FitBit_Fitness_tracker Notebook

Massiel Raya

7/15/2023

FitBit Tracker Project dataset found on kaggle by user MÖBIUS

***Purpose*:**

The purpose of this project is to determine user trends of the BellaBeat dataset. We will look closely at a subset of data that contains the hourly calorie output, intensity & steps of 33 users. This will help identify trends & possible areas of improvement in functionality.

***Problems*:**

One of the problems in the dataset is that Intensity was not defined in the Kaggle page. For the purpose of this Analysis, it was inferred that Intensity is measured using heartrate as a way to indicate how strenuous an activity was.

Another problem was that in the orginal dataset, the column name "Calories" was not descriptive. It was assumed that this column was indicating the Calories burned or the "Caloric Output" as referenced in this analysis.

It was apparent that BellaBeat Dataset was large & therefore, required powerful tools such as RStudio to handle the dataset. GoogleSheets was not a useful tool for such a large dataset. Additionally upon inspection, it was found that data was incomplete particularly sleepDay_merged & weightLogInfo_merged sheets. So I decided to focus solely on the hourly datasets that were complete.

Other problems were minor & could be remedied with data cleaning/prepping such as ensuring proper datatypes & correcting data formats.

Loading the datasets

```
hourlyIntensities_merged <- read.csv("C:/Users/massi/OneDrive/Desktop/class/proyectos - datasets/Fitaba

hourlyCalories_merged <- read.csv("C:/Users/massi/OneDrive/Desktop/class/proyectos - datasets/Fitabase

hourlySteps_merged <- read.csv("C:/Users/massi/OneDrive/Desktop/class/proyectos - datasets/Fitabase Data
```

```r
# install packages
library(ggplot2)
```

I will only be focusing on the subset of data that monitors Hourly Calories, Intensities, & Steps.

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

```r
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 4.2.3

##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.2.3

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.2.3

## Warning: package 'tibble' was built under R version 4.2.3

## Warning: package 'tidyr' was built under R version 4.2.3

## Warning: package 'readr' was built under R version 4.2.3

## Warning: package 'purrr' was built under R version 4.2.3

## Warning: package 'stringr' was built under R version 4.2.3
```

```
## Warning: package 'forcats' was built under R version 4.2.3

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats 1.0.0     v stringr 1.5.0
## v purrr   1.0.1     v tibble  3.2.1
## v readr   2.1.4     v tidyr   1.3.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(tinytex)
```

Here we clean up the data we will be reformatting the date-time column

```
# reformat ActivityHour column into date-time format
hourlyCalories_merged$ActivityHour <- mdy_hms(hourlyCalories_merged$ActivityHour)

# reformat hourly calories merged
hourlyCalories_merged$Day <- day(hourlyCalories_merged$ActivityHour)
hourlyCalories_merged$Hour <- hour(hourlyCalories_merged$ActivityHour)
hourlyCalories_merged$DayOfWeek <- weekdays(hourlyCalories_merged$ActivityHour)

# reformat hourly Intensities merged
hourlyIntensities_merged$ActivityHour <- mdy_hms(hourlyIntensities_merged$ActivityHour)
hourlyIntensities_merged$Day <- day(hourlyIntensities_merged$ActivityHour)
hourlyIntensities_merged$Hour <- hour(hourlyIntensities_merged$ActivityHour)
hourlyIntensities_merged$DayOfWeek <- weekdays(hourlyIntensities_merged$ActivityHour)

# reformat hourly Steps merged
hourlySteps_merged$ActivityHour <- mdy_hms(hourlySteps_merged$ActivityHour)
hourlySteps_merged$Day <- day(hourlySteps_merged$ActivityHour)
hourlySteps_merged$Hour <- hour(hourlySteps_merged$ActivityHour)
hourlySteps_merged$DayOfWeek <- weekdays(hourlySteps_merged$ActivityHour)
```

To futher clean up the data, we will change datatype for ALL ID columns to be character types instead of number types.

```
hourlyCalories_merged$Id <- as.character(hourlyCalories_merged$Id)
hourlyIntensities_merged$Id <- as.character(hourlyIntensities_merged$Id)
hourlySteps_merged$Id <- as.character(hourlySteps_merged$Id)
```

Calculate the sum & average of caloric output for each customer ID

```


sum_caloric <- aggregate(Calories ~ Id, data = hourlyCalories_merged, FUN = sum)

av_caloric <- aggregate(Calories ~ Id, data = hourlyCalories_merged, FUN = mean)
```

note: we are assuming this dataset refers to caloric output (ie calories burned)

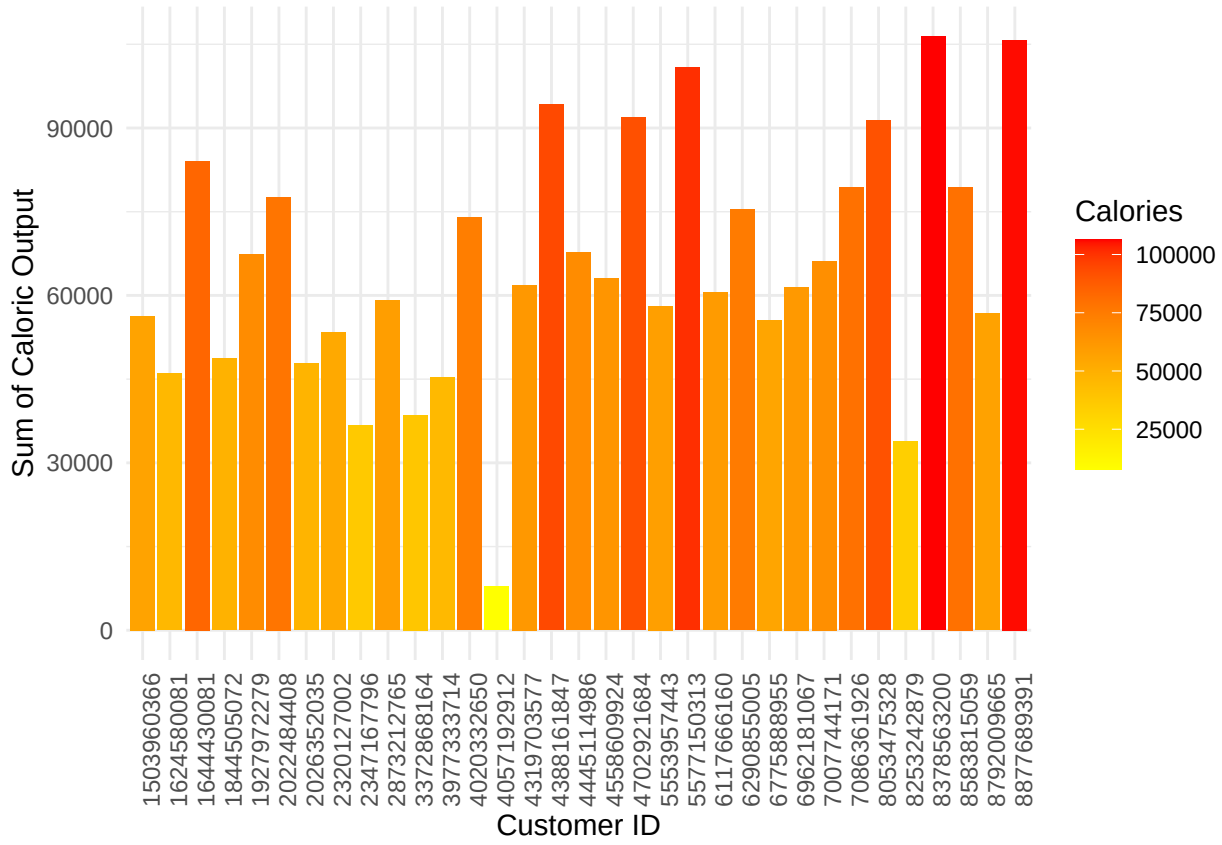
Find summary stats of calories

```
mean_cal <- mean(hourlyCalories_merged$Calories) #this is the mean "trend line" that we will use for th

#Calculate summary stats of hourly_Calories_merged
summary_table <- summary(hourlyCalories_merged$Calories)

#print table
print(summary_table)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   42.00   63.00   83.00   97.39  108.00  948.00
```

