Capstone: Bellabeats Fitness Analysis

BellaBeats Fitness App Analysis

In this project I will be analyzing the cleaned data set for BellaBeats. I have previous cleaned the data using Google Sheets.

I will begin by importing the tidyverse package.

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

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

Loading the Packages

Next I will load the packages.

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
fitness_df <- read_csv("capstone_activity_data_cleaned.csv")
```

Importing the Cleaned CSV Files

```
## Rows: 940 Columns: 16
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): ActivityDate, Weekday
## dbl (11): Id, TotalDistance, TrackerDistance, LoggedActivitiesDistance, Very...
## num  (3): TotalSteps, SedentaryMinutes, 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.
```

```r
hourly_calories <- read_csv("hourlyCalories_merged_cleaned.csv")
```

```
## Rows: 22099 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): ActivityHour, Date
## dbl  (3): Id, Calories, HourNumber
## time (1): Hour
##
## i Use `spec()` to retrieve the full column specification for this data.
```

```
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
hourly_intensity <- read_csv("hourlyIntensities_merged_cleaned.csv")
```

```
## Rows: 22099 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): ActivityHour, Date
## dbl  (4): Id, TotalIntensity, AverageIntensity, HourNumber
## time (1): Time
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
hourly_steps <- read_csv("hourlySteps_merged_cleaned.csv")
```

```
## Rows: 22099 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): ActivityHour, Date
## dbl  (3): Id, StepTotal, HourNumber
## time (1): Hour
##
## 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.
```

Viewing the data

I next take a look at the data and view the column names as well as the data types in each column. I will only show the outputs of the head() funciton for the sake of the length of the document, but I have used the following functions: `str()`, `head()`, `colnames`)

```r
head(fitness_df)
```

```
## # A tibble: 6 x 16
##           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 11 more variables: LoggedActivitiesDistance <dbl>,
## #   VeryActiveDistance <dbl>, ModeratelyActiveDistance <dbl>,
## #   LightActiveDistance <dbl>, SedentaryActiveDistance <dbl>,
## #   VeryActiveMinutes <dbl>, FairlyActiveMinutes <dbl>,
## #   LightlyActiveMinutes <dbl>, SedentaryMinutes <dbl>, Calories <dbl>,
## #   Weekday <chr>
```

Next I would like to take a look at the highest and lowest calories burnt.

```r
arrange(fitness_df, desc(Calories))
```

```
## # A tibble: 940 x 16
##            Id ActivityDate TotalSteps TotalDistance TrackerDistance
##         <dbl> <chr>             <dbl>         <dbl>           <dbl>
##  1 6117666160 4/21/2016         19542          15.0            15.0
```

```
##  2 5577150313 4/17/2016         12231          9.14            9.14
##  3 8877689391 4/16/2016         29326         25.3            25.3
##  4 5577150313 5/1/2016          13368          9.99            9.99
##  5 5577150313 4/30/2016         12363          9.24            9.24
##  6 8877689391 4/30/2016         27745         26.7            26.7
##  7 5577150313 4/24/2016         15764         11.8            11.8
##  8 5577150313 4/16/2016         14269         10.7            10.7
##  9 8378563200 4/21/2016         15148         12.0            12.0
## 10 8378563200 4/14/2016         13318         10.6            10.6
## # i 930 more rows
## # i 11 more variables: LoggedActivitiesDistance <dbl>,
## #   VeryActiveDistance <dbl>, ModeratelyActiveDistance <dbl>,
## #   LightActiveDistance <dbl>, SedentaryActiveDistance <dbl>,
## #   VeryActiveMinutes <dbl>, FairlyActiveMinutes <dbl>,
## #   LightlyActiveMinutes <dbl>, SedentaryMinutes <dbl>, Calories <dbl>,
## #   Weekday <chr>
```

Interesting, that's a lot of calories burnt in a day. Lets take a look at the highest daily calorie burners.

```
calorie_summary <-
  fitness_df %>%
  group_by(Id) %>%
  summarise(avg_daily_calories_burned=mean(Calories))
head(arrange(calorie_summary, desc(avg_daily_calories_burned)))
```

```
## # A tibble: 6 x 2
##           Id avg_daily_calories_burned
##        <dbl>                     <dbl>
## 1 8378563200                     3437.
## 2 8877689391                     3420.
## 3 5577150313                     3360.
## 4 4388161847                     3094.
## 5 4702921684                     2966.
## 6 8053475328                     2946.
```

Now that we have an idea of the data, we can look to see if any activity type correlates more closely than others to calories burnt.

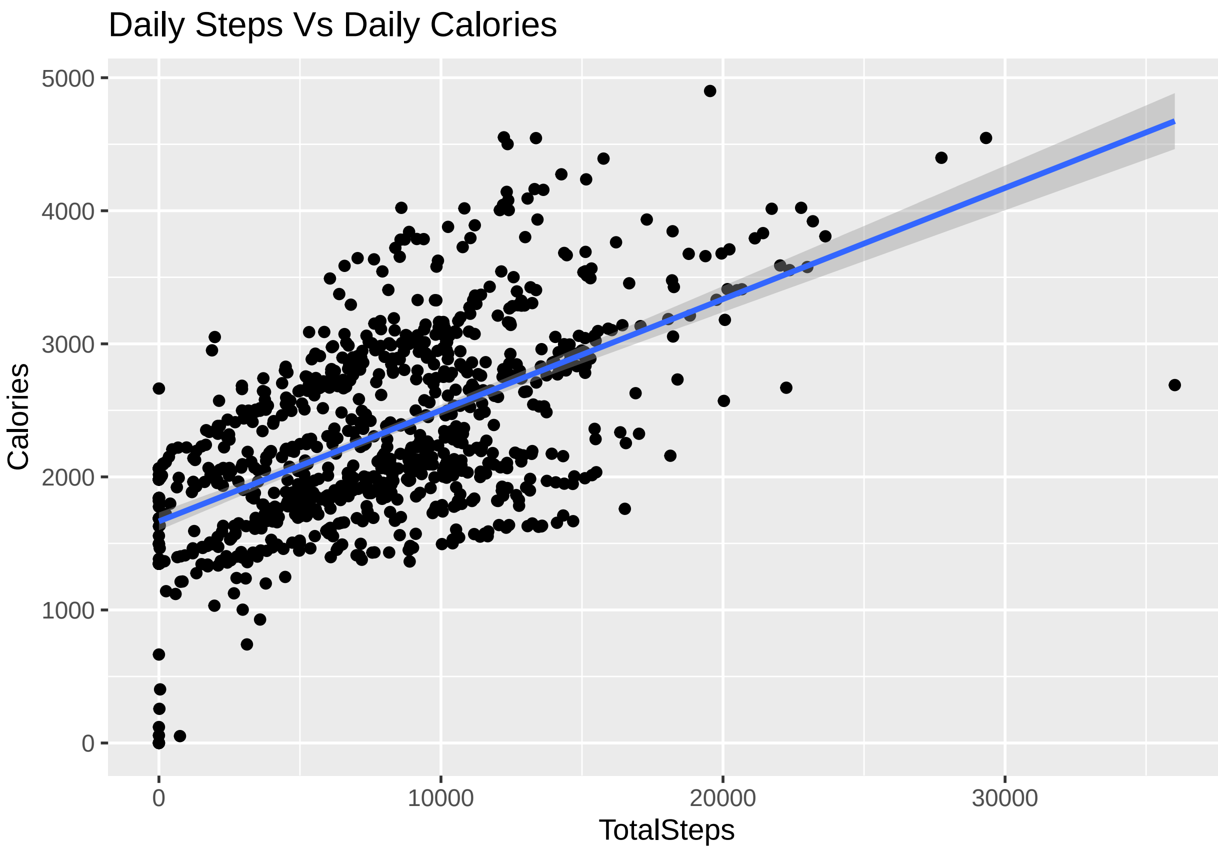
Analysis 1: Finding Correlations Between Activity Types

Next I will find which activity types correlate to most calories burnt.

Lets first take a look at Steps vs Calories

