Homework 5

Ian Mulchrone

```
library(readr)
library(tidyr)
f1 <- read_csv("f1data2024.csv", col_names = TRUE, show_col_types = FALSE)
head(f1)
```

```
## # A tibble: 6 x 32
##   Time        Driver DriverNumber LapTime LapNumber Stint PitOutTime PitInTime Sector1Time Sector2Tim
##   <chr>       <chr>         <dbl>   <dbl>     <dbl> <dbl> <chr>      <chr>           <dbl>       <dbl
## 1 0 days 01:~ VER               1    97.3         1     1 <NA>       <NA>             NA          41
## 2 0 days 01:~ VER               1    96.3         2     1 <NA>       <NA>             30.9        41
## 3 0 days 01:~ VER               1    96.8         3     1 <NA>       <NA>             31.0        42
## 4 0 days 01:~ VER               1    96.6         4     1 <NA>       <NA>             30.9        41
## 5 0 days 01:~ VER               1    97.2         5     1 <NA>       <NA>             31.3        42
## 6 0 days 01:~ VER               1    97.1         6     1 <NA>       <NA>             31.0        42
## # i 11 more variables: FreshTyre <lgl>, Team <chr>, LapStartTime <chr>, LapStartDate <dttm>, TrackSta
```

```
summary(f1)
```

```
##      Time              Driver           DriverNumber      LapTime           LapNumber         Stint
##  Length:26606       Length:26606       Min.   : 1.00   Min.   :  67.69   Min.   : 1.00   Min.   :1.00
##  Class :character   Class :character   1st Qu.:11.00   1st Qu.:  81.96   1st Qu.:15.00   1st Qu.:1.00
##  Mode  :character   Mode  :character   Median :22.00   Median :  89.08   Median :30.00   Median :2.00
##                                        Mean   :28.99   Mean   :  92.73   Mean   :30.51   Mean   :1.99
##                                        3rd Qu.:44.00   3rd Qu.:  98.70   3rd Qu.:45.00   3rd Qu.:3.00
##                                        Max.   :81.00   Max.   :2526.25   Max.   :78.00   Max.   :5.00
##                                                        NA's   :225
##  Sector3SessionTime    SpeedI1        SpeedI2         SpeedFL         SpeedST      IsPersonalBest
##  Length:26606       Min.   : 59    Min.   : 57.0   Min.   :  1.0   Min.   : 64.0   Mode :logical   Le
##  Class :character   1st Qu.:214    1st Qu.:211.0   1st Qu.:248.0   1st Qu.:289.0   FALSE:21094     Cl
##  Mode  :character   Median :273    Median :257.0   Median :275.0   Median :302.0   TRUE :5497      Mo
##                     Mean   :255    Mean   :247.6   Mean   :268.1   Mean   :296.6   NA's :15
##                     3rd Qu.:289    3rd Qu.:280.0   3rd Qu.:291.0   3rd Qu.:313.0
##                     Max.   :357    Max.   :343.0   Max.   :357.0   Max.   :362.0
##                     NA's   :4146   NA's   :54      NA's   :878     NA's   :2184
##   TrackStatus         Position       Deleted         DeletedReason      FastF1Generated IsAccurate
##  Min.   :   1.00   Min.   : 1.000   Mode :logical   Length:26606       Mode :logical   Mode :logical
##  1st Qu.:   1.00   1st Qu.: 5.000   FALSE:26280     Class :character   FALSE:26578     FALSE:3049
##  Median :   1.00   Median :10.000   TRUE :326       Mode  :character   TRUE :28        TRUE :23557
##  Mean   :   7.03   Mean   : 9.793
##  3rd Qu.:   1.00   3rd Qu.:14.000
##  Max.   :2671.00   Max.   :20.000
##                    NA's   :28
```

The data consists of information from laps during Formula One races. There are multiple time variables documenting the time each lap takes place, variuos points throughout each individual lap, as well as a date variable that is also a measure of time. All of these time variables are irrelevant and will be removed from the data. Categorical variables such as driver name, team name, and tyre type will all need to converted to dummy variables. Finally, lap time, sector times, and the speed stats (measured in km/h) will need to be cleaned for nulls and outliers and explore the relationship between speed and lap time.

```
library(dplyr)
f1 <- f1 %>% select(-Time, -DriverNumber, -Stint, -Sector1SessionTime, -Sector2SessionTime, -Sector3Ses
head(f1)
```

```
## # A tibble: 6 x 20
##   Driver LapTime LapNumber PitOutTime PitInTime Sector1Time Sector2Time Sector3Time SpeedI1 SpeedI2 
##   <chr>    <dbl>     <dbl> <chr>      <chr>           <dbl>       <dbl>       <dbl>   <dbl>   <dbl>
## 1 VER       97.3         1 <NA>       <NA>             NA         41.3        23.6     234     250
## 2 VER       96.3         2 <NA>       <NA>             30.9       41.7        23.7     232     248
## 3 VER       96.8         3 <NA>       <NA>             31.0       42.0        23.8     231     243
## 4 VER       96.6         4 <NA>       <NA>             30.9       41.9        23.8     233     253
## 5 VER       97.2         5 <NA>       <NA>             31.3       42.1        23.9     231     245
## 6 VER       97.1         6 <NA>       <NA>             31.0       42.2        23.9      NA     247
```

```
dim(f1)
```

```
## [1] 26606    20
```

Before we begin exploring our data, I believe it is important to remove the rows with null values in the continuous variables.

```
f1 <- f1 %>% drop_na(LapTime,Sector1Time,Sector2Time,Sector3Time,SpeedI1,SpeedI2,SpeedFL,SpeedST)
dim(f1)
```

```
## [1] 19548    20
```

The rows where Deleted = True can be removed since those laps were considered invalid times. Also, any row where TrackStatus is not 1 or when a pit stop is completed (PitOutTime not Null) should also be removed since we want to focus on "normal" laps without problems on the track or when doing a pit stop. Those columns can then be removed.

```
f1 <- subset(f1, Deleted == 'FALSE')
f1 <- subset(f1, TrackStatus == 1)
f1 <- f1[is.na(f1$PitOutTime),]
f1 <- f1 %>% select(-Deleted, -TrackStatus, -PitOutTime, -PitInTime)
head(f1)
```

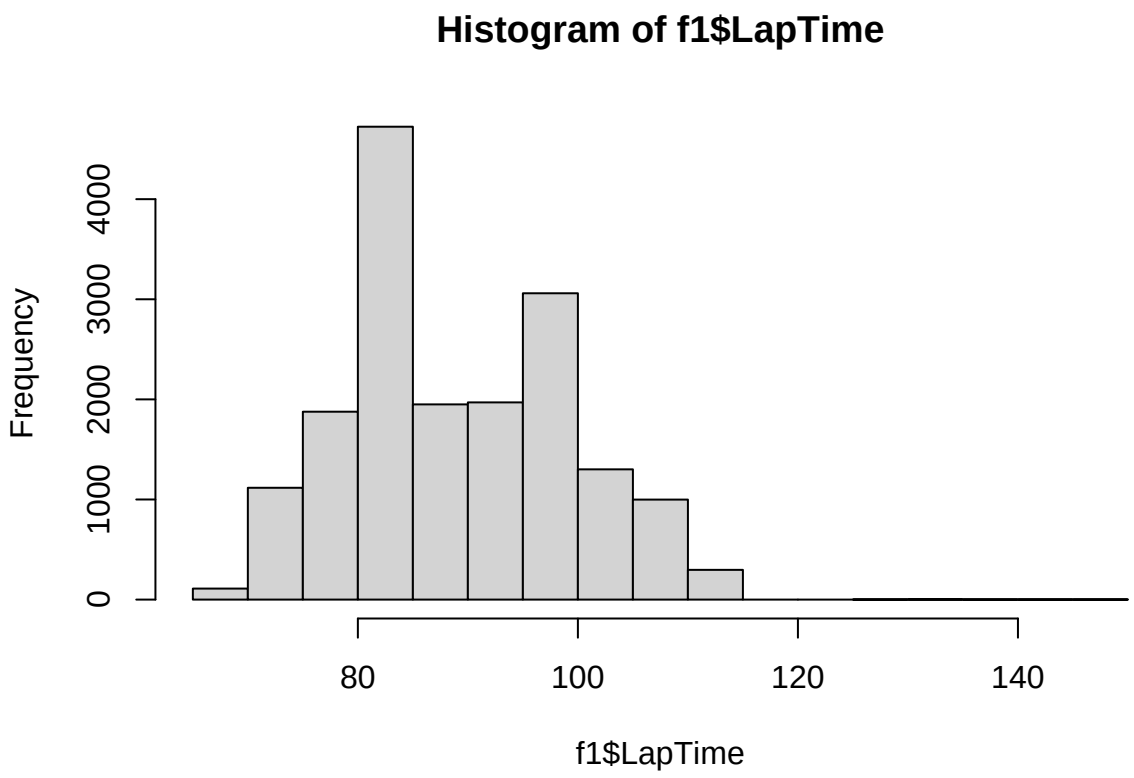
```
## # A tibble: 6 x 16
##   Driver LapTime LapNumber Sector1Time Sector2Time Sector3Time SpeedI1 SpeedI2 SpeedFL SpeedST IsPers
##   <chr>    <dbl>     <dbl>       <dbl>       <dbl>       <dbl>   <dbl>   <dbl>   <dbl>   <dbl> <lgl>
## 1 VER       96.3         2        30.9        41.7        23.7     232     248     276     287 TRUE 
## 2 VER       96.8         3        31.0        42.0        23.8     231     243     276     290 FALSE
## 3 VER       97.2         5        31.3        42.1        23.9     231     245     276     289 FALSE
## 4 VER       97.0         7        31.0        42.1        23.9     232     242     277     291 FALSE
## 5 VER       97.2        16        31.0        42.0        24.1     233     260     277     292 FALSE
## 6 VER       95.3        19        30.6        41.2        23.5     233     248     278     288 TRUE 
```

```r
dim(f1)
```

```
## [1] 17411    16
```