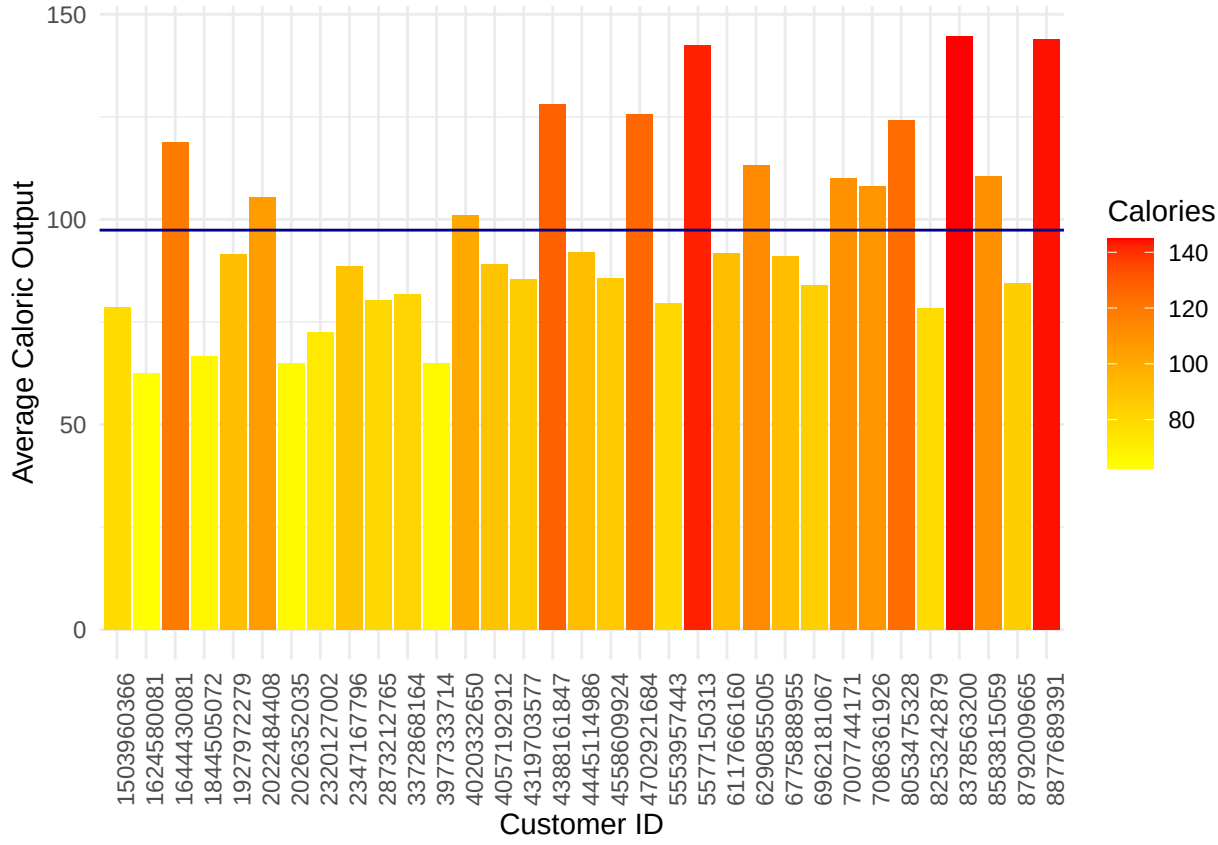
Create the bar plot that demonstrates the total Caloric Output of each User

```
ggplot(sum_caloric, aes(x = as.factor(Id), y = Calories, fill = Calories)) +
  geom_bar(stat = "identity") +
  scale_fill_gradient(low = "yellow", high = "red") +
  labs(x = "Customer ID", y = "Sum of Caloric Output") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90))
```



The following shows a barplot of the average caloric output for each User

```
ggplot(av_caloric, aes(x = as.factor(Id), y = Calories, fill = Calories)) +
  geom_bar(stat = "identity") +
  geom_hline(yintercept = mean_cal, color = "darkblue") +
  scale_fill_gradient(low = "yellow", high = "red") +
  labs(x = "Customer ID", y = "Average Caloric Output") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90))
```

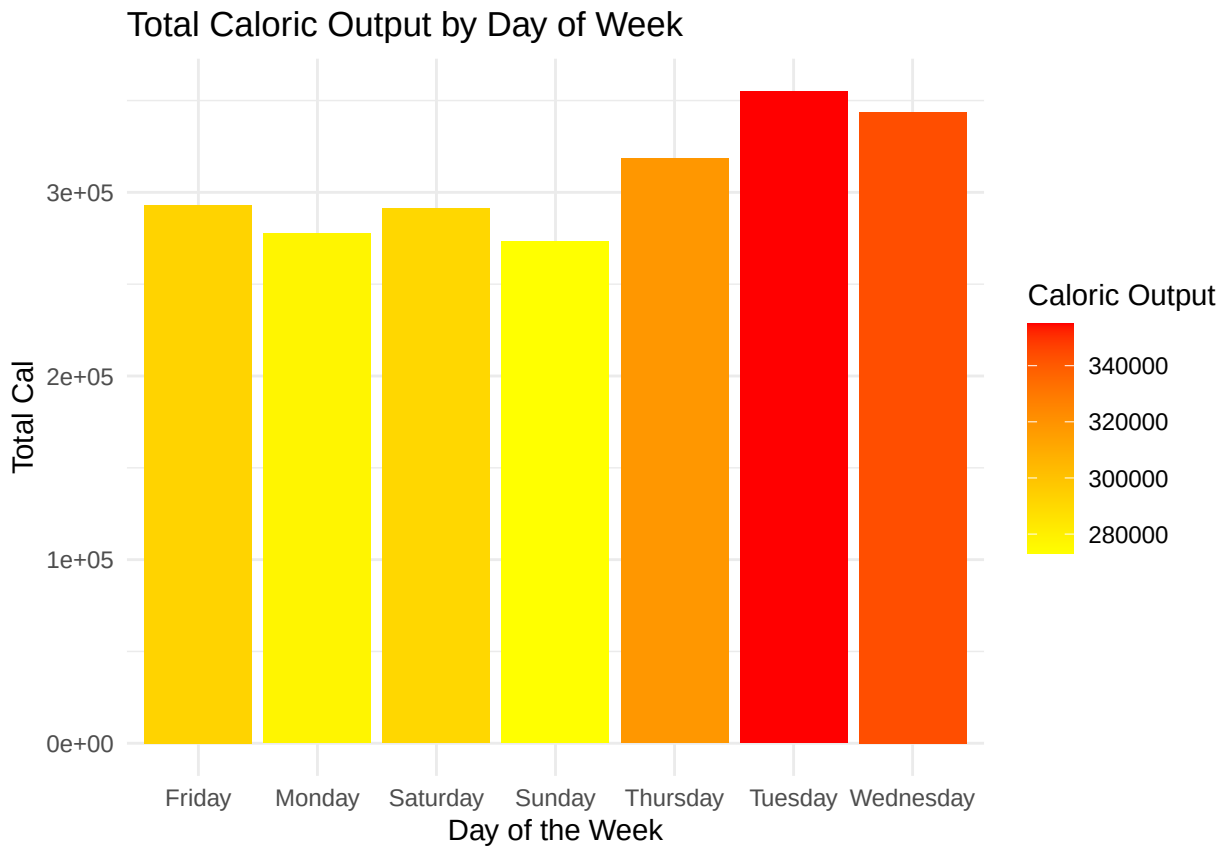


We see that the average Caloric output falls around 97.39.

