Capstone Project Google Analytics

Fakontis Christos

2025-05-06

Installing Packages

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

#install.packages("lubridate")

#install.packages("dplyr")

#install.packages("ggplot2")

#install.packages("skimr")

#install.packages("janitor")

#install.packages("here")
```

Loading Packages

```
library("tidyverse")

## Warning: package 'ggplot2' was built under R version 4.3.3

## Warning: package 'purrr' was built under R version 4.3.3

## Warning: package 'lubridate' was built under R version 4.3.3

## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✔ dplyr     1.1.4     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ ggplot2   3.5.2     ✔ tibble    3.2.1
## ✔ lubridate 1.9.4     ✔ tidyr     1.3.0
## ✔ purrr     1.0.4
## ── Conflicts ──────────────────────────────────────────
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force
all conflicts to become errors
```

```r
library("lubridate")

library("dplyr")

library("ggplot2")

library("skimr")

library("janitor")

## Warning: package 'janitor' was built under R version 4.3.3

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

library("here")

## Warning: package 'here' was built under R version 4.3.3

## here() starts at /Users/christosfacondis/Desktop/Google Capstone
```

Loading dataset

About the Dataset

This dataset was collected from thirty Fitbit users via a distributed survey conducted through Amazon Mechanical Turk between **March 12, 2016, and May 12, 2016**. The dataset is publicly available on Kaggle and was published by user Möbius.

- **Dataset link**: Fitbit Fitness Tracker Data on Kaggle
- **Format**: CSV files
- **Storage**: The data was downloaded and stored in a secure, organized folder structure.

Importing and Preparing the Dataset

The dataset will be imported into **RStudio Cloud** for analysis. After import, we will proceed with the following steps:

1. **View** the raw data structure
2. **Clean** the column names for consistency using `janitor::clean_names()`
3. **Format** the columns to appropriate types (e.g., date/time, numeric)
4. **Organize** the datasets to support exploratory data analysis

---

```
activity <- read_csv("/Users/christosfacondis/Desktop/Google
Capstone/data/fitbit2/dailyActivity_merged.csv")
```

```
## Rows: 940 Columns: 15
## ── Column specification
────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (1): ActivityDate
## dbl (14): Id, TotalSteps, TotalDistance, TrackerDistance,
LoggedActivitiesDi...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

```
sleep <- read_csv("/Users/christosfacondis/Desktop/Google
Capstone/data/fitbit2/sleepDay_merged.csv")
```

```
## Rows: 413 Columns: 5
## ── Column specification
────────────────────────────────────────────────────────
## Delimiter: ","
## chr (1): SleepDay
## dbl (4): Id, TotalSleepRecords, TotalMinutesAsleep, TotalTimeInBed
##
## 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.
```

```
steps <- read_csv("/Users/christosfacondis/Desktop/Google
Capstone/data/fitbit2/dailySteps_merged.csv")
```

```
## Rows: 940 Columns: 3
## ── Column specification
────────────────────────────────────────────────────────
## Delimiter: ","
## chr (1): ActivityDay
## dbl (2): Id, StepTotal
##
## 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.
```

```
weight <- read_csv("/Users/christosfacondis/Desktop/Google
Capstone/data/fitbit2/weightLogInfo_merged.csv")
```

```
## Rows: 67 Columns: 8
## ── Column specification
────────────────────────────────────────────────────────
## Delimiter: ","
## chr (1): Date
## dbl (6): Id, WeightKg, WeightPounds, Fat, BMI, LogId
## lgl (1): IsManualReport
##
## 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.
```

```
intensities <-  read_csv("/Users/christosfacondis/Desktop/Google
Capstone/data/fitbit2/dailyIntensities_merged.csv")
```

```
## Rows: 940 Columns: 10
## ── Column specification
────────────────────────────────────────────────────────
## Delimiter: ","
## chr (1): ActivityDay
## dbl (9): Id, SedentaryMinutes, LightlyActiveMinutes, FairlyActiveMinutes,
Ve...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

```
calories <-  read_csv("/Users/christosfacondis/Desktop/Google
Capstone/data/fitbit2/dailyCalories_merged.csv")
```

```
## Rows: 940 Columns: 3
## ── Column specification
────────────────────────────────────────────────────────
## Delimiter: ","
## chr (1): ActivityDay
## dbl (2): Id, Calories
##
## 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.
```

```
heartrate <- read_csv("/Users/christosfacondis/Desktop/Google
Capstone/data/fitbit2/heartrate_seconds_merged.csv")
```

```
## 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.
```

Preview of our data-sets

```
head(activity)

## # A tibble: 6 × 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(sleep)

## # A tibble: 6 × 5
##           Id SleepDay        TotalSleepRecords TotalMinutesAsleep
TotalTimeInBed
##        <dbl> <chr>                       <dbl>              <dbl>
<dbl>
## 1 1503960366 4/12/2016 12:0…                 1                327
346
## 2 1503960366 4/13/2016 12:0…                 2                384
407
## 3 1503960366 4/15/2016 12:0…                 1                412
442
## 4 1503960366 4/16/2016 12:0…                 2                340
367
## 5 1503960366 4/17/2016 12:0…                 1                700
712
## 6 1503960366 4/19/2016 12:0…                 1                304
320

head(steps)

## # A tibble: 6 × 3
##           Id ActivityDay StepTotal
##        <dbl> <chr>           <dbl>
## 1 1503960366 4/12/2016       13162
## 2 1503960366 4/13/2016       10735
## 3 1503960366 4/14/2016       10460
## 4 1503960366 4/15/2016        9762
## 5 1503960366 4/16/2016       12669
## 6 1503960366 4/17/2016        9705

head(weight)
```

```
## # A tibble: 6 × 8
##            Id Date       WeightKg WeightPounds   Fat   BMI IsManualReport
LogId
##         <dbl> <chr>         <dbl>        <dbl> <dbl> <dbl> <lgl>
<dbl>
## 1 1503960366 5/2/2016 …     52.6         116.    22  22.6 TRUE
1.46e12
## 2 1503960366 5/3/2016 …     52.6         116.    NA  22.6 TRUE
1.46e12
## 3 1927972279 4/13/2016…    134.          294.    NA  47.5 FALSE
1.46e12
## 4 2873212765 4/21/2016…     56.7         125.    NA  21.5 TRUE
1.46e12
## 5 2873212765 5/12/2016…     57.3         126.    NA  21.7 TRUE
1.46e12
## 6 4319703577 4/17/2016…     72.4         160.    25  27.5 TRUE
1.46e12
```

```
head(intensities)
```

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

```
head(calories)
```

```
## # A tibble: 6 × 3
##            Id ActivityDay Calories
##         <dbl> <chr>          <dbl>
## 1 1503960366 4/12/2016       1985
## 2 1503960366 4/13/2016       1797
## 3 1503960366 4/14/2016       1776
## 4 1503960366 4/15/2016       1745
## 5 1503960366 4/16/2016       1863
## 6 1503960366 4/17/2016       1728
```

```r
head(heartrate)
```

```
## # A tibble: 6 × 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
```

Clean datasets

Check for duplicates

```r
print(paste("Duplicate rows in activity:", sum(duplicated(activity))))
```

```
## [1] "Duplicate rows in activity: 0"
```

```r
print(paste("Duplicate rows in sleep:", sum(duplicated(sleep))))
```

```
## [1] "Duplicate rows in sleep: 3"
```

```r
print(paste("Duplicate rows in weight:", sum(duplicated(weight))))
```

```
## [1] "Duplicate rows in weight: 0"
```

```r
print(paste("Duplicate rows in calories:", sum(duplicated(calories))))
```

```
## [1] "Duplicate rows in calories: 0"
```

```r
print(paste("Duplicate rows in heartrate:", sum(duplicated(heartrate))))
```

```
## [1] "Duplicate rows in heartrate: 0"
```

```r
print(paste("Duplicate rows in intensities:", sum(duplicated(intensities))))
```

```
## [1] "Duplicate rows in intensities: 0"
```

```r
print(paste("Duplicate rows in steps:", sum(duplicated(steps))))
```

```
## [1] "Duplicate rows in steps: 0"
```

Check for missing values

```r
print(paste("Missing values in activity:", sum(is.na(activity))))
```

```
## [1] "Missing values in activity: 0"
```

```r
print(paste("Missing values in sleep:", sum(is.na(sleep))))
```

```
## [1] "Missing values in sleep: 0"
```

```r
print(paste("Missing values in calories:", sum(is.na(calories))))

## [1] "Missing values in calories: 0"

print(paste("Missing values in heartrate:", sum(is.na(heartrate))))

## [1] "Missing values in heartrate: 0"

print(paste("Missing values in intensities:", sum(is.na(intensities))))

## [1] "Missing values in intensities: 0"

print(paste("Missing values in weight:", sum(is.na(weight))))

## [1] "Missing values in weight: 65"

print(paste("Missing values in steps:", sum(is.na(steps))))

## [1] "Missing values in steps: 0"
```

Cleaning initialization

Activity data-set cleaning

```r
activity <- clean_names(activity) %>% # Clean column names
  mutate(id = as.character(id)) %>% # from double to chr
  mutate(activity_date = as.Date(activity_date, format = "%m/%d/%Y")) %>%
  distinct()   # Remove duplicate rows

# Check activity data-set after cleaning
glimpse(activity)

## Rows: 940
## Columns: 15
## $ id                         <chr> "1503960366", "1503960366",
"1503960366", "…
## $ activity_date              <date> 2016-04-12, 2016-04-13, 2016-04-14,
2016-0…
## $ total_steps                <dbl> 13162, 10735, 10460, 9762, 12669, 9705,
130…
## $ total_distance             <dbl> 8.50, 6.97, 6.74, 6.28, 8.16, 6.48,
8.59, 9…
## $ tracker_distance           <dbl> 8.50, 6.97, 6.74, 6.28, 8.16, 6.48,
8.59, 9…
## $ logged_activities_distance <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0…
## $ very_active_distance       <dbl> 1.88, 1.57, 2.44, 2.14, 2.71, 3.19,
3.25, 3…
## $ moderately_active_distance <dbl> 0.55, 0.69, 0.40, 1.26, 0.41, 0.78,
0.64, 1…
## $ light_active_distance      <dbl> 6.06, 4.71, 3.91, 2.83, 5.04, 2.51,
4.71, 5…
## $ sedentary_active_distance  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0…
## $ very_active_minutes        <dbl> 25, 21, 30, 29, 36, 38, 42, 50, 28, 19,
66,…
## $ fairly_active_minutes      <dbl> 13, 19, 11, 34, 10, 20, 16, 31, 12, 8,
27, …
## $ lightly_active_minutes     <dbl> 328, 217, 181, 209, 221, 164, 233, 264,
205…
## $ sedentary_minutes          <dbl> 728, 776, 1218, 726, 773, 539, 1149,
775, 8…
## $ calories                   <dbl> 1985, 1797, 1776, 1745, 1863, 1728,
1921, 2…
```

Activity data-set more exploration

Print summary statistics to have a better idea of the dataset

```
summary(activity)
```

```
##       id            activity_date         total_steps    total_distance
##  Length:940         Min.   :2016-04-12   Min.   :    0   Min.   : 0.000
##  Class :character   1st Qu.:2016-04-19   1st Qu.: 3790   1st Qu.: 2.620
##  Mode  :character   Median :2016-04-26   Median : 7406   Median : 5.245
##                     Mean   :2016-04-26   Mean   : 7638   Mean   : 5.490
##                     3rd Qu.:2016-05-04   3rd Qu.:10727   3rd Qu.: 7.713
##                     Max.   :2016-05-12   Max.   :36019   Max.   :28.030
##  tracker_distance logged_activities_distance very_active_distance
##  Min.   : 0.000   Min.   :0.0000             Min.   : 0.000
##  1st Qu.: 2.620   1st Qu.:0.0000             1st Qu.: 0.000
##  Median : 5.245   Median :0.0000             Median : 0.210
##  Mean   : 5.475   Mean   :0.1082             Mean   : 1.503
##  3rd Qu.: 7.710   3rd Qu.:0.0000             3rd Qu.: 2.053
##  Max.   :28.030   Max.   :4.9421             Max.   :21.920
##  moderately_active_distance light_active_distance
sedentary_active_distance
##  Min.   :0.0000             Min.   : 0.000        Min.   :0.000000
##  1st Qu.:0.0000             1st Qu.: 1.945        1st Qu.:0.000000
##  Median :0.2400             Median : 3.365        Median :0.000000
##  Mean   :0.5675             Mean   : 3.341        Mean   :0.001606
##  3rd Qu.:0.8000             3rd Qu.: 4.782        3rd Qu.:0.000000
##  Max.   :6.4800             Max.   :10.710        Max.   :0.110000
##  very_active_minutes fairly_active_minutes lightly_active_minutes
##  Min.   :  0.00      Min.   :  0.00        Min.   :  0.0
##  1st Qu.:  0.00      1st Qu.:  0.00        1st Qu.:127.0
##  Median :  4.00      Median :  6.00        Median :199.0
##  Mean   : 21.16      Mean   : 13.56        Mean   :192.8
##  3rd Qu.: 32.00      3rd Qu.: 19.00        3rd Qu.:264.0
##  Max.   :210.00      Max.   :143.00        Max.   :518.0
##  sedentary_minutes    calories
##  Min.   :   0.0    Min.   :   0
##  1st Qu.: 729.8    1st Qu.:1828
##  Median :1057.5    Median :2134
##  Mean   : 991.2   Mean   :2304  
##  3rd Qu.:1229.5   3rd Qu.:2793  
##  Max.   :1440.0   Max.   :4900  
```

Observation: Some attributes have a minimum value of zero (total_step, total_distance, calories etc.). That is impossible and we need to explore more our data-set.

```
filter(activity, total_steps == 0) # Check each entry where total_steps is zero

## # A tibble: 77 × 15
##    id         activity_date total_steps total_distance tracker_distance
##    <chr>      <date>              <dbl>          <dbl>            <dbl>
##  1 1503960366 2016-05-12              0              0                0
##  2 1844505072 2016-04-24              0              0                0
##  3 1844505072 2016-04-25              0              0                0
##  4 1844505072 2016-04-26              0              0                0
##  5 1844505072 2016-05-02              0              0                0
##  6 1844505072 2016-05-07              0              0                0
##  7 1844505072 2016-05-08              0              0                0
##  8 1844505072 2016-05-09              0              0                0
##  9 1844505072 2016-05-10              0              0                0
## 10 1844505072 2016-05-11              0              0                0
## # ℹ 67 more rows
## # ℹ 10 more variables: logged_activities_distance <dbl>,
## #   very_active_distance <dbl>, moderately_active_distance <dbl>,
## #   light_active_distance <dbl>, sedentary_active_distance <dbl>,
## #   very_active_minutes <dbl>, fairly_active_minutes <dbl>,
## #   lightly_active_minutes <dbl>, sedentary_minutes <dbl>, calories <dbl>
```

```
filter(activity, total_distance == 0) # Check each entry where total_distance is zero

## # A tibble: 78 × 15
##    id         activity_date total_steps total_distance tracker_distance
##    <chr>      <date>              <dbl>          <dbl>            <dbl>
##  1 1503960366 2016-05-12              0              0                0
##  2 1844505072 2016-04-24              0              0                0
##  3 1844505072 2016-04-25              0              0                0
##  4 1844505072 2016-04-26              0              0                0
##  5 1844505072 2016-04-27              4              0                0
##  6 1844505072 2016-05-02              0              0                0
##  7 1844505072 2016-05-07              0              0                0
##  8 1844505072 2016-05-08              0              0                0
##  9 1844505072 2016-05-09              0              0                0
## 10 1844505072 2016-05-10              0              0                0
## # ℹ 68 more rows
## # ℹ 10 more variables: logged_activities_distance <dbl>,
## #   very_active_distance <dbl>, moderately_active_distance <dbl>,
## #   light_active_distance <dbl>, sedentary_active_distance <dbl>,
## #   very_active_minutes <dbl>, fairly_active_minutes <dbl>,
## #   lightly_active_minutes <dbl>, sedentary_minutes <dbl>, calories <dbl>
```

```
filter(activity, calories == 0) # Check each entry where calories is zero
```

```
## # A tibble: 4 × 15
##   id         activity_date total_steps total_distance tracker_distance
##   <chr>      <date>              <dbl>          <dbl>            <dbl>
## 1 1503960366 2016-05-12              0              0                0
## 2 6290855005 2016-05-10              0              0                0
## 3 8253242879 2016-04-30              0              0                0
## 4 8583815059 2016-05-12              0              0                0
## # ℹ 10 more variables: logged_activities_distance <dbl>,
## #   very_active_distance <dbl>, moderately_active_distance <dbl>,
## #   light_active_distance <dbl>, sedentary_active_distance <dbl>,
## #   very_active_minutes <dbl>, fairly_active_minutes <dbl>,
## #   lightly_active_minutes <dbl>, sedentary_minutes <dbl>, calories <dbl>
```

```
filter(activity, tracker_distance == 0) # Check each entry where
tracker_distance is zero
```

```
## # A tibble: 78 × 15
##    id         activity_date total_steps total_distance tracker_distance
##    <chr>      <date>              <dbl>          <dbl>            <dbl>
##  1 1503960366 2016-05-12              0              0                0
##  2 1844505072 2016-04-24              0              0                0
##  3 1844505072 2016-04-25              0              0                0
##  4 1844505072 2016-04-26              0              0                0
##  5 1844505072 2016-04-27              4              0                0
##  6 1844505072 2016-05-02              0              0                0
##  7 1844505072 2016-05-07              0              0                0
##  8 1844505072 2016-05-08              0              0                0
##  9 1844505072 2016-05-09              0              0                0
## 10 1844505072 2016-05-10              0              0                0
## # ℹ 68 more rows
## # ℹ 10 more variables: logged_activities_distance <dbl>,
## #   very_active_distance <dbl>, moderately_active_distance <dbl>,
## #   light_active_distance <dbl>, sedentary_active_distance <dbl>,
## #   very_active_minutes <dbl>, fairly_active_minutes <dbl>,
## #   lightly_active_minutes <dbl>, sedentary_minutes <dbl>, calories <dbl>
```

There are 77 entries where the **TotalSteps** value is recorded as zero, 78 entries of **total_distance** as zero and 4 entries of calories as zero . These likely represent instances when the user did not wear their Fitbit device, rather than actual inactivity. To ensure the accuracy of our analysis—particularly when calculating metrics like the mean and median—we will remove these records from the dataset.