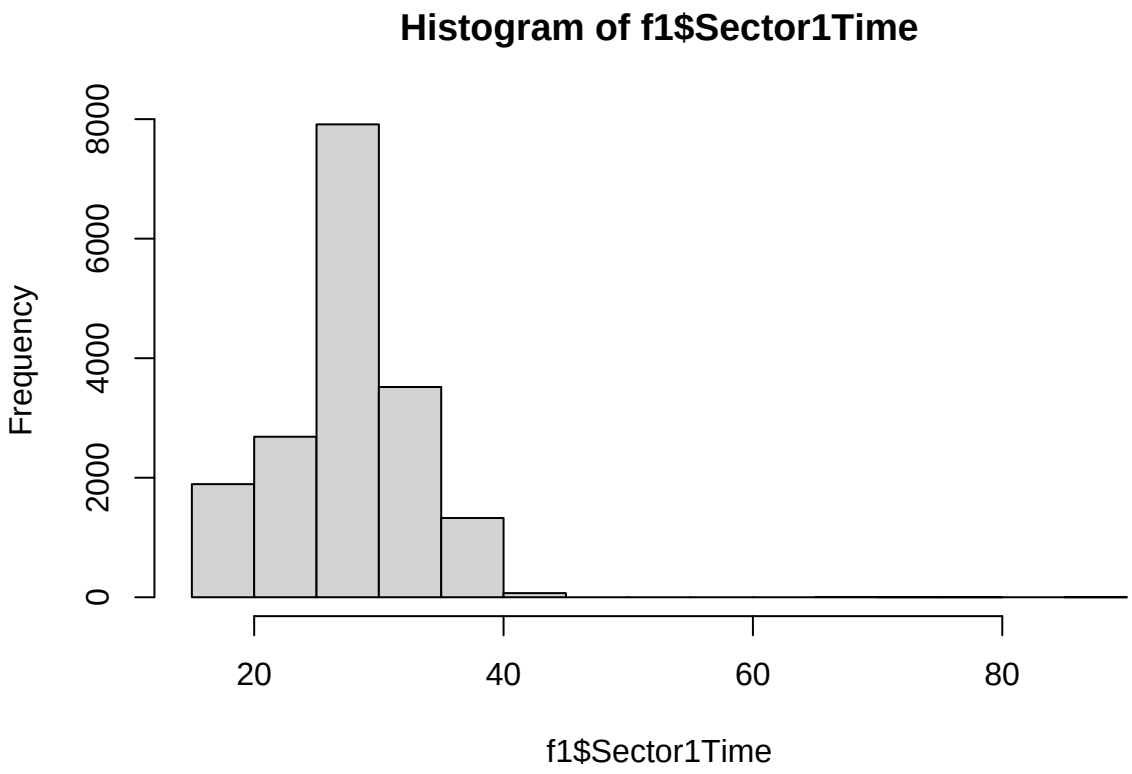
We can now begin exploring the data by examining the continuous variables.

```r
hist(f1$LapTime)
```



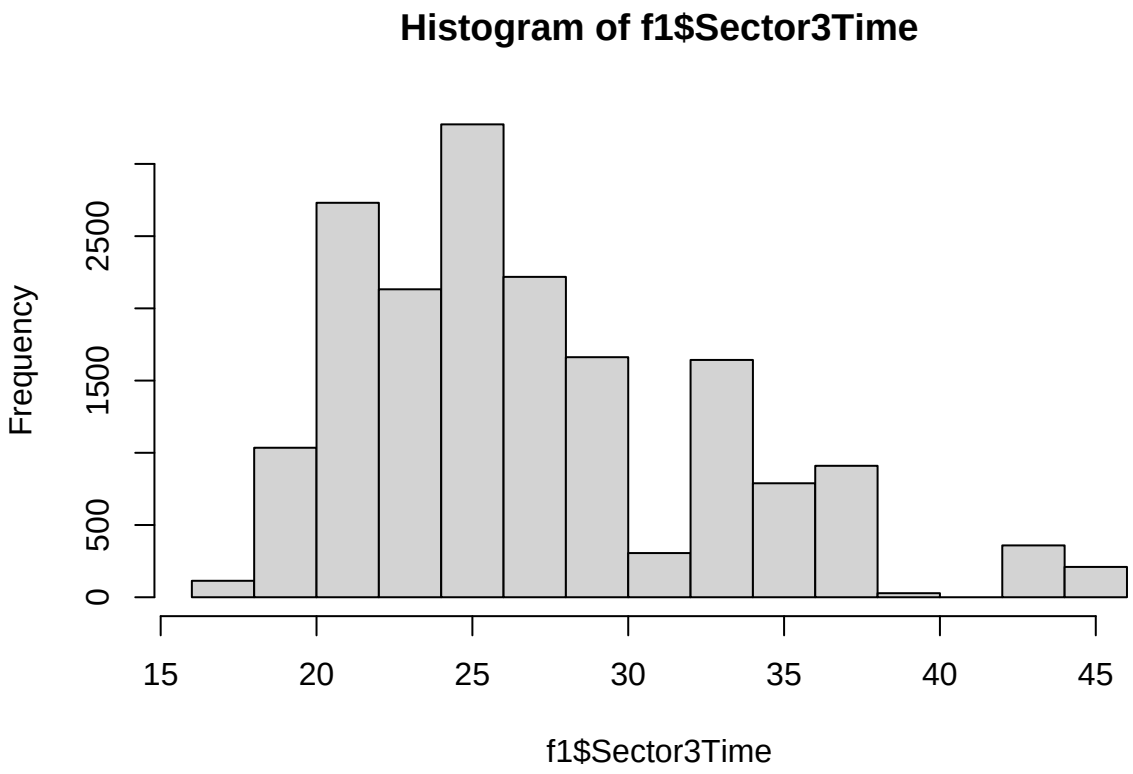
The distribution of lap time is bimodal with a right skew.

```r
hist(f1$Sector1Time)
```



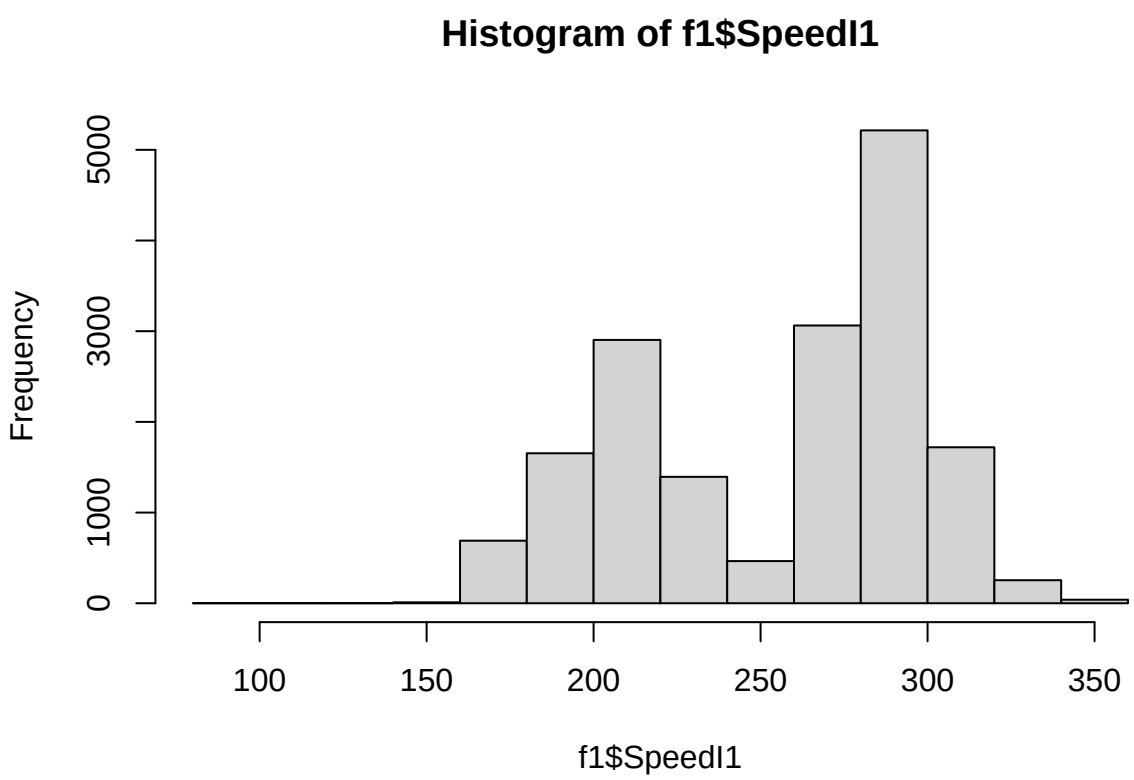

```r
hist(f1$Sector2Time)
```


```
hist(f1$Sector3Time)
```

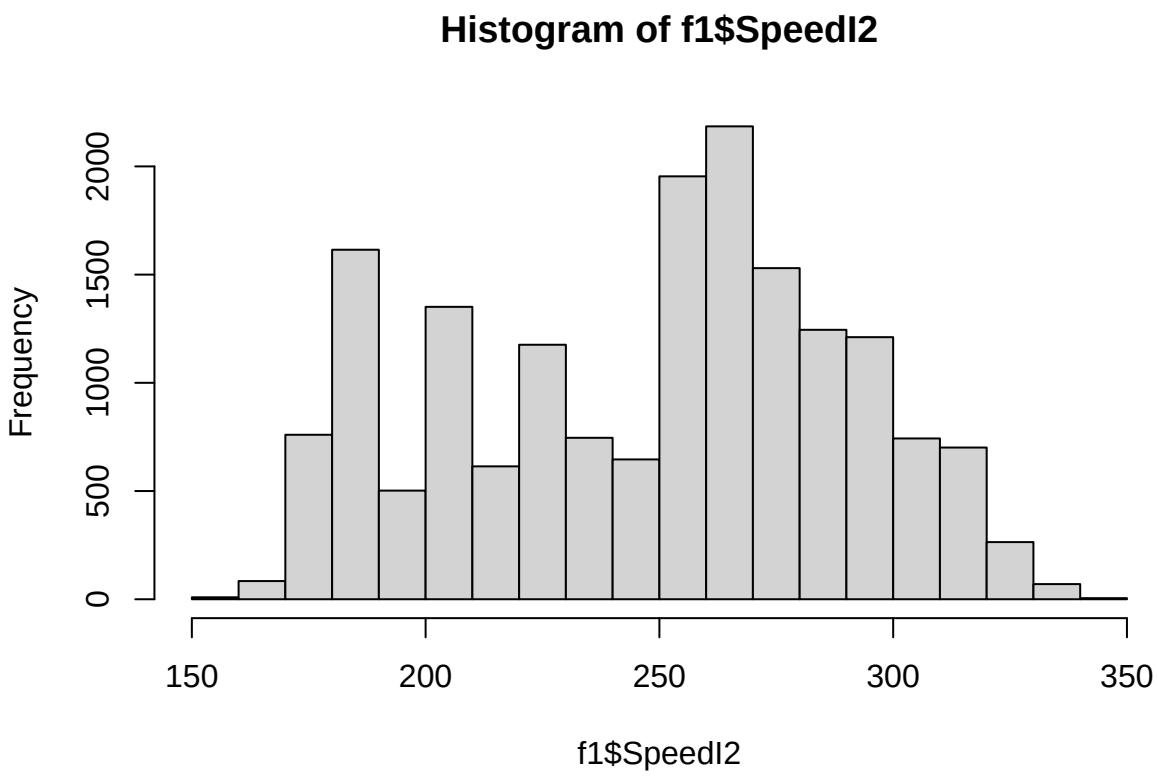


The distribution of Sector2Time is much more evenly distributed, while Sectors 1 and 3 are both right skewed like LapTime.

