DAT 301 Project 1

Jason Kong

2024-04-11

Introduction

Dataset

For this project, I collected and compiled my own dataset using `f1dataR`, an R package that wraps the FastF1 (https://github.com/theOehrly/Fast-F1) Python package, which gives me access to detailed Formula 1 telemetry information and timing data.

My dataset consists of every driver's individual lap time data and various other data entries from all 22 races of the 2023 Formula 1 season. Since `f1dataR` takes time pulling data from an online server and requires loading and setup of the `reticulate` library, I will only be presenting the code I used to combine all the data instead of putting it in an execution block.

```r
library(f1dataR)

# Create 22 dataframes, each contains individual lap times of every driver from one round.
for (i in 1:22) {
  name = paste0("r", i)
  value = load_session_laps(season = 2023, round = i, session = "R")
  assign(name, value)
}

# Add a column of repeating number to represent the round number of each dataframe,
# for ease of data manipulation when all dataframes are combined.
for (i in 1:22) {
  name = paste0("r", i)
  round = c(rep(i, nrow(get(name))))
  new_value = cbind(get(name), round)
  assign(name, new_value)
}

# Combine all 22 dataframes into one, and writing it to a csv file.
results = r1
for (i in 2:22) {
  results = rbind(results, get(paste0("r",i)))
}
write.csv(lapply(results, as.character), "2023results.csv", row.names = FALSE)
```

Background

The 2023 F1 season saw one of the most dominant performances from a single team in the sport's history. Red Bull Racing, with their RB19 race car, took victory in 21 out of the 22 races, a record-breaking win rate of over 95%. Although it was not always the fastest car when it came to single-lap pace, it was apparent throughout the season that the RB19 was extremely gentle on its tires, which allowed its drivers to maintain fast lap times over long periods of time, something the rival teams' cars could not achieve.

The problem I will be trying to solve is this:

Given the lap time data, can we create a model that compares the tire degradation performance of all cars on the grid?

Loading the Libraries

```r
library(dplyr)
library(ggplot2)
library(plotly)
library(knitr)
```

Dataset Loading and Overview

```r
results = read.csv("2023results.csv")
head(results, n = 3)
```

