Bellabeat

Hira Afzal

2025-04-11

Background

Bellabeat is a high-tech manufacturer of health-focused products for women. The objective is to analyze Bellabeat's product into gaining insight on how consumers are using their smart devices. The analysis will be used to make high-level recommendations for marketing strategy.

The questions that should guide the analysis 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 marketing strategy?

Data source

Bellabeat data was taken from *FitBit Fitness tracker Data*

License: *CC0: Public Domain*
Data set made available through Mobius

Data is generated from eligible Fitbit users who consented to the use of their personal tracker data that includes physical activity, sleep, calories etc. . .
Data is dated from 03-12-2016 to 05-12-2016

Potential Bias and Limitations
- Small sample size of 33 users could have limitations of statistical analysis with false trends and high variability risk.
- The eligible Fitbit users who chose to share data may differ from those who did not (e.g, more health-conscious).
- Temporal Biases: March to May Data is provided and don't capture seasonal changes (e.g, Spring leading to more fitness as weather warms but unclear about summer).

Installing Packages

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

Import Data

Importing March - April and April - May Datasets

— Importing daily_activity data from March - April & April - May data sets —

```
mar_apr_daily_activity <- read_csv('mar_apr_dailyActivity_merged.csv')
apr_may_daily_activity <- read_csv('apr_may_dailyActivity_merged.csv')
```

```
head(mar_apr_daily_activity, n = 2)
```

```
## # A tibble: 2 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
## # 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(apr_may_daily_activity, n = 2)
```

```
## # A tibble: 2 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
## # i 10 more variables: LoggedActivitiesDistance <dbl>,
## #   VeryActiveDistance <dbl>, ModeratelyActiveDistance <dbl>,
## #   LightActiveDistance <dbl>, SedentaryActiveDistance <dbl>,
## #   VeryActiveMinutes <dbl>, FairlyActiveMinutes <dbl>,
## #   LightlyActiveMinutes <dbl>, SedentaryMinutes <dbl>, Calories <dbl>
```

— Importing hourly intensity and steps from March - April & April - May data sets —

```
mar_apr_hourly_intensities <- read_csv('mar_apr_hourlyIntensities_merged.csv')
mar_apr_hourly_steps <- read_csv('mar_apr_hourlySteps_merged.csv')
apr_may_hourly_intensities <- read_csv('apr_may_hourlyIntensities_merged.csv')
apr_may_hourly_steps <- read_csv('apr_may_hourlySteps_merged.csv')
```

```
head(mar_apr_hourly_intensities, n = 2)
```

```
## # A tibble: 2 x 4
##           Id ActivityHour          TotalIntensity AverageIntensity
##        <dbl> <chr>                          <dbl>             <dbl>
## 1 1503960366 3/12/2016 12:00:00 AM              0                 0
```

```
## 2 1503960366 3/12/2016 1:00:00 AM               0                0
```

```
head(mar_apr_hourly_steps, n = 2)
```

```
## # A tibble: 2 x 3
##           Id ActivityHour          StepTotal
##        <dbl> <chr>                     <dbl>
## 1 1503960366 3/12/2016 12:00:00 AM         0
## 2 1503960366 3/12/2016 1:00:00 AM          0
```

```
head(apr_may_hourly_intensities, n = 2)
```

```
## # A tibble: 2 x 4
##           Id ActivityHour          TotalIntensity AverageIntensity
##        <dbl> <chr>                          <dbl>            <dbl>
## 1 1503960366 4/12/2016 12:00:00 AM             20            0.333
## 2 1503960366 4/12/2016 1:00:00 AM               8            0.133
```

```
head(apr_may_hourly_steps, n = 2)
```

```
## # A tibble: 2 x 3
##           Id ActivityHour          StepTotal
##        <dbl> <chr>                     <dbl>
## 1 1503960366 4/12/2016 12:00:00 AM       373
## 2 1503960366 4/12/2016 1:00:00 AM        160
```

— Importing sleep data in minutes from March - April & April - May data sets —

```
mar_apr_minute_sleep <- read_csv('mar_apr_minuteSleep_merged.csv')
apr_may_minute_sleep <- read_csv('apr_may_minuteSleep_merged.csv')
```

```
head(mar_apr_minute_sleep, n = 2)
```

```
## # A tibble: 2 x 4
##           Id date                 value       logId
##        <dbl> <chr>                <dbl>       <dbl>
## 1 1503960366 3/13/2016 2:39:30 AM     1 11114919637
## 2 1503960366 3/13/2016 2:40:30 AM     1 11114919637
```

```
head(apr_may_minute_sleep, n = 2)
```

```
## # A tibble: 2 x 4
##           Id date                 value       logId
##        <dbl> <chr>                <dbl>       <dbl>
## 1 1503960366 4/12/2016 2:47:30 AM     3 11380564589
## 2 1503960366 4/12/2016 2:48:30 AM     2 11380564589
```

— Importing logged weight data from March - April & April - May data sets —

```
mar_apr_weight <- read_csv('mar_apr_weightLogInfo_merged.csv')
apr_may_weight <- read_csv('apr_may_weightLogInfo_merged.csv')
```

```
head(mar_apr_weight, n = 2)
```

```
## # A tibble: 2 x 8
##           Id Date       WeightKg WeightPounds   Fat   BMI IsManualReport   LogId
##        <dbl> <chr>         <dbl>        <dbl> <dbl> <dbl> <lgl>            <dbl>
## 1 1503960366 4/5/2016 ~     53.3         118.    22  23.0 TRUE           1.46e12
```

```
## 2 1927972279 4/10/2016~    130.          286.    NA  46.2 FALSE          1.46e12
```

```
head(apr_may_weight, n = 2)
```

```
## # A tibble: 2 x 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
```

Data Exploration

Perform data exploration analysis to determine quality of data and identify data requiring cleaning

— Identify duplicate records in data —

```
sum(duplicated(mar_apr_daily_activity))
```

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

```
sum(duplicated(apr_may_daily_activity))
```

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

```
sum(duplicated(apr_may_hourly_intensities))
```

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

```
sum(duplicated(mar_apr_hourly_intensities))
```

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

```
sum(duplicated(apr_may_hourly_steps))
```

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

```
sum(duplicated(mar_apr_hourly_steps))
```

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

```
sum(duplicated(mar_apr_weight))
```

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

```
sum(duplicated(apr_may_weight))
```

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

```
sum(duplicated(mar_apr_minute_sleep))
```

```
## [1] 525
```

```
sum(duplicated(apr_may_minute_sleep))
```

```
## [1] 543
```

— Identify null values in data —

```
sum(is.na(apr_may_daily_activity))
```

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

```
sum(is.na(mar_apr_daily_activity))
```

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

```
sum(is.na(apr_may_hourly_intensities))
```