```
activity_clean <- filter(activity,
         total_steps != 0,
         total_distance != 0,
         calories != 0)

print(activity_clean)

## # A tibble: 862 × 15
##    id         activity_date total_steps total_distance tracker_distance
##    <chr>      <date>              <dbl>          <dbl>            <dbl>
##  1 1503960366 2016-04-12          13162           8.5              8.5 
##  2 1503960366 2016-04-13          10735           6.97             6.97
##  3 1503960366 2016-04-14          10460           6.74             6.74
##  4 1503960366 2016-04-15           9762           6.28             6.28
##  5 1503960366 2016-04-16          12669           8.16             8.16
##  6 1503960366 2016-04-17           9705           6.48             6.48
##  7 1503960366 2016-04-18          13019           8.59             8.59
##  8 1503960366 2016-04-19          15506           9.88             9.88
##  9 1503960366 2016-04-20          10544           6.68             6.68
## 10 1503960366 2016-04-21           9819           6.34             6.34
## # ℹ 852 more rows
## # ℹ 10 more variables: logged_activities_distance <dbl>,
## #   very_active_distance <dbl>, moderately_active_distance <dbl>,
## #   light_active_distance <dbl>, sedentary_active_distance <dbl>,
## #   very_active_minutes <dbl>, fairly_active_minutes <dbl>,
## #   lightly_active_minutes <dbl>, sedentary_minutes <dbl>, calories <dbl>
```

Check the data-set again

```
# Summary before removing zero step records
print("Summary BEFORE removing entries with 0:")

## [1] "Summary BEFORE removing entries with 0:"

print(summary(activity[,c("total_steps", "total_distance","calories")]))

##   total_steps    total_distance      calories   
##  Min.   :    0   Min.   : 0.000   Min.   :   0  
##  1st Qu.: 3790   1st Qu.: 2.620   1st Qu.:1828  
##  Median : 7406   Median : 5.245   Median :2134  
##  Mean   : 7638   Mean   : 5.490   Mean   :2304  
##  3rd Qu.:10727   3rd Qu.: 7.713   3rd Qu.:2793  
##  Max.   :36019   Max.   :28.030   Max.   :4900  

# Summary after removing zero step records
print("Summary AFTER removing entries with 0:")

## [1] "Summary AFTER removing entries with 0:"

print(summary(activity_clean[,c("total_steps",
"total_distance","calories")]))

##   total_steps    total_distance      calories   
##  Min.   :    8   Min.   : 0.010   Min.   :  52  
##  1st Qu.: 4927   1st Qu.: 3.373   1st Qu.:1857  
##  Median : 8054   Median : 5.590   Median :2220
##  Mean   : 8329   Mean   : 5.986   Mean   :2362
##  3rd Qu.:11096   3rd Qu.: 7.905   3rd Qu.:2832
##  Max.   :36019   Max.   :28.030   Max.   :4900
```

Sleep data-set cleaning

```
sleep_clean <-  clean_names(sleep) %>%   # Clean column names
  mutate(id = as.character(id)) %>% # from double to chr
  mutate(sleep_day = as.Date(sleep_day,
                             format = "%m/%d/%Y")) %>% # from chr to date
  distinct() #remove dublicates

# Check clean daily_sleep dataset
glimpse(sleep_clean)
```

```
## Rows: 410
## Columns: 5
## $ id                   <chr> "1503960366", "1503960366", "1503960366",
"150396…
## $ sleep_day            <date> 2016-04-12, 2016-04-13, 2016-04-15, 2016-04-
16, …
## $ total_sleep_records  <dbl> 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1…
## $ total_minutes_asleep <dbl> 327, 384, 412, 340, 700, 304, 360, 325, 361,
430,…
## $ total_time_in_bed    <dbl> 346, 407, 442, 367, 712, 320, 377, 364, 384,
449,…
```

```
# more exploration
summary(sleep_clean)
```

```
##       id              sleep_day          total_sleep_records
##  Length:410         Min.   :2016-04-12   Min.   :1.00
##  Class :character   1st Qu.:2016-04-19   1st Qu.:1.00
##  Mode  :character   Median :2016-04-27   Median :1.00
##                     Mean   :2016-04-26   Mean   :1.12
##                     3rd Qu.:2016-05-04   3rd Qu.:1.00
##                     Max.   :2016-05-12   Max.   :3.00
##  total_minutes_asleep total_time_in_bed
##  Min.   : 58.0        Min.   : 61.0
##  1st Qu.:361.0        1st Qu.:403.8
##  Median :432.5        Median :463.0
##  Mean   :419.2        Mean   :458.5
##  3rd Qu.:490.0        3rd Qu.:526.0
##  Max.   :796.0        Max.   :961.0
```

Calories data-set cleaning

```
calories_clean <-  clean_names(calories) %>%   # Clean column names
  mutate(id = as.character(id)) %>% # from double to chr
  mutate(activity_day = as.Date(activity_day,
                             format = "%m/%d/%Y")) %>% # from chr to date
  distinct() #remove dublicates

# Check clean daily_sleep dataset
glimpse(sleep_clean)

## Rows: 410
## Columns: 5
## $ id                   <chr> "1503960366", "1503960366", "1503960366",
"150396…
## $ sleep_day            <date> 2016-04-12, 2016-04-13, 2016-04-15, 2016-04-
16, …
## $ total_sleep_records  <dbl> 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1…
## $ total_minutes_asleep <dbl> 327, 384, 412, 340, 700, 304, 360, 325, 361,
430,…
## $ total_time_in_bed    <dbl> 346, 407, 442, 367, 712, 320, 377, 364, 384,
449,…

# more exploration
summary(sleep_clean)

##       id               sleep_day          total_sleep_records
##  Length:410         Min.   :2016-04-12   Min.   :1.00
##  Class :character   1st Qu.:2016-04-19   1st Qu.:1.00
##  Mode  :character   Median :2016-04-27   Median :1.00
##                     Mean   :2016-04-26   Mean   :1.12
##                     3rd Qu.:2016-05-04   3rd Qu.:1.00
##                     Max.   :2016-05-12   Max.   :3.00
##  total_minutes_asleep total_time_in_bed
##  Min.   : 58.0        Min.   : 61.0
##  1st Qu.:361.0        1st Qu.:403.8
##  Median :432.5        Median :463.0
##  Mean   :419.2        Mean   :458.5
##  3rd Qu.:490.0        3rd Qu.:526.0
##  Max.   :796.0        Max.   :961.0
```

Heartrate data-set cleaning

```
heartrate_clean <- clean_names(heartrate) %>%   # Clean column names
  mutate(id = as.character(id)) %>% # from double to chr
  mutate(time = as_datetime(time,
                            format = "%m/%d/%Y %I:%M:%S %p")) %>% # from chr
to datetime
  rename(date_time = time,
         heart_rate = value) %>%   # Rename columns
  distinct()   # Remove duplicate rows

glimpse(heartrate_clean)

## Rows: 2,483,658
## Columns: 3
## $ id         <chr> "2022484408", "2022484408", "2022484408", "2022484408",
"20…
## $ date_time  <dttm> 2016-04-12 07:21:00, 2016-04-12 07:21:05, 2016-04-12
07:21…
## $ heart_rate <dbl> 97, 102, 105, 103, 101, 95, 91, 93, 94, 93, 92, 89, 83,
61,…
```

Steps data-set cleaning

```
steps_clean <-  clean_names(steps) %>%   # Clean column names
  mutate(id = as.character(id)) %>% # from double to chr
  mutate(activity_day = as.Date(activity_day,
                             format = "%m/%d/%Y")) %>% # from chr to date
  distinct() #remove dublicates

# Check clean daily_sleep dataset
glimpse(steps_clean)

## Rows: 940
## Columns: 3
## $ id           <chr> "1503960366", "1503960366", "1503960366",
"1503960366", "…
## $ activity_day <date> 2016-04-12, 2016-04-13, 2016-04-14, 2016-04-15,
2016-04-…
## $ step_total   <dbl> 13162, 10735, 10460, 9762, 12669, 9705, 13019, 15506,
105…

# more exploration
summary(steps_clean)

##       id             activity_day          step_total
##  Length:940         Min.   :2016-04-12   Min.   :    0
##  Class :character   1st Qu.:2016-04-19   1st Qu.: 3790
##  Mode  :character   Median :2016-04-26   Median : 7406
##                     Mean   :2016-04-26   Mean   : 7638
##                     3rd Qu.:2016-05-04   3rd Qu.:10727
##                     Max.   :2016-05-12   Max.   :36019
```

Weight data-set cleaning

```
weight_clean <- clean_names(weight) %>%   # Clean column names
  mutate(id = as.character(id)) %>% # from double to chr
  mutate(date = as_datetime(date,
                            format = "%m/%d/%Y %I:%M:%S %p")) %>% # from chr
to datetime
  rename(date_time = date) %>% # Rename columns
  # Remove duplicate rows
  distinct()

# Change NA to 0 in the column "fat"
weight_clean$fat[is.na(weight_clean$fat)] <- 0
```

```
# Check clean daily_activity dataset
glimpse(weight_clean)

## Rows: 67
## Columns: 8
## $ id               <chr> "1503960366", "1503960366", "1927972279",
"2873212765…
## $ date_time        <dttm> 2016-05-02 23:59:59, 2016-05-03 23:59:59, 2016-
04-13…
## $ weight_kg        <dbl> 52.6, 52.6, 133.5, 56.7, 57.3, 72.4, 72.3, 69.7,
70.3…
## $ weight_pounds    <dbl> 115.9631, 115.9631, 294.3171, 125.0021, 126.3249,
159…
## $ fat              <dbl> 22, 0, 0, 0, 0, 25, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, …
## $ bmi              <dbl> 22.65, 22.65, 47.54, 21.45, 21.69, 27.45, 27.38,
27.2…
## $ is_manual_report <lgl> TRUE, TRUE, FALSE, TRUE, TRUE, TRUE, TRUE, TRUE,
TRUE…
## $ log_id           <dbl> 1.462234e+12, 1.462320e+12, 1.460510e+12,
1.461283e+1…
```

Intensitie data-set cleaning

```
intensities_clean <- clean_names(intensities) %>%   # Clean column names
  mutate(id = as.character(id)) %>% # from double to chr
  mutate(activity_day = as.Date(activity_day,
                           format = "%m/%d/%Y")) %>% # from chr to date
  distinct() #remove dublicates


# Check clean daily_activity dataset
glimpse(intensities_clean)

## Rows: 940
## Columns: 10
## $ id                         <chr> "1503960366", "1503960366",
"1503960366", "…
## $ activity_day               <date> 2016-04-12, 2016-04-13, 2016-04-14,
2016-0…
## $ sedentary_minutes          <dbl> 728, 776, 1218, 726, 773, 539, 1149,
775, 8…
## $ lightly_active_minutes     <dbl> 328, 217, 181, 209, 221, 164, 233, 264,
205…
## $ fairly_active_minutes      <dbl> 13, 19, 11, 34, 10, 20, 16, 31, 12, 8,
27, …
## $ very_active_minutes        <dbl> 25, 21, 30, 29, 36, 38, 42, 50, 28, 19,
66,…
## $ sedentary_active_distance  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0…
## $ light_active_distance      <dbl> 6.06, 4.71, 3.91, 2.83, 5.04, 2.51,
4.71, 5…
## $ moderately_active_distance <dbl> 0.55, 0.69, 0.40, 1.26, 0.41, 0.78,
0.64, 1…
## $ very_active_distance       <dbl> 1.88, 1.57, 2.44, 2.14, 2.71, 3.19,
3.25, 3…
```

Unique IDs per data-set

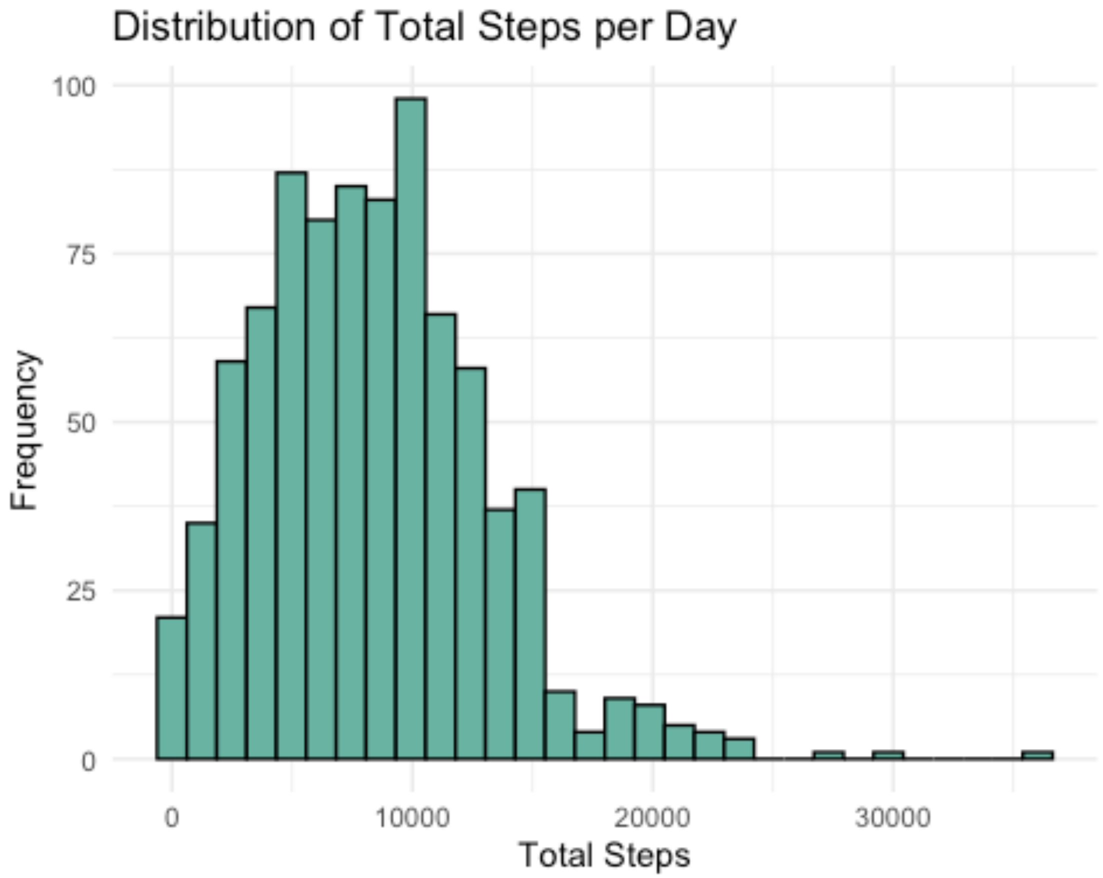
```
datasets <- c(
  "activity_clean",
  "sleep_clean",
  "steps_clean",
  "calories_clean",
  "heartrate_clean",
  "weight_clean",
  "intensities_clean"
)
# Empty data frame
distinct_id_summary <- data.frame(
  Dataset = character(),
  Distinct_IDs = integer(),
  stringsAsFactors = FALSE
)

