Visualizing Formula 1 Data in R

Casan Scott, Ph.D.

2024-01-09

# 1 About the book

This is a book for anyone that finds themselves located within the fairly obscure intersection of the data enthusiast-Formula 1 fan Venn diagram (see Figure below!). I am a Data Scientist by trade and a relatively new fan of Formula 1. For these reasons, this book will probably fail to adequately introduce someone to both R and Formula 1 racing. However, I’ll do my best to introduce approaches for analyzing Formula 1 data using R and include sources for additional information on topics.

## 1.1 About me

TLDR: Casan Scott, Ph.D., Senior Data Scientist

Supplemental information: I am a bit of a nomad, professionally. During my high school years, I played as many sports as possible and I could not have cared less about academics. I eventually graduated with a B.S. and Ph.D. in environmental science. Naturally, I decided to *not* use my doctorate degree and became a data scientist! Over the past five years, I’ve worked as a data scientist in both the public and private sectors. Sometimes, I also contribute to research on human performance topics. Outside of my nerdier pursuits, I enjoy competing in powerlifting, trail running, reading westerns, and hanging out with my wife and dogs.

## 1.2 Why did I write this book?

To scratch my own itch *per se*. Formula 1 is actually a very complicated sport, and while learning about it I had lots of questions: *Do practice times mean anything? Does the qualifying time predict race pace? How much do the cars improve each year? Why is Red Bull so good in Mexico City?* Of course, there were plenty of answers to these questions online, but I really wanted to try and answer some of these questions myself. So, this book documents my attempt at doing so. Along the way, I include the code to create each plot and some brief annotations about the functions used.

## 1.3 Organization of this book

I tried to organize this book like a typical Formula 1 weekend: (1) practice data, (2) qualifying data , and (3) results from the Grand Prix. For the most part, I try to summarize the historical data, visualize possible trends in the data, explore important differences, plot potential correlations, and perhaps build a model that can explain some relationship(s) that seems interesting. Along the way, I may pursuit a tangent or two. Each chapter begins with a racing-centric introduction, followed by a data analysis.

## 1.4 ggplot2

I am assuming that you have some experience using the R programming language. If you have never used R before, you will quickly become very confused! This entire book is based on the **ggplot2** package in R. **ggplot2** is a package used for producing statistical and other data graphics. What makes **ggplot2** unique, is it has an underlying *grammar*, based on the Grammar of Graphics (Wilkinson 2005). This attribute allows you to create graphs by combining independent components to suite your particular problem. That may seem like a trivial distinction, but it makes **ggplot2** very powerful!

Note: Throughout this text, I may refer to **ggplot2** as simply **ggplot**. If that is annoying… apologies in advance!

## 1.5 The *tidyverse*

Throughout this book, I rely on the **tidyverse**. The **tidyverse** is actually a collection of other packages that share common data representations and design structures. This enables these packages to work together very conveniently. For instance, I nearly always use functions from the **dplyr** package to wrangle data prior to plotting with **ggplot2**. You can install the **tidyverse** core packages with a single command:

# Install the package from CRAN  
install.packages("tidyverse")  
  
# Load the package  
library(tidyverse)

The core pakcages in **tidyverse** are:

* **ggplot2**, for data visualization.
* **dplyr**, for data manipulation.
* **tidyr**, for data tidying.
* **readr**, for data import.
* **purrr**, for functional programming.
* **tibble**, for tibbles, a modern re-imagining of data frames.
* **stringr**, for strings.
* **forcats**, for factors.
* **lubridate**, for date/times.

Aside from **ggplot2**, I most heavily use **dplyr** in this book. There are six key **dplyr** functions that can solve most data manipulation challenges:

* select(): subset columns from a dataframe. In other words, you use this to chooswe variables by name.
* filter(): filter the dataframe by values of a variable. This can be both numeric (i.e. age > 21 years) or categorical (i.e. month == June).
* arrange(): order the rows of the dataframe
* mutate(): create a new variable
* summarize(): collapse values down to a summarized metric (i.e. mean, minimum, maximum).
* group\_by(): operates any of the previous functions on a group basis (i.e. a grouped mean).

I highly recommend this book chapter for more information on **dplyr** basics:

<https://r4ds.had.co.nz/transform.html>

For a thorough introduction to the **tidyverse**, read the entire fantastic book:

<https://r4ds.had.co.nz/index.html>

## 1.6 Data

The data used in this book was pulled from formula1.com. To make it easier for readers to use this data, I developed the R packaged **drs** (**D**ata for **R**acing **S**imulations). If you currently follow Formula 1, you’ll recognize that DRS also standing for *Drag Reduction System*, which is a crucial component of Formula 1 cars that opens space in the rear wing thereby reducing drag. **drs** is also an R package that utilizes functions from the **rvest** and **dplyr** packages to scrape and tidy data from the formula1.com website. It is designed to scrape data for all Grand Prix weekends during a given year.

**Notes about a Formula 1 Grand Prix Weekend**

A Grand Prix weekend consists of practice sessions, qualifying, and a race. Additionally, some weekends will also include a sprint race (an abbreviated sprint race that is typically about 1/3 the length of a normal race). A typical Formula 1 weekend begins with three practice sessions. The first two practice sessions (FP1 and FP2) are held on Friday. On Saturday, the third practice session (FP3) is held, followed by Qualifying. Qualifying determines the grid for the race on Sunday. Currently, there are three heats in qualifying (Q1, Q2, and Q3). All cars compete in the first heat (*Q1*), and the top 15 fastest times advance to the 2nd heat, *Q2*. From there, the top 10 fastest times during heat 2 advance to *Q3* (heat 3). The starting grid is determined by a driver’s final qualifying position (sans penalties). The race takes place on Sunday.

**drs** currently consists of four scraping functions:

* practice\_session\_scraper(): Scrapes the best times for a given practice session.
* qualifying\_scraper(): Scrapes the best qualifying times during Q1, Q2, and Q3.
* starting\_grid\_scraper(): Scrapes the final starting grid positions for the Grand Prix.
* race\_result\_scraper()(): Scrapes the race results for a Grand Prix (i.e. finishing position and total time).

#### 1.6.0.1 Installation of drs

You can install the development version of **drs** here:

install\_github("casanscott/drs")

#### 1.6.0.2 Using the drs package

These functions from **drs** require both the **tidyverse** and **rvest** packages. The practice\_session\_scraper() requires two arguments: year and practice\_session\_number. After loading those libraries, along with **drs**, you can easily scrape data using a function call like this:

library(tidyverse)  
library(rvest)  
library(drs)  
  
# pull FP3 practice data  
p32022 <- practice\_session\_scraper(2022, 3)  
  
# View the first 6 rows  
head(p32022)

## Position CarNumber First Last Driver Car Time Race Circuit Year  
## 1 1 1 Max Verstappen VER Red Bull Racing RBPT 1:32.544 bahrain bahrain 2022  
## 2 2 16 Charles Leclerc LEC Ferrari 1:32.640 bahrain bahrain 2022  
## 3 3 11 Sergio Perez PER Red Bull Racing RBPT 1:32.791 bahrain bahrain 2022  
## 4 4 63 George Russell RUS Mercedes 1:32.935 bahrain bahrain 2022  
## 5 5 55 Carlos Sainz SAI Ferrari 1:33.053 bahrain bahrain 2022  
## 6 6 44 Lewis Hamilton HAM Mercedes 1:33.121 bahrain bahrain 2022  
## Time\_secs  
## 1 92.544  
## 2 92.640  
## 3 92.791  
## 4 92.935  
## 5 93.053  
## 6 93.121

The rest of the **drs** web scraping functions require a single argument: year.

The following function will scrape all qualifying results from 2022:

# pull qualifying data  
quali2022 <- qualifying\_scraper(2022)  
  
# View the first 6 rows  
head(quali2022)

## Position CarNumber First Last Driver Car Laps Q1 Q2  
## 1 1 16 Charles Leclerc LEC Ferrari 15 1:31.471 1:30.932  
## 2 2 1 Max Verstappen VER Red Bull Racing RBPT 14 1:31.785 1:30.757  
## 3 3 55 Carlos Sainz SAI Ferrari 15 1:31.567 1:30.787  
## 4 4 11 Sergio Perez PER Red Bull Racing RBPT 18 1:32.311 1:31.008  
## 5 5 44 Lewis Hamilton HAM Mercedes 17 1:32.285 1:31.048  
## 6 6 77 Valtteri Bottas BOT Alfa Romeo Ferrari 15 1:31.919 1:31.717  
## Q3 Race Circuit Year Q1\_secs Q2\_secs Q3\_secs  
## 1 1:30.558 bahrain bahrain 2022 91.471 90.932 90.558  
## 2 1:30.681 bahrain bahrain 2022 91.785 90.757 90.681  
## 3 1:30.687 bahrain bahrain 2022 91.567 90.787 90.687  
## 4 1:30.921 bahrain bahrain 2022 92.311 91.008 90.921  
## 5 1:31.238 bahrain bahrain 2022 92.285 91.048 91.238  
## 6 1:31.560 bahrain bahrain 2022 91.919 91.717 91.560

To scrape the starting grids for every Grand Prix during 2022, use the following function call:

# pull starting grids  
grids2022 <- starting\_grid\_scraper(2022)  
  
# View the first 6 rows  
head(grids2022)

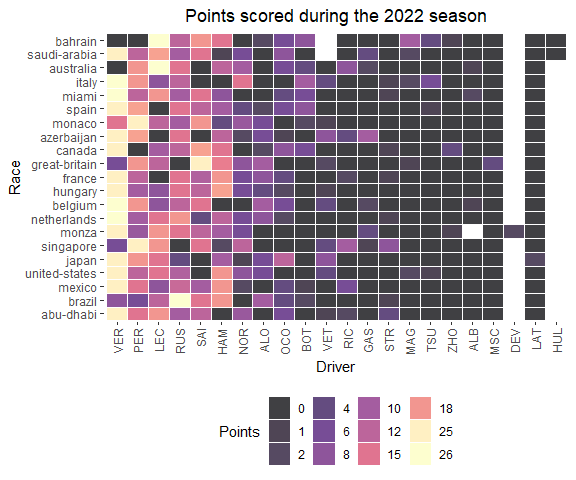
## Position CarNumber First Last Driver Car Time Race Circuit Year  
## 1 1 16 Charles Leclerc LEC Ferrari 1:30.558 bahrain bahrain 2022  
## 2 2 1 Max Verstappen VER Red Bull Racing RBPT 1:30.681 bahrain bahrain 2022  
## 3 3 55 Carlos Sainz SAI Ferrari 1:30.687 bahrain bahrain 2022  
## 4 4 11 Sergio Perez PER Red Bull Racing RBPT 1:30.921 bahrain bahrain 2022  
## 5 5 44 Lewis Hamilton HAM Mercedes 1:31.238 bahrain bahrain 2022  
## 6 6 77 Valtteri Bottas BOT Alfa Romeo Ferrari 1:31.560 bahrain bahrain 2022  
## Time\_secs  
## 1 90.558  
## 2 90.681  
## 3 90.687  
## 4 90.921  
## 5 91.238  
## 6 91.560

To scrape the race results for every Grand Prix during 2022, use the following function call:

# Pull race results  
races2022 <- race\_result\_scraper(2022)  
  
# View the first 6 rows  
head(races2022)

## # A tibble: 6 × 13  
## Position CarNumber First Last Driver Car Laps Time Points Race Circuit Year Time\_secs  
## <chr> <int> <chr> <chr> <chr> <chr> <int> <chr> <int> <chr> <chr> <dbl> <dbl>  
## 1 1 16 Charles Leclerc LEC Ferr… 57 1:37… 26 bahr… bahrain 2022 5854.  
## 2 2 55 Carlos Sainz SAI Ferr… 57 +5.5… 18 bahr… bahrain 2022 6752.  
## 3 3 44 Lewis Hamilton HAM Merc… 57 +9.6… 15 bahr… bahrain 2022 7069.  
## 4 4 63 George Russell RUS Merc… 57 +11.… 12 bahr… bahrain 2022 6725.  
## 5 5 20 Kevin Magnuss… MAG Haas… 57 +14.… 10 bahr… bahrain 2022 7448.  
## 6 6 77 Valtteri Bottas BOT Alfa… 57 +16.… 8 bahr… bahrain 2022 6933.

You can then use these dataframes to create cool data visualizations like this one:

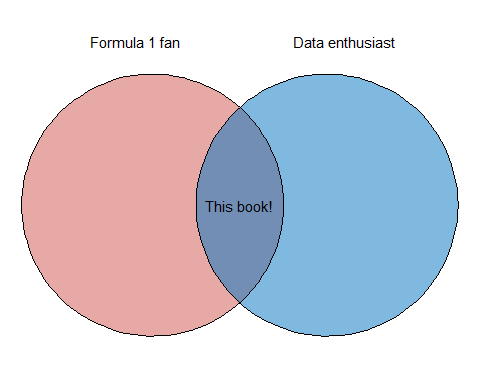


## 1.7 Data Visualization

In addition to commentary about Formula 1 racing, I’ll also include brief instructions about how to create the data visualizations in this book. For example, use this chunk of code…

library(ggvenn)  
library(ggplot2)  
  
x <- list(`Formula 1 fan` = rep('This book!', 1),  
 `Data enthusiast` = rep('This book!', 1))  
  
  
ggvenn(  
 x,   
 show\_elements = T,  
 show\_percentage = F,  
 fill\_color = c("#CD534CFF", "#0073C2FF"),  
 stroke\_size = 0.5, stroke\_alpha = 0.5, set\_name\_size = 4  
 )

… to create this figure:



I’ll also occasionally include shaded boxes with supplemental information about a particular chart type, an interesting R package, or anything else that seems noteworthy. For example, the following shaded box includes additional information on the R package used to create the Venn diagram (**ggvenn**):

**How to create a Venn diagram in R**

I used the **ggvenn** and **ggplot2** packages to create this simple Venn diagram. The **ggvenn** package was created by Linlin Yan. In my opinion, **ggvenn** is the easiest way to create simple Venn diagrams that follow the typical **ggplot2** styling and syntax.

For more information about the **ggvenn** package, visit this link: <https://cran.r-project.org/web/packages/ggvenn/index.html>

If you’ve never used **ggplot2** before (boy, are you in for a treat!), check out this link to the **ggplot** book written by legends of the tidyverse Hadley Wickham, Danielle Navarro, and Thomas Lin Pedersen: <https://ggplot2-book.org/>

In the next chapter, we will dive into Formula 1 data. I’ll start with a gentle introduction to Formula 1 and a typical Grand Prix weekend.

# 2 Introduction to Formula 1

What is Formula 1? The word *formula* refers to a particular ruleset that teams’ cars must conform to. The numeral *1* designates that this classification is the highest level of motorsport competition. So, *Formula 1* racing is the highest class of international open-wheel single-seater formula racing sanctioned by the Fédération Internationale de l’Automobile (FIA). A Formula 1 season includes a series of *Grands Prix* (races) that take place across the globe on various types of tracks, ranging from closed public city streets, designated racing circuits, or modern manufactured street-circuits.

Like many professional sports, Formula 1 is also big business! Thanks in part to the popular Netflix series *Drive to Survive*, Formula has experienced a recent explosion of popularity in the United States (I am one of these recent converts!). Along with this uptick in popularity, Formula 1 teams have also experienced a surge in valuation. To visualize this trend, I will create a bar chart. Prior to creating the bar chart with the ggplot() function, I have to create the dataframe and then reshape to a *longer format* it using the pivot\_longer() function.

**How to pivot data into a longer format in R**

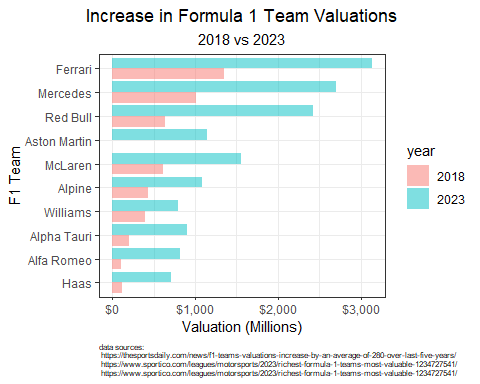
pivot\_longer() *lengthens* data, increasing the number of rows and decreasing the number of columns.

To *widen* the data, use the pivot\_wider() function, which is the inverse transformation.

For more information on pivot\_longer() and pivot\_wider(), visit this link: <https://tidyr.tidyverse.org/reference/pivot_longer.html>

The following figure was created using data compiled by <https://thesportsdaily.com/>, and originally retrieved from Sportico and Statista:

library(tidyverse)  
  
valuation\_table <- data.frame(`F1 Team` = c('Ferrari', 'Mercedes', 'Red Bull', 'McLaren', 'Aston Martin',  
 'Alpine', 'Alpha Tauri', 'Alfa Romeo', 'Williams', 'Haas'),  
 `2018 Valuation` = c(1350, 1015, 640, 620, NA, 430, 200, 105, 400, 115),  
 `2023 Valuation` = c(3130, 2700, 2420, 1560, 1140, 1080, 905, 815, 795, 710)) %>%  
 pivot\_longer(- F1.Team, names\_to = 'year', values\_to = 'valuation') %>%  
 mutate(year = str\_remove(year, 'X'),  
 year = str\_remove(year, '.Valuation'))  
  
valuation\_table %>%  
 ggplot(aes(y = fct\_reorder(F1.Team, valuation), x = valuation, fill = year)) +  
 geom\_histogram(stat = 'identity', position = 'dodge', alpha = 0.5) +  
 labs(title = 'Increase in Formula 1 Team Valuations',  
 subtitle = '2018 vs 2023',  
 y = 'F1 Team',  
 x = 'Valuation (Millions)',  
 caption = 'data sources: \n https://thesportsdaily.com/news/f1-teams-valuations-increase-by-an-average-of-280-over-last-five-years/ \n https://www.sportico.com/leagues/motorsports/2023/richest-formula-1-teams-most-valuable-1234727541/ \n https://www.sportico.com/leagues/motorsports/2023/richest-formula-1-teams-most-valuable-1234727541/') +  
 theme\_bw() +  
 scale\_x\_continuous(labels = scales::dollar\_format()) +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0, size = 6))



**How to create a bar chart in R**

To create the figure above, I used the bar geometry, geom\_bar(), in **ggplot2**. I often use bar charts to plot counts or sums by some grouping variable.

I used two arguments within geom\_bar(): stat = 'identity' ensures that **ggplot** uses the dollar sum as listed in the column, while position = 'dodge' preserves the position along the x-axis.

For more information on *dodging* in **ggplot2**, visit this link: <https://ggplot2.tidyverse.org/reference/position_dodge.html>

Each team has two cars driven by two drivers. Drivers and teams are awarded points based on their finishing position in each Grand Prix. The final tally of points scored will decide two competitions that are taking place during a Formula 1 season: (1) a driver’s championship, and (2) a constructors championship (team championship). Currently, Formula 1 has 10 teams, 20 drivers, and awards the following points for each finishing position. Additionally, 1 point is awarded to the driver with the fastest lap during a Grand Prix.

| Placing | Pts |
| --- | --- |
| 1 | 25 |
| 2 | 18 |
| 3 | 15 |
| 4 | 12 |
| 5 | 10 |
| 6 | 8 |
| 7 | 6 |
| 8 | 4 |
| 9 | 2 |
| 10 | 1 |
| 11 - 20 | 0 |

**How to create a table in R**

To create the table above, I simply pipe (i.e. %>%) a manually-created dataframe to the gt() function (i.e. df %>% gt()). The gt() function is found in the **gt** package. The **gt** package is the best way (in my opinion!) to create beautiful tables using the R programming language. **gt** functions work great with dataframes and tibbles, and can easily become an extension of yur typical tidy workflow. This example above is an absolute bare-bones examples of a **gt** table.

Another way to create this type of table would be to use the kable() function from the **knitr** package. The kable() function is a very simple table generator in R.

For more information about the **gt** package, visit this link:

<https://gt.rstudio.com/>

This guide provides lots of information about styling aesthetics of a table, including table headers and footers, column labels, and much, much more!

## 2.1 A Formula 1 Weekend

A Formula One Grand Prix is actually not just a single race, but rather a sporting event spanning three days, typically from Friday to Sunday. Beginning on Friday, there is typically two practices, followed by a third practice and a qualifying session on Saturday. Practice sessions provide teams an opportunity to *practice* on the circuit, and experiment with the setup of the car. Qualifying determines the grid for the race on Sunday. Currently, there are three heats in qualifying. All cars compete in the first heat, and the top 15 fastest times advance to the 2nd heat. From there, the top 10 fastest times during heat 2 advance to *Q3* (heat 3). The fastest qualifying times start the race at the front of the *grid*, and the slowest start at the back. The race takes place on Sunday, and top 3 placed drivers take their places on a *podium*.

## 2.2 Practice Sessions

Currently, there are three free practice sessions (often abbreviated to FP1, FP2, and FP3) that are held before the race. The first (FP1) is held on Friday morning, the second (FP2) on Friday afternoon, and the third session (FP3) is held on Saturday morning. Since 2021, practice sessions last for one hour, but previously Friday sessions were 90 minutes long.

Typically, teams will use each practice session for a different purpose. For instance, FP1 is often used to test the car and ensure it is working as expected, while also collecting information on the track and car setup. FP2 is often used for additional reconnaissance and testing of the car’s performance on *long runs*. FP3 is most commonly used to understand the car’s speed over lap (i.e. qualifying pace).

When I first became interested in Formula 1, I had tons of questions about the cars, racing, and structure of the Grand Prix weekend. While there are countless resources available to learn about Formula (shout out to r/formula1), I particularly enjoy learning by exploring data. Luckily, you can find practice, qualifying, and Grand Prix data on <https://formula1.com/>. To make that data a little more accessible, feel free to use the practice\_session\_scraper() function from my **drs** package.

To scrape FP1 data for 2014 through 2023, use the following code:

# Scrape FP1 data  
fp1\_2023 <- practice\_session\_scraper(2023, 1)  
fp1\_2022 <- practice\_session\_scraper(2022, 1)  
fp1\_2021 <- practice\_session\_scraper(2021, 1)  
fp1\_2020 <- practice\_session\_scraper(2020, 1)  
fp1\_2019 <- practice\_session\_scraper(2019, 1)  
fp1\_2018 <- practice\_session\_scraper(2018, 1)  
fp1\_2017 <- practice\_session\_scraper(2017, 1)  
fp1\_2016 <- practice\_session\_scraper(2016, 1)  
fp1\_2015 <- practice\_session\_scraper(2015, 1)  
fp1\_2014 <- practice\_session\_scraper(2014, 1)

To scrape FP2 data for the same time period, use this code:

# Scrape FP2 data  
fp2\_2023 <- practice\_session\_scraper(2023, 2)  
fp2\_2022 <- practice\_session\_scraper(2022, 2)  
fp2\_2021 <- practice\_session\_scraper(2021, 2)  
fp2\_2020 <- practice\_session\_scraper(2020, 2)  
fp2\_2019 <- practice\_session\_scraper(2019, 2)  
fp2\_2018 <- practice\_session\_scraper(2018, 2)  
fp2\_2017 <- practice\_session\_scraper(2017, 2)  
fp2\_2016 <- practice\_session\_scraper(2016, 2)  
fp2\_2015 <- practice\_session\_scraper(2015, 2)  
fp2\_2014 <- practice\_session\_scraper(2014, 2)

And finally, to scrape FP3 data, use this:

# Scrape FP3 data  
fp3\_2023 <- practice\_session\_scraper(2023, 3)  
fp3\_2022 <- practice\_session\_scraper(2022, 3)  
fp3\_2021 <- practice\_session\_scraper(2021, 3)  
fp3\_2020 <- practice\_session\_scraper(2020, 3)  
fp3\_2019 <- practice\_session\_scraper(2019, 3)  
fp3\_2018 <- practice\_session\_scraper(2018, 3)  
fp3\_2017 <- practice\_session\_scraper(2017, 3)  
fp3\_2016 <- practice\_session\_scraper(2016, 3)  
fp3\_2015 <- practice\_session\_scraper(2015, 3)  
fp3\_2014 <- practice\_session\_scraper(2014, 3)

Now that the data is scraped from www.formula1.com, I need to combine them together into one dataframe. I will use the rbind() function to row-bind data for each practice session.

**How to bind data in R**

The cbind() and rbind() functions combine by columns or rows, respectively.

For more information on these two functions, visit this link: <https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/cbind>

#Combine all practice data  
practice\_times <- rbind(fp3\_2023,  
 fp3\_2022,  
 fp3\_2021,  
 fp3\_2020,  
 fp3\_2019,  
 fp3\_2018,  
 fp3\_2017,  
 fp3\_2016,  
 fp3\_2015,  
 fp3\_2014) %>%  
 left\_join(rbind(fp2\_2023,  
 fp2\_2022,  
 fp2\_2021,  
 fp2\_2020,  
 fp2\_2019,  
 fp2\_2018,  
 fp2\_2017,  
 fp2\_2016,  
 fp2\_2015,  
 fp2\_2014) %>%  
 dplyr::select('Driver', 'Race', 'Year', 'Time', 'Time\_secs'), by = c('Driver', 'Race', 'Year'), suffix = c("\_3", "\_2")) %>%  
 left\_join(rbind(fp1\_2023,  
 fp1\_2022,  
 fp1\_2021,  
 fp1\_2020,  
 fp1\_2019,  
 fp1\_2018,  
 fp1\_2017,  
 fp1\_2016,  
 fp1\_2015,  
 fp1\_2014) %>%  
 dplyr::select('Driver', 'Race', 'Year', 'Time', 'Time\_secs'), by = c('Driver', 'Race', 'Year')) %>%  
 rename(Time\_1 = Time, Time\_secs\_1 = Time\_secs) %>%   
 relocate(Time\_3, .after = Year)

One of my early questions was: *Do cars get faster with each practice session?*

To begin exploring this question, I will initially focus on 2023 practice data.

I can use left\_join() from the **dplyr** package to merge all practice sessions for the 2023 season.

