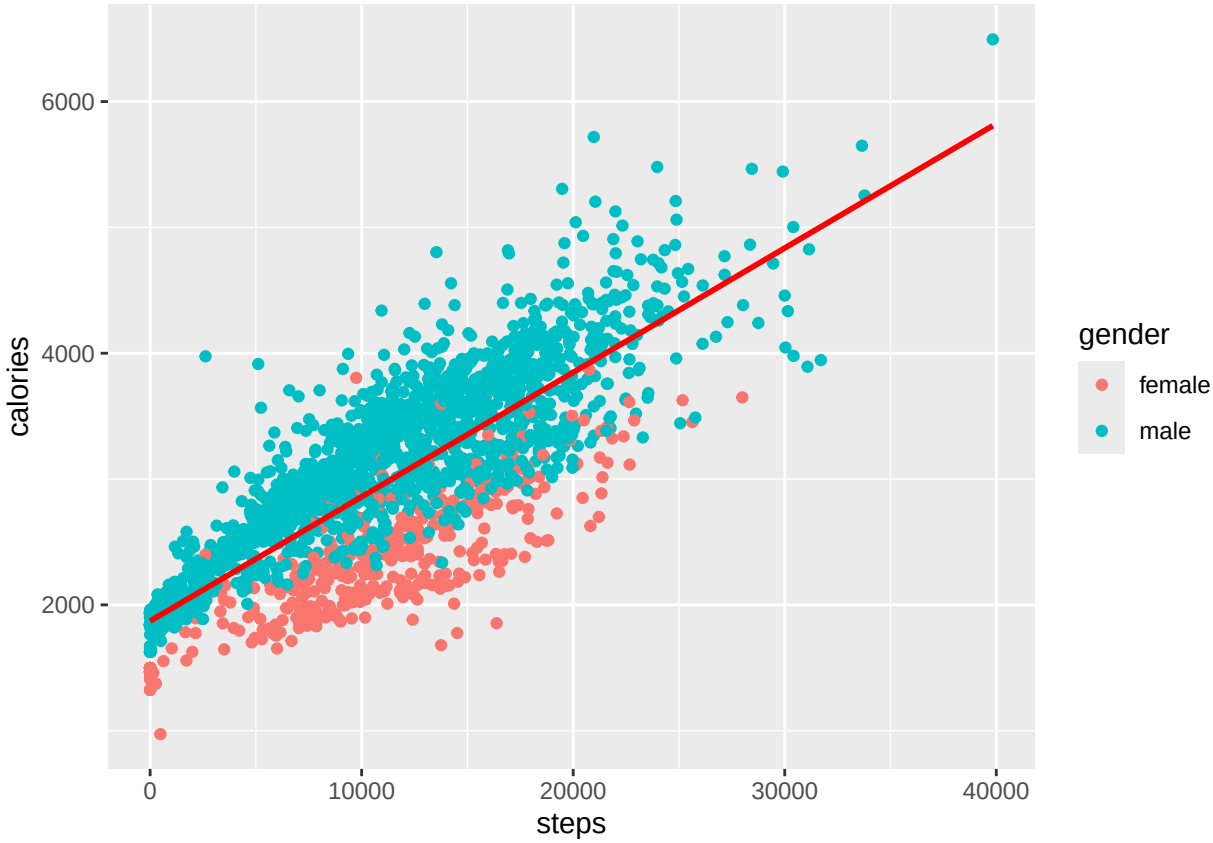
```
ggplot(data=pm_daily_activity, aes(x=sedentary_minutes, y=calories, colour = gender)) + geom_point() + g
```