```
steps_v_calories_plot <- ggplot(fitness_df, aes(x=TotalSteps, y=Calories)) + geom_point() + labs(title=
steps_v_calories_plot + geom_smooth(method = lm)
```

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



Lets check the correlation using the Pearson Correlation test.

```
res <- cor.test(fitness_df$TotalSteps, fitness_df$Calories, method = "pearson")
res$estimate
```

```
##       cor
## 0.5915681
```

Now lets check the other activity types. Then We'll plot the ones with the strongest correlation values.

```
# very active minutes
very_active_min_correlation <- cor.test(fitness_df$VeryActiveMinutes, fitness_df$Calories, method = "pea
print(paste0("very active minutes:  ", very_active_min_correlation$estimate))
```

```
## [1] "very active minutes:  0.615838268270338"
```

```
#fairly active minutes
fairly_active_min_correlation <- cor.test(fitness_df$FairlyActiveMinutes, fitness_df$Calories, method =
print(paste0("fairly active minutes: ", fairly_active_min_correlation$estimate))
```

```
## [1] "fairly active minutes: 0.297623468265122"
```

```
#lightly active minutes
lightly_active_min_correlation <- cor.test(fitness_df$LightlyActiveMinutes, fitness_df$Calories, method
print(paste0("lightly active minutes: ", lightly_active_min_correlation$estimate))
```

```
## [1] "lightly active minutes: 0.286717534017549"
```

```
#total distance
total_distance_correlation<- cor.test(fitness_df$TotalDistance, fitness_df$Calories, method = "pearson")
print(paste0("total distance: ", total_distance_correlation$estimate))
```

```
## [1] "total distance: 0.644961872790222"
```

```r
#very active distance
very_active_distance_correlation<- cor.test(fitness_df$VeryActiveDistance, fitness_df$Calories, method =
print(paste0("very active distance: ", very_active_distance_correlation$estimate))
```

```
## [1] "very active distance: 0.491958565066386"
```

```r
#moderately active distance
moderately_active_distance_correlation<- cor.test(fitness_df$ModeratelyActiveDistance, fitness_df$Calori
print(paste0("moderately active distance: ", moderately_active_distance_correlation$estimate))
```

```
## [1] "moderately active distance: 0.216789870324992"
```

```r
#lightly active distance
lightly_active_distance_correlation<- cor.test(fitness_df$LightActiveDistance, fitness_df$Calories, met
print(paste0("lightly active distance: ", lightly_active_distance_correlation$estimate))
```

```
## [1] "lightly active distance: 0.466916760945079"
```

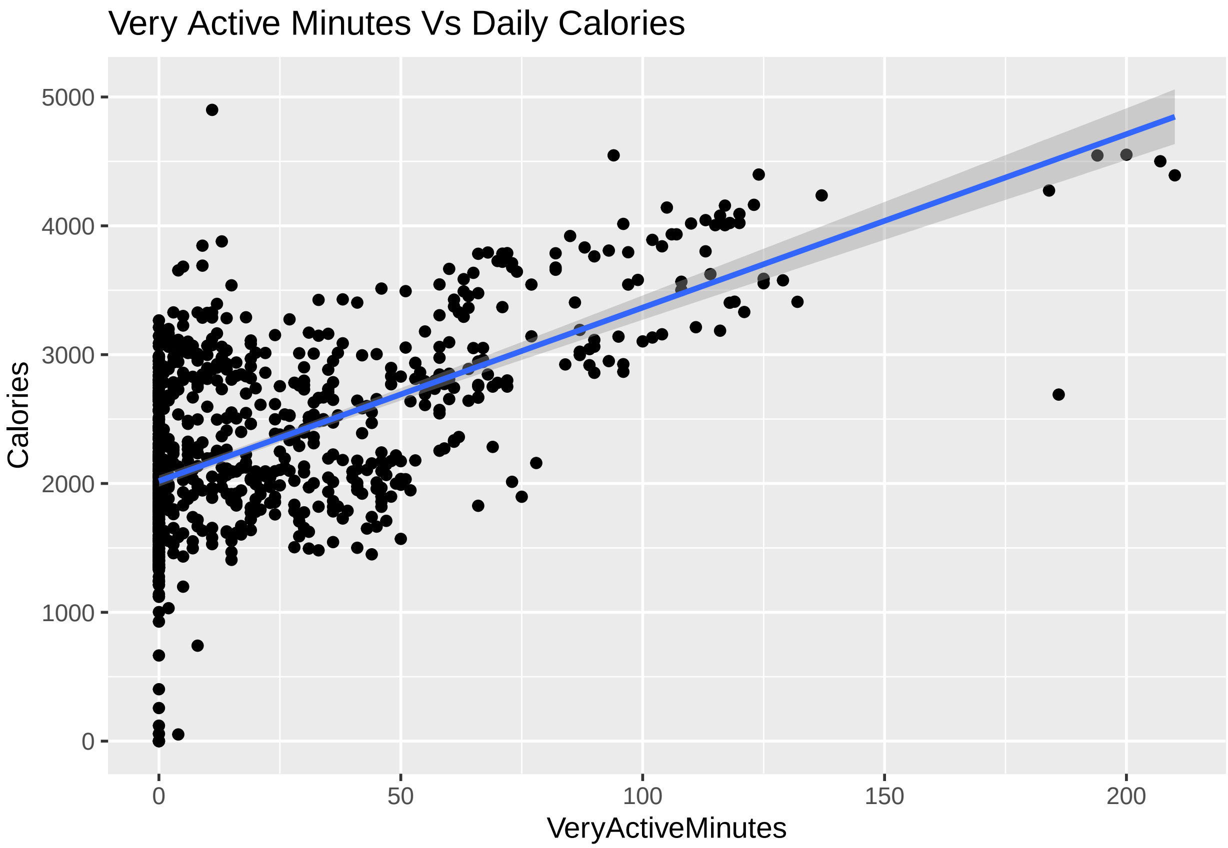
So From this analysis we can see that the strongest correlations are from the values total distance, very active minutes, and total steps.

Since total distance and total steps accounts for all distances and steps of each effort level, we can see that the type of activity that burns the most calories is very active activity.

Lets take a look at the plot.