# Pull 2023 data  
# Scrape FP1 data  
fp1\_2023 <- practice\_session\_scraper(2023, 1)  
# Scrape FP2 data  
fp2\_2023 <- practice\_session\_scraper(2023, 2)  
# Scrape FP2 data  
fp3\_2023 <- practice\_session\_scraper(2023, 3)  
  
# Merge Practices times for 2022  
practice\_times\_2023 <- fp3\_2023 %>%  
 left\_join(fp2\_2023 %>%  
 dplyr::select('Driver', 'Race', 'Year', 'Time', 'Time\_secs'), by = c('Driver', 'Race', 'Year'), suffix = c("\_3", "\_2")) %>%  
 left\_join(fp1\_2023 %>%  
 dplyr::select('Driver', 'Race', 'Year', 'Time', 'Time\_secs'), by = c('Driver', 'Race', 'Year')) %>%  
 rename(Time\_1 = Time, Time\_secs\_1 = Time\_secs) %>%   
 relocate(Time\_3, .after = Year)

**How to join data in R**

Joining data is pretty simple using functions from the **dplyr** package. In dplyr, you will be mutating joins, which adds columns from y to x, matching observations based on the keys. There are four mutating joins:

* inner\_join()
* left\_join()
* right\_join()
* full\_join()

For more information on mutating joins in **dplyr**, visit this link: <https://dplyr.tidyverse.org/reference/mutate-joins.html>

Here’s a nother tip on reordering columns:

**How to relocate a column of a dataframe in R**

Notice that in the final line of the above code, I used the relocate() function to reorder a column. This is not necessary, but can help keep your dataframe more organized.

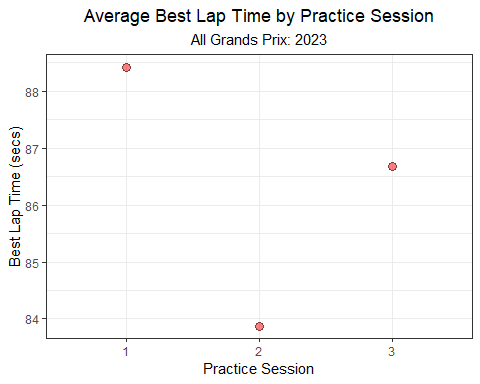
As an example, you could use the following code to move column A after column C in a dataframe:

df %>% relocate(a, .after = c)

For more information on the relocate() function in **dplyr**, visit this link: <https://dplyr.tidyverse.org/reference/relocate.html>

I used the following code to plot the average times by practice session in 2023.

practice\_times\_2023 %>%  
 dplyr::select(Time\_secs\_3, Time\_secs\_2, Time\_secs\_1) %>%  
 pivot\_longer(c(Time\_secs\_3, Time\_secs\_2, Time\_secs\_1), names\_to = 'practice\_session', values\_to = 'time') %>%  
 mutate(practice\_session = str\_remove(practice\_session, 'Time\_secs\_')) %>%  
 ggplot(aes(practice\_session, time)) +  
 stat\_summary(fun.y = mean,   
 geom = "point", pch = 21, col = 'black', fill = 'red', alpha = 0.5, size = 3) +   
 theme\_bw() +  
 labs(y = 'Best Lap Time (secs)',  
 x = 'Practice Session',  
 title = 'Average Best Lap Time by Practice Session',  
 subtitle = 'All Grands Prix: 2023') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5))



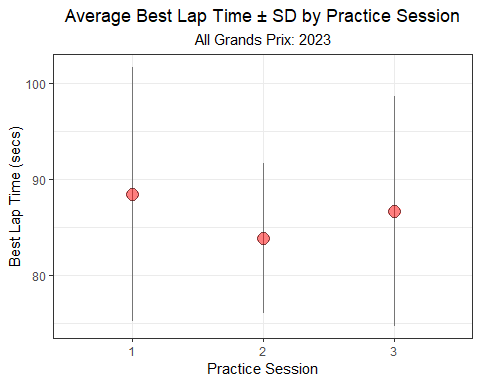
**How to create a categorical scatter plot with a computed mean in R**

If there are many observations per group, you can use the stat\_summary() function to calculate a grouped mean of the variable plotted along the y-axis. Specifically, the argument fun.y = mean specifies that you want to calculate the mean of the y-axis variable. If you wish to *only* plot the mean, include the geom = "point" argument.

For more information on stat\_summary() in **ggplot2**, visit this link: <https://ggplot2.tidyverse.org/reference/stat_summary.html>

When pooling all teams, drivers, and circuits together, it looks like the average time during P1 is slower than both P2 and P3. P3’s average time is slightly slower than P2. But, these times will obviously vary considerably by year, Grand Prix, team, and driver. How much variability surrounds the average times in the figure above? I’ll re-plot that data but also include a standard deviation bar that describes the mean ± 1 standard deviation (SD).

practice\_times\_2023 %>%  
 dplyr::select(Time\_secs\_3, Time\_secs\_2, Time\_secs\_1) %>%  
 pivot\_longer(c(Time\_secs\_3, Time\_secs\_2, Time\_secs\_1), names\_to = 'practice\_session', values\_to = 'time') %>%  
 mutate(practice\_session = str\_remove(practice\_session, 'Time\_secs\_')) %>%  
 ggplot(aes(practice\_session, time)) +  
 stat\_summary(fun.y = mean,  
 fun.ymin = function(x) mean(x) - sd(x),   
 fun.ymax = function(x) mean(x) + sd(x),   
 geom = "pointrange",  
 pch = 21, col = 'black', fill = 'red', alpha = 0.5, size = 1) +  
 theme\_bw() +  
 labs(y = 'Best Lap Time (secs)',  
 x = 'Practice Session',  
 title = 'Average Best Lap Time \u00b1 SD by Practice Session',  
 subtitle = 'All Grands Prix: 2023') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5))



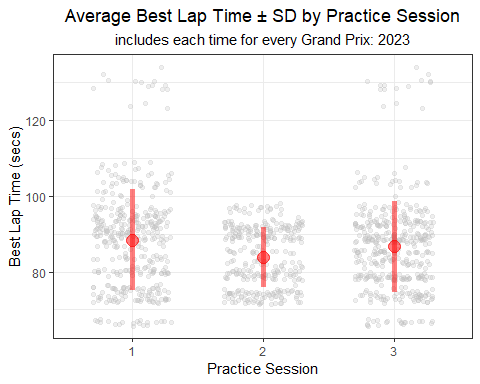
**How to create a categorical scatter plot with a computed mean ± standard deviation in R**

Again, you can use the stat\_summary() function to calculate a grouped mean ± standard deviation of the variable plotted along the y-axis. To plot a computed *point-interval* or *point-range*, specify geom = "pointrange". Additionally, you’ll need to include arguments for the calculated minimum and maximum of that range. In the example above, I plot the mean ± 1 standard deviation using the arguments fun.ymin = function(x) mean(x) - sd(x) and fun.ymax = function(x) mean(x) + sd(x).

For more information on stat\_summary() in **ggplot2**, visit this link: <https://ggplot2.tidyverse.org/reference/stat_summary.html>

Despite the differences in average times between practice sessions, the variability of times within each practice sessions far exceeds the differences between the sessions. This seems obvious. Circuits are very different, not to mention all of the other sources of variability including yearly, team, and differences by driver. Approximately 68% of the data will fall within 1 standard deviation of the mean. So, the entire distribution of practice times extends beyond the intervals in the figure above. If we’d like to know how the entire distributions of times compare, we can include the individual times beneath the mean ± SD.

practice\_times\_2023 %>%  
 dplyr::select(Time\_secs\_3, Time\_secs\_2, Time\_secs\_1) %>%  
 pivot\_longer(c(Time\_secs\_3, Time\_secs\_2, Time\_secs\_1), names\_to = 'practice\_session', values\_to = 'time') %>%  
 mutate(practice\_session = str\_remove(practice\_session, 'Time\_secs\_')) %>%  
 ggplot(aes(practice\_session, time)) +  
 geom\_point(position = position\_jitter(w= 0.3, h = 0), alpha = 0.25, col = 'grey') +  
 theme\_bw() +  
 stat\_summary(fun.y = mean,  
 fun.ymin = function(x) mean(x) - sd(x),   
 fun.ymax = function(x) mean(x) + sd(x),   
 geom = "pointrange",   
 col = 'red', linewidth = 2, size = 1, alpha = 0.5) +  
 labs(y = 'Best Lap Time (secs)',  
 x = 'Practice Session',  
 title = 'Average Best Lap Time \u00b1 SD by Practice Session',  
 subtitle = 'includes each time for every Grand Prix: 2023') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5))



**How to create a strip plot in R**

In a 2015 paper, Tracey Weissgerber proposed packing as much information as possible into a figure. One of her recommendations for plotting a continuous variable across groups is the *strip plot*.

Here is a link to Tracy’s fantastic paper: <https://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.1002128>

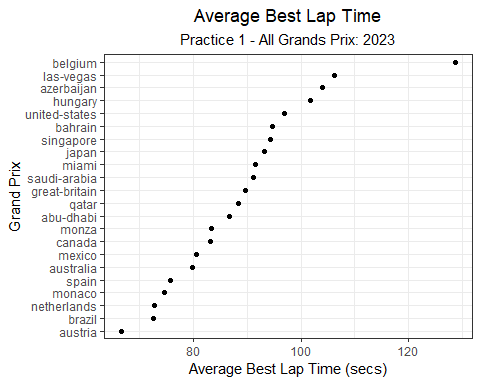
To create a strip plot in **ggplot**, I simply use geom\_point() and include the position = position\_jitter() argument.

I’ll mention more about geom\_point() shortly, but visit this link for more information on point geometries in **ggplot2**: <https://ggplot2.tidyverse.org/reference/geom_point.html>

And, here’s a link to background information on *jittering*: <https://ggplot2.tidyverse.org/reference/position_jitter.html>

Much of this variability can actually be explained away. The largest source of variability among these times is probably driven by differences in circuits. For instance, times at the United States Grand Prix (Circuit of the Americas) are longer than those at the Austrian Grand Prix. In the figure below, I plot the average P1 lap time for each Grand Prix. Clearly, times vary considerably by circuit!

practice\_times\_2023 %>%  
 group\_by(Race) %>%  
 summarize(mean = mean(Time\_secs\_1, na.rm = T)) %>%  
 ggplot(aes(mean, y = fct\_reorder(Race, mean))) +  
 geom\_point() +  
 theme\_bw() +  
 labs(x = 'Average Best Lap Time (secs)',  
 y = 'Grand Prix',  
 title = 'Average Best Lap Time',  
 subtitle = 'Practice 1 - All Grands Prix: 2023') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5))



**How to create a categorical scatter plot in R**

I used the simple point geometry (geom\_point()) in **ggplot2** to create the plot above. geom\_point() is most commonly used to create scatterplots that display the relationship between two continuous variables. However, you can also use them to plot a continuous variables across groups.

For more information on point geometries in **ggplot2**, visit this link: <https://ggplot2.tidyverse.org/reference/geom_point.html>

Building on this idea, let’s explore the historical practice times for a single Grand Prix: Bahrain. Below, I’ll calculate the standard deviation for each practice session, including all Grands Prix. Then, I’ll calculate the standard deviations for the Bahrain Grand Prix only, which is about 4-5 times smaller.

# All Grands Prix  
practice\_times\_2023 %>%  
 pivot\_longer(c(Time\_secs\_3, Time\_secs\_2, Time\_secs\_1), names\_to = 'practice\_session', values\_to = 'time') %>%  
 mutate(practice\_session = str\_remove(practice\_session, 'Time\_secs\_')) %>%  
 group\_by(practice\_session) %>%   
 summarize(sd = round(sd(time, na.rm = T), 2)) %>%  
 gt() %>%  
 tab\_options(column\_labels.font.weight = "bold") %>%  
 tab\_style(  
 style = cell\_text(align = "center"),  
 locations = cells\_body(columns = c(practice\_session, sd))) %>%  
 cols\_align(align = "center", columns = c(practice\_session, sd)) %>%   
 tab\_header(  
 title = md("\*\*data from all Grands Prix\*\*"),  
 subtitle = md("2023")  
 )

Table 1: **data from all Grands Prix**

2023

| practice\_session | sd |
| --- | --- |
| 1 | 13.28 |
| 2 | 7.84 |
| 3 | 11.99 |

# Bahrain Only  
practice\_times\_2023 %>%  
 filter(Race %in% c('bahrain')) %>%   
 pivot\_longer(c(Time\_secs\_3, Time\_secs\_2, Time\_secs\_1), names\_to = 'practice\_session', values\_to = 'time') %>%  
 mutate(practice\_session = str\_remove(practice\_session, 'Time\_')) %>%  
 group\_by(practice\_session) %>%   
 summarize(sd = round(sd(time, na.rm = T), 2)) %>%  
 gt() %>%   
 tab\_options(column\_labels.font.weight = "bold") %>%  
 tab\_style(  
 style = cell\_text(align = "center"),  
 locations = cells\_body(columns = c(practice\_session, sd))) %>%  
 cols\_align(align = "center", columns = c(practice\_session, sd)) %>%   
 tab\_header(  
 title = md("\*\*data from the Bahrain Grand Prix only\*\*"),  
 subtitle = md("2023")  
 )

Table 1: **data from the Bahrain Grand Prix only**

2023

| practice\_session | sd |
| --- | --- |
| secs\_1 | 0.89 |
| secs\_2 | 0.53 |
| secs\_3 | 0.50 |

**How to add a header to a table in R**

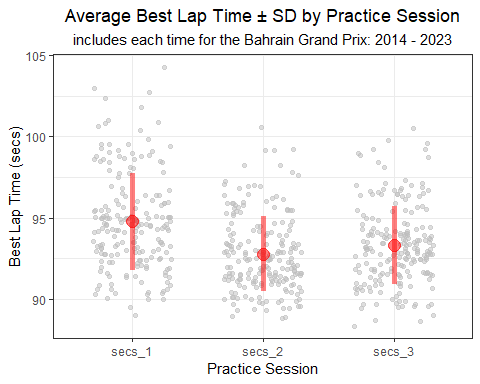
It is very easy to add a header to the tables above. To add a title and/or subtitle, simply pass the title and subtitle arguments to the tab\_header() function. The table header is positioned just above the column labels in the **gt** table.

To **bold** or *italicize* the title or subtitle in the header, **gt** actually allows the flexibility to use Markdown or HTML formatting. To use Markdown or HTML formatting, wrap your title and/or subtitle text in md() or html() functions, respectively.

For more information, visit this link: <https://gt.rstudio.com/reference/tab_header.html>

Below, I’ll plot the average practice times for the Bahrain Grand Prix, along with all of the underlying times (grey points).

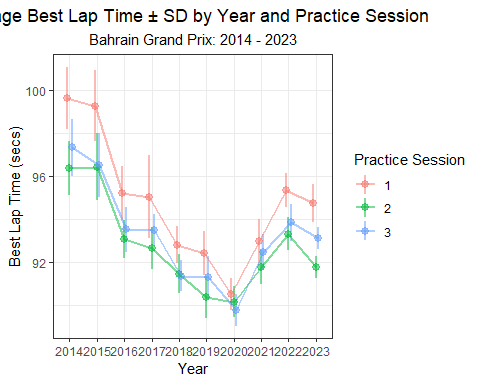
practice\_times %>%  
 filter(Race %in% c('bahrain')) %>%   
 pivot\_longer(c(Time\_secs\_3, Time\_secs\_2, Time\_secs\_1), names\_to = 'practice\_session', values\_to = 'time') %>%  
 mutate(practice\_session = str\_remove(practice\_session, 'Time\_')) %>%  
 ggplot(aes(practice\_session, time)) +  
 geom\_point(position = position\_jitter(w= 0.3, h = 0), alpha = 0.5, col = 'grey') +  
 theme\_bw() +  
 stat\_summary(fun.y = mean,  
 fun.ymin = function(x) mean(x) - sd(x),   
 fun.ymax = function(x) mean(x) + sd(x),   
 geom = "pointrange",   
 col = 'red', linewidth = 2, size = 1, alpha = 0.5) +  
 labs(y = 'Best Lap Time (secs)',  
 x = 'Practice Session',  
 title = 'Average Best Lap Time \u00b1 SD by Practice Session',  
 subtitle = 'includes each time for the Bahrain Grand Prix: 2014 - 2023') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5))



Judging by this figure, P1 times appear to be significantly slower than times during P2 and P3 in Bahrain. And while the average for P2 is slightly faster than P3, the fastest times were set during P3. Interesting.

If we plot the times by year, but group the times by practice session, we can see that times steadily improve each year until 2021, at which point they begin to get a bit slower.

practice\_times %>%  
 filter(Race %in% c('bahrain')) %>%   
 pivot\_longer(c(Time\_secs\_3, Time\_secs\_2, Time\_secs\_1), names\_to = 'practice\_session', values\_to = 'time') %>%  
 mutate(practice\_session = str\_remove(practice\_session, 'Time\_secs\_'),  
 Year = factor(Year)) %>%  
 ggplot(aes(Year, time,   
 group = practice\_session, col = practice\_session)) +  
 theme\_bw() +  
 stat\_summary(fun.y = mean,  
 fun.ymin = function(x) mean(x) - sd(x),   
 fun.ymax = function(x) mean(x) + sd(x),   
 geom = "pointrange",   
 linewidth = 1, size = 0.5, alpha = 0.5,  
 position = position\_dodge(w = 0.25)) +  
 stat\_summary(fun.y = mean,  
 geom = "line",   
 linewidth = 1, size = 0.5, alpha = 0.5,  
 position = position\_dodge(w = 0.25)) +  
 labs(x = 'Year',  
 y = 'Best Lap Time (secs)',  
 col = 'Practice Session',  
 title = 'Average Best Lap Time \u00b1 SD by Year and Practice Session',  
 subtitle = 'Bahrain Grand Prix: 2014 - 2023') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5))



**How to create a line plot with a computed mean ± standard deviation in R**

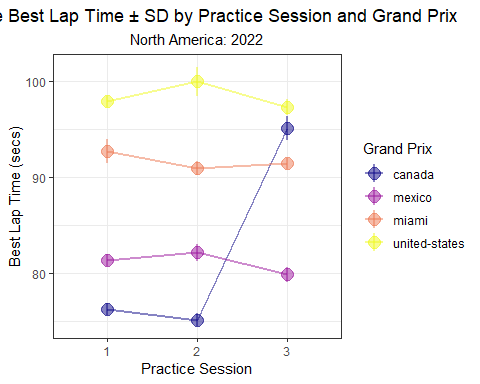
Like the previous examples with points and point-ranges, you can also use the stat\_summary() function to create a line plot using grouped means ± standard deviation. Simply pass geom = "line" as an argument to the stat\_summary() function. In the figure above, I wanted to plot grouped lines, so I also passed group = practice\_session and col = practice\_session as arguments to the original ggplot() call.

In addition to these summary lines, I also included the point-ranges using the same approach as previous figures (i.e. a separate stat\_summary() call using the geom = "pointrange" argument.).

For more information on stat\_summary() in **ggplot2**, visit this link: <https://ggplot2.tidyverse.org/reference/stat_summary.html>

Knowing that times vary by Grand Prix and year, let’s take a deeper look at differences in times within a single year (2022) and for one continent (North America). Below, I’ll plot the average practice times across North American Grands Prix for 2022 only.

practice\_times %>%  
 filter(Year == 2022) %>%   
 pivot\_longer(c(Time\_secs\_3, Time\_secs\_2, Time\_secs\_1), names\_to = 'practice\_session', values\_to = 'time') %>%  
 mutate(practice\_session = str\_remove(practice\_session, 'Time\_secs\_'),  
 Year = factor(Year)) %>%  
 filter(Race %in% c('canada', 'miami', 'united-states', 'mexico', 'brazil')) %>%   
 ggplot(aes(practice\_session, time,   
 group = Race, col = Race)) +  
 theme\_bw() +  
 stat\_summary(fun.y = mean,  
 fun.ymin = function(x) mean(x) - sd(x),   
 fun.ymax = function(x) mean(x) + sd(x),   
 geom = "pointrange",   
 linewidth = 1, size = 1, alpha = 0.5) +  
 stat\_summary(fun.y = mean,  
 geom = "line",   
 linewidth = 1, size = 1, alpha = 0.5) +  
 labs(y = 'Best Lap Time (secs)',  
 x = 'Practice Session',  
 col = 'Grand Prix',  
 title = 'Average Best Lap Time \u00b1 SD by Practice Session and Grand Prix',  
 subtitle = 'North America: 2022') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5)) +  
 scale\_color\_viridis\_d(option = 'plasma')



**How to use colorblind-friendly palettes in R**

As someone with color blindness, I find the use of colorblind-friendly palettes very helpful! I often rely on the viridis color scales for **ggplot**. The viridis scales provides various color mappings that are designed to be perceived by viewers with common forms of color blindness.

For a continuous color mapping, use scale\_color\_viridis\_c().

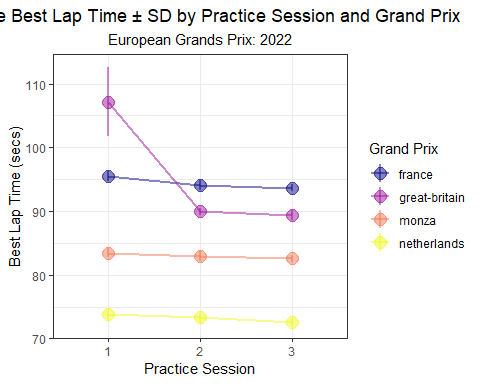
For a discrete color mapping, use scale\_color\_viridis\_d().

For more information about viridis, visit this link: <https://ggplot2.tidyverse.org/reference/scale_viridis.html>

For more context about perceptually uniform color scales, see this link: <https://bids.github.io/colormap/>.

Interestingly, there doesn’t seem to be a definitive pattern in practice times across these four North American races in 2022. Compared this to four European races, where there is a clear decrease in times from P1 to P3.

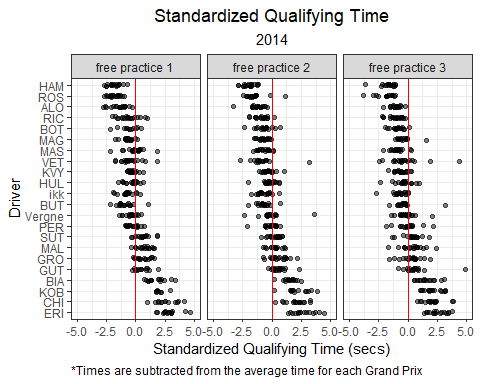
practice\_times %>%  
 filter(Year == 2022) %>%   
 pivot\_longer(c(Time\_secs\_3, Time\_secs\_2, Time\_secs\_1), names\_to = 'practice\_session', values\_to = 'time') %>%  
 mutate(practice\_session = str\_remove(practice\_session, 'Time\_secs\_'),  
 Year = factor(Year)) %>%  
 filter(Race %in% c('france', 'monza', 'great-britain', 'netherlands')) %>%   
 ggplot(aes(practice\_session, time,   
 group = Race, col = Race)) +  
 theme\_bw() +  
 stat\_summary(fun.y = mean,  
 fun.ymin = function(x) mean(x) - sd(x),   
 fun.ymax = function(x) mean(x) + sd(x),   
 geom = "pointrange",   
 linewidth = 1, size = 1, alpha = 0.5) +  
 stat\_summary(fun.y = mean,  
 geom = "line",   
 linewidth = 1, size = 1, alpha = 0.5) +  
 labs(y = 'Best Lap Time (secs)',  
 x = 'Practice Session',  
 col = 'Grand Prix',  
 title = 'Average Best Lap Time \u00b1 SD by Practice Session and Grand Prix',  
 subtitle = 'European Grands Prix: 2022') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5)) +  
 scale\_color\_viridis\_d(option = 'plasma')



## 2.3 Practice Times by Driver

Perhaps the most fun way to visualize data is to plot the times by Driver. Let’s first focus on the 2014 season, which Mercedes dominated. The two Mercedes drivers, Lewis Hamilton and Nico Rosberg, are clearly much faster than all other drivers during P2 and P3. During this season, Australia appears to be the circuit where Mercedes had the largest standardized gap in P3.

practice\_times %>%  
 filter(Year == 2014) %>%   
 pivot\_longer(c(Time\_secs\_3, Time\_secs\_2, Time\_secs\_1), names\_to = 'practice\_session', values\_to = 'time') %>%  
 mutate(practice\_session = str\_remove(practice\_session, 'Time\_secs\_'),  
 Year = factor(Year)) %>%  
 group\_by(Race, practice\_session) %>%  
 mutate(track\_mean = mean(time, na.rm = T),  
 Time\_std\_track = time - track\_mean) %>%   
 ungroup() %>%   
 group\_by(Driver) %>%   
 mutate(driver\_n = n(),  
 mean = mean(Time\_std\_track, na.rm = T)) %>%  
 ungroup() %>%  
 dplyr::filter(driver\_n > 15) %>%   
 mutate(practice\_session = paste0('free practice ', practice\_session)) %>%   
 ggplot(aes(Time\_std\_track, y = fct\_reorder(Driver, desc(mean)))) +  
 geom\_point(position = position\_jitter(w = 0, h = 0.1), alpha = 0.5) +  
 theme\_bw() +  
 labs(x = 'Standardized Qualifying Time (secs)',  
 y = 'Driver',  
 title = 'Standardized Qualifying Time',  
 subtitle = '2014',  
 caption = '\*Times are subtracted from the average time for each Grand Prix') +  
 geom\_vline(xintercept = 0, col = 'red') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 facet\_wrap(~ practice\_session) +  
 xlim(-5, 5)



**Jittering data points in R**

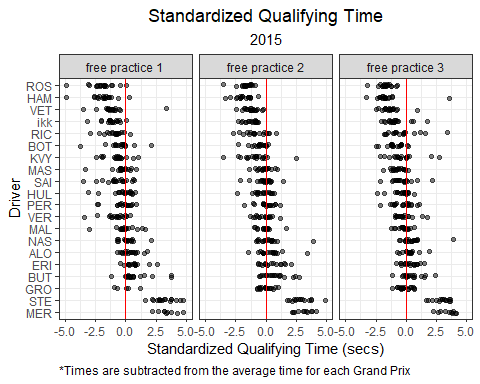
I’ll often use *jittering* to slightly separate data points that would otherwise be plotted on top of each other. Interestingly (and counterintuitively!), using a *jitter* argument in **ggplot** to add small amounts of random noise to a plot can actually make the plot easier to interpret.