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



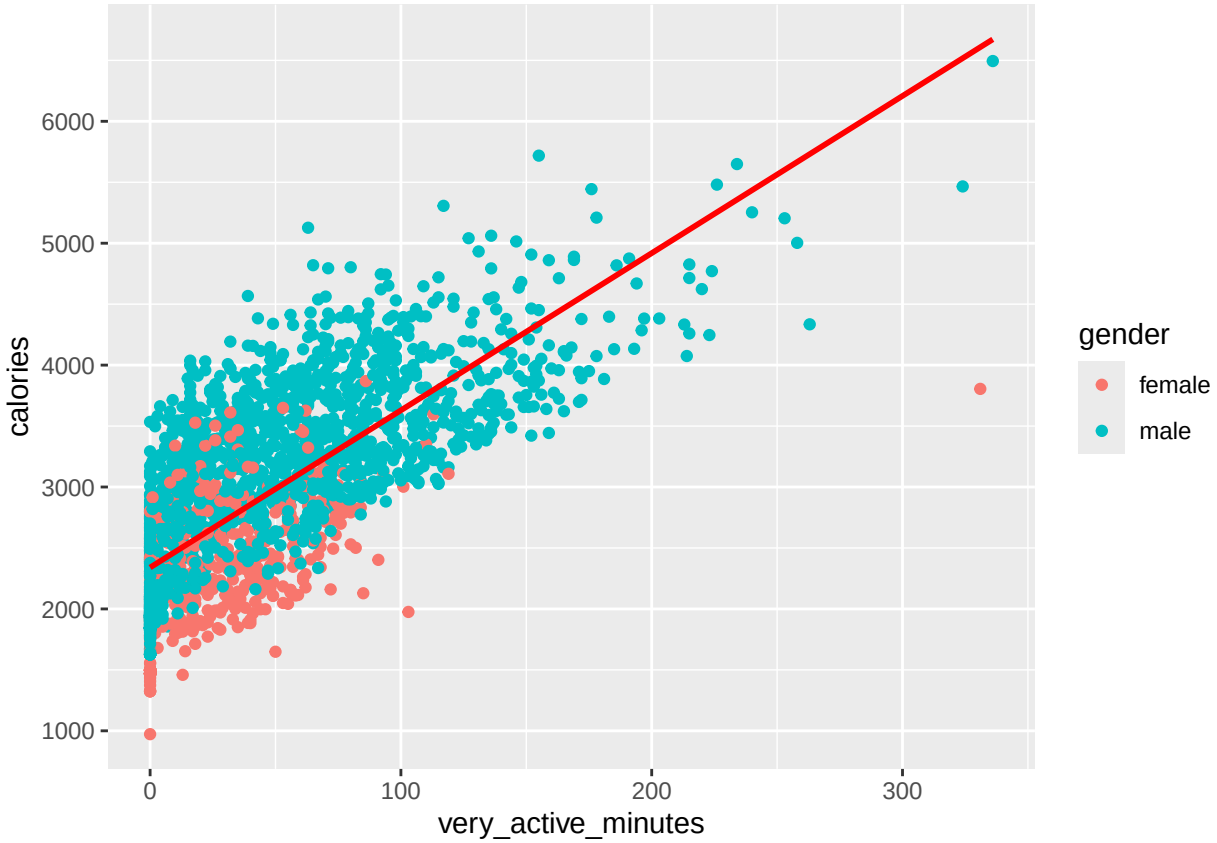

```
ggplot(data=pm_daily_activity, aes(x=steps, y=calories, colour = gender)) + geom_point() + geom_smooth(
```

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



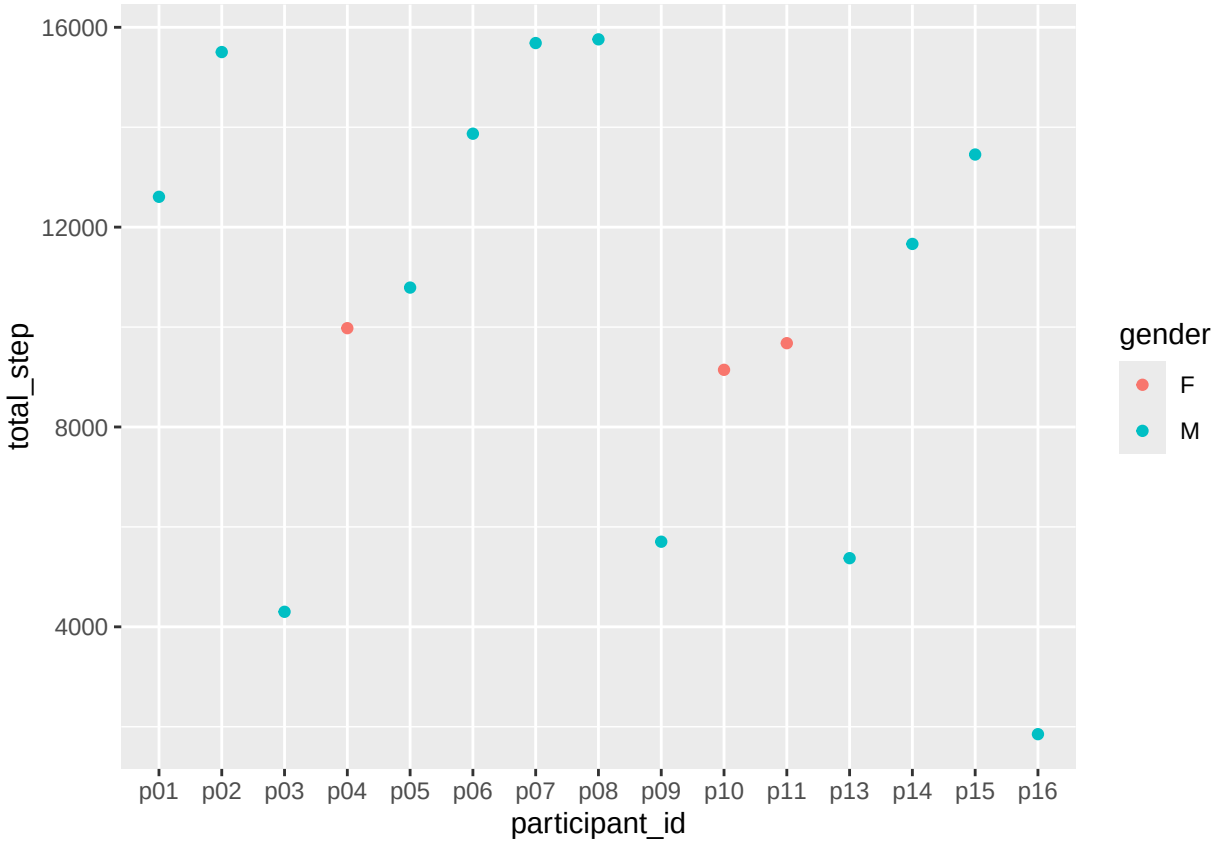

```
ggplot(data=pm_daily_activity, aes(x=very_active_minutes, y=calories, colour = gender)) + geom_point() +
```

