Google Data Analytics Case Study 2

Amit Kumar

2023-03-02

Bellabeat

Process

Lets load install and load the necessary packages required for this process which would be ‘Tidyverse’, ‘Janitor’, ‘Lubridate’, and ‘Skimr’

```
#Loading the packages
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.0     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.1     ✓ tibble    3.1.8
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ purrr     1.0.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✕ dplyr::filter() masks stats::filter()
## ✕ dplyr::lag()    masks stats::lag()
## ℹ Use the  ]8;;http://conflicted.r-lib.org/ conflicted package ]8;;  to force all conflicts to become errors
```

```
library(janitor)
```

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

```
library(lubridate)
library(skimr)
```

Before importing the dataset, we selected the directory of the file. To do this, the path is

```
# Set the working directory
setwd("/cloud/project")
```

After this, we would need to import the datasets into RStudio using “read.csv()”.

```
# df_name <- read.csv(dataset_name)
daily_activity <- read.csv("dailyActivity_merged.csv")
daily_sleep <- read.csv("sleepDay_merged.csv")
weight_log <- read.csv("weightLogInfo_merged.csv")
```

Let’s inspect our data to see if there are any errors with formatting by using “**str()**”.

```
#str(dataframe_name)
str(daily_activity)
```

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

```
str(daily_sleep)
```

```
## 'data.frame':    413 obs. of  5 variables:
##  $ Id                : num  1.5e+09 1.5e+09 1.5e+09 1.5e+09 1.5e+09 ...
##  $ SleepDay          : chr  "4/12/2016 12:00:00 AM" "4/13/2016 12:00:00 AM" "4/15/2016 12:00:00 AM" "4/16/2016 12:0
0:00 AM" ...
##  $ TotalSleepRecords : int  1 2 1 2 1 1 1 1 1 1 ...
##  $ TotalMinutesAsleep: int  327 384 412 340 700 304 360 325 361 430 ...
##  $ TotalTimeInBed    : int  346 407 442 367 712 320 377 364 384 449 ...
```

```
str(weight_log)
```

```
## 'data.frame':    67 obs. of  8 variables:
##  $ Id            : num  1.50e+09 1.50e+09 1.93e+09 2.87e+09 2.87e+09 ...
##  $ Date          : chr  "5/2/2016 11:59:59 PM" "5/3/2016 11:59:59 PM" "4/13/2016 1:08:52 AM" "4/21/2016 11:59:59 PM"
...
##  $ WeightKg      : num  52.6 52.6 133.5 56.7 57.3 ...
##  $ WeightPounds  : num  116 116 294 125 126 ...
##  $ Fat           : int  22 NA NA NA NA 25 NA NA NA NA ...
##  $ BMI           : num  22.6 22.6 47.5 21.5 21.7 ...
##  $ IsManualReport: chr  "True" "True" "False" "True" ...
##  $ LogId         : num  1.46e+12 1.46e+12 1.46e+12 1.46e+12 1.46e+12 ...
```

After a brief view of the output, there are a few issues that we need to address:

- The naming of the column names (camelCase)
- daily_activity $ ActivityDate - Is formatted as CHR not as a date format
- daily_sleep $ SleepDay - Is formatted as CHR not as a date format
- weight_log $ Date - Is formatted as CHR not as a date format
- weight_log $ IsManualReport is formatted as CHR not logical (for boolean values)

To clean the column names, we would use “clean_names()”.

```
#Change the column name style
daily_activity <- clean_names(daily_activity)
daily_sleep <- clean_names(daily_sleep)
weight_log <- clean_names(weight_log)
```

Let’s also format ‘daily_activity $ ActivityDate’, ‘daily_sleep $ SleepDay’, and ‘weight_log $ Date’ into the proper date format using “**as.date()**”.

```
# Convert string into date using as.date().
daily_activity $ activity_date <- as.Date(daily_activity $ activity_date,'%m/%d/%y')
daily_sleep $ sleep_day <- as.Date(daily_sleep $ sleep_day, '%m/%d/%y')
```

For weight_log$date, it’s a little tricky because if you look closely, there’s the PM indicator at the end. POSIX.ct does not recognize this and will return all values as NA, so we will need to use “**parse_date_time**” from Lubridate.

```
# Change string to date using parse_date_time.
weight_log $ date <- parse_date_time(weight_log$date, '%m/%d/%y %H:%M:%S %p')
```

And to format weight_log$is_manual_report to a logical format, we will use “**as.logical()**”.

```
# Convert string into logical type
weight_log $is_manual_report <- is.logical(weight_log $is_manual_report)
```

After a quick look at our current data, let’s add a day of the week, sedentary hours & total active hours column for further analysis in daily_activity. I will not be adding a month column since the dataset only provides information collected within a month.

Let’s also add new columns which convert the current minutes of collection to hours and round it using “**round()**” in daily_sleep. I will also be adding a column to indicate the time taken to fall asleep in daily_sleep as well.