To add *jittering* to a plot, simply pass position = position\_jitter() as an argument inside geom\_point(). In the example above, I specify that I wish to add a small degree of *jittering* vertically and **no** *jittering* horizontally using position = position\_jitter(w = 0, h = 0.1).

<https://ggplot2.tidyverse.org/reference/position_jitter.html>

Hamilton and Rosberg again dominated practice sessions 2 and 3 during the 2015 season. During 2015, Malaysia seemed to be a particularly strong circuit for Mercedes.

practice\_times %>%  
 filter(Year == 2015) %>%   
 pivot\_longer(c(Time\_secs\_3, Time\_secs\_2, Time\_secs\_1), names\_to = 'practice\_session', values\_to = 'time') %>%  
 mutate(practice\_session = str\_remove(practice\_session, 'Time\_secs\_'),  
 Year = factor(Year)) %>%  
 group\_by(Race, practice\_session) %>%  
 mutate(track\_mean = mean(time, na.rm = T),  
 Time\_std\_track = time - track\_mean) %>%   
 ungroup() %>%   
 group\_by(Driver) %>%   
 mutate(driver\_n = n(),  
 mean = mean(Time\_std\_track, na.rm = T)) %>%  
 ungroup() %>%  
 dplyr::filter(driver\_n > 15) %>%   
 mutate(practice\_session = paste0('free practice ', practice\_session)) %>%   
 ggplot(aes(Time\_std\_track, y = fct\_reorder(Driver, desc(mean)))) +  
 geom\_point(position = position\_jitter(w = 0, h = 0.1), alpha = 0.5) +  
 theme\_bw() +  
 labs(x = 'Standardized Qualifying Time (secs)',  
 y = 'Driver',  
 title = 'Standardized Qualifying Time',  
 subtitle = '2015',  
 caption = '\*Times are subtracted from the average time for each Grand Prix') +  
 geom\_vline(xintercept = 0, col = 'red') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 facet\_wrap(~ practice\_session) +  
 xlim(-5, 5)



**Adding a reference line in R**

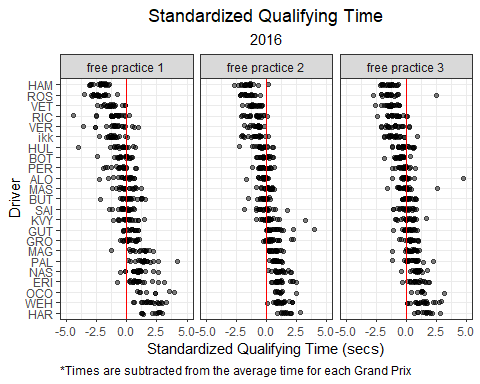
In the plot above, the solid red vertical line serves a reference line for a standardized time of 0 (i.e. the average time at a Grand Prix). Sometimes, it’s helpful to add a reference line to a figure. Within **ggplot2**, there are three common geometries that are used to add references lines to a plot:

* geom\_hline() adds a horizontal reference line. Specify the yintercept.
* geom\_vline() adds a vertical reference line. Specify the xintercept.
* geom\_abline() adds a diagonal reference line. Specify the slope.

For more information see this link: <https://ggplot2.tidyverse.org/reference/geom_abline.html>

The Mercedes domination of practice sessions continued through 2016, but Ferrari has started chipping away a the gap.

practice\_times %>%  
 filter(Year == 2016) %>%   
 pivot\_longer(c(Time\_secs\_3, Time\_secs\_2, Time\_secs\_1), names\_to = 'practice\_session', values\_to = 'time') %>%  
 mutate(practice\_session = str\_remove(practice\_session, 'Time\_secs\_'),  
 Year = factor(Year)) %>%  
 group\_by(Race, practice\_session) %>%  
 mutate(track\_mean = mean(time, na.rm = T),  
 Time\_std\_track = time - track\_mean) %>%   
 ungroup() %>%   
 group\_by(Driver) %>%   
 mutate(driver\_n = n(),  
 mean = mean(Time\_std\_track, na.rm = T)) %>%  
 ungroup() %>%  
 dplyr::filter(driver\_n > 15) %>%   
 mutate(practice\_session = paste0('free practice ', practice\_session)) %>%   
 ggplot(aes(Time\_std\_track, y = fct\_reorder(Driver, desc(mean)))) +  
 geom\_point(position = position\_jitter(w = 0, h = 0.1), alpha = 0.5) +  
 theme\_bw() +  
 labs(x = 'Standardized Qualifying Time (secs)',  
 y = 'Driver',  
 title = 'Standardized Qualifying Time',  
 subtitle = '2016',  
 caption = '\*Times are subtracted from the average time for each Grand Prix') +  
 geom\_vline(xintercept = 0, col = 'red') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 facet\_wrap(~ practice\_session) +  
 xlim(-5, 5)



**Faceting Plots**

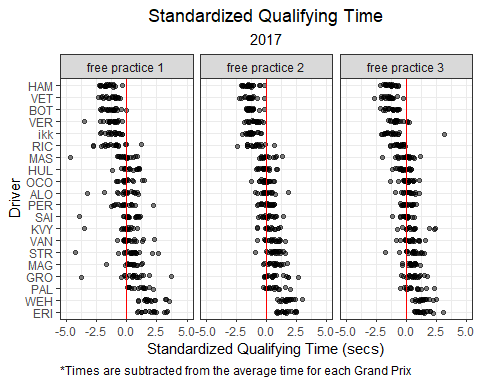
In the previous few figures, you will notice that the data is split into panes, or *facets*. Faceting splits a larger graph into two or more smaller graphs that are organized within a grid. Each *facet* within the grid will display the same style of graph for a designated group of the dataset.

In the example above, I use the facet\_wrap() function to split the graph into facets representing each practice session using the following function and argument: facet\_wrap(~ practice\_session)

For more information on the facet\_wrap() function, see this link: <https://ggplot2.tidyverse.org/reference/facet_wrap.html>

By 2017, Ferrari had seemingly closed the gap to Mercedes. Sebastian Vettel led the driver’s championship for the first 12 rounds of the season, and we can see that Vettel’s practice times are comparable to those posted by Hamilton. in 2017, the Ferrari SF70H was thought to be initially a more consistent car in race trim, and Vettel’s P2 times seem to support this claim.

practice\_times %>%  
 filter(Year == 2017) %>%   
 pivot\_longer(c(Time\_secs\_3, Time\_secs\_2, Time\_secs\_1), names\_to = 'practice\_session', values\_to = 'time') %>%  
 mutate(practice\_session = str\_remove(practice\_session, 'Time\_secs\_'),  
 Year = factor(Year)) %>%  
 group\_by(Race, practice\_session) %>%  
 mutate(track\_mean = mean(time, na.rm = T),  
 Time\_std\_track = time - track\_mean) %>%   
 ungroup() %>%   
 group\_by(Driver) %>%   
 mutate(driver\_n = n(),  
 mean = mean(Time\_std\_track, na.rm = T)) %>%  
 ungroup() %>%  
 dplyr::filter(driver\_n > 15) %>%   
 mutate(practice\_session = paste0('free practice ', practice\_session)) %>%   
 ggplot(aes(Time\_std\_track, y = fct\_reorder(Driver, desc(mean)))) +  
 geom\_point(position = position\_jitter(w = 0, h = 0.1), alpha = 0.5) +  
 theme\_bw() +  
 labs(x = 'Standardized Qualifying Time (secs)',  
 y = 'Driver',  
 title = 'Standardized Qualifying Time',  
 subtitle = '2017',  
 caption = '\*Times are subtracted from the average time for each Grand Prix') +  
 geom\_vline(xintercept = 0, col = 'red') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 facet\_wrap(~ practice\_session) +  
 xlim(-5, 5)



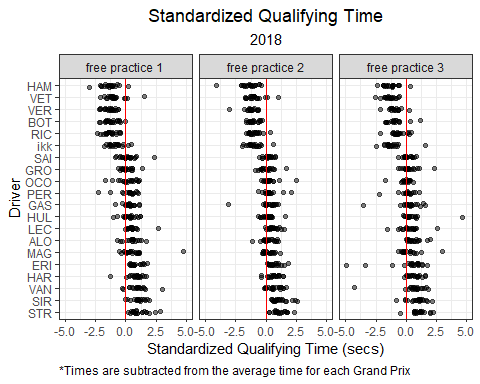
**Sorting an axis in ggplot**

You may have noticed that the last few plots have a y-axis that is sorted by the values along the x-axis. **ggplot2** does not sort an axis in this manner by default. However, this can easily be accomplished using the fct\_reorder() function. In the plots above, I sort the y-axis (Drivers) by their average standardized time in descending order.

For more information on the fct\_reorder() function, see this link: <https://forcats.tidyverse.org/reference/fct_reorder.html>

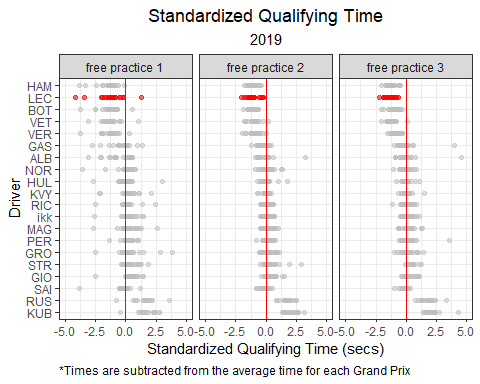
In 2018, Vettel’s distribution of practice times were not quite as fast. Hamilton’s times during P2 seemed to stand apart from the field. Interestingly, Max Verstappen’s times during P2 were comparable to those posted by Mercedes’ other driver, Valtteri Bottas.

practice\_times %>%  
 filter(Year == 2018) %>%   
 pivot\_longer(c(Time\_secs\_3, Time\_secs\_2, Time\_secs\_1), names\_to = 'practice\_session', values\_to = 'time') %>%  
 mutate(practice\_session = str\_remove(practice\_session, 'Time\_secs\_'),  
 Year = factor(Year)) %>%  
 group\_by(Race, practice\_session) %>%  
 mutate(track\_mean = mean(time, na.rm = T),  
 Time\_std\_track = time - track\_mean) %>%   
 ungroup() %>%   
 group\_by(Driver) %>%   
 mutate(driver\_n = n(),  
 mean = mean(Time\_std\_track, na.rm = T)) %>%  
 ungroup() %>%  
 dplyr::filter(driver\_n > 15) %>%   
 mutate(practice\_session = paste0('free practice ', practice\_session)) %>%   
 ggplot(aes(Time\_std\_track, y = fct\_reorder(Driver, desc(mean)))) +  
 geom\_point(position = position\_jitter(w = 0, h = 0.1), alpha = 0.5) +  
 theme\_bw() +  
 labs(x = 'Standardized Qualifying Time (secs)',  
 y = 'Driver',  
 title = 'Standardized Qualifying Time',  
 subtitle = '2018',  
 caption = '\*Times are subtracted from the average time for each Grand Prix') +  
 geom\_vline(xintercept = 0, col = 'red') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 facet\_wrap(~ practice\_session) +  
 xlim(-5, 5)



During the 2019 season, Ferrari scored six consecutive pole positions between the Belgian and Mexican Grands Prix. However, after the FIA issued a technical directive reminding competitors of the regulations regarding fuel sensors, Ferrari’s performance faded and they failed to score a pole position or race win for the remainder of the season. In the plot below, we can see that Charles LeClerc’s distribution is fairly wide, which is likely indicative of this shift in performance.

practice\_times %>%  
 filter(Year == 2019) %>%   
 pivot\_longer(c(Time\_secs\_3, Time\_secs\_2, Time\_secs\_1), names\_to = 'practice\_session', values\_to = 'time') %>%  
 mutate(practice\_session = str\_remove(practice\_session, 'Time\_secs\_'),  
 Year = factor(Year)) %>%  
 group\_by(Race, practice\_session) %>%  
 mutate(track\_mean = mean(time, na.rm = T),  
 Time\_std\_track = time - track\_mean) %>%   
 ungroup() %>%   
 group\_by(Driver) %>%   
 mutate(driver\_n = n(),  
 mean = mean(Time\_std\_track, na.rm = T)) %>%  
 ungroup() %>%  
 dplyr::filter(driver\_n > 15) %>%   
 mutate(practice\_session = paste0('free practice ', practice\_session)) %>%   
 ggplot(aes(Time\_std\_track, y = fct\_reorder(Driver, desc(mean)))) +  
 geom\_point(alpha = 0.5, col = 'red') +  
 gghighlight(Driver == 'LEC', calculate\_per\_facet = T) +   
 theme\_bw() +  
 labs(x = 'Standardized Qualifying Time (secs)',  
 y = 'Driver',  
 title = 'Standardized Qualifying Time',  
 subtitle = '2019',  
 caption = '\*Times are subtracted from the average time for each Grand Prix') +  
 geom\_vline(xintercept = 0, col = 'red') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 facet\_wrap(~ practice\_session) +  
 xlim(-5, 5)



**Highlighting data in ggplot**

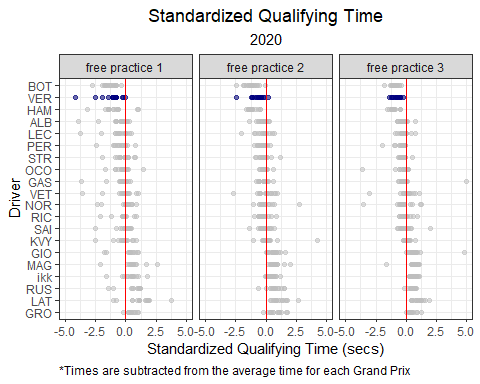
In the plot above, I highlight Charles Leclerc’s data using the gghighlight() function from the **gghighlight** package. This function is version easy to use! Because this plot also utilizes facet\_wrap() and I wanted to highlight each facet individually, I set the calculate\_per\_facet argument to TRUE.

To highlight Charles Leclerc, I used the following line of code: gghighlight(Abbr == 'LEC', calculate\_per\_facet = T).

For more information on the **gghighlight** package, visit this link: <https://cran.r-project.org/web/packages/gghighlight/vignettes/gghighlight.html>

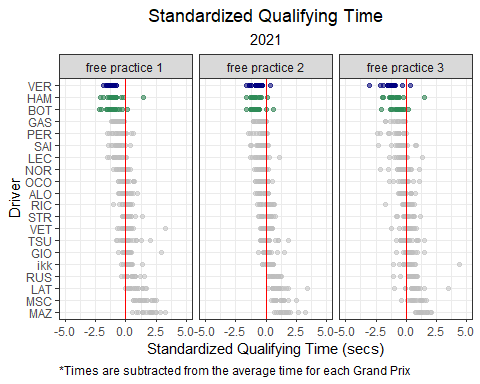
In 2020, Max Verstappen competed with the two Mercedes Drivers (Hamilton and Bottas) in P3 (qualifying simulations), but struggled in the longer runs of P2. As we can see in the plot below, Verstappen’s P3 pace was the closest to Mercedes, and far quicker than his teammate Alex Albon.

practice\_times %>%  
 filter(Year == 2020) %>%   
 pivot\_longer(c(Time\_secs\_3, Time\_secs\_2, Time\_secs\_1), names\_to = 'practice\_session', values\_to = 'time') %>%  
 mutate(practice\_session = str\_remove(practice\_session, 'Time\_secs\_'),  
 Year = factor(Year)) %>%  
 group\_by(Race, practice\_session) %>%  
 mutate(track\_mean = mean(time, na.rm = T),  
 Time\_std\_track = time - track\_mean) %>%   
 ungroup() %>%   
 group\_by(Driver) %>%   
 mutate(driver\_n = n(),  
 mean = mean(Time\_std\_track, na.rm = T)) %>%  
 ungroup() %>%  
 dplyr::filter(driver\_n > 15) %>%   
 mutate(practice\_session = paste0('free practice ', practice\_session)) %>%   
 ggplot(aes(Time\_std\_track, y = fct\_reorder(Driver, desc(mean)))) +  
 geom\_point(alpha = 0.5, col = 'navy') +  
 gghighlight(Driver == 'VER', calculate\_per\_facet = T) +   
 theme\_bw() +  
 labs(x = 'Standardized Qualifying Time (secs)',  
 y = 'Driver',  
 title = 'Standardized Qualifying Time',  
 subtitle = '2020',  
 caption = '\*Times are subtracted from the average time for each Grand Prix') +  
 geom\_vline(xintercept = 0, col = 'red') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 facet\_wrap(~ practice\_session) +  
 xlim(-5, 5)



For the first time since 2014, Mercedes faced a legitimate challenge to their dominance. While Ferrari had their moments during the turbo-hybrid era, Red Bull and Max Verstappen competed with Mercedes during every session of the season.

practice\_times %>%  
 filter(Year == 2021) %>%   
 pivot\_longer(c(Time\_secs\_3, Time\_secs\_2, Time\_secs\_1), names\_to = 'practice\_session', values\_to = 'time') %>%  
 mutate(practice\_session = str\_remove(practice\_session, 'Time\_secs\_'),  
 Year = factor(Year)) %>%  
 group\_by(Race, practice\_session) %>%  
 mutate(track\_mean = mean(time, na.rm = T),  
 Time\_std\_track = time - track\_mean) %>%   
 ungroup() %>%   
 group\_by(Driver) %>%   
 mutate(driver\_n = n(),  
 mean = mean(Time\_std\_track, na.rm = T)) %>%  
 ungroup() %>%  
 dplyr::filter(driver\_n > 15) %>%   
 mutate(practice\_session = paste0('free practice ', practice\_session)) %>%   
 ggplot(aes(Time\_std\_track, y = fct\_reorder(Driver, desc(mean)), col = Driver)) +  
 geom\_point(alpha = 0.5) +  
 gghighlight(Driver %in% c('VER', 'HAM', 'BOT'), calculate\_per\_facet = T) +   
 theme\_bw() +  
 labs(x = 'Standardized Qualifying Time (secs)',  
 y = 'Driver',  
 title = 'Standardized Qualifying Time',  
 subtitle = '2021',  
 caption = '\*Times are subtracted from the average time for each Grand Prix') +  
 geom\_vline(xintercept = 0, col = 'red') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 facet\_wrap(~ practice\_session) +  
 xlim(-5, 5) +  
 scale\_color\_manual('', values = c('seagreen', 'seagreen', 'navy'))



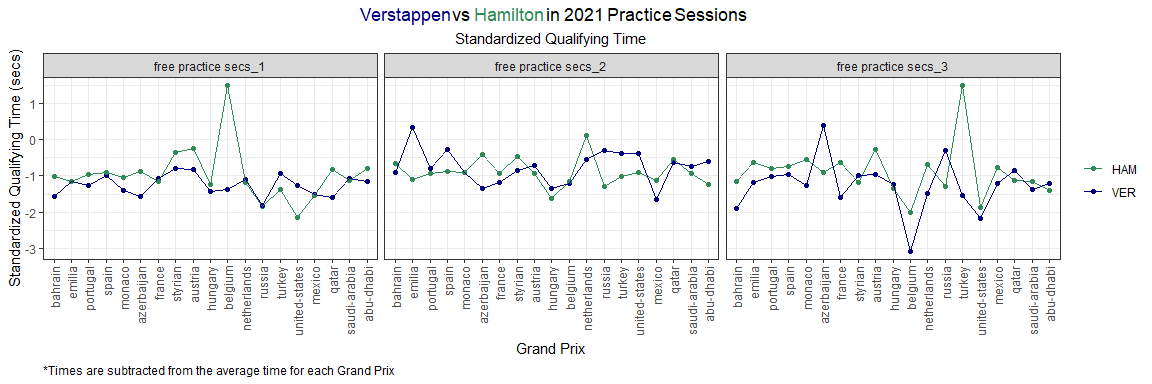
**Custom discrete color scales in ggplot**

In the plot above, I created a custom color scale that corresponds to two teams: Mercedes and Red Bull. The scale\_color\_manual() function allows you to specify your own set of mappings from levels/groups in the data to color values.

For more information on this, visit this link: <https://ggplot2.tidyverse.org/reference/scale_manual.html>

The plot below shows the head-to-head practice battles for Verstappen vs Hamilton in 2021. This figure also seems to suggest that Hamilton had the edge in race pace (P2), while Verstappen had the advantage in qualifying simulations (P3).

practice\_times %>%  
 filter(Year == 2021) %>%   
 mutate(round = match(Race, unique(Race))) %>%  
 ungroup() %>%  
 pivot\_longer(c(Time\_secs\_3, Time\_secs\_2, Time\_secs\_1), names\_to = 'practice\_session', values\_to = 'time') %>%  
 mutate(practice\_session = str\_remove(practice\_session, 'Time\_'),  
 Year = factor(Year)) %>%  
 group\_by(Race, practice\_session) %>%  
 mutate(track\_mean = mean(time, na.rm = T),  
 Time\_std\_track = time - track\_mean) %>%   
 ungroup() %>%   
 group\_by(Driver) %>%   
 mutate(driver\_n = n(),  
 mean = mean(Time\_std\_track, na.rm = T)) %>%  
 ungroup() %>%  
 dplyr::filter(driver\_n > 15) %>%   
 mutate(practice\_session = paste0('free practice ', practice\_session)) %>%   
 filter(Driver %in% c('VER', 'HAM')) %>%  
 ggplot(aes(x = fct\_reorder(Race, round), y = Time\_std\_track,  
 group = Driver, col = Driver)) +  
 geom\_point() +  
 geom\_path() +   
 theme\_bw() +  
 facet\_wrap(~ practice\_session) +   
 theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1),  
 plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 scale\_color\_manual('', values = c('seagreen', 'navy')) +  
 labs(y = 'Standardized Qualifying Time (secs)',  
 x = 'Grand Prix',  
 title = title\_color\_coder("", "Verstappen", 'navy', " vs ", "Hamilton", 'seagreen'," in 2021 Practice Sessions"),  
 subtitle = 'Standardized Qualifying Time',  
 caption = '\*Times are subtracted from the average time for each Grand Prix')+  
 theme(plot.title = ggtext::element\_markdown()) ## render the provided text as markdown/html



**Color-coding the title in ggplot**

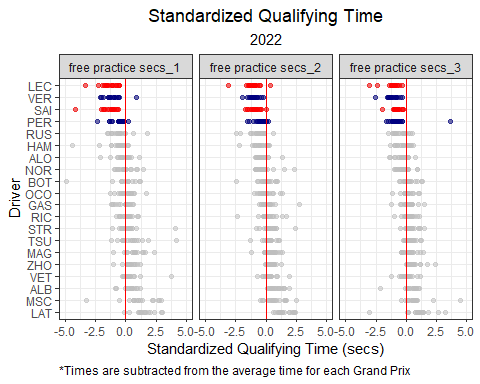
The **ggtext** package provides Markdown and HTML rendering for **ggplot2**. Creating color-coded titles in **ggplot** can get tedious if there are many different colors being used, so I created a simple function (title\_color\_coder()) to add color to titles or subtitles in a **ggplot**.

Be sure to add the following function to **ggplot** code: theme(plot.title = ggtext::element\_markdown())

For more information on the **ggtext** package, visit this link: <https://cran.r-project.org/web/packages/ggtext/readme/README.html>

The 2022 Formula 1 season brought an overhaul of the technical regulations. The technical regulations reintroduced the use of ground effects for the first time since 1983. Additionally, teams now had to abide by a financial cost cap. These changes upset the dominance of Mercedes, and seemingly ushered in a new era of dominance by Red Bull. While Ferrari often had the fastest qualifying pace, Red Bull enjoyed a considerable advantage in race pace.

practice\_times %>%  
 filter(Year == 2022) %>%   
 pivot\_longer(c(Time\_secs\_3, Time\_secs\_2, Time\_secs\_1), names\_to = 'practice\_session', values\_to = 'time') %>%  
 mutate(practice\_session = str\_remove(practice\_session, 'Time\_'),  
 Year = factor(Year)) %>%  
 group\_by(Race, practice\_session) %>%  
 mutate(track\_mean = mean(time, na.rm = T),  
 Time\_std\_track = time - track\_mean) %>%   
 ungroup() %>%   
 group\_by(Driver) %>%   
 mutate(driver\_n = n(),  
 mean = mean(Time\_std\_track, na.rm = T)) %>%  
 ungroup() %>%  
 dplyr::filter(driver\_n > 15) %>%   
 mutate(practice\_session = paste0('free practice ', practice\_session)) %>%   
 ggplot(aes(Time\_std\_track, y = fct\_reorder(Driver, desc(mean)), col = Driver)) +  
 geom\_point(alpha = 0.5) +  
 gghighlight(Driver %in% c('VER', 'LEC', 'SAI', 'PER'), calculate\_per\_facet = T) +  
 theme\_bw() +  
 labs(x = 'Standardized Qualifying Time (secs)',  
 y = 'Driver',  
 title = 'Standardized Qualifying Time',  
 subtitle = '2022',  
 caption = '\*Times are subtracted from the average time for each Grand Prix') +  
 geom\_vline(xintercept = 0, col = 'red') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 facet\_wrap(~ practice\_session) +  
 xlim(-5, 5) +  
 scale\_color\_manual('', values = c('red', 'navy', 'red', 'navy'))

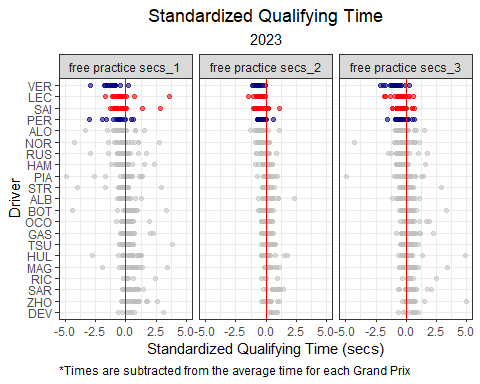


**How to rotate axis text in ggplot**

At times, you may wish to rotate the text of labels along the x- or y-axis. To rotate the x-axis labels in the plot above, I simply added the following function to the **ggplot**: theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1))

The rich get richer, and the fast got faster in 2023. Max Verstappen was even more dominant in his third title-winning season. Compared to the previous year, Max was even quicker during practice sessions during 2023.

practice\_times %>%  
 filter(Year == 2023) %>%   
 pivot\_longer(c(Time\_secs\_3, Time\_secs\_2, Time\_secs\_1), names\_to = 'practice\_session', values\_to = 'time') %>%  
 mutate(practice\_session = str\_remove(practice\_session, 'Time\_'),  
 Year = factor(Year)) %>%  
 group\_by(Race, practice\_session) %>%  
 mutate(track\_mean = mean(time, na.rm = T),  
 Time\_std\_track = time - track\_mean) %>%   
 ungroup() %>%   
 group\_by(Driver) %>%   
 mutate(driver\_n = n(),  
 mean = mean(Time\_std\_track, na.rm = T)) %>%  
 ungroup() %>%  
 dplyr::filter(driver\_n > 15) %>%   
 mutate(practice\_session = paste0('free practice ', practice\_session)) %>%   
 ggplot(aes(Time\_std\_track, y = fct\_reorder(Driver, desc(mean)), col = Driver)) +  
 geom\_point(alpha = 0.5) +  
 gghighlight(Driver %in% c('VER', 'LEC', 'SAI', 'PER'), calculate\_per\_facet = T) +  
 theme\_bw() +  
 labs(x = 'Standardized Qualifying Time (secs)',  
 y = 'Driver',  
 title = 'Standardized Qualifying Time',  
 subtitle = '2023',  
 caption = '\*Times are subtracted from the average time for each Grand Prix') +  
 geom\_vline(xintercept = 0, col = 'red') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 facet\_wrap(~ practice\_session) +  
 xlim(-5, 5) +  
 scale\_color\_manual('', values = c('red', 'navy', 'red', 'navy'))



In the next chapter, we will move onto qualifying!