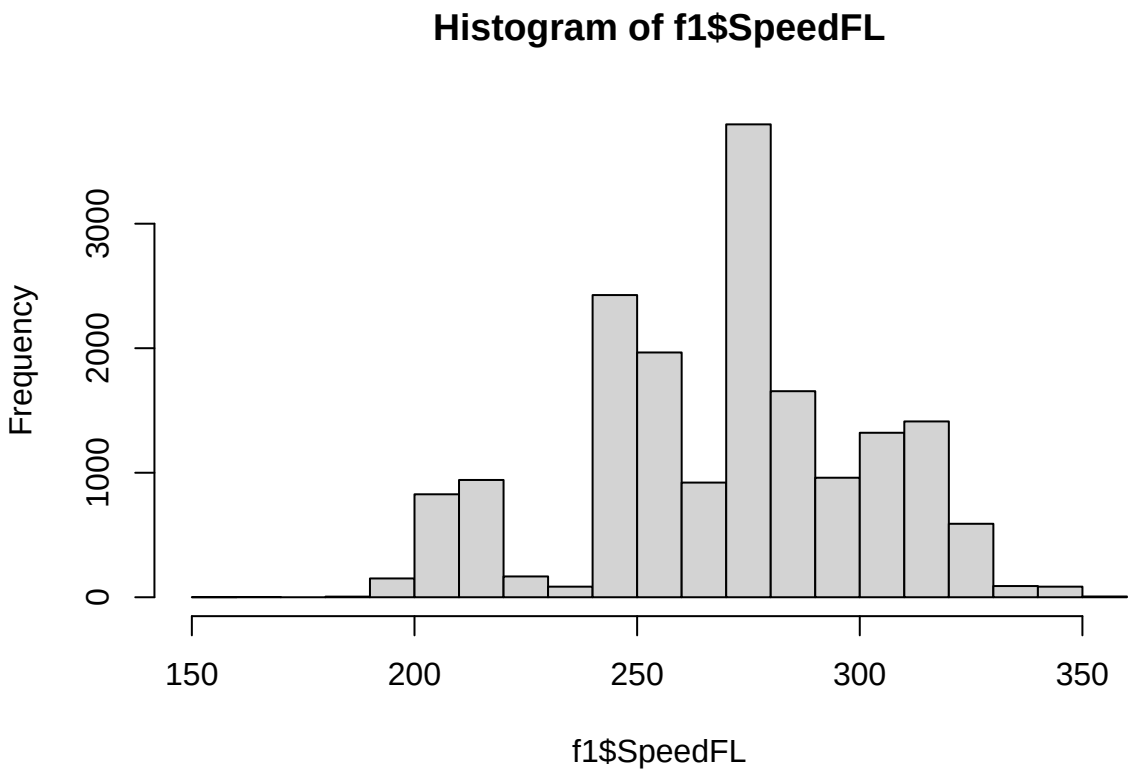
```
hist(f1$SpeedI1)
```



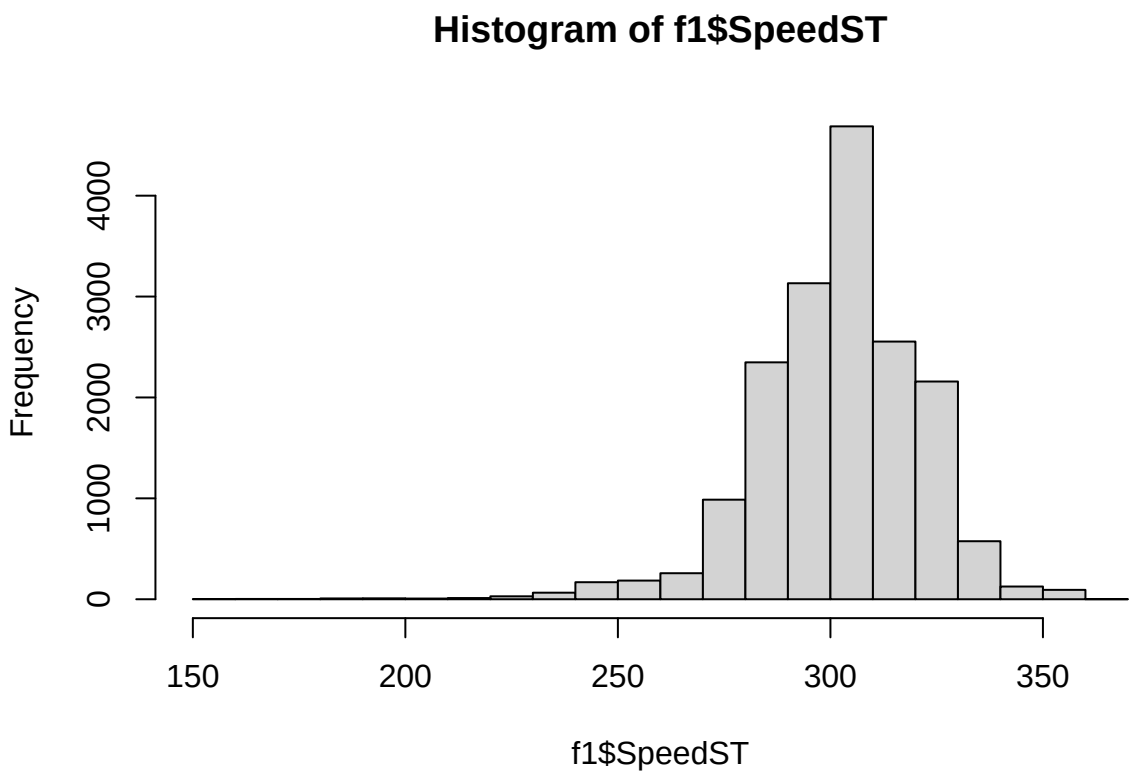

```
hist(f1$SpeedI2)
```




```r
hist(f1$SpeedFL)
```




```r
hist(f1$SpeedST)
```