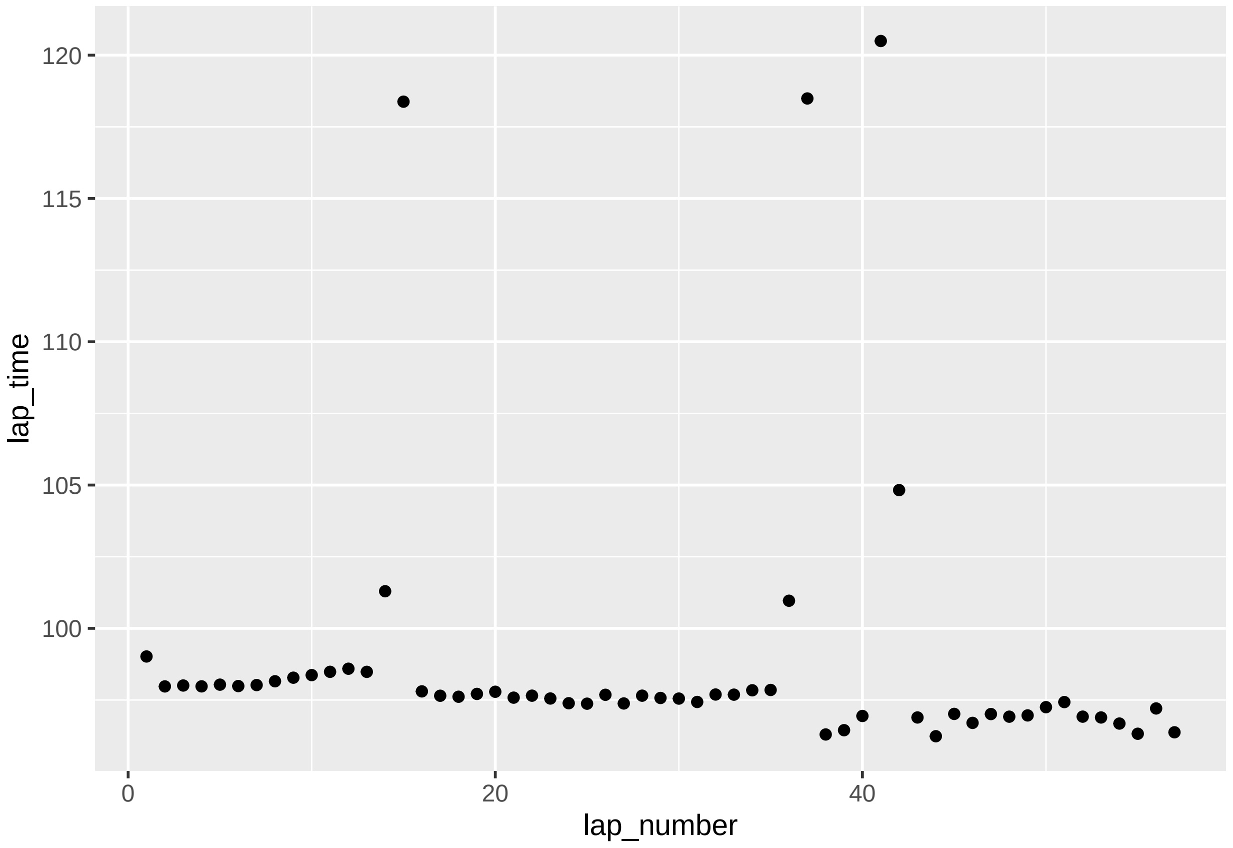
```
##       time driver driver_number lap_time lap_number stint pit_out_time
## 1 3855.961    VER             1   99.019          1     1          NaN
## 2 3953.935    VER             1   97.974          2     1          NaN
## 3 4051.941    VER             1   98.006          3     1          NaN
##   pit_in_time sector1time sector2time sector3time sector1session_time
## 1         NaN         NaN      42.414      23.842                 NaN
## 2         NaN      31.342      42.504      24.128            3887.303
## 3         NaN      31.388      42.469      24.149            3985.323
##   sector2session_time sector3session_time speed_i1 speed_i2 speed_fl speed_st
## 1            3832.119            3856.010      232      231      278      252
## 2            3929.807            3953.935      227      238      278      288
## 3            4027.792            4051.941      NaN      238      278      287
##   is_personal_best compound tyre_life fresh_tyre            team lap_start_time
## 1            FALSE     SOFT         4      FALSE Red Bull Racing       3756.652
## 2             TRUE     SOFT         5      FALSE Red Bull Racing       3855.961
## 3            FALSE     SOFT         6      FALSE Red Bull Racing       3953.935
##            lap_start_date track_status position deleted deleted_reason
## 1 2023-03-05 07:03:38.501           12        1   FALSE
## 2  2023-03-05 07:05:17.81           12        1   FALSE
## 3 2023-03-05 07:06:55.784            1        1   FALSE
##   fast_f1generated is_accurate session_type round
## 1            FALSE       FALSE            R     1
## 2            FALSE        TRUE            R     1
## 3            FALSE        TRUE            R     1
```

Initial Exploration

Our dataset contains many variables, but for our initial step, we only need to focus on `"driver"`, `"round"` and `"lap_time"`.

To visualize what we are trying to achieve, here is a plot of an individual driver's lap times during the first race of 2023:

```r
r1ver = filter(results, driver == "VER", round == 1)
ggplot(r1ver, aes(x = lap_number, y = lap_time)) +
  geom_point()
```



For this step, we filter `results` by the first round of the year, with Max Verstappen (`"VER"`) set as our driver, and plot out his lap times as the race went on.

Considerations

First, we notice 6 outliers in our lap times. These are caused by either tire changes, or yellow flag events indicating hazards on track, which require all drivers to slow down.

Second, we notice a downward trend in lap times as the race goes on. This can be explained by the weight of the fuel on-board the car. Each car is allowed a maximum of 110kg of fuel in the beginning of each race, and no refueling is allowed mid-race. Therefore, as the car burns fuel, its weight steadily decreases, resulting in faster lap times over the course of the race.