Next, we find the Days of the week & Hours that had the most Caloric Output

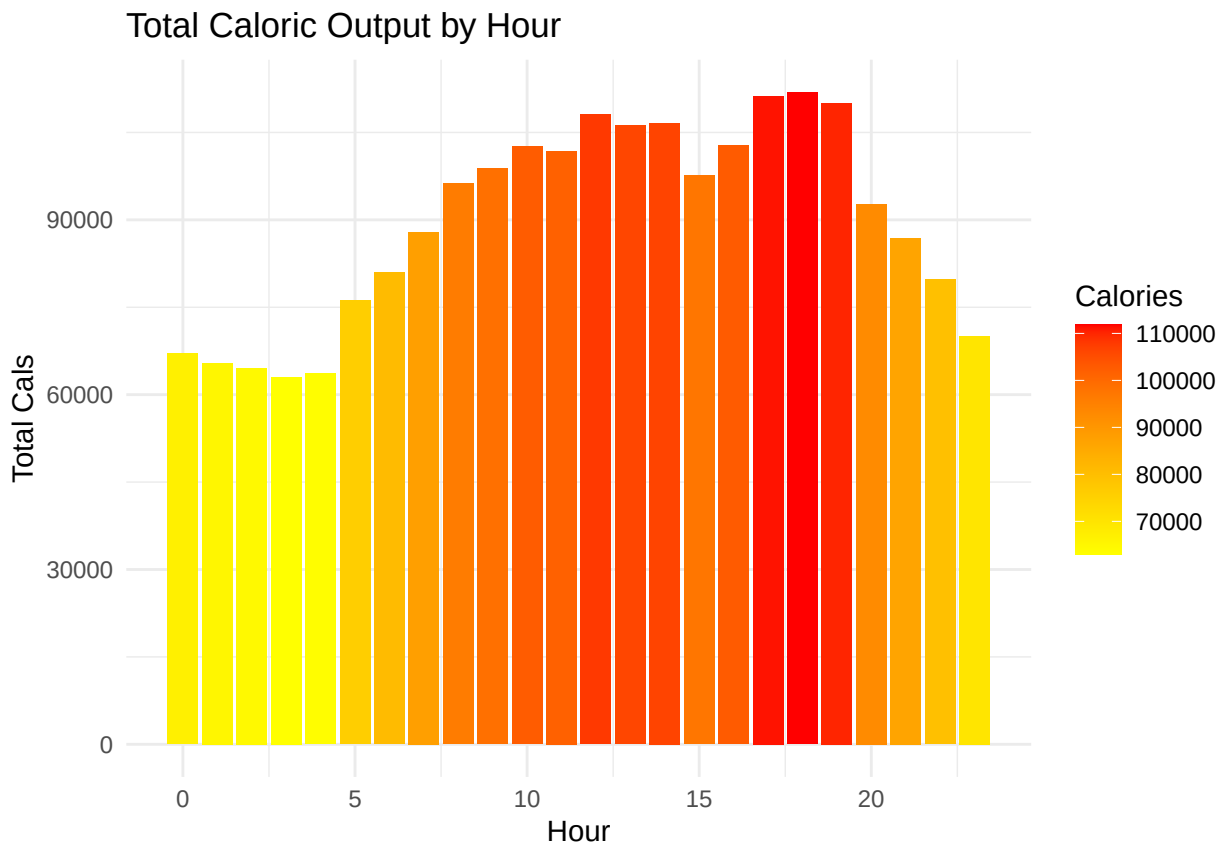
```
# Create the heatmap
# this shows the day of the week against the hour to determine highest caloric output
sum_caloric_DOW <- aggregate(Calories ~ DayOfWeek, data = hourlyCalories_merged, FUN = sum)

ggplot(sum_caloric_DOW, aes(x = DayOfWeek, y = Calories, fill = Calories)) +
  geom_bar(stat="identity") +
  labs(x = "Day of the Week", y = "Total Cal", fill = "Caloric Output") +
  scale_fill_gradient(low = "yellow", high = "red") +
```

```
  ggtitle("Total Caloric Output by Day of Week") +
  theme_minimal()
```



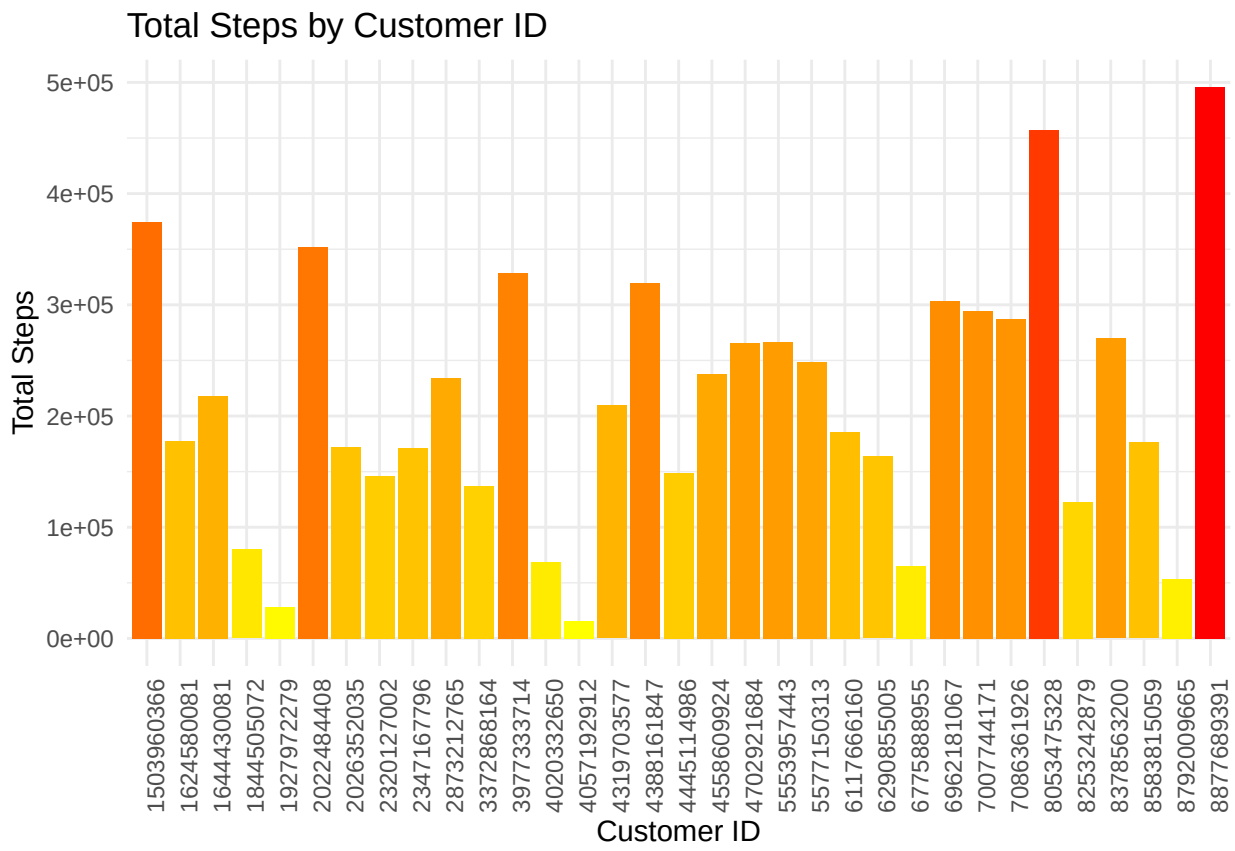
The goal now is to find the hours with the highest caloric output

```
#create table of total cals for each hours
sum_caloric_hr <- aggregate(Calories ~ Hour, data = hourlyCalories_merged, FUN = sum)

#lineplot of the caloric output of those hours
ggplot(sum_caloric_hr, aes(x = Hour, y = Calories, fill = Calories)) +
  geom_bar(stat = "identity") +
  scale_fill_gradient(low = "yellow", high = "red") +
  labs(x = "Hour", y = "Total Cals") +
  ggtitle("Total Caloric Output by Hour") +
  theme_minimal()
```



