Formula 1 Project

Neo Kok

2023-11-01

Contents

Project Outline

Data

The data used for this project include relational datasets with information about Formula 1 circuits, drivers, constructors, and races. It was downloaded from this Kaggle source. Some data goes as far back as to the inaugural race in 1950 and the most recent is as new as 7/30/2023.

Goals

1. What effect has changing the point system had?

2. How has competitiveness changed over time?

a) Driver standings
b) Constructor standings
c) Teammates

3. Predicting race results

a) Simple linear models
b) Logistic regression model

Motivation

I have been a big Formula 1 fan for most of my life and have always wanted to do a project on Formula 1 data. This challenge provided me the opportunity to showcase my skills while working with data that is fun to work with and is meaningful to me.

Setting Up

Import Libraries

library(tidyverse)
library(car)
library(glmnet)
library(caret)
library(knitr)

Import Data

```
driver_standings <- read.csv("driver_standings.csv")
drivers <- read.csv("drivers.csv")
lap_times <- read.csv("lap_times.csv")
pit_stops <- read.csv("pit_stops.csv")
races <- read.csv("races.csv")
results <- read.csv("results.csv")
```

1. What effect has changing the point system had?

Scaling up all results to the modern system

```
adjust_points_up = function(positionOrder, rank){
  rank = as.numeric(rank)
  points <- c(25, 18, 15, 12, 10, 8, 6, 4, 2, 1) # Points for 1st to 10th position

  if(positionOrder <= 10 & rank == 1 & !is.na(rank) & !is.na(positionOrder)){
    points[positionOrder] = points[positionOrder] + 1
  }
  if(positionOrder > 10 | is.na(positionOrder)){
    return(0)
  }
  return(points[positionOrder])
}

results$adjusted_points_up = mapply(adjust_points_up, results$positionOrder, results$rank)
```

Scaling down all results to original 1950s point system

```
adjust_points_down = function(positionOrder, rank){
  rank = as.numeric(rank)
  points <- c(8, 6, 4, 3, 2) # Points for 1st to 5th position

  if(positionOrder <= 5 & rank == 1 & !is.na(rank) & !is.na(positionOrder)){
    points[positionOrder] = points[positionOrder] + 1
  }
  if(positionOrder > 5 | is.na(positionOrder)){
    return(0)
  }
  return(points[positionOrder])
}

results$adjusted_points_down = mapply(adjust_points_down, results$positionOrder, results$rank)
```

Re-calculated leaderboards

```
# Create podium and join year variables
results <- results %>% mutate(podium = ifelse(as.numeric(positionOrder) > 3, 0, 1),
                              win = ifelse(as.numeric(positionOrder) == 1, 1, 0))
results <- races %>% select(raceId, year) %>% left_join(results)
```

```
## Joining with `by = join_by(raceId)`
```

```
head(results, 10) %>% kable()
```

raceId	year	resultId	driverId	constructorId	number	grid	position	positionText	positionOrder	points	laps	time	milliseconds	fastestLap	rank	fastestLapTime	fastestLapSpeed	statusId	podium	win	adjusted_points	adjusted_points_down
1	2009	755418	23	22	1	1	1	1	1	10	58	1:34:[illegible]	[illegible]	3	1:28.0[illegible]	206.891	1	25	8	1	1	1
1	2009	755522	23	23	2	2	2	2	2	8	58	+0.8[illegible]	[illegible]	14	1:29.0[illegible]	204.344	1	18	6	1	0	1
1	2009	755615	7	9	20	3	3	3	3	6	58	+1.6[illegible]	[illegible]	10	1:28.9[illegible]	204.706	1	15	4	1	0	0

raceId	year	resultId	driverId	constructorId	number	grid	position	positionText	positionOrder	points	laps	time	milliseconds	fastestLap	rank	fastestLapTime	fastestLapSpeed	statusId	adjusted_points	adjusted_podium	points_down	podium_down	win	adjusted_points_down
1	2009	7557	10	7	10	19	4	4	4	5	58	+4.435	5660213	53	6	1:28.426	165.920	1	12	3	0	0	0	0
1	2009	7558	4	4	7	10	5	5	5	4	58	+4.879	5660663	53	9	1:28.721	165.199	1	10	2	0	0	0	0
1	2009	7559	3	3	16	5	6	6	6	3	58	+5.723	5661506	46	1	1:27.706	167.668	1	9	0	0	0	0	0
1	2009	7560	67	5	12	13	7	7	7	2	58	+6.005	5661738	48	16	1:29.230	163.950	1	6	0	0	0	0	0
1	2009	7561	7	5	11	17	8	8	8	1	58	+6.298	5662082	52	17	1:29.823	162.537	1	4	0	0	0	0	0
1	2009	7562	16	10	20	16	9	9	9	0	58	+6.336	5662143	49	11	1:28.924	164.640	1	2	0	0	0	0	0
1	2009	7563	2	2	6	9	10	10	10	0	58	+7.085	5662868	49	5	1:28.283	166.245	1	1	0	0	0	0	0

```
#Leaderboards with modern scoring
scores_adjusted_up = results %>% group_by(driverId) %>%
  summarise(points = sum(adjusted_points_up), podiums = sum(podium), wins = sum(win)) %>%
  arrange(-points)
head(scores_adjusted_up, 10) %>% kable()
```