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

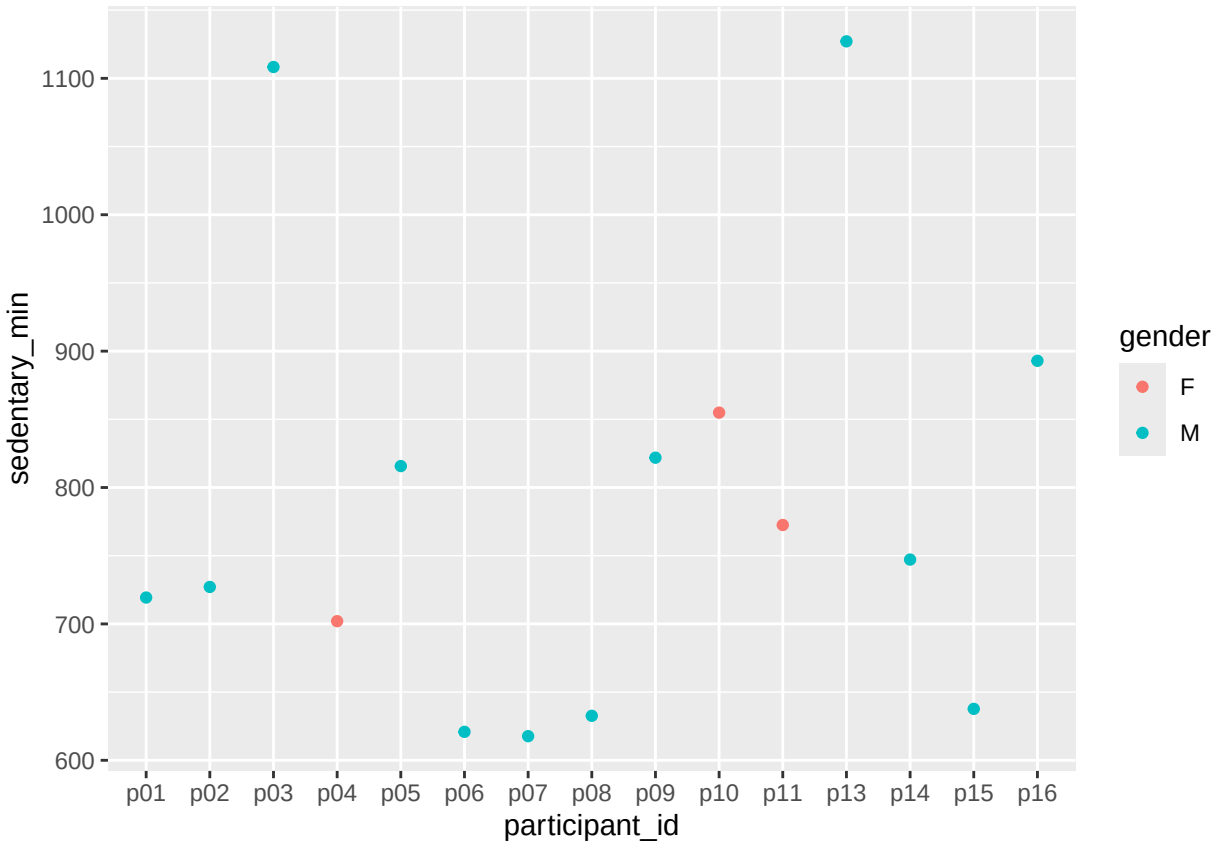


These observations are particularly interesting to this study because the limited data with gender information included point to females, which are Bellabeat's target demographic, having very low averages for