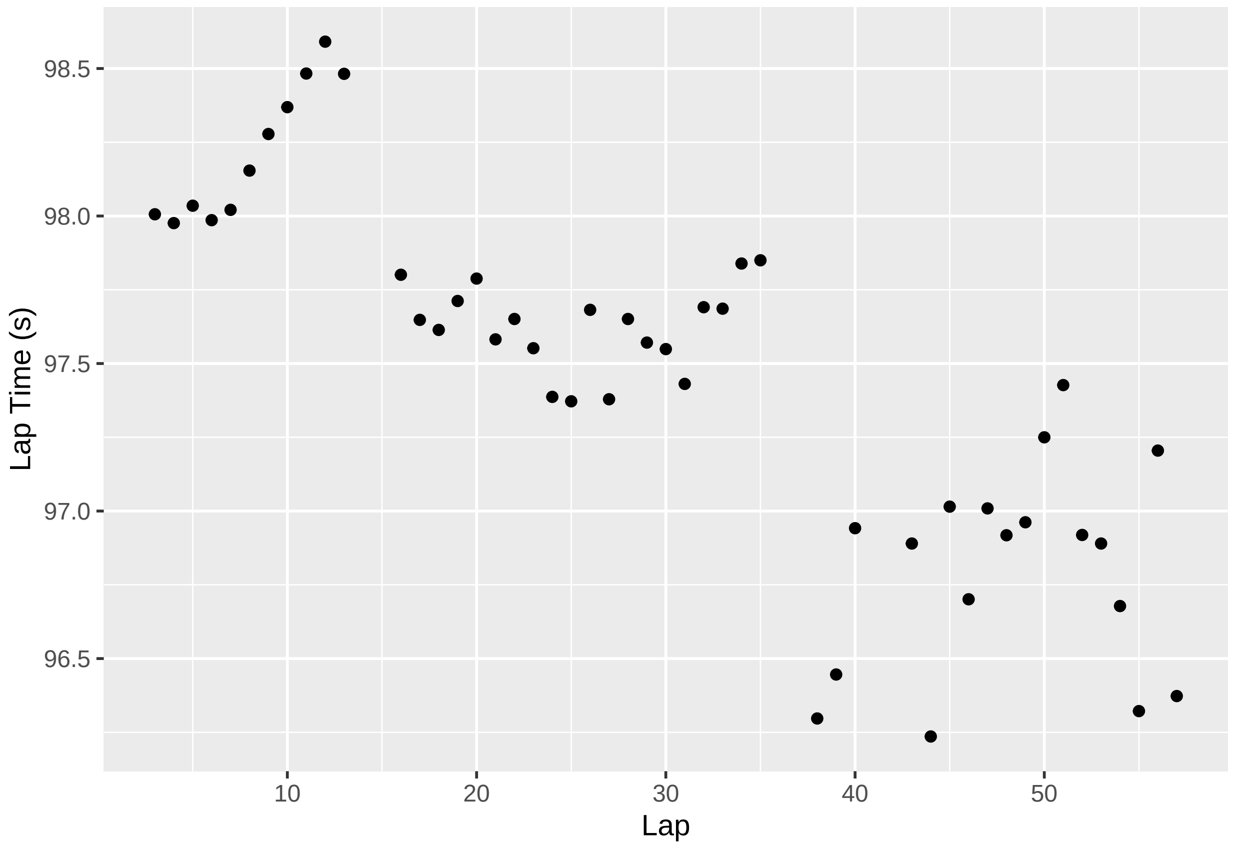
So how do we exclude these data points and remove fuel weight as a factor?

Luckily, our dataset contains information about in-laps and out-laps (laps that end/begin with the driver inside the pitlane getting a tire change), as well as track status.

To remove the outliers, all we need to do is further filter our dataset:

We will be getting rid of lap times where `"pit_in_time"` and `"pit_out_time"` contain non-NA values, as well as filtering the time table by the `"track_status"` of 1, indicating that the race is going on as normal.

```r
r1ver = filter(results, driver == "VER", round == 1,
               is.na(pit_in_time), is.na(pit_out_time), track_status == 1)
ggplot(r1ver, aes(x = lap_number, y = lap_time)) +
  geom_point() +
  labs (x = "Lap", y = "Lap Time (s)")
```



With the outliers removed, we can see all of Verstappen's undisturbed lap times during the race.

Now let's solve the second problem and remove on-board fuel weight as a factor of lap times.

It is difficult to determine exactly how much extra fuel weight slows down the car. The number could vary wildly depending on the race track and the inherent design of each car. However, for simplicity's sake, we will use the rule of thumb of around 0.03 second per lap per extra kg of fuel, as this seems to be the common consensus among various sources.

http://www.gurneyflap.com/prostap04araigne.html

https://www.the-race.com/formula-1/why-2022-f1-cars-are-so-heavy-and-where-weight-can-be-cut/

https://www.quora.com/Why-are-formula-1-cars-so-much-faster-in-qualifying-than-in-the-actual-race

All races on the F1 calendar are about 300km in total length. Assuming all cars start each race with 110kg of fuel, and finish each race with an almost empty fuel tank, we can use a simple linear slope to correct the lap times.

$$T_{\text{Corrected}} = T_0 - (110 \times 0.03) \cdot (1 - \frac{\text{Current Lap}}{\text{Total Laps}})$$

where

$$(110 \times 0.03) \cdot (1 - \frac{\text{Current Lap}}{\text{Total Laps}})$$

represents the magnitude of lap time correction as a function of the current lap number and the total number of laps.