driverId	points	podiums	wins
1	4940	195	103
30	3910	155	91
20	3325	122	53
4	3064	104	32
8	2831	103	21
117	2486	106	51
830	2292	89	45
22	1906	68	11
102	1885	80	41
18	1859	50	15

```
#Top 50 with modern scoring
leaders_adjusted_up = scores_adjusted_up %>% top_n(50, points) %>%
  left_join(drivers, by = c("driverId")) %>% select(forename, surname,
                                                    points, podiums, wins) %>%
  arrange(-points)
head(leaders_adjusted_up, 10) %>% kable()
```

forename	surname	points	podiums	wins
Lewis	Hamilton	4940	195	103
Michael	Schumacher	3910	155	91
Sebastian	Vettel	3325	122	53
Fernando	Alonso	3064	104	32
Kimi	Räikkönen	2831	103	21
Alain	Prost	2486	106	51
Max	Verstappen	2292	89	45
Rubens	Barrichello	1906	68	11
Ayrton	Senna	1885	80	41
Jenson	Button	1859	50	15

```
#Leaderboards with original scoring
scores_adjusted_down = results %>% group_by(driverId) %>%
```

```
  summarise(points = sum(adjusted_points_down), podiums = sum(podium),
            wins = sum(win)) %>%  arrange(-points)
head(scores_adjusted_down, 10) %>% kable()
```

driverId	points	podiums	wins
1	1494	195	103
30	1158	155	91
20	952	122	53
4	779	104	32
117	738	106	51
8	735	103	21
830	683	89	45
102	563	80	41
22	475	68	11
822	468	67	10

```
#Top 50 with original scoring
leaders_adjusted_down = scores_adjusted_down %>% top_n(50, points) %>%
  left_join(drivers, by = c("driverId")) %>%
  select(forename, surname, points, podiums, wins) %>% arrange(-points)
head(leaders_adjusted_down, 10) %>% kable()
```

forename	surname	points	podiums	wins
Lewis	Hamilton	1494	195	103
Michael	Schumacher	1158	155	91
Sebastian	Vettel	952	122	53
Fernando	Alonso	779	104	32
Alain	Prost	738	106	51
Kimi	Räikkönen	735	103	21
Max	Verstappen	683	89	45
Ayrton	Senna	563	80	41
Rubens	Barrichello	475	68	11
Valtteri	Bottas	468	67	10

Difference in leaderboard

```
#Included in top 50
different = 0
for(i in 1:nrow(leaders_adjusted_down)){
  if(!(leaders_adjusted_down$surname[i] %in% leaders_adjusted_up$surname)){
    different = different + 1
  }
}

different / nrow(leaders_adjusted_down)
```

```
## [1] 0.1
```

```r
#Exactly the same position
different = 0
for(i in 1:nrow(leaders_adjusted_down)){
  if(!(leaders_adjusted_down$surname[i] == leaders_adjusted_up$surname[i])){
    different = different + 1
  }
}

different / nrow(leaders_adjusted_down)
```

```
## [1] 0.84
```

10% of the top 50 drivers would not be in the top 50 if the point system stayed the same from 1950 to today. 84% of the top 50 drivers would be in a different position in the leaderboards if the point system stayed the same from 1950 to today.