```r
very_active_min_v_calories_plot <- ggplot(fitness_df, aes(x=VeryActiveMinutes, y=Calories)) + geom_poin
very_active_min_v_calories_plot + geom_smooth(method = lm)
```

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



Reccomendation results from Analysis 1 We can see that Very Active Activity correlates with more calories burned. BellaBeats should let this data be known to its customers and encourage them to participate in very active activities if they are looking to burn more calories.

Analysis Part 2: What days of the week are users most active?

While I was cleaning the data using Google Sheets, I added the day of the week to the data set using the WEEKDAY() function. I knew it would come in handy during this analysis.

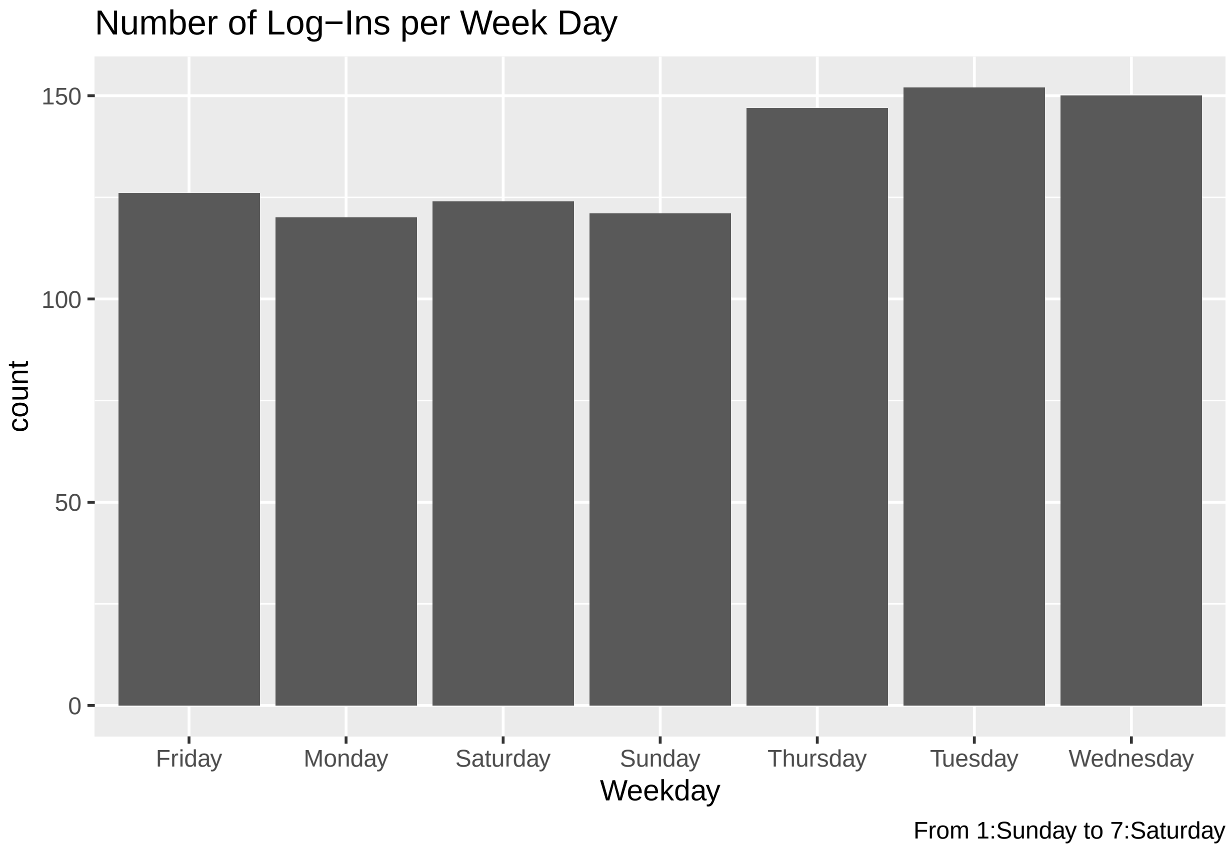
The objective is to figure out what days the the week users are most active. Lets first look at the instances of user log ons per weekday.

```
log_ons_per_day <- count(fitness_df, Weekday)
arrange(log_ons_per_day, n())
```

```
## # A tibble: 7 x 2
##   Weekday       n
##   <chr>     <int>
## 1 Friday      126
## 2 Monday      120
## 3 Saturday    124
## 4 Sunday      121
## 5 Thursday    147
## 6 Tuesday     152
## 7 Wednesday   150
```

Lets visualize the data.