For example, in a race with 50 laps, lap 1/50's lap time will be subtracted by

$$(110 \times 0.03) \cdot (1 - \frac{1}{50}) = 3.234s$$

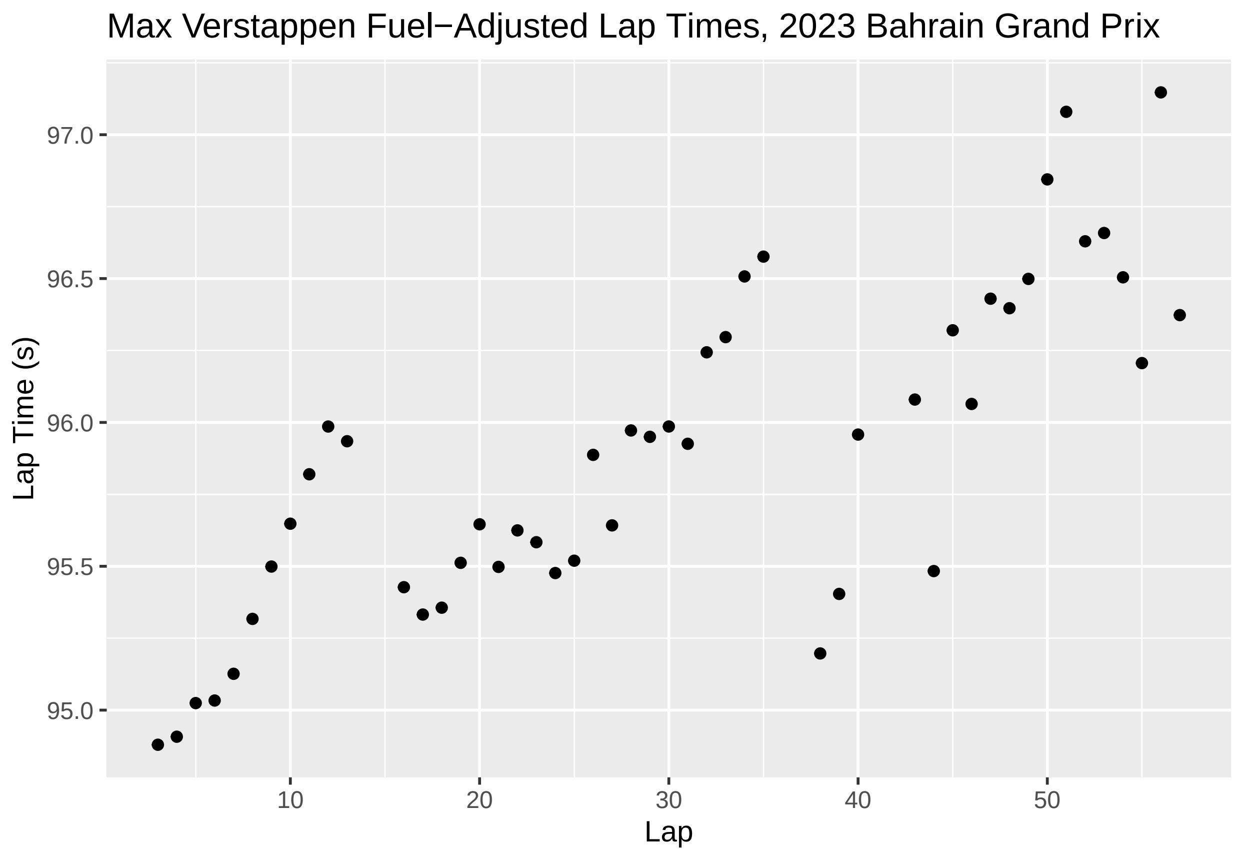
as the car has a full tank of fuel worth 110kg, but by lap 50/50, the lap time will not be adjusted at all as the fuel tank is near empty.

In R, this fuel correction function will be:

```r
fuel_correction = function(laptime, total_laps, current_lap){
  laptime = laptime - (110 * 0.03) * (1 - current_lap / total_laps)
}
```

With lap times now adjusted for fuel weight, let's plot the lap times again:

```r
r1ver = filter(results, driver == "VER", round == 1,
               is.na(pit_in_time), is.na(pit_out_time), track_status==1) %>%
        mutate(lap_time = fuel_correction(lap_time, max(lap_number), lap_number))
ggplot(r1ver, aes(x = lap_number, y = lap_time)) +
  geom_point() +
  labs (x = "Lap", y = "Lap Time (s)",
        title = "Max Verstappen Fuel-Adjusted Lap Times, 2023 Bahrain Grand Prix")
```



As we can see, there is no longer a downward trend of lap times. We now have a much more accurate representation of lap times and tire wear.

The overall upward trend of lap times can now be explained by fuel saving and the use of different tire compounds. Verstappen used a softer tire compound in the first part of this race, resulting in more grip and faster adjusted lap times. As the race went on, apart from switching to a harder tire compound, he also needed to conserve fuel, resulting in slower lap times.