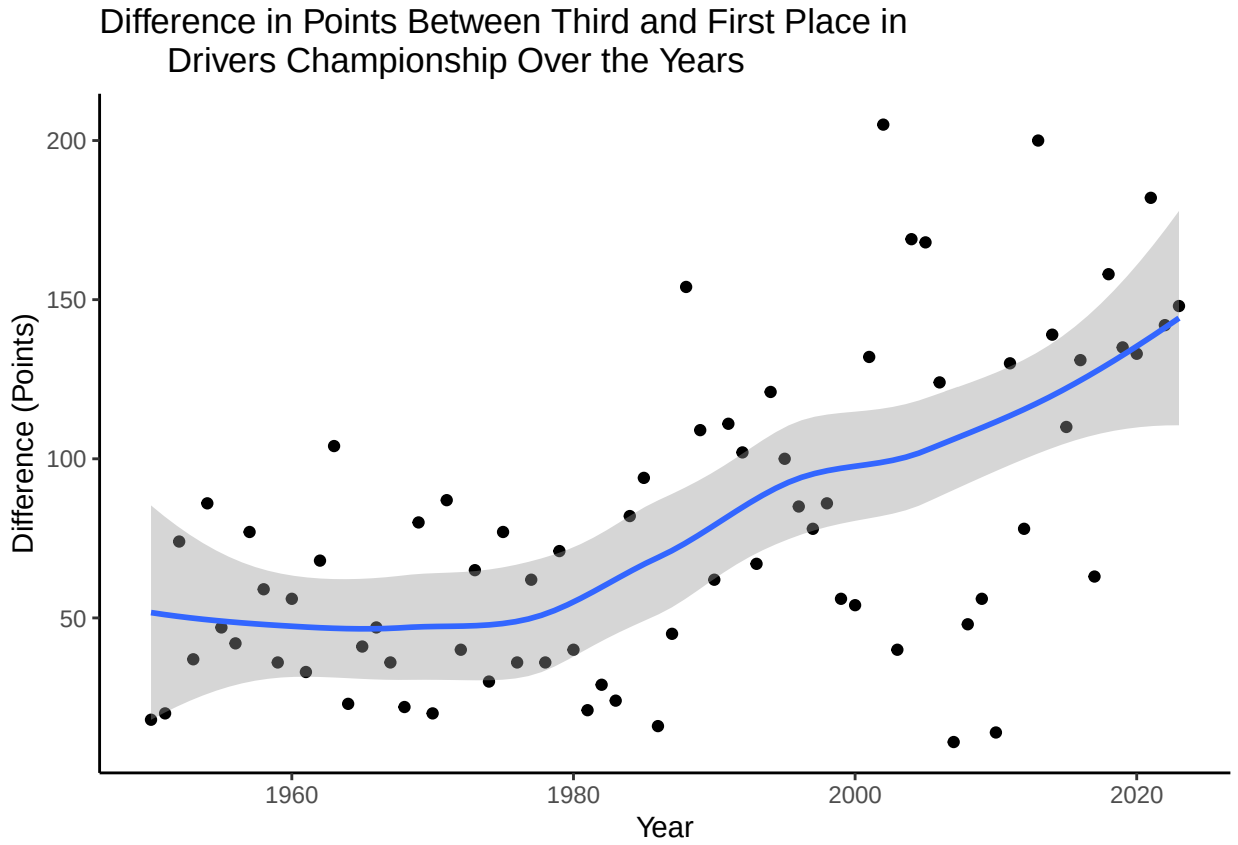
2. How has competitiveness changed over time?

For the rest of this analysis, comparisons will be made using points adjusted to the modern scoring system to keep things consistent and serve to create a fair comparison.

a) Driver standings

Comparing difference in final points for the top three drivers in the drivers standings - how competitive the drivers championship is.

```r
results %>% group_by(driverId, year) %>% summarise(wins = sum(positionOrder == 1),
                                                   points = sum(adjusted_points_up)) %>%
  arrange(-year, -points) %>% ungroup() %>% group_by(year) %>%
  summarise(difference = points[1] - points[3]) %>%
  ggplot(aes(x = year, y = difference)) + geom_point() + geom_smooth()+ theme_classic() +
  labs(title = "Difference in Points Between Third and First Place in
       Drivers Championship Over the Years",
       x = "Year", y = "Difference (Points)")
```