# 3 Qualifying

Qualifying is one of the most important components of a Formula 1 Grand Prix weekend. Qualifying position determines the position where that driver starts the Grand Prix. Qualifying is actually composed of three different sessions: Q1, Q2, and Q3. All drivers take part in Q1, but only the top 15 fastest times are allowed to continue on to Q2. The fastest 10 times in Q2 make it to Q3. Q1 lasts 18 minutes, while Q2 lasts 15 minutes, and Q3 lasts 12 minutes. If a driver violates a rule or regulation, the penalties are issued following qualifying.

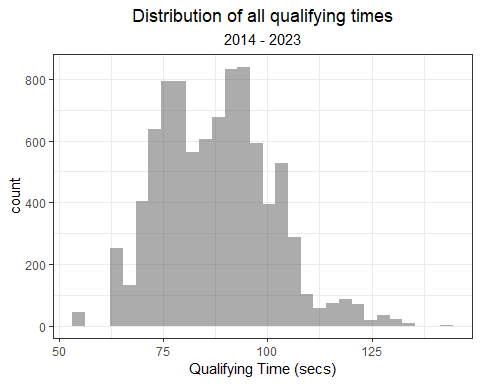
Qualifying data is readily available online. To scrape qualifying data drectly from www.formula1.com, you can use the qualifying\_scraper() function from the **drs** package.

library(tidyverse)  
library(drs)  
  
  
# Scrape qualifying data  
qualifying\_2023 <- qualifying\_scraper(2023)  
qualifying\_2022 <- qualifying\_scraper(2022)  
qualifying\_2021 <- qualifying\_scraper(2021)  
qualifying\_2020 <- qualifying\_scraper(2020)  
qualifying\_2019 <- qualifying\_scraper(2019)  
qualifying\_2018 <- qualifying\_scraper(2018)  
qualifying\_2017 <- qualifying\_scraper(2017)  
qualifying\_2016 <- qualifying\_scraper(2016)  
qualifying\_2015 <- qualifying\_scraper(2015)  
qualifying\_2014 <- qualifying\_scraper(2014)  
  
# Combine all qualifying data  
qualifying\_allyears <- rbind(qualifying\_2023,  
 qualifying\_2022,  
 qualifying\_2021,   
 qualifying\_2020,  
 qualifying\_2019,  
 qualifying\_2018,  
 qualifying\_2017,  
 qualifying\_2016,  
 qualifying\_2015,  
 qualifying\_2014)

## 3.1 Distribution of Qualifying Times by Grand Prix

One of the more basic ways to visualize qualifying times is to use a histogram. We can use histograms to plot the entire distribution of qualifying times from 2014 to 2023. In the figure below, we lump all qualifying times together (all sessions, drivers, teams, years, and circuits are included in the same distribution).

qualifying\_allyears %>%  
 dplyr::select(Q1\_secs, Q2\_secs, Q3\_secs) %>%  
 pivot\_longer(everything(), names\_to = 'qualifying\_session', values\_to = 'time') %>%  
 ggplot(aes(time)) +  
 geom\_histogram(alpha = 0.5) +  
 theme\_bw() +  
 labs(x = 'Qualifying Time (secs)',  
 title = 'Distribution of all qualifying times',  
 subtitle = '2014 - 2023')+  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 legend.position = "none")



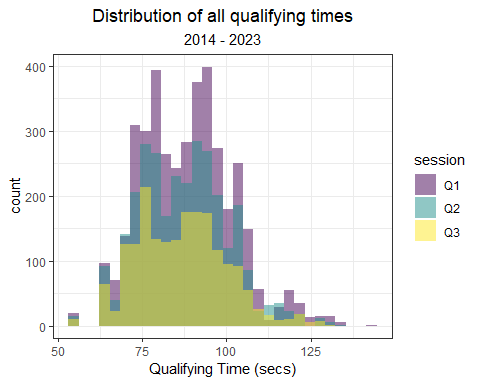
**How to create a histogram in R**

The histogram geometry, geom\_histogram(), can be used to plot the distribution of a single continuous variable. geom\_histogram() counts the number of observations in a given gib, and plots along the x-axis.

For more information on using geom\_histogram() in **ggplot2**, visit this link: <https://ggplot2.tidyverse.org/reference/geom_histogram.html>

The plot above is not all that informative (at least in my opinion). Most people are probably not that interested in the distribution of all qualifying times but rather the distribution by session or some other grouping variable. Here’s a look at the distribution of qualifying times by session. Obviously, Q3 times should be faster than Q2 which is faster than Q1.

qualifying\_allyears %>%  
 pivot\_longer(c('Q1\_secs', 'Q2\_secs', 'Q3\_secs'),  
 names\_to = 'session',  
 values\_to = 'time') %>%  
 mutate(Year = factor(Year),  
 session = str\_remove(session, '\_secs')) %>%  
 ggplot() +  
 geom\_histogram(aes(x = time, fill = session),  
 alpha = 0.5, show.legend = T,  
 position="identity") +  
 theme\_bw() +  
 labs(x = 'Qualifying Time (secs)',  
 title = 'Distribution of all qualifying times',  
 subtitle = '2014 - 2023')+  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5)) +  
 scale\_fill\_viridis\_d()



**How to remove matched patterns in R**

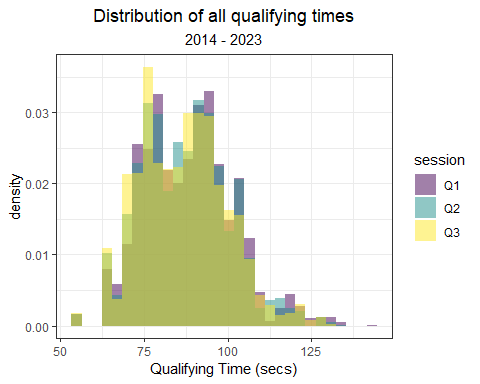
In the code chunk above, I use the str\_remove() function to remove text from the session variable name. At times, you will be left with sloppy variable names (like I did here). I wanted to remove the “\_secs” suffix from the session factor names. str\_remove() makes this incredibly easy to do.

For more information on using str\_remove(), visit this link: <https://stringr.tidyverse.org/reference/str_remove.html>

There are more participants in the slower sessions, so the height of the distributions are going to be different. We can standardize the heights by using the following argument:

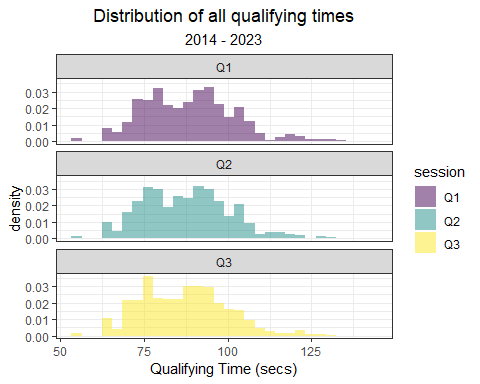
geom\_histogram(aes(x = time, y = ..density.., fill = session)

qualifying\_allyears %>%  
 pivot\_longer(c('Q1\_secs', 'Q2\_secs', 'Q3\_secs'),  
 names\_to = 'session',  
 values\_to = 'time') %>%  
 mutate(Year = factor(Year),  
 session = str\_remove(session, '\_secs')) %>%  
 ggplot() +  
 geom\_histogram(aes(x = time, y = ..density.., fill = session),  
 alpha = 0.5, show.legend = T,  
 position="identity") +  
 theme\_bw() +  
 labs(x = 'Qualifying Time (secs)',  
 title = 'Distribution of all qualifying times',  
 subtitle = '2014 - 2023')+  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5)) +  
 scale\_fill\_viridis\_d()



With this adjustment, the plot is still not very visually informative. Maybe, we can split the plot by session:

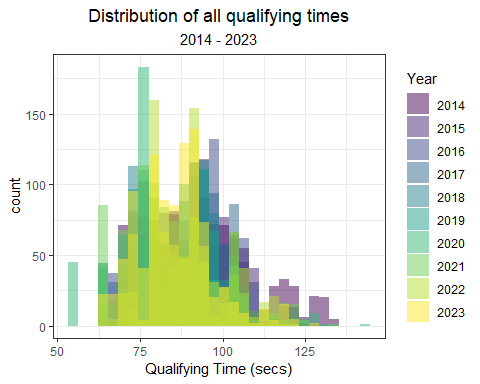
qualifying\_allyears %>%  
 pivot\_longer(c('Q1\_secs', 'Q2\_secs', 'Q3\_secs'),  
 names\_to = 'session',  
 values\_to = 'time') %>%  
 mutate(Year = factor(Year),  
 session = str\_remove(session, '\_secs')) %>%  
 ggplot() +  
 geom\_histogram(aes(x = time, y = ..density.., fill = session),  
 alpha = 0.5, show.legend = T,  
 position="identity") +  
 theme\_bw() +  
 labs(x = 'Qualifying Time (secs)',  
 title = 'Distribution of all qualifying times',  
 subtitle = '2014 - 2023')+  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5)) +  
 scale\_fill\_viridis\_d() +  
 facet\_wrap(~ session, ncol = 1)



A little better, but… not great. We should revisit the best way to visualize times by session later. But for now, we will shift our focus to yearly differences.

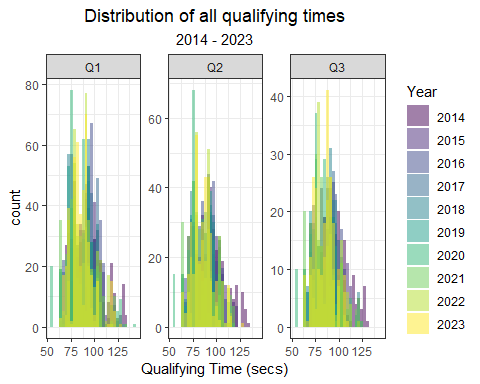
Visualizing the distribution of times by year is somewhat interesting, however. It is very easy to add a new dimension (year) to this histogram:

qualifying\_allyears %>%  
 pivot\_longer(c('Q1\_secs', 'Q2\_secs', 'Q3\_secs'),  
 names\_to = 'session',  
 values\_to = 'time') %>%  
 mutate(Year = factor(Year)) %>%  
 ggplot() +  
 geom\_histogram(aes(x = time, fill = Year),  
 alpha = 0.5, show.legend = T,  
 position="identity") +  
 theme\_bw() +  
 labs(x = 'Qualifying Time (secs)',  
 title = 'Distribution of all qualifying times',  
 subtitle = '2014 - 2023')+  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5)) +  
 scale\_fill\_viridis\_d()



This includes all qualifying sessions together, so you could also split the data by session while retaining different plots for each year:

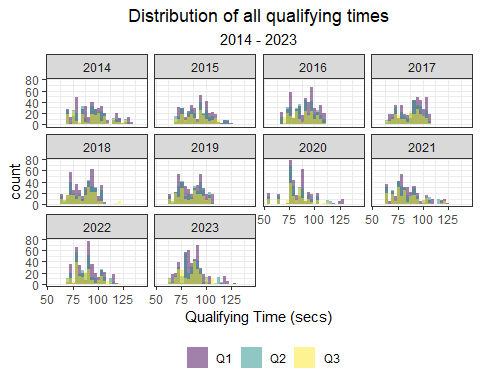
qualifying\_allyears %>%  
 pivot\_longer(c('Q1\_secs', 'Q2\_secs', 'Q3\_secs'),  
 names\_to = 'session',  
 values\_to = 'time') %>%  
 mutate(Year = factor(Year),  
 session = str\_remove(session, '\_secs')) %>%  
 ggplot() +  
 geom\_histogram(aes(x = time, fill = Year),  
 alpha = 0.5, show.legend = T,  
 position="identity") +  
 theme\_bw() +  
 labs(x = 'Qualifying Time (secs)',  
 title = 'Distribution of all qualifying times',  
 subtitle = '2014 - 2023')+  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5)) +  
 scale\_fill\_viridis\_d() +  
 facet\_wrap(~ session, scales = 'free\_y')



This plot is very tough to read! Like the previous example above, there’s a couple of ways to try and remedy this:

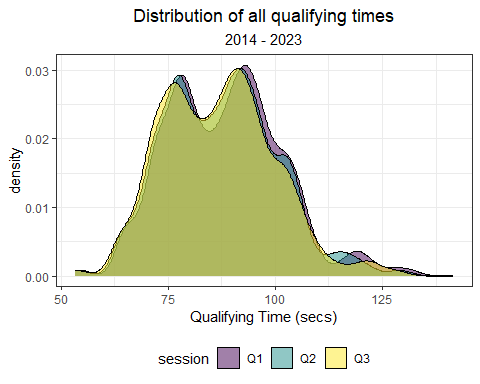
Option 1: Use facet\_wrap() to split the distributions into facets:

qualifying\_allyears %>%  
 pivot\_longer(c('Q1\_secs', 'Q2\_secs', 'Q3\_secs'),  
 names\_to = 'session',  
 values\_to = 'time') %>%  
 mutate(Year = factor(Year),  
 session = str\_remove(session, '\_secs')) %>%  
 ggplot() +  
 geom\_histogram(aes(x = time, fill = session),  
 alpha = 0.5, show.legend = T,  
 position="identity") +  
 theme\_bw() +  
 labs(x = 'Qualifying Time (secs)',  
 title = 'Distribution of all qualifying times',  
 subtitle = '2014 - 2023',  
 fill = '')+  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 legend.position = 'bottom') +  
 scale\_fill\_viridis\_d() +  
 facet\_wrap(~ Year)



Option 2: Use geom\_density(), in lieu of geom\_histogram(). **ggplot**’s density geometry is a smooth alternative to the histogram geometry.

qualifying\_allyears %>%  
 pivot\_longer(c('Q1\_secs', 'Q2\_secs', 'Q3\_secs'),  
 names\_to = 'session',  
 values\_to = 'time') %>%  
 mutate(Year = factor(Year),  
 session = str\_remove(session, '\_secs')) %>%  
 ggplot() +  
 geom\_density(aes(x = time, fill = session),  
 alpha = 0.5, show.legend = T,  
 position="identity") +  
 theme\_bw() +  
 labs(x = 'Qualifying Time (secs)',  
 title = 'Distribution of all qualifying times',  
 subtitle = '2014 - 2023')+  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 legend.position = 'bottom') +  
 scale\_fill\_viridis\_d()



**How to create a kernel density plot in R**

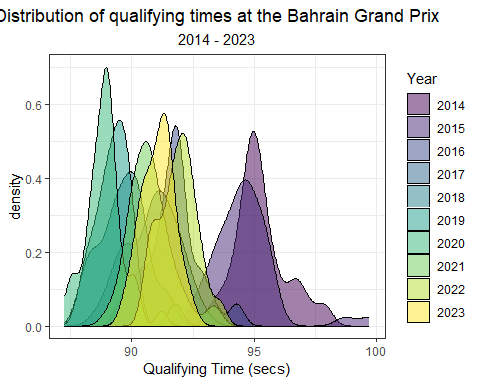
An alternative to using the histogram geometry, geom\_histogram(), is the smoothed kernel density geometry, geom\_density(). geom\_density() calculates and plots a smoothed version of a histogram, commonly referred to as the kernel density estimate.

For more information on using geom\_density() in **ggplot2**, visit this link: <https://ggplot2.tidyverse.org/reference/geom_density.html>

These distributions include all qualifying sessions from 2014 to 2023, and the multi-modal nature of the distributions reflect this.

A more useful way to use histograms is to plot the distribution for one Grand Prix at a time. For instance, here’s the distribution of qualifying times at the Bahrain Grand Prix only:

qualifying\_allyears %>%  
 pivot\_longer(c('Q1\_secs', 'Q2\_secs', 'Q3\_secs'),  
 names\_to = 'session',  
 values\_to = 'time') %>%  
 mutate(Year = factor(Year),  
 session = str\_remove(session, '\_secs')) %>%  
 filter(Race == 'bahrain') %>%  
 ggplot() +  
 geom\_density(aes(x = time, fill = Year),  
 alpha = 0.5, show.legend = T,  
 position="identity") +  
 theme\_bw() +  
 labs(x = 'Qualifying Time (secs)',  
 title = 'Distribution of qualifying times at the Bahrain Grand Prix',  
 subtitle = '2014 - 2023')+  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5)) +  
 scale\_fill\_viridis\_d()

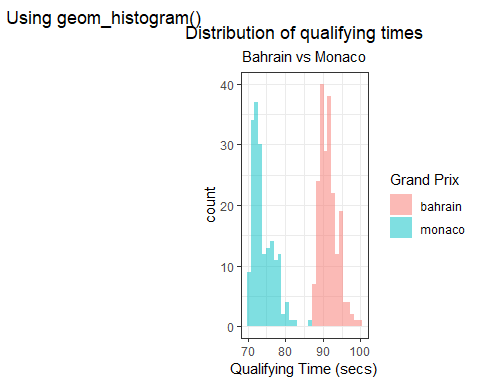


One challenge with using the qualifying data is that the fastest drivers are counted three times and the slowest only once. All drivers will typically have a Q1 time, but the fastest cars are probably driving more conservatively. So, how do we make comparisons? I prefer taking each driver’s fastest time in their last session. So, if a driver made it to Q2 but not Q3, we should use his Q2 time. To use this data, we will need to further tidy up the data that we currently have using the following code:

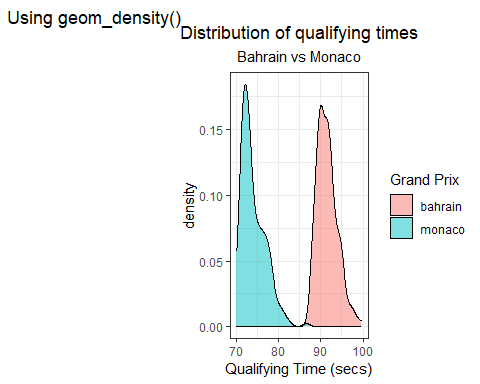
qualifying\_allyears <- qualifying\_allyears %>%  
 mutate(Q\_secs = case\_when(  
 !is.na(Q3\_secs) ~ Q3\_secs,  
 is.na(Q3\_secs) & !is.na(Q2\_secs) ~ Q2\_secs,  
 is.na(Q3\_secs) & is.na(Q2\_secs) & !is.na(Q1\_secs) ~ Q1\_secs))

Or, you can use geom\_histogram() / geom\_density() to compare two Grands Prix:

# histogram  
qualifying\_allyears %>%  
 mutate(Year = factor(Year)) %>%  
 filter(Race %in% c('bahrain', 'monaco')) %>%  
 ggplot() +  
 geom\_histogram(aes(x = Q\_secs, fill = Race),  
 alpha = 0.5, show.legend = T,  
 position="identity") +  
 theme\_bw() +  
 labs(fill = 'Grand Prix',  
 x = 'Qualifying Time (secs)',  
 title = 'Distribution of qualifying times',  
 subtitle = 'Bahrain vs Monaco',  
 tag = 'Using geom\_histogram()')+  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5))

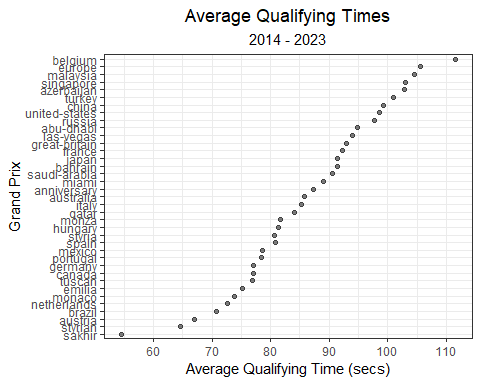


#density  
qualifying\_allyears %>%  
 mutate(Year = factor(Year)) %>%  
 filter(Race %in% c('bahrain', 'monaco')) %>%  
 ggplot() +  
 geom\_density(aes(x = Q\_secs, fill = Race),  
 alpha = 0.5, show.legend = T,  
 position="identity") +  
 theme\_bw() +  
 labs(fill = 'Grand Prix',  
 x = 'Qualifying Time (secs)',  
 title = 'Distribution of qualifying times',  
 subtitle = 'Bahrain vs Monaco',  
 tag = 'Using geom\_density()')+  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5))



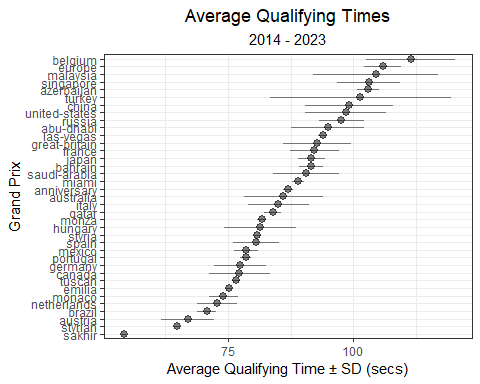
Clearly times in Bahrain are different than those at Monaco. In fact, all circuits are going to vary to some degree. The following figure shows the average qualifying times by Grand Prix.

qualifying\_allyears %>%  
 group\_by(Race) %>%  
 summarize(mean = mean(Q\_secs, na.rm = T),  
 sd = sd(Q\_secs, na.rm = T)) %>%   
 ggplot(aes(x = mean, y = fct\_reorder(Race, mean))) +  
 geom\_point(position = position\_jitter(h = 0, w = 0.3), alpha = 0.5) +  
 theme\_bw() +  
 labs(x = 'Average Qualifying Time (secs)',  
 y = 'Grand Prix',  
 title = 'Average Qualifying Times',  
 subtitle = '2014 - 2023')+  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 legend.position = "none")



A lot can happen during a qualifying session causing variability among the cars. There is a spectrum of performance across cars, drivers, sessions, and year. Therefore, it’s a good idea to include a measure of variance in this plot. So, we can add standard error bars to the plot we just made.

qualifying\_allyears %>%  
 group\_by(Race) %>%  
 summarize(mean = mean(Q\_secs, na.rm = T),  
 sd = sd(Q\_secs, na.rm = T)) %>%   
 mutate(lower = mean - sd,  
 upper = mean + sd) %>%   
 ggplot(aes(x = mean, xmin = lower, xmax = upper, y = fct\_reorder(Race, mean))) +  
 geom\_pointrange(alpha = 0.5) +  
 theme\_bw() +  
 labs(x = 'Average Qualifying Time \u00b1 SD (secs)',  
 y = 'Grand Prix',  
 title = 'Average Qualifying Times',  
 subtitle = '2014 - 2023')+  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 legend.position = "none")



**Plotting a point-range in ggplot**

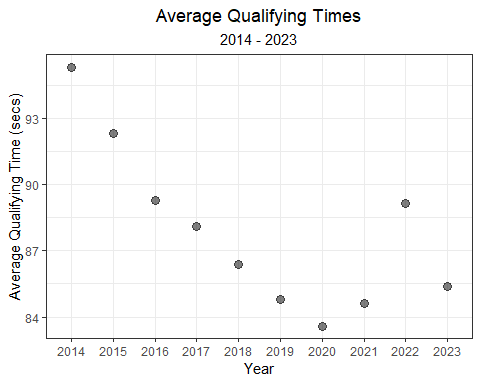
Similar to earlier plots using stat\_summary(), we can manually construct a point-range in **ggplot** using the geom\_pointrange().

In the example above, the continuous variable is plotted along the x-axis. I wanted to plot the mean ± 1 standard deviation, so I pass arguments for x, xmin and xmax that define the mean, minimum, and maximum values in this geometry.