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

```
sum(is.na(mar_apr_hourly_intensities))
```

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

```r
sum(is.na(apr_may_hourly_steps))
```

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

```r
sum(is.na(mar_apr_hourly_steps))
```

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

```r
sum(is.na(apr_may_minute_sleep))
```

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

```r
sum(is.na(mar_apr_minute_sleep))
```

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

```r
sum(is.na(mar_apr_weight))
```

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

```r
sum(is.na(apr_may_weight))
```

```
## [1] 65
```

— Cross-reference users between daily_activity datasets to identify users missing from either datasets —

```r
unique(anti_join(mar_apr_daily_activity, apr_may_daily_activity, by = 'Id')$Id)
```

```
## [1] 2891001357 6391747486
```

— Cross-reference users between hourly_steps and hourly_intensities datasets to identify users missing from either datasets —

```r
unique(anti_join(mar_apr_hourly_steps, apr_may_hourly_steps, by = 'Id')$Id)
```

```
## [1] 2891001357 6391747486
```

```r
unique(anti_join(mar_apr_hourly_intensities, apr_may_hourly_intensities, by = 'Id')$Id)
```

```
## [1] 2891001357 6391747486
```

— Cross-reference users between minute_sleep and weight datasets to identify users missing from either datasets —

```r
n_distinct(mar_apr_minute_sleep$Id)
```

```
## [1] 23
```

```r
n_distinct(apr_may_minute_sleep$Id)
```

```
## [1] 24
```

```r
n_distinct(mar_apr_weight$Id)
```

```
## [1] 11
```

```r
n_distinct(apr_may_weight$Id)
```

```
## [1] 8
```

Data to be cleaned:

- Standardize date columns from a char to a date object field.
- Remove duplicated data from mar_apr_minute_sleep_data and apr_may_minute_sleep_data
- Remove users that are not represented in both monthly datasets for daily_activity, hourly_steps and hourly_intensities

- *Preserving null values from weight datasets as it is user logged and reflect user behavior*
- *Retaining all user records from sleep and weight data set as it reflects user behavior*

Data Combining and Cleaning

Merging and preparing datasets for analysis

— Combine mar_apr_daily_activity and apr_may_daily_activity for users that exist in both data sets. Convert ActivityDate field from char to Date. —

```
ids_to_filter <- c(2891001357, 6391747486)

daily_activity <- bind_rows(mar_apr_daily_activity, apr_may_daily_activity) %>%
  filter(!Id %in% ids_to_filter) #combining data sets

daily_activity <- transform(daily_activity, ActivityDate = mdy(ActivityDate)) #char to Date
head(daily_activity, n = 2)
```

```
##           Id ActivityDate TotalSteps TotalDistance TrackerDistance
## 1 1503960366   2016-03-25      11004          7.11            7.11
## 2 1503960366   2016-03-26      17609         11.55           11.55
##   LoggedActivitiesDistance VeryActiveDistance ModeratelyActiveDistance
## 1                        0              2.57                     0.46
## 2                        0              6.92                     0.73
##   LightActiveDistance SedentaryActiveDistance VeryActiveMinutes
## 1                4.07                       0                33
## 2                3.91                       0                89
##   FairlyActiveMinutes LightlyActiveMinutes SedentaryMinutes Calories
## 1                  12                  205              804     1819
## 2                  17                  274              588     2154
```

— Combine mar_apr_hourly_activity and apr_may_hourly_activity for users that exist in both data sets. Convert ActivityHour field from char to Date. Create separate column to store date and hour from ActivityHour column. —

```
apr_may_hourly_activity <- merge(apr_may_hourly_intensities, apr_may_hourly_steps,
                                 by = c("Id", "ActivityHour")) #joining data set by id and date

mar_apr_hourly_activity <- merge(mar_apr_hourly_intensities, mar_apr_hourly_steps,
                                 by = c("Id", "ActivityHour")) #joining data set by id and date

hourly_activity <- bind_rows(mar_apr_hourly_activity, apr_may_hourly_activity) %>%
  filter(!Id %in% ids_to_filter) #combining data sets

hourly_activity <- hourly_activity %>%
  mutate(ActivityHour = mdy_hms(ActivityHour),
         date = as.Date(ActivityHour),
         hour = hour(ActivityHour)) #char to Date

head(hourly_activity, n = 2)
```

```
##           Id        ActivityHour TotalIntensity AverageIntensity StepTotal
## 1 1503960366 2016-03-12 01:00:00              0         0.000000         0
## 2 1503960366 2016-03-12 13:00:00             20         0.333333       253
##         date hour
## 1 2016-03-12    1
## 2 2016-03-12   13
```

— Remove duplicated records from both datasets. Sum each users total sleep (filter value to 1 which is indication that user is asleep) and store it as a separate column. Convert date field from char to Date. Generate separate date column from date (mdy_hms) field. Combine both datasets. —

```
#removing duplication
mar_apr_minute_sleep_cleaned <- mar_apr_minute_sleep %>% distinct()
apr_may_minute_sleep_cleaned <- apr_may_minute_sleep %>% distinct()

df <- apr_may_minute_sleep_cleaned %>%
  filter(value==1) %>%  #value of 1 is equal to user sleeping
  mutate(
    # Convert to datetime
    datetime = mdy_hms(date),
    # Extract date component
    date_only = as.Date(datetime)
  ) %>%
  group_by(Id, date_only) %>%
  summarise(
    totalsleep = sum(value, na.rm = TRUE), #total sleep added
    distinct_logIds = n_distinct(logId, na.rm = TRUE) #distinct times sleep was logged
  ) %>%
  ungroup()

df_two <- mar_apr_minute_sleep_cleaned %>%
  filter(value==1) %>%
  mutate(
    # Convert to datetime
    datetime = mdy_hms(date),
    date_only = as.Date(datetime)
  ) %>%
  group_by(Id, date_only) %>%
  summarise(
    totalsleep = sum(value, na.rm = TRUE),
    distinct_logIds = n_distinct(logId, na.rm = TRUE)
  ) %>%
  ungroup()

sleep_data <- bind_rows(df, df_two) #combining data sets
head(sleep_data, n = 2)
```

```
## # A tibble: 2 x 4
##           Id date_only  totalsleep distinct_logIds
##        <dbl> <date>          <dbl>           <int>
## 1 1503960366 2016-04-12        327               1
## 2 1503960366 2016-04-13        384               2
```

— Combine March - April and April - May weight data set. —

```
weight_data <- bind_rows(mar_apr_weight, apr_may_weight) #combining data sets
head(weight_data, n = 2)
```

```
## # A tibble: 2 x 8
##           Id Date       WeightKg WeightPounds   Fat   BMI IsManualReport   LogId
##        <dbl> <chr>         <dbl>        <dbl> <dbl> <dbl> <lgl>            <dbl>
```

```
## 1 1503960366 4/5/2016 ~     53.3         118.    22  23.0 TRUE          1.46e12
## 2 1927972279 4/10/2016~    130.          286.    NA  46.2 FALSE         1.46e12
```