We will also be removing weight_log$fat, as it has little to no context and would not be helpful during the analysis phase by using “**select(-c())**”.

```
# Round basically rounds off a number, the syntax would be = round(object, digits = x)
daily_activity$day_of_week <- wday(daily_activity$activity_date, label = T, abbr = T)
daily_activity$total_active_hours = round((daily_activity$very_active_minutes + daily_activity$fairly_active_minutes +
daily_activity$lightly_active_minutes)/60, digits = 2)
daily_activity$sedentary_hours = round((daily_activity$sedentary_minutes)/60, digits = 2)
```

```
daily_sleep $ hours_in_bed = round((daily_sleep $ total_time_in_bed)/60, digits = 2)
daily_sleep $ hours_asleep = round((daily_sleep $ total_minutes_asleep)/60, digits = 2)
daily_sleep $ time_taken_to_sleep = (daily_sleep$total_time_in_bed - daily_sleep$total_minutes_asleep)
```

```
# Remove the fat column from weight_log
weight_log <- weight_log %>%
  select(-c(fat))
```

Lastly, I will also be adding a new column in weight_log called bmi2 which will indicate whether the user is underweight, healthy, or overweight by using a line of code I recently learned about which is “**case_when**”!

```
#case_when is an R equivalent of of SQL's CASE WHEN
#The last TRUE is basically the R equivalent of SQL's ELSE

weight_log <- weight_log %>%
  mutate(bmi2 = case_when(
    bmi > 24.9 ~ 'Overweight',
    bmi < 18.5 ~ 'Underweight',
    TRUE ~ 'Healthy'
  ))
```

Before we move onto the phase where we actually start to analyze the data frame, we need to remove any outliers from the data.

In this case, let’s remove rows in which the total_active_hours & calories burned are 0. The reasoning behind this is that we’re using data collected from Fitbits, which are wearables. If they don’t wear their smart devices it doesn’t collect information, hence we will remove the clutter from the data frame. Users might have also disabled GPS/accelerometer functions that allow for the collection of steps taken.

```
#In laymans term, '!' means is not equals to
daily_activity_cleaned <- daily_activity[!(daily_activity$calories<=0),]
daily_activity_cleaned <- daily_activity_cleaned[!(daily_activity_cleaned$total_active_hours<=0.00),]
```

Analyse

I will be using ggplot for this section of the analysis phase. I will also be including another section in which I used Tableau instead.

As per usual, let’s revisit our business task to ensure we are not plotting or trying to hypothesize information/relationships which will not help in solving the business task which are: 1. What are some trends in smart device usage? 2. How could these trends apply to Bellabeat customers 3. How could these trends help influence Bellabeat’s marketing strategy?

After having a brief view of the current data, I will be plotting a few observations revolving around: * The average: Steps taken, sedentary hours, very active minutes & total hours asleep. * Which days are users the most active? * The relationship between total active hours, total steps taken, and sedentary hours against calories burned. * The relationship between weight, total active hours & steps taken * The number of overweight users

Let’s have a quick look at the average steps taken, sedentary hours, very active minutes & total hours of sleep using “**summary()**”.

```
summary(daily_activity_cleaned $total_steps)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0    4920    8053    8319   11100   36019
```

```
summary(daily_activity_cleaned $ sedentary_hours)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   12.02   17.00   15.87   19.80   23.98
```

```
summary (daily_activity_cleaned $very_active_minutes)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    0.00    7.00   23.21   36.00  210.00
```

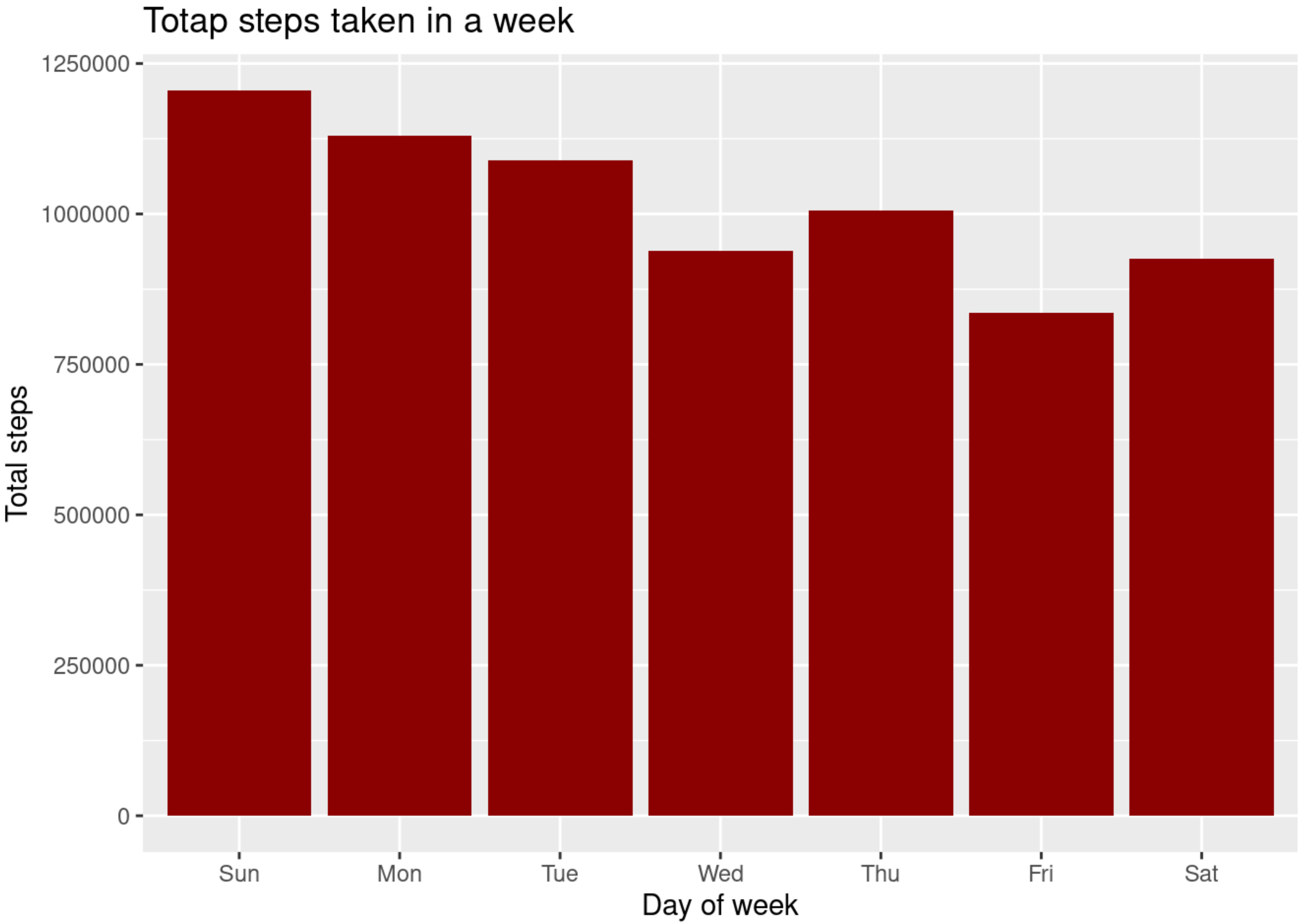
```
summary(daily_sleep $ hours_asleep)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   0.970   6.020   7.220   6.992   8.170  13.270
```