For more information on using geom\_pointrange() in **ggplot2**, visit this link: <https://ggplot2.tidyverse.org/reference/geom_linerange.html>

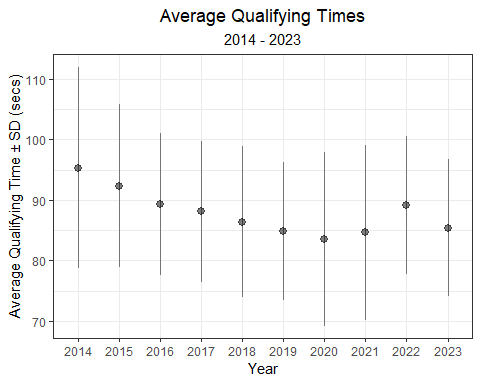
Previously, we demonstrated that practice times vary by year. Qualifying times must follow a similar pattern. Using the code below, we can plot the average qualifying times by year.

qualifying\_allyears %>%  
 mutate(Year = factor(Year)) %>%  
 group\_by(Year) %>%  
 summarize(mean = mean(Q\_secs, na.rm = T),  
 sd = sd(Q\_secs, na.rm = T)) %>%   
 mutate(lower = mean - sd,  
 upper = mean + sd) %>%  
 ggplot(aes(y = mean, x = Year)) +  
 geom\_point(alpha = 0.5, size = 3) +  
 theme\_bw() +  
 labs(x = 'Year',  
 y = 'Average Qualifying Time (secs)',  
 title = 'Average Qualifying Times',  
 subtitle = '2014 - 2023')+  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 legend.position = "none")



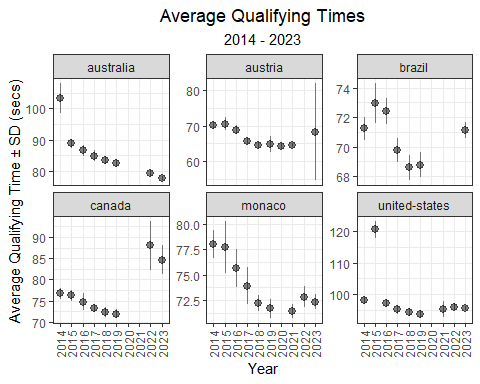
This plot computes an average for all cars and circuits. This collection of qualifying times will undoubtedly include lots of variability! Using geom\_pointrange(), we can try to capture that vairability in the plot below:

qualifying\_allyears %>%  
 mutate(Year = factor(Year)) %>%  
 group\_by(Year) %>%  
 summarize(mean = mean(Q\_secs, na.rm = T),  
 sd = sd(Q\_secs, na.rm = T)) %>%   
 mutate(lower = mean - sd,  
 upper = mean + sd) %>%  
 ggplot(aes(y = mean, ymin = lower, ymax = upper, x = Year)) +  
 geom\_pointrange(alpha = 0.5) +  
 theme\_bw() +  
 labs(x = 'Year',  
 y = 'Average Qualifying Time \u00b1 SD (secs)',  
 title = 'Average Qualifying Times',  
 subtitle = '2014 - 2023')+  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 legend.position = "none")



To chip away at this variability, we can replicate this plot for just a handful of circuits (i.e. Australia, Monaco, Brazil, Austria, Canada, and the United States). Below, I use facet\_wrap() to split the figure and assign each circuit its own facet. Notice how tight the standard deviations have become.. much better!

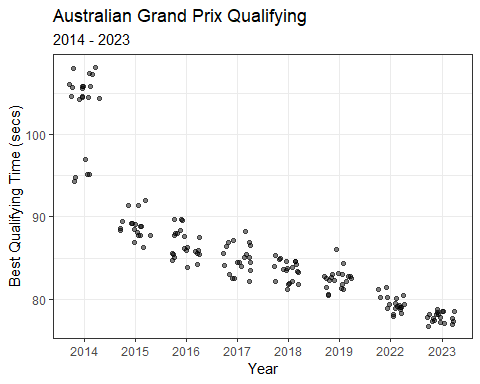
qualifying\_allyears %>%  
 filter(Race %in% c('australia', 'monaco', 'brazil', 'austria', 'canada', 'united-states')) %>%   
 mutate(Year = factor(Year)) %>%  
 group\_by(Year, Race) %>%  
 summarize(mean = mean(Q\_secs, na.rm = T),  
 sd = sd(Q\_secs, na.rm = T)) %>%   
 mutate(lower = mean - sd,  
 upper = mean + sd) %>%   
 ggplot(aes(y = mean, ymin = lower, ymax = upper, x = Year)) +  
 geom\_pointrange(alpha = 0.5) +  
 theme\_bw() +  
 labs(x = 'Year',  
 y = 'Average Qualifying Time \u00b1 SD (secs)',  
 title = 'Average Qualifying Times',  
 subtitle = '2014 - 2023')+  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 legend.position = "none",  
 axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1)) +  
 facet\_wrap(~ Race, scales = 'free\_y')



## 3.2 How does car development progress over time?

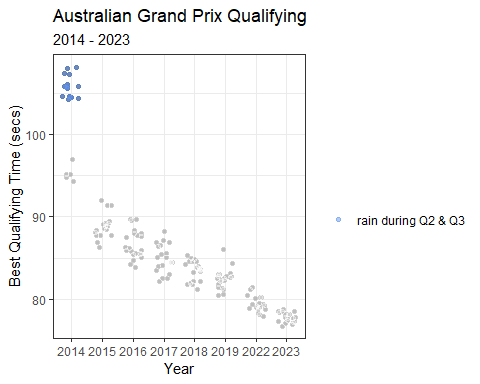
Below, we can look at the qualifying pace for the first race of most years, the Australian Grand Prix.

qualifying\_allyears %>%  
 mutate(Year = factor(Year)) %>%  
 filter(Race == 'australia') %>%  
 ggplot(aes(Year, Q\_secs)) +  
 geom\_point(position = position\_jitter(h = 0, w = 0.3), alpha = 0.5) +  
 theme\_bw() +  
 labs(x = 'Year',  
 y = 'Best Qualifying Time (secs)',  
 title = 'Australian Grand Prix Qualifying',  
 subtitle = '2014 - 2023')



During 2014 qualifying, rain arrived during Q2.

qualifying\_allyears %>%  
 mutate(Year = factor(Year)) %>%  
 filter(Race == 'australia') %>%  
 mutate(rain = ifelse(Year == '2014' & Q\_secs > 100, 'rain during Q2 & Q3', ' ')) %>%   
 ggplot() +  
 geom\_point(aes(Year, Q\_secs),  
 position = position\_jitter(seed= 123, h = 0, w = 0.3), alpha = 0.5) +  
 geom\_point(aes(Year, Q\_secs, col = rain),   
 position = position\_jitter(seed = 123, h = 0, w = 0.3), alpha = 0.5) +  
 theme\_bw() +  
 labs(x = 'Year',  
 y = 'Best Qualifying Time (secs)',  
 title = 'Australian Grand Prix Qualifying',  
 subtitle = '2014 - 2023') +  
 scale\_colour\_manual("", values = c('rain during Q2 & Q3' = 'cornflowerblue', ' ' = 'transparent'))



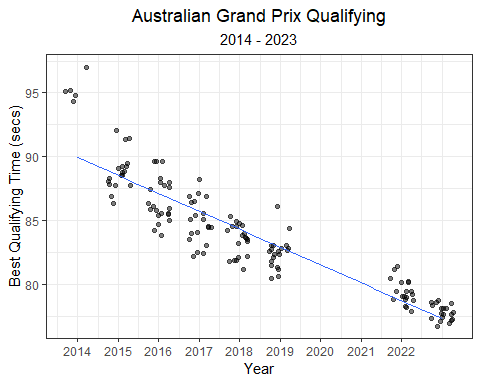
**Reproducible *jittering* in ggplot**

In the plot above, I had to get a bit creative. My goal was to highlight the times in 2014 by using a blue color. And, I only wanted a single label for those times. My solution was to plot all points as a first layer, and then re-plot the data using a conditional coloring if the data was labeled with rain. The first layer was randomly jittered, so I needed to preserve that randomness for the second layer. To accomplish this, I passed an argument for a random seed (i.e. seed = 123) to ensure that the *jitter* was reproducible.

For more information on *jittering* in **ggplot2**, visit this link: <https://ggplot2.tidyverse.org/reference/position_jitter.html>

These times set during a rainy qualifying session would complicate our understanding of a car’s progression over time. To avoid their influence, we can filter the rain-influenced times from Q2 and Q3 in 2014, and re-plot below. I’ll add a best-fit linear regression line that describes the improvement of qualifying times from 2014 to 2022, on average.

qualifying\_allyears %>%  
 filter(Race == 'australia') %>%  
 mutate(rain = ifelse(Year == 2014 & Q\_secs > 100, 'rain during Q2', ' ')) %>%   
 filter(rain == ' ') %>%   
 ggplot(aes(Year, Q\_secs)) +  
 geom\_point(position = position\_jitter(seed= 123, h = 0, w = 0.3), alpha = 0.5) +  
 stat\_smooth(method = 'lm', se = F, size = 0.3, alpha = 0.5) +   
 theme\_bw() +  
 labs(x = 'Year',  
 y = 'Best Qualifying Time (secs)',  
 title = 'Australian Grand Prix Qualifying',  
 subtitle = '2014 - 2023') +  
 scale\_x\_continuous(breaks = c(2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022)) +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5))



**Adding a best-fit line in ggplot**

The stat\_smooth() function in **ggplot2** can help aid the eye in observing patterns in the data. If no method is declared in the function call, a **loess** (locally estimated scatter plot smoothing) function is fit to the data.

In the example above, I add method = "lm" to the function call, which adds a linear model to the data. By default, stat\_smooth() also adds a 95% confidence interval around the fit. In the example above, I disabled the confidence interval by adding se = FALSE to the call.

For more information on best-fit lines, or smoothed conditional means, in **ggplot2**, visit this link: <https://ggplot2.tidyverse.org/reference/geom_smooth.html>

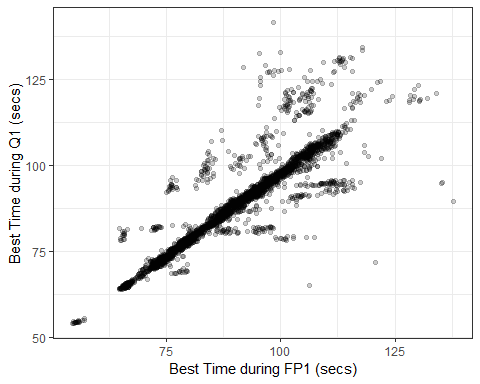
While the stat\_smooth() function allows us to easily add a linear regression line to the scatter plot, it doesn’t quantify the coefficient estimates. In the next chapter, we will explore using linear regression models to estimate the relationship between practice times, qualifying times, and more!

# 4 Modeling Practice and Qualifying Times

As mentioned in previous chapters, some runs during practice sessions can often be used to simulate qualifying or race pace. For instance, FP1 is typically used to test the car and ensure it is working as expected, while FP2 and FP3 are often used to test the car’s performance on *long runs* and the car’s speed over laps, respectively. Unsurprisingly, one would expect practice times to correlate to qualifying times. This chapter will take a deep-dive into that relationship and try to better understand it.

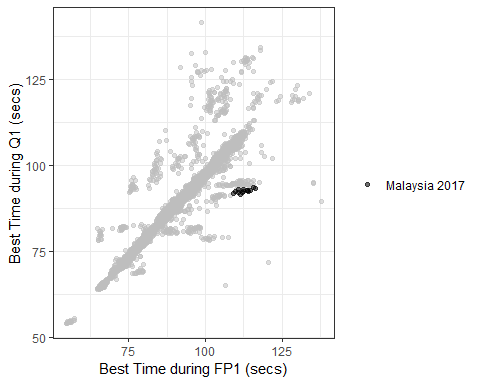
Below, we can plot the best time set during FP1 versus the best time during Q1. For the most part, there is a near 1:1 ratio. However, there are several outlying clusters of times that don’t follow a 1:1 ratio.

practice\_and\_qualifying %>%  
 ggplot(aes(Time\_secs\_1, Q1\_secs)) +  
 geom\_point(alpha = 0.2) +  
 theme\_bw() +  
 labs(x = 'Best Time during FP1 (secs)',  
 y = 'Best Time during Q1 (secs)')



One of these outlying clusters is data collected from the Malaysian Grand Prix in 2017. Free Practice 1 of the Malaysian Grand Prix in 2017 was delayed for half an hour due to heavy rain at the Sepang International Circuit. Max Verstappen eventually finished the wet FP1 session with a fastest time of 1:48.962. Qualifying was not impacted by rain, and Lewis Hamilton took pole with a time of 1:30.076. In the figure below, we can clearly see this relationship displayed by the *flatter* slope for the Malaysian Grand Prix in 2017.

practice\_and\_qualifying %>%  
 mutate(malaysia\_2017 = ifelse(Race == 'malaysia' & Year == 2017, 'Malaysia 2017', 'No')) %>%   
 ggplot(aes(Time\_secs\_1, Q1\_secs)) +  
 geom\_point(aes(col = malaysia\_2017), alpha = 0.5, show.legend = T) +  
 theme\_bw() +  
 labs(x = 'Best Time during FP1 (secs)',  
 y = 'Best Time during Q1 (secs)') +  
 gghighlight(malaysia\_2017 == 'Malaysia 2017', use\_direct\_label = FALSE) +  
 scale\_color\_manual('', values = c('Malaysia 2017' = 'black'))



**How to create a custom color scale in ggplot**

At times, creating custom color scales can be very convenient. The simplest way to create a custom color scale is by using scale\_color\_manual(). In the figure above, I used the following function to color data for *Malaysia 2017* black:

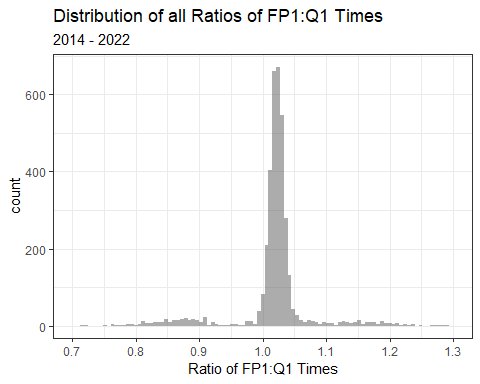
scale\_color\_manual('', values = c('Malaysia 2017' = 'black'))

For more information on using scale\_color\_manual() in **ggplot2**, visit this link: <https://ggplot2.tidyverse.org/reference/scale_manual.html>

I want to estimate the relationship between FP1 times and Q1 times, but these outlying clusters are compromising the relationship. In the Malaysia example above, rain was the root cause for a distinctly different slope (FP1 vs Q1). I want to remove these abnormal sessions, but it would be quite tedious to research all FP1 or Q1 sessions that were impacted by rain. However, I could try and utilize exploratory data analysis (EDA) and data wrangling to filter out abnormal times.

For the vast majority of Grands Prix, the slope of this relationship (FP1 times vs Q1 times) is slightly larger than 1. I’ll use a histogram below to plot the distribution of ratios of all FP1:Q1 times.

practice\_and\_qualifying %>%  
 mutate(ratio = Time\_secs\_1 / Q1\_secs) %>%   
 ggplot(aes(ratio)) +   
 geom\_histogram(bins = 100, alpha = 0.5) +   
 theme\_bw() +  
 labs(x = 'Ratio of FP1:Q1 Times',  
 title = 'Distribution of all Ratios of FP1:Q1 Times',  
 subtitle = '2014 - 2022') +  
 scale\_x\_continuous(limits = c(0.7, 1.3),   
 breaks = c(0.7, 0.8, 0.9, 1, 1.1, 1.2, 1.3))

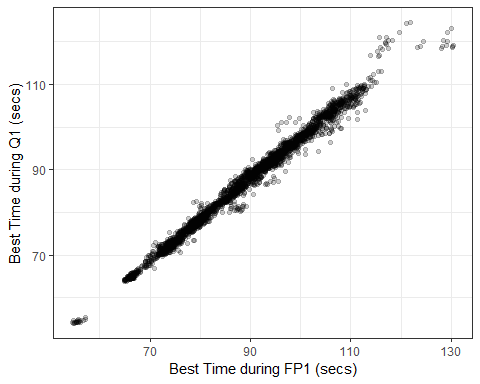


There are three separate peaks in this distribution:

* slightly larger than 1.0
* just below 0.90
* at approximately 1.15

Knowing this information, I will try to filter out any ratios lower than 0.95 or larger than 1.10.

practice\_and\_qualifying %>%  
 mutate(ratio = Time\_secs\_1 / Q1\_secs) %>%   
 filter(ratio >= 0.95 & ratio <= 1.10) %>%   
 ggplot(aes(Time\_secs\_1, Q1\_secs)) +  
 geom\_point(alpha = 0.2) +  
 theme\_bw() +  
 labs(x = 'Best Time during FP1 (secs)',  
 y = 'Best Time during Q1 (secs)')



Much better! There’s still a few funky outliers, but I feel comfortable that this distribution represents the typical Formula 1 weekend. While this graph is certainly some evidence to support that FP1 times are *correlated* to Q1 times, I actually want to quantify this relationship. Next, I’ll use a **simple linear regression model** to estimate the relationship between FP1 and Q1 times.

## 4.1 Models

Why build a model? At the most basic level, we are looking for *relationships* in our data. In fact, that is also one of the core reasons that people even record data. Most minds are searching for a *causal* relationship between two (or more) variables (i.e. measured quantities). Models are essentially mathematical mappings between variables… in a very structured way. From a technical standpoint, people build and use models for a number of different reasons. These purposes generally fall into one of these three groups:

* models used for inference (i.e. using a model to answer questions)
* predictive models
* models used for both inference and prediction

An example of an inferential model could be a logistic regression model used to estimate the effect that smoking has on someone’s likelihood of developing cancer. An example of predictive model is an election forecaster. But, inferential models can still be used to make predictions, and sometimes the predictive models can make inferences. In this chapter, I want to use an inferential models to make general claims about my collected data. In the next section, I’ll introduce regression models and show some examples of how you can use them to learn about Formula 1.

## 4.2 Simple Linear Regression Model: Practice vs Qualifying Times

Dr. Andrew Gelman summarizes regression much more succinctly than I ever could:

*Regression is a method that allows researchers to summarize how predictions or average values of an outcome vary across individuals defined by a set of predictors.*

If you’ve ever created a scatterplot in Excel, the *best-fit* line is based on a linear regression between the variable on the x-axis and the variable on the y-axis. However, linear regression can be far more useful than a simple method to find the best-fit line. Throughout the rest of this chapter, I will introduce ways to summarize and communicate results from regression models. In this book, I will only briefly cover particularly important aspects of models. So, if you are in need of an introduction or refresher on regression, I *highly* recommend checking out the following introductory book on regression:

**More information about linear regression in R**

My favorite introductory book on regression in R is *Regression and Other Stories* by Andrew Gelman, Jennifer Hill, and Aki Vehtari. This book focuses on the practical application of regression models in R, and includes tons of very helpful code. While the authors build bayesian models, I think it is helpful for anyone looking to learn about regression. Here’s a link to the free book and lots of example code:

<https://avehtari.github.io/ROS-Examples/index.html>

The most simple **linear regression** model uses a single predictor variable. In our model, we will use a single predictor (FP1 times). We will use linear regression to fit a linear relationship (straight line) between the predictor variable (FP1 times) and the response variable (Q1 times). Using the predictor variable (FP1) and response variable (Q1), we will estimate the *intercept*, *slope*, and *error* using a model.

The formula for this simple linear regression model is:

We will use the filtered dataset (i.e. ratios between 0.95 and 1.10). Below, I’ll re-write this data to a new dataframe.

fp1\_v\_q1\_clean <- practice\_and\_qualifying %>%  
 mutate(ratio = Time\_secs\_1 / Q1\_secs) %>%   
 filter(ratio >= 0.95 & ratio <= 1.10)

To fit this model, we will use the lm() function.

# Fir the linear regression model  
fp1\_v\_q1.model <- lm(Q1\_secs ~ Time\_secs\_1, data = fp1\_v\_q1\_clean)

**How to fit a simple linear regression model in R**

To fit a simple linear regression model in R, we can use the lm() function.

The lm() function takes uses the following notation:

lm(response variable ~ predictor variable(s), data source)

For more information on using lm(), visit this link: <https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/lm>

To summarize the output from the linear model, I will use the summary() function.

summary(fp1\_v\_q1.model)

##   
## Call:  
## lm(formula = Q1\_secs ~ Time\_secs\_1, data = fp1\_v\_q1\_clean)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -7.9836 -0.5859 0.0210 0.6408 8.3919   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.610673 0.163619 9.844 <2e-16 \*\*\*  
## Time\_secs\_1 0.958841 0.001828 524.432 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1.295 on 3200 degrees of freedom  
## Multiple R-squared: 0.9885, Adjusted R-squared: 0.9885   
## F-statistic: 2.75e+05 on 1 and 3200 DF, p-value: < 2.2e-16

The summary output from this model contains a lot of information about this model fit. This list gives a bit more information about the most noteworthy items in this output:

* The model formula can be found under *Call*.
* The quantiles of the residuals are listed below the model formula.
* the coefficient values are the model estimates for intercept and slope. Each coefficient also has a standard error, t-values, and statistical significance.
* The residual standard error
* degrees of freedom
* R-squared
* F-statistic
* p-value

In this particular model, the *Intercept* is somewhat meaningless. The intercept describes the value of the response (Q1 times) when the predictor variable (FP1 time) is = 0. However, in this example, there is no situation where FP1 is = 0.

We can interpret the slope as:

*A driver’s Q1 time is expected to be about 0.036 seconds faster than their best time during Free Practice 1, on average.*

I arrived at 0.036 seconds by subtracting the slope from 1.

The **R2** is another important metric from this output. The **R2** value, also known as the coefficient of determination, is the proportion of the variance in Q1 times that are explained by FP1 time. In this case, the R2 = 0.989, which can be interpreted as ~ 98.9% of the variance in a driver’s Q1 time can be explained by their FP1 time.

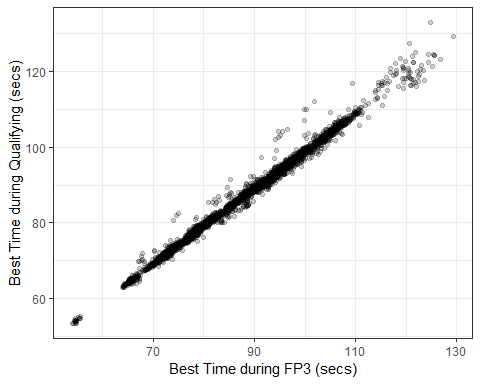
The F-statistic is of little use to us in this case. The *F-statistic* is the test statistic for the *F-test*, which evaluates whether this particular model provides a better fit to our data than a model that uses no predictor variables. If the *p-value* is less than 0.05 (which it is), we conclude that this model is better than a model using no predictor variables. But, we already knew this linear relationship was legit!

For our purposes, we should be most concerned with the R2 and the slope of this model. The R2 explains the amount of variance our model explains, while the slope describes how FP1 times and Q1 times related. We can apply this same approach to a different comparison: FP3 vs fastest qualifying time. Because FP3 is used for qualifying simulations, it may be more representative of a car’s true qualifying pace. Using the fastest time in qualifying is likely a better measure because front-running teams likely take less risks during Q1.

fp3\_v\_q\_clean <- practice\_and\_qualifying %>%  
 mutate(ratio = Time\_secs\_3 / Q\_secs) %>%   
 filter(ratio >= 0.95 & ratio <= 1.10)  
  
# Fir the linear regression model  
fp3\_v\_q.model <- lm(Q\_secs ~ Time\_secs\_3, data = fp3\_v\_q\_clean)  
  
summary(fp3\_v\_q.model)

##   
## Call:  
## lm(formula = Q\_secs ~ Time\_secs\_3, data = fp3\_v\_q\_clean)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -8.1310 -0.3914 0.0386 0.5222 6.5447   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.210783 0.139942 8.652 <2e-16 \*\*\*  
## Time\_secs\_3 0.973464 0.001576 617.569 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1.194 on 3584 degrees of freedom  
## Multiple R-squared: 0.9907, Adjusted R-squared: 0.9907   
## F-statistic: 3.814e+05 on 1 and 3584 DF, p-value: < 2.2e-16

practice\_and\_qualifying %>%  
 mutate(ratio = Q\_secs / Time\_secs\_3) %>%   
 filter(ratio >= 0.95 & ratio <= 1.10) %>%   
 ggplot(aes(Time\_secs\_3, Q\_secs)) +  
 geom\_point(alpha = 0.2) +  
 theme\_bw() +  
 labs(x = 'Best Time during FP3 (secs)',  
 y = 'Best Time during Qualifying (secs)')



In this FP3 model, we can interpret the slope as:

*A driver’s best time in qualifying is expected to be about 0.012 seconds faster than their best time during Free Practice 3, on average.*

Again, I arrived at 0.012 seconds by subtracting the slope from 1.

This FP3 model explains fractionally more variance than the previous FP1 model (the R2 = 0.992 vs 0.989). So, FP3 times explain about 99.2% of the variance in a driver’s best qualifying time.

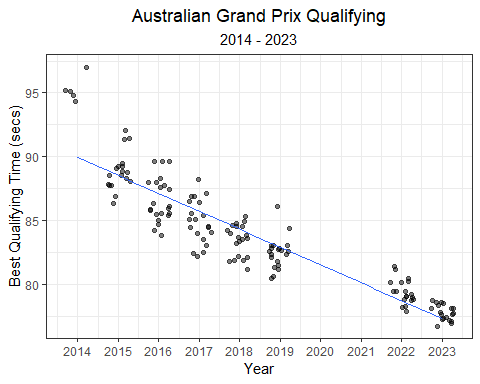
Pretty interesting!

Next, we’ll use a linear regression model to estimate the progression of qualifying times over time (i.e. by year).

## 4.3 Simple Linear Regression Model: Qualifying time vs Year

In the last chapter, we created this figure to show the progression of qualifying times at the Australian Grand Prix.

practice\_and\_qualifying %>%  
 filter(Race == 'australia') %>%  
 mutate(rain = ifelse(Year == 2014 & Q\_secs > 100, 'rain during Q2', ' ')) %>%   
 filter(rain == ' ') %>%   
 ggplot(aes(Year, Q\_secs)) +  
 geom\_point(position = position\_jitter(seed= 123, h = 0, w = 0.3), alpha = 0.5) +  
 stat\_smooth(method = 'lm', se = F, size = 0.3, alpha = 0.5) +   
 theme\_bw() +  
 labs(x = 'Year',  
 y = 'Best Qualifying Time (secs)',  
 title = 'Australian Grand Prix Qualifying',  
 subtitle = '2014 - 2023') +  
 scale\_x\_continuous(breaks = c(2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023)) +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5))



We used the stat\_smooth() function to fit a best-fit line through the data. But, this was just a visual aid and we didn’t actually quantify the relationship between year and qualifying time. To quantify this relationship, a simple linear regression model can be used. The formula for this model is simply:

In the code below, I (1) filter the data to include only the Australian Grand Prix, (2) remove sessions that were impacted by rain.