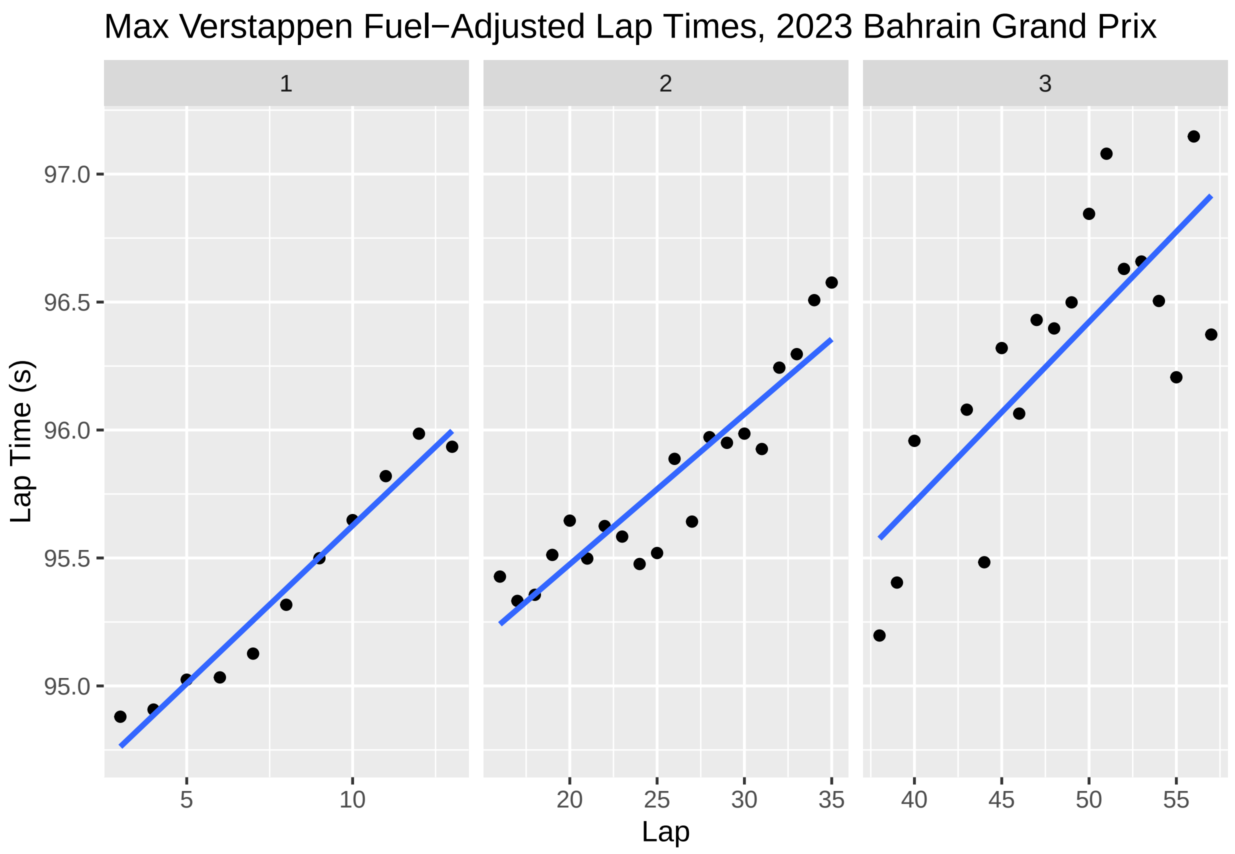
Note: Fuel saving is a tactic used by all drivers in Formula 1, as 110kg of fuel is not enough to support over 300km of maximum power output. Choices of tire compounds are also similar among drivers. Therefore, these two factors will not affect our calculations.

Further refinements

Notice that the data points on the above plot are congregated into 3 clusters. Each cluster represents continuous laps completed on the same set of tires. In motor racing, these continuous laps in between tire changes are called a “stint”.

We can split the graph into individual stint facets using the `"stint"` column for an even more accurate representation of tire wear. This also allows us to create a linear regression line for each stint:

```r
r1ver = filter(results, driver == "VER", round == 1,
               is.na(pit_in_time), is.na(pit_out_time), track_status==1) %>%
        mutate(lap_time = fuel_correction(lap_time, max(lap_number), lap_number))
ggplot(r1ver, aes(x = lap_number, y = lap_time)) +
  geom_point() +
  facet_wrap( . ~ stint , scales = "free_x") +
  geom_smooth(se = FALSE, method = 'lm') +
  scale_x_continuous(breaks = seq(0, max(r1ver$lap_number), by = 5)) +
  labs (x = "Lap", y = "Lap Time (s)",
        title = "Max Verstappen Fuel-Adjusted Lap Times, 2023 Bahrain Grand Prix")
```