calories consumed, as well as low to mid range averages for total steps and very active minutes. On the other hand, they have high averages for sedentary minutes.

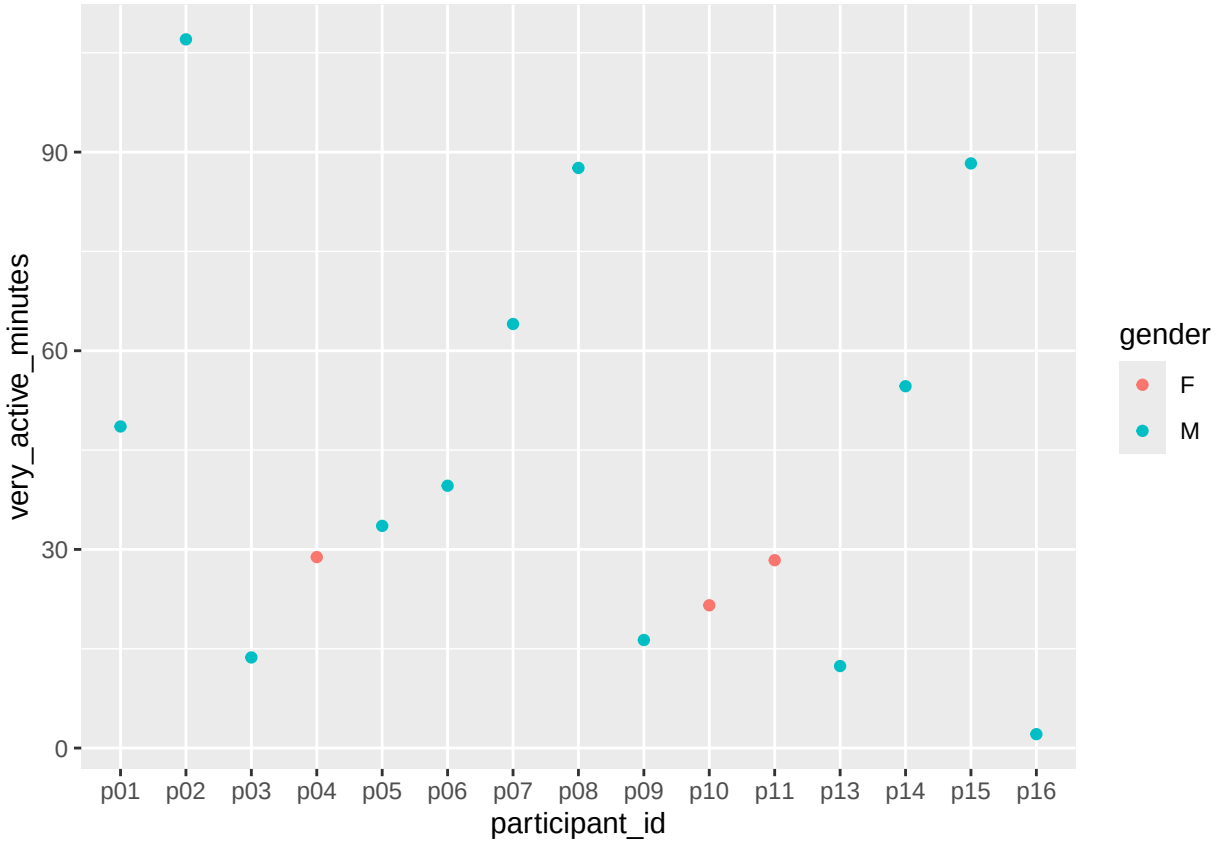
```
ggplot(data=combined_data_pm, aes(x=participant_id, y=total_step, colour = gender)) + geom_point()
```