I then fit a simple linear regression model using the lm() function.

# filter times to include only clean Australian data  
aus\_quali\_times <- practice\_and\_qualifying %>%  
 filter(Race == 'australia') %>%  
 mutate(rain = ifelse(Year == 2014 & Q\_secs > 100, 'rain during Q2', ' ')) %>%   
 filter(rain == ' ',  
 !is.na(Q\_secs))  
  
# fit linear model  
aus.lm <- lm(Q\_secs ~ Year, aus\_quali\_times)

Below is a summary of the model output.

summary(aus.lm)

##   
## Call:  
## lm(formula = Q\_secs ~ Year, data = aus\_quali\_times)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3.5408 -1.0523 -0.0226 0.7590 7.0584   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 2913.43233 106.09863 27.46 <2e-16 \*\*\*  
## Year -1.40194 0.05257 -26.67 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1.767 on 141 degrees of freedom  
## Multiple R-squared: 0.8346, Adjusted R-squared: 0.8334   
## F-statistic: 711.3 on 1 and 141 DF, p-value: < 2.2e-16

Let’s try and interpret some of this output.

Our model fitted line equation is simply:

Of most importance are the **coefficient estimates, residual standard error, and R2**.

We can interpret the slope estimate like this:

**intercept**: For this particular model, the intercept reported in this ouput isn’t immediately useful. It tells us the estimated mean qualifying time for the year zero. A year zero does not exist in the *Anno Domini* calendar year system (the year 1 BC is followed directly by year AD 1), so not helpful! However, we can rescale it to the year 2014 and interpret the value like this:

*The average qualifying time at the Australian Grand Prix during the initial year of the hybrid era (2014) was 93.4 seconds (2913 + (-1.40 x 2014).*

**Slope**: Since the beginning of the hybrid era (2014), qualifying times in Australia decrease by 1.40 seconds per year, on average.

The **R2** value (coefficient of determination) is the proportion of the variance in Australian qualifying times that can be explained by yearly progression. In this case, the R2 = 0.83, which can be interpreted as 83% of the variance in qualifying times can be explained by the year.

An approximate interpretation of the **residual standard error/deviation (RSE)** is: A year’s qualifying times will deviate from the linear regression model fit line by 1.77 seconds, on average.

### 4.3.1 Uncertainty in model estimates

Above we described the slope point estimate as: *Since the beginning of the hybrid era (2014), qualifying times in Australia decrease by 1.45 seconds per year, on average.* But, how are sure are we of this number? A confidence interval allows us to also describe the uncertainty in that estimate. Below, I’ll calculate the 95% confidence interval around the slope estimate, and interpret it.

confint(aus.lm)

## 2.5 % 97.5 %  
## (Intercept) 2703.682609 3123.182046  
## Year -1.505856 -1.298015

**95% Confidence Interval around the Slope**: Since the beginning of the hybrid era (2014), qualifying times in Australia decrease by 1.45 seconds per year, on average. We have 95% confidence that the true average decrease in Australian qualifying times per year is between 1.30 and 1.51 seconds, on average.

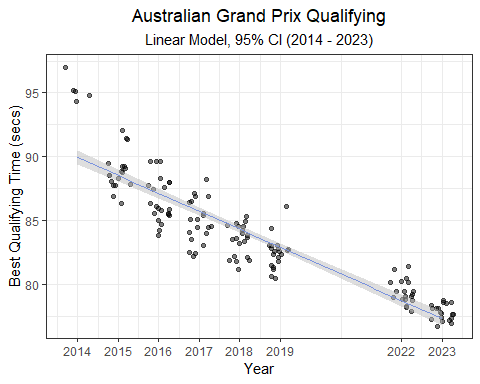
Let’s revisit the earlier plot of Australian Grand Prix qualifying times, with a best-fit line included. This line describes the linear regression model’s estimate of average qualifying time for each year of the race. Like we did with our slope estimate, we should describe our uncertainty around this line using a confidence and prediction interval.

**The confidence interval is uncertainty surrounding a *mean response*, and the prediction interval is uncertainty surrounding a prediction of a *future observation*.**

The interpretation of the **95% confidence interval** of a predicted value is: *“95% of intervals of this form will contain the expected value of average qualifying time given a particular year.”*

The interpretation of the **95% prediction interval** of a predicted value is: *“95% of intervals of this form will contain the true qualifying time for this particular year.”*

library(broom)  
  
# calculate the 95% confidence interval for the fit line  
aus\_aug <- augment(aus.lm, data = aus\_quali\_times, interval = "confidence", se\_fit = T)  
  
  
aus\_aug %>%   
 ggplot(aes(Year, Q\_secs)) +  
 geom\_point(position = position\_jitter(seed= 123, h = 0, w = 0.3), alpha = 0.5) +  
 stat\_smooth(method = 'lm', se = F, size = 0.3, alpha = 0.5) +   
 theme\_bw() +  
 labs(x = 'Year',  
 y = 'Best Qualifying Time (secs)',  
 title = 'Australian Grand Prix Qualifying',  
 subtitle = 'Linear Model, 95% CI (2014 - 2023)') +  
 scale\_x\_continuous(breaks = c(2014, 2015, 2016, 2017, 2018, 2019, 2022, 2023)) +  
 geom\_ribbon(aes(ymin = .lower, ymax = .upper), fill = 'grey', alpha = 0.5) +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5))



**The augment() function from the broom package**

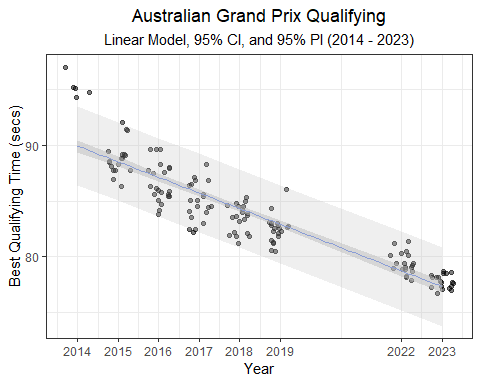
Augment accepts a model object and a dataset and adds information about each observation in the dataset. One of many conveniences provided by augment, is the ability to attach 95% confidence or prediction intervals to a dataframe.

In the example above, I use the following code to attach a predicted value and the 95% confidence interval to the Australia qualifying dataset:

augment(aus.lm, data = aus\_quali\_times, interval = "confidence", se\_fit = T)

For more information on using augment(), visit this link: <https://broom.tidymodels.org/reference/augment.lm.html>

# calculate the 95% prediction interval for the fit line  
aus\_aug\_pr <- augment(aus.lm, data = aus\_quali\_times, interval = "prediction", se\_fit = T) %>%  
rename("lower\_PI" = ".lower", "upper\_PI" = ".upper")  
  
# combine the intervals  
aus\_aug <- aus\_aug %>%  
bind\_cols(aus\_aug\_pr %>% dplyr::select(lower\_PI, upper\_PI))  
  
aus\_aug %>%   
 ggplot(aes(Year, Q\_secs)) +  
 geom\_point(position = position\_jitter(seed= 123, h = 0, w = 0.3), alpha = 0.5) +  
 stat\_smooth(method = 'lm', se = F, size = 0.3, alpha = 0.5) +   
 theme\_bw() +  
 labs(x = 'Year',  
 y = 'Best Qualifying Time (secs)',  
 title = 'Australian Grand Prix Qualifying',  
 subtitle = 'Linear Model, 95% CI, and 95% PI (2014 - 2023)') +  
 scale\_x\_continuous(breaks = c(2014, 2015, 2016, 2017, 2018, 2019, 2022, 2023)) +  
 geom\_ribbon(aes(ymin = lower\_PI, ymax = upper\_PI), fill = 'grey', alpha = 0.25) +  
 geom\_ribbon(aes(ymin = .lower, ymax = .upper), fill = 'grey', alpha = 0.5) +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5))



**Creating a prediction interval using the augment() function**

It’s just as easy to create a 95% prediction interval using augment.

In the example above, I use the following code to attach the 95% prediction interval to the Australia qualifying dataset:

augment(aus.lm, data = aus\_quali\_times, interval = "prediction", se\_fit = T) %>% rename("lower\_PI" = ".lower", "upper\_PI" = ".upper")

To avoid confusion, I renamed the lower and upper bounds of this prediction interval lower\_PI and upper\_PI respectively.

For more information on using augment(), visit this link: <https://broom.tidymodels.org/reference/augment.lm.html>

**The confidence interval is uncertainty surrounding a mean response, and the prediction interval is uncertainty surrounding a prediction of a future observation.**

Or, in our racing example above:

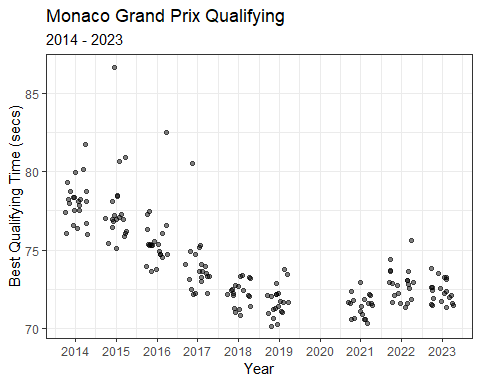
A confidence interval around the fit describes the uncertainty in predicting an *average* time for a given year, while the prediction interval describes the uncertainty in predicting any *single* time for a particular year.

## 4.4 Interaction Model

***How does qualifying time progression compare between the Australian Grand Prix and another race?***

To answer this question, we can take a similar modeling approach that we used for Australia, but apply it to another race. Let’s take a look at the Monaco Grand Prix. I’ll run through the previous modeling steps below.

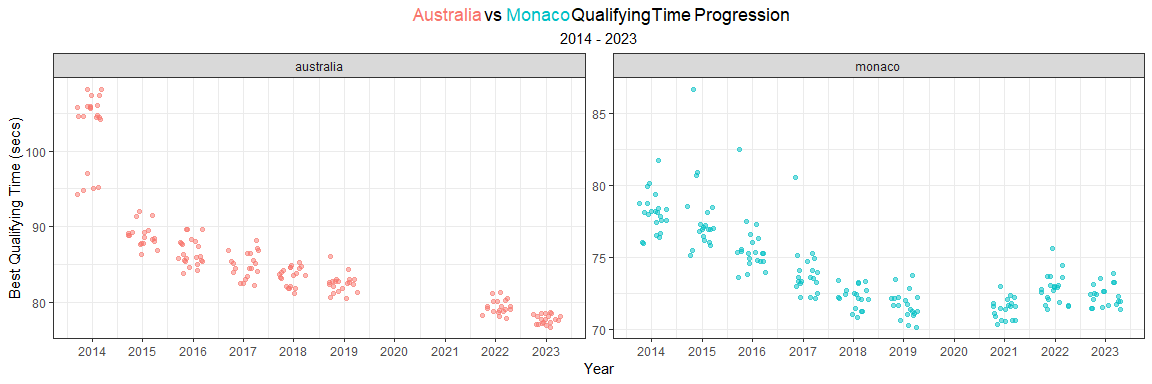
practice\_and\_qualifying %>%  
 filter(Race == 'monaco') %>%  
 ggplot(aes(Year, Q\_secs)) +  
 geom\_point(position = position\_jitter(h = 0, w = 0.3), alpha = 0.5) +  
 theme\_bw() +  
 labs(x = 'Year',  
 y = 'Best Qualifying Time (secs)',  
 title = 'Monaco Grand Prix Qualifying',  
 subtitle = '2014 - 2023') +  
 scale\_x\_continuous(breaks = c(2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023))



Interesting! We did not have data for Australia in 2020 and 2021 (due to the pandemic). In 2022, the new ground-effects regulations were introduced. Because we were missing data for 2020 and 2021, our model assumed a constant decrease in qualifying times.

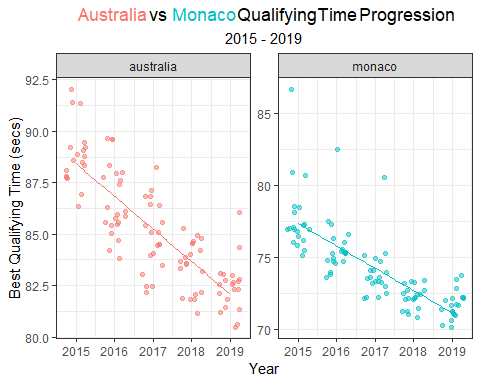
Compare this to Monaco, where we at least had data for 2021. Progression does appear to steadily decrease through 2021, but time then increase in 2022 as the new regulations are introduced. So, if we wish to compare time progression, we should probably limit the data to one set of regulations.

practice\_and\_qualifying %>%  
 filter(Race %in% c('australia', 'monaco')) %>%  
 ggplot(aes(Year, Q\_secs, col = Race)) +  
 geom\_point(position = position\_jitter(h = 0, w = 0.3), alpha = 0.5) +  
 theme\_bw() +  
 labs(x = 'Year',  
 y = 'Best Qualifying Time (secs)',  
 title = title\_color\_coder("", "Australia", '#F8766D', " vs ", "Monaco", '#00BFC4' ," Qualifying Time Progression"),   
 subtitle = '2014 - 2023',  
 col = '') + # use the title\_color\_coder() function  
 facet\_wrap(~ Race, scales= 'free\_y') +  
 scale\_x\_continuous(breaks = c(2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023)) +   
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 legend.position = "none") +  
 theme(plot.title = ggtext::element\_markdown()) ## render the provided text as markdown/html



To quantitatively compare these two progression trends, we can build linear models for each race and compare the output. We should probably remove 2014 data (because Australia times were compromised by the rain), and 2021 data (because the Australian Grand Prix wasn’t held). Below, I’ll re-plot the scatterplot with best-fit linear model lines for each race.

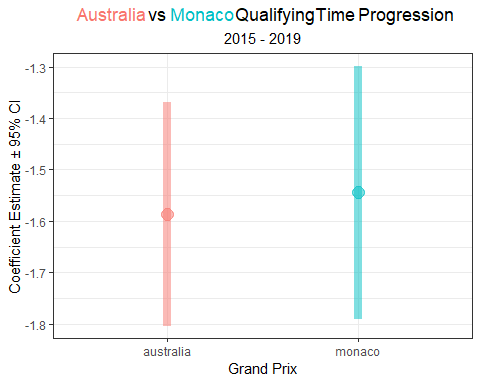
practice\_and\_qualifying %>%  
 filter(Race %in% c('australia', 'monaco'),  
 Year > 2014 & Year < 2020) %>%  
 ggplot(aes(Year, Q\_secs, col = Race)) +  
 geom\_point(position = position\_jitter(h = 0, w = 0.3), alpha = 0.5) +  
 stat\_smooth(method = 'lm', se = F, alpha = 0.5, size = 0.3) +  
 theme\_bw() +  
 labs(x = 'Year',  
 y = 'Best Qualifying Time (secs)',  
 title = title\_color\_coder("", "Australia", '#F8766D', " vs ", "Monaco", '#00BFC4' ," Qualifying Time Progression"),   
 subtitle = '2015 - 2019',  
 col = '') + # use the title\_color\_coder() function  
 facet\_wrap(~ Race, scales= 'free\_y') +  
 scale\_x\_continuous(breaks = c(2014, 2015, 2016, 2017, 2018, 2019)) +   
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 legend.position = "none") +  
 theme(plot.title = ggtext::element\_markdown()) ## render the provided text as markdown/html



With the naked eye, it looks like Australia’s slope is steeper, suggesting that times improved at a quicker rate relative to Monaco. We can calculate the 95% confidence intervals around the slop estimate, and compare.

# filter times to include only clean Australian data  
aus\_quali\_times <- practice\_and\_qualifying %>%  
 filter(Race == 'australia') %>%  
 filter(!is.na(Q\_secs),  
 Year > 2014 & Year < 2020)  
  
# fit linear model for Australia  
aus.lm <- lm(Q\_secs ~ Year, aus\_quali\_times)  
  
# 95% CI slope  
aus\_coef\_ci <- confint(aus.lm)  
  
# filter times to include only clean Australian data  
mon\_quali\_times <- practice\_and\_qualifying %>%  
 filter(Race == 'monaco') %>%  
 filter(!is.na(Q\_secs),  
 Year > 2014 & Year < 2020)  
  
# fit linear model for Monaco  
mon.lm <- lm(Q\_secs ~ Year, data = mon\_quali\_times)  
  
# 95% CI slope  
mon\_coef\_ci <- confint(mon.lm)

aus\_v\_mon <- cbind(data.frame(australia = cbind(est = coef(aus.lm), aus\_coef\_ci)[2,]),   
data.frame(monaco = cbind(est = coef(mon.lm), mon\_coef\_ci)[2,])) %>% t() %>%  
 as.data.frame() %>%  
 rownames\_to\_column(var = 'race')  
  
  
aus\_v\_mon %>%  
 ggplot() +  
 geom\_pointrange(aes(x = race, y = est, ymin = `2.5 %`, ymax = `97.5 %`, col = race),   
 linewidth = 3, alpha = 0.5, size = 1) +  
 theme\_bw() +  
 scale\_color\_manual('', values = c("australia" = '#F8766D', "monaco" = '#00BFC4')) +   
 labs(x = 'Grand Prix', y = 'Coefficient Estimate \u00b1 95% CI',  
 title = title\_color\_coder("", "Australia", '#F8766D', " vs ", "Monaco", '#00BFC4' ," Qualifying Time Progression"),   
 subtitle = '2015 - 2019') +   
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 legend.position = "none") +  
 theme(plot.title = ggtext::element\_markdown()) ## render the provided text as markdown/html



So, while my initial guess that qualifying times decreased at a more rapid rate in Australia, the 95% confidence intervals for the slope estimates largely overlap. This suggests that there’s little to no statistical evidence that these slopes are different.

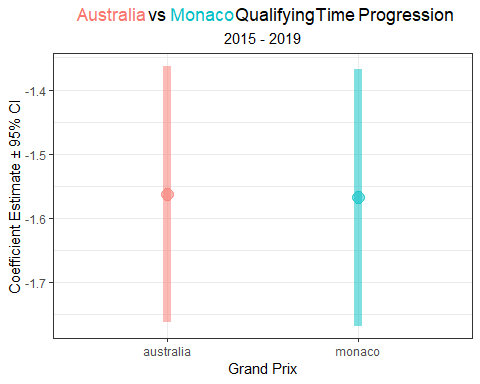
But, the proper way to make this comparison is by using an **interaction model**. It can answer the question that we had (i.e. *Was qualifying time progression different between the two races?*), and it can do a lot more. Below, I’ll fit an interaction model to the 2015 - 2019 data for the Australian and Monaco Grands Prix.

# filter data  
aus\_mon\_data <- practice\_and\_qualifying %>%  
 filter(Race %in% c('australia', 'monaco'),  
 !is.na(Q\_secs),  
 Year > 2014 & Year < 2020) %>%  
 mutate(Race = factor(Race))  
  
# fit an interaction model  
aus.mon.int.lm <- lm(Q\_secs ~ Year:Race, data = aus\_mon\_data)  
  
# print the model output  
summary(aus.mon.int.lm)

##   
## Call:  
## lm(formula = Q\_secs ~ Year:Race, data = aus\_mon\_data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3.0609 -0.9699 -0.1678 0.7731 9.2456   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 3236.76129 166.49551 19.44 <2e-16 \*\*\*  
## Year:Raceaustralia -1.56248 0.08255 -18.93 <2e-16 \*\*\*  
## Year:Racemonaco -1.56793 0.08255 -18.99 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1.638 on 197 degrees of freedom  
## Multiple R-squared: 0.9292, Adjusted R-squared: 0.9285   
## F-statistic: 1294 on 2 and 197 DF, p-value: < 2.2e-16

At first, the output may seem a bit confusing, but it only requires a small adjustment. Basically, this model produces coefficient estimates for a race-specific intercept and slope, relative to the ‘base-level’. In this case, Australia is the base level. Luckily, we can actually use functions from the **emmeans** package to easily estimate the coefficient and confidence intervals by race. **emmeans** stands for *estimated marginal means*, which are means for treatment levels that are adjusted for means of other factors in the model. In this example, we are interested in the influence of year on qualifying times, but it is also impacted by race.

emtrends(aus.mon.int.lm, pairwise ~ Race, var = 'Year') %>%  
 as.data.frame() %>%  
 filter(Race %in% c('australia', 'monaco')) %>%  
 ggplot() +  
 geom\_pointrange(aes(x = Race, y = Year.trend, ymin = lower.CL, ymax = upper.CL, col = Race),   
 linewidth = 3, alpha = 0.5, size = 1) +  
 theme\_bw() +  
 scale\_color\_manual('', values = c("australia" = '#F8766D', "monaco" = '#00BFC4')) +   
 labs(x = 'Grand Prix', y = 'Coefficient Estimate \u00b1 95% CI',  
 title = title\_color\_coder("", "Australia", '#F8766D', " vs ", "Monaco", '#00BFC4' ," Qualifying Time Progression"),   
 subtitle = '2015 - 2019') +   
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 legend.position = "none") +  
 theme(plot.title = ggtext::element\_markdown()) ## render the provided text as markdown/html



**the emmeans package**

Estimated marginal means (EMMs), a.k.a. least-squares means, are predictions on a reference grid of predictor settings, or marginal averages.

To estimate the marginal means of a model, simply run emmeans(model, pairwise ~ treatment). However, more thought should go into the use of this function. This is particularly true for more complicated models.

The emtrends() function is useful when a fitted model involves a numerical predictor x interacting with another predictor a (typically a factor). This si the case with our interaction model where we have year (numeric) interacting with race (factor).

For more information on using the **emmeans** package, visit these links: <https://cran.r-project.org/web/packages/emmeans/vignettes/comparisons.html>

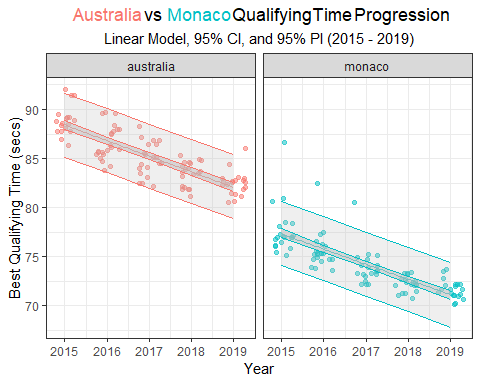
<https://rdrr.io/cran/emmeans/f/vignettes/FAQs.Rmd>

<https://rdrr.io/cran/emmeans/man/emtrends.html>

While the estimates for the slope remain the same to those estimated with individual models, the 95% confidence intervals are narrower. Nonetheless, it doesn’t change our opinion that the rate of qualifying time improvement is not statistically different between these two races during this time period (2015 - 2019).

As we did for a single race earlier, we use the augment() function to estimate the 95% confidence and 95% prediction intervals for this interaction model.

# calculate the 95% confidence interval for the fit line  
aus\_v\_mon\_aug <- augment(aus.mon.int.lm, data = aus\_mon\_data, interval = "confidence", se\_fit = T)  
  
# calculate the 95% prediction interval for the fit line  
aus\_mon\_aug\_pr <- augment(aus.mon.int.lm, data = aus\_mon\_data, interval = "prediction", se\_fit = T) %>%  
rename("lower\_PI" = ".lower", "upper\_PI" = ".upper")  
  
# combine the intervals  
aus\_v\_mon\_aug <- aus\_v\_mon\_aug %>%  
bind\_cols(aus\_mon\_aug\_pr %>% dplyr::select(lower\_PI, upper\_PI))  
  
aus\_v\_mon\_aug %>%   
 ggplot(aes(Year, Q\_secs, col = Race), show.legend = F) +  
 geom\_point(position = position\_jitter(seed= 123, h = 0, w = 0.3), alpha = 0.5) +  
 stat\_smooth(method = 'lm', se = F, size = 0.3, alpha = 0.5) +   
 theme\_bw() +  
 labs(x = 'Year',  
 y = 'Best Qualifying Time (secs)',  
 title = title\_color\_coder("", "Australia", '#F8766D', " vs ", "Monaco", '#00BFC4' ," Qualifying Time Progression"),  
 subtitle = 'Linear Model, 95% CI, and 95% PI (2015 - 2019)') +  
 scale\_x\_continuous(breaks = c(2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023)) +  
 geom\_ribbon(aes(ymin = lower\_PI, ymax = upper\_PI), fill = 'grey', alpha = 0.25) +  
 geom\_ribbon(aes(ymin = .lower, ymax = .upper), fill = 'grey', alpha = 0.5) +   
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 legend.position = "none") +  
 theme(plot.title = ggtext::element\_markdown()) + ## render the provided text as markdown/html  
 facet\_wrap(~ Race)



This visual helps highlight that these slopes are not statistically different from each other.

As a reminder, each Grand Prix’s line describes the linear regression model’s estimate of average qualifying time for each year of the race.The inner band represents the 95% confidence interval of the prediction (confidence in the average response), and the outer band represents the 95% prediction interval of the prediction (confidence in a single response).

And for yet another reminder….

**The confidence interval is uncertainty surrounding a *mean response*, and the prediction interval is uncertainty surrounding a prediction of a *future observation*.**

The confidence interval width is driven by our uncertainty in the coefficient estimates. The prediction interval width is driven by both the uncertainty in the coefficient estimates and *unmodeled* variance. So, even if we had a very large sample size and were very confident in our coefficient estimates, our predicted values would still vary by the residual standard error of the model (i.e. the RSE in the summary() output).

**Residual Standard Error (RSE)**

The Residual Standard Error (RSE) is the standard deviation of the residuals for a model. It is commonly used as a measure of how well a regression model fits a dataset.

In the interaction model above, the RSE is 1.587, meaning that the actual qualifying time with deviate from the true regression line by about 1.587 seconds, on average.