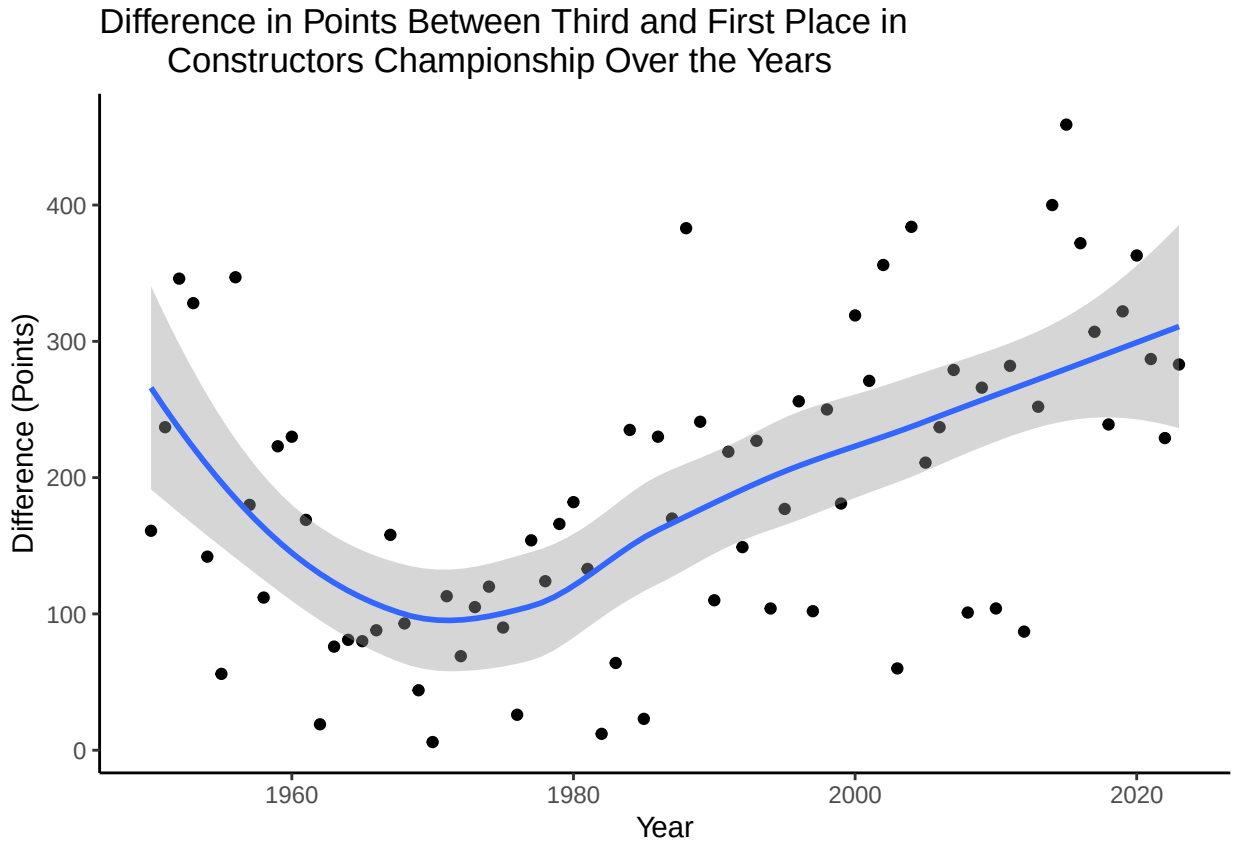


Scatterplot shows a relatively consistent increase in difference in points for the top three, indicating a decrease in competitiveness in the drivers championship over the years. Notable outliers in the 2007 and 2010 seasons. Smoothed LOESS line included with shaded standard errors.

b) Constructors standings

Comparing difference in final points for the top three constructors (teams) in the constructors standings - how competitive the constructors championship is.

```
results %>% group_by(constructorId, year) %>%
  summarise(wins = sum(positionOrder == 1), points = sum(adjusted_points_up)) %>%
  arrange(-year, -points) %>% ungroup() %>% group_by(year) %>%
  summarise(difference = points[1] - points[3]) %>%
  ggplot(aes(x = year, y = difference)) + geom_point() + geom_smooth()+theme_classic() +
  labs(title = "Difference in Points Between Third and First Place in
       Constructors Championship Over the Years",
       x = "Year", y = "Difference (Points)")
```