# Loop through each dataset, calculate distinct IDs and store the result
for (name in datasets) {
  data <- get(name)
  num_ids <- n_distinct(data$id)
  distinct_id_summary <- bind_rows(
    distinct_id_summary,
    data.frame(Dataset = name, Distinct_IDs = num_ids)
  )
}

distinct_id_summary <- distinct_id_summary %>%
  arrange(desc(Distinct_IDs))

print(distinct_id_summary)
```

```
##             Dataset Distinct_IDs
## 1    activity_clean           33
## 2       steps_clean           33
## 3    calories_clean           33
## 4 intensities_clean           33
## 5       sleep_clean           24
## 6   heartrate_clean           14
## 7      weight_clean            8
```

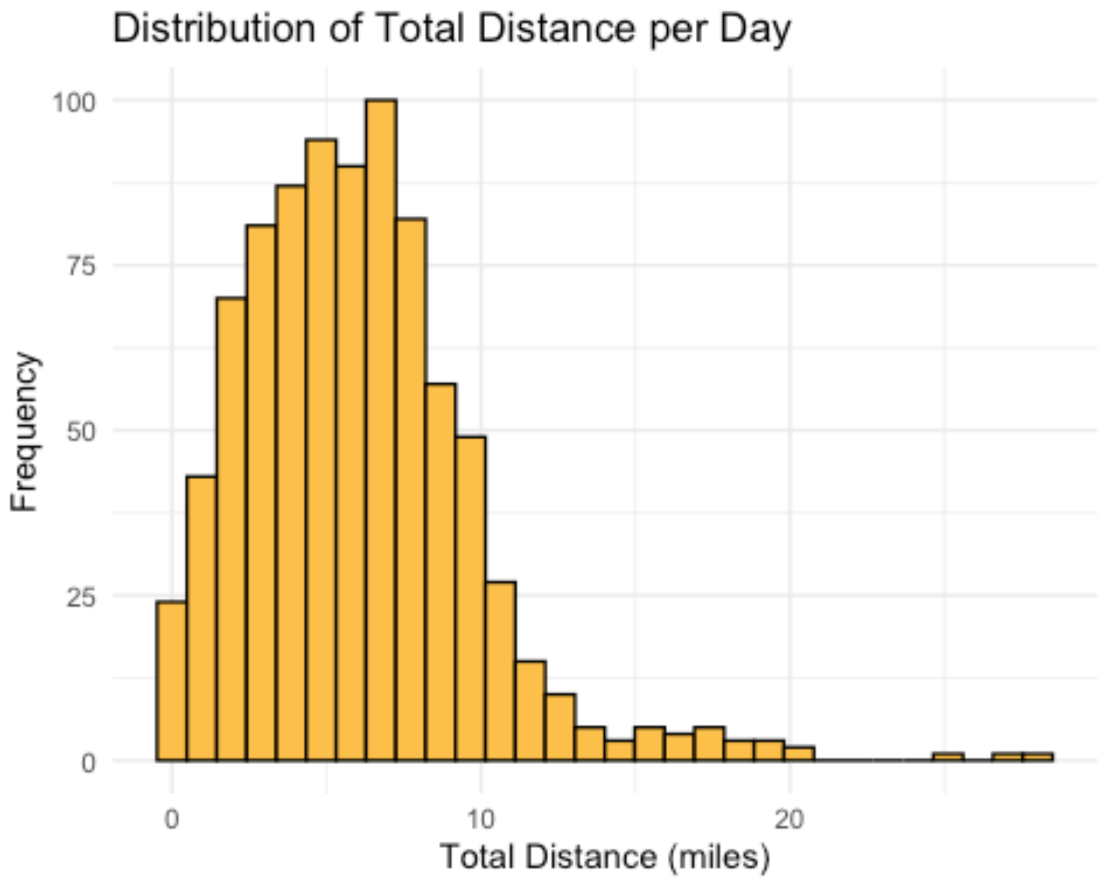
Observations: The variation in the number of unique user IDs across the datasets suggests possible gaps in data collection or that not all users engaged equally with every feature. This might reflect inconsistent usage patterns or missing data in certain areas.

Analyze Activity data-set

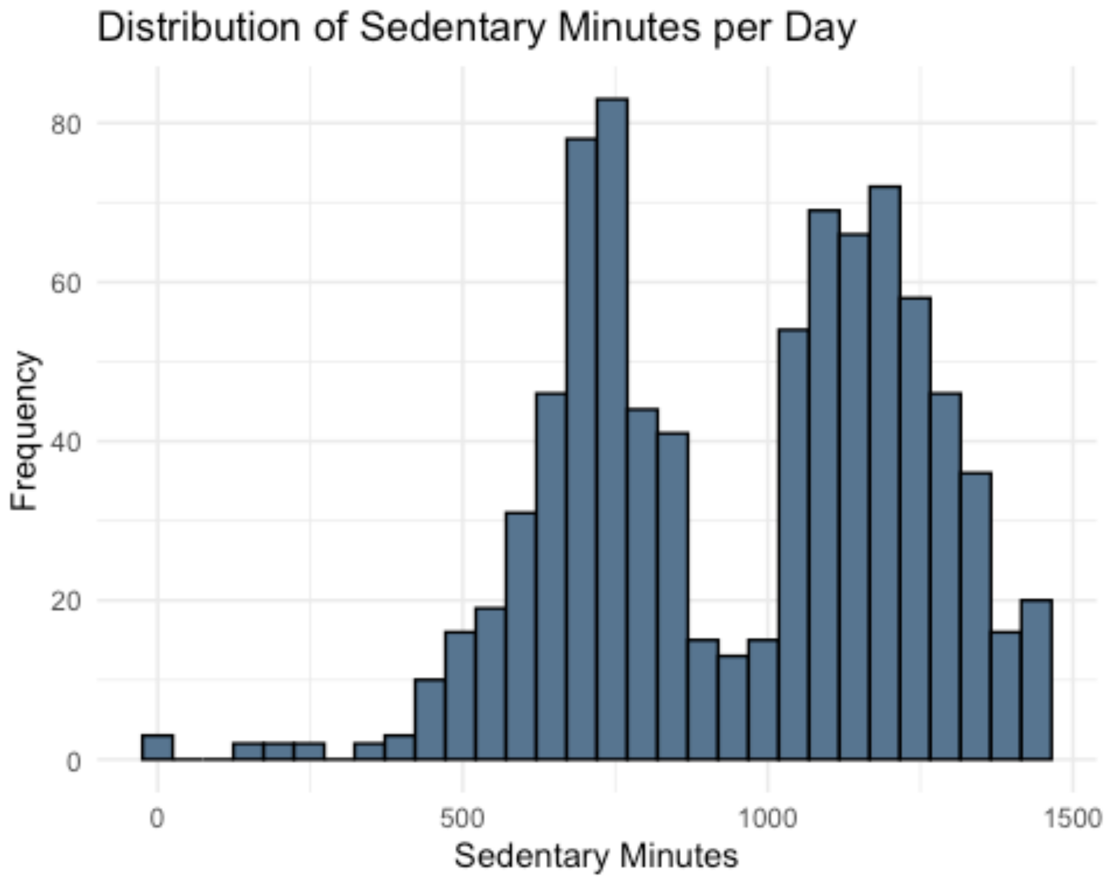
```
# Activity
activity_clean %>%
  select(total_steps,
         total_distance,
         sedentary_minutes, calories) %>%
  summary()

##   total_steps    total_distance   sedentary_minutes    calories
##  Min.   :    8   Min.    : 0.010   Min.    :   0.0    Min.   :  52
##  1st Qu.: 4927   1st Qu.: 3.373   1st Qu.: 721.2    1st Qu.:1857
##  Median : 8054   Median : 5.590   Median :1020.5    Median :2220
##  Mean   : 8329   Mean    : 5.986   Mean    : 955.2    Mean   :2362
##  3rd Qu.:11096   3rd Qu.: 7.905   3rd Qu.:1189.0    3rd Qu.:2832
##  Max.   :36019   Max.    :28.030   Max.    :1440.0    Max.   :4900

# Histogram for Total Steps
ggplot(activity_clean, aes(x = total_steps)) +
  geom_histogram(fill = "#69b3a2", bins = 30, color = "black") +
  labs(title = "Distribution of Total Steps per Day",
       x = "Total Steps",
       y = "Frequency") +
  theme_minimal()
```



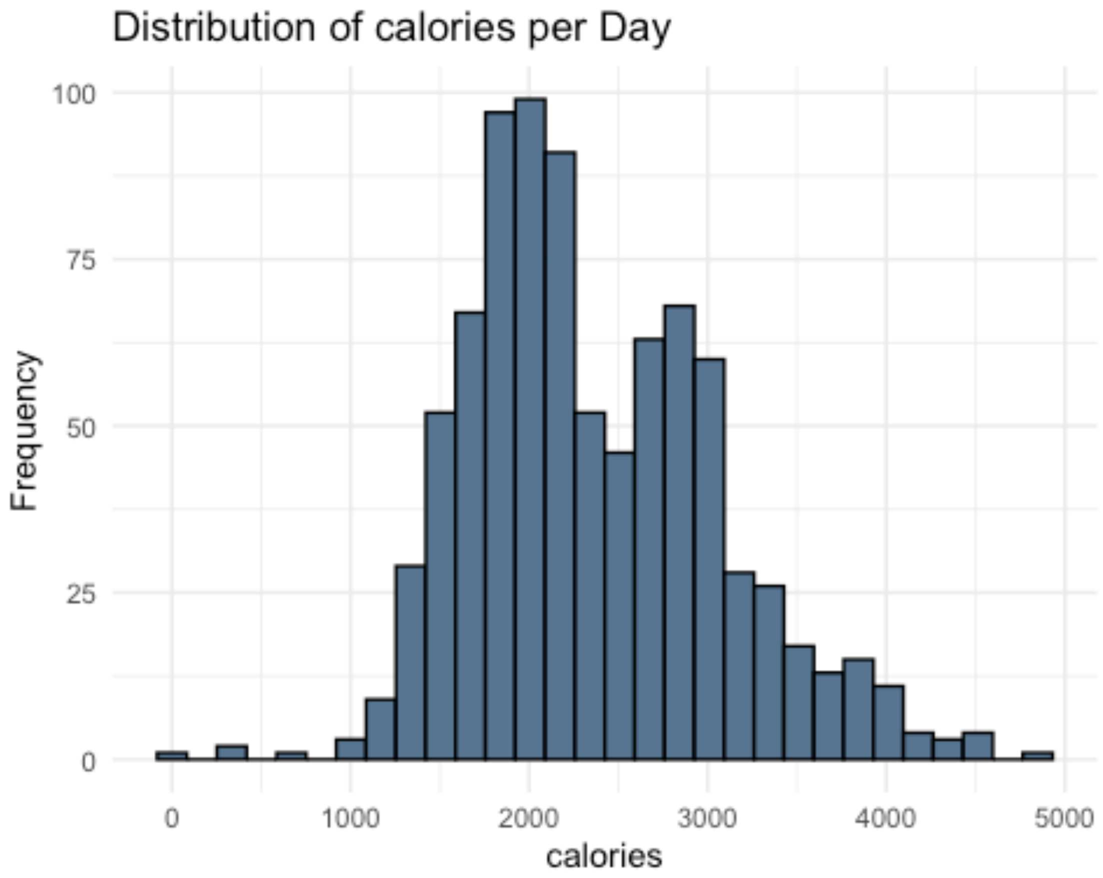
```r
# Histogram for Total Distance
ggplot(activity_clean, aes(x = total_distance)) +
  geom_histogram(fill = "#fcbf49", bins = 30, color = "black") +
  labs(title = "Distribution of Total Distance per Day",
       x = "Total Distance (miles)",
       y = "Frequency") +
  theme_minimal()
```



```r
# Histogram for Sedentary Minutes
ggplot(activity_clean, aes(x = sedentary_minutes)) +
  geom_histogram(fill = "#577590", bins = 30, color = "black") +
  labs(title = "Distribution of Sedentary Minutes per Day",
       x = "Sedentary Minutes",
       y = "Frequency") +
  theme_minimal()
```



```r
# Histogram for calories
ggplot(activity_clean, aes(x = calories)) +
  geom_histogram(fill = "#577590", bins = 30, color = "black") +
  labs(title = "Distribution of calories per Day",
       x = "calories",
       y = "Frequency") +
  theme_minimal()
```



Summary Statistics of Key Activity Metrics

After cleaning the `daily_activity` dataset, I reviewed the core variables to understand user behavior and daily patterns. Below is a summary of key metrics:

- **Total Steps**
  - Range: *8 steps* to *36,019 steps* per day
  - Median: **8,054**
  - Mean: **8,329**
  - Insight: Slightly right-skewed distribution, indicating some users walk significantly more than average
- **Total Distance (miles)**
  - Range: *0.01* to *28.03 miles*
  - Median: **5.59 miles**
  - Mean: **5.99 miles**

  - Insight: Daily movement varies, with most users covering between 3–8 miles
- **Sedentary Minutes**
  - Range: *0* to *1,440 minutes*

  - Median: **1,020.5 minutes** (~17 hours)

  - Mean: **955.2 minutes**

  - Insight: Users are generally sedentary for most of the day
- **Calories Burned**
  - Range: *52* to *4,900 calories*

  - Median: **2,220 calories**

  - Mean: **2,362 calories**

  - Insight: Energy expenditure is fairly normally distributed, with outliers on high-activity days

These statistics offer a foundational understanding of Fitbit user habits and provide behavioral trends that can inform Bellabeat's product and marketing strategy.

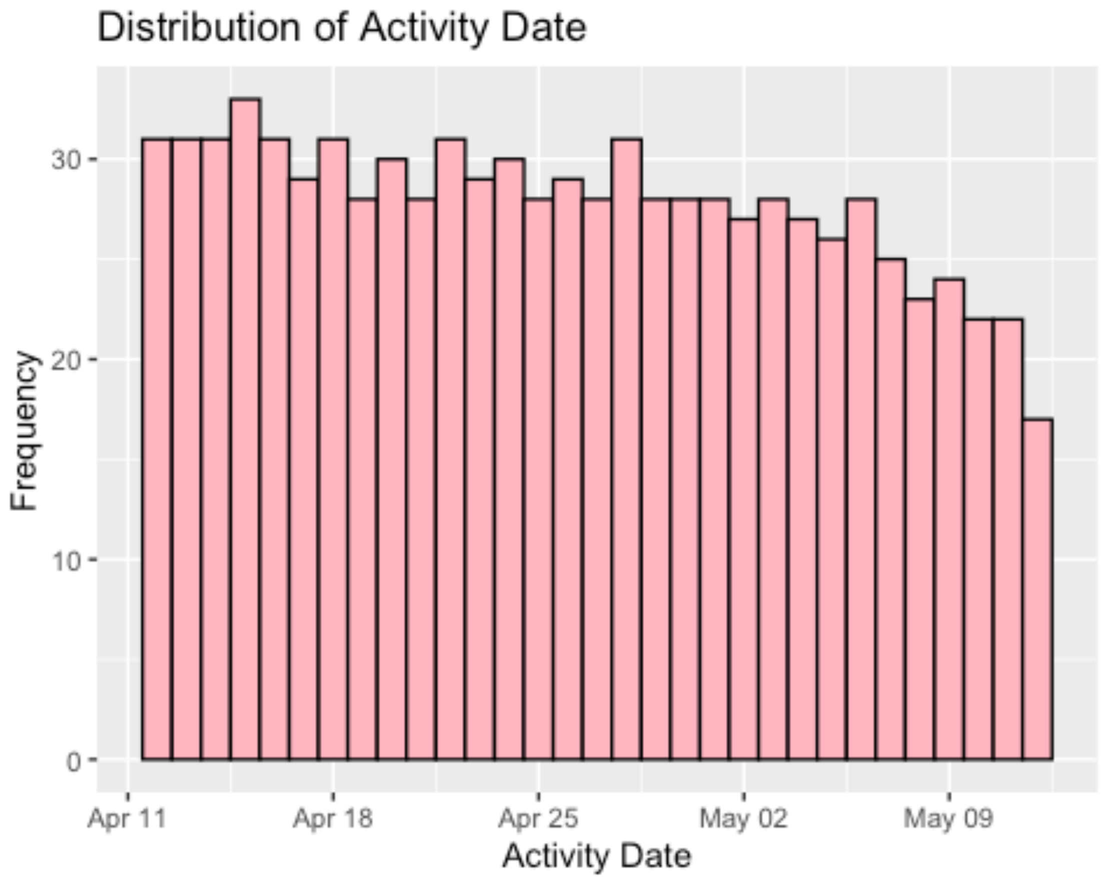
Observations from Histogram Analysis

- The distributions of most variables are **right-skewed**, meaning most values are concentrated on the lower end with a few high outliers.

- **`total_steps`** and **`total_distance`**have similar distribution shapes, indicating a possible correlation worth analyzing further.

- Since the data is **not normally distributed**, the **median** serves as a more reliable measure of central tendency than the mean

```r
#Distributiuon of Activity Date
ggplot(data=activity_clean , aes(x = activity_date)) +
  geom_histogram(binwidth = 1, color = "black", fill = "lightpink") +
  labs(x = "Activity Date", y = "Frequency", title = "Distribution of
Activity Date")
```



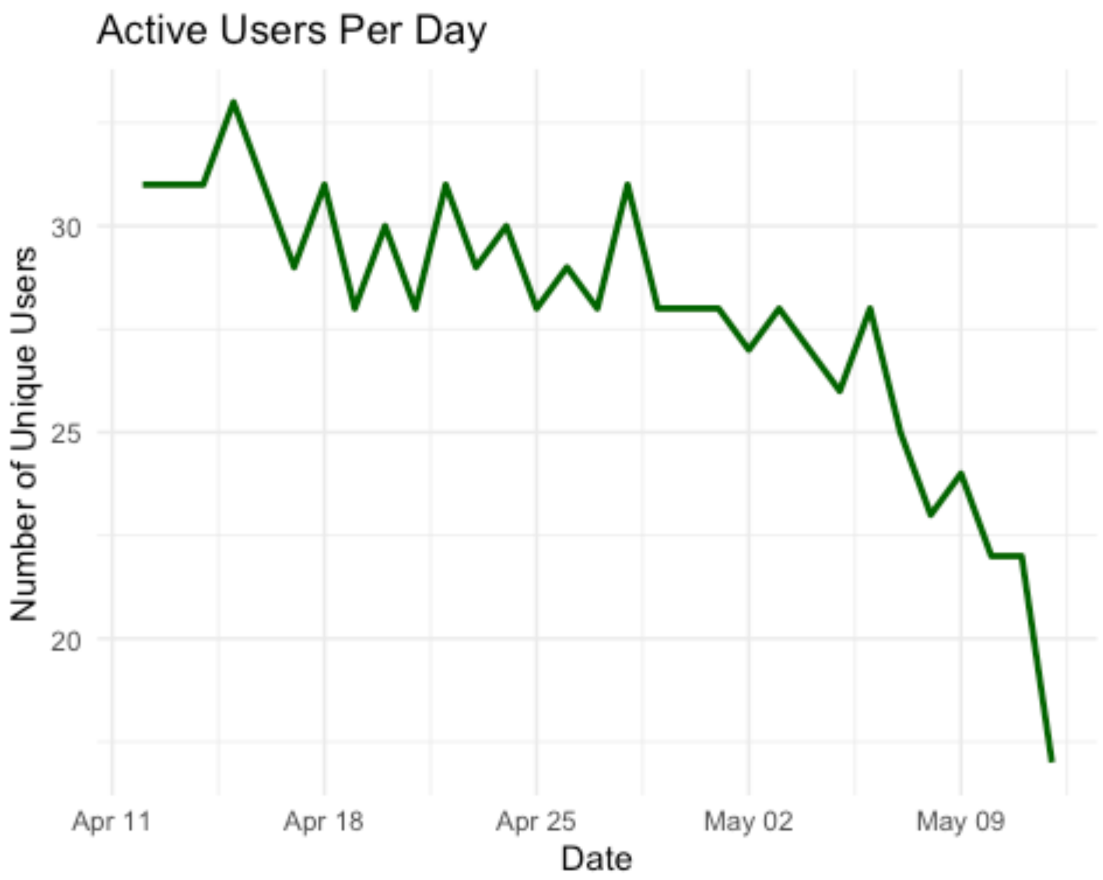
Observations

Toward the end of the data collection period — particularly in early May — we observe a noticeable decline in recorded activity. This reduction in entries may indicate a drop in user engagement or possible non-usage of the tracking devices.

It’s also plausible that seasonal factors, such as the beginning of summer, influenced user routines, leading to reduced activity tracking during this period.