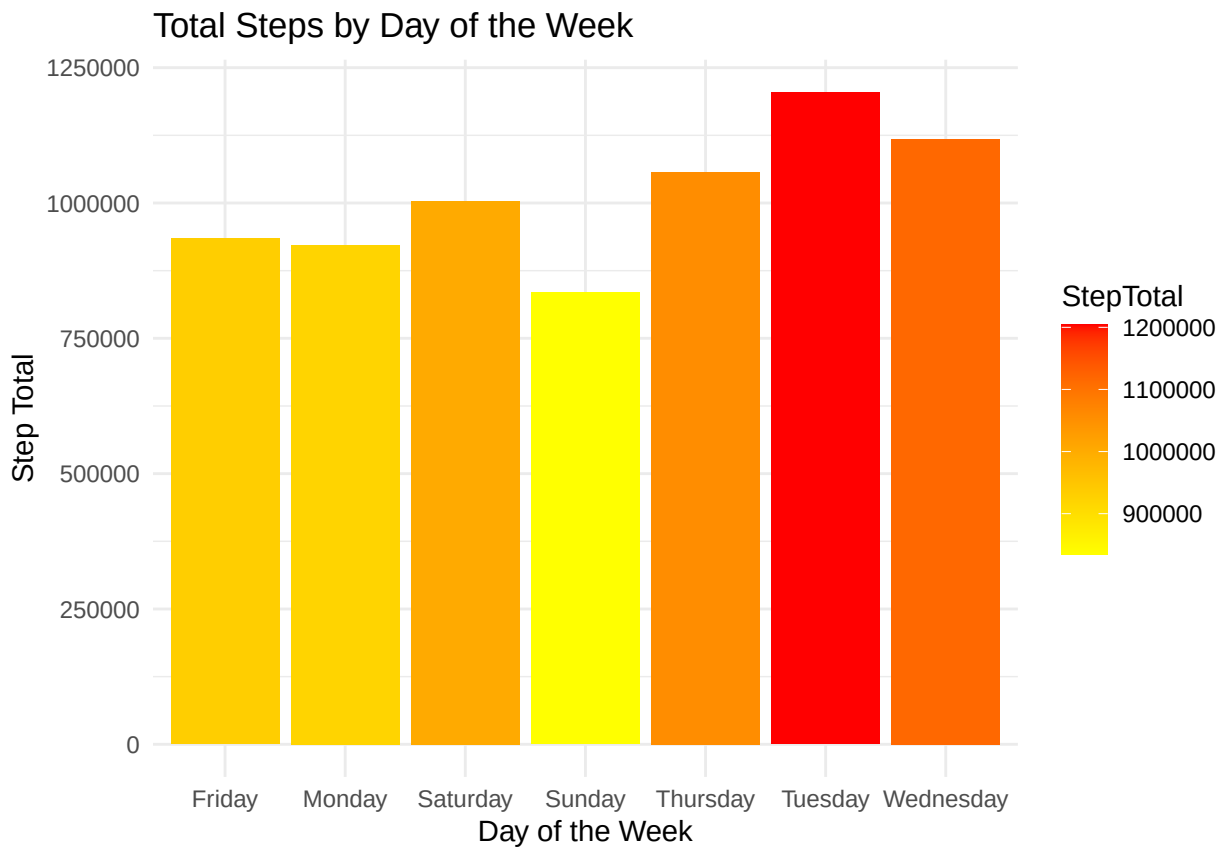
We follow the same process for the steps taken to find the users with the most/least steps

```
sum_steps <- aggregate(StepTotal ~ Id, data = hourlySteps_merged, FUN = sum)

# Convert the ID column to character data type
sum_steps$Id <- as.character(sum_steps$Id)

# Create the bar graph
ggplot(sum_steps, aes(x = Id, y = StepTotal, fill = StepTotal)) +
  geom_bar(stat = "identity") +
  scale_fill_gradient(low = "yellow", high = "red") +
  labs(x = "Customer ID", y = "Total Steps", fill = "Step Total") +
  ggtitle("Total Steps by Customer ID") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90),
        legend.position = "none")
```



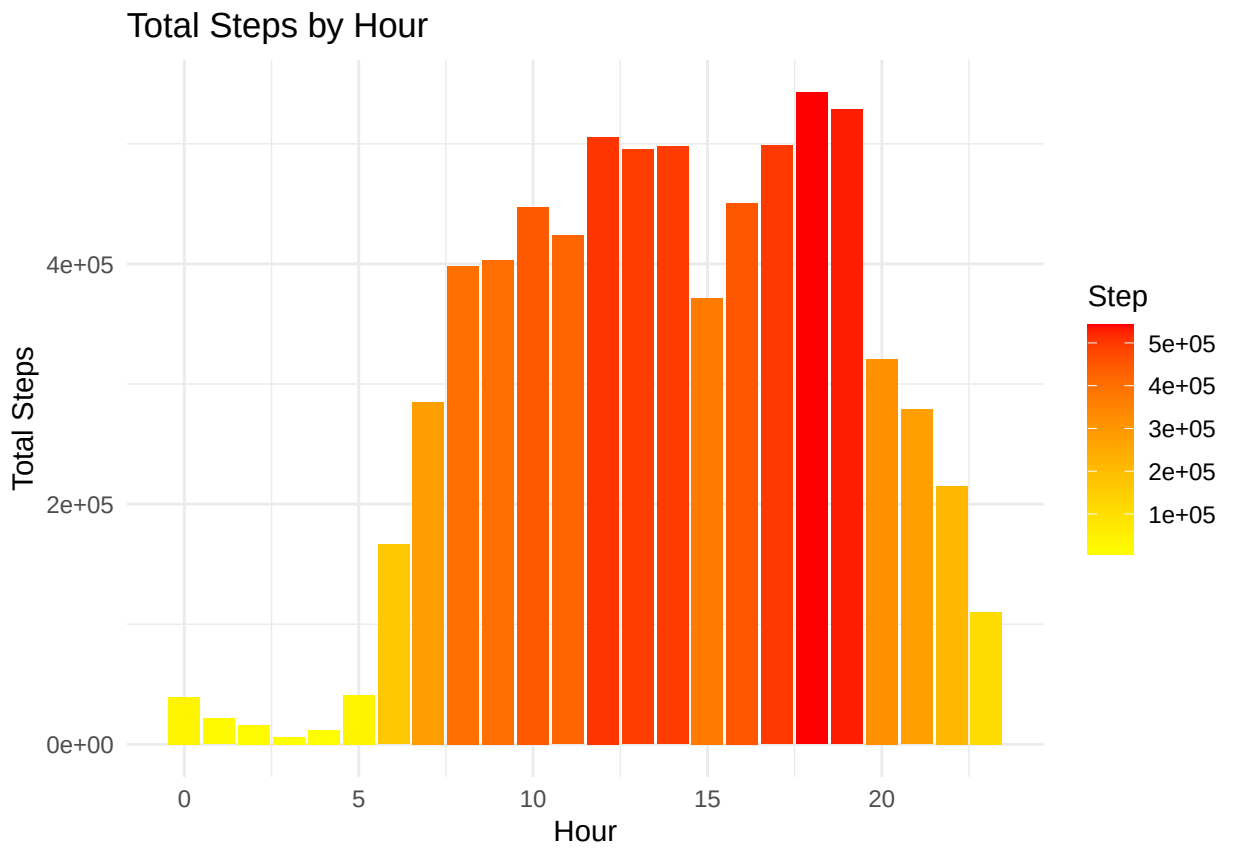
The next step find the days of week with the most steps taken

```
sum_step_day <- aggregate(StepTotal ~ DayOfWeek, data = hourlySteps_merged, FUN = sum)

# Create a bar plot with color based on total steps
ggplot(sum_step_day, aes(x = DayOfWeek, y = StepTotal, fill = StepTotal)) +
  geom_bar(stat = "identity") +
  scale_fill_gradient(low = "yellow", high = "red") +
  labs(x = "Day of the Week", y = "Step Total") +
  ggtitle("Total Steps by Day of the Week") +
  theme_minimal()
```



Again, we want to ensure that the hours with the most steps taken makes sense with the hours where most calories were burned. It was found that hours 17-19 had the most Caloric Output, so we want to see if steps taken & calorie output parallels the hours of highest step activity.

```
#create table of total steps taken for each hour
sum_step_hr <- aggregate(StepTotal ~ Hour, data = hourlySteps_merged, FUN = sum)

#line map of the total steps taken of those hours
ggplot(sum_step_hr, aes(x = Hour, y = StepTotal, fill = StepTotal)) +
  geom_bar(stat = "identity") +
  scale_fill_gradient(low = "yellow", high = "red") +
  labs(x = "Hour", y = "Total Steps", fill = "Step") +
  ggtitle("Total Steps by Hour") +
  theme_minimal()
```