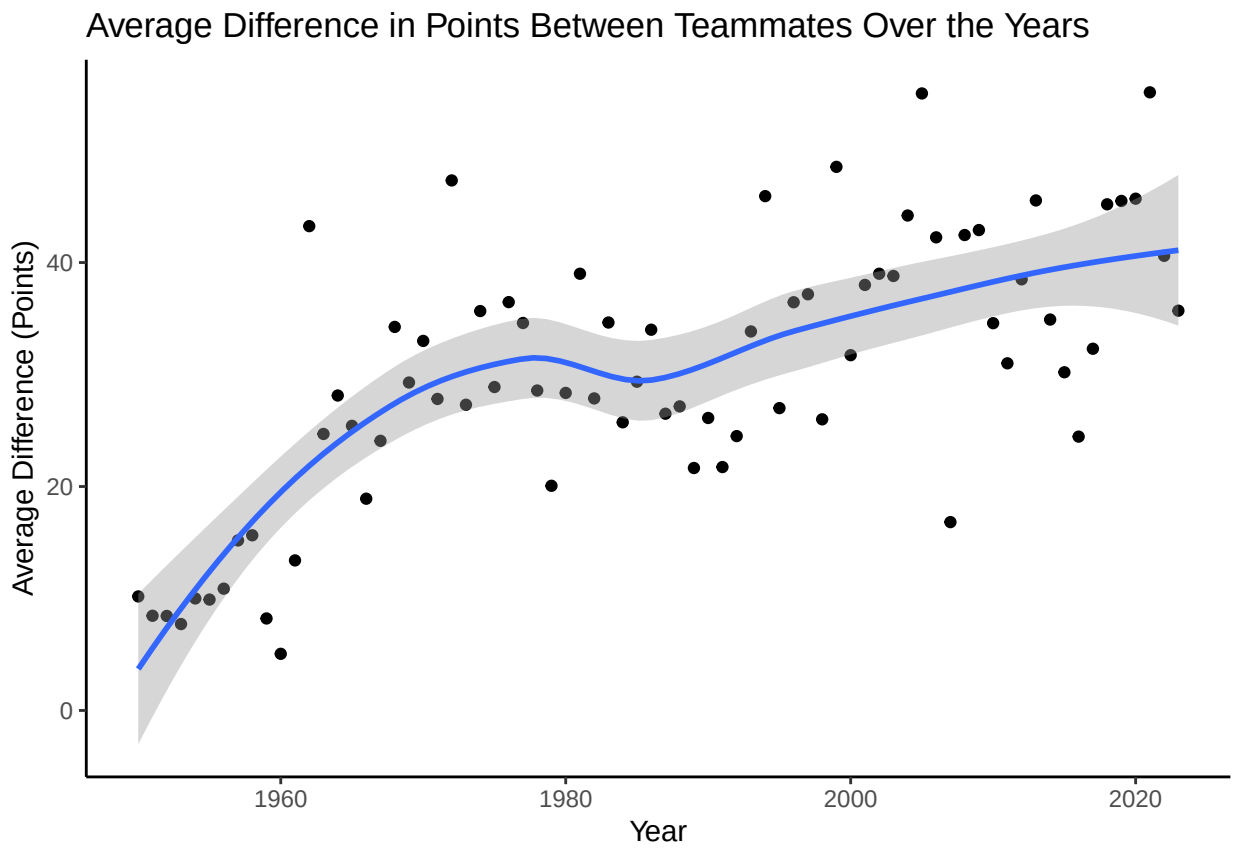


Scatterplot shows an initial increase in competitiveness, which has decreased relatively consistently since the 1970s. Smoothed LOESS line included with shaded standard errors.

c) Teammates

Comparing average difference in final points between teammates in the drivers standings - how competitive the teammate battle is. Most constructors only have 2 drivers, although some may have more or less depending on mid-season seat changes/single driver teams.

```
results %>% group_by(driverId, year, constructorId) %>%
  summarise(wins = sum(positionOrder == 1), points = sum(adjusted_points_up)) %>%
  arrange(-year, -points) %>% ungroup() %>%
  group_by(constructorId, year) %>% summarise(difference = points[1] - points[2]) %>%
  ungroup()%>% group_by(year) %>%
  summarise(average_diff = mean(difference, na.rm = T)) %>%
  ggplot(aes(x = year, y = average_diff)) + geom_point() + geom_smooth() +theme_classic() +
  labs(title = "Average Difference in Points Between Teammates Over the Years",
       x = "Year", y = "Average Difference (Points)")
```



Scatterplot shows an general decrease in competitiveness over time, with a period between 1975-1990 where it stayed quite consistent and followed with a shallower decrease. Smoothed LOESS line included with shaded standard errors.

3. Predicting race results

a) Simple linear models

Data preparation

Selecting variables of interest and manipulating data to create a table with the variables. Response variables of interest include: winner/not winner, number of points won, finishing position. Predictor variables of interest include: starting qualifying position, number of pit stops in race, average pit stop time in race, drivers championship standings.

```
# Creating number of pit stops and average pit stop duration variable
stops = pit_stops %>% group_by(raceId, driverId) %>%
  summarise(stops = max(stop), avg_time = mean(as.numeric(duration), na.rm = T))
```

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

```
# Selecting race & driver ID keys, qualification position, finishing position,
# fastest lap ranking, and number of points won (adjusted to modern scoring).
```

```r
# Creating "wins" variable where finishing position = 1.
# Joined with pit stop data by race and driver ID keys.
# Joined with driver standings data to select drivers standings before race start
total = results %>% select(raceId, driverId, grid, positionOrder, adjusted_points_up, rank) %>%
  filter(rank != "\\N", rank != "0") %>% left_join(stops, by = c("raceId", "driverId")) %>%
  left_join(driver_standings, by = c("raceId", "driverId")) %>%
  select(finish = positionOrder, points = adjusted_points_up, quali = grid, stops,
         stop_time = avg_time, driver_standing = position, fastest_lap = rank) %>%
  mutate(win = ifelse(finish == 1, 1, 0))

# Removing NA data - primarily all of the data before 1996 when pit stop times
# were not tracked.
total = total[complete.cases(total),]
# Ensuring all variables are numeric
total = lapply(total, as.numeric)
total = data.frame(total)
# Altering "win" variable as a factor - win or no win
total$win = as.factor(total$win)
head(total, 10) %>% kable()
```

finish	points	quali	stops	stop_time	driver_standing	fastest_lap	win
1	25	1	2	23.31950	1	4	1
2	18	2	2	23.21300	2	8	0
3	15	6	2	25.10900	3	7	0
4	12	5	3	24.05500	4	2	0
5	10	3	3	24.05867	5	3	0
6	8	4	3	20.95033	6	5	0
7	7	8	3	24.14567	7	1	0
8	4	10	2	24.22100	8	11	0
9	2	16	2	24.92450	9	13	0
10	1	14	2	24.59750	10	14	0