```
users_per_day <- activity_clean %>%
  group_by(activity_date) %>%
  summarise(active_users = n_distinct(id))

ggplot(users_per_day, aes(x = activity_date, y = active_users)) +
  geom_line(color = "darkgreen", size = 1) +
  labs(title = "Active Users Per Day",
       x = "Date",
       y = "Number of Unique Users") +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

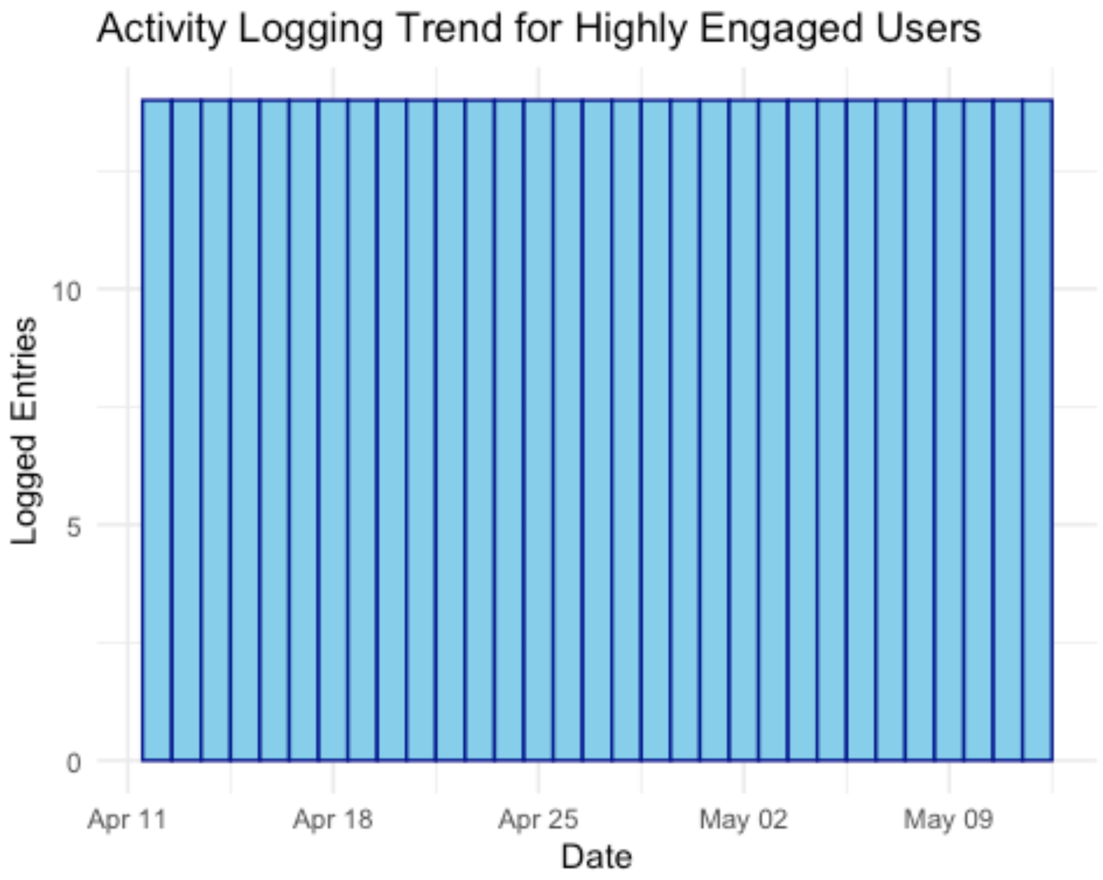


Observations on User Activity Over Time

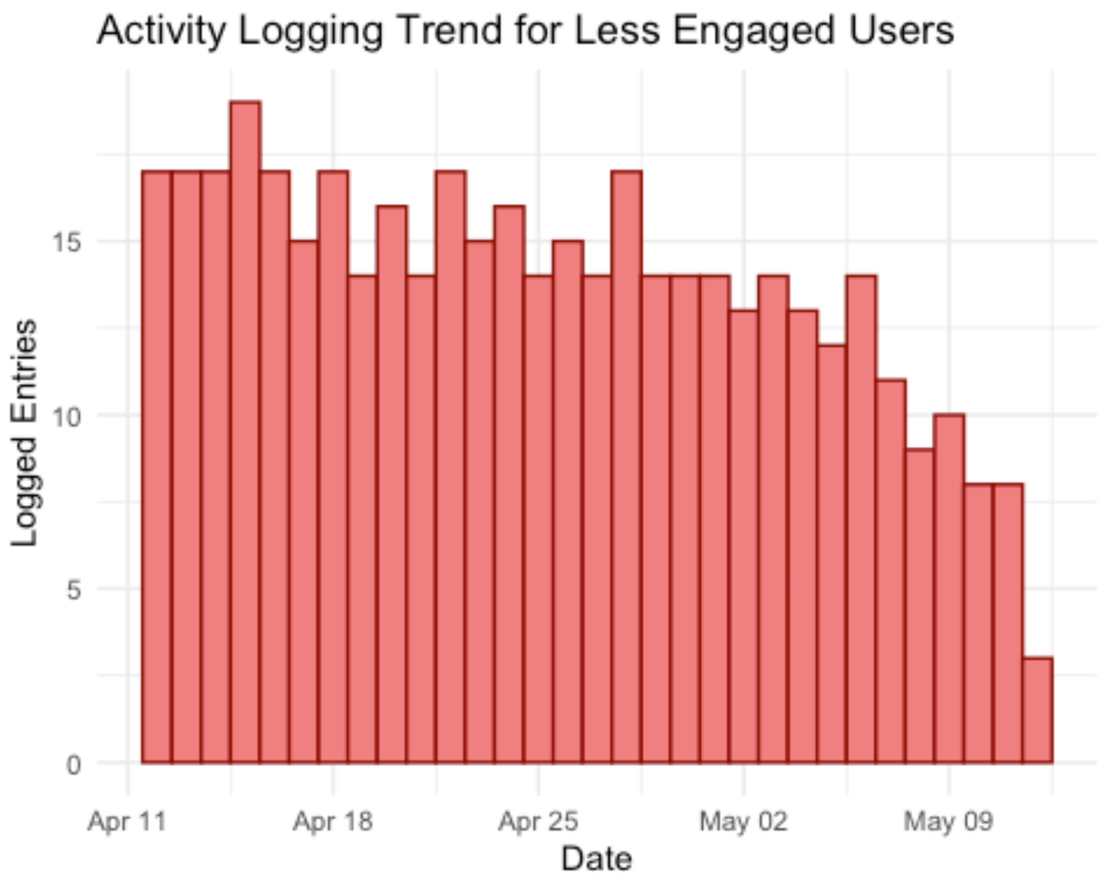
Based on the analysis of unique user activity per day, we observe a noticeable decline in the number of active users toward the end of the data collection period in early May. This trend suggests that the reduced activity is not simply due to incomplete logging, but rather a decrease in user participation.

One possible explanation could be seasonal factors—such as the beginning of summer—which may influence user engagement with their fitness trackers.

```r
# Filter users with high engagement (top 75% based on number of activity
entries)
top_active_users <- activity_clean %>%
  group_by(id) %>%
  summarise(entry_count = n()) %>%
  filter(entry_count >= quantile(entry_count, 0.75))

# Plot activity date distribution for highly engaged users
ggplot(data = activity_clean %>% filter(id %in% top_active_users$id),
       aes(x = activity_date)) +
  geom_histogram(binwidth = 1, fill = "skyblue", color = "darkblue") +
  labs(
    title = "Activity Logging Trend for Highly Engaged Users",
    x = "Date",
    y = "Logged Entries"
  ) +
  theme_minimal()
```



```
# Filter users with low engagement (under 75% based on number of activity
entries)
low_active_users <- activity_clean %>%
  group_by(id) %>%
  summarise(entry_count = n()) %>%
  filter(entry_count < quantile(entry_count, 0.75))

# Plot activity date distribution for less engaged users
ggplot(data = activity_clean %>% filter(id %in% low_active_users$id),
       aes(x = activity_date)) +
  geom_histogram(binwidth = 1, fill = "lightcoral", color = "darkred") +
  labs(
    title = "Activity Logging Trend for Less Engaged Users",
    x = "Date",
    y = "Logged Entries"
  ) +
  theme_minimal()
```



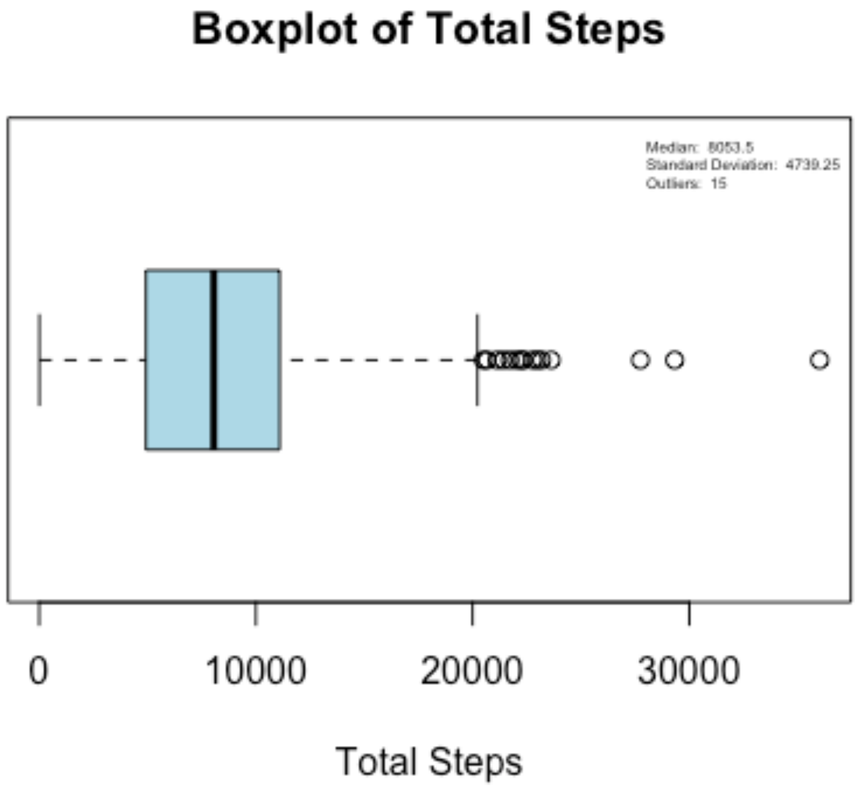
Observations:

Users with more than 75% of data consistently report activity dates, while those with less than 75% of data show a decline in reporting starting from the end of April. The decline in Activity Date seems to be primarily due to a lack of data reporting from some users during that period.

Analyze steps

```
# Create a horizontal boxplot for the 'total_steps' variable

boxplot(activity_clean$total_steps,
        main = "Boxplot of Total Steps",
        xlab = "Total Steps",
        col = "lightblue",
        border = "black",
        horizontal = TRUE)  # Making the boxplot horizontal

# Calculate the median and standard deviation of 'total_steps'
median_value <- median(activity_clean$total_steps)
std_dev <- round(sd(activity_clean$total_steps), 2)
```

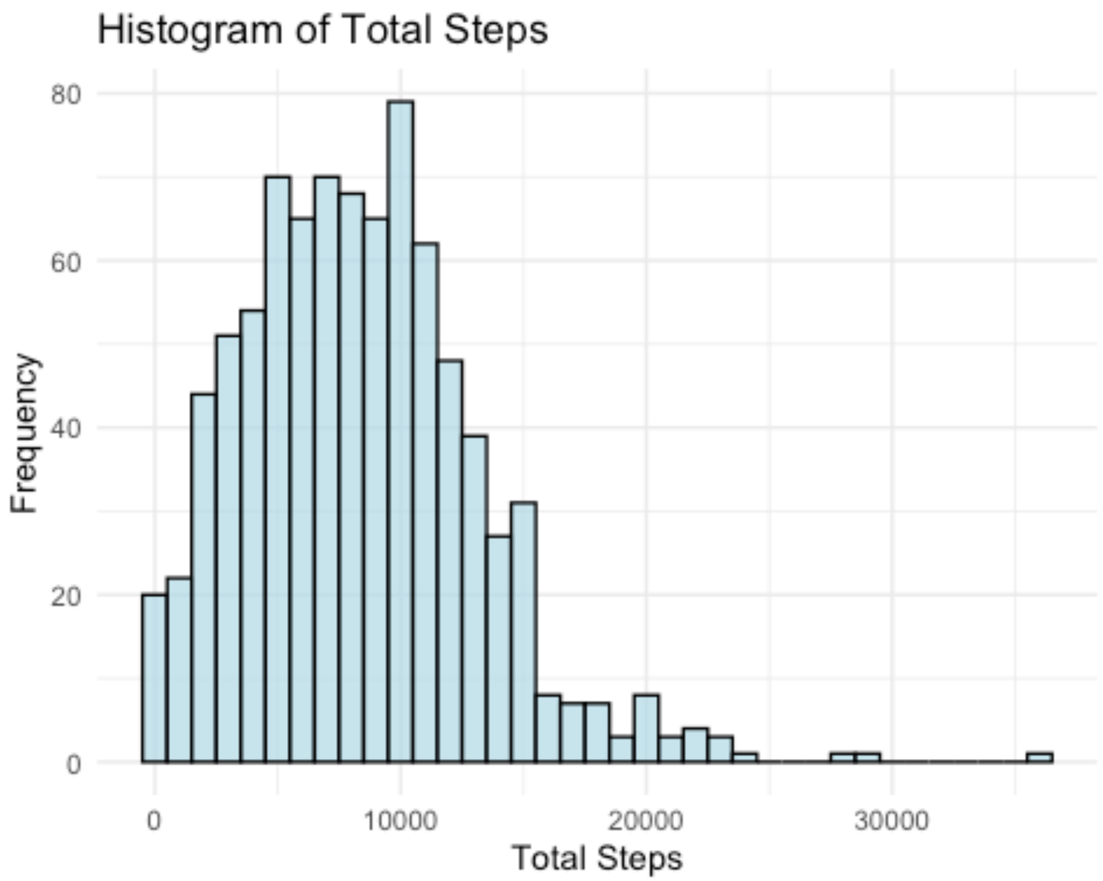
```
# Identify outliers using boxplot.stats
outliers <- boxplot.stats(activity_clean$total_steps)$out

# Count the number of outliers
num_outliers <- length(outliers)

# Construct the legend label with the calculated statistics
legend_label <- paste("Median: ", median_value,
                      "\nStandard Deviation: ", std_dev,
                      "\nOutliers: ", num_outliers)

# Add a legend to the right of the plot
legend("topright", legend = legend_label,
       pch = "", col = "black", bty = "n", cex = 0.4)
```



Observations from Boxplot of Total Step

Median The median total steps is 8,053, representing the typical daily step count. **Standard Deviation:** The standard deviation is 4,739, indicating considerable variability in users' activity levels. **Outliers:** There are 15 outliers, showing that some users take significantly more or fewer steps than the majority.

```
# Create a histogram for total_steps
ggplot(activity_clean, aes(x = total_steps)) +
  geom_histogram(binwidth = 1000, fill = "lightblue", color = "black", alpha
= 0.7) +
  labs(
    title = "Histogram of Total Steps",
    x = "Total Steps",
    y = "Frequency"
  ) +
  theme_minimal()
```