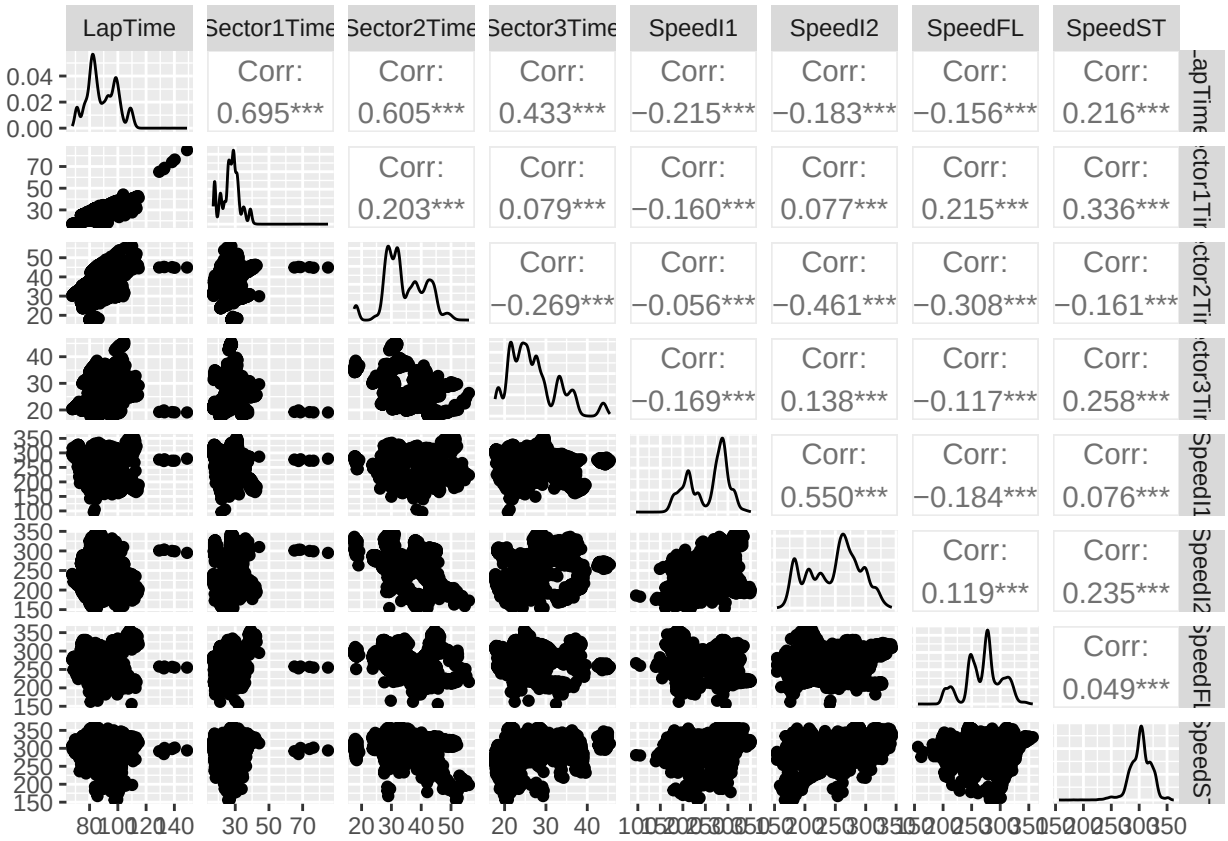


The speed variables all display a left tail skew. We can plot the various speed stats against lap time and sector times to see if there is a relationship between them.

```
library(ggplot2)
library(GGally)
ggpairs(f1, columns=c("LapTime","Sector1Time","Sector2Time","Sector3Time","SpeedI1","SpeedI2","SpeedFL"
```

```
## plot: [1, 1] [==>-------------------------------------------------------------------------------
## plot: [1, 2] [=====>-------------------------------------------------------------------------------
## plot: [1, 3] [=========>-------------------------------------------------------------------------------
## plot: [1, 4] [============>-------------------------------------------------------------------------------
## plot: [1, 5] [==============>-------------------------------------------------------------------------------
## plot: [1, 6] [==================>-------------------------------------------------------------------------------
## plot: [1, 7] [======================>-------------------------------------------------------------------------------
## plot: [1, 8] [=========================>-------------------------------------------------------------------------------
## plot: [2, 1] [============================>-------------------------------------------------------------------------------
## plot: [2, 2] [==============================>-------------------------------------------------------------------------------
## plot: [2, 3] [==================================>-------------------------------------------------------------------------------
## plot: [2, 4] [====================================>-------------------------------------------------------------------------------
## plot: [2, 5] [========================================>-------------------------------------------------------------------------------
## plot: [2, 6] [==========================================>-------------------------------------------------------------------------------
## plot: [2, 7] [============================================>-------------------------------------------------------------------------------
## plot: [2, 8] [==============================================>-------------------------------------------------------------------------------
## plot: [3, 1] [================================================>-------------------------------------------------------------------------------
## plot: [3, 2] [==================================================>-------------------------------------------------------------------------------
## plot: [3, 3] [====================================================>-------------------------------------------------------------------------------
```

```
## plot: [3, 4] [=============================================================>----------------------
## plot: [3, 5] [==================================================================>-------------------
## plot: [3, 6] [=======================================================================>----------------
## plot: [3, 7] [============================================================================>-----------
## plot: [3, 8] [=================================================================================>---------
## plot: [4, 1] [======================================================================================>------
## plot: [4, 2] [=======================================================================================>----
## plot: [4, 3] [=========================================================================================>
## plot: [4, 4] [==========================================================================================
## plot: [4, 5] [==========================================================================================
## plot: [4, 6] [==========================================================================================
## plot: [4, 7] [==========================================================================================
## plot: [4, 8] [==========================================================================================
## plot: [5, 1] [==========================================================================================
## plot: [5, 2] [==========================================================================================
## plot: [5, 3] [==========================================================================================
## plot: [5, 4] [==========================================================================================
## plot: [5, 5] [==========================================================================================
## plot: [5, 6] [==========================================================================================
## plot: [5, 7] [==========================================================================================
## plot: [5, 8] [==========================================================================================
## plot: [6, 1] [==========================================================================================
## plot: [6, 2] [==========================================================================================
## plot: [6, 3] [==========================================================================================
## plot: [6, 4] [==========================================================================================
## plot: [6, 5] [==========================================================================================
## plot: [6, 6] [==========================================================================================
## plot: [6, 7] [==========================================================================================
## plot: [6, 8] [==========================================================================================
## plot: [7, 1] [==========================================================================================
## plot: [7, 2] [==========================================================================================
## plot: [7, 3] [==========================================================================================
## plot: [7, 4] [==========================================================================================
## plot: [7, 5] [==========================================================================================
## plot: [7, 6] [==========================================================================================
## plot: [7, 7] [==========================================================================================
## plot: [7, 8] [==========================================================================================
## plot: [8, 1] [==========================================================================================
## plot: [8, 2] [==========================================================================================
## plot: [8, 3] [==========================================================================================
## plot: [8, 4] [==========================================================================================
## plot: [8, 5] [==========================================================================================
## plot: [8, 6] [==========================================================================================
## plot: [8, 7] [==========================================================================================
## plot: [8, 8] [==========================================================================================
```



Looking at the relationship between these values, there is not a very high correlation between speed and lap time. The most significant relationship is in sector 2, where the correlation between Sector2Time and SpeedI2 is -0.461.

Now, for the driver, personal best, compound, and event variables, dummies will need to be created.

```
table(f1$Driver)
```

```
##
## ALB ALO BEA BOT COL DOO GAS HAM HUL LAW LEC MAG NOR OCO PER PIA RIC RUS SAI SAR STR TSU VER ZHO
## 746 925 111 956 272  31 801 893 848 216 936 804 930 767 852 936 678 924 915 526 828 782 831 903
```

```
library(fastDummies)
f1 <- dummy_cols(f1, select_columns = 'Driver')
f1 <- f1 %>% select(-Driver)
head(f1)
```

```
## # A tibble: 6 x 39
##   LapTime LapNumber Sector1Time Sector2Time Sector3Time SpeedI1 SpeedI2 SpeedFL SpeedST IsPersonalBes
##     <dbl>     <dbl>       <dbl>       <dbl>       <dbl>   <dbl>   <dbl>   <dbl>   <dbl> <lgl>
## 1    96.3         2        30.9        41.7        23.7     232     248     276     287 TRUE
## 2    96.8         3        31.0        42.0        23.8     231     243     276     290 FALSE
## 3    97.2         5        31.3        42.1        23.9     231     245     276     289 FALSE
## 4    97.0         7        31.0        42.1        23.9     232     242     277     291 FALSE
## 5    97.2        16        31.0        42.0        24.1     233     260     277     292 FALSE
## 6    95.3        19        30.6        41.2        23.5     233     248     278     288 TRUE
```

```
## # i 16 more variables: Driver_HUL <int>, Driver_LAW <int>, Driver_LEC <int>, Driver_MAG <int>, Drive
## #   Driver_SAR <int>, Driver_STR <int>, Driver_TSU <int>, Driver_VER <int>, Driver_ZHO <int>
```

```
f1$IsPersonalBest <- ifelse(f1$IsPersonalBest == 'TRUE',1,0)
table(f1$IsPersonalBest)
```

```
##
##     0     1
## 13216  4195
```

```
f1 <- dummy_cols(f1, select_columns = 'Compound')
f1 <- f1 %>% select(-Compound)
head(f1)
```

```
## # A tibble: 6 x 43
##   LapTime LapNumber Sector1Time Sector2Time Sector3Time SpeedI1 SpeedI2 SpeedFL SpeedST IsPersonalBes
##     <dbl>     <dbl>       <dbl>       <dbl>       <dbl>   <dbl>   <dbl>   <dbl>   <dbl>           <db
## 1    96.3         2        30.9        41.7        23.7     232     248     276     287
## 2    96.8         3        31.0        42.0        23.8     231     243     276     290
## 3    97.2         5        31.3        42.1        23.9     231     245     276     289
## 4    97.0         7        31.0        42.1        23.9     232     242     277     291
## 5    97.2        16        31.0        42.0        24.1     233     260     277     292
## 6    95.3        19        30.6        41.2        23.5     233     248     278     288
## # i 20 more variables: Driver_LAW <int>, Driver_LEC <int>, Driver_MAG <int>, Driver_NOR <int>, Drive
## #   Driver_STR <int>, Driver_TSU <int>, Driver_VER <int>, Driver_ZHO <int>, Compound_HARD <int>, Comp
```

```
# f1 <- dummy_cols(f1, select_columns = 'Team')
f1 <- f1 %>% select(-Team)
# head(f1)
```

Create new dummy variable to determine if the driver is in the lead

```
# f1$Leader <- ifelse(f1$Position == 1,1,0)
# f1 <- f1 %>% select(-Position)
# table(f1$Leader)
# f1 <- f1 %>% select(-Leader)
```

Final Processed Data

```
head(f1)
```

```
## # A tibble: 6 x 42
##   LapTime LapNumber Sector1Time Sector2Time Sector3Time SpeedI1 SpeedI2 SpeedFL SpeedST IsPersonalBes
##     <dbl>     <dbl>       <dbl>       <dbl>       <dbl>   <dbl>   <dbl>   <dbl>   <dbl>           <db
## 1    96.3         2        30.9        41.7        23.7     232     248     276     287
## 2    96.8         3        31.0        42.0        23.8     231     243     276     290
## 3    97.2         5        31.3        42.1        23.9     231     245     276     289
## 4    97.0         7        31.0        42.1        23.9     232     242     277     291
## 5    97.2        16        31.0        42.0        24.1     233     260     277     292
## 6    95.3        19        30.6        41.2        23.5     233     248     278     288
## # i 19 more variables: Driver_LEC <int>, Driver_MAG <int>, Driver_NOR <int>, Driver_OCO <int>, Drive
## #   Driver_TSU <int>, Driver_VER <int>, Driver_ZHO <int>, Compound_HARD <int>, Compound_INTERMEDIATE
```

```r
dim(f1)
```

```
## [1] 17411    42
```

In total, there are 41 columns and 17,411 rows

We can now use clustering on the lap data and compare them to the event labels.

```r
f1_cluster <- f1 %>% select(-Event)
set.seed(65236)
```

```r
library(caret)
preproc <- preProcess(f1_cluster, method = c("center","scale"))
predictors <- predict(preproc, f1_cluster)
```