```
ggplot(fitness_df) + geom_bar(mapping=aes(x=Weekday)) + labs(title="Number of Log-Ins per Week Day", ca
```



Seems that Tuesday, Wednesday and Thursday have the highest numbers of log ins. However this might not reveal the whole story.

Next I'll check each weekday to the average of each activity type.

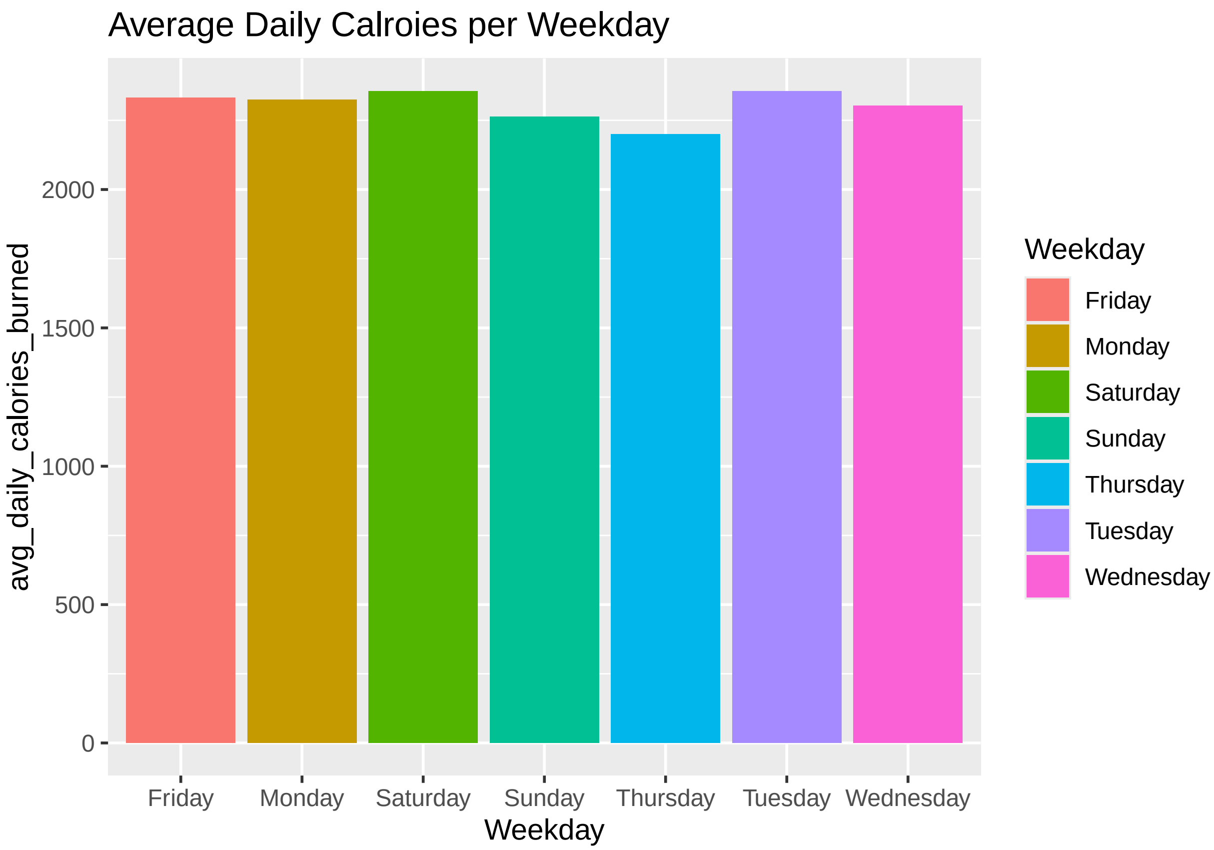
```
weekday_activity_level <-
  fitness_df %>%
  group_by(Weekday) %>%
  summarise(avg_daily_calories_burned=mean(Calories),avg_daily_total_distance=mean(TotalDistance),avg_da