###

Observation: The distribution is right-skewed, meaning most users take fewer steps, with a few highly active users contributing to the higher end.

```
# Steps averages by IDs
steps_per_id <- activity_clean %>%
  group_by(id) %>%
  summarise(average_steps = mean(total_steps),
            median_steps =median(total_steps), n = n())

steps_per_id_sorted <- steps_per_id %>%
  arrange(desc(median_steps))
```

```
# View the sorted data
steps_per_id_sorted
```

```
## # A tibble: 33 × 4
##    id         average_steps median_steps     n
##    <chr>              <dbl>        <dbl> <int>
##  1 8877689391        16040.        15118    31
##  2 8053475328        14763.        15108    31
##  3 7007744171        12267.        13642.   24
##  4 1503960366        12521.        12438    30
##  5 3977333714        10985.        11604    30
##  6 2022484408        11371.        11548    31
##  7 6962181067         9795.        10433    31
##  8 4388161847        10814.        10243    31
##  9 7086361926         9684.        10190.   30
## 10 2347167796         9520.         9781    18
## # ℹ 23 more rows
```

```
steps_per_id
```

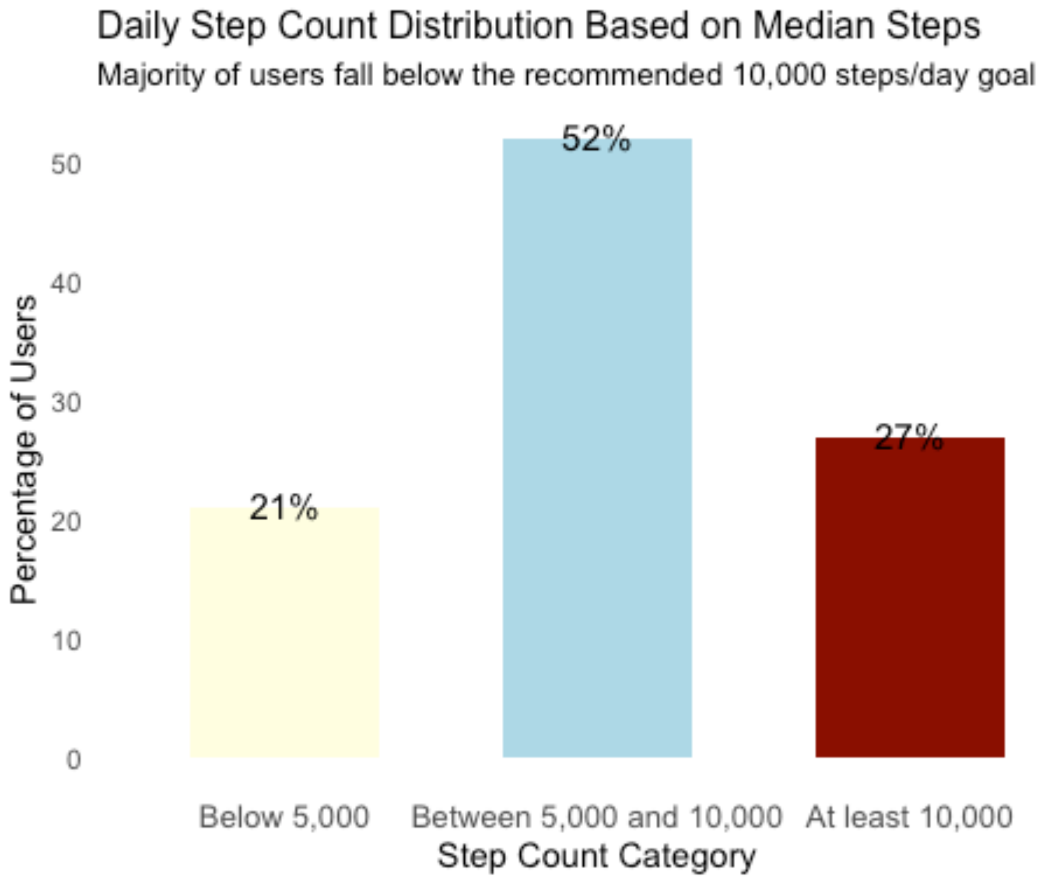
```
## # A tibble: 33 × 4
##    id         average_steps median_steps     n
##    <chr>              <dbl>        <dbl> <int>
##  1 1503960366        12521.        12438    30
##  2 1624580081         5744.         4026    31
##  3 1644430081         7283.         6684.   30
##  4 1844505072         3999.         4036.   20
##  5 1927972279         1671.         1675    17
##  6 2022484408        11371.        11548    31
##  7 2026352035         5567.         5528    31
##  8 2320127002         4717.         5057    31
##  9 2347167796         9520.         9781    18
## 10 2873212765         7556.         7762    31
## # ℹ 23 more rows
```

```
# Calculate percentages for the 'average_steps' column
below_5k_avg <- sum(steps_per_id$average_steps < 5000) / nrow(steps_per_id) *
100
between_5k_10k_avg <- sum(steps_per_id$average_steps >= 5000 &
steps_per_id$average_steps < 10000) / nrow(steps_per_id) * 100
at_least_10k_avg <- sum(steps_per_id$average_steps >= 10000) /
nrow(steps_per_id) * 100

# Calculate percentages for the 'median_steps' column
below_5k_med <- sum(steps_per_id$median_steps < 5000) / nrow(steps_per_id) *
100
between_5k_10k_med <- sum(steps_per_id$median_steps >= 5000 &
steps_per_id$median_steps < 10000) / nrow(steps_per_id) * 100
at_least_10k_med <- sum(steps_per_id$median_steps >= 10000) /
nrow(steps_per_id) * 100
```

```r
# Create a data frame for the steps categories and their percentages
steps_percentage_df <- data.frame(
  Category = c("Below 5,000", "Between 5,000 and 10,000", "At least 10,000"),
  Percentage_Average = round(c(below_5k_avg, between_5k_10k_avg,
at_least_10k_avg)),
  Percentage_Median = round(c(below_5k_med, between_5k_10k_med,
at_least_10k_med))
)

steps_percentage_df
```

```
##                   Category Percentage_Average Percentage_Median
## 1              Below 5,000                 21                21
## 2 Between 5,000 and 10,000                 58                52
## 3          At least 10,000                 21                27
```

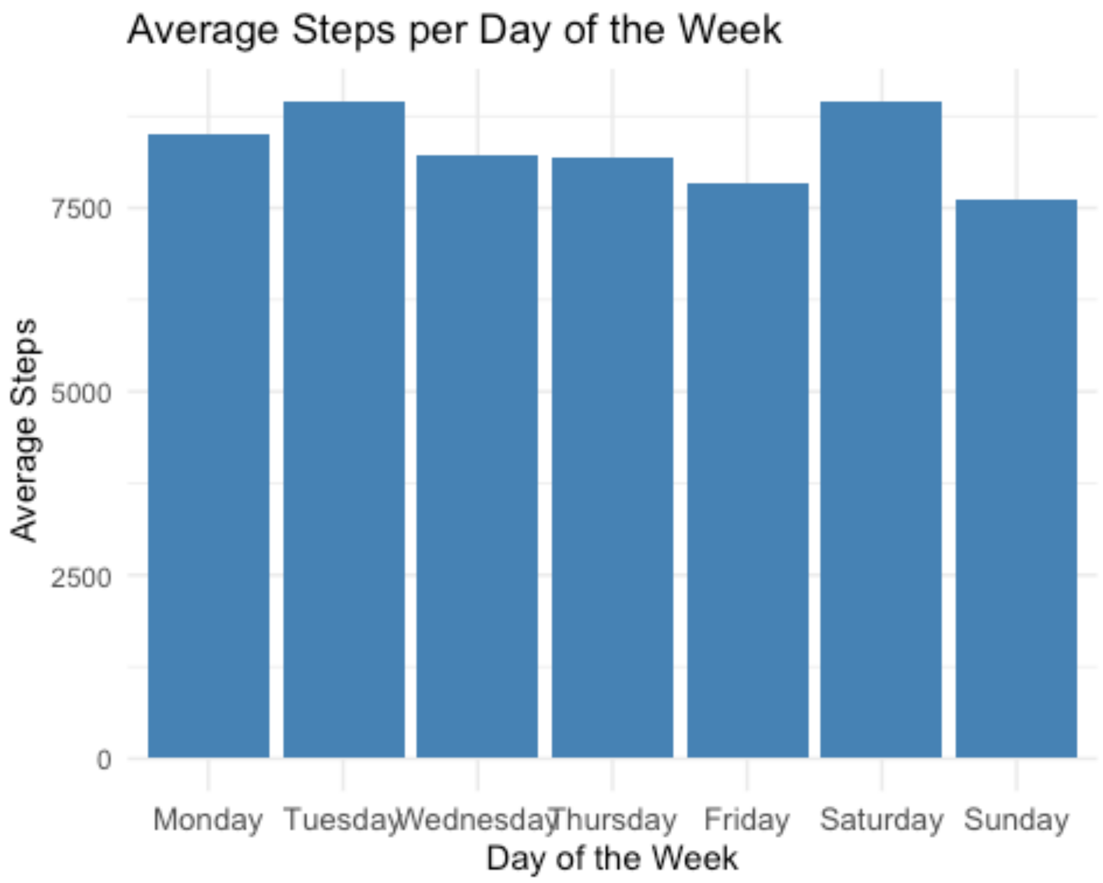
```r
# Ensure the order of categories is preserved
steps_percentage_df$Category <- factor(steps_percentage_df$Category,
                                       levels = c("Below 5,000", "Between
5,000 and 10,000", "At least 10,000"))

# Plot only the Median Steps
ggplot(steps_percentage_df, aes(x = Category, y = Percentage_Median, fill =
Category)) +
  geom_bar(stat = "identity", width = 0.6, show.legend = FALSE) +
  geom_text(aes(label = paste0(Percentage_Median, "%")),
            vjust = 0.4, color = "black", size = 4) +
  labs(title = "Daily Step Count Distribution Based on Median Steps",
       subtitle = "Majority of users fall below the recommended 10,000
steps/day goal",
       x = "Step Count Category",
       y = "Percentage of Users") +
  scale_fill_manual(values = c("Below 5,000" = "lightyellow",
                               "Between 5,000 and 10,000" = "lightblue",
                               "At least 10,000" = "darkred")) +
  theme_minimal() +
  theme(panel.grid = element_blank(),
        plot.title = element_text(size = 12),
        plot.subtitle = element_text(size = 10),
        axis.text.x = element_text(size = 10))
```



Observations:

Over half of users maintain a healthy daily step count range of 5,000 to 10,000 steps, but only one-fifth achieve the 10,000-step milestone.

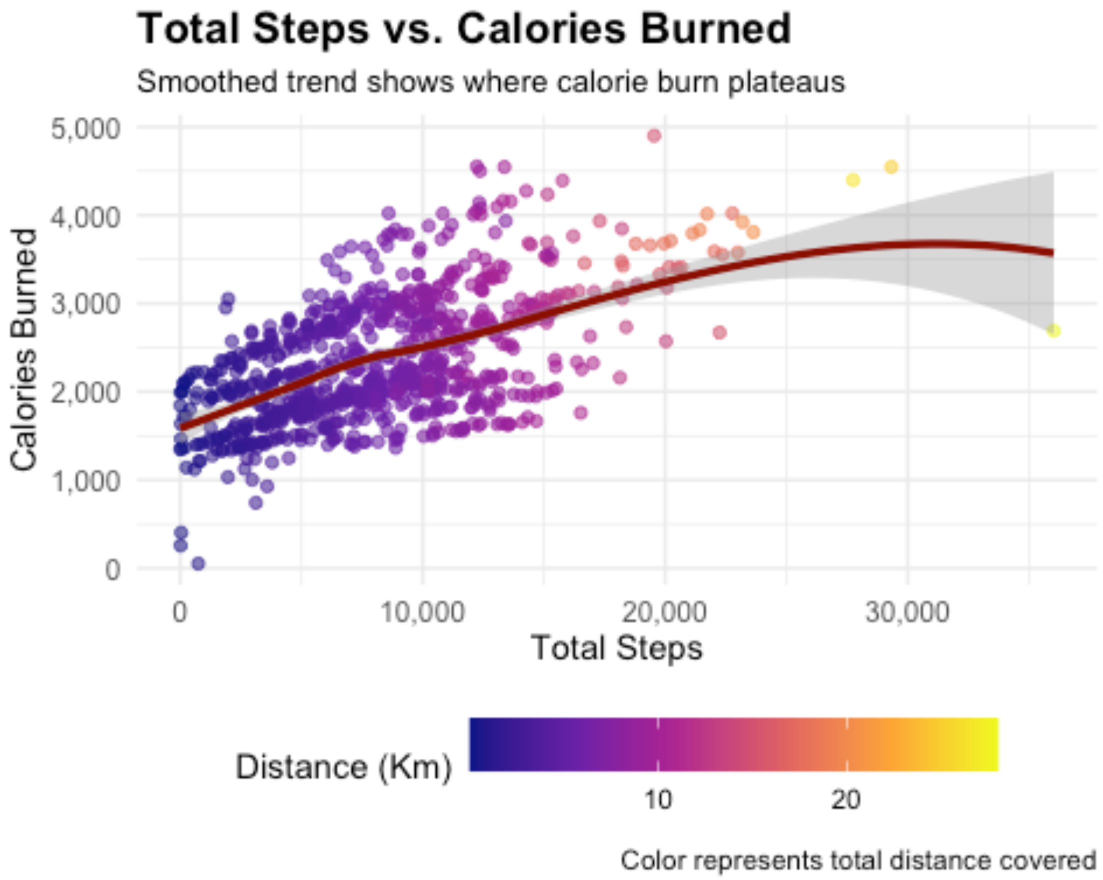
```
# Create a weekday column
activity_clean$weekday <- weekdays(activity_clean$activity_date)

# Order the weekdays to start from Monday
activity_clean$weekday <- factor(activity_clean$weekday,
                                levels = c("Monday", "Tuesday", "Wednesday",
                                           "Thursday", "Friday", "Saturday",
"Sunday"))

# Calculate average steps per weekday
avg_steps_weekday <- activity_clean %>%
  group_by(weekday) %>%
  summarise(average_steps = round(mean(total_steps), 0))

# Plot: Vertical bar chart
ggplot(avg_steps_weekday, aes(x = weekday, y = average_steps)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(title = "Average Steps per Day of the Week",
       x = "Day of the Week",
       y = "Average Steps") +
  theme_minimal() +
  theme(axis.text.x = element_text(size = 10))  # Adjust size as needed
```



Observations:

Users took the most steps on Saturday and the least number of steps on Sunday

```
# Steps vs Calories Burned

ggplot(activity_clean, aes(x = total_steps, y = calories)) +
  geom_point(aes(color = total_distance), alpha = 0.6) +
  geom_smooth(se = TRUE, color = "darkred", size = 1.2) +
  labs(
    title = "Total Steps vs. Calories Burned",
    subtitle = "Smoothed trend shows where calorie burn plateaus",
    x = "Total Steps",
    y = "Calories Burned",
    caption = "Color represents total distance covered"
  ) +
  scale_color_viridis_c(name = "Distance (Km)", option = "plasma") +
  scale_x_continuous(labels = scales::comma) +
  scale_y_continuous(labels = scales::comma) +
  theme_minimal() +
  theme(
    legend.position = "bottom",
    legend.key.width = unit(1.2, "cm"),
    plot.title = element_text(face = "bold", size = 14),
    plot.subtitle = element_text(size = 10),
    axis.text = element_text(size = 9)
  )

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

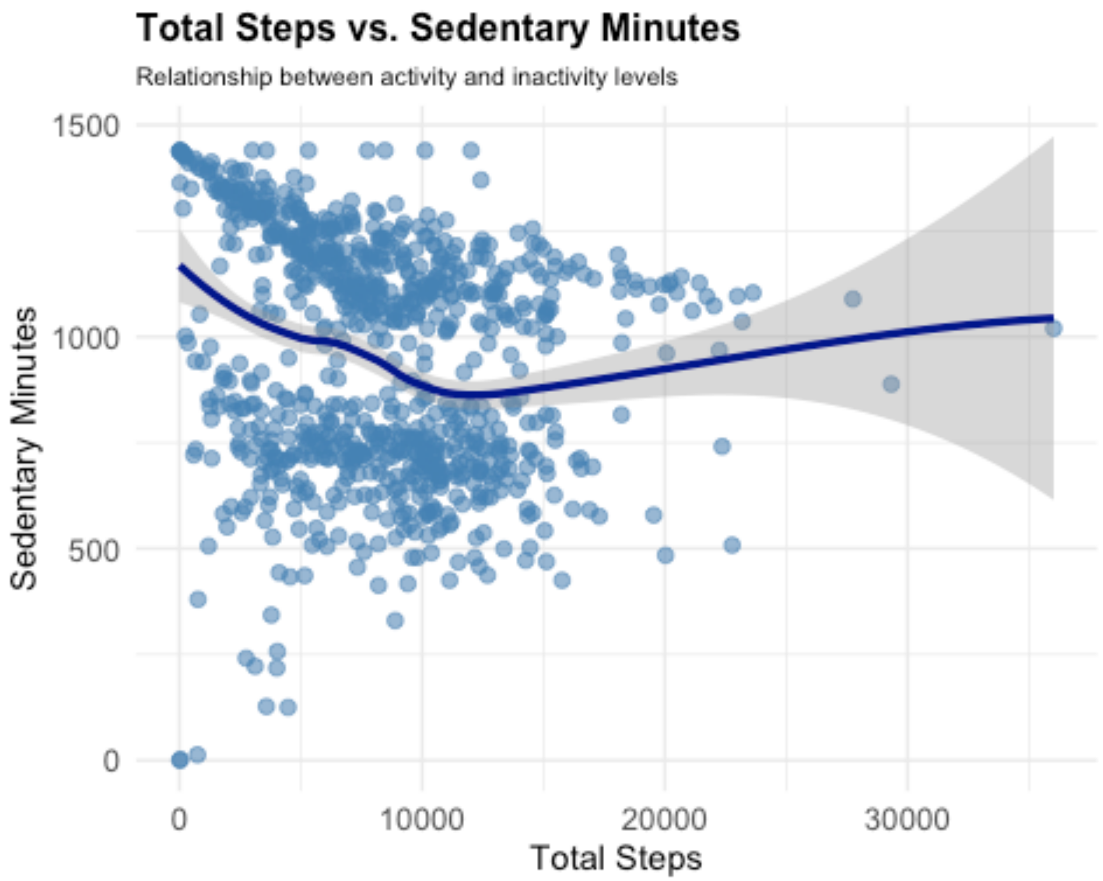


Observations:

The plot **“Relationship Between Steps and Calories Burned”** shows a clear positive trend—more steps typically lead to more calories burned. However, the increase appears to slow down and plateau after approximately 30,000 steps, likely due to fewer data points in that range. The color gradient represents total distance walked, with lighter shades indicating longer distances.