With a brief view of the outputs above: * The average number of steps per day was 8319, which is within the 6000–8000 recommended steps per day, however, 25% of people do not hit that recommended quota. * The average sedentary hours were 15.87 hours, which is absurdly high, shattering the recommended limit of 7–10 hours * The average very active minutes also falls short of the recommended 30 minutes of vigorous exercise every day. Only 25% of people manage to hit this quota * The average hours spent asleep (6.9) also barely hits the quota of the recommended sleep time of 7–9 hours

Now let’s have a look at which days are users most active:

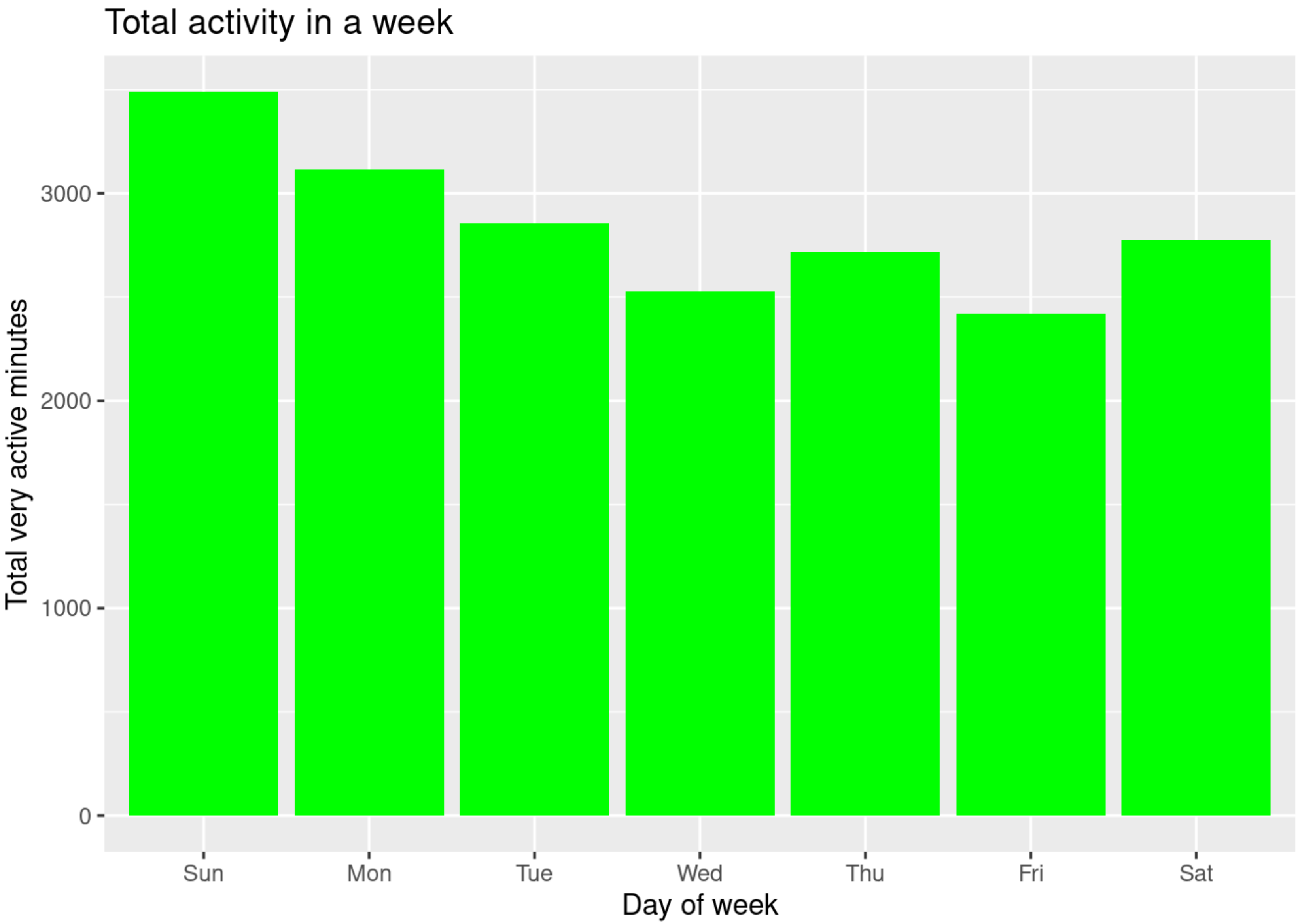
```
ggplot(data = daily_activity_cleaned) +
  aes(x = day_of_week, y = total_steps) +
  geom_col(fill =  'darkred') +
  labs(x = 'Day of week', y = 'Total steps', title = 'Totap steps taken in a week')
```



```
ggsave('total_steps.png')
```

```
## Saving 7 x 5 in image
```

```
ggplot(data = daily_activity_cleaned) +
  aes(x = day_of_week, y = very_active_minutes) +
  geom_col(fill =  'green') +
  labs(x = 'Day of week', y = 'Total very active minutes', title = 'Total activity in a week')
```