weekday_activity_level
```

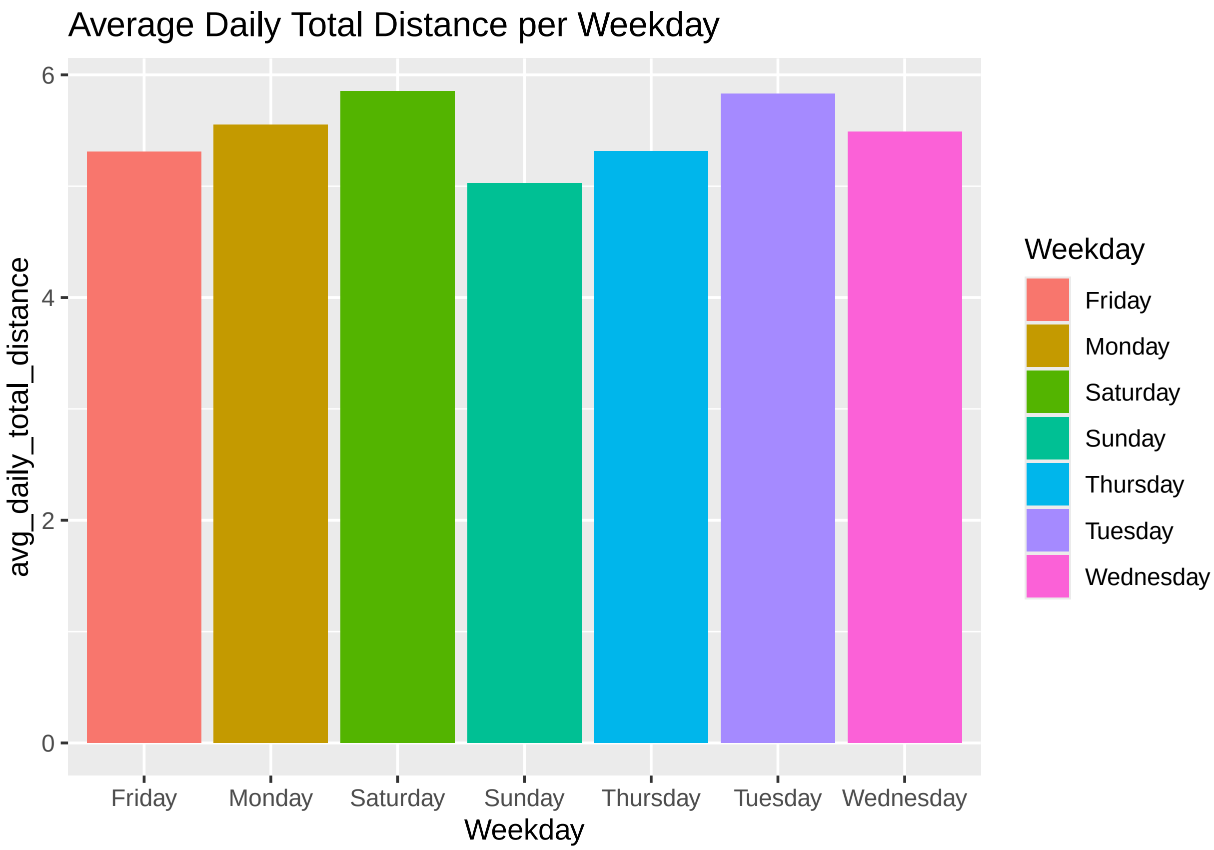
```
## # A tibble: 7 x 6
##   Weekday   avg_daily_calories_bu~1 avg_daily_total_dist~2 avg_daily_total_steps
##   <chr>                       <dbl>                  <dbl>                 <dbl>
## 1 Friday                      2332.                   5.31                 7448.
## 2 Monday                      2324.                   5.55                 7781.
## 3 Saturday                    2355.                   5.85                 8153.
## 4 Sunday                      2263                    5.03                 6933.
## 5 Thursday                    2200.                   5.31                 7406.
## 6 Tuesday                     2356.                   5.83                 8125.
## 7 Wednesday                   2303.                   5.49                 7559.
## # i abbreviated names: 1: avg_daily_calories_burned,
## #   2: avg_daily_total_distance
## # i 2 more variables: avg_daily_very_active_min <dbl>,
## #   avg_daily_fairly_active_min <dbl>
```

Above we can see the average activity level for each day of the week. Lets visualize the data.

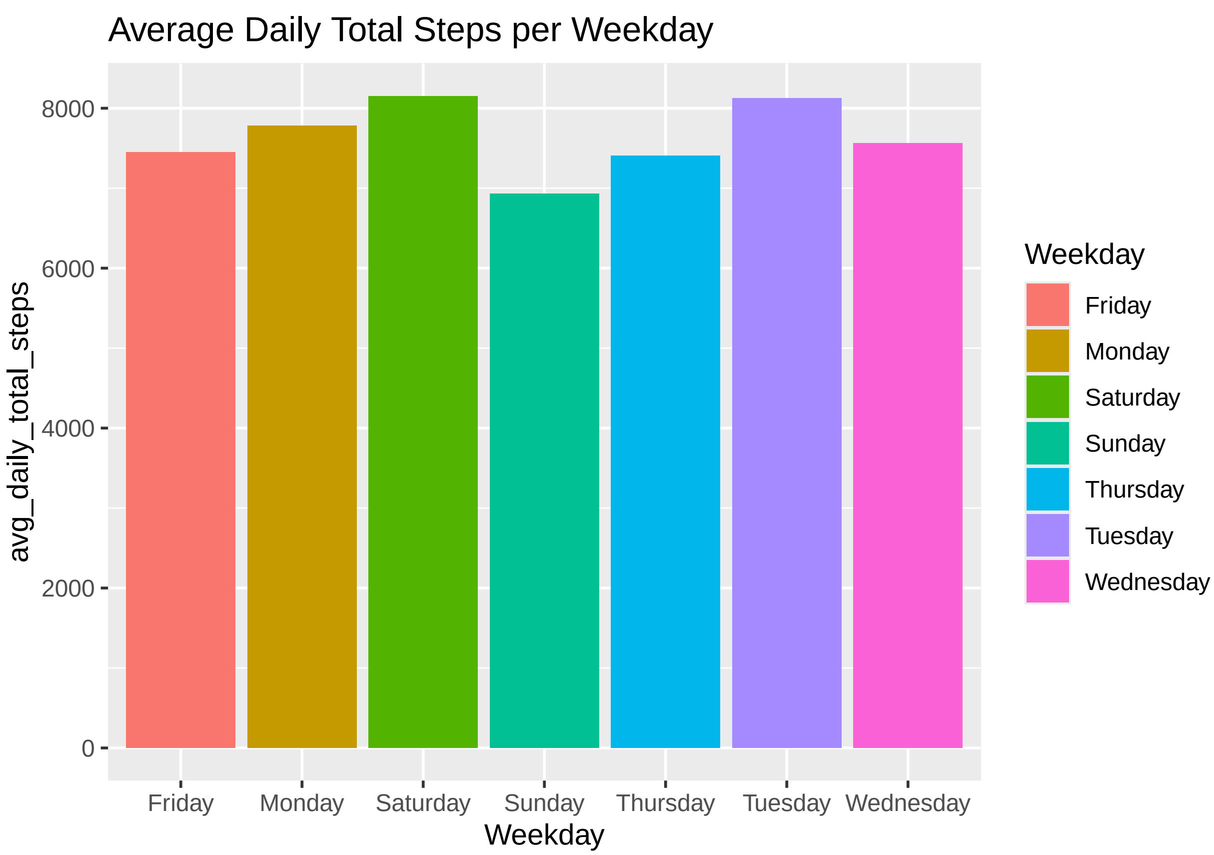
```
ggplot(weekday_activity_level) + geom_col(mapping=aes(x=Weekday, y=avg_daily_calories_burned, fill=Weekd
```



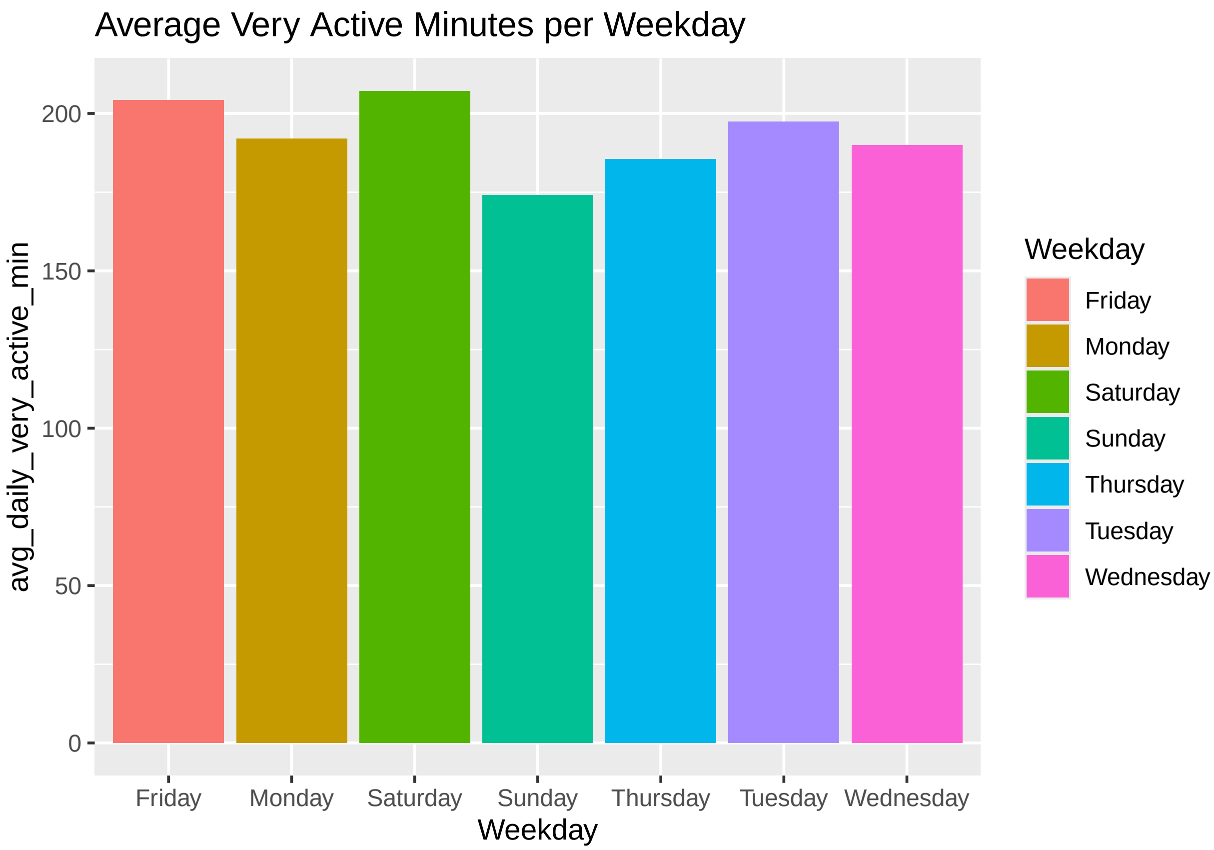