```
ggplot(data = activity_clean, aes(x = total_steps, y = sedentary_minutes)) +
  geom_point(alpha = 0.6, color = "steelblue", size = 2) +
  geom_smooth(method = "loess", se = TRUE, color = "darkblue", size = 1.2) +
# Smoother curve
  labs(
    title = "Total Steps vs. Sedentary Minutes",
    subtitle = "Relationship between activity and inactivity levels",
    x = "Total Steps",
    y = "Sedentary Minutes",
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(size = 12, face = "bold"),
    plot.subtitle = element_text(size = 8),
    axis.text = element_text(size = 10),
    legend.position = "none"
  )
```

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



Observations

The analysis indicates a clear negative correlation between total steps and sedentary minutes. In other words, as sedentary time increases, users tend to take fewer steps throughout the day. This finding could help guide marketing strategies by identifying less active users. Encouraging these individuals to increase their daily step count—possibly through personalized goals or activity challenges—could lead to higher engagement with fitness tracking tools.

Analyze Sedentary Minutes

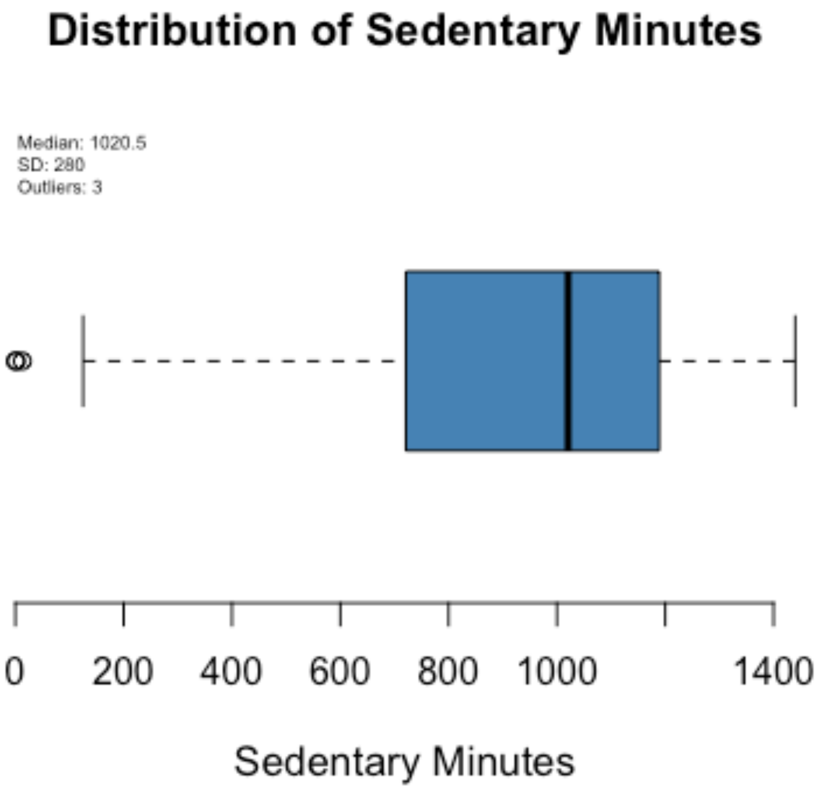
```
# Check sedentary_minutes stats
activity_clean$sedentary_minutes %>% summary()
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##     0.0   721.2  1020.5   955.2  1189.0  1440.0
```

```
outliers
```

```
##  [1] 36019 22244 22770 20669 22359 22988 20500 22026 23186 21129 29326
23629
## [13] 27745 21727 21420
```

```
# Create the horizontal boxplot
boxplot(activity_clean$sedentary_minutes,
        main = "Distribution of Sedentary Minutes",
        xlab = "Sedentary Minutes",
        col = "steelblue",          # Light salmon fill
        border = "black",       # Dark red border
                     # Notch for median
        horizontal = TRUE,
        frame.plot = FALSE,
        cex.main = 1.2,
        cex.lab = 1.1)

# Stats for legend
median_value <- median(activity_clean$sedentary_minutes)
std_dev <- sd(activity_clean$sedentary_minutes)
outliers <- boxplot.stats(activity_clean$sedentary_minutes)$out
num_outliers <- length(outliers)

# Add legend at top-right (manually adjust coordinates if needed)
legend("topleft",
       legend = c(paste("Median:", round(median_value, 1)),
                  paste("SD:", round(std_dev, 1)),
                  paste("Outliers:", num_outliers)),
       bty = "n",
       text.col = "black",
       cex = 0.5
       )
```



Observations on Sedentary Minutes

The data shows notably high sedentary time values — for example, 1,020 minutes equals 17 hours, and 1,400 minutes spans a full 24 hours. A quick investigation suggests that Fitbit may default to a value around 1,400 when the device isn't being worn. This implies that some of the recorded sedentary minutes could actually represent periods when the device was off the wrist, including during sleep.

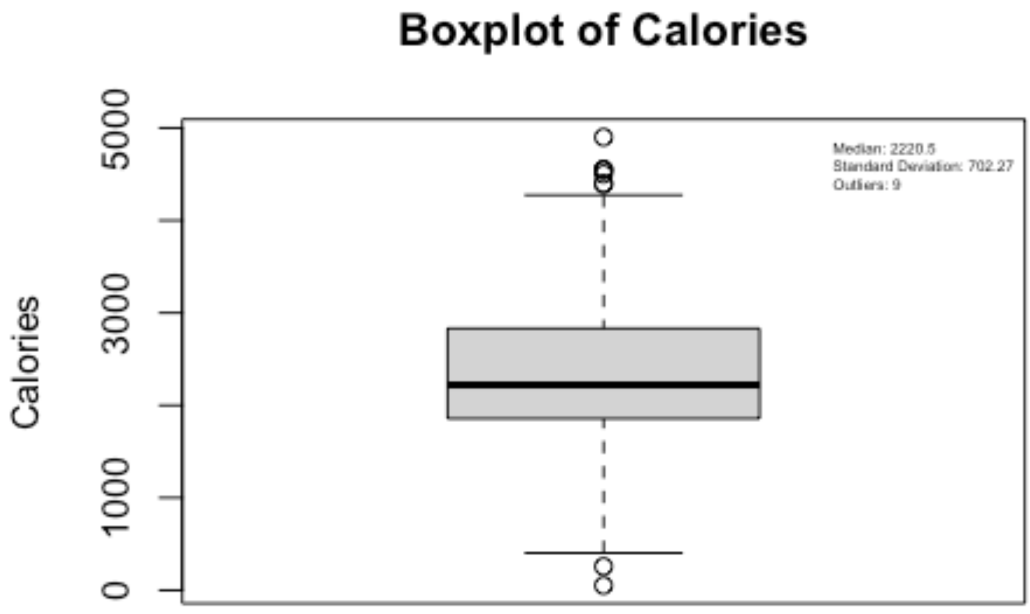
According to the data documentation, `sedentary_minutes` reflects the total time a user was inactive. However, since sleep time is not categorized as sedentary, subtracting sleep duration can provide a more accurate picture of waking inactivity.

> "Sleep time is not considered sedentary time, so it was removed to determine the waking day and to allow the proportion of the day spent sedentary to be calculated."

Analyze calories

```
# Create a boxplot for calories
boxplot(activity_clean$calories,
        main = "Boxplot of Calories",
        ylab = "Calories")
```

```r
# Calculate the median and standard deviation
median_value <- median(activity_clean$calories)
std_dev <- round(sd(activity_clean$calories),2)

# Identify outliers
outliers <- boxplot.stats(activity_clean$calories)$out

# Count the number of outliers
num_outliers <- length(outliers)

# Create the legend label with median, standard deviation, and outlier count
legend_label <- paste("Median:", median_value,
                      "\nStandard Deviation:", std_dev,
                      "\nOutliers:", num_outliers)

# Add the legend with median, standard deviation, and outlier count
legend("topright", legend = legend_label, pch = "", col = "black", bty = "n",
cex = 0.4)
```



```r
# 📊 Average and Median Calories Burned Per User

calories_id_unique <- activity_clean %>%
  group_by(id) %>%
  summarise(
    average_calories = round(mean(calories), 1),
    median_calories = round(median(calories), 1),
    .groups = "drop"
  )

calories_id_unique
```

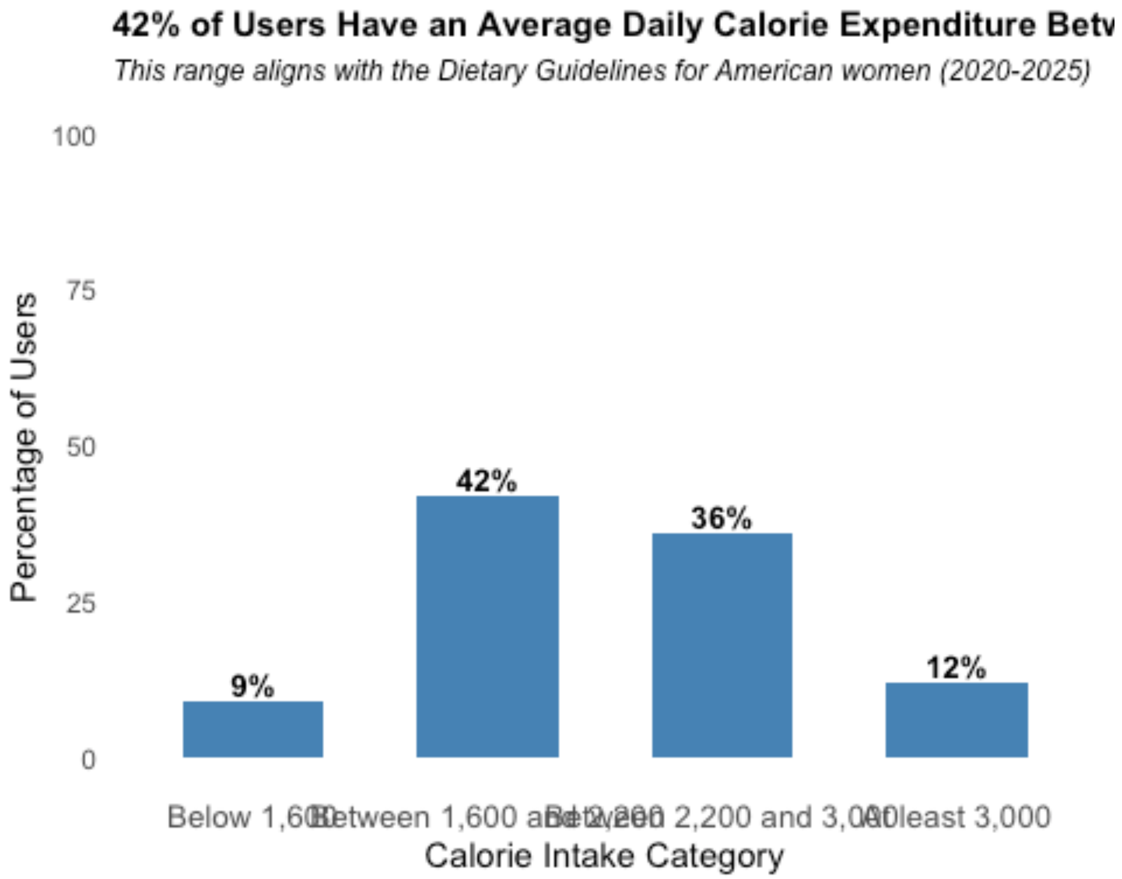
```
## # A tibble: 33 × 3
##    id         average_calories median_calories
##    <chr>                 <dbl>           <dbl>
##  1 1503960366            1877            1848
##  2 1624580081            1483.           1435
##  3 1644430081            2811.           2802.
##  4 1844505072            1732            1752.
##  5 1927972279            2303.           2324
##  6 2022484408            2510            2529
##  7 2026352035            1541.           1521
##  8 2320127002            1724.           1779
##  9 2347167796            2043.           2072.
## 10 2873212765            1917            1907
## # ℹ 23 more rows
```

```
# Calculate percentages for the average column
below_1600_avg <- sum(calories_id_unique$average_calories < 1600) /
nrow(calories_id_unique) * 100
between_1600_2200_avg <- sum(calories_id_unique$average_calories >= 1600 &
calories_id_unique$average_calories < 2200) / nrow(calories_id_unique) * 100
between_2200_3000_avg <- sum(calories_id_unique$average_calories >= 2200 &
calories_id_unique$average_calories < 3000) / nrow(calories_id_unique) * 100
at_least_3000_avg <- sum(calories_id_unique$average_calories >= 3000) /
nrow(calories_id_unique) * 100

# Calculate percentages for the median column
below_1600_med <- sum(calories_id_unique$median_calories < 1600) /
nrow(calories_id_unique) * 100
between_1600_2200_med <- sum(calories_id_unique$median_calories >= 1600 &
calories_id_unique$median_calories < 2200) / nrow(calories_id_unique) * 100
between_2200_3000_med <- sum(calories_id_unique$median_calories >= 2200 &
calories_id_unique$median_calories < 3000) / nrow(calories_id_unique) * 100
at_least_3000_med <- sum(calories_id_unique$median_calories >= 3000) /
nrow(calories_id_unique) * 100

# Create a data frame for the calories categories
percentage_calories_df <- data.frame(
  Category = c("Below 1,600", "Between 1,600 and 2,200", "Between 2,200 and
3,000", "At least 3,000"),
  Percentage_Average = round(c(below_1600_avg, between_1600_2200_avg,
between_2200_3000_avg, at_least_3000_avg)),
  Percentage_Median = round(c(below_1600_med, between_1600_2200_med,
```

```r
between_2200_3000_med, at_least_3000_med))
)

# Convert "Category" to a factor with custom factor levels
percentage_calories_df$Category <- factor(percentage_calories_df$Category,
levels = c("Below 1,600", "Between 1,600 and 2,200", "Between 2,200 and
3,000", "At least 3,000"))

percentage_calories_df
```

```
##                  Category Percentage_Average Percentage_Median
## 1             Below 1,600                  9                 9
## 2 Between 1,600 and 2,200                 42                39
## 3 Between 2,200 and 3,000                 36                33
## 4          At least 3,000                 12                18
```