Calculation

To get the average tire wear trend of the entire race, we take the lap time regression slope of all three stints, and calculate their mean:

```
r1ver.stints = group_by(r1ver, stint) %>%
      summarize(slope = coef(lm(lap_time ~ lap_number))[2])
r1ver.stints
```

```
## # A tibble: 3 x 2
##   stint  slope
##   <int>  <dbl>
## 1     1 0.123
## 2     2 0.0586
## 3     3 0.0706
```

```
summarize(r1ver.stints, mean_wear_coef = mean(slope))
```

```
## # A tibble: 1 x 1
##   mean_wear_coef
##            <dbl>
## 1         0.0842
```

Here we can see the slope of each of his 3 stints in the race and their mean of 0.842. This will be Verstappen's average tire wear coefficient for round 1.

Repeating this for all 22 rounds of the 2023 F1 season:

```
# Create empty data frame.
wear.ver = data.frame()

# Iterate through Verstappen's lap time data of each race,
for (i in 1:22) {
  times = filter(results,driver == "VER", round == i,
                 !is.na(lap_time), is.na(pit_in_time), is.na(pit_out_time),
                 compound !="WET", compound != "INTERMEDIATE", track_status==1)

  # Grouping all lap time data by stint.
  stints = group_by(times,stint) %>%

    # Adjusting lap times for fuel weight.
    mutate(lap_time = fuel_correction(lap_time, max(lap_number), lap_number)) %>%

    # Creating a wear coefficient for each stint.
    summarize(slope = coef(lm(lap_time~lap_number))[2])

  # Combining them to a single data frame.
  wear.ver = rbind(wear.ver, stints)
}

# Obtain the mean tire wear coefficient of all stints.
summarize(filter(wear.ver, !is.na(slope)), mean_wear_coef = mean(slope))
```

```
## # A tibble: 1 x 1
##   mean_wear_coef
##            <dbl>
## 1         0.0796
```

In the above model, we also added some extra filters that exclude lap times completed with Wet or Intermediate tires, as the tire wear pattern in wet condition are not consistent with dry condition.

There is one more problem:

```
filter(wear.ver, slope <= 0)
```

```
## # A tibble: 8 x 2
##   stint    slope
##   <int>    <dbl>
## 1     1 -0.157
## 2     2 -0.00355
## 3     1 -0.00435
## 4     2 -0.0389
## 5     2 -0.0218
## 6     3 -0.465
## 7     5 -0.00599
## 8     3 -0.0266
```

Even adjusted for the weight of on-board fuel, there are still many occasions where the driver's lap times appear to trend downwards.

These are occasions when the driver is aggressively managing tire or fuel, or when weather is a factor, causing the driver to slow down as the race track transitions from dry to wet.

We will exclude these negative slopes from our model, as well as the top and bottom 5% of positive slopes to remove any remaining outliers.

```
summarize(filter(wear.ver, !is.na(slope), slope >= 0),
          mean_wear_coef = mean(slope[slope >= quantile(slope, probs = 0.05) &
                                      slope <= quantile(slope, probs = 0.95)]))
```

```
## # A tibble: 1 x 1
##   mean_wear_coef
##            <dbl>
## 1         0.0995
```

Final Model

With a solid baseline, let's now expand our model to all drivers on the grid.

```r
# Create an empty main data frame for the entire grid.
wear.all = data.frame()

# Iterate through all driver names that appear in the 2023 season,
# and create an individual empty data frame for each driver.
for (i in unique(results$driver)){
  name = i
  assign(name, data.frame())
}

# Iterate through every race of the season.
for (i in 1:22) {

  # Filter the "result" dataset so it only contains data from the current race.
  race = filter(results, round == i,
                !is.na(lap_time), is.na(pit_in_time), is.na(pit_out_time),
                compound !="WET", compound != "INTERMEDIATE", track_status==1)

  # Iterate through every driver in the currently iterated race.
  for (j in unique(race$driver)) {

    # Filter the data frame by the currently iterated driver.
    name = j
    drivers = filter(race, driver == j)

    # Grouping all lap time data by stint.
    wear.coef = group_by(drivers, stint) %>%

      # Adjusting lap times for fuel weight.
      mutate(lap_time = fuel_correction(lap_time, max(lap_number), lap_number)) %>%

      # Creating a wear coefficient for each stint.
      summarize(slope = coef(lm(lap_time ~ lap_number))[2],driver=j)

    # Assign wear coefficient data to the individual driver data frame.
    assign(name, wear.coef)

    # Append the current driver's data to our main data frame
    wear.all = rbind(wear.all, get(name))
  }
}

# Obtain the mean tire wear coefficient of all drivers and all stints.
summary = summarize(group_by(filter(wear.all, !is.na(slope), slope >= 0), driver),
                    mean_wear_coef = mean(slope[slope >= quantile(slope, probs = 0.05) &
                                                slope <= quantile(slope, probs = 0.95)]))
```

```r
print(arrange(summary, mean_wear_coef), n = 22)
```

```
## # A tibble: 22 x 2
##    driver mean_wear_coef
##    <chr>           <dbl>
##  1 PER            0.0898
##  2 ALO            0.0934
##  3 LAW            0.0952
##  4 VER            0.0995
##  5 DEV            0.108
##  6 STR            0.108
##  7 GAS            0.110
##  8 PIA            0.114
##  9 SAI            0.115
## 10 OCO            0.115
## 11 BOT            0.116
## 12 HAM            0.119
## 13 RUS            0.119
## 14 LEC            0.123
## 15 ALB            0.130
## 16 HUL            0.130
## 17 TSU            0.135
## 18 ZHO            0.141
## 19 MAG            0.142
## 20 SAR            0.149
## 21 NOR            0.150
## 22 RIC            0.151
```