```
ggplot(weekday_activity_level) + geom_col(mapping=aes(x=Weekday, y=avg_daily_total_distance, fill=Weekda
```



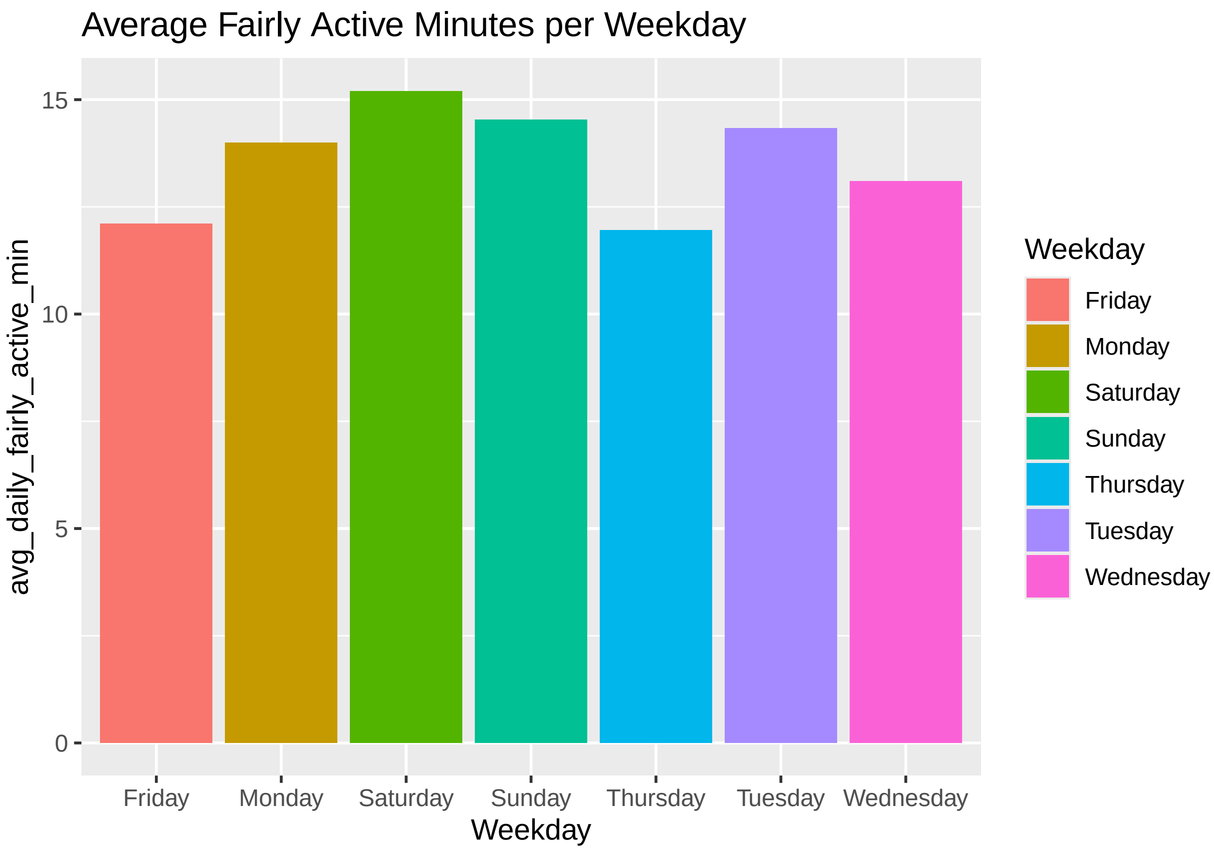

```
ggplot(weekday_activity_level) + geom_col(mapping=aes(x=Weekday, y=avg_daily_total_steps,fill=Weekday))
```




```
ggplot(weekday_activity_level) + geom_col(mapping=aes(x=Weekday, y=avg_daily_very_active_min,fill=Weekda
```




```
ggplot(weekday_activity_level) + geom_col(mapping=aes(x=Weekday, y=avg_daily_fairly_active_min,fill=Weel
```



From the visuals we can see that Saturday and Tuesday are consistently at the top for each category.

Reccomendation results from Analysis 2 It would be safe to assume that Saturday and Tuesday are the top days and the marketing team should focus there campaigns more heavily on Saturday and Tuesdays. However, while these days see more activity level, they do not exceed the other days by a large amount. In general Sunday and Thursday are usually the least active days. Therefore less marketing can be done on those days.

Analysis Part 3: Hours with the highest activity levels

Here I will do a similar task to what I've done above. This time looking at which hours of the day are most active.

First lets explore our data sets for hourly data.

```r
str(hourly_calories)
```

```
## spc_tbl_ [22,099 x 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Id          : num [1:22099] 1.5e+09 1.5e+09 1.5e+09 1.5e+09 1.5e+09 ...
##  $ ActivityHour: chr [1:22099] "4/12/16 0:00" "4/12/16 1:00" "4/12/16 2:00" "4/12/16 3:00" ...
##  $ Calories    : num [1:22099] 81 61 59 47 48 48 48 47 68 141 ...
##  $ Date        : chr [1:22099] "4/12/16" "4/12/16" "4/12/16" "4/12/16" ...
##  $ Hour        : 'hms' num [1:22099] 00:00:00 01:00:00 02:00:00 03:00:00 ...
##   ..- attr(*, "units")= chr "secs"
##  $ HourNumber  : num [1:22099] 0 1 2 3 4 5 6 7 8 9 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Id = col_double(),
##   ..   ActivityHour = col_character(),
```

```
##   ..   Calories = col_double(),
##   ..   Date = col_character(),
##   ..   Hour = col_time(format = ""),
##   ..   HourNumber = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