Since there are 24 events, we will create 24 clusters

```r
fit <- kmeans(predictors, centers = 24, nstart = 30)
fit
```

```
## K-means clustering with 24 clusters of sizes 678, 746, 800, 526, 828, 541, 216, 896, 898, 747, 868, 9
##
## Cluster means:
##         LapTime     LapNumber  Sector1Time   Sector2Time  Sector3Time      SpeedI1      SpeedI2
## 1  -0.060376718 -2.893542e-02  0.071979190 -0.1050364777 -0.051322313 -0.013404754 -0.083422561  0.15
## 2   0.039439276 -7.802205e-02 -0.005453887 -0.0613875738  0.145148216 -0.133189116 -0.115872979 -0.01
## 3   0.030278503 -2.794308e-02  0.059175017 -0.1246736923  0.142777181  0.165332625  0.225444902 -0.00
## 4  -0.105395027 -3.417017e-02 -0.026600495  0.0080376327 -0.170734609  0.055459083 -0.092235553 -0.07
## 5  -0.022529781  1.030841e-02 -0.046061641 -0.0221891696  0.027880669 -0.079952530 -0.023813069 -0.08
## 6  -0.380962392  1.732989e-01 -1.052853360  1.5353209745 -1.475985003  0.893842532 -0.721896745  1.01
## 7   0.171430186  1.466232e-01 -0.277708574  0.3414715235  0.162339448 -0.065469304 -0.185399769 -0.45
## 8  -0.066539519  1.596182e-02 -0.053313622 -0.0511855974 -0.009855563 -0.039858007 -0.005341563 -0.07
## 9   0.196530334 -1.896189e-02  0.257959932  0.0008325494  0.110186326  0.085286563  0.154522360  0.00
## 10  0.016602979  5.697120e-04 -0.047235608 -0.0134961748  0.087953418 -0.126963056 -0.054081996 -0.11
## 11  0.004407413 -1.028032e-03  0.027844177 -0.1290996996  0.130845061 -0.182855178 -0.029234046 -0.01
## 12 -0.022085003  1.388712e-05  0.006247217 -0.0341358646 -0.005529612  0.009721369  0.047045893 -0.07
## 13  0.124403551 -4.432383e-02  0.234999833 -0.0864324818  0.104303096  0.054434625  0.083289004  0.04
## 14 -0.088298947 -7.816607e-04 -0.023280307 -0.1341565628  0.019766998 -0.012334872  0.082432200 -0.01
## 15 -0.079104354  3.174026e-02 -0.091821000 -0.0399098285 -0.009842367 -0.075871765 -0.056486701 -0.10
## 16 -0.095349951  1.236619e-02 -0.068308833 -0.0638548556 -0.032402037  0.028380278  0.145653634 -0.00
## 17  0.549997942 -1.415954e-01  0.283043905  0.4435378732  0.201508954 -0.068481719 -0.089780356  0.01
## 18  0.135640219 -1.875061e-02  0.236843050 -0.0795115513  0.114475375  0.019265492  0.045945573  0.07
## 19 -0.062818159  1.340308e-02 -0.018650721 -0.1039855888  0.025778030 -0.011222941  0.062030310 -0.01
## 20 -0.057223624  2.975611e-02 -0.007847897 -0.1193173997  0.043508025 -0.017062279  0.091293874 -0.01
## 21  0.001153051  2.577321e-02 -0.009841354  0.0117104561 -0.002394403 -0.083823039 -0.032013681 -0.09
## 22  0.026133252 -6.995623e-03  0.011029493 -0.0583200512  0.103118124 -0.057260421  0.017394210 -0.01
## 23  0.073910645  4.579739e-03  0.157486573 -0.0151757034  0.003901668 -0.095373054 -0.015032568 -0.01
## 24  0.684242597 -1.279513e+00  0.107072819 -0.3098590926  1.464999757 -1.296299259  0.243609661 -0.28
##    Driver_GAS Driver_HAM  Driver_HUL Driver_LAW    Driver_LEC Driver_MAG  Driver_NOR  Driver_OCO Driv
## 1  -0.2195932 -0.2325062 -0.22626426 -0.1120762 -0.2383486742  -0.220024 -0.23754026 -0.21466259 -0.1
## 2  -0.2195932 -0.2325062 -0.22626426 -0.1120762 -0.2383486742  -0.220024 -0.23754026 -0.21466259 -0.1
## 3  -0.2195932 -0.2325062 -0.22626426 -0.1120762 -0.2383486742   4.544699 -0.23754026 -0.21466259 -0.1
```

```
## 4  -0.2195932 -0.2325062 -0.22626426 -0.1120762 -0.2383486742  -0.220024 -0.23754026 -0.21466259 -0.1
## 5  -0.2195932 -0.2325062 -0.22626426 -0.1120762 -0.2383486742  -0.220024 -0.23754026 -0.21466259 -0.1
## 6   0.0450947  0.1110465 -0.09745775 -0.1120762 -0.0006858027  -0.220024  0.07482615  0.07356625  0.1
## 7  -0.2195932 -0.2325062 -0.22626426  8.9219902 -0.2383486742  -0.220024 -0.23754026 -0.21466259 -0.1
## 8  -0.2195932 -0.2325062 -0.22626426 -0.1120762 -0.2383486742  -0.220024 -0.23754026 -0.21466259 -0.1
## 9  -0.2195932 -0.2325062  4.04170643 -0.1120762 -0.2383486742  -0.220024 -0.23754026 -0.21466259 -0.1
## 10 -0.2195932 -0.2325062 -0.22626426 -0.1120762 -0.2383486742  -0.220024 -0.23754026 -0.21466259 -0.1
## 11 -0.2195932 -0.2325062 -0.22626426 -0.1120762 -0.2383486742  -0.220024 -0.23754026 -0.21466259 -0.1
## 12 -0.2195932 -0.2325062 -0.22626426 -0.1120762  4.1952931704  -0.220024 -0.23754026 -0.21466259 -0.1
## 13 -0.2195932 -0.2325062 -0.22626426 -0.1120762 -0.2383486742  -0.220024 -0.23754026 -0.21466259  4.4
## 14 -0.2195932 -0.2325062 -0.22626426 -0.1120762 -0.2383486742  -0.220024  4.20957095 -0.21466259 -0.1
## 15 -0.2195932 -0.2325062 -0.22626426 -0.1120762 -0.2383486742  -0.220024 -0.23754026 -0.21466259 -0.1
## 16 -0.2195932  4.1415633 -0.22626426 -0.1120762 -0.2383486742  -0.220024 -0.23754026 -0.21466259 -0.1
## 17 -0.2195932 -0.2325062 -0.22626426 -0.1120762 -0.2383486742  -0.220024 -0.23754026 -0.21466259 -0.1
## 18 -0.2195932 -0.2325062 -0.22626426 -0.1120762 -0.2383486742  -0.220024 -0.23754026  4.65820610 -0.1
## 19 -0.2195932 -0.2325062 -0.22626426 -0.1120762 -0.2383486742  -0.220024 -0.23754026 -0.21466259 -0.1
## 20 -0.2195932 -0.2325062 -0.22626426 -0.1120762 -0.2383486742  -0.220024 -0.23754026 -0.21466259 -0.1
## 21  4.5536127 -0.2325062 -0.22626426 -0.1120762 -0.2383486742  -0.220024 -0.23754026 -0.21466259 -0.1
## 22 -0.2195932 -0.2325062 -0.22626426 -0.1120762 -0.2383486742  -0.220024 -0.23754026 -0.21466259 -0.1
## 23 -0.2195932 -0.2325062 -0.22626426 -0.1120762 -0.2383486742  -0.220024 -0.23754026 -0.21466259 -0.1
## 24 -0.2195932 -0.2325062  2.63257970 -0.1120762 -0.2383486742   1.246045 -0.23754026 -0.21466259  0.1
##    Compound_INTERMEDIATE Compound_MEDIUM Compound_SOFT Compound_WET
## 1            -0.09027664    -0.191015915   0.103781183   -0.0273344
## 2            -0.10841036    -0.006747031   0.026579191   -0.0273344
## 3            -0.14871458     0.041990857  -0.008159147   -0.0273344
## 4            -0.10401821    -0.059469737   0.146361327   -0.0273344
## 5            -0.10369131     0.101566271   0.112343996   -0.0273344
## 6             3.45744508    -0.702188241  -0.259809009   -0.0273344
## 7             0.35257457    -0.022977160   0.006487141   -0.0273344
## 8            -0.12195272     0.076162346  -0.163514419   -0.0273344
## 9            -0.15570305    -0.098415826  -0.108825955   -0.0273344
## 10           -0.10363680    -0.138610267   0.191198584   -0.0273344
## 11           -0.09929148    -0.057951563   0.161461332   -0.0273344
## 12           -0.12398125     0.024526266  -0.042376462   -0.0273344
## 13           -0.09373642     0.158867255   0.005589248   -0.0273344
## 14           -0.11280209     0.067734400   0.011946127   -0.0273344
## 15           -0.07269881     0.157918254  -0.053348247   -0.0273344
## 16           -0.13221951     0.012974975   0.270629140   -0.0273344
## 17           -0.06882257    -0.115912691  -0.259809009   -0.0273344
## 18           -0.09041197    -0.028160137  -0.114471868   -0.0273344
## 19           -0.12924458     0.124411977  -0.139783393   -0.0273344
## 20           -0.11358964     0.090400467  -0.080976199   -0.0273344
## 21           -0.14828937     0.122379297  -0.051982553   -0.0273344
## 22           -0.12834079     0.189144916  -0.018401555   -0.0273344
## 23           -0.14698754    -0.144111570   0.092226744   -0.0273344
## 24           -0.28921430    -0.702188241  -0.259809009   36.5818374
##
## Clustering vector:
##    [1] 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 13 13 13 13 13 13 13 13 13 13 13 13 13
##   [77]  8  8  8  8  8  8  8 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
##  [153] 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 16 16 16 16 16 16 16 16 16 16 16 16
##  [229] 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22  5  5
##  [305] 11 11 11 11 11 11 11 11 11 11 11  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
##  [381] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10  2  2
```

```
##  [457]  9  9  9  9  9  9  9  9  9  9  9  9 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18
##  [533] 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
##  [609] 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 13 13 13 13 13 13 13 13 13
##  [685] 22 22 22 22 22 22 22 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
##  [761]  4  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3 10 10 10 10
##  [837]  2  2  2  2  2  2  2  2  2  2  2  2  2  2 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
##  [913]  9  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
##  [989]  9  9  9  9  9  9  9  9  9  9  9  9
##  [ reached getOption("max.print") -- omitted 16411 entries ]
##
## Within cluster sum of squares by cluster:
##  [1] 10306.7374 11809.0776 10785.4243  7386.3033 12311.5525 18601.7476  3843.5631 12429.0588 24217.72
## [22] 12966.6385 13224.9697   214.7446
##  (between_SS / total_SS =  58.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"
```

```r
cluster_result <- data.frame(Event = f1$Event, Kmeans = fit$cluster)
head(cluster_result, n = 100)
```

```
##                  Event Kmeans
## 1   Bahrain Grand Prix     15
## 2   Bahrain Grand Prix     15
## 3   Bahrain Grand Prix     15
## 4   Bahrain Grand Prix     15
## 5   Bahrain Grand Prix     15
## 6   Bahrain Grand Prix     15
## 7   Bahrain Grand Prix     15
## 8   Bahrain Grand Prix     15
## 9   Bahrain Grand Prix     15
## 10  Bahrain Grand Prix     15
## 11  Bahrain Grand Prix     15
## 12  Bahrain Grand Prix     15
## 13  Bahrain Grand Prix     15
## 14  Bahrain Grand Prix     15
## 15  Bahrain Grand Prix     15
## 16  Bahrain Grand Prix     15
## 17  Bahrain Grand Prix     15
## 18  Bahrain Grand Prix     15
## 19  Bahrain Grand Prix     15
## 20  Bahrain Grand Prix     13
## 21  Bahrain Grand Prix     13
## 22  Bahrain Grand Prix     13
## 23  Bahrain Grand Prix     13
## 24  Bahrain Grand Prix     13
## 25  Bahrain Grand Prix     13
## 26  Bahrain Grand Prix     13
## 27  Bahrain Grand Prix     13
## 28  Bahrain Grand Prix     13
## 29  Bahrain Grand Prix     13
## 30  Bahrain Grand Prix     13
## 31  Bahrain Grand Prix     13
```

```
## 32  Bahrain Grand Prix     13
## 33  Bahrain Grand Prix     13
## 34  Bahrain Grand Prix     13
## 35  Bahrain Grand Prix     13
## 36  Bahrain Grand Prix     13
## 37  Bahrain Grand Prix     13
## 38  Bahrain Grand Prix     13
## 39  Bahrain Grand Prix     13
## 40  Bahrain Grand Prix     13
## 41  Bahrain Grand Prix     13
## 42  Bahrain Grand Prix     13
## 43  Bahrain Grand Prix     13
## 44  Bahrain Grand Prix     13
## 45  Bahrain Grand Prix     13
## 46  Bahrain Grand Prix     13
## 47  Bahrain Grand Prix     13
## 48  Bahrain Grand Prix     13
## 49  Bahrain Grand Prix     13
## 50  Bahrain Grand Prix     13
## 51  Bahrain Grand Prix     13
## 52  Bahrain Grand Prix     13
## 53  Bahrain Grand Prix     13
## 54  Bahrain Grand Prix     13
## 55  Bahrain Grand Prix      8
## 56  Bahrain Grand Prix      8
## 57  Bahrain Grand Prix      8
## 58  Bahrain Grand Prix      8
## 59  Bahrain Grand Prix      8
## 60  Bahrain Grand Prix      8
## 61  Bahrain Grand Prix      8
## 62  Bahrain Grand Prix      8
## 63  Bahrain Grand Prix      8
## 64  Bahrain Grand Prix      8
## 65  Bahrain Grand Prix      8
## 66  Bahrain Grand Prix      8
## 67  Bahrain Grand Prix      8
## 68  Bahrain Grand Prix      8
## 69  Bahrain Grand Prix      8
## 70  Bahrain Grand Prix      8
## 71  Bahrain Grand Prix      8
## 72  Bahrain Grand Prix      8
## 73  Bahrain Grand Prix      8
## 74  Bahrain Grand Prix      8
## 75  Bahrain Grand Prix      8
## 76  Bahrain Grand Prix      8
## 77  Bahrain Grand Prix      8
## 78  Bahrain Grand Prix      8
## 79  Bahrain Grand Prix      8
## 80  Bahrain Grand Prix      8
## 81  Bahrain Grand Prix      8
## 82  Bahrain Grand Prix      8
## 83  Bahrain Grand Prix      8
## 84  Bahrain Grand Prix     12
## 85  Bahrain Grand Prix     12
```

```
## 86  Bahrain Grand Prix     12
## 87  Bahrain Grand Prix     12
## 88  Bahrain Grand Prix     12
## 89  Bahrain Grand Prix     12
## 90  Bahrain Grand Prix     12
## 91  Bahrain Grand Prix     12
## 92  Bahrain Grand Prix     12
## 93  Bahrain Grand Prix     12
## 94  Bahrain Grand Prix     12
## 95  Bahrain Grand Prix     12
## 96  Bahrain Grand Prix     12
## 97  Bahrain Grand Prix     12
## 98  Bahrain Grand Prix     12
## 99  Bahrain Grand Prix     12
## 100 Bahrain Grand Prix     12
```