We now have a complete table with the average tire wear coefficient of all drivers throughout the season.

Refining the Results

Let's make the table more legible by replacing the abbreviations with drivers' full names and their teams.

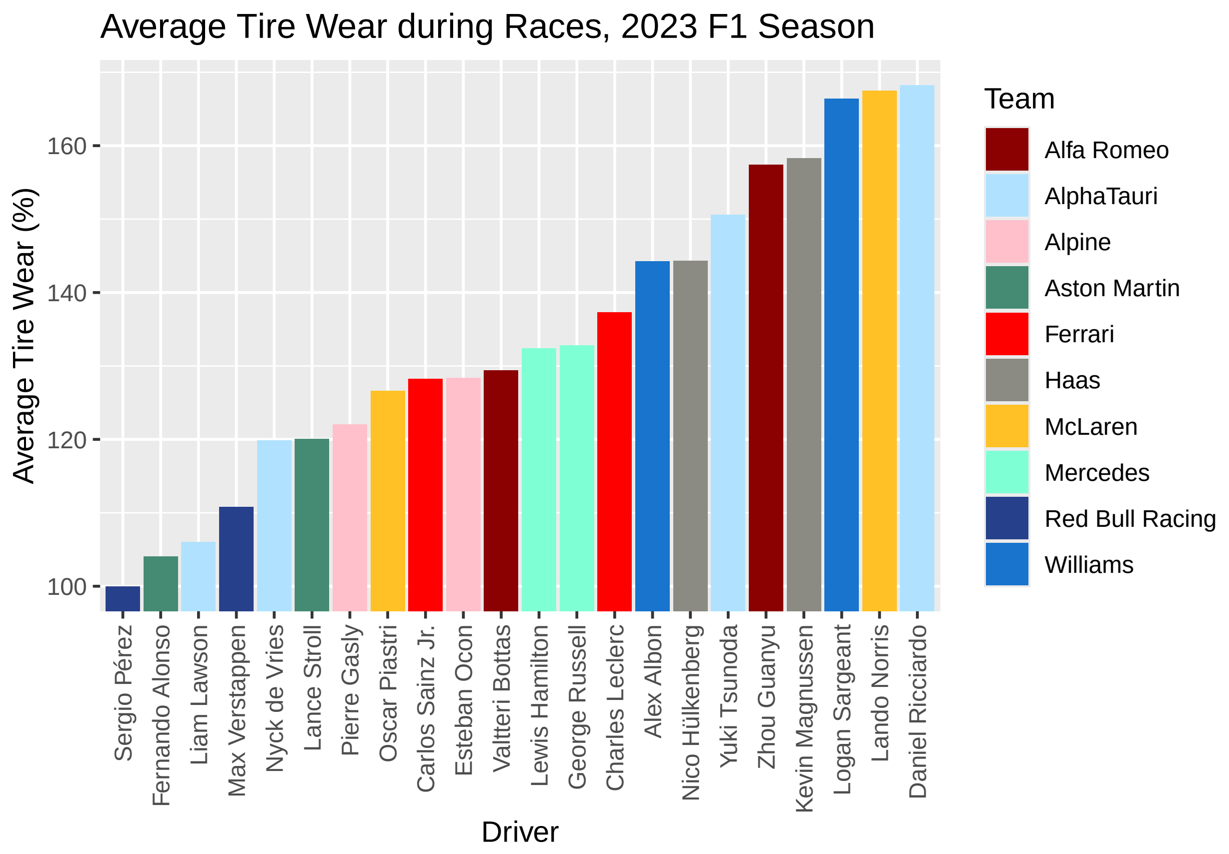
We will also replace the `"mean_wear_coef"` with a percentage scale using the driver with the least average tire wear as the benchmark, as the coefficients are quite arbitrary.

```r
# Create a key matching driver abbreviations with their full names and their teams.
driver_index = data.frame(
  abb = select(summary, driver),
  full_name = c("Alex Albon", "Fernando Alonso", "Valtteri Bottas", "Nyck de Vries",
           "Pierre Gasly", "Lewis Hamilton", "Nico Hülkenberg", "Liam Lawson",
           "Charles Leclerc", "Kevin Magnussen", "Lando Norris", "Esteban Ocon",
           "Sergio Pérez", "Oscar Piastri", "Daniel Ricciardo", "George Russell",
           "Carlos Sainz Jr.", "Logan Sargeant", "Lance Stroll", "Yuki Tsunoda",
           "Max Verstappen", "Zhou Guanyu"),
  team = c("Williams", "Aston Martin", "Alfa Romeo", "AlphaTauri", "Alpine", "Mercedes",
           "Haas", "AlphaTauri", "Ferrari", "Haas", "McLaren", "Alpine", "Red Bull Racing",
           "McLaren", "AlphaTauri", "Mercedes", "Ferrari", "Williams", "Aston Martin",
           "AlphaTauri", "Red Bull Racing", "Alfa Romeo")
)
# Merge our raw result with the key.
sum.names = merge(driver_index, summary, by = "driver")
# Remove the abbreviations, sort the table and turn the results into a percentage scale.
ranking = select(sum.names, -driver) %>% arrange(mean_wear_coef) %>%
  mutate(mean_wear_coef = paste0(format((mean_wear_coef - min(mean_wear_coef)) /
                                        min(mean_wear_coef) * 100 + 100,
                                      digits = 4), "%"))
kable(ranking, col.names = c("Driver", "Team", "Avg. Tire Wear"), align = "llc")
```