```r
# 📊 Daily Calorie Expenditure Distribution (Average)

ggplot(percentage_calories_df, aes(x = Category, y = Percentage_Average)) +
  geom_bar(stat = "identity", fill = "steelblue", width = 0.6) +
  geom_text(aes(label = paste0(Percentage_Average, "%")),
            vjust = -0.3, size = 3.5, fontface = "bold") +
  labs(
    title = "42% of Users Have an Average Daily Calorie Expenditure Between
1,600 and 2,200",
    subtitle = "This range aligns with the Dietary Guidelines for American
women (2020-2025)",
    x = "Calorie Intake Category",
    y = "Percentage of Users"
  ) +
  ylim(0, 100) +
  theme_minimal() +
  theme(
    panel.grid = element_blank(),
    axis.text.x = element_text(size = 10),
    plot.title = element_text(size = 11, face = "bold"),
    plot.subtitle = element_text(size = 9, face = "italic")
  )
```



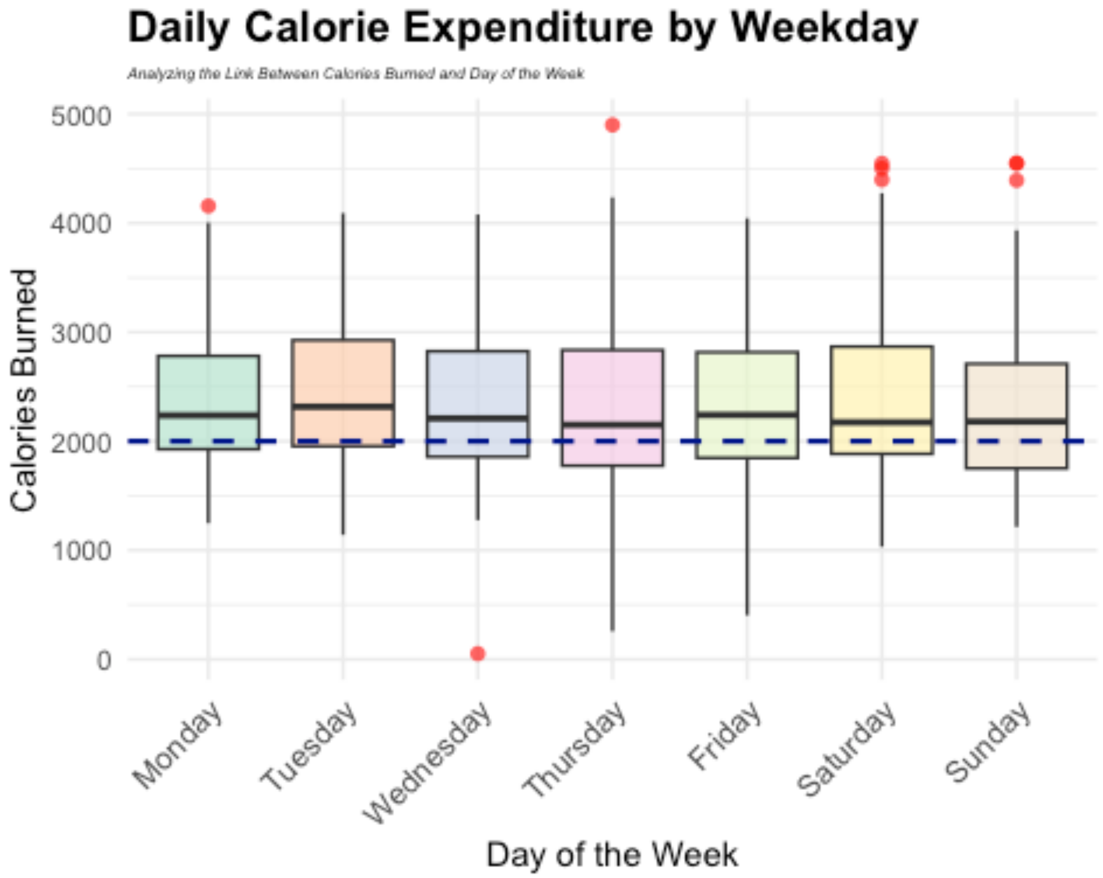
42% of Users Have an Average Daily Calorie Expenditure Betv

This range aligns with the Dietary Guidelines for American women (2020-2025)

Observations:

Approximately 42% of users maintain an average daily calorie expenditure ranging from 1,600 to 2,200

```
# Calories Burned by Weekday
ggplot(activity_clean, aes(x = weekday, y = calories, fill = weekday)) +
  geom_boxplot(outlier.color = "red", outlier.shape = 16, outlier.size = 2,
alpha = 0.7) +
  geom_hline(yintercept = 2000, linetype = "dashed", color = "darkblue", size
= 0.8) +
  labs(
    title = "Daily Calorie Expenditure by Weekday",
    subtitle = "Analyzing the Link Between Calories Burned and Day of the
Week",
    x = "Day of the Week",
    y = "Calories Burned",
  ) +
  scale_fill_brewer(palette = "Pastel2") +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1, size = 10),
    plot.title = element_text(face = "bold", size = 14),
    plot.subtitle = element_text(size = 5, face = "italic"),
    legend.position = "none"
  )
```



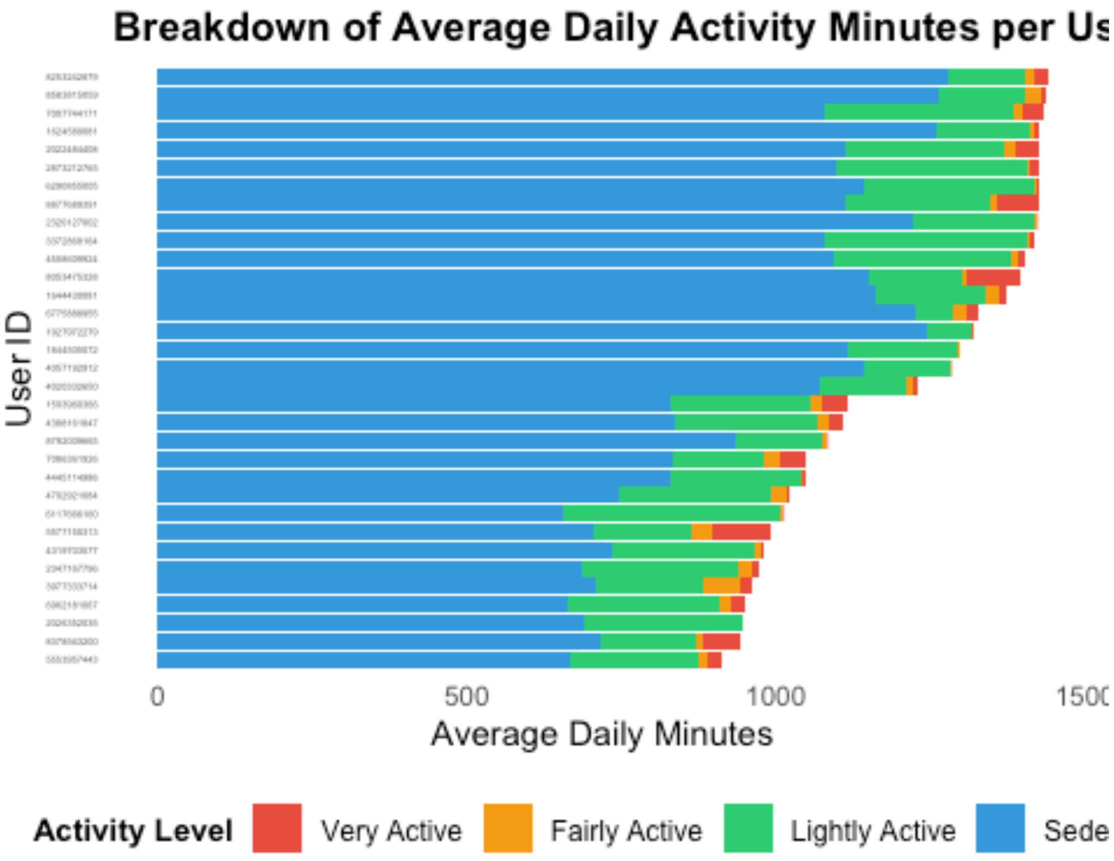
###Observation: The boxplot illustrates that the majority of participants do not exceed the recommended daily calorie burn of 2,000.

Analyze Intensity Minutes: Time spent in one of four intensity categories.

```
activity_minutes_unique <-
  activity_clean %>%
  group_by(id) %>%
  summarise(
    average_very_active_minutes = mean(very_active_minutes),
    average_fairly_active_minutes = mean(fairly_active_minutes),
    average_lightly_active_minutes = mean(lightly_active_minutes),
    average_sedentary_minutes = mean(sedentary_minutes)
  )

activity_minutes_unique

## # A tibble: 33 × 5
##     id        average_very_active_…¹ average_fairly_activ…²
average_lightly_acti…³
```

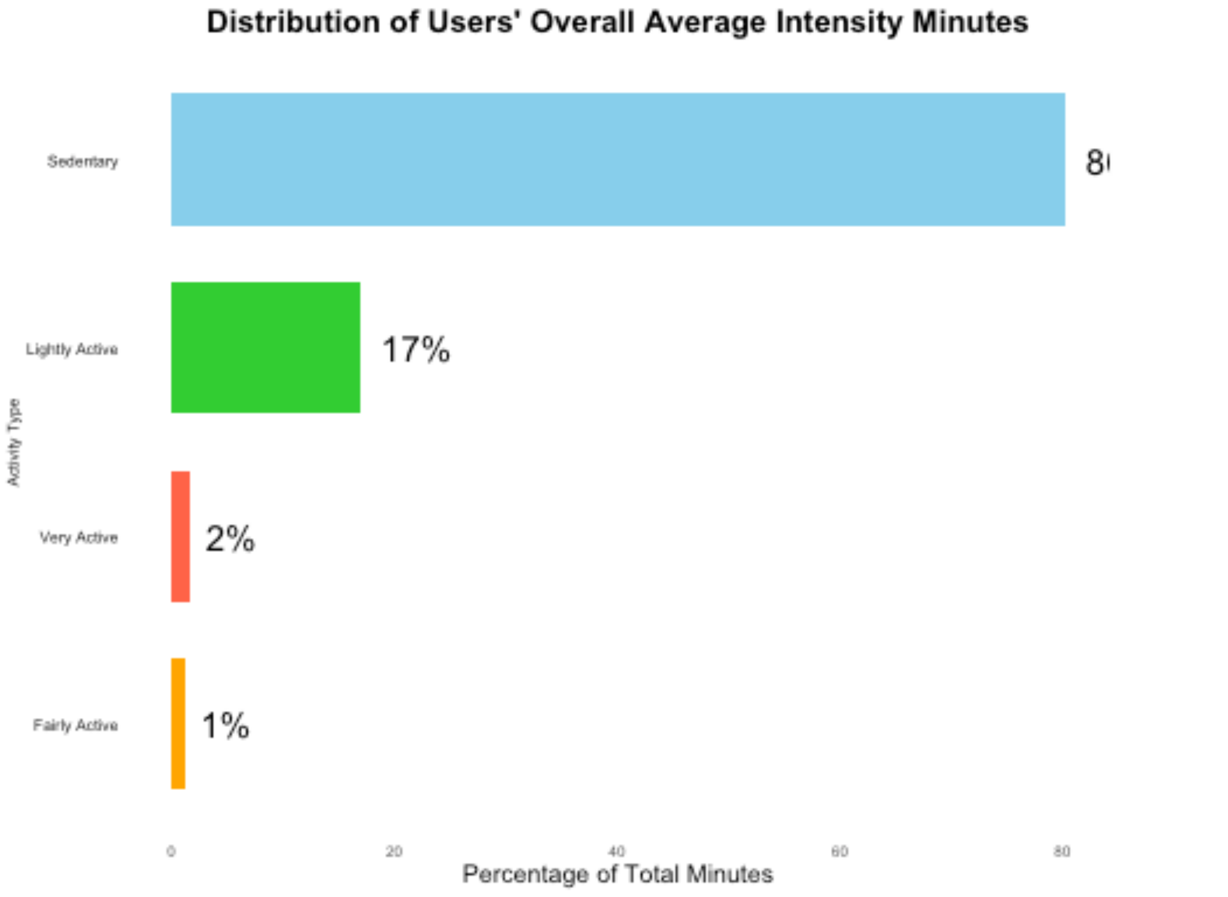
```
##    <chr>                      <dbl>                    <dbl>
<dbl>
##  1 1503960…                 40                       19.8
227.
##  2 1624580…                  8.68                     5.81
153.
##  3 1644430…                  9.57                    21.4
178.
##  4 1844505…                  0.2                      2
179.
##  5 1927972…                  2.41                     1.41
70.4
##  6 2022484…                 36.3                     19.4
257.
##  7 2026352…                  0.0968                   0.258
257.
##  8 2320127…                  1.35                     2.58
198.
##  9 2347167…                 13.5                     20.6
252.
## 10 2873212…                 14.1                      6.13
308
## # ℹ 23 more rows
## # ℹ abbreviated names: ¹average_very_active_minutes,
## #   ²average_fairly_active_minutes, ³average_lightly_active_minutes
## # ℹ 1 more variable: average_sedentary_minutes <dbl>

# Reshape your data to long format for easier plotting
activity_minutes_long <- activity_minutes_unique %>%
  select(id, average_very_active_minutes, average_fairly_active_minutes,
average_lightly_active_minutes, average_sedentary_minutes) %>%
  rename(
    `Very Active` = average_very_active_minutes,
    `Fairly Active` = average_fairly_active_minutes,
    `Lightly Active` = average_lightly_active_minutes,
    Sedentary = average_sedentary_minutes
  ) %>%
  pivot_longer(cols = -id, names_to = "Activity_Level", values_to =
"Minutes")

# Define custom order of activity levels
activity_minutes_long$Activity_Level <-
factor(activity_minutes_long$Activity_Level,
                                               levels = c("Very Active",
"Fairly Active", "Lightly Active", "Sedentary"))

# Create a horizontal stacked bar plot
ggplot(activity_minutes_long, aes(x = Minutes, y = reorder(id, Minutes), fill
= Activity_Level)) +
```

```r
  geom_bar(stat = "identity") +
  scale_fill_manual(
    values = c(
      "Very Active" = "#E74C3C",
      "Fairly Active" = "#F39C12",
      "Lightly Active" = "#2ECC71",
      "Sedentary" = "#3498DB"
    )
  ) +
  labs(
    title = "Breakdown of Average Daily Activity Minutes per User",
    x = "Average Daily Minutes",
    y = "User ID",
    fill = "Activity Level"
  ) +
  theme_minimal() +
  theme(
    legend.position = "bottom",
    legend.title = element_text(size = 10, face = "bold"),
    legend.text = element_text(size = 9),
    plot.title = element_text(face = "bold", size = 13),
    axis.text.y = element_text(size = 3),
    panel.grid = element_blank()
  )
```



```
activity_cols <- c("average_very_active_minutes",
"average_fairly_active_minutes",
                   "average_lightly_active_minutes",
"average_sedentary_minutes")

# Calculate averages and proportions
averages <- colMeans(activity_minutes_unique[activity_cols])
proportions <- prop.table(averages) * 100

# Create a tidy dataframe
overall_average_df <- data.frame(
  Activity = c("Very Active", "Fairly Active", "Lightly Active",
"Sedentary"),
  Average_Minutes = round(averages, 1),
  Percentage = round(proportions, 1)
)

# View result
overall_average_df

##                                     Activity Average_Minutes Percentage
## average_very_active_minutes      Very Active            21.1        1.7
## average_fairly_active_minutes  Fairly Active            14.0        1.2
## average_lightly_active_minutes Lightly Active          204.8       16.9
## average_sedentary_minutes            Sedentary          971.9       80.2
```

```
ggplot(overall_average_df, aes(x = Percentage, y = reorder(Activity,
Percentage), fill = Activity)) +
  geom_bar(stat = "identity", width = 0.7, show.legend = FALSE) +
  geom_text(aes(label = paste0(round(Percentage), "%")), hjust = -0.3, color
= "black", size = 4) +
  ylab("Activity Type") +
  xlab("Percentage of Total Minutes") +
  ggtitle("Distribution of Users' Overall Average Intensity Minutes") +
  scale_fill_manual(values = c("Very Active" = "#FF6347", "Fairly Active" =
"#FFA500",
                               "Lightly Active" = "#32CD32", "Sedentary" =
"#87CEEB")) +
  scale_x_continuous(labels = scales::comma_format()) +
  theme_minimal(base_size = 6) +
  theme(legend.position = "none",
        panel.grid = element_blank(),
        axis.text.y = element_text(size = 5, color = "black"),
        plot.title = element_text(hjust = 0.5, size = 10, face = "bold"),
        axis.title.x = element_text(size = 8),
        axis.title.y = element_text(size = 5))
```

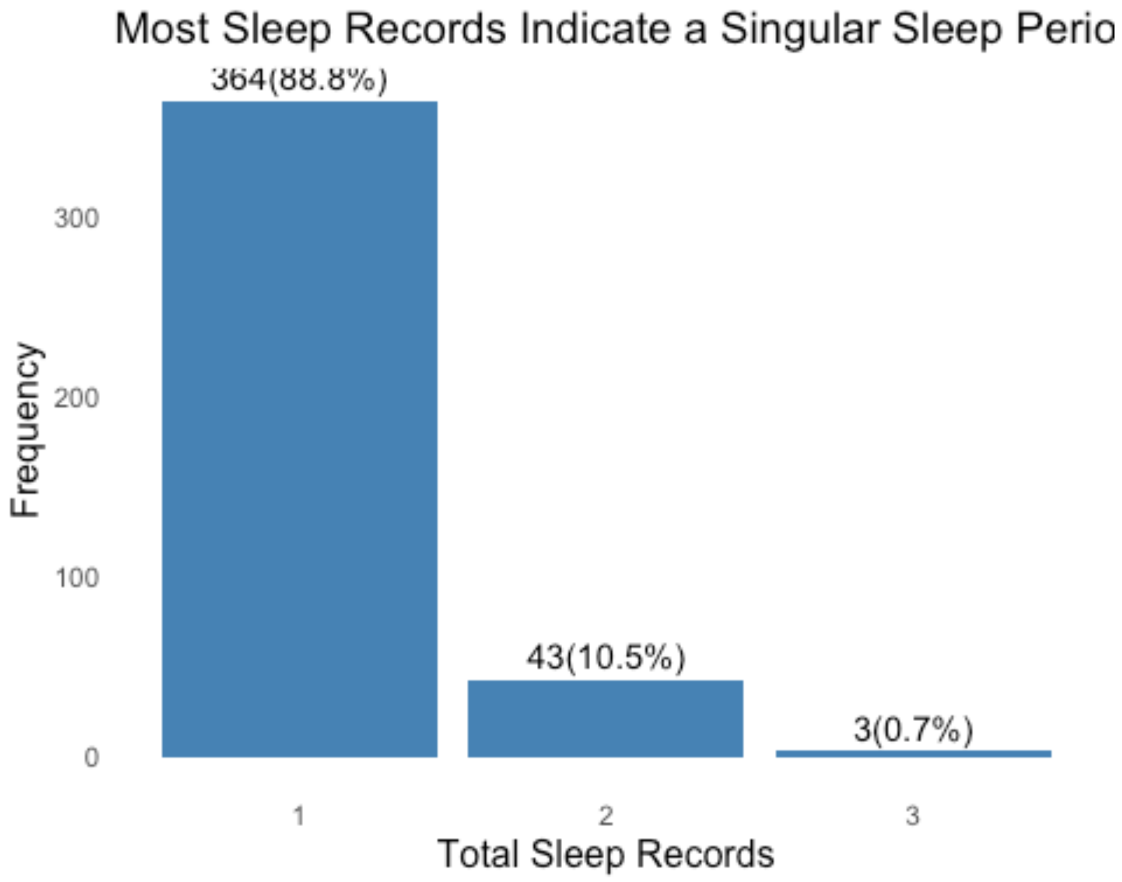


Observations:

Sedentary activities make up the majority of users' average intensity minutes, at around 80%. Lightly active time accounts for 17%, while very active and fairly active minutes are limited to 2% and 1%, respectively.

Analyze Sleep data set

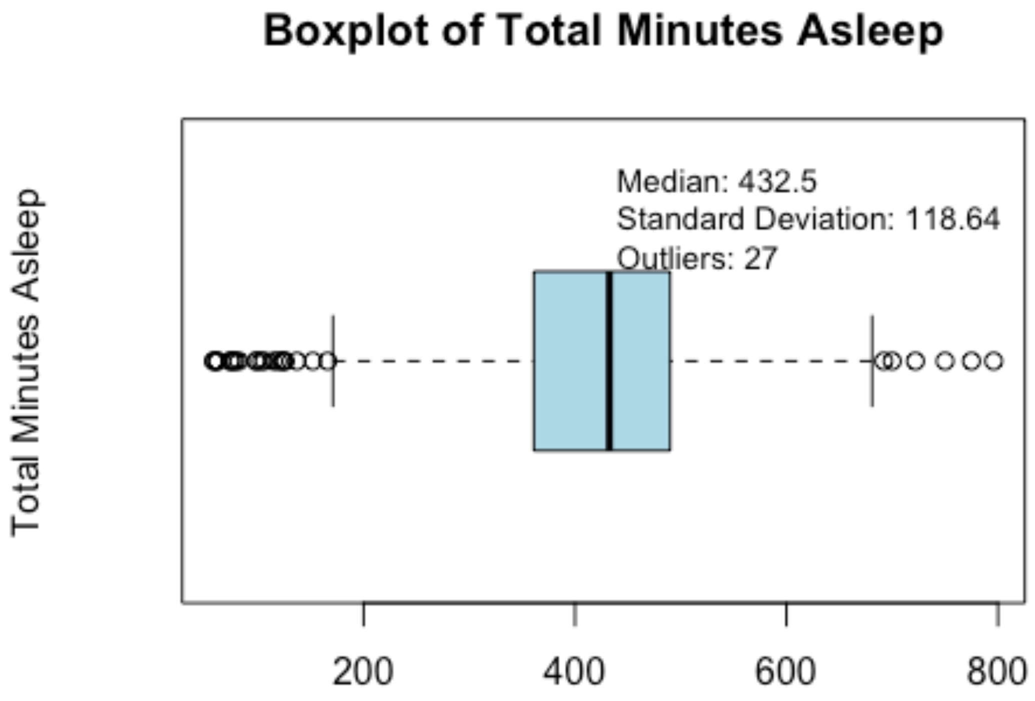
```
# Frequency table for total sleep records
frequency_table <- as.data.frame(table(sleep_clean$total_sleep_records))
frequency_table$Percentage <- frequency_table$Freq /
sum(frequency_table$Freq) * 100