So, a lose interpretation of RSE is:

*RSE, is the average amount that the response will deviate from the true regression line.*

In the next chapter, we will take a closer look at driver performance.

# 5 Drivers

Driver skill is perhaps a surprisingly difficult domain to understand. What are the essential skills of a driver? And how do you measure those skills? From what I have read, the following attributes separate the very best from tehe very good:

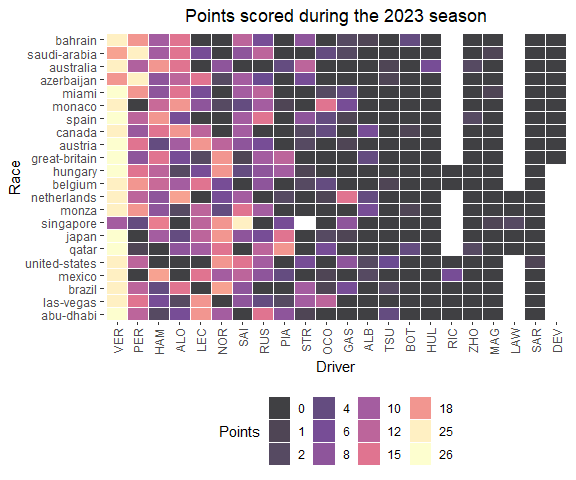
* adaptability
* qualifying pace
* consistent race pace
* tire management
* an understanding of the car’s limits and how to extract all of the performance from a car
* performance in wet conditions

Is there data available that captures differences in these attributes? Probably not! So, what data could we use to explore differences in drivers. The most obvious and simplest place to start is by plotting race results by driver and circuit. I find heat maps an enjoyable way to visualize race results. So, in the following section, I’ll create heat maps for each year from 2014 to 2023.

## 5.1 Race Performance

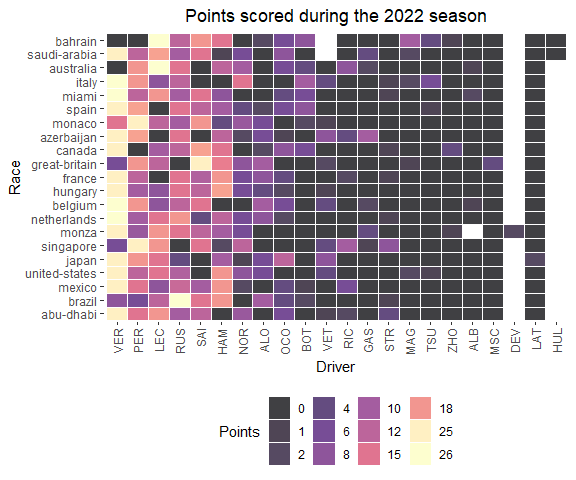
2023 was the Max Verstappen show. Max won his third world title and Red Bull collected to their 2nd consecutive (6th total) constructors championship. From the very first race, Red Bull appeared to be nearly unbeatable. And ultimately, Max completed *arguably* the most dominant season of all time by winning 19 of 22 races and finishing on the podium in all but one GRand Prix. In 2023, Max had a winning percentage of 86.36%, eclipsing the previous long-standing record set by Alberto Ascari in 1952.

library(drs)  
library(ggthemes)  
  
# Pull race results  
races2023 <- race\_result\_scraper(2023)  
  
races2023 %>%  
 mutate( n = 1:n(),  
 Race = fct\_reorder(Race, desc(n))) %>%  
 group\_by(Race) %>%   
 mutate(Race\_number = cur\_group\_id()) %>%  
 group\_by(Driver) %>%   
 mutate(sum\_pts = sum(Points, na.rm = T)) %>%  
 ungroup() %>%   
 ggplot(aes(fct\_reorder(Driver, desc(sum\_pts)), Race, fill = Points)) +  
 geom\_tile(color="white", size=0.1, alpha = 0.75) +  
 theme\_bw() +  
 labs(title = 'Points scored during the 2023 season',  
 y = 'Race',  
 x = 'Driver') +   
 theme\_tufte(base\_family="Helvetica") +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 legend.position="bottom",  
 axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1)) +  
 scale\_fill\_viridis\_c(option = 'magma',  
 guide = 'legend',  
 breaks = c(0,1,2,4,6,8,10,12,15,18,25,26))



The 2022 Formula 1 season was dominated by Red Bull. Max Verstappen secured the Driver’s Championship by the 18th race of the season, the Japanese Grand Prix. However, Red Bull’s 2022 season opened with some uncertainty after experiencing 3 retirements in first two races. both Max Verstappen and Sergio Pérez retired from the Bahrain Grand Prix in the closing laps with fuel issues. In the Australian Grand Prix, Verstappen again retired with fuel issues. Ultimately, Max finished 2022 with 15 wins and 454 points, the most in F1 history. Charles LeClerc took 2nd (308 points) and Sergio Perez took 3rd (305 points) in the Driver’s Championship.

# Pull race results  
races2022 <- race\_result\_scraper(2022)  
  
races2022 %>%  
 mutate( n = 1:n(),  
 Race = fct\_reorder(Race, desc(n))) %>%  
 group\_by(Race) %>%   
 mutate(Race\_number = cur\_group\_id()) %>%  
 group\_by(Driver) %>%   
 mutate(sum\_pts = sum(Points, na.rm = T)) %>%  
 ungroup() %>%   
 ggplot(aes(fct\_reorder(Driver, desc(sum\_pts)), Race, fill = Points)) +  
 geom\_tile(color="white", size=0.1, alpha = 0.75) +  
 theme\_bw() +  
 labs(title = 'Points scored during the 2022 season',  
 y = 'Race',  
 x = 'Driver') +   
 theme\_tufte(base\_family="Helvetica") +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 legend.position="bottom",  
 axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1)) +  
 scale\_fill\_viridis\_c(option = 'magma',  
 guide = 'legend',  
 breaks = c(0,1,2,4,6,8,10,12,15,18,25,26))



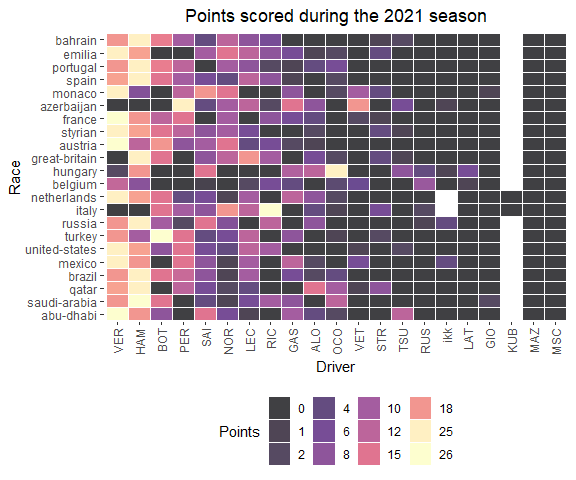
**How to create a heat map in ggplot**

To create a heap map in ggplot2, use the geom\_tile() geometry. You will need to pass categorical variables to x and y arguments and the continuous color fill variable to the fill argument.

For more information on using geom\_tile() in **ggplot2**, visit this link: <https://plotly.com/ggplot2/geom_tile/>

The 2021 Formula 1 season was a modern classic! There was a year-long battle between Red Bull and Mercedes (the reigning 7-time World Constructor’s Champions). Max Verstappen and Lewis Hamilton (the reigning and 7-time World Driver’s Champion) combined to win 18 out of 22 Grands Prix in 2021 (Verstappen with 10 wins, and Hamilton with 8 wins). Both drivers entered the final race of the season (the Abu Dhabi Grand Prix) tied on points. Theis final race, and the 2021 Driver’s Championship, was decided on the final lap of the season when Max Verstappen overtook Hamilton following a safety car restart.

# Pull race results  
races2021 <- race\_result\_scraper(2021)  
  
races2021 %>%  
 mutate( n = 1:n(),  
 Race = fct\_reorder(Race, desc(n))) %>%  
 group\_by(Race) %>%   
 mutate(Race\_number = cur\_group\_id()) %>%  
 group\_by(Driver) %>%   
 mutate(sum\_pts = sum(Points, na.rm = T)) %>%  
 ungroup() %>%   
 ggplot(aes(fct\_reorder(Driver, desc(sum\_pts)), Race, fill = Points)) +  
 geom\_tile(color="white", size=0.1, alpha = 0.75) +  
 theme\_bw() +  
 labs(title = 'Points scored during the 2021 season',  
 y = 'Race',  
 x = 'Driver') +   
 theme\_tufte(base\_family="Helvetica") +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 legend.position="bottom",  
 axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1)) +  
 scale\_fill\_viridis\_c(option = 'magma',  
 guide = 'legend',  
 breaks = c(0,1,2,4,6,8,10,12,15,18,25,26))



**Using a pretty theme for a heat map in ggplot**

The theme\_tufte() function in ggplot2 cleans up the formatting for a figure, by using no border, no axis lines, and no grids.

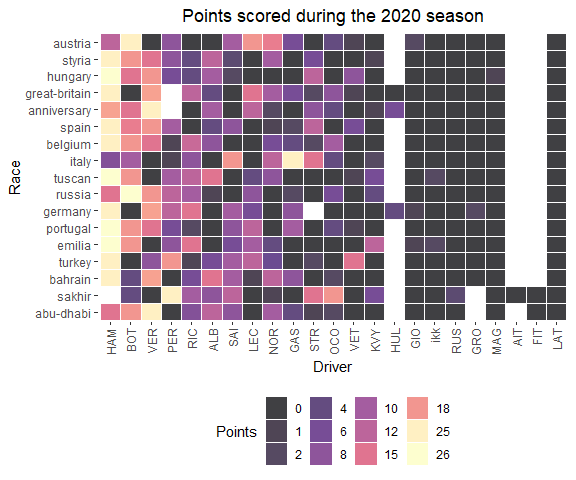
For more information on using theme\_tufte() in **ggplot2**, visit this link: <https://rdrr.io/cran/ggthemes/man/theme_tufte.html>

For more information about Edward Tufte’s principles, see this reference:

Tufte, Edward R. (2001) The Visual Display of Quantitative Information, Chapter 6.

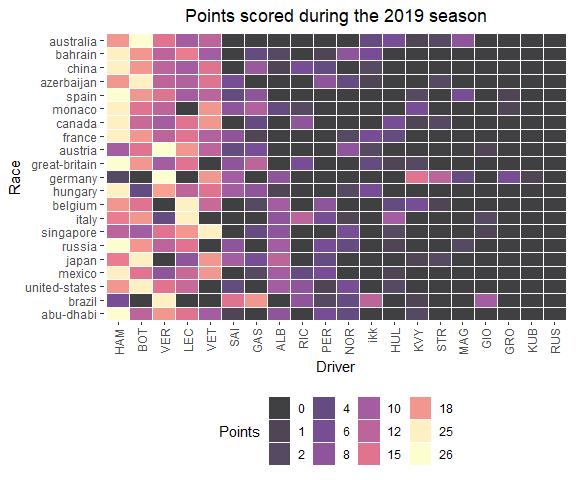
The 2020 season was shortened and delayed due to the COVID-19 pandemic. Originally planned for 22 Grands Prix, this season only contested 17 races and several venues cancelled. Hamilton and Valtteri Bottas finished 1st and 2nd in the Driver’s standings as Mercedes dominated the 2022 season.

# Pull race results  
races2020 <- race\_result\_scraper(2020)  
  
races2020 %>%  
 mutate( n = 1:n(),  
 Race = fct\_reorder(Race, desc(n))) %>%  
 group\_by(Race) %>%   
 mutate(Race\_number = cur\_group\_id()) %>%  
 group\_by(Driver) %>%   
 mutate(sum\_pts = sum(Points, na.rm = T)) %>%  
 ungroup() %>%   
 ggplot(aes(fct\_reorder(Driver, desc(sum\_pts)), Race, fill = Points)) +  
 geom\_tile(color="white", size=0.1, alpha = 0.75) +  
 theme\_bw() +  
 labs(title = 'Points scored during the 2020 season',  
 y = 'Race',  
 x = 'Driver') +   
 theme\_tufte(base\_family="Helvetica") +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 legend.position="bottom",  
 axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1)) +  
 scale\_fill\_viridis\_c(option = 'magma',  
 guide = 'legend',  
 breaks = c(0,1,2,4,6,8,10,12,15,18,25,26))



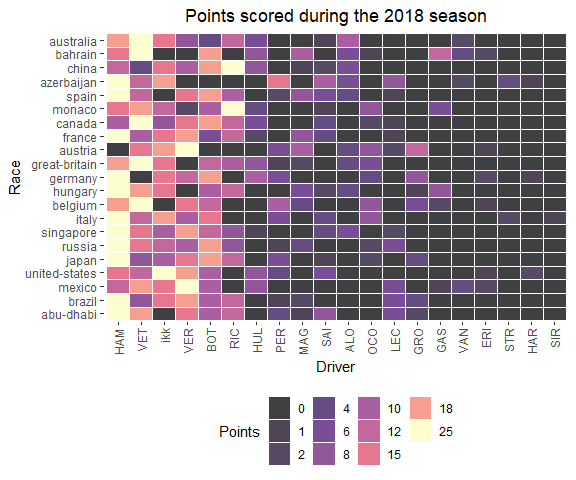
Lewis Hamilton won 11 races and secured his sixth Driver’s Championship in 2019, as Mercedes won a total of 15 our of 21 races. Valtteri Bottas won 4 races, while Verstappen and LeClerc won 3 and 2 races, respectively.

# Pull race results  
races2019 <- race\_result\_scraper(2019)  
  
races2019 %>%  
 mutate( n = 1:n(),  
 Race = fct\_reorder(Race, desc(n))) %>%  
 group\_by(Race) %>%   
 mutate(Race\_number = cur\_group\_id()) %>%  
 group\_by(Driver) %>%   
 mutate(sum\_pts = sum(Points, na.rm = T)) %>%  
 ungroup() %>%   
 ggplot(aes(fct\_reorder(Driver, desc(sum\_pts)), Race, fill = Points)) +  
 geom\_tile(color="white", size=0.1, alpha = 0.75) +  
 theme\_bw() +  
 labs(title = 'Points scored during the 2019 season',  
 y = 'Race',  
 x = 'Driver') +   
 theme\_tufte(base\_family="Helvetica") +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 legend.position="bottom",  
 axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1)) +  
 scale\_fill\_viridis\_c(option = 'magma',  
 guide = 'legend',  
 breaks = c(0,1,2,4,6,8,10,12,15,18,25,26))



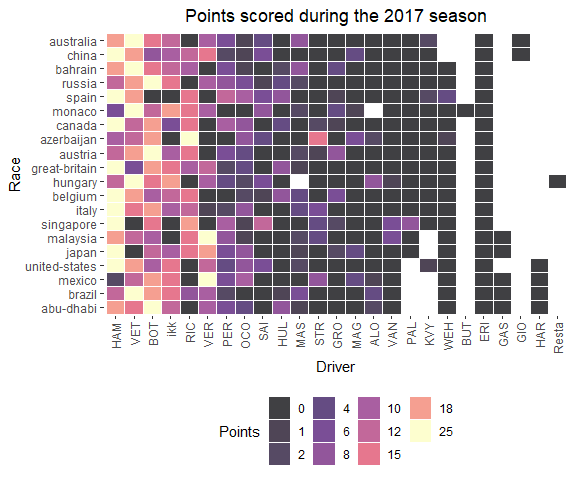
Four-time World Champions Lewis Hamilton and Sebastian Vettel battled during 2018. This was the first season in which two four-time world champions battled for a fifth championship. The Championship lead swapped hands between Hamilton and Vettel five times during 2018, and was ultimately secured by Hamilton at the Mexican Grand Prix.

# Pull race results  
races2018 <- race\_result\_scraper(2018)  
  
races2018 %>%  
 mutate( n = 1:n(),  
 Race = fct\_reorder(Race, desc(n))) %>%  
 group\_by(Race) %>%   
 mutate(Race\_number = cur\_group\_id()) %>%  
 group\_by(Driver) %>%   
 mutate(sum\_pts = sum(Points, na.rm = T)) %>%  
 ungroup() %>%   
 ggplot(aes(fct\_reorder(Driver, desc(sum\_pts)), Race, fill = Points)) +  
 geom\_tile(color="white", size=0.1, alpha = 0.75) +  
 theme\_bw() +  
 labs(title = 'Points scored during the 2018 season',  
 y = 'Race',  
 x = 'Driver') +   
 theme\_tufte(base\_family="Helvetica") +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 legend.position="bottom",  
 axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1)) +  
 scale\_fill\_viridis\_c(option = 'magma',  
 guide = 'legend',  
 breaks = c(0,1,2,4,6,8,10,12,15,18,25,26))



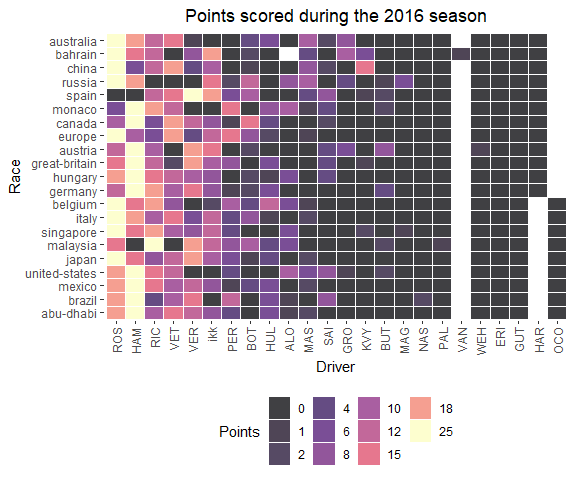
2017 marked the first time during Mercedes’ turbo-hybrid era reign that another team presented a legitimate challenge. Sebastian Vettel led the championship deep into this season, but Hamilton ultimately won his 4th title by 46 points.

# Pull race results  
races2017 <- race\_result\_scraper(2017)  
  
races2017 %>%  
 mutate( n = 1:n(),  
 Race = fct\_reorder(Race, desc(n))) %>%  
 group\_by(Race) %>%   
 mutate(Race\_number = cur\_group\_id()) %>%  
 group\_by(Driver) %>%   
 mutate(sum\_pts = sum(Points, na.rm = T)) %>%  
 ungroup() %>%   
 ggplot(aes(fct\_reorder(Driver, desc(sum\_pts)), Race, fill = Points)) +  
 geom\_tile(color="white", size=0.1, alpha = 0.75) +  
 theme\_bw() +  
 labs(title = 'Points scored during the 2017 season',  
 y = 'Race',  
 x = 'Driver') +   
 theme\_tufte(base\_family="Helvetica") +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 legend.position="bottom",  
 axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1)) +  
 scale\_fill\_viridis\_c(option = 'magma',  
 guide = 'legend',  
 breaks = c(0,1,2,4,6,8,10,12,15,18,25,26))



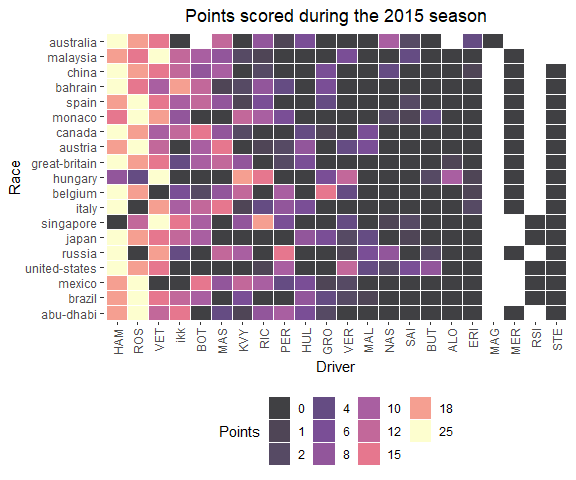
Nico Rosberg won his only World Driver’s Championship title in 2016. During an intense intrateam battle, Rosberg won the first four races while Hamilton won the final four. Rosberg secured his title over teammate Hamilton in the final race of the season. Shortly after winning the title, Rosberg announced his retirement from Formula 1.

# Pull race results  
races2016 <- race\_result\_scraper(2016)  
  
races2016 %>%  
 mutate( n = 1:n(),  
 Race = fct\_reorder(Race, desc(n))) %>%  
 group\_by(Race) %>%   
 mutate(Race\_number = cur\_group\_id()) %>%  
 group\_by(Driver) %>%   
 mutate(sum\_pts = sum(Points, na.rm = T)) %>%  
 ungroup() %>%   
 ggplot(aes(fct\_reorder(Driver, desc(sum\_pts)), Race, fill = Points)) +  
 geom\_tile(color="white", size=0.1, alpha = 0.75) +  
 theme\_bw() +  
 labs(title = 'Points scored during the 2016 season',  
 y = 'Race',  
 x = 'Driver') +   
 theme\_tufte(base\_family="Helvetica") +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 legend.position="bottom",  
 axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1)) +  
 scale\_fill\_viridis\_c(option = 'magma',  
 guide = 'legend',  
 breaks = c(0,1,2,4,6,8,10,12,15,18,25,26))



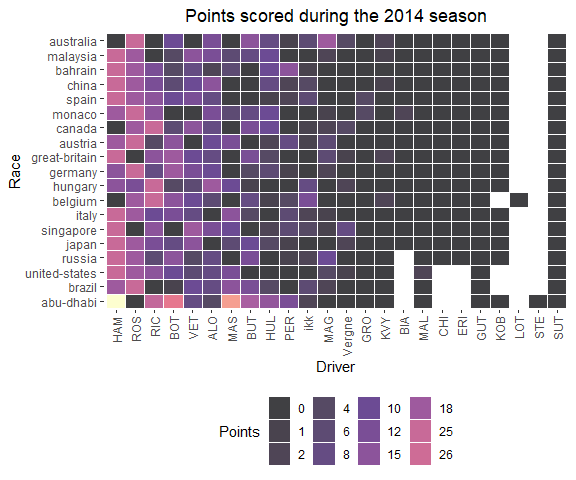
Lewis Hamilton defended his World Championship in 2015, securing the title with three races to go in the season. Nico Rosberg (Hamilton’s teammate) and Sebastian Vettel finished 2nd and 3rd, respectively.

# Pull race results  
races2015 <- race\_result\_scraper(2015)  
  
races2015 %>%  
 mutate( n = 1:n(),  
 Race = fct\_reorder(Race, desc(n))) %>%  
 group\_by(Race) %>%   
 mutate(Race\_number = cur\_group\_id()) %>%  
 group\_by(Driver) %>%   
 mutate(sum\_pts = sum(Points, na.rm = T)) %>%  
 ungroup() %>%   
 ggplot(aes(fct\_reorder(Driver, desc(sum\_pts)), Race, fill = Points)) +  
 geom\_tile(color="white", size=0.1, alpha = 0.75) +  
 theme\_bw() +  
 labs(title = 'Points scored during the 2015 season',  
 y = 'Race',  
 x = 'Driver') +   
 theme\_tufte(base\_family="Helvetica") +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 legend.position="bottom",  
 axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1)) +  
 scale\_fill\_viridis\_c(option = 'magma',  
 guide = 'legend',  
 breaks = c(0,1,2,4,6,8,10,12,15,18,25,26))



The 2014 season marked the introduction of a new engine formula. The previous era’s 2.4 liter V8 engine was replaced by a 1.6 liter turbocharged V6 engine with an energy recovery system. As a result, Mercedes experienced a great engine advantage in 2014, an edge that carried through the entirety of this turbo-hybrid era. Mercedes won their first Constructor’s Championship by nearly 300 points, and Hamilton won bis second Driver’s Championship.

# Pull race results  
races2014 <- race\_result\_scraper(2014)  
  
races2014 %>%  
 mutate( n = 1:n(),  
 Race = fct\_reorder(Race, desc(n))) %>%  
 group\_by(Race) %>%   
 mutate(Race\_number = cur\_group\_id()) %>%  
 group\_by(Driver) %>%   
 mutate(sum\_pts = sum(Points, na.rm = T)) %>%  
 ungroup() %>%   
 ggplot(aes(fct\_reorder(Driver, desc(sum\_pts)), Race, fill = Points)) +  
 geom\_tile(color="white", size=0.1, alpha = 0.75) +  
 theme\_bw() +  
 labs(title = 'Points scored during the 2014 season',  
 y = 'Race',  
 x = 'Driver') +   
 theme\_tufte(base\_family="Helvetica") +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 legend.position="bottom",  
 axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1)) +  
 scale\_fill\_viridis\_c(option = 'magma',  
 guide = 'legend',  
 breaks = c(0,1,2,4,6,8,10,12,15,18,25,26))

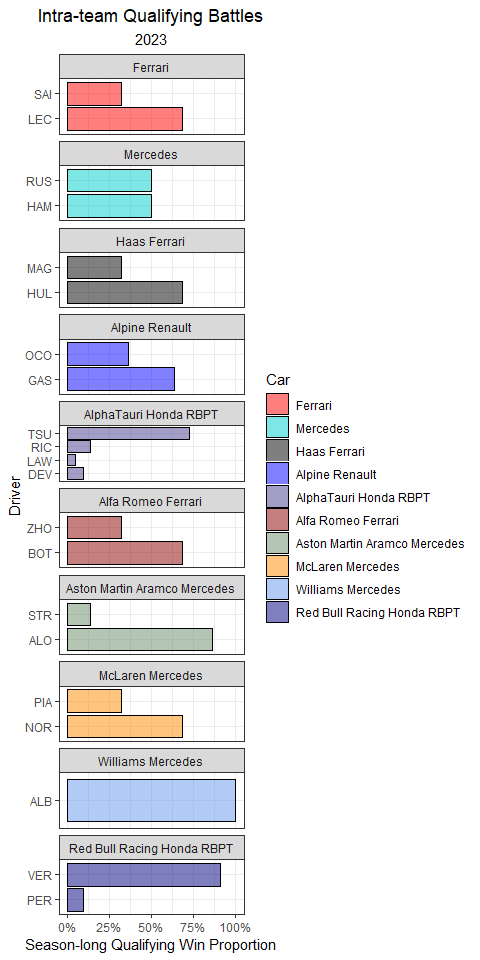


## 5.2 Qualifying

Qualifying pace, or a driver’s flat-out speed over 1 lap, is another attribute that sets drivers apart. The driver effect can sometimes be difficult to separate from a car’s attributes. However, looking at the long-term performance of drivers in qualifying can reveal that some drivers are simply more capable of very fast qualifying times.