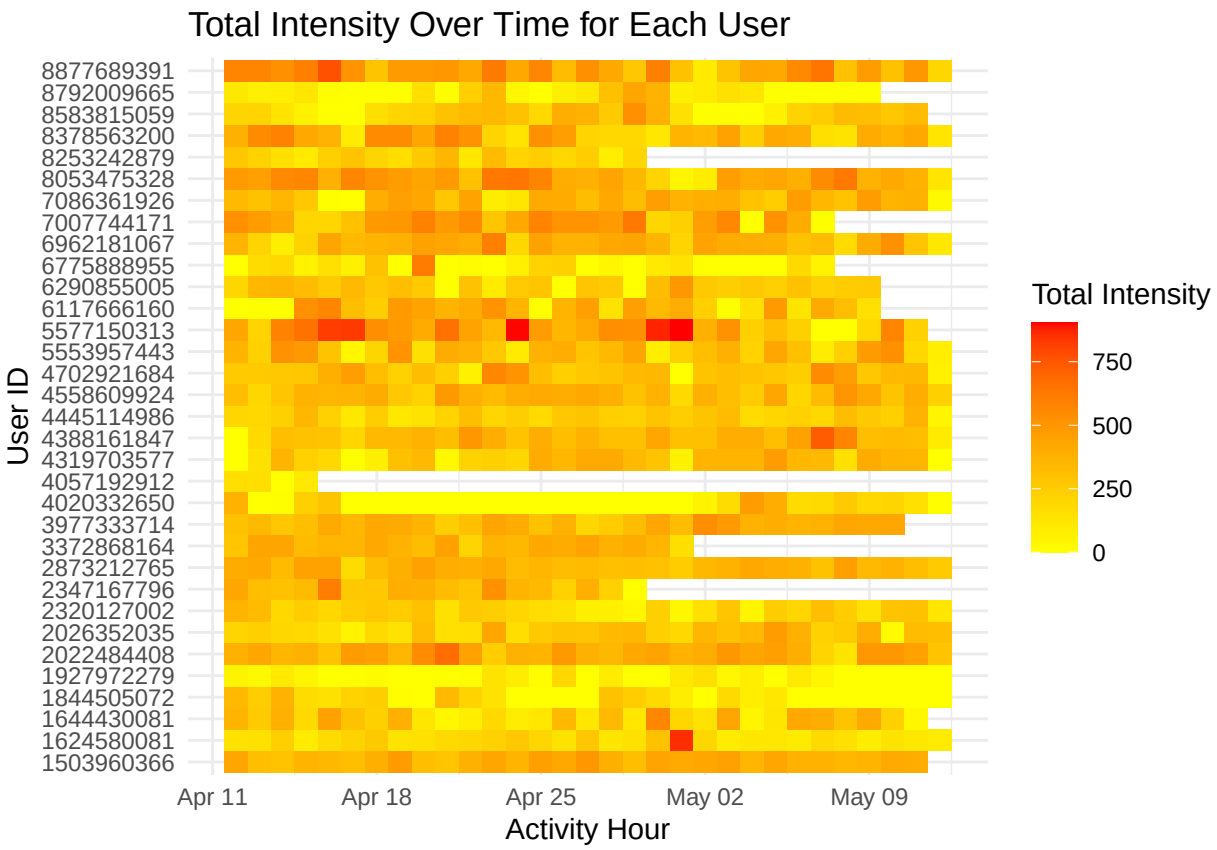
Now we look at hourly intensities. Intensities is believed to use heartrate to determine how strenous an activity is. The goal here is to find patterns with the hopes of offering users more personalized functions

For this plot, we show the summed Intensities across time of each User

```
#Total intensity of each user
Total_Int <- aggregate(TotalIntensity ~ Id, data = hourlyIntensities_merged, FUN = sum)

#Average intensity of each user & rename colname
Av_Tot_Int <- aggregate(TotalIntensity ~ Id, data = hourlyIntensities_merged, FUN = mean)
colnames(Av_Tot_Int)[colnames(Av_Tot_Int) == "TotalIntensity"] <- "AverageIntensity"

#ensure date format
hourlyIntensities_merged$ActivityHour <- as.Date(hourlyIntensities_merged$ActivityHour)

sum_intensity <- hourlyIntensities_merged %>%
  group_by(Id, ActivityHour) %>%
  summarise(TotalIntensity = sum(TotalIntensity))
```

```
## `summarise()` has grouped output by 'Id'. You can override using the `.groups`
## argument.
```

```r
# Create the bar graph
ggplot(sum_intensity, aes(x = ActivityHour, y = Id, fill = TotalIntensity)) +
  geom_tile() +
  labs(x = "Activity Hour", y = "User ID", fill = "Total Intensity") +
  ggtitle("Total Intensity Over Time for Each User") +
  scale_fill_gradient(low = "yellow", high = "red") +
  theme_minimal()
```



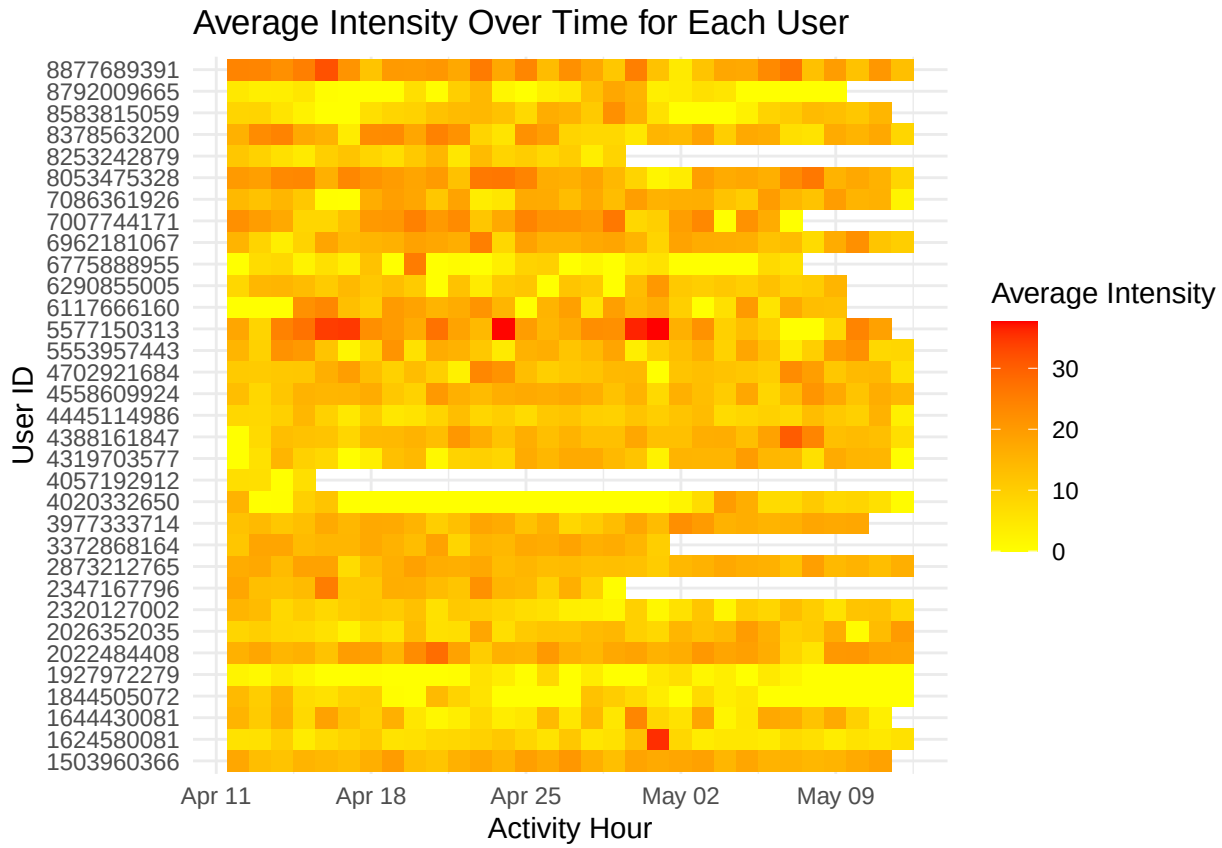
This plots the average intensity across time of each User

```r
#average intensity of each user
avg_intensity <- hourlyIntensities_merged %>%
  group_by(Id, ActivityHour) %>%
  summarise(AverageIntensity = mean(TotalIntensity))
```

```
## `summarise()` has grouped output by 'Id'. You can override using the `.groups`
## argument.
```