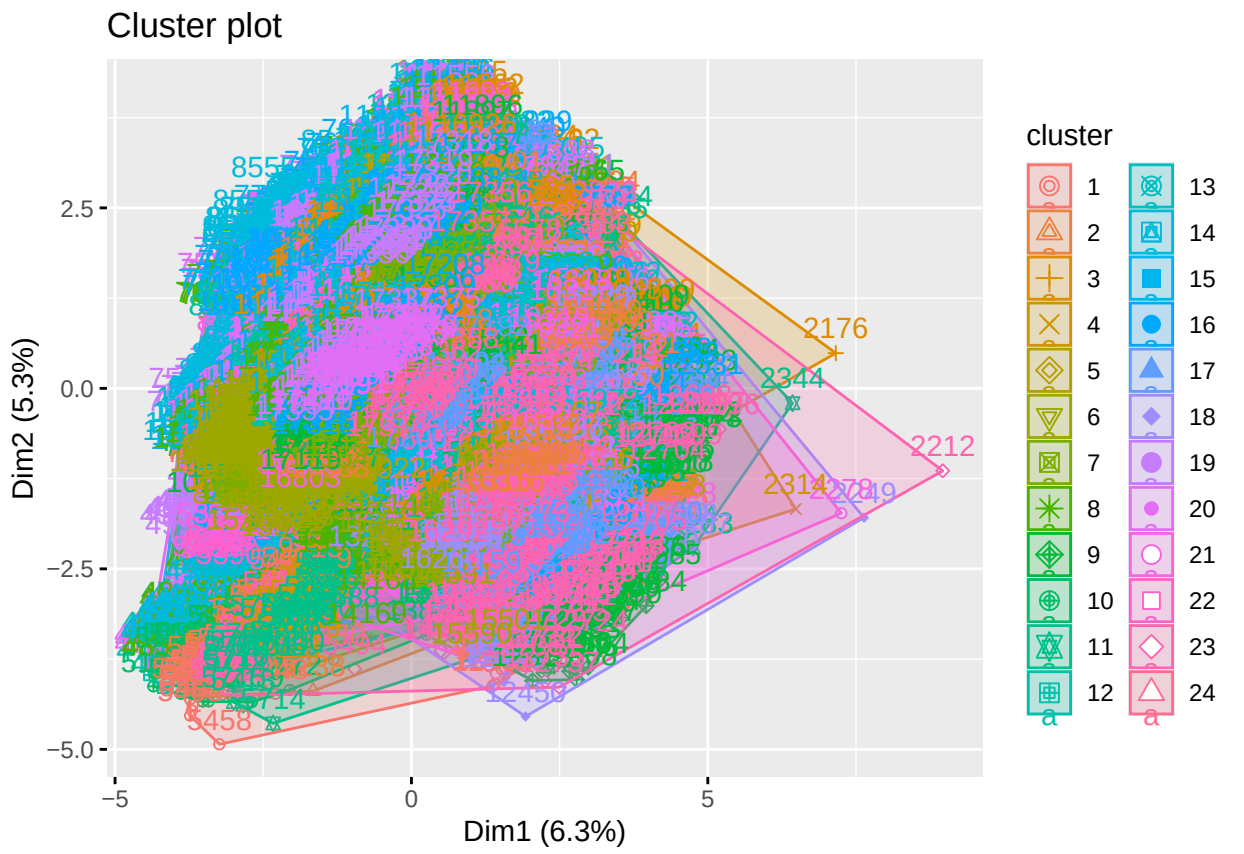
```
cluster_result %>% group_by(Kmeans) %>% select(Kmeans, Event) %>% table()
```

```
##       Event
## Kmeans Abu Dhabi Grand Prix Australian Grand Prix Austrian Grand Prix Azerbaijan Grand Prix Bahrain G
##      1                    0                   36                   51                    44
##      2                   39                   34                   54                    41
##      3                   36                   31                   54                     0
##      4                    0                    0                   58                     0
##      5                   38                   39                   58                    36
##      6                    0                    0                    0                     0
##      7                   35                    0                    0                     0
##      8                   39                   38                   61                    41
##      9                   42                   32                   57                    82
##      10                  39                   26                   51                     8
##      11                  38                   39                   55                    40
##      12                  42                   35                   56                    41
##      13                   0                   36                   58                    43
##      14                  37                   41                   46                    41
##      15                  42                    2                   51                    38
##      16                  71                   10                   50                    41
##      17                  16                    0                    0                    39
##      18                   0                   34                   61                    44
##      19                  38                   36                   58                    41
##      20                  37                   41                   50                    42
##      21                  37                   36                   54                    44
##      22                  42                   43                   54                    42
##      23                  19                   38                   59                    42
##      24                   0                    0                    0                     0
##       Event
## Kmeans Hungarian Grand Prix Italian Grand Prix Japanese Grand Prix Las Vegas Grand Prix Mexico City G
##      1                   39                 34                  0                    0
##      2                   32                 31                  0                   19
##      3                   36                 33                 36                   41
##      4                   42                  0                 30                    0
##      5                   43                 29                 33                   33
##      6                    0                  0                  0                    0
##      7                    0                  0                  0                   34
##      8                   32                 27                 38                   38
```