Data Visualization and Analysis

Performing data visualization techniques to identify trends, followed by analysis to validate insights

— Identify trend of how consistent user was in wearing Fitbit across the 2 months by analyzing how many days with zero steps each user had —

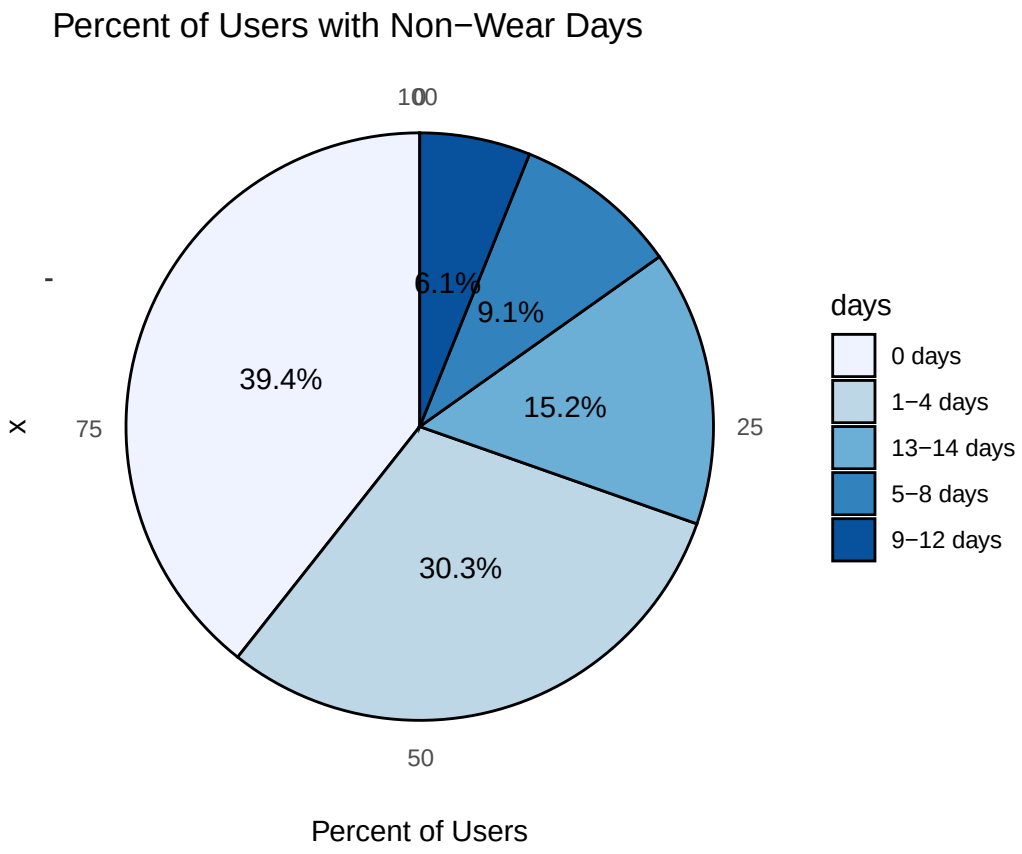
```
# data frame with total zero step of each user
daily_users <- daily_activity %>%
  group_by(Id) %>%
  summarize(
    days_with_zero_steps = sum(TotalSteps == 0)
  )

#creating subset of groups and setting user to each group based on their total zero steps
daily_users$days <- ifelse(daily_users$days_with_zero_steps == 0, "0 days",
      ifelse(daily_users$days_with_zero_steps <= 4, "1-4 days",
      ifelse(daily_users$days_with_zero_steps <= 8, "5-8 days",
      ifelse(daily_users$days_with_zero_steps <= 12, "9-12 days","13-14 days"))))

# data frame of percentage of each subset
df_daily_users_percent <- daily_users %>%
  group_by( days ) %>%
  summarise( percent = 100 * n() / nrow( daily_users ) )

#1 decimal point only
df_daily_users_percent <- df_daily_users_percent %>%
  mutate(percent = as.numeric(format(round(df_daily_users_percent$percent, 1)), 1))

#pie chart
ggplot(df_daily_users_percent, aes(x = "", y = percent, fill = days)) +
  geom_col(color = "black") +
  geom_text(aes(label = paste0(percent,"%")),
            position = position_stack(vjust = 0.5)) +
  coord_polar(theta = "y") +
  scale_fill_brewer() +
  labs(y= 'Percent of Users',title = "Percent of Users with Non-Wear Days") +
  theme(panel.background = element_rect(fill = "white"))
```



Summary:
The majority of users(39.4%) recorded consistent Fitbit use with no non-wear days which indicate strong habit formation or high motivation subset group. Users that had exhibited limited non-wear period (1-4 days) were 30.4% which is a mostly consistent subset of group.The third highest subset group that had inactivity were the user that had a significantly higher inactivity of 13-14 days which accounted for 15.2% suggesting partial abandonment and disengagement.