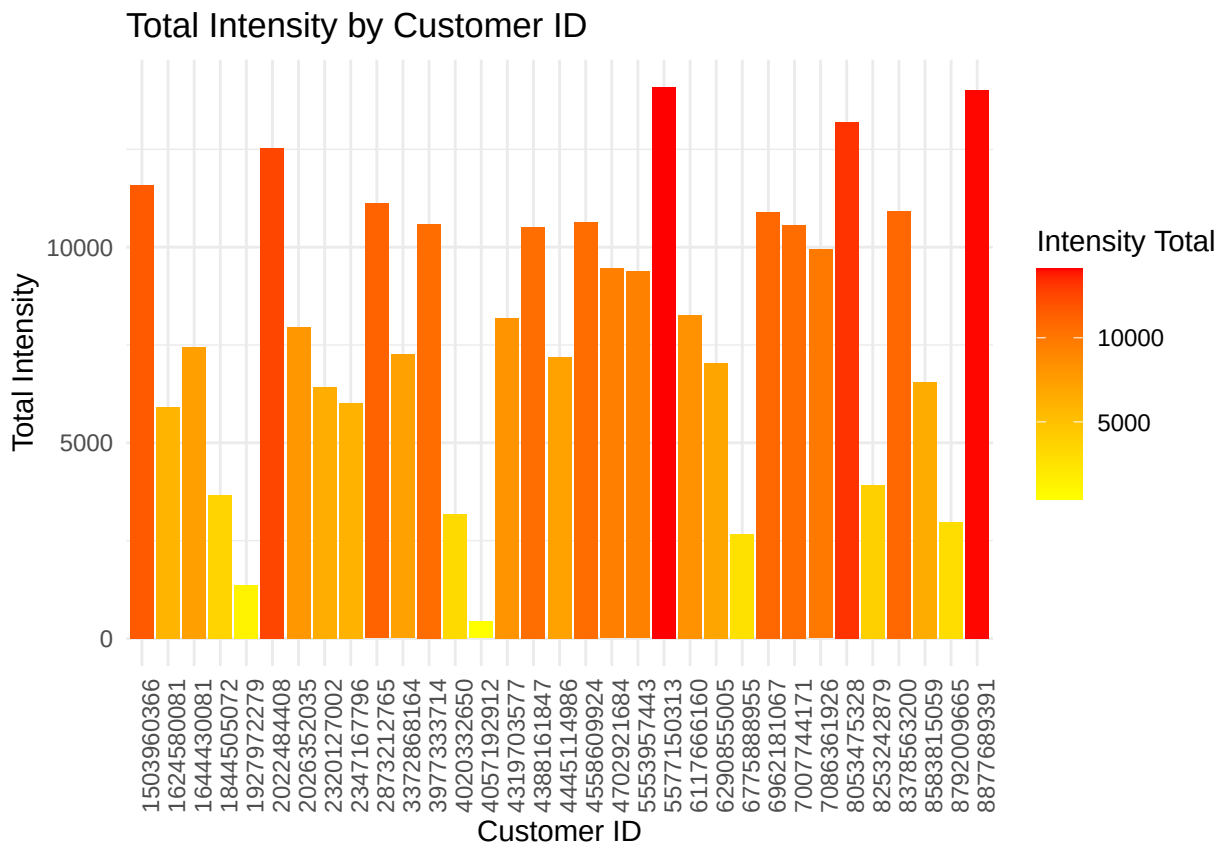
```r
# Create a heatmap to display average total intensity over time for each user
ggplot(avg_intensity, aes(x = ActivityHour, y = Id, fill = AverageIntensity)) +
  geom_tile() +
  labs(x = "Activity Hour", y = "User ID", fill = "Average Intensity") +
  ggtitle("Average Intensity Over Time for Each User") +
```

```
  scale_fill_gradient(low = "yellow", high = "red") +
  theme_minimal()
```



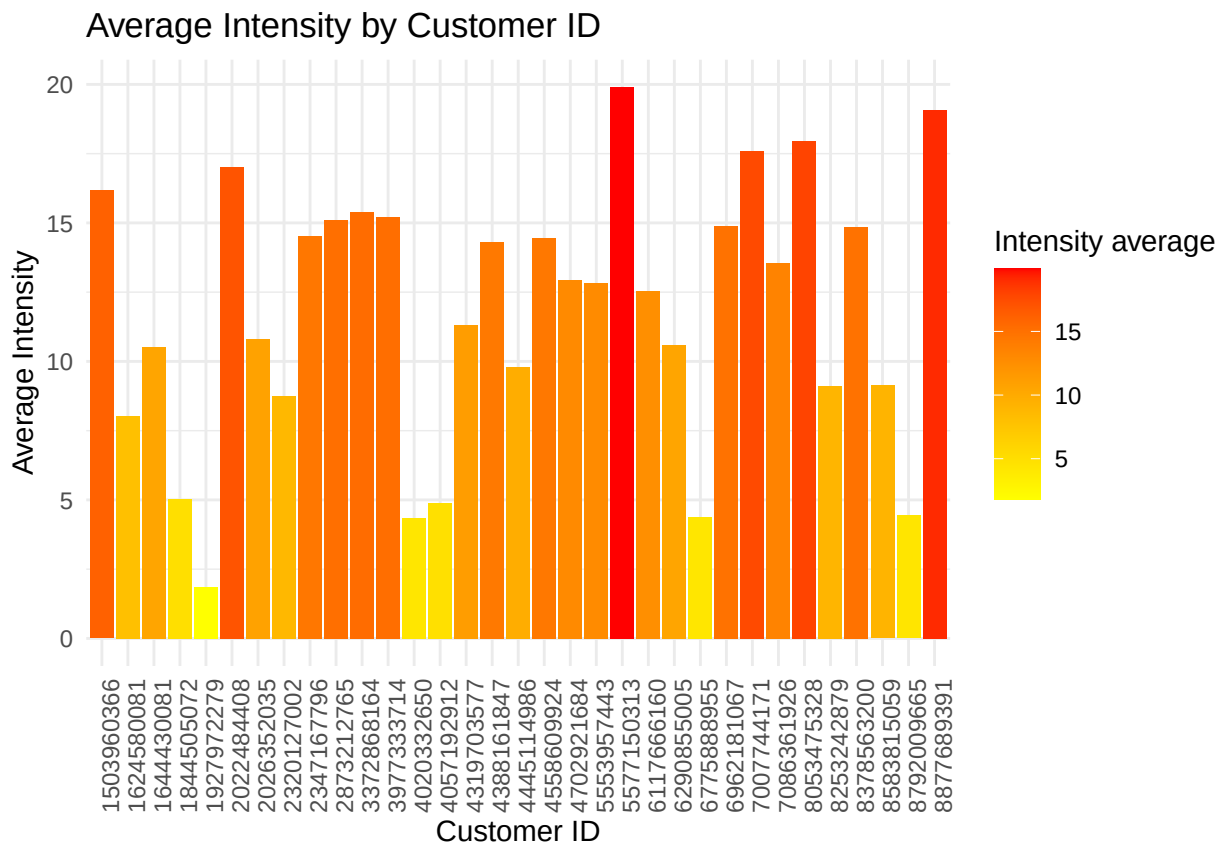
This plot shows the Sum of Intensities of each User

```
#Create barplot showing ID & the sum of Intensity
ggplot(Total_Int, aes(x = Id, y = TotalIntensity, fill = TotalIntensity)) +
  geom_bar(stat = "identity") +
  scale_fill_gradient(low = "yellow", high = "red") +
  labs(x = "Customer ID", y = "Total Intensity", fill = "Intensity Total") +
  ggtitle("Total Intensity by Customer ID") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90))
```