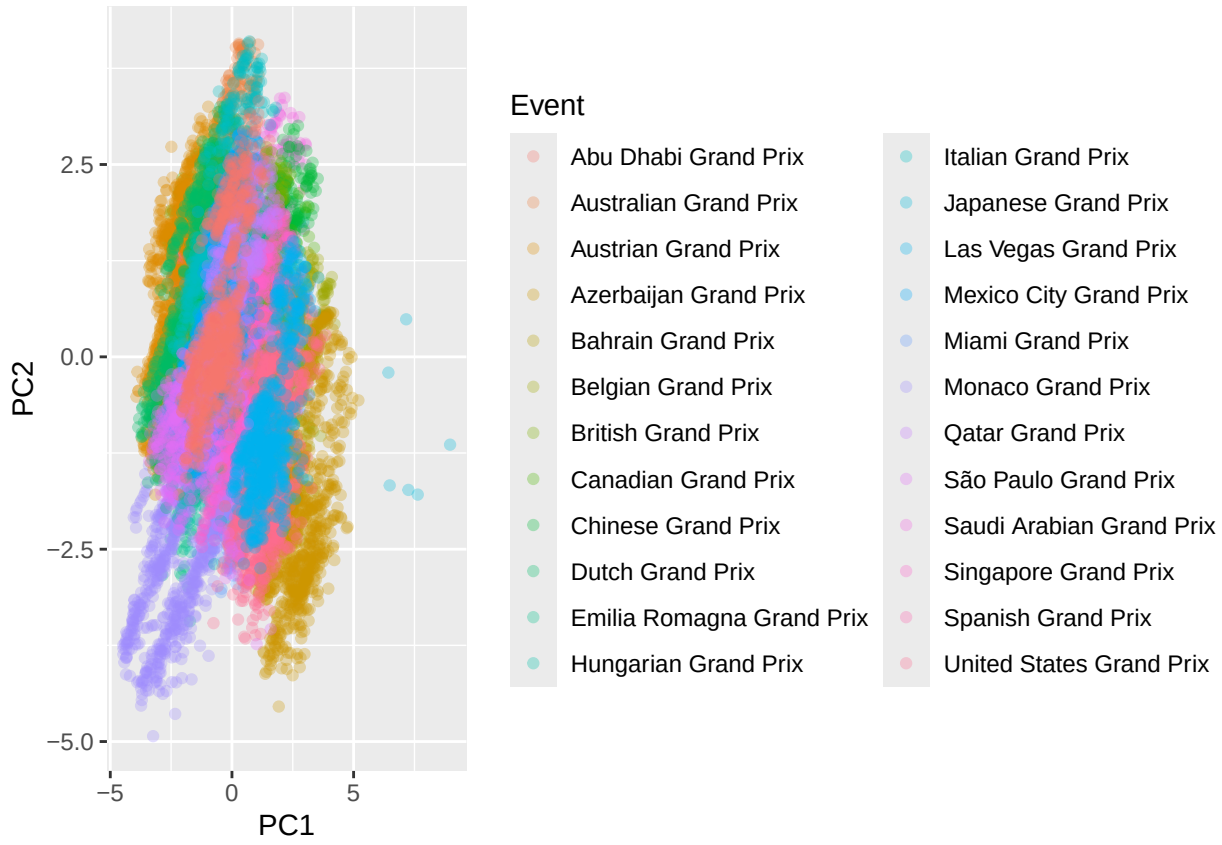
```
##      9                    41                   31                   36                   35
##      10                   28                    4                   25                   36
##      11                   35                   30                    6                   36
##      12                   30                   34                   39                   36
##      13                   40                   35                   36                   38
##      14                   35                   39                   32                   35
##      15                   30                   25                   18                   38
##      16                   45                   35                   34                   36
##      17                    0                   33                    0                   38
##      18                   31                   27                   29                   34
##      19                   44                   39                   33                   33
##      20                   33                   28                   26                   35
##      21                   14                   33                   36                    6
##      22                   42                   37                   36                   35
##      23                   42                   38                   37                   37
##      24                    0                    0                    0                    0
##        Event
## Kmeans Spanish Grand Prix United States Grand Prix
##      1                   49                      0
##      2                   41                     40
##      3                   51                     40
##      4                   38                      0
##      5                   46                     32
##      6                    0                      0
##      7                    0                     43
##      8                   49                     39
##      9                   50                     42
##      10                  47                     42
##      11                  46                     42
##      12                  44                     41
##      13                  40                     44
##      14                  52                     36
##      15                  43                     34
##      16                  47                      0
##      17                   0                     41
##      18                  43                     37
##      19                  47                     42
##      20                  48                     39
##      21                  39                     39
##      22                  53                     40
##      23                  48                     43
##      24                   0                      0
```

The only event kmeans clustering able to predict with any real accuracy was the São Paulo Grand Prix. Out of 599 laps, 541 were accurately identified as being at the São Paulo Grand Prix. Other than that, all other events don't have any meaningful result. Perhaps a more interesting test would be to cluster laps and compare them to see which events have similar lap data.

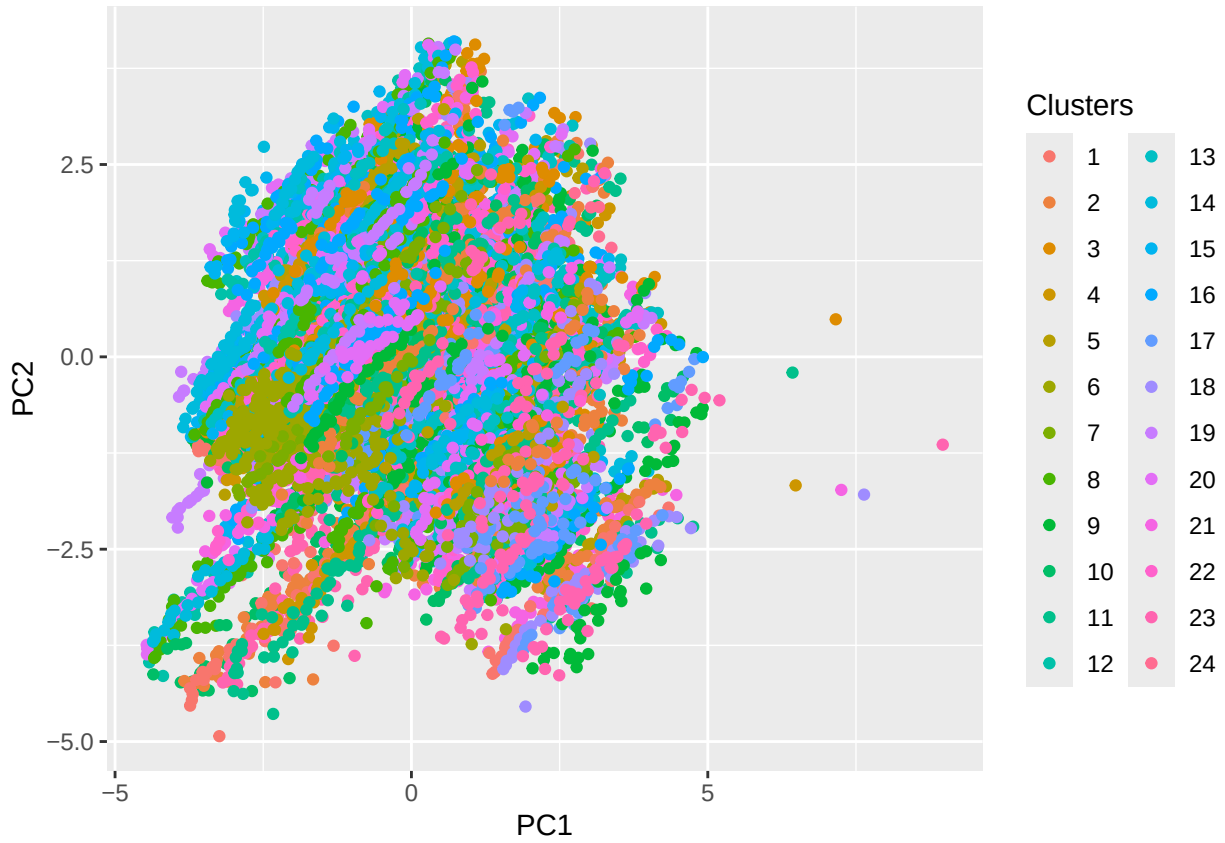
```
library(factoextra)
fviz_cluster(fit, data = predictors)
```




```
pca = prcomp(predictors)
rotated_data = as.data.frame(pca$x)
rotated_data$Event <- f1$Event
ggplot(data = rotated_data, aes(x = PC1, y = PC2, col = Event)) + geom_point(alpha = 0.3)
```




```
rotated_data$Clusters = as.factor(fit$cluster)
ggplot(data = rotated_data, aes(x = PC1, y = PC2, col = Clusters)) + geom_point()
```



The cluster plots show that the model was unable to distinguish between the events.

For classification, we will try to predict the IsPersonalBest column, which denotes that driver's personal best lap time up until that point in the race. Dummy variables for Event can now be created and used for this clasification.

```
table(f1$Event)
```

```
##
##      Abu Dhabi Grand Prix     Australian Grand Prix       Austrian Grand Prix     Azerbaijan Grand Pr
##                       687                       627                      1096
##          Dutch Grand Prix Emilia Romagna Grand Prix      Hungarian Grand Prix        Italian Grand Pr
##                       909                       967                       714
##          Qatar Grand Prix      São Paulo Grand Prix  Saudi Arabian Grand Prix      Singapore Grand Pr
##                       402                       599                       564
```

```
f1$AbuDhabi <- ifelse(f1$Event == 'Abu Dhabi Grand Prix',1,0)
f1$Australian <- ifelse(f1$Event == 'Australian Grand Prix',1,0)
f1$Austrian <- ifelse(f1$Event == 'Austrian Grand Prix',1,0)
f1$Azerbaijan <- ifelse(f1$Event == 'Azerbaijan Grand Prix',1,0)
f1$Bahrain <- ifelse(f1$Event == 'Bahrain Grand Prix',1,0)
f1$Belgian <- ifelse(f1$Event == 'Belgian Grand Prix',1,0)
f1$British <- ifelse(f1$Event == 'British Grand Prix',1,0)
f1$Canadian <- ifelse(f1$Event == 'Canadian Grand Prix',1,0)
f1$Chinese <- ifelse(f1$Event == 'Chinese Grand Prix',1,0)
f1$Dutch <- ifelse(f1$Event == 'Dutch Grand Prix',1,0)
f1$EmiliaRomagna <- ifelse(f1$Event == 'Emilia Romagna Grand Prix',1,0)
```

```
f1$Hungarian <- ifelse(f1$Event == 'Hungarian Grand Prix',1,0)
f1$Italian <- ifelse(f1$Event == 'Italian Grand Prix',1,0)
f1$Japanese <- ifelse(f1$Event == 'Japanese Grand Prix',1,0)
f1$LasVegas <- ifelse(f1$Event == 'Las Vegas Grand Prix',1,0)
f1$MexicoCity <- ifelse(f1$Event == 'Mexico City Grand Prix',1,0)
f1$Miami <- ifelse(f1$Event == 'Miami Grand Prix',1,0)
f1$Monaco <- ifelse(f1$Event == 'Monaco Grand Prix',1,0)
f1$Qatar <- ifelse(f1$Event == 'Qatar Grand Prix',1,0)
f1$SaoPaulo <- ifelse(f1$Event == 'São Paulo Grand Prix',1,0)
f1$SaudiArabian <- ifelse(f1$Event == 'Saudi Arabian Grand Prix',1,0)
f1$Singapore <- ifelse(f1$Event == 'Singapore Grand Prix',1,0)
f1$Spanish <- ifelse(f1$Event == 'Spanish Grand Prix',1,0)
f1$UnitedStates <- ifelse(f1$Event == 'United States Grand Prix',1,0)
```