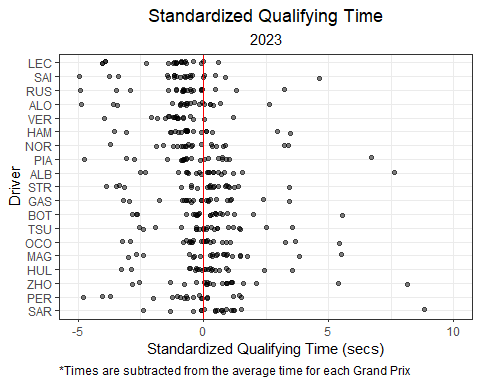
### 5.2.1 2023

quali2023 <- qualifying\_scraper(2023)  
  
quali2023 %>%  
 ungroup() %>%  
 dplyr::select(Race, Position, Car, Driver) %>%  
 mutate(Position = as.integer(Position),  
 Position = ifelse(is.na(Position), 21, Position)) %>%   
 group\_by(Car, Race) %>%   
 arrange(Driver) %>%   
 mutate(driver\_num = 1:n()) %>%   
 pivot\_wider(names\_from = 'driver\_num', values\_from = c('Driver', 'Position')) %>%  
 mutate(best\_qualifier = ifelse(Position\_1 < Position\_2, Driver\_1, Driver\_2)) %>%  
 ungroup() %>%  
 group\_by(Car) %>%  
 count(best\_qualifier) %>%  
 mutate(percentage = n / sum(n)) %>%  
 filter(!is.na(best\_qualifier)) %>%   
 ggplot(aes(y = fct\_reorder(best\_qualifier, Car), x = percentage, fill = Car)) +  
 geom\_bar(stat = 'identity', position = 'dodge', alpha = 0.5, col = 'black') +  
 theme\_bw() +  
 scale\_x\_continuous(labels = scales::percent\_format()) +  
 facet\_wrap(~ factor(Car,  
 levels = c("Ferrari",  
 "Mercedes",  
 "Haas Ferrari",  
 "Alpine Renault",  
 "AlphaTauri Honda RBPT",  
 "Alfa Romeo Ferrari",   
 "Aston Martin Aramco Mercedes",  
 "McLaren Mercedes",  
 "Williams Mercedes",  
 "Red Bull Racing Honda RBPT")),  
 ncol = 1, strip.position="top", scales = 'free\_y') +  
 labs(y = 'Driver',  
 x = 'Season-long Qualifying Win Proportion',  
 title = 'Intra-team Qualifying Battles',  
 subtitle = '2023') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 scale\_fill\_manual(values = c("red",   
 "cyan3",   
 "black",   
 "blue",  
 "darkslateblue",   
 "darkred",   
 "darkseagreen4",   
 "darkorange",   
 "cornflowerblue",  
 "navy"),  
 breaks = c( "Ferrari",  
 "Mercedes",  
 "Haas Ferrari",  
 "Alpine Renault",  
 "AlphaTauri Honda RBPT",  
 "Alfa Romeo Ferrari",   
 "Aston Martin Aramco Mercedes",  
 "McLaren Mercedes",  
 "Williams Mercedes",  
 "Red Bull Racing Honda RBPT"))



In the figure below, I plot each driver’s standardized qualifying time (standardized to a given circuit).

qualifying\_allyears %>%  
 filter(!is.na(Q\_secs),  
 Year == 2023) %>%  
 group\_by(Race) %>%  
 mutate(track\_mean = mean(Q\_secs, na.rm = T),  
 Time\_std\_track = Q\_secs - track\_mean) %>%   
 ungroup() %>%   
 group\_by(Driver) %>%   
 mutate(driver\_n = n(),  
 mean = mean(Time\_std\_track, na.rm = T)) %>%  
 ungroup() %>%  
 dplyr::filter(driver\_n > 15) %>%   
 ggplot(aes(Time\_std\_track, y = fct\_reorder(Driver, desc(mean)))) +  
 geom\_point(position = position\_jitter(w = 0, h = 0.1), alpha = 0.5) +  
 theme\_bw() +  
 labs(x = 'Standardized Qualifying Time (secs)',  
 y = 'Driver',  
 title = 'Standardized Qualifying Time',  
 subtitle = '2023',  
 caption = '\*Times are subtracted from the average time for each Grand Prix') +  
 geom\_vline(xintercept = 0, col = 'red') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 xlim(-5, 10)

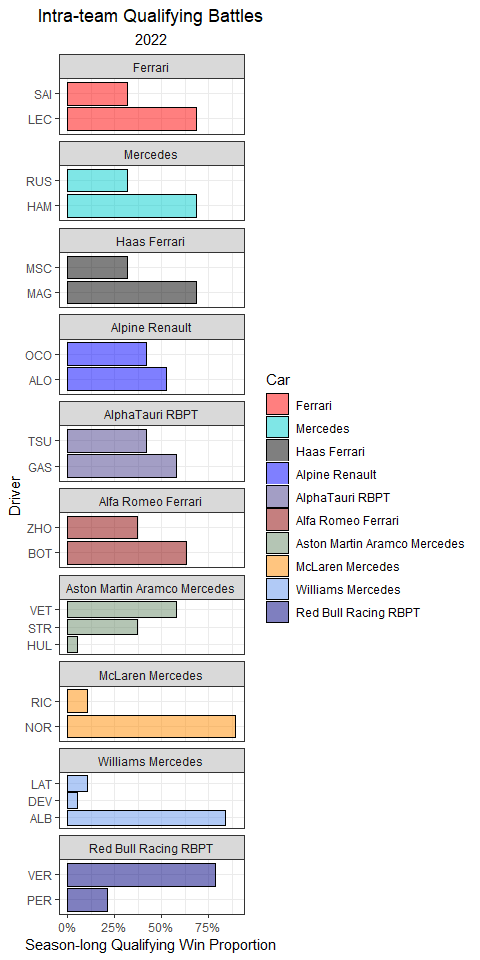


### 5.2.2 2022

Charles Leclerc had the most most poles in 2022 (9), followed my Max Verstappen with 7. Some could argue that the Ferrari car was more suited for qualifying pace, but it’s difficult to definitely say the Ferrari was consistently the faster car over 1 lap. Leclerc and Verstappen both out-qualified their teammates by a wide margin (9-3 in favor of LeClerc and 7-1 in favor of Varstappen). Leclerc’s worst performance was 15th in Canada, while Verstappen’s worst qualifying was 10th in Hungary. Additionally, Verstappen made it to Q3 in every weekend of the season.

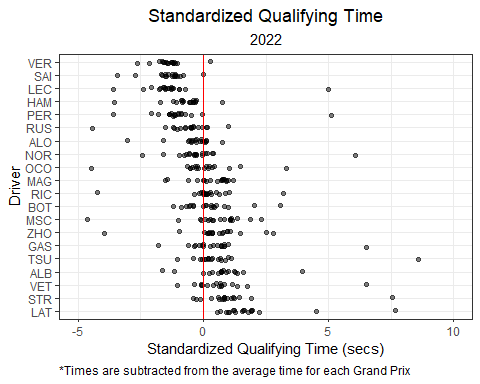
While the vast majority of the drivers on the grid were not in serious contenion for poles, some drivers clearly outperformed their teammates in 2022. For instance, Alex Albon outqualified teammate Nicholas Latifi by a margin of 19-2.

quali2022 <- qualifying\_scraper(2022)  
  
quali2022 %>%  
 ungroup() %>%  
 dplyr::select(Race, Position, Car, Driver) %>%  
 mutate(Position = as.integer(Position),  
 Position = ifelse(is.na(Position), 21, Position)) %>%   
 group\_by(Car, Race) %>%   
 arrange(Driver) %>%   
 mutate(driver\_num = 1:n()) %>%   
 pivot\_wider(names\_from = 'driver\_num', values\_from = c('Driver', 'Position')) %>%  
 mutate(best\_qualifier = ifelse(Position\_1 < Position\_2, Driver\_1, Driver\_2)) %>%  
 ungroup() %>%  
 group\_by(Car) %>%  
 count(best\_qualifier) %>%  
 mutate(percentage = n / sum(n)) %>%  
 filter(!is.na(best\_qualifier)) %>%   
 ggplot(aes(y = fct\_reorder(best\_qualifier, Car), x = percentage, fill = Car)) +  
 geom\_bar(stat = 'identity', position = 'dodge', alpha = 0.5, col = 'black') +  
 theme\_bw() +  
 scale\_x\_continuous(labels = scales::percent\_format()) +  
 facet\_wrap(~ factor(Car,  
 levels = c("Ferrari",  
 "Mercedes",  
 "Haas Ferrari",  
 "Alpine Renault",  
 "AlphaTauri RBPT",  
 "Alfa Romeo Ferrari",   
 "Aston Martin Aramco Mercedes",  
 "McLaren Mercedes",  
 "Williams Mercedes",  
 "Red Bull Racing RBPT")),  
 ncol = 1, strip.position="top", scales = 'free\_y') +  
 labs(y = 'Driver',  
 x = 'Season-long Qualifying Win Proportion',  
 title = 'Intra-team Qualifying Battles',  
 subtitle = '2022') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 scale\_fill\_manual(values = c("red",   
 "cyan3",   
 "black",   
 "blue",  
 "darkslateblue",   
 "darkred",   
 "darkseagreen4",   
 "darkorange",   
 "cornflowerblue",  
 "navy"),  
 breaks = c( "Ferrari",  
 "Mercedes",  
 "Haas Ferrari",  
 "Alpine Renault",  
 "AlphaTauri RBPT",  
 "Alfa Romeo Ferrari",   
 "Aston Martin Aramco Mercedes",  
 "McLaren Mercedes",  
 "Williams Mercedes",  
 "Red Bull Racing RBPT"))



It’s pretty clear that both LeClerc and Verstappen stood apart from any other driver in 2022.

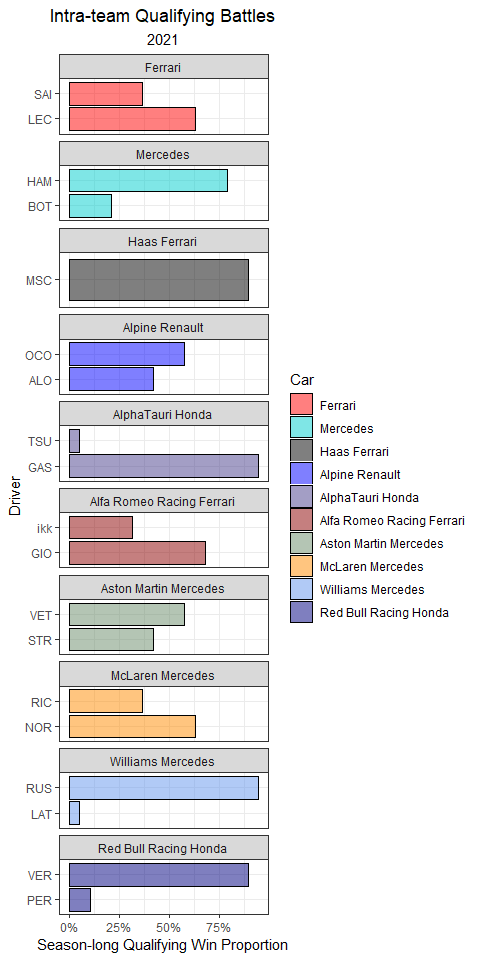
qualifying\_allyears %>%  
 filter(!is.na(Q\_secs),  
 Year == 2022) %>%  
 group\_by(Race) %>%  
 mutate(track\_mean = mean(Q\_secs, na.rm = T),  
 Time\_std\_track = Q\_secs - track\_mean) %>%   
 ungroup() %>%   
 group\_by(Driver) %>%   
 mutate(driver\_n = n(),  
 mean = mean(Time\_std\_track, na.rm = T)) %>%  
 ungroup() %>%  
 dplyr::filter(driver\_n > 15) %>%   
 ggplot(aes(Time\_std\_track, y = fct\_reorder(Driver, desc(mean)))) +  
 geom\_point(position = position\_jitter(w = 0, h = 0.1), alpha = 0.5) +  
 theme\_bw() +  
 labs(x = 'Standardized Qualifying Time (secs)',  
 y = 'Driver',  
 title = 'Standardized Qualifying Time',  
 subtitle = '2022',  
 caption = '\*Times are subtracted from the average time for each Grand Prix') +  
 geom\_vline(xintercept = 0, col = 'red') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 xlim(-5, 10)



### 5.2.3 2021

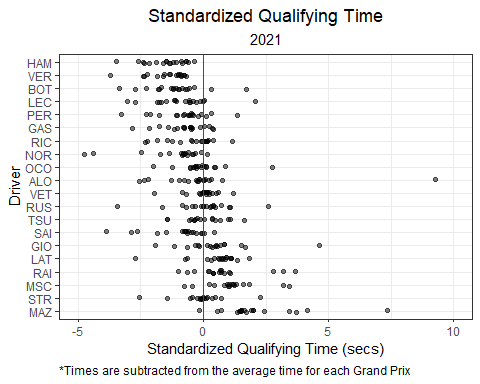
During 2021, some head-to-head battles were particularly lop-sided. Pierre Gasly out-qualified Yuki Tsunoda in 21 out of 22 races, while Verstappen out-qualified Perez in 20 out of 22.

quali2021 <- qualifying\_scraper(2021)  
  
quali2021 %>%  
 ungroup() %>%  
 dplyr::select(Race, Position, Car, Driver) %>%  
 mutate(Position = as.integer(Position),  
 Position = ifelse(is.na(Position), 21, Position)) %>%   
 group\_by(Car, Race) %>%   
 arrange(Driver) %>%   
 mutate(driver\_num = 1:n()) %>%   
 pivot\_wider(names\_from = 'driver\_num', values\_from = c('Driver', 'Position')) %>%  
 mutate(best\_qualifier = ifelse(Position\_1 < Position\_2, Driver\_1, Driver\_2)) %>%  
 ungroup() %>%  
 group\_by(Car) %>%  
 count(best\_qualifier) %>%  
 mutate(percentage = n / sum(n)) %>%  
 filter(!is.na(best\_qualifier)) %>%  
 ggplot(aes(y = fct\_reorder(best\_qualifier, Car), x = percentage, fill = Car)) +  
 geom\_bar(stat = 'identity', position = 'dodge', alpha = 0.5, col = 'black') +  
 theme\_bw() +  
 scale\_x\_continuous(labels = scales::percent\_format()) +  
 facet\_wrap(~ factor(Car,  
 levels = c("Ferrari",  
 "Mercedes",  
 "Haas Ferrari",  
 "Alpine Renault",  
 "AlphaTauri Honda",  
 "Alfa Romeo Racing Ferrari",   
 "Aston Martin Mercedes",  
 "McLaren Mercedes",  
 "Williams Mercedes",  
 "Red Bull Racing Honda")), ncol = 1, strip.position="top", scales = 'free\_y') +  
 labs(y = 'Driver',  
 x = 'Season-long Qualifying Win Proportion',  
 title = 'Intra-team Qualifying Battles',  
 subtitle = '2021') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 scale\_fill\_manual(values = c("red",   
 "cyan3",   
 "black",   
 "blue",  
 "darkslateblue",   
 "darkred",   
 "darkseagreen4",   
 "darkorange",   
 "cornflowerblue",  
 "navy"),  
 breaks = c("Ferrari",  
 "Mercedes",  
 "Haas Ferrari",  
 "Alpine Renault",  
 "AlphaTauri Honda",  
 "Alfa Romeo Racing Ferrari",   
 "Aston Martin Mercedes",  
 "McLaren Mercedes",  
 "Williams Mercedes",  
 "Red Bull Racing Honda"))



Max Verstappen (10) and Lewis Hamilton (5) took 15 of the 22 possible poles in 2021. In fact, Red Bull and Mercedes combined to take all but 3 of the 2021 pole positions.

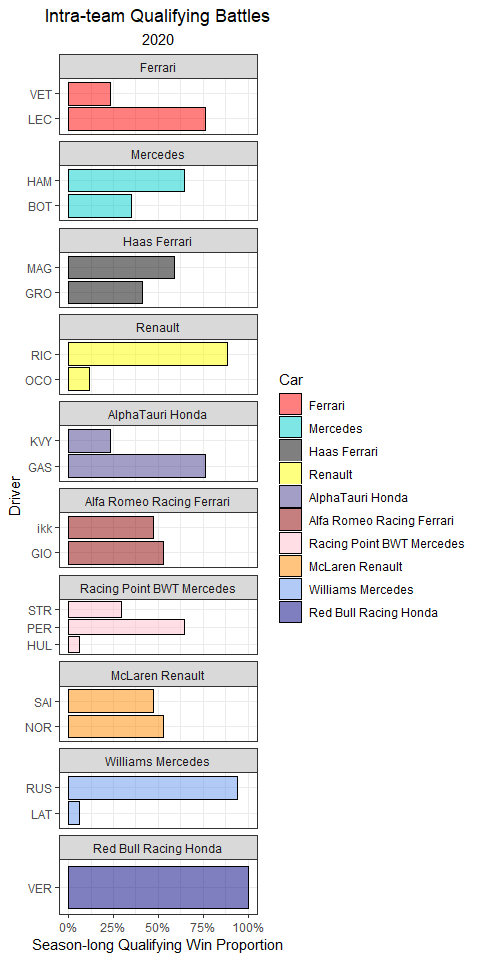
qualifying\_allyears %>%  
 mutate(Driver = ifelse(Driver == 'ikk', 'RAI', Driver)) %>%   
 filter(!is.na(Q\_secs),  
 Year == 2021) %>%  
 group\_by(Race) %>%  
 mutate(track\_mean = mean(Q\_secs, na.rm = T),  
 Time\_std\_track = Q\_secs - track\_mean) %>%   
 ungroup() %>%   
 group\_by(Driver) %>%   
 mutate(driver\_n = n(),  
 mean = mean(Time\_std\_track, na.rm = T)) %>%  
 ungroup() %>%  
 dplyr::filter(driver\_n > 15) %>%   
 ggplot(aes(Time\_std\_track, y = fct\_reorder(Driver, desc(mean)))) +  
 geom\_point(position = position\_jitter(w = 0, h = 0.1), alpha = 0.5) +  
 theme\_bw() +  
 labs(x = 'Standardized Qualifying Time (secs)',  
 y = 'Driver',  
 title = 'Standardized Qualifying Time',  
 subtitle = '2021',  
 caption = '\*Times are subtracted from the average time for each Grand Prix') +  
 geom\_vline(xintercept = 0, col = 'red') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 xlim(-5, 10)



### 5.2.4 2020

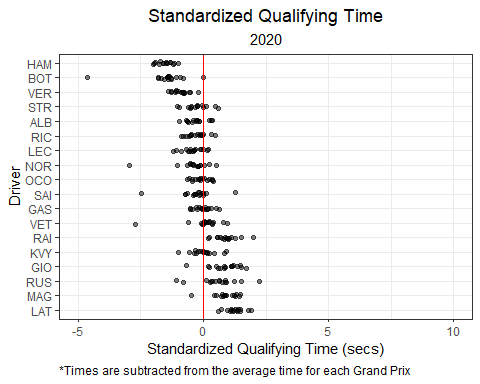
Mercedes dominated the COVID-shortened 2020 season, winning all but two poles. Lance Stroll took his first career pole during a rainy qualifying session at the Turkish Grand Prix. In the final Grand Prix of the year, Verstappen qualified on pole with the only car *not* powered by a Mercedes engine in 2020.

quali2020 <- qualifying\_scraper(2020)  
  
quali2020 %>%  
 ungroup() %>%  
 dplyr::select(Race, Position, Car, Driver) %>%  
 mutate(Position = as.integer(Position),  
 Position = ifelse(is.na(Position), 21, Position)) %>%   
 group\_by(Car, Race) %>%   
 arrange(Driver) %>%   
 mutate(driver\_num = 1:n()) %>%   
 pivot\_wider(names\_from = 'driver\_num', values\_from = c('Driver', 'Position')) %>%  
 mutate(best\_qualifier = ifelse(Position\_1 < Position\_2, Driver\_1, Driver\_2)) %>%  
 ungroup() %>%  
 group\_by(Car) %>%  
 count(best\_qualifier) %>%  
 mutate(percentage = n / sum(n)) %>%  
 filter(!is.na(best\_qualifier)) %>%  
 ggplot(aes(y = fct\_reorder(best\_qualifier, Car), x = percentage, fill = Car)) +  
 geom\_bar(stat = 'identity', position = 'dodge', alpha = 0.5, col = 'black') +  
 theme\_bw() +  
 scale\_x\_continuous(labels = scales::percent\_format()) +  
 facet\_wrap(~ factor(Car,  
 levels = c("Ferrari",  
 "Mercedes",  
 "Haas Ferrari",  
 "Renault",  
 "AlphaTauri Honda",  
 "Alfa Romeo Racing Ferrari",   
 "Racing Point BWT Mercedes",  
 "McLaren Renault",  
 "Williams Mercedes",  
 "Red Bull Racing Honda")), ncol = 1, strip.position="top", scales = 'free\_y') +  
 labs(y = 'Driver',  
 x = 'Season-long Qualifying Win Proportion',  
 title = 'Intra-team Qualifying Battles',  
 subtitle = '2020') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 scale\_fill\_manual(values = c("red",   
 "cyan3",   
 "black",   
 "yellow",  
 "darkslateblue",   
 "darkred",   
 "pink",   
 "darkorange",   
 "cornflowerblue",  
 "navy"),  
 breaks = c("Ferrari",  
 "Mercedes",  
 "Haas Ferrari",  
 "Renault",  
 "AlphaTauri Honda",  
 "Alfa Romeo Racing Ferrari",   
 "Racing Point BWT Mercedes",  
 "McLaren Renault",  
 "Williams Mercedes",  
 "Red Bull Racing Honda"))



And, here’s a look at 2020’s standardized qualifying times.

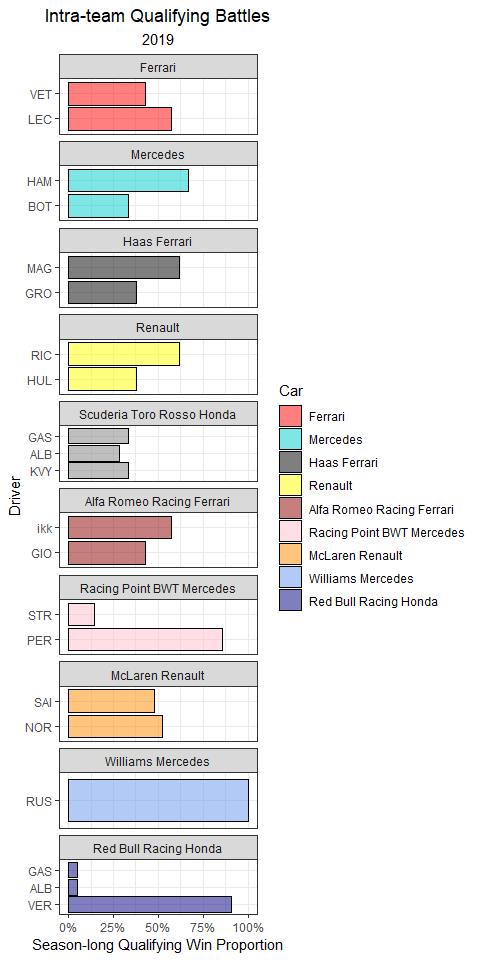
qualifying\_allyears %>%  
 mutate(Driver = ifelse(Driver == 'ikk', 'RAI', Driver)) %>%   
 filter(!is.na(Q\_secs),  
 Year == 2020) %>%  
 group\_by(Race) %>%  
 mutate(track\_mean = mean(Q\_secs, na.rm = T),  
 Time\_std\_track = Q\_secs - track\_mean) %>%   
 ungroup() %>%   
 group\_by(Driver) %>%   
 mutate(driver\_n = n(),  
 mean = mean(Time\_std\_track, na.rm = T)) %>%  
 ungroup() %>%  
 dplyr::filter(driver\_n > 15) %>%   
 ggplot(aes(Time\_std\_track, y = fct\_reorder(Driver, desc(mean)))) +  
 geom\_point(position = position\_jitter(w = 0, h = 0.1), alpha = 0.5) +  
 theme\_bw() +  
 labs(x = 'Standardized Qualifying Time (secs)',  
 y = 'Driver',  
 title = 'Standardized Qualifying Time',  
 subtitle = '2020',  
 caption = '\*Times are subtracted from the average time for each Grand Prix') +  
 geom\_vline(xintercept = 0, col = 'red') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 xlim(-5, 10)



### 5.2.5 2019

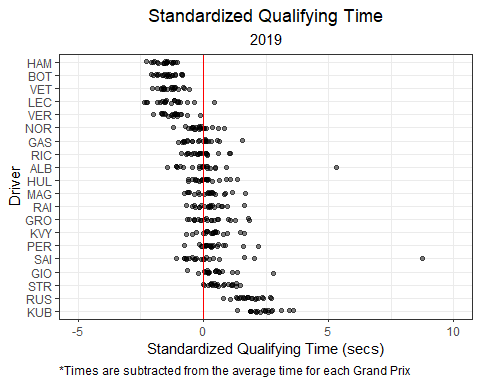
Qualifying during 2019 was quite close between Mercedes (10 poles) and Ferrari (9 poles). Charles LeClerc qualified on pole 7 times, while Valterri Bottas and Lewis Hamilton both finished with 5 pole positions.

quali2019 <- qualifying\_scraper(2019)  
  
quali2019 %>%  
 ungroup() %>%  
 dplyr::select(Race, Position, Car, Driver) %>%  
 mutate(Position = as.integer(Position),  
 Position = ifelse(is.na(Position), 21, Position)) %>%   
 group\_by(Car, Race) %>%   
 arrange(Driver) %>%   
 mutate(driver\_num = 1:n()) %>%   