Creating models

```r
# Predicting points:
# All predictor variables
lm_full = lm(points ~ quali + stops + stop_time + driver_standing + fastest_lap, data = total)
summary(lm_full)
```

```
##
## Call:
## lm(formula = points ~ quali + stops + stop_time + driver_standing +
##     fastest_lap, data = total)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.4255  -3.0112  -0.3032   2.6729  17.6551
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)     18.82804    0.46807  40.225  < 2e-16 ***
## quali           -0.23068    0.01639 -14.077  < 2e-16 ***
## stops           -0.38532    0.06898  -5.586 2.46e-08 ***
## stop_time       -0.07800    0.01716  -4.545 5.62e-06 ***
## driver_standing -0.43606    0.01814 -24.038  < 2e-16 ***
## fastest_lap     -0.34717    0.01608 -21.591  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.647 on 4883 degrees of freedom
## Multiple R-squared:  0.5933, Adjusted R-squared:  0.5929
## F-statistic:  1425 on 5 and 4883 DF,  p-value: < 2.2e-16
```

```
brief(lm_full)
```

```
##             (Intercept)   quali  stops stop_time driver_standing fastest_lap
## Estimate         18.828 -0.2307 -0.385   -0.0780         -0.4361     -0.3472
## Std. Error        0.468  0.0164  0.069    0.0172          0.0181      0.0161
##
##  Residual SD = 4.65 on 4883 df, R-squared = 0.593
```

```
# Checking for multicollinearity
vif(lm_full)
```

```
##           quali           stops       stop_time driver_standing     fastest_lap
##        2.337061        1.045501        1.027856        2.962869        2.016168
```

```
# Excluding two least significant - average stop time, number of pit stops
lm_part = lm(points ~ quali + driver_standing + fastest_lap, data = total)
summary(lm_part)
```

```
##
## Call:
## lm(formula = points ~ quali + driver_standing + fastest_lap,
##     data = total)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.6941  -2.9985  -0.3024   2.6775  16.8679
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     16.18278    0.14898  108.62   <2e-16 ***
## quali           -0.23490    0.01645  -14.28   <2e-16 ***
## driver_standing -0.45028    0.01810  -24.88   <2e-16 ***
## fastest_lap     -0.33373    0.01590  -20.99   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.668 on 4885 degrees of freedom
## Multiple R-squared:  0.5894, Adjusted R-squared:  0.5891
## F-statistic:  2337 on 3 and 4885 DF,  p-value: < 2.2e-16
```

```r
brief(lm_part)
```

```
##             (Intercept)   quali driver_standing fastest_lap
## Estimate         16.183 -0.2349         -0.4503     -0.3337
## Std. Error        0.149  0.0164          0.0181      0.0159
##
##  Residual SD = 4.67 on 4885 df, R-squared = 0.589
```

```r
# Checking for multicollinearity
vif(lm_part)
```

```
##           quali driver_standing     fastest_lap
##        2.333325        2.922331        1.953438
```

```r
# Only variables known prior to race start
lm_before = lm(points ~ quali + driver_standing, data = total)
brief(lm_before)
```

```
##             (Intercept)  quali driver_standing
## Estimate         15.204 -0.285         -0.6278
## Std. Error        0.148  0.017          0.0167
##
##  Residual SD = 4.87 on 4886 df, R-squared = 0.552
```

```r
# Checking for multicollinearity
summary(lm_before)
```

```
##
## Call:
## lm(formula = points ~ quali + driver_standing, data = total)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.2916  -3.0994  -0.1437   2.7103  18.3541
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     15.20447    0.14773  102.92   <2e-16 ***
## quali           -0.28506    0.01699  -16.78   <2e-16 ***
## driver_standing -0.62781    0.01670  -37.58   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.874 on 4886 degrees of freedom
## Multiple R-squared:  0.5524, Adjusted R-squared:  0.5522
## F-statistic:  3014 on 2 and 4886 DF,  p-value: < 2.2e-16
```

```r
vif(lm_before)
```

```
##           quali driver_standing
##        2.284062        2.284062
```

Created three separate simple linear models. All have adjusted R^2 between 0.55 and 0.6. Checked variance inflation factors (VIF) to ensure no multicollinearity. All are below 5 which indicates low multicollinearity.

Comparing variable importance

```
anova(lm_part, lm_full)
```

```
## Analysis of Variance Table
##
## Model 1: points ~ quali + driver_standing + fastest_lap
## Model 2: points ~ quali + stops + stop_time + driver_standing + fastest_lap
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1   4885 106463
## 2   4883 105447  2    1016.6 23.538 6.704e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Since the p-value of the ANOVA between the full and reduced models is less than 0.05, we have convincing evidence that the addition of the number of pit stops and average pit stop time variables are important to increasing the model's accuracy.

b) Logistic regression model

We will create a logistic regression model to predict the winner of this weekend's (10/29/23) F1 race. The regression model will only include the qualification position and driver standings as predictors since those are the only ones that we know before the race start.

Creating and evaluating the model using train/test sets

```
set.seed(1)

#Creating test/training set
sample <- sample(c(TRUE, FALSE), nrow(total), replace=TRUE, prob=c(0.7,0.3))
train  <- total[sample, ]
test   <- total[!sample, ]

#Creating logistic regression model
LR_before <- glm(win ~ quali + driver_standing, data = train, family = binomial)
summary(LR_before)
```