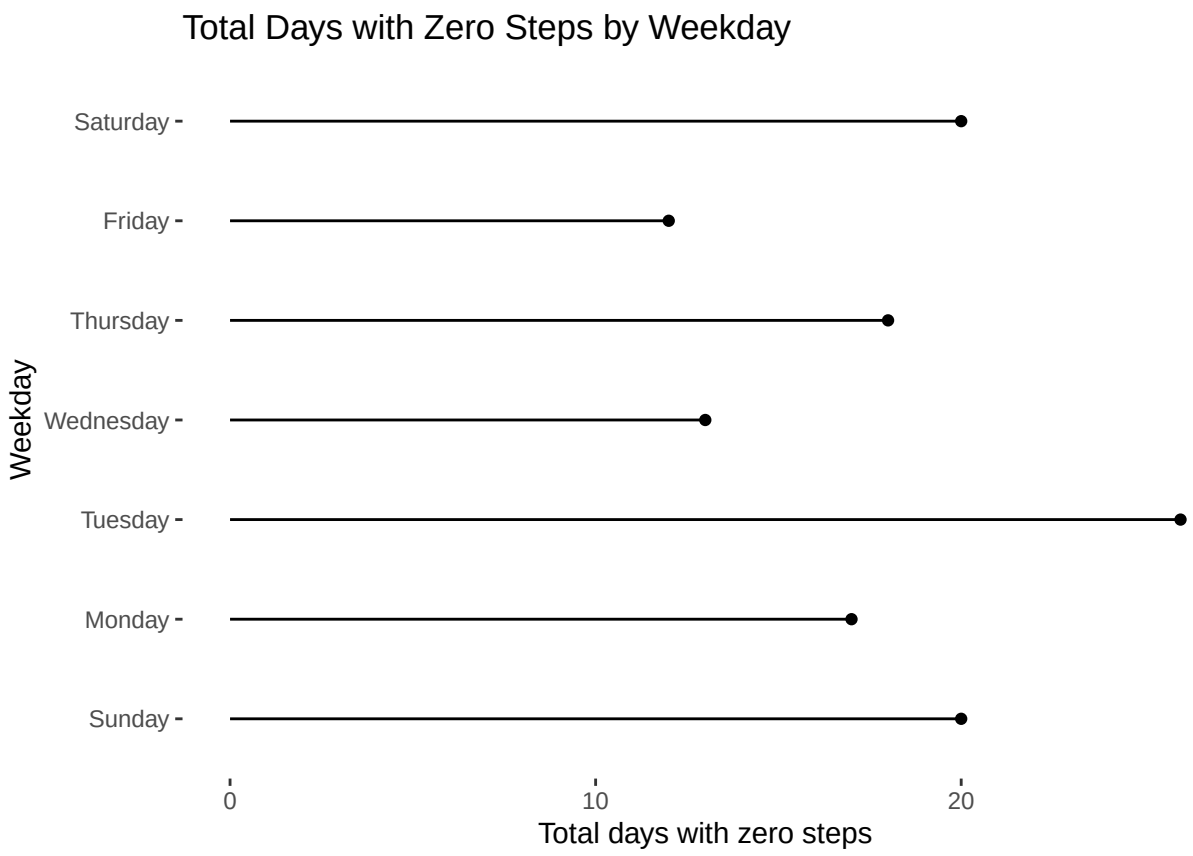
— Identify trend of how consistent user was in wearing Fitbit across the 2 months by analyzing how many days with zero steps each user had across the week —

```
#extracting weekday from Date
daily_activity <- daily_activity %>%
  mutate(weekday = weekdays(ActivityDate))

#data frame of each users total zero steps by weekday
daily_users_wd <- daily_activity %>%
  group_by(Id, weekday) %>%
  summarize(
    days_with_zero_steps = sum(TotalSteps == 0)
  )

#Total zero steps each weekday
df_daily_users_percent_wd <- daily_users_wd %>%
  group_by( weekday ) %>%
  summarise(days_zero_step = sum(days_with_zero_steps))

#order the weekdays
df_daily_users_percent_wd$weekday <- factor(df_daily_users_percent_wd$weekday,
                                        levels=c('Sunday', 'Monday', 'Tuesday',
                                                 'Wednesday', 'Thursday',
                                                 'Friday', 'Saturday'))
#lollipop chart
ggplot(df_daily_users_percent_wd, aes(x = days_zero_step, y = weekday)) +
  geom_segment(aes(x = 0, y = weekday, xend = days_zero_step, yend = weekday)) +
  geom_point() +
  labs(x = 'Total days with zero steps', y = 'Weekday', title = "Total Days with Zero Steps by Weekday")
  theme(panel.background = element_rect(fill = "white"))
```



Summary:
Tuesday showed the highest frequency of inactive days with over 25 days, followed closely by weekends (Saturday and Sunday) with around 20 days for each day. In contrast Wednesday and Fridays recorded the lowest inactivity rate (less than 15 days) while Mondays and Thursday had moderate inactivity occurrence (more than 15 days).

— Identifying user engagement pattern by measuring conversion rate from wearing Fitbit device to user logging sleep and weight data —

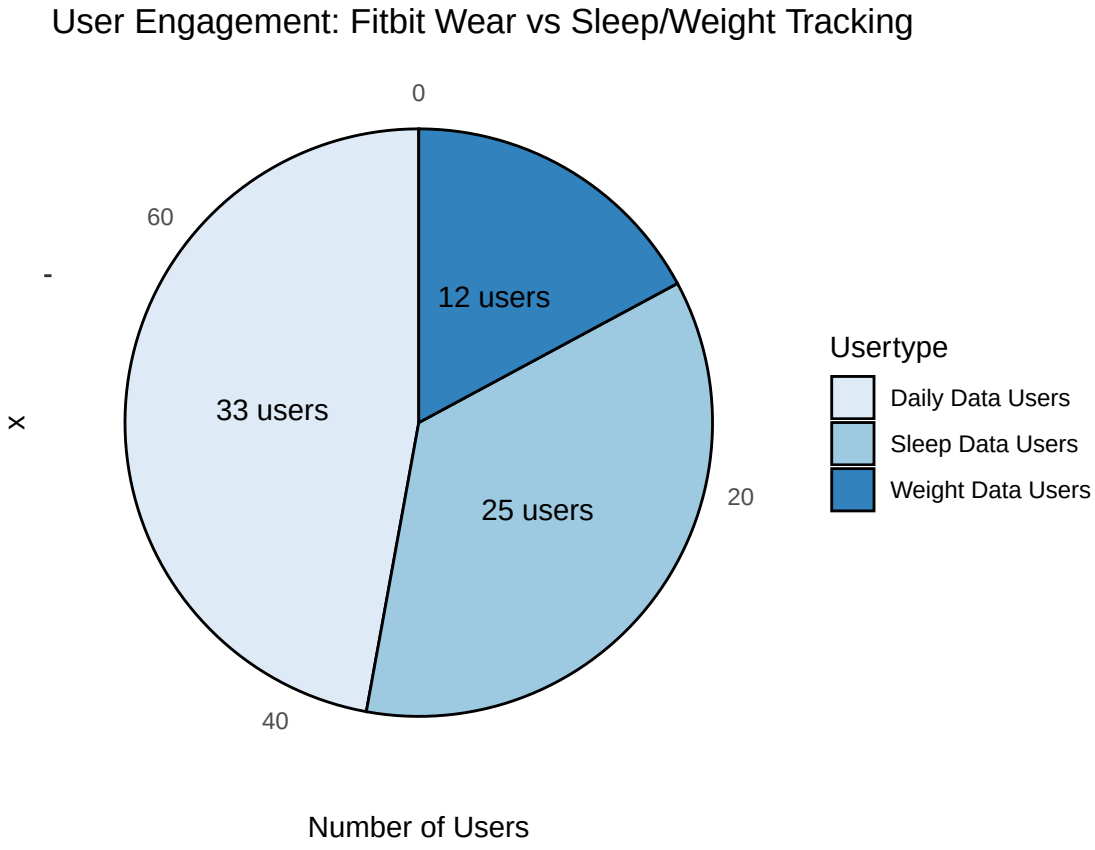
```r
#confirming if sleep data sets users match daily_activity users
sleep_id_check <- sleep_data$Id %in% daily_activity$Id  #all match

#confirming if weight data sets users match daily_activity users
weight_id_check <- weight_data$Id %in% daily_activity$Id  #1 inconsistency

#taking out the inconsistency
weight_data <-  weight_data %>%
  filter(!Id %in% '2891001357')

#data frame of each user and qty of each user
users <- data.frame(Usertype = c("Daily Data Users",
                                 "Sleep Data Users", "Weight Data Users"),
                    num_of_user = c(n_distinct(daily_activity$Id),
                                    n_distinct(sleep_data$Id),
                                    n_distinct(weight_data$Id)))

#pie chart
ggplot(users, aes(x = "", y = num_of_user, fill = Usertype)) +
  geom_col(color = "black") +
  geom_text(aes(label = paste0(num_of_user," users")),
            position = position_stack(vjust = 0.5)) +
  coord_polar(theta = "y") +
  scale_fill_brewer() +
  labs(y= 'Number of Users',title = "User Engagement: Fitbit Wear vs Sleep/Weight Tracking") +
  theme(panel.background = element_rect(fill = "white"))
```