ggplot(data = frequency_table, aes(x = Var1, y = Freq)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  geom_text(aes(label = paste(Freq, "(", round(Percentage, 1), "%)", sep =
"")),
            vjust = -0.5, color = "black") +
  labs(x = "Total Sleep Records", y = "Frequency",
       title = "Most Sleep Records Indicate a Singular Sleep Period") +
  theme_minimal() +
  theme(panel.grid = element_blank(),
        plot.title = element_text(size = 14),
        axis.title = element_text(size = 12))
```



Observations:

In 89% of the sleep records, users typically have a single continuous sleep period, including naps that last over 60 minutes.

```r
# Create a boxplot for total_minutes_asleep
boxplot(sleep_clean$total_minutes_asleep,
        main = "Boxplot of Total Minutes Asleep",
        ylab = "Total Minutes Asleep",
        col = "lightblue",
        border = "black",
        horizontal = TRUE,
        las = 1)

# Calculate the median and standard deviation
median_value <- median(sleep_clean$total_minutes_asleep)
std_dev <- round(sd(sleep_clean$total_minutes_asleep), 2)

# Identify outliers
outliers <- boxplot.stats(sleep_clean$total_minutes_asleep)$out

# Count the number of outliers
num_outliers <- length(outliers)

# Create the legend label with median, standard deviation, and outlier count
legend_label <- paste("Median:", median_value,
                      "\nStandard Deviation:", std_dev,
                      "\nOutliers:", num_outliers)

# Add the legend with median, standard deviation, and outlier count
legend("topright", legend = legend_label, pch = "", col = "black", bty = "n",
cex = 0.85)
```



Observation:

The median of user sleep is 432.5 minutes, indicating that half of the users sleep more or less than this amount. Additionally, there are 27 outliers in the data, which represent extreme values outside the typical sleep durations. These outliers might suggest irregular sleeping patterns or possible data entry issues that warrant further investigation.

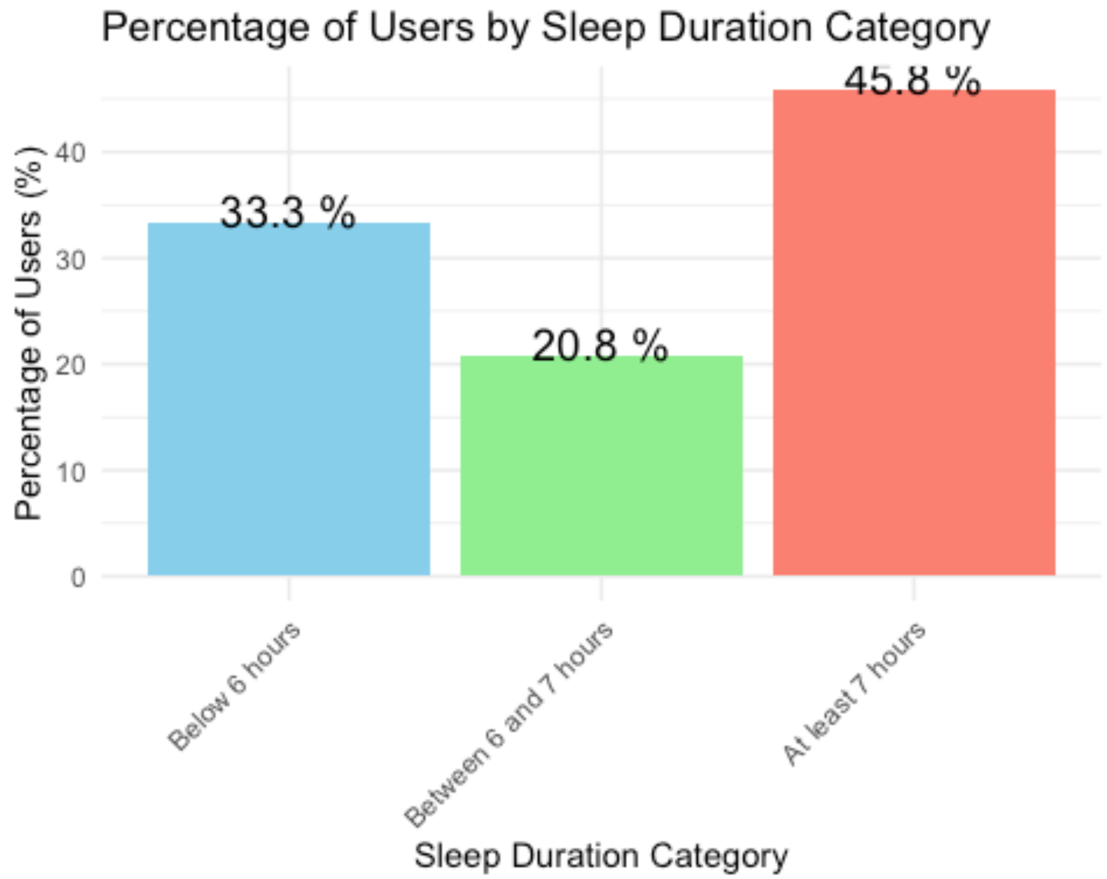
```
 # Sleep duration averages by IDs with standard deviation and count (n)
sleep_df <- sleep_clean %>%
  group_by(id) %>%
  summarise(
    `Avg_Sleep (mins)` = round(mean(total_minutes_asleep), 2),
    `Std_Dev_Sleep (mins)` = round(sd(total_minutes_asleep), 2),
    `N` = n(),
    `Min_leep (mins)` = min(total_minutes_asleep),
    `Max_Sleep (mins)` = max(total_minutes_asleep)
  ) %>%
  arrange(desc(`Avg_Sleep (mins)`))  # Optional: Sort by average sleep time

# Preview the updated dataframe
print(sleep_df)
```

```
## # A tibble: 24 × 6
##    id         `Avg_Sleep (mins)` `Std_Dev_Sleep (mins)`     N `Min_leep
(mins)`
##    <chr>                   <dbl>                  <dbl> <int>
<dbl>
##  1 1844505072               652                    66.4     3
590
##  2 2026352035               506.                   42.3    28
357
##  3 6117666160               479.                   84.1    18
336
##  4 4319703577               477.                  114.     26
59
##  5 5553957443               463.                  108.     31
322
##  6 7086361926               453.                   69.4    24
322
##  7 6962181067               448                    62.5    31
298
##  8 2347167796               447.                   43.0    15
374
##  9 8378563200               445.                   76.8    31
323
## 10 8792009665               436.                   65.5    15
339
## # ℹ 14 more rows
## # ℹ 1 more variable: `Max_Sleep (mins)` <dbl>

# Calculate percentages for the average sleep durations
below_6_hours <- sum(sleep_df$`Avg_Sleep (mins)` < 360) / nrow(sleep_df) *
100
between_6_7_hours <- sum(sleep_df$`Avg_Sleep (mins)` >= 360 &
sleep_df$`Avg_Sleep (mins)` < 420) / nrow(sleep_df) * 100
at_least_7_hours <- sum(sleep_df$`Avg_Sleep (mins)` >= 420) / nrow(sleep_df)
* 100

# Calculate the count for each category
below_6_count <- sum(sleep_df$`Avg_Sleep (mins)` < 360)
between_6_7_count <- sum(sleep_df$`Avg_Sleep (mins)` >= 360 &
sleep_df$`Avg_Sleep (mins)` < 420)
at_least_7_count <- sum(sleep_df$`Avg_Sleep (mins)` >= 420)

# Create a data frame with enhanced details
percentage_sleep_df <- data.frame(
  Category = c("Below 6 hours", "Between 6 and 7 hours", "At least 7 hours"),
  `Percentage_Users` = round(c(below_6_hours, between_6_7_hours,
at_least_7_hours), 1),
  `Count_Users` = c(below_6_count, between_6_7_count, at_least_7_count)
)
```

```r
# Convert Category to a factor with custom factor levels for proper ordering
percentage_sleep_df$Category <- factor(percentage_sleep_df$Category, levels =
c("Below 6 hours", "Between 6 and 7 hours", "At least 7 hours"))

# Print the enhanced data frame
print(percentage_sleep_df)
```

```
##                Category Percentage_Users Count_Users
## 1         Below 6 hours             33.3           8
## 2 Between 6 and 7 hours             20.8           5
## 3      At least 7 hours             45.8          11
```

```r
# Create the bar plot
ggplot(percentage_sleep_df, aes(x = Category, y =  Percentage_Users, fill =
Category)) +
  geom_bar(stat = "identity", show.legend = FALSE) +  # `stat = "identity"`
uses the y-values directly
  geom_text(aes(label = paste(Percentage_Users, "%")), vjust = .1, hjust =
0.5,  size = 5) +  # Add percentage labels above the bars
  scale_fill_manual(values = c("skyblue", "lightgreen", "salmon")) +  #
Custom colors for bars
  labs(
    title = "Percentage of Users by Sleep Duration Category",
    x = "Sleep Duration Category",
    y = "Percentage of Users (%)"
  ) +
  theme_minimal() +  # A clean, minimal theme
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Angle x-axis
labels for readability
```



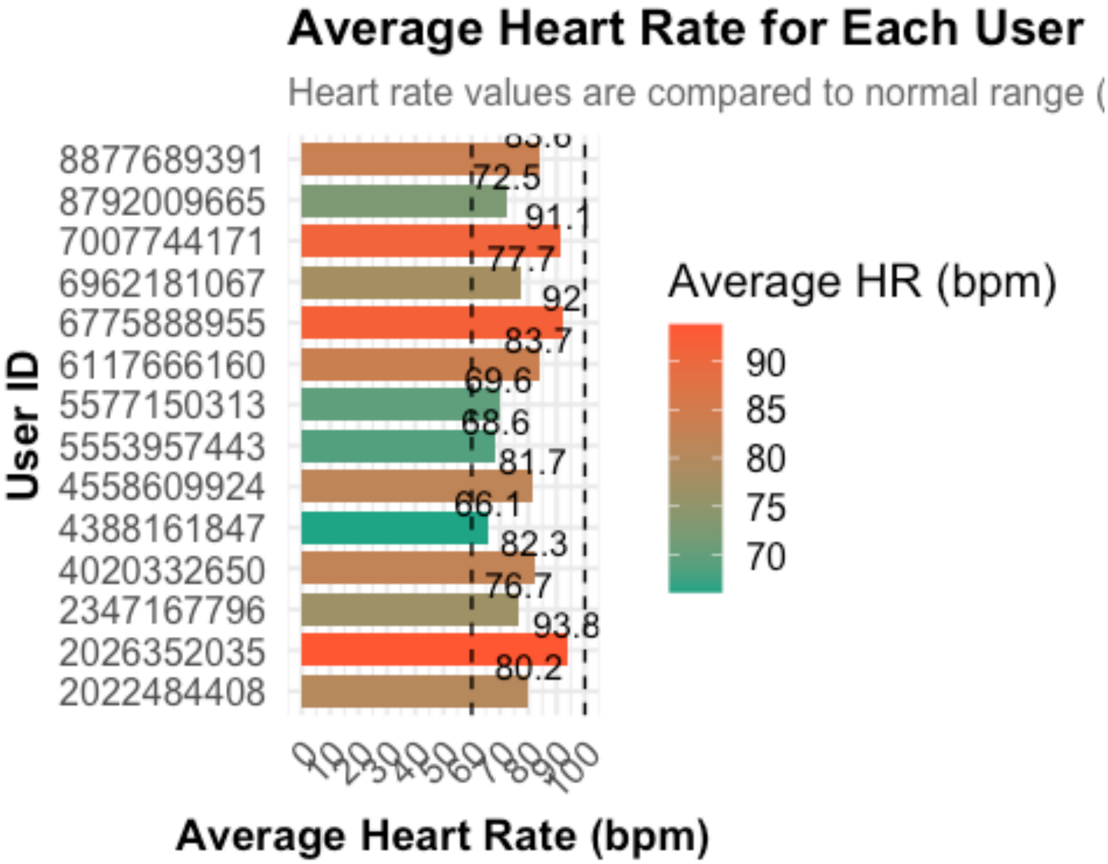
Observations:

A significant portion of users, 53%, have an average daily sleep duration of less than 7 hours.

Analyze heart rate

```
# Group by 'id' and calculate the average heart rate for each user
average_heart_rate <- heartrate_clean %>%
  group_by(id) %>%
  summarise(average_heart_rate = mean(heart_rate, na.rm = TRUE))

# Enhanced bar plot with additional styling
bar_plot <- ggplot(average_heart_rate, aes(x = id, y = average_heart_rate)) +
  geom_bar(stat = "identity", aes(fill = average_heart_rate), width = 0.8) +
# Apply gradient fill based on average heart rate
  scale_fill_gradient(low = "#20A486FF", high = "#FF5733") +  # Gradient fill
from green to red
  labs(
    x = "User ID",
    y = "Average Heart Rate (bpm)",
    title = "Average Heart Rate for Each User",
    subtitle = "Heart rate values are compared to normal range (60-100 bpm)"
  ) +
```

```r
  theme_minimal(base_size = 15) +  # Base font size for better readability
  coord_flip() +  # Flip coordinates for better user ID display
  scale_y_continuous(breaks = seq(0, 100, 10), limits = c(0, 100)) +  # Custom y-axis
  geom_hline(yintercept = 60, linetype = "dashed", color = "black") +  # Normal heart rate lower boundary
  geom_hline(yintercept = 100, linetype = "dashed", color = "black") +  # Normal heart rate upper boundary
  geom_text(aes(label = round(average_heart_rate, 1)), vjust = -0.5, color = "black", size = 4) +  # Show values on top of bars
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),  # Rotate x-axis labels for better fit
    axis.title = element_text(face = "bold", size = 14),  # Bold axis titles
    plot.title = element_text(face = "bold", size = 16),  # Bold title with larger font size
    plot.subtitle = element_text(size = 12, color = "gray40")  # Subtitle text
  ) +
  guides(fill = guide_colorbar(title = "Average HR (bpm)", title.position = "top"))  # Add a colorbar legend for the fill gradient

# Display the bar plot
print(bar_plot)
```



Observation:

Users' average heart rates fall within the normal range, so no notable issues were identified.

However, a potential improvement for the app could be to implement a feature that sends notifications to users whose average resting heart rate falls outside the normal range.