```
##
## Call:
## glm(formula = win ~ quali + driver_standing, family = binomial,
##     data = train)
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)      1.54362    0.21705   7.112 1.15e-12 ***
## quali           -0.32645    0.05589  -5.841 5.18e-09 ***
## driver_standing -0.75269    0.08039  -9.363  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1326.4  on 3401  degrees of freedom
## Residual deviance:  630.7  on 3399  degrees of freedom
## AIC: 636.7
##
## Number of Fisher Scoring iterations: 10
```

```
#Predicting win/no win
train_predictions <- predict(LR_before, newdata = train, type = "response")
test_predictions <- predict(LR_before, newdata = test, type = "response")

#Evaluating accuracy
train_accuracy <- sum(ifelse(train_predictions > 0.5, 1, 0) == train$win) / nrow(train)
test_accuracy <- sum(ifelse(test_predictions > 0.5, 1, 0) == test$win) / nrow(test)
cat("In-sample accuracy:", train_accuracy)
```

```
## In-sample accuracy: 0.9600235
```

```
cat(" Out-of-sample accuracy:", test_accuracy)
```

```
##  Out-of-sample accuracy: 0.9650303
```

```
#Confusion matrix of out-of-sample accuracy of the model
test_confusion <- confusionMatrix(as.factor(ifelse(test_predictions > 0.5, 1, 0)), as.factor(test$win))
test_confusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1390   38
##          1   14   45
##
##                Accuracy : 0.965
##                  95% CI : (0.9544, 0.9738)
##     No Information Rate : 0.9442
##     P-Value [Acc > NIR] : 0.0001237
##
##                   Kappa : 0.616
##
##  Mcnemar's Test P-Value : 0.0014251
##
##             Sensitivity : 0.9900
##             Specificity : 0.5422
##          Pos Pred Value : 0.9734
##          Neg Pred Value : 0.7627
##              Prevalence : 0.9442
##          Detection Rate : 0.9348
##    Detection Prevalence : 0.9603
##       Balanced Accuracy : 0.7661
```

```
##
##        'Positive' Class : 0
##
```

The model has an accuracy of 96.5% at predicting whether or not a driver will win an F1 race. This seems exceptionally high although when looking at the confusion matrix, is quite misleading. Since there are far more "non-winners" than winners, it is a lot easier for the model to correctly predict "non-winner" and be correct. When looking at the specificity of 54%, it is evident that the model is not incredibly accurate at predicting the winner, although still performs relatively well.

Predicting this weekend's race

Using grid starting position, driver championship standings, and logistic regression model prediction for this weekend's F1 race to predict who will win and what place drivers will end up in.

```
set.seed(1)
#Driver names, grid starting position, driver championship standings, and model
#prediction for this weekend
real_data = data.frame(driver_name = c("Verstappen", "Perez", "Hamilton", "Sainz", "Leclerc", "Alonso",




                                       ),
                       quali = c(3, 5, 6, 2, 1, 14, 8, 7, 17, 11, 9, 15, 13, 20, 12, 10, 4, 18, 16, 19)
                       driver_standing = c(1, 2, 3, 5, 7, 4, 8, 9, 6, 10, 14, 12, 13, 11, 15, 17, 22, 1
real_data$predicted = predict(LR_before, newdata = real_data, type = "response")

# Sports betting odds rankings
# Scaled up the probability so that it equals 1
predicted_finish = real_data %>%
  mutate(odds = c(1, 3, 2, 6, 8, 9, 5, 7, 4, 11, 17, 10, 13, 14, 18, 16, 15, 12, 19, 20)) %>%
  arrange(-predicted) %>% mutate(predicted = predicted/sum(predicted), predicted_finish = rank(-predict
predicted_finish %>% rename(expert = odds) %>% kable()
```

driver_name	quali	driver_standing	predicted	expert	predicted_finish
Verstappen	3	1	0.5952393	1	1
Perez	5	2	0.2218148	3	2
Hamilton	6	3	0.0848482	2	3
Sainz	2	5	0.0703186	6	4
Leclerc	1	7	0.0224630	8	5
Alonso	14	4	0.0031298	9	6
Russel	8	8	0.0010947	5	7
Piastri	7	9	0.0007150	7	8
Norris	17	6	0.0002614	4	9
Gasly	11	10	0.0000913	11	10
Bottas	9	14	0.0000086	17	11
Ocon	15	12	0.0000055	10	12
Albon	13	13	0.0000050	13	13
Stroll	20	11	0.0000023	14	14
Hulkenberg	12	15	0.0000015	18	15
Zhou	10	17	0.0000007	16	16

driver_name	quali	driver_standing	predicted	expert	predicted_finish
Ricciardo	4	22	0.0000001	15	17
Tsunoda	18	16	0.0000001	12	18
Magnussen	16	18	0.0000000	19	19
Sargeant	19	20	0.0000000	20	20

The predicted winner is Max Verstappen with a 59% chance of winning.

Comparing to the true results

We will use the probability of winning a race using the model to rank their predicted positions. We will compare this to the sports betting probabilities prior to the start of the race to see how this simple model compares to expert's predictions.