Summary:
In total of the 33 users that wore FitBit device, 13 users (39.4%) recorded weight data while 25 users (75.8 %) recorded sleep data. This reflects strong interest for sleep monitoring and prioritize sleep duration and quality over weight. This is also an indication of passive tracking (sleep) vs manual entry (weight).

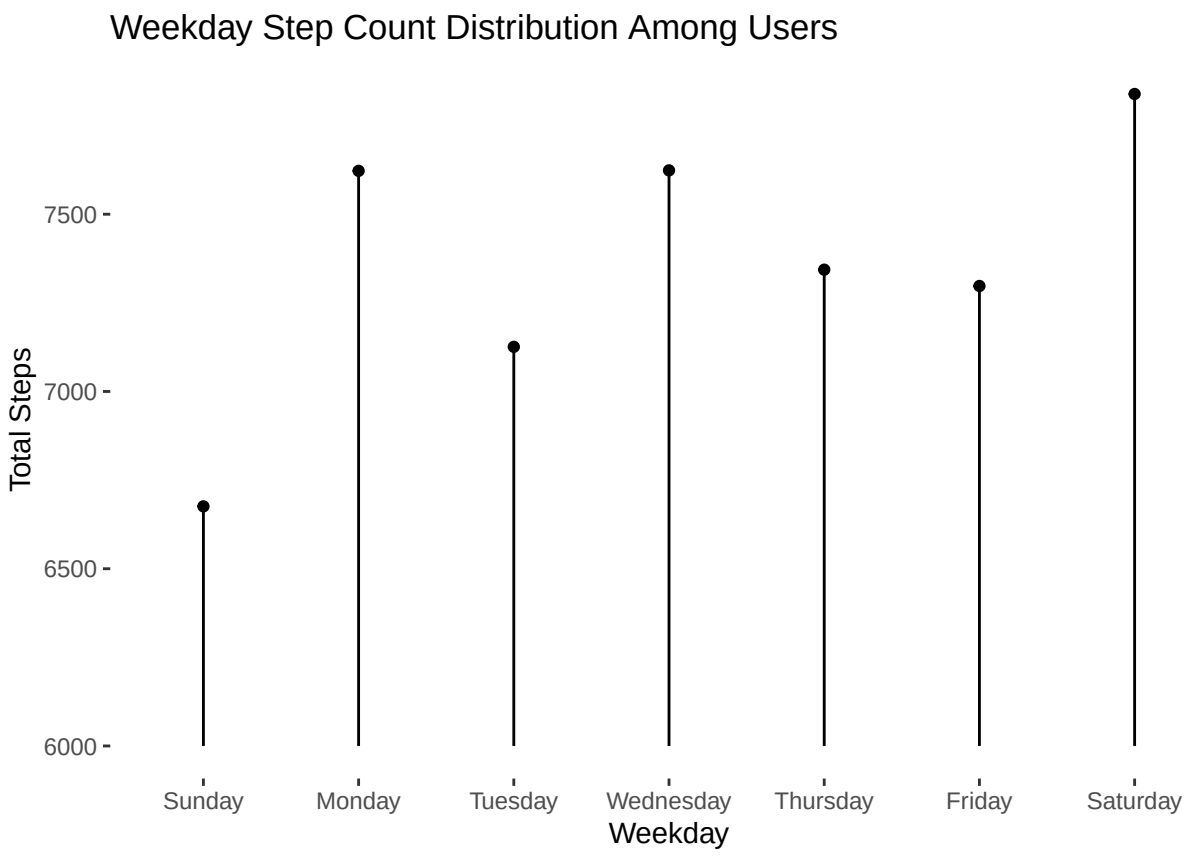
— Analyzing users weekly physical activity trends by variations in step count across weekday —

```r
#extracting weekday
daily_activity <- daily_activity %>%
  mutate(weekday = weekdays(ActivityDate))

#average of each weekday's step count
avg_weekday <- group_by(daily_activity, weekday) %>%  summarise(total_steps=
                                                      mean(TotalSteps))

#order of weekday
avg_weekday$weekday <- factor(avg_weekday$weekday, levels=c('Sunday', 'Monday',
                                                    'Tuesday','Wednesday',
                                                    'Thursday', 'Friday',
                                                    'Saturday'))
#2 decimal point
avg_weekday <- avg_weekday %>%
  mutate(total_steps = as.numeric(format(round(avg_weekday$total_steps, 2)), 2))

#lollipop chart
ggplot(avg_weekday, aes(x = total_steps, y = weekday)) +
  geom_segment(aes(x = 6000, xend = total_steps, y = weekday, yend = weekday)) +
  geom_point() +
  coord_flip() +
  labs(x = 'Total Steps', y = 'Weekday', title = "Weekday Step Count Distribution Among Users") +
  theme(panel.background = element_rect(fill = "white"))
```