Verify dummies created correctly

```
table(f1$AbuDhabi)
```

```
##
##     0     1
## 16724   687
```

```
table(f1$Australian)
```

```
##
##     0     1
## 16784   627
```

```
table(f1$Austrian)
```

```
##
##     0     1
## 16315  1096
```

```
table(f1$Azerbaijan)
```

```
##
##     0     1
## 16621   790
```

```
table(f1$Bahrain)
```

```
##
##     0     1
## 16813   598
```

```
table(f1$Belgian)
```

```
##
##     0     1
## 16941   470
```

```r
table(f1$British)
```

```
##
##     0     1
## 16700   711
```

```r
table(f1$Canadian)
```

```
##
##     0     1
## 16584   827
```

```r
table(f1$Chinese)
```

```
##
##     0     1
## 16842   569
```

```r
table(f1$Dutch)
```

```
##
##     0     1
## 16502   909
```

```r
table(f1$EmiliaRomagna)
```

```
##
##     0     1
## 16444   967
```

```r
table(f1$Hungarian)
```

```
##
##     0     1
## 16697   714
```

```r
table(f1$Italian)
```

```
##
##     0     1
## 16789   622
```

```r
table(f1$Japanese)
```

```
##
##     0     1
## 16851   560
```

```r
table(f1$LasVegas)
```

```
##
##     0     1
## 16738   673
```

```r
table(f1$MexicoCity)
```

```
##
##     0     1
## 16654   757
```

```r
table(f1$Miami)
```

```
##
##     0     1
## 16648   763
```

```r
table(f1$Monaco)
```

```
##
##     0     1
## 16340  1071
```

```r
table(f1$Qatar)
```

```
##
##     0     1
## 17009   402
```

```r
table(f1$SaoPaulo)
```

```
##
##     0     1
## 16812   599
```

```r
table(f1$SaudiArabian)
```

```
##
##     0     1
## 16847   564
```

```r
table(f1$Singapore)
```

```
##
##     0     1
## 16653   758
```

```r
table(f1$Spanish)
```

```
##
##     0     1
## 16490   921
```

```r
table(f1$UnitedStates)
```

```
##
##     0     1
## 16655   756
```

```r
head(f1)
```

```
## # A tibble: 6 x 66
##   LapTime LapNumber Sector1Time Sector2Time Sector3Time SpeedI1 SpeedI2 SpeedFL SpeedST IsPersonalBes
##     <dbl>     <dbl>       <dbl>       <dbl>       <dbl>   <dbl>   <dbl>   <dbl>   <dbl>           <db
## 1    96.3         2        30.9        41.7        23.7     232     248     276     287
## 2    96.8         3        31.0        42.0        23.8     231     243     276     290
## 3    97.2         5        31.3        42.1        23.9     231     245     276     289
## 4    97.0         7        31.0        42.1        23.9     232     242     277     291
## 5    97.2        16        31.0        42.0        24.1     233     260     277     292
## 6    95.3        19        30.6        41.2        23.5     233     248     278     288
## # i 43 more variables: Driver_LEC <int>, Driver_MAG <int>, Driver_NOR <int>, Driver_OCO <int>, Drive
## #   Driver_TSU <int>, Driver_VER <int>, Driver_ZHO <int>, Compound_HARD <int>, Compound_INTERMEDIATE
## #   Azerbaijan <dbl>, Bahrain <dbl>, Belgian <dbl>, British <dbl>, Canadian <dbl>, Chinese <dbl>, Du
## #   Monaco <dbl>, Qatar <dbl>, SaoPaulo <dbl>, SaudiArabian <dbl>, Singapore <dbl>, Spanish <dbl>, U
```

Split data into 75/25 train/test

```r
f1$IsPersonalBest <- as.factor(f1$IsPersonalBest)
index = createDataPartition(y=f1$IsPersonalBest, p=0.75, list=FALSE)
train_set = f1[index,]
test_set = f1[-index,]
```

Run SVM classifier

```r
svm_split <- train(IsPersonalBest ~., data = train_set, method = "svmLinear")
pred_split <- predict(svm_split, test_set)
sum(pred_split == test_set$IsPersonalBest) / nrow(test_set)
```

```
## [1] 0.7679228
```

```r
confusionMatrix(test_set$IsPersonalBest, pred_split)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
```

```
##          0 3219   85
##          1  925  123
##
##                Accuracy : 0.7679
##                  95% CI : (0.7551, 0.7804)
##     No Information Rate : 0.9522
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1262
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7768
##             Specificity : 0.5913
##          Pos Pred Value : 0.9743
##          Neg Pred Value : 0.1174
##              Prevalence : 0.9522
##          Detection Rate : 0.7397
##    Detection Prevalence : 0.7592
##       Balanced Accuracy : 0.6841
##
##        'Positive' Class : 0
##
```

Run decision tree classifier

```
tree <- train(IsPersonalBest ~., data = train_set, method = "rpart")
tree
```

```
## CART
##
## 13059 samples
##    65 predictor
##     2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 13059, 13059, 13059, 13059, 13059, 13059, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.01874801  0.8018760  0.3122796
##   0.01938354  0.8010196  0.3057976
##   0.13441373  0.7678010  0.1021161
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.01874801.
```

```
pred_split <- predict(tree, test_set)
cm <- confusionMatrix(test_set$IsPersonalBest, pred_split)
cm
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction    0    1
##          0 3274   30
##          1  834  214
##
##                Accuracy : 0.8015
##                  95% CI : (0.7893, 0.8132)
##     No Information Rate : 0.9439
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2644
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7970
##             Specificity : 0.8770
##          Pos Pred Value : 0.9909
##          Neg Pred Value : 0.2042
##              Prevalence : 0.9439
##          Detection Rate : 0.7523
##    Detection Prevalence : 0.7592
##       Balanced Accuracy : 0.8370
##
##        'Positive' Class : 0
##
```

SVM Accuracy = 76.79% Tree Accuracy = 80.15%

The decision tree model performed better than SVM. Further evaluation will use the tree model.

```
precision <- 214/(30+214)
precision
```

```
## [1] 0.8770492
```

```
recall <- 214/(834+214)
recall
```

```
## [1] 0.2041985
```

```
library(pROC)
pred_prob <- predict(tree, test_set, type = 'prob')
head(pred_prob)
```

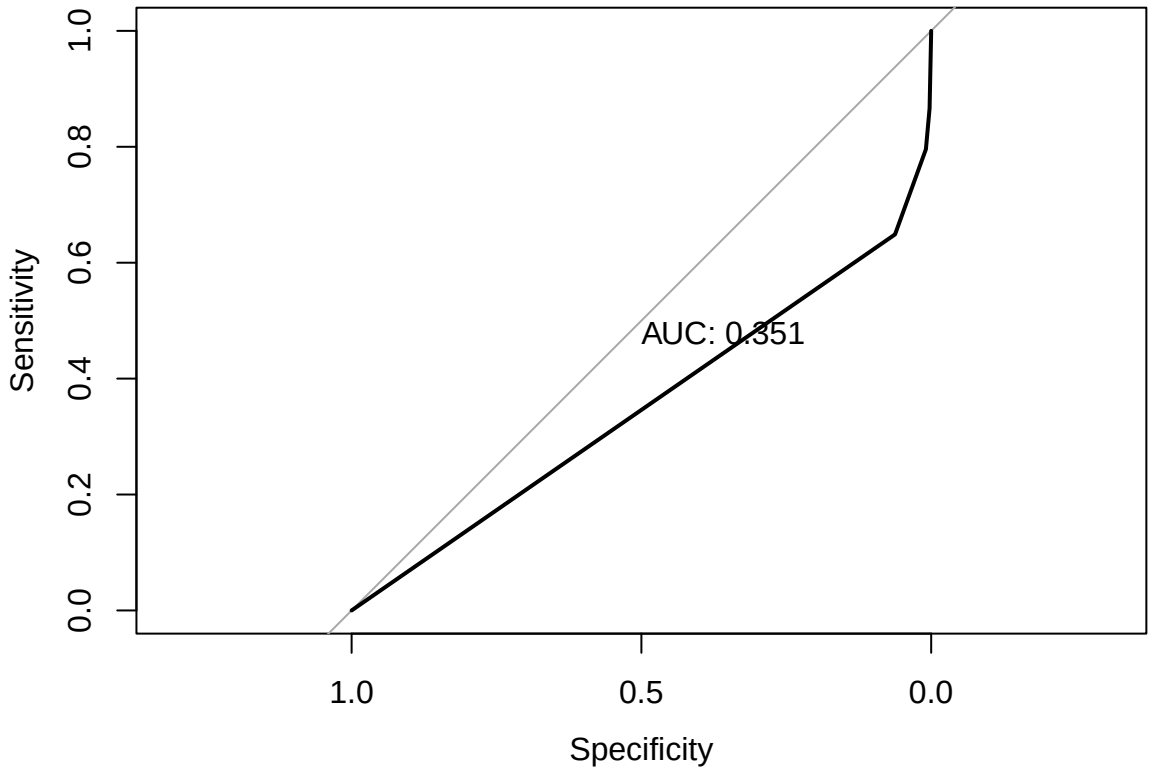
```
##           0         1
## 1 0.8178281 0.1821719
## 2 0.8178281 0.1821719
## 3 0.8178281 0.1821719
## 4 0.8178281 0.1821719
## 5 0.5651709 0.4348291
## 6 0.8178281 0.1821719
```

```
roc_obj <- roc((test_set$IsPersonalBest), pred_prob[,1])
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(roc_obj, print.auc=TRUE)
```



The precision of the model is fairly high at 87.7% but the model struggles with recall at only 20.42%. The accuracy is high because of the imbalance in the data, whereby classifying each lap as not being a personal best it can achieve good results. The model's struggles are further seen in the ROC plot with a poor AUC of 0.351. This shows that there was really no connection found between a personal best lap and the independent variables, so the model is essentially guessing when it predicts one.

Overall, I feel like I could have chosen better targets for classification and clustering. Since Formula One circuits are all different from one another, I was hoping that their lap data would differentiate enough to cluster them effectively, but that was not the case. Interestingly though, there was one event, the São Paulo Grand Prix, that stood out from the rest but I'll have to do some more research to figure out why that was the case.

This course has allowed me to gain a better knowledge of techniques and strategies to gain meaningful insight from the data mining process. Having a plan of action to clean and process relevant data and knowing what type of models are most effective at finding useful information from that data is the core of data science. I feel more equipped to evaluate the outcomes of my models and know how the process can be changed to achieve a better result. Within each model themselves there is also room for improvement. Tuning for certain evaluation metrics or interpretability are essential components when building a model. This

requires knowledge of how each model technique operates and also creativity as a data scientist to adapt your application process based on the problem presented to you. There are no simple and easy fixes to these issues and even if an improved solution is found it may come at a cost of efficiency, making it less desirable. Ultimately, functionality is the most important aspect of data science and speaking with experts in the field of research you are working in will give direction to your process to achieve a useful result.