```
# True results
predicted_finish$true = c(1, NA, 2, 4, 3, NA, 6, 8, 5, 11, 14, 10, 9, NA, 13, 15, 7, 12, NA, NA)
kable(predicted_finish %>% select(-predicted, -quali, -driver_standing) %>% rename(expert = odds) %>% a
```

driver_name	expert	predicted_finish	true
Verstappen	1	1	1
Hamilton	2	3	2
Leclerc	8	5	3
Sainz	6	4	4
Norris	4	9	5
Russel	5	7	6
Ricciardo	15	17	7
Piastri	7	8	8
Albon	13	13	9
Ocon	10	12	10
Gasly	11	10	11
Tsunoda	12	18	12
Hulkenberg	18	15	13
Bottas	17	11	14
Zhou	16	16	15
Perez	3	2	NA
Alonso	9	6	NA
Stroll	14	14	NA
Magnussen	19	19	NA
Sargeant	20	20	NA

```
# How many the model predicted the position correctly & how many the sports
# bets odds predicted correctly
correct_prediction = 0
correct_odds = 0
for(i in 1:20){
  if(!is.na(predicted_finish$true[i]) & predicted_finish$true[i]==i){
    correct_prediction = correct_prediction + 1
  }
  if(!is.na(predicted_finish$true[i]) & predicted_finish$true[i]==predicted_finish$odds[i]){
    correct_odds = correct_odds + 1
```

```
  }
}

cat("Model % correct predictions:", correct_prediction/20)
```

```
## Model % correct predictions: 0.15
```

```
cat("Sports bets % correct predictions:", correct_odds/20)
```

```
## Sports bets % correct predictions: 0.25
```

```
# Absolute difference in model's position error and difference in sports bets
# position error
total_diff = 0
odds_diff = 0
for(i in 1:nrow(predicted_finish)){
  if(!is.na(predicted_finish$true[i])){
  total_diff = total_diff + abs(predicted_finish$true[i] - i)
  odds_diff = odds_diff + abs(predicted_finish$true[i] - predicted_finish$odds[i])
  }
}

cat("Model average absolute difference:", total_diff/20)
```

```
## Model average absolute difference: 1.85
```

```
cat("Sports bets average absolute difference:", odds_diff/20)
```

```
## Sports bets average absolute difference: 1.55
```

```
#Ignoring drivers that DNFed
predicted_finish = predicted_finish[complete.cases(predicted_finish),]
#Updating odds by ignoring drivers that DNFed
predicted_finish = predicted_finish %>%
  mutate(odds = case_when((odds > 3 & odds < 9) ~ (odds - 1),
                          (odds > 9 & odds < 11) ~  (odds - 2),
                          (odds > 11 & odds < 14) ~ (odds - 3),
                          TRUE ~ odds))


# How many the model predicted the position correctly & how many the sports bets odds
# predicted correctly when removing DNF drivers
correct_prediction = 0
correct_odds = 0
for(i in 1:nrow(predicted_finish)){
  if(!is.na(predicted_finish$true[i]) & predicted_finish$true[i]==i){
    correct_prediction = correct_prediction + 1
  }
  if(!is.na(predicted_finish$true[i]) & predicted_finish$true[i]==predicted_finish$odds[i]){
    correct_odds = correct_odds + 1
```

```
  }
}

cat("Model % correct predictions ignoring DNFs:", correct_prediction/15)
```

```
## Model % correct predictions ignoring DNFs: 0.2
```

```
cat("Sports bets % correct predictions ignoring DNFs:", correct_odds/15)
```

```
## Sports bets % correct predictions ignoring DNFs: 0.2
```

```
# Absolute difference in model's position error and difference in sports bets position
# error when removing DNF drivers
total_diff = 0
odds_diff = 0
for(i in 1:nrow(predicted_finish)){
  total_diff = total_diff + abs(predicted_finish$true[i] - i)
  odds_diff = odds_diff + abs(predicted_finish$true[i] - predicted_finish$odds[i])
}

cat("Model average absolute difference ignoring DNFs:", total_diff/15)
```

```
## Model average absolute difference ignoring DNFs: 2
```

```
cat("Sports bets average absolute difference ignoring DNFs:", odds_diff/15)
```

```
## Sports bets average absolute difference ignoring DNFs: 2.266667
```

Overall, the model performed quite well when compared to the experts. Both predicted Max Verstappen to win, which he did. If we exclude all of the DNFs, which are frequently due to misfortune and not race pace, both the model and sports bets exactly predicted 3 positions correctly. The model was off by two positions, on average, which was slightly better than the sports betting odds at 2.23 positions on average. If Sergio Perez (predicted as 2nd) did not DNF on the first corner and ended up finishing close to the front like his teammate, the model would have been even more accurate compared to the sports betting odds. Overall, this is quite impressive considering the model only takes 2 predictors. Future models could be tested using other predictors.