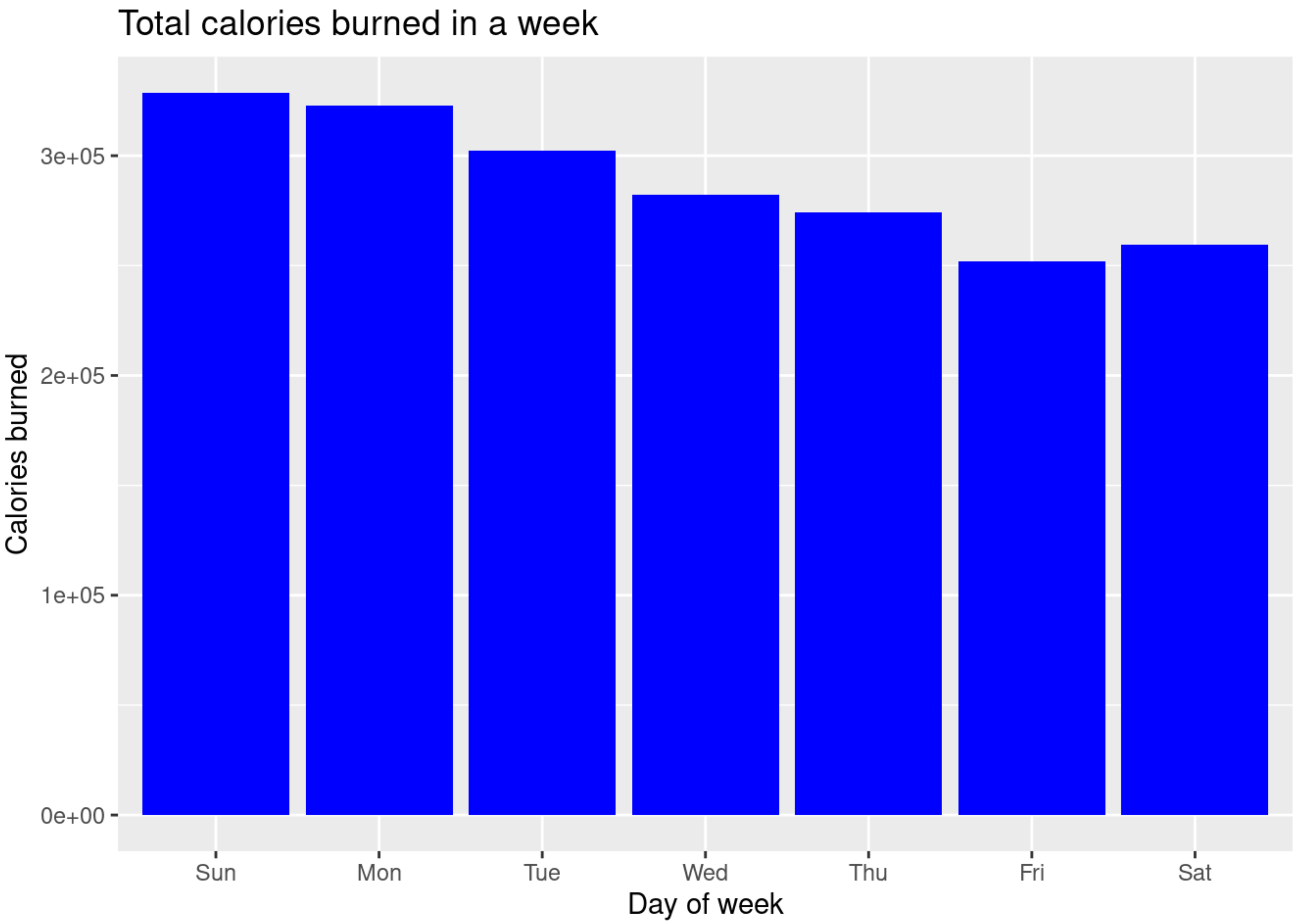


Total activity in a week

```
ggsave('total_activity.png')
```

```
## Saving 7 x 5 in image
```

```
ggplot(data = daily_activity_cleaned) +
  aes(x = day_of_week, y = calories) +
  geom_col(fill =  'blue') +
  labs(x = 'Day of week', y = 'Calories burned', title = 'Total calories burned in a week')
```



```
ggsave('total_calories.png')
```