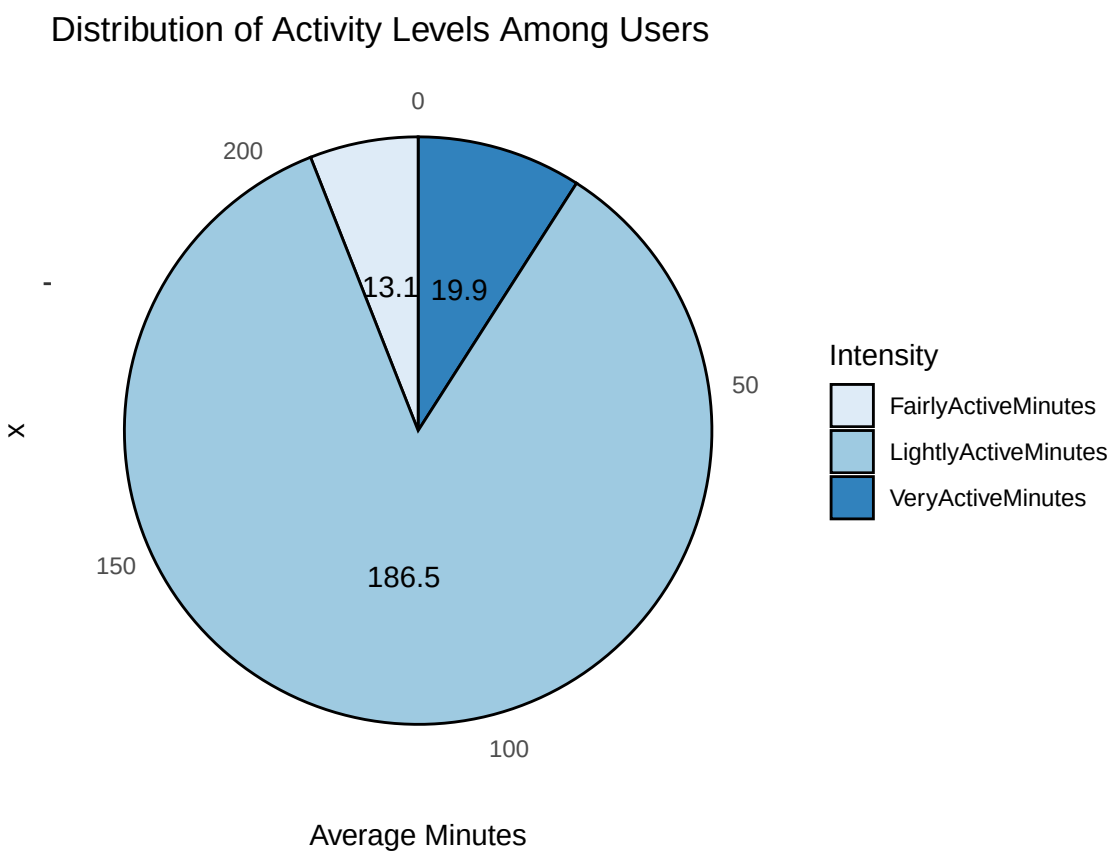
Summary:
Step counts peaked on Saturdays (7800 steps) with similar high activity on Monday and Wednesday which exceeded 7500 steps. Weekend peak of Saturday suggests workouts, errands or social outing for users. A Monday high suggests a structured routines of gym session or active commute while a Wednesday high suggests a midweek energy rebound. Midweek Days (Tuesday, Thursday and Friday) maintained moderate steps at 7000+ steps that could reflect work fatigue while Sunday showed the lowest activity (6500-7000) which suggests rest and relaxation day after a Saturday of high activity.

— Analyzing users activity levels by average time spent very active, fairly active and lightly active —

```r
#data frame of each intensity's average across all user
avg_intensity <- apply(daily_activity[ , c('VeryActiveMinutes',
                                           'FairlyActiveMinutes',
                                           'LightlyActiveMinutes')], 2, mean)
#naming columns
avg_intensity <- enframe(avg_intensity, name = 'Intensity', value = 'Average')

#1 decimal
avg_intensity <- avg_intensity %>%
  mutate(Average = as.numeric(format(round(avg_intensity$Average, 1)), 1))

#pie chart
ggplot(avg_intensity, aes(x = "", y = Average, fill = Intensity)) +
  geom_col(color = "black") +
  geom_text(aes(label = Average),
            position = position_stack(vjust = 0.5)) +
  coord_polar(theta = "y") +
  scale_fill_brewer() +
   labs(y = 'Average Minutes', title = "Distribution of Activity Levels Among Users") +
  theme(panel.background = element_rect(fill = "white"))
```



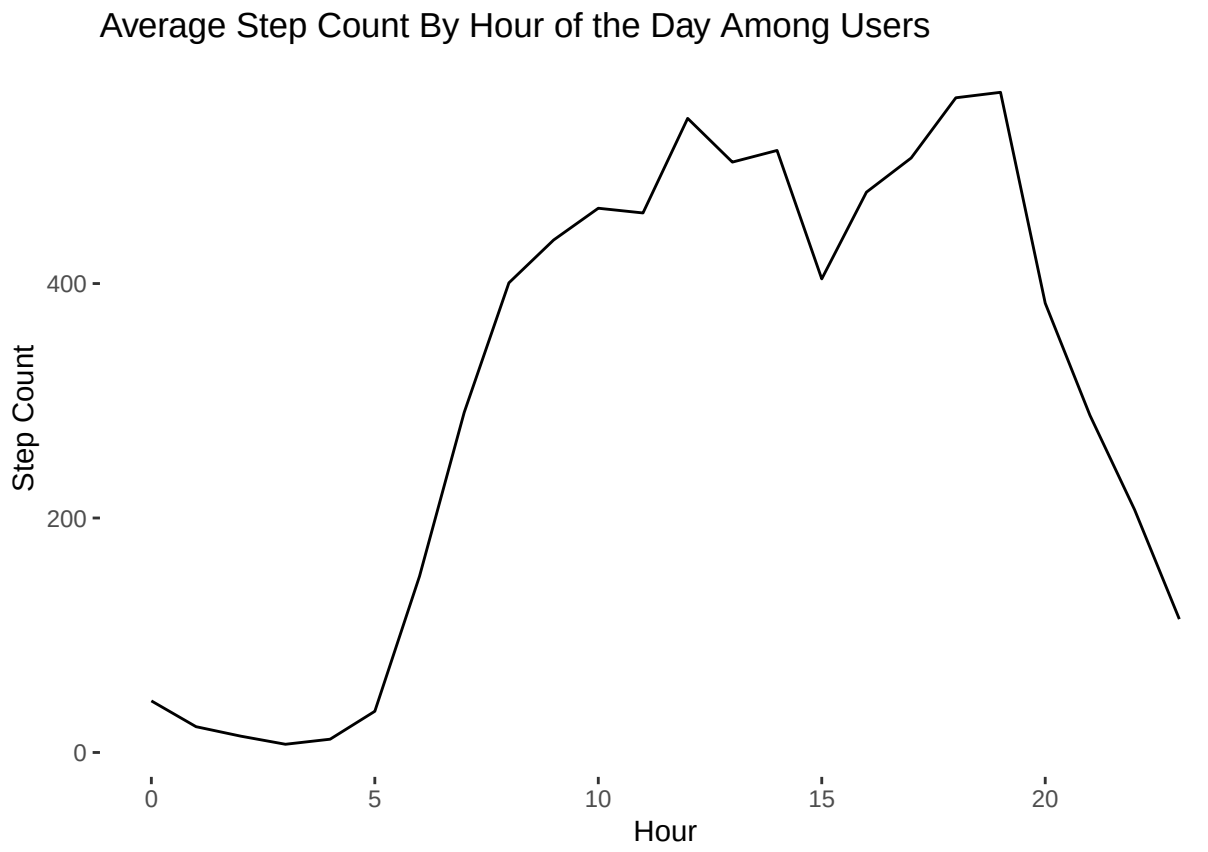
Summary:
Users 186.5 minutes lightly active minutes daily (3.1 hours), constituting the majority of their movement. However more users spent being Very Active (19.9 minutes) than they were fairly Active (13.1 minutes). Users engaged in 53% more very active intensity than Fairly active (19.9 vs 13.1 minutes). It suggests a preference for high-intensity vigorous exercise than moderate exercise.