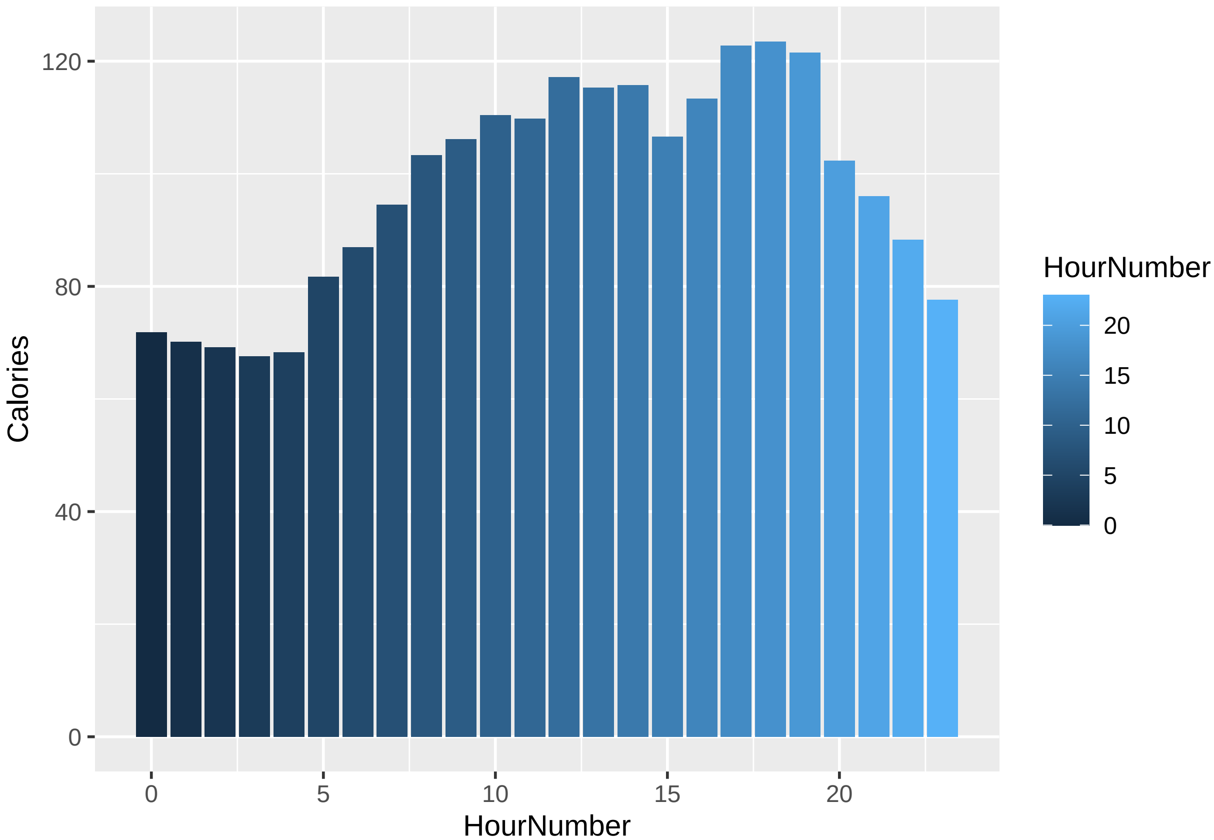
Next, I'll see how many calories on average are burnned each hour.

```
avg_calories_per_hour <- aggregate(Calories ~ HourNumber, hourly_calories, mean)
avg_calories_per_hour
```

```
##    HourNumber  Calories
## 1           0  71.80514
## 2           1  70.16506
## 3           2  69.18650
## 4           3  67.53805
## 5           4  68.26180
## 6           5  81.70815
## 7           6  86.99678
## 8           7  94.47798
## 9           8 103.33727
## 10          9 106.14286
## 11         10 110.46071
## 12         11 109.80690
## 13         12 117.19740
## 14         13 115.30945
## 15         14 115.73290
## 16         15 106.63716
## 17         16 113.32745
## 18         17 122.75276
## 19         18 123.49227
## 20         19 121.48455
## 21         20 102.35762
## 22         21  96.05635
## 23         22  88.26549
## 24         23  77.59358
```

Let's visualize the data.

```
ggplot(avg_calories_per_hour) + geom_col(mapping=aes(x=HourNumber, y=Calories,fill=HourNumber))
```



Here we can see that the most calories are being burnt 17th to 19th hour. in general there is the range 10-19 produces on average a higher amount of calories being burnt, with a significant low at the 15th hour.

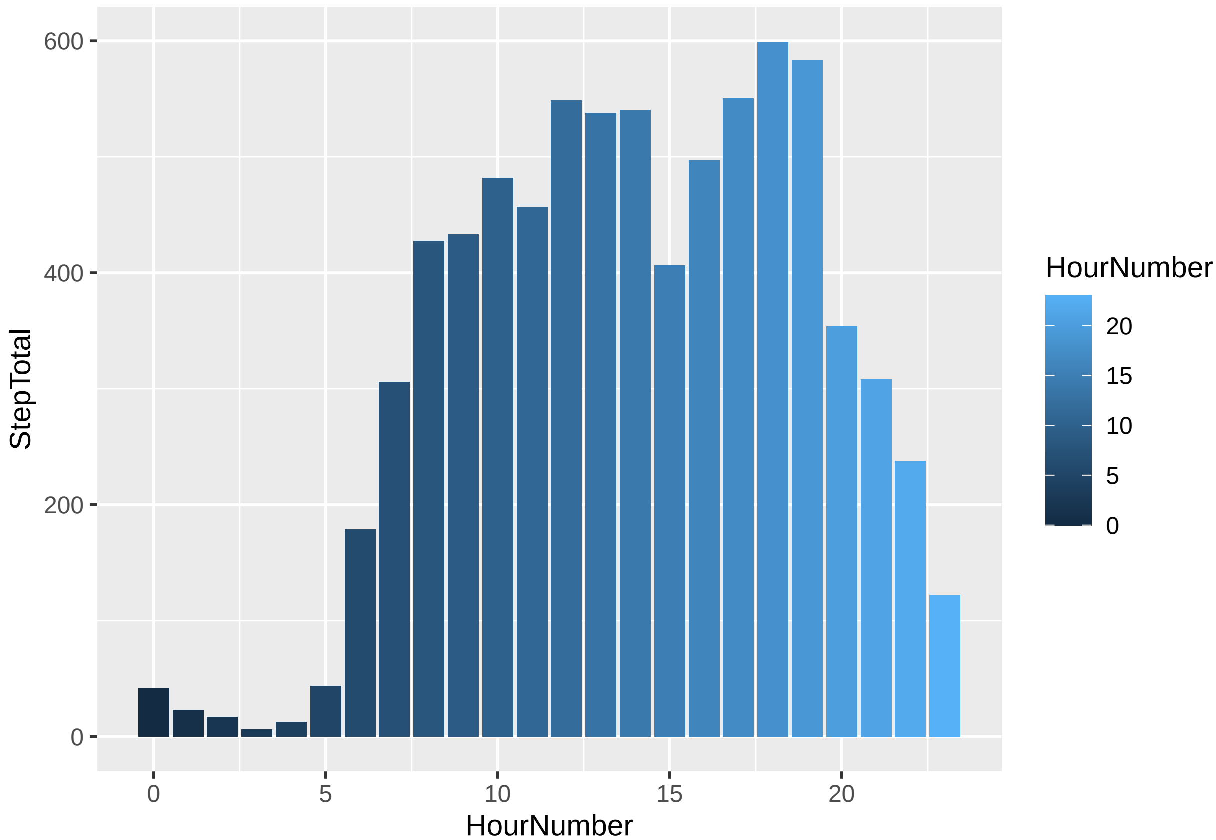
Lets analyze the rest of the activity data ina similar way.

```
avg_steps_per_hour <- aggregate(StepTotal~ HourNumber, hourly_steps, mean)
avg_steps_per_hour
```

```
##    HourNumber  StepTotal
## 1           0  42.188437
## 2           1  23.102894
## 3           2  17.110397
## 4           3   6.426581
## 5           4  12.699571
## 6           5  43.869099
## 7           6 178.508056
## 8           7 306.049409
## 9           8 427.544576
## 10          9 433.301826
## 11         10 481.665231
## 12         11 456.886731
## 13         12 548.642082
## 14         13 537.698154
## 15         14 540.513572
## 16         15 406.319126
## 17         16 496.845645
## 18         17 550.232892
## 19         18 599.169978
## 20         19 583.390728
```

```
## 21          20 353.905077
## 22          21 308.138122
## 23          22 237.987832
## 24          23 122.132890
```

```r
ggplot(avg_steps_per_hour) + geom_col(mapping=aes(x=HourNumber, y=StepTotal,fill=HourNumber))
```