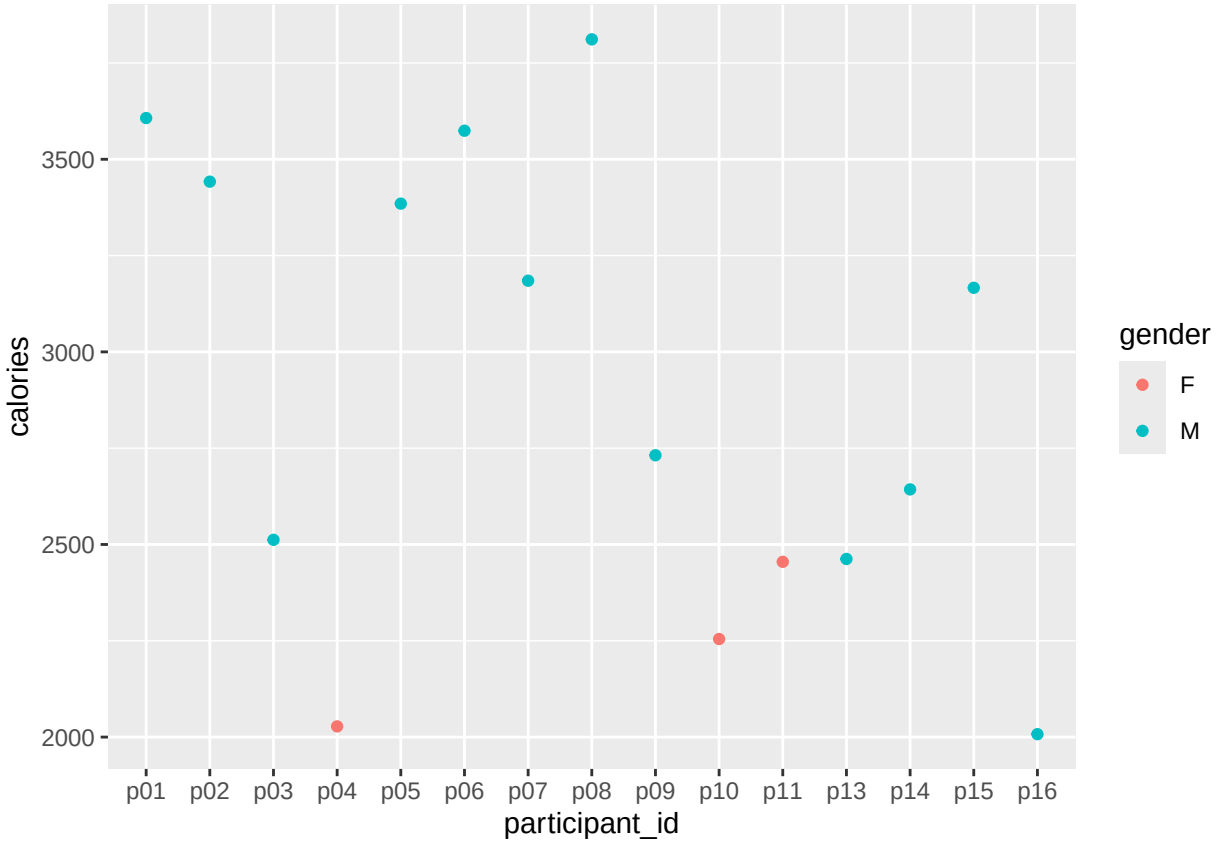

```
ggplot(data=combined_data_pm, aes(x=participant_id, y=sedentary_min, colour = gender)) + geom_point()
```




```
ggplot(data=combined_data_pm, aes(x=participant_id, y=very_active_minutes, colour = gender)) + geom_poi
```




```
ggplot(data=combined_data_pm, aes(x=participant_id, y=calories, colour = gender)) + geom_point()
```



Conclusion and Recommendations

A marketing strategy that emphasizes both the negative correlation of sedentary time and the positive correlation of higher total steps and very active minutes on calorie consumption would likely help Bellabeat's consumer base see the value of its Leaf tracker.

Targeting this marketing strategy to the large consumer segment with high levels of sedentary time by emphasizing the Leaf's ability to monitor activity levels while still being stylish and comfortable is a winning strategy for the Leaf.

However, due to the limited data on fitness tracking containing gender information as well as the concerns for completeness of data in general, I do suggest further data collection with demographic data included and analysis to confirm the trends noted and gender differences potentially highlighted in this study.

Thank You!