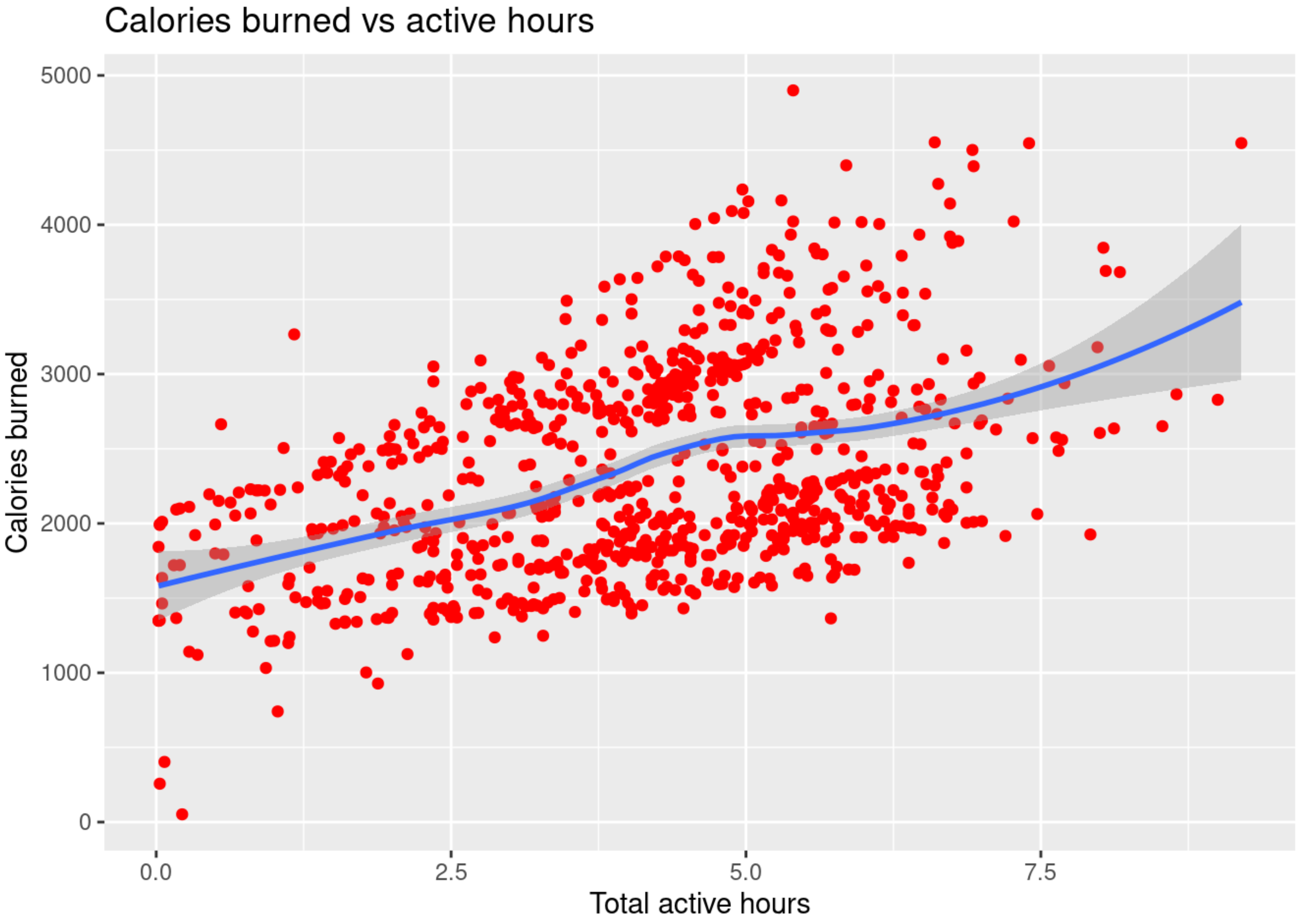
```
## Saving 7 x 5 in image
```

As we can see, the most active days for Fitbit users were Sundays, with a slow decline throughout the week. This could be due to motivation levels being fairly high during the end of the week.

Next, let’s investigate the relationship between total active hours, total steps taken, and sedentary hours against calories burned by using the following:

```
ggplot(data = daily_activity_cleaned) +
  aes(x= total_active_hours, y = calories) +
  geom_point(color = 'red') +
  geom_smooth() +
  labs(x = 'Total active hours', y = 'Calories burned', title = 'Calories burned vs active hours')
```