Driver	Team	Avg. Tire Wear
Sergio Pérez	Red Bull Racing	100.0%
Fernando Alonso	Aston Martin	104.1%
Liam Lawson	AlphaTauri	106.0%
Max Verstappen	Red Bull Racing	110.8%
Nyck de Vries	AlphaTauri	119.9%
Lance Stroll	Aston Martin	120.1%
Pierre Gasly	Alpine	122.0%
Oscar Piastri	McLaren	126.6%
Carlos Sainz Jr.	Ferrari	128.3%
Esteban Ocon	Alpine	128.4%
Valtteri Bottas	Alfa Romeo	129.4%
Lewis Hamilton	Mercedes	132.4%
George Russell	Mercedes	132.8%
Charles Leclerc	Ferrari	137.3%
Alex Albon	Williams	144.3%
Nico Hülkenberg	Haas	144.3%
Yuki Tsunoda	AlphaTauri	150.6%
Zhou Guanyu	Alfa Romeo	157.4%
Kevin Magnussen	Haas	158.3%
Logan Sargeant	Williams	166.4%
Lando Norris	McLaren	167.5%
Daniel Ricciardo	AlphaTauri	168.2%

Here is the comparison visualized:

```r
ranking.plot = select(sum.names, -driver) %>% arrange(mean_wear_coef) %>%
  mutate(mean_wear_coef = (mean_wear_coef - min(mean_wear_coef)) /
                                  min(mean_wear_coef) * 100 + 100, digits = 4)
ggplot(ranking.plot, aes(x = reorder(full_name, mean_wear_coef),
                         y = mean_wear_coef, fill = team)) +
  geom_bar(stat = "identity") +
  coord_cartesian(ylim = c(100,NA)) +
  scale_fill_manual(name = "Team", values = c("red4", "lightskyblue1", "pink",
                                              "aquamarine4", "red", "ivory4",
                                              "goldenrod1", "aquamarine",
                                              "royalblue4", "dodgerblue3")) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
  labs(x = "Driver", y = "Average Tire Wear (%)",
     title = "Average Tire Wear during Races, 2023 F1 Season")
```



Conclusions

This tire wear comparison does not serve as an overall performance ranking of drivers and teams, because a fast car with bad tire wear characteristics will still outperform a much slower car that is more gentle on its tires. In addition, drivers' driving styles will also have a minor impact on tire wear.

With that being said, we can clearly see that the two Red Bull cars' superior tire wear characteristics greatly contributed to the team's dominance. Looking at the next two best teams of 2023, Mercedes and Ferrari; Despite their occasional showing of faster single-lap pace, their cars' tire wear was around 20-30% more than that of the Red Bull cars, which prevented them from converting their single-lap performances to race wins.