— Analyzing users weekly physical activity trends by variations in step count across 24 hours —

```r
#data frame of average step count by the hour
avg_step_hour <- group_by(hourly_activity, hour) %>%
  summarise(avg_steps = mean(StepTotal))

#2 decimal point
avg_step_hour <- avg_step_hour %>%
  mutate(avg_steps = as.numeric(format(round(avg_step_hour$avg_steps, 2)), 2))

#line graph
ggplot(avg_step_hour, aes(x = hour, y = avg_steps)) +
  geom_line() +
  labs(x = 'Hour', y = 'Step Count',title = "Average Step Count By Hour of the Day Among Users") +
  theme(panel.background = element_rect(fill = "white"))
```



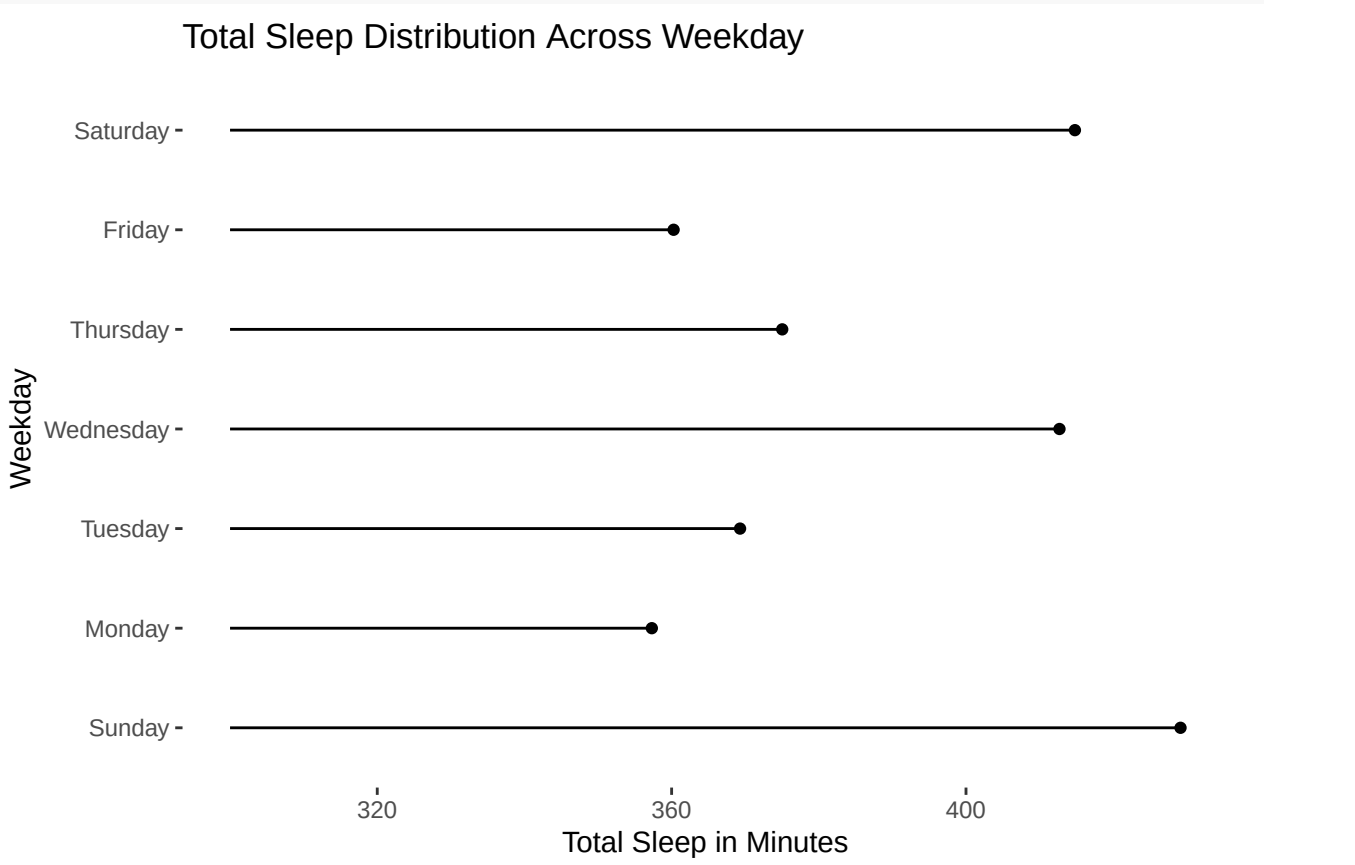
Summary:
Users tend to have their highest step count between 6 and 7 PM, likely due to evening commutes and post-work activity. Another peak occurs around 12-2 PM, coinciding with lunchtime. From 8-11 AM, step counts remain moderate (over 400 steps) with morning commutes and workday routines. The lowest afternoon activity is observed at 3 PM. After 8-9 PM, steps decline sharply, dropping below 350 as the downward trend begins which suggests relaxation or reduced movement at home.

— Analyzing users sleep duration pattern across the weekday —

```r
#extracting weekday
df_sleep_data <- sleep_data %>%
  mutate(weekday = weekdays(date_only))

#each weekdays average total sleep
avg_weekday_sleep <- group_by(df_sleep_data, weekday) %>%
  summarise(totalsleep= mean(totalsleep))

#order weekday
avg_weekday_sleep$weekday <- factor(avg_weekday_sleep$weekday, levels=c('Sunday',
                                                                        'Monday', 'Tuesday',
                                                                        'Wednesday', 'Thursday',
                                                                        'Friday', 'Saturday'))
#2 decimal point
avg_weekday_sleep <- avg_weekday_sleep %>%
  mutate(totalsleep = as.numeric(format(round
                                        (avg_weekday_sleep$totalsleep, 2)), 2))
#lollipop chart graph
ggplot(avg_weekday_sleep, aes(x = totalsleep, y = weekday)) +
  geom_segment(aes(x = 300, y = weekday, xend = totalsleep, yend = weekday)) +
  geom_point() +
  labs(x = 'Total Sleep in Minutes', y = 'Weekday', title =
                                                    "Total Sleep Distribution Across Weekday") +
  theme(panel.background = element_rect(fill = "white"))
```