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

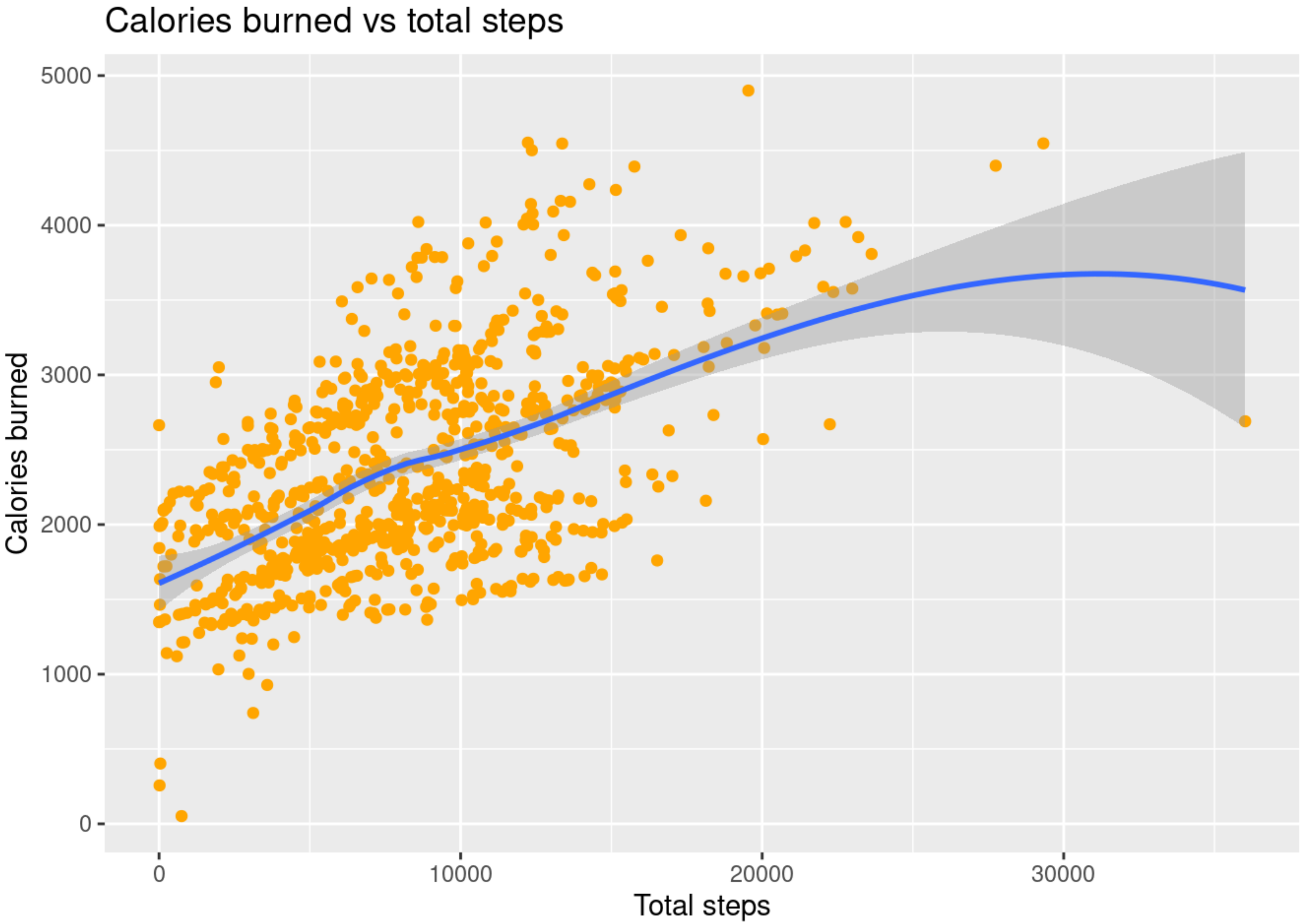


```
ggsave('calories_burned_vs_active_hours.png')
```

```
## Saving 7 x 5 in image
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

```
ggplot(data = daily_activity_cleaned) +
  aes(x= total_steps, y = calories) +
  geom_point(color = 'orange') +
  geom_smooth() +
  labs(x = 'Total steps', y = 'Calories burned', title = 'Calories burned vs total steps')
```

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

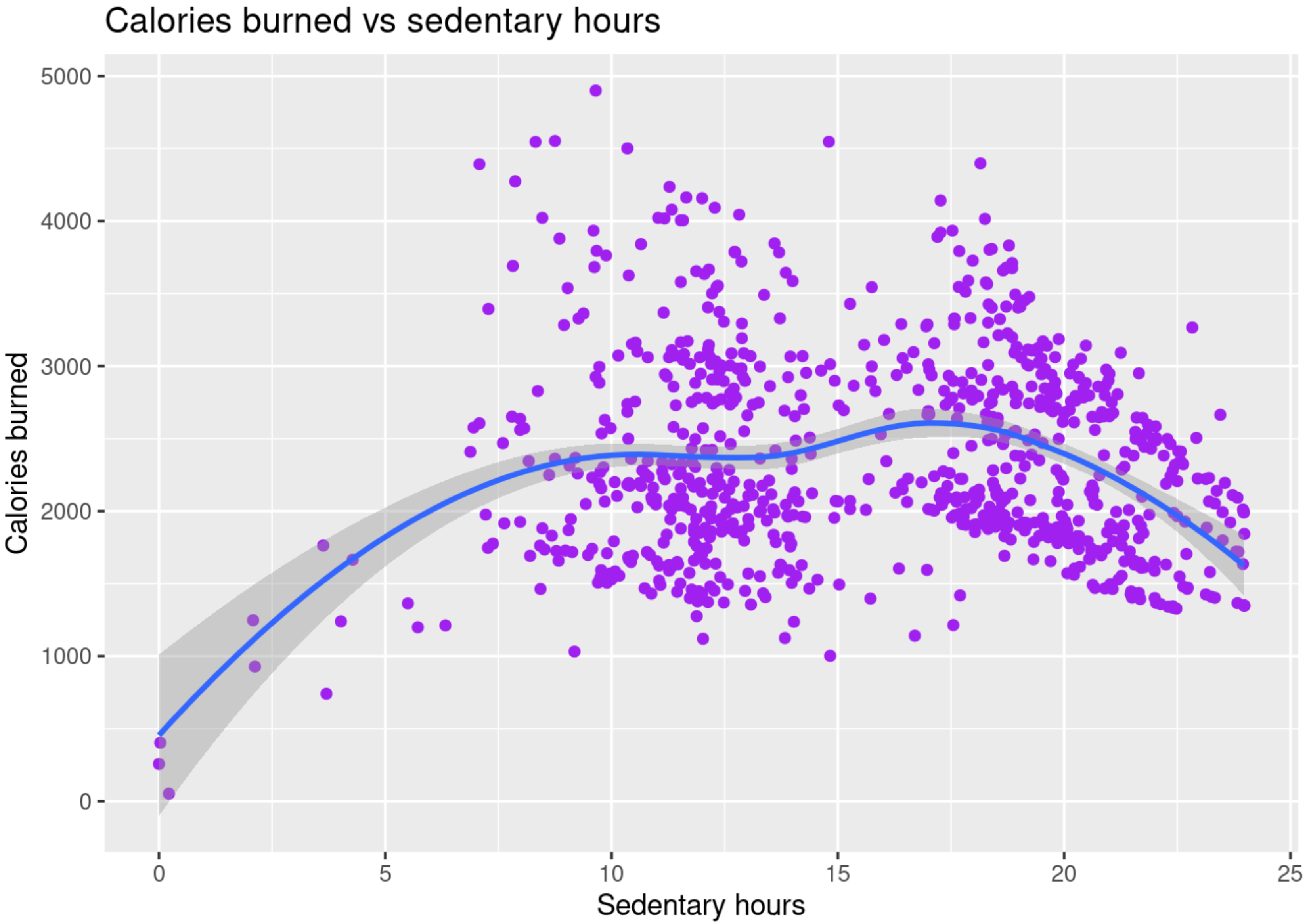