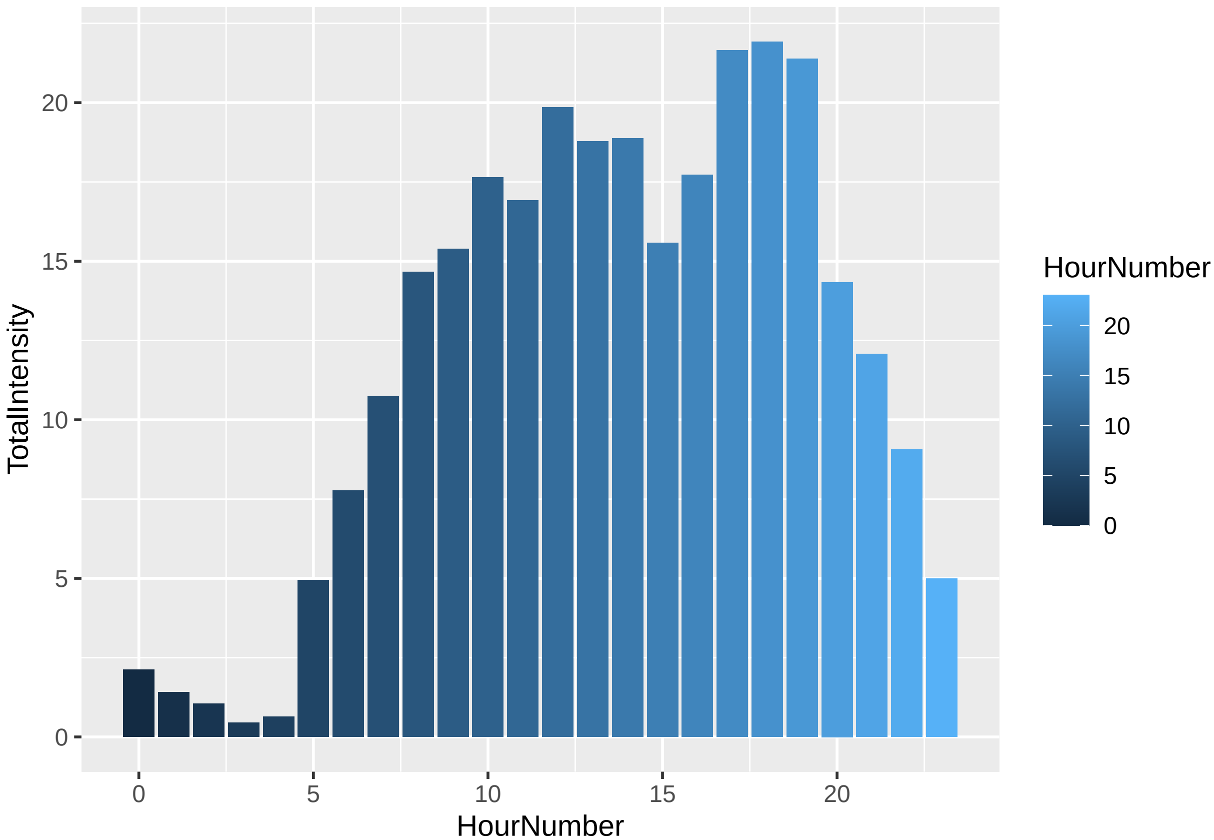



```r
avg_intensity_per_hour <- aggregate(TotalIntensity~ HourNumber, hourly_intensity, mean)
avg_intensity_per_hour
```

```
##    HourNumber TotalIntensity
## 1           0      2.1295503
## 2           1      1.4190782
## 3           2      1.0439443
## 4           3      0.4437299
## 5           4      0.6330472
## 6           5      4.9506438
## 7           6      7.7712137
## 8           7     10.7336198
## 9           8     14.6680988
## 10          9     15.3877551
## 11         10     17.6437029
## 12         11     16.9212513
## 13         12     19.8470716
## 14         13     18.7752443
## 15         14     18.8686211
## 16         15     15.5846995
## 17         16     17.7166483
```

```
## 18          17      21.6556291
## 19          18      21.9216336
## 20          19      21.3852097
## 21          20      14.3399558
## 22          21      12.0729282
## 23          22       9.0630531
## 24          23       4.9966777
```

```r
ggplot(avg_intensity_per_hour) + geom_col(mapping=aes(x=HourNumber, y=TotalIntensity,fill=HourNumber))
```



In all categories we see the highest activity level in the 17th - 19th (5:00PM - 7:00Pm) hours of the day. In a wider range we see the activity level 12th - 19th hour (12:00PM - 7:00PM) with a low at the 15th and 16th (3:00PM and 4:00PM) hours.

Reccomendation results from Analysis 3 The recommendation would be to focus marketing during the 17th - 19th hour or (5:00PM - 7:00Pm) and 12th - 14th hour (12:00PM - 2:00PM).