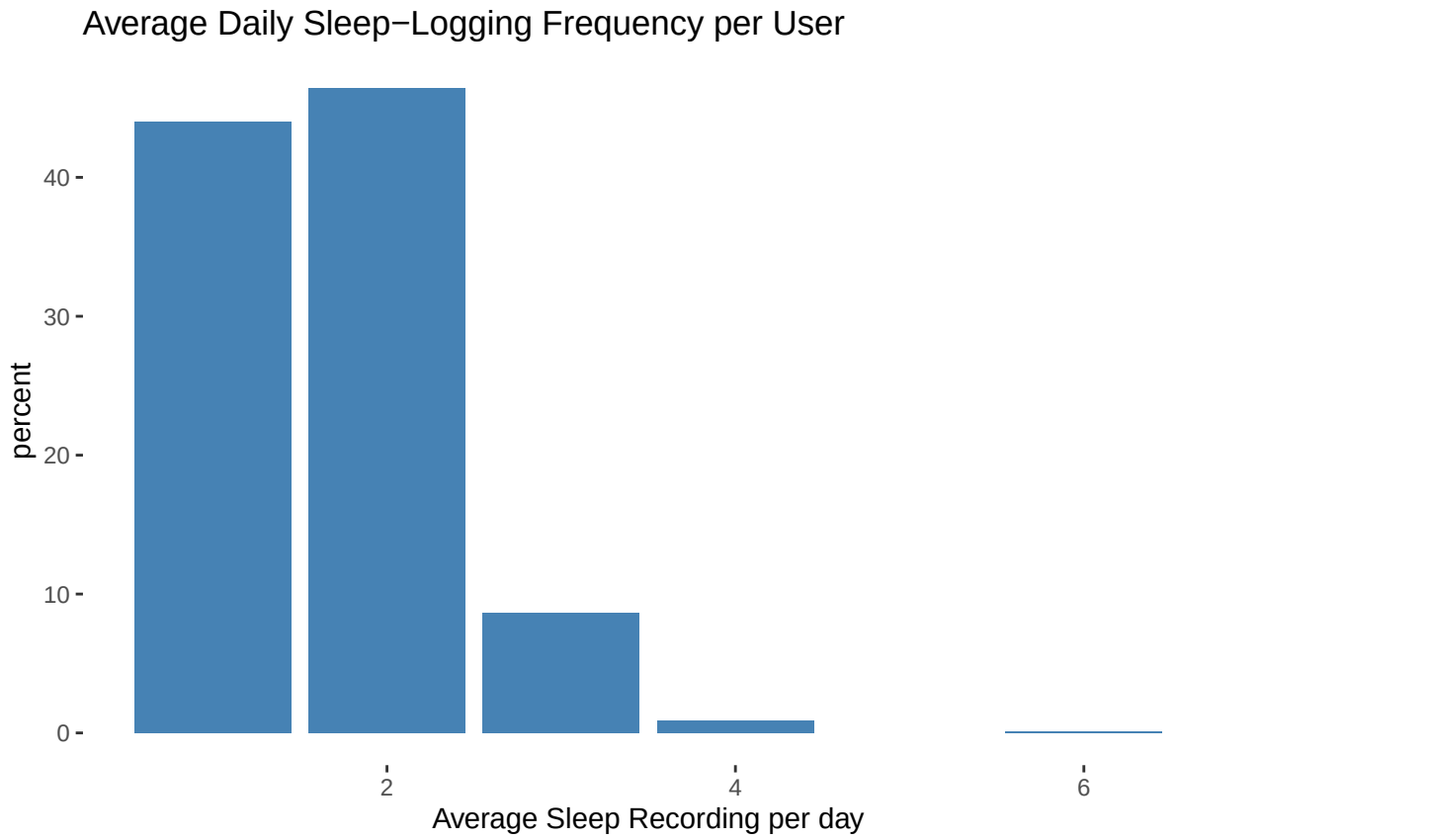


Summary:
Users obtain 400+ minutes (around 7 hours) of sleep on the weekends which indicates weekend recovery of catching up on sleep. They also hit a peak on Wednesday with another 7 hours of sleep on average which reflects a midweek recovery period. While Monday, Tuesday, Thursday and Friday stays less than 375 minutes (6.25 hour).

— Asses sleep tracking of users by computing the average number of daily sleep recording across user —

```r
#data frame of each distinct log frequency
sleep_df <- sleep_data %>%
  group_by( distinct_logIds ) %>%
  summarise( percent = 100 * n() / nrow( sleep_data ) )

#bar graph
ggplot(sleep_df, aes(x = distinct_logIds, y = percent )) +
  geom_bar(fill = '#4682b4', stat = "identity") +
  labs(x = 'Average Sleep Recording per day', title = "Average Daily Sleep-Logging Frequency per User") +
  theme(panel.background = element_rect(fill = "white"))
```



Summary:
An average sleep tracking of users distributed around 1-2 daily sleep recording (40% each) while three recordings were around 10% of the time and four recording was rare (less than 1%). A two and three daily sleep record indicates nappers.

Recommendations

Streaks, Rewards, and Premium

For consistent users who wore device without breaks are suggestive of high motivation of fitness and health tracking:
- Reinforce habits with rewards and streaks
- Most engaged group and ideal for retention and premium features.

For moderate skippers who are mostly consistent but take short breaks are suggestive of an opportunity for improvement:
- Wear time reminders and streaks to encourage consistency
- Potential Loyalist and also can be ideal for retention and premium features

For high-inactive users who are not wearing device for more than 13-14 days show disengagement and abandonment.
- Targeted re-engagement with personalized incentives
- Targeted re-motivation with alerts and progress summary

Introduce Low Effort Weight Tracking

Users prefer automated or low-effort tracking (sleep) over manual inputs (weight).
- Syncing with smart scales to eliminating manual entry
- Reward or streak for consistent logging of weight
- Reminder alerts and progress summary of weights to retain users who are logging weight

Encourage Sleep Monitoring

There is interest in sleep monitoring among users as 75.8% of users chose to sleep monitor.
- Develop weekday sleep hygiene interventions targeting Monday/Tuesday and Thursday/Friday
- Reminder alerts to emphasize importance of 7 hour of sleep
- Provide progress summary of light sleep, deep sleep, REM sleep through out the night to entice users to use sleep feature and retain users who do.

Physical Activity

Promote activity throughout the week:
- Encourage Consistency by targeting Sundays and midweek lulls with prompts (e.g., 'time for a short walk', stretch breaks).
- Leverage High-Activity Days by Reinforcing habits on Saturdays/Mondays/Wednesdays when motivation is naturally higher.