```
ggsave('calories_burned_vs_total_steps.png')
```

```
## Saving 7 x 5 in image
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

```
ggplot(data = daily_activity_cleaned) +
  aes(x= sedentary_hours, y = calories) +
  geom_point(color = 'purple') +
  geom_smooth() +
  labs(x = 'Sedentary hours', y = 'Calories burned', title = 'Calories burned vs sedentary hours')
```

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



```
ggsave('sedentary_hours_vs_calories_burned.png')
```

```
## Saving 7 x 5 in image
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

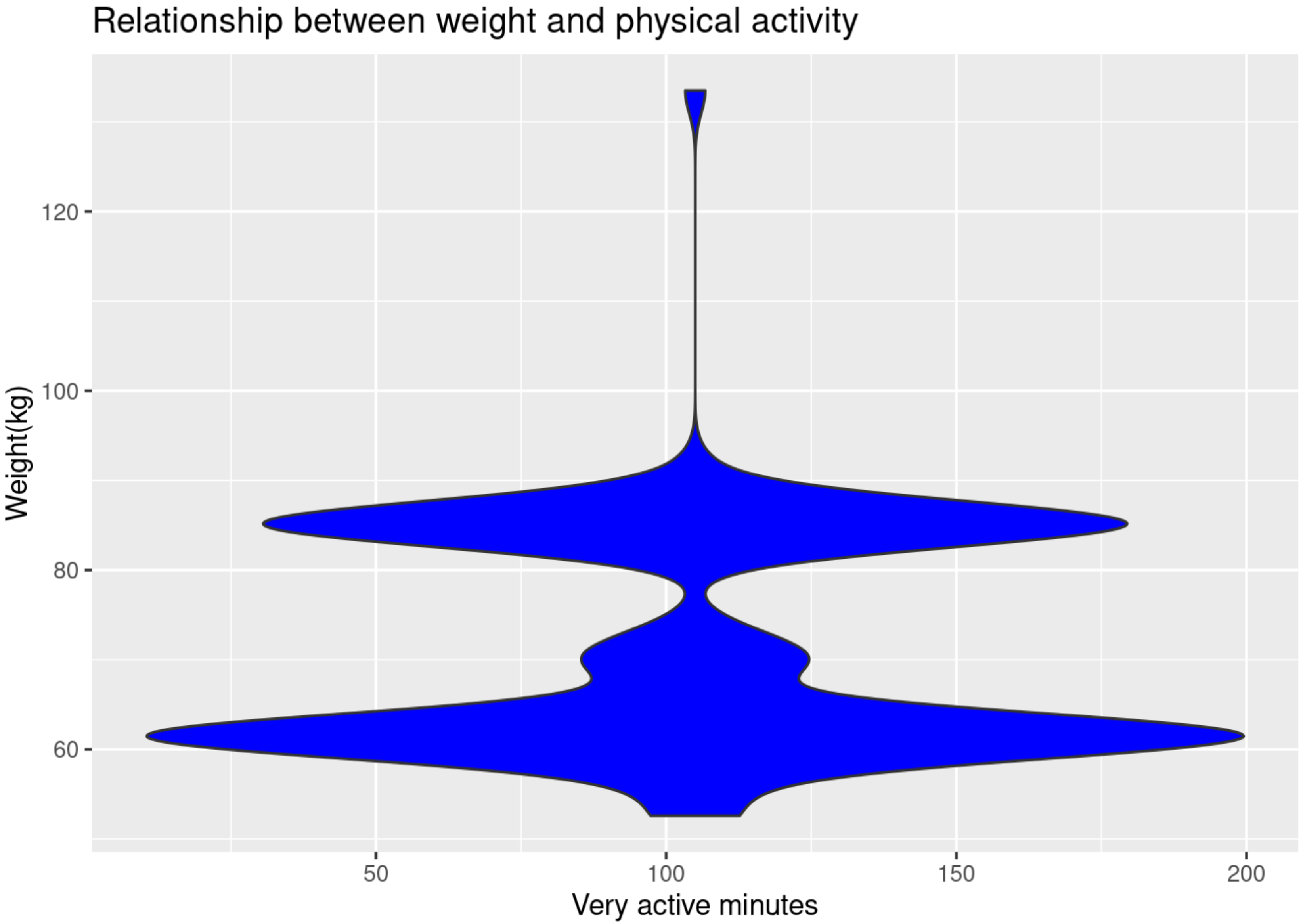
At a glance, we can tell that there is a positive correlation between calories burned and total steps taken/total active hours. However, in the last chart, we can see that the correlation is confusing.

I was expecting an inverse relationship with the first 2 charts however I was wrong. The relationship between sedentary hours and calories burned was fairly positive up till about the 17-hour mark.

For the relationship between weight & physical activity, we would use:

```
# merge the daily_activity_cleaned and weight_log vectors
merged_weight <- merge(daily_activity_cleaned, weight_log, by = c('id'))
```

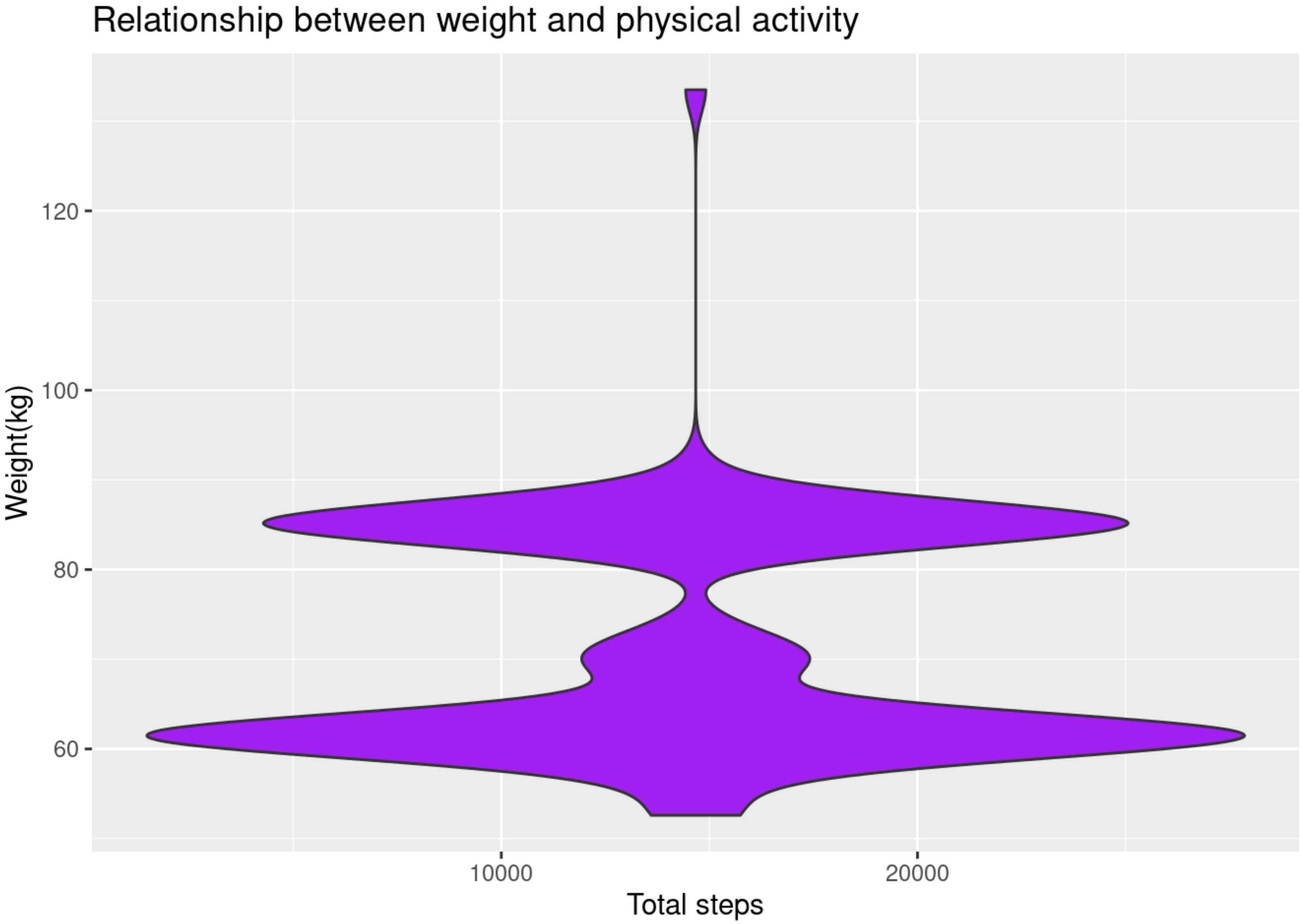
```
# Create voilin chart
ggplot(data = merged_weight) +
  aes(x = very_active_minutes, y = weight_kg) +
  geom_violin(fill = 'blue') +
  labs(x = 'Very active minutes', y = 'Weight(kg)', title = 'Relationship between weight and physical activity')
```



```
ggsave('weight_physical_activity.png')
```

```
## Saving 7 x 5 in image
```

```
ggplot(data = merged_weight) +
  aes(x = total_steps, y = weight_kg) +
  geom_violin(fill = 'purple') +
  labs(x = 'Total steps', y = 'Weight(kg)', title = 'Relationship between weight and physical activity')
```



```
ggsave('weight_physical_activity.png')
```

```
## Saving 7 x 5 in image
```

From the chart above, we can infer that users weighing around **60kg & 85kg** are the most active.

We will carry out descriptive analysis to observe how many overweight & healthy users by using the following

```
#The amount of healthy users
nrow(filter(distinct(weight_log, id, .keep_all = T),bmi2 == 'Healthy'))
```

```
## [1] 3
```

```
#The amount of underweight users
nrow(filter(distinct(weight_log, id, .keep_all = T),bmi2 == 'Underweight'))
```

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

```
#The amount of overweight users
nrow(filter(distinct(weight_log, id, .keep_all = T),bmi2 == 'Overweight'))
```

```
## [1] 5
```

Out of the 30 users, only 8 submitted their responses regarding weight. 5 users are overweight and only 3 are within the healthy BMI range of 18.5–24.9.