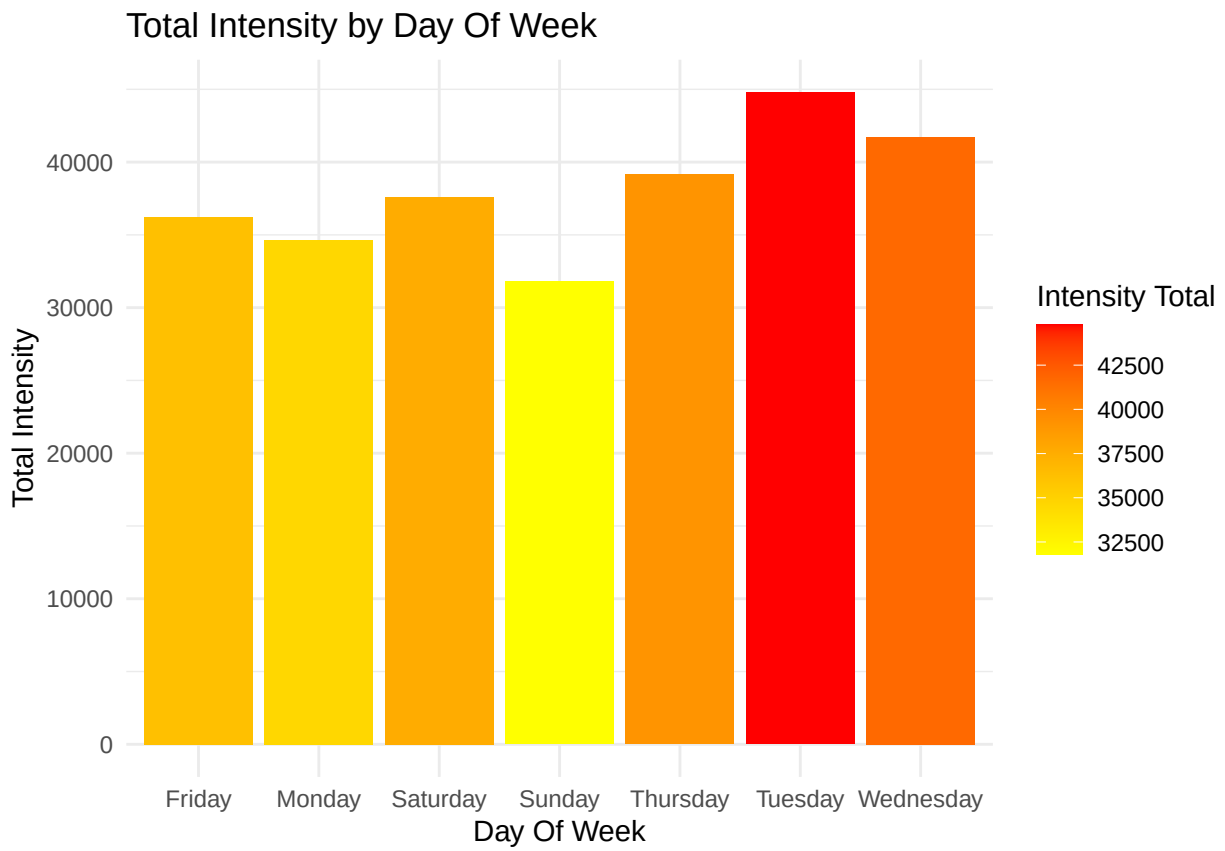
This plot demonstrates the Average Intensity of each User

```
#Create barplot showing ID & the average of Intensity
ggplot(Av_Tot_Int, aes(x = Id, y = AverageIntensity, fill = AverageIntensity)) +
  geom_bar(stat = "identity") +
  scale_fill_gradient(low = "yellow", high = "red") +
  labs(x = "Customer ID", y = "Average Intensity", fill = "Intensity average") +
  ggtitle("Average Intensity by Customer ID") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90))
```



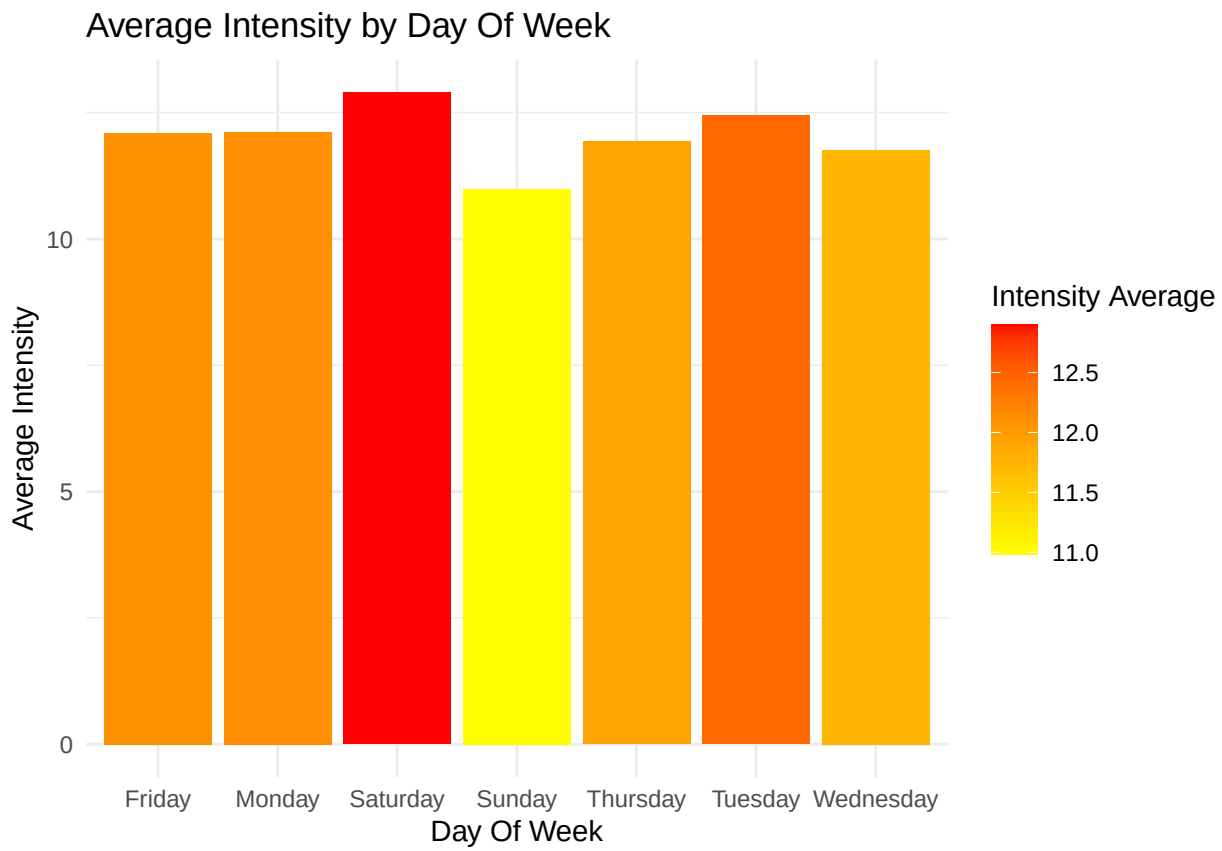
Now, we look at trends for days of the week & the total (summed) intensity

```r
#total intensity
Total_Int_day <- aggregate(TotalIntensity ~ DayOfWeek, data = hourlyIntensities_merged, FUN = sum)

#average intensity
Av_Tot_Int_DOW <- aggregate(TotalIntensity ~ DayOfWeek, data = hourlyIntensities_merged, FUN = mean)
colnames(Av_Tot_Int_DOW)[colnames(Av_Tot_Int_DOW) == "TotalIntensity"] <- "AverageIntensity"


#plots of days & total intensity
ggplot(Total_Int_day, aes(x = DayOfWeek, y = TotalIntensity, fill = TotalIntensity)) +
  geom_bar(stat = "identity") +
  scale_fill_gradient(low = "yellow", high = "red") +
  labs(x = "Day Of Week", y = "Total Intensity", fill = "Intensity Total") +
  ggtitle("Total Intensity by Day Of Week") +
  theme_minimal()
```



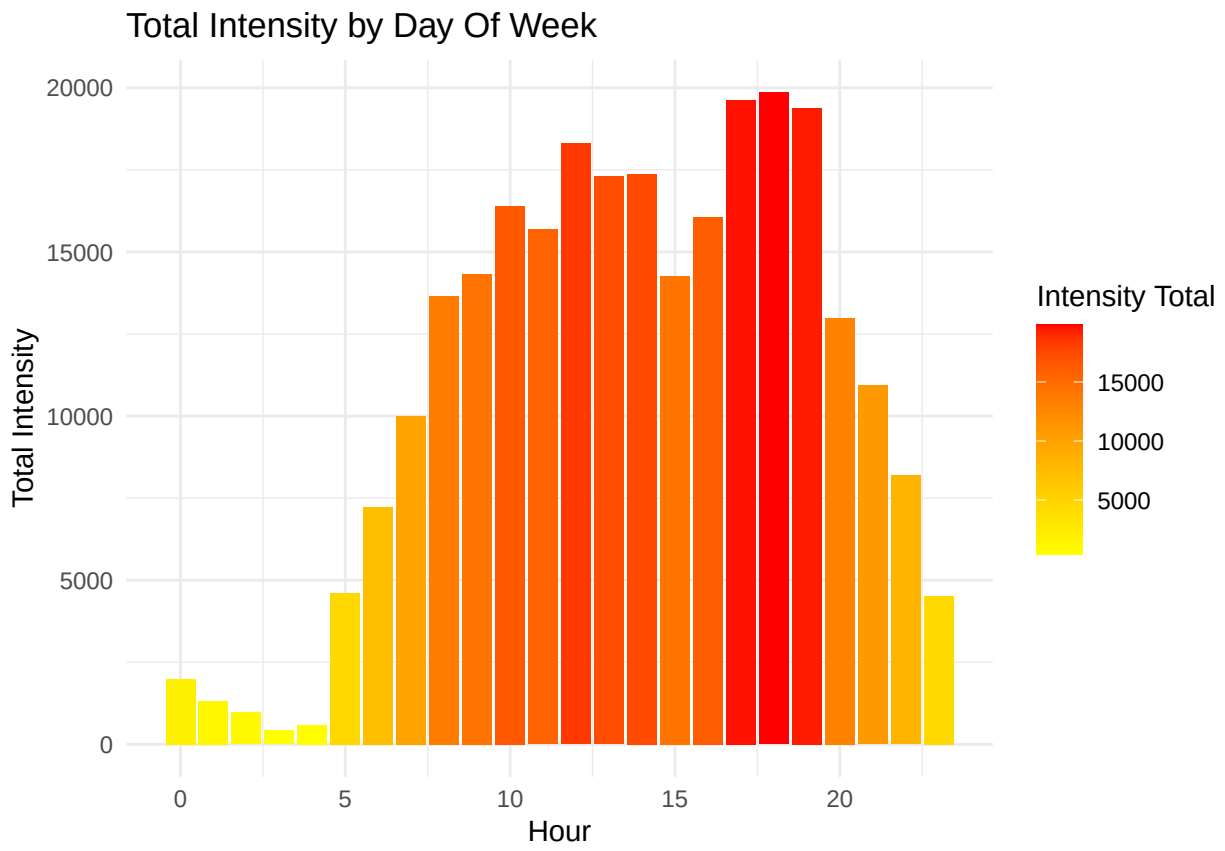
This plot shows the days & average intensity

```
ggplot(Av_Tot_Int_DOW, aes(x = DayOfWeek, y = AverageIntensity, fill = AverageIntensity)) +
  geom_bar(stat = "identity") +
  scale_fill_gradient(low = "yellow", high = "red") +
  labs(x = "Day Of Week", y = "Average Intensity", fill = "Intensity Average") +
  ggtitle("Average Intensity by Day Of Week") +
  theme_minimal()
```



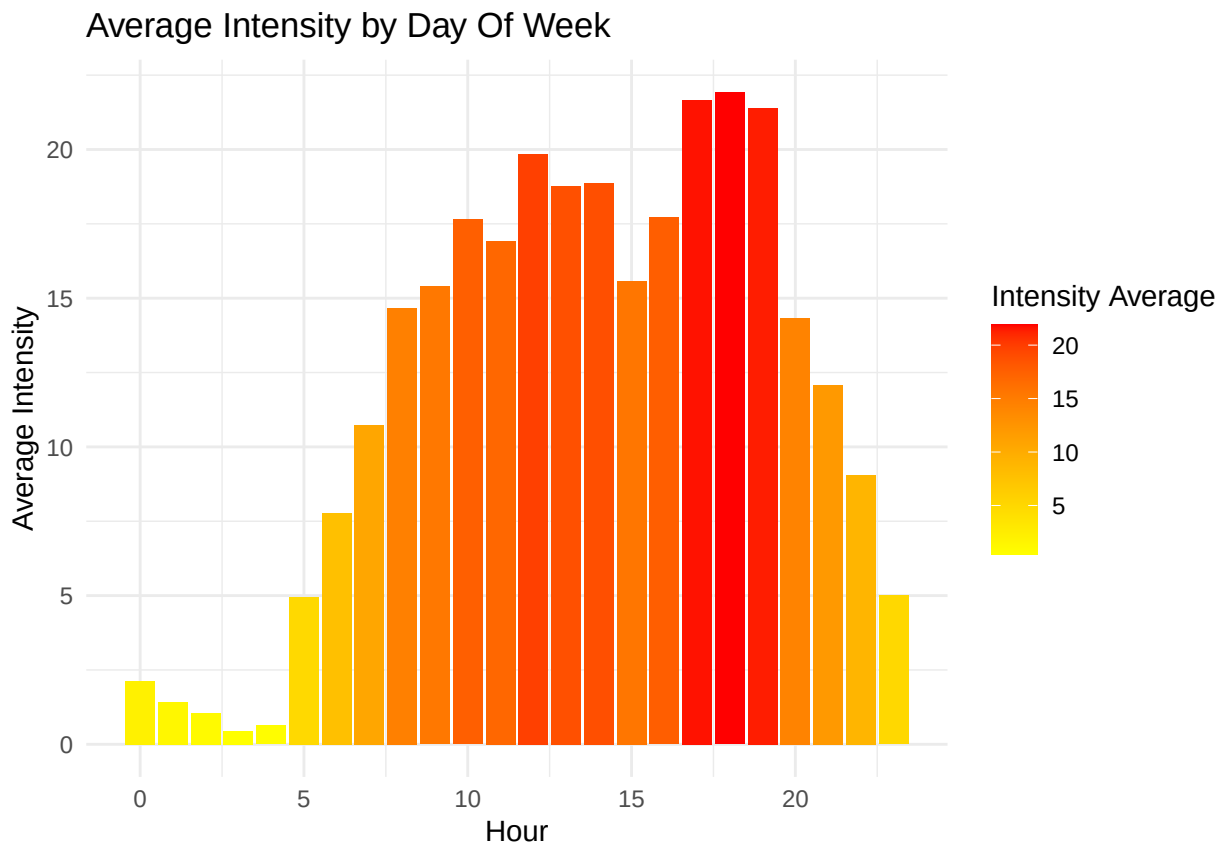
Now we compare hours and total intensity to average intensity

```
Total_Int_hr <- aggregate(TotalIntensity ~ Hour, data = hourlyIntensities_merged, FUN = sum)

Av_Tot_Int_hr <- aggregate(TotalIntensity ~ Hour, data = hourlyIntensities_merged, FUN = mean)
colnames(Av_Tot_Int_hr)[colnames(Av_Tot_Int_hr) == "TotalIntensity"] <- "AverageIntensity"

#plots of days & total intensity
ggplot(Total_Int_hr, aes(x = Hour, y = TotalIntensity, fill = TotalIntensity)) +
  geom_bar(stat = "identity") +
  scale_fill_gradient(low = "yellow", high = "red") +
  labs(x = "Hour", y = "Total Intensity", fill = "Intensity Total") +
  ggtitle("Total Intensity by Day Of Week") +
  theme_minimal()
```



This plot shows the average intensity by day of the week.

```
ggplot(Av_Tot_Int_hr, aes(x = Hour, y = AverageIntensity, fill = AverageIntensity)) +
  geom_bar(stat = "identity") +
  scale_fill_gradient(low = "yellow", high = "red") +
  labs(x = "Hour", y = "Average Intensity", fill = "Intensity Average") +
  ggtitle("Average Intensity by Day Of Week") +
  theme_minimal()
```



***Conclusion*:**

Data was gathered from 33 unique users examining their steps, intensities, and caloric output.

Less than half of BellaBeat's users (approximately 39%) exceeded the average caloric output of 97.39 calories. Additionally, Tuesdays were identified as the day of the week with the highest total caloric output, while Sundays had the lowest caloric output. This suggests that Sundays are commonly observed as rest days, allowing the body to recover from work. Furthermore, it was found that the hours between 5-7 PM had the highest caloric output, steps, total intensity, and average intensity. This implies that the 33 users engage in physical activity or go to the gym after work.

***Next steps*:**

To enhance the user experience, it would be beneficial to introduce a feature in the BellaBeat app that sends notifications to users' phones indicating total step goals, caloric output goals, and intensity goals. Additionally, incorporating a personal fitness trainer functionality could provide personalized training plans based on user goals and biometric monitoring. Real-time feedback and motivational support for users of all athletic levels could also be integrated to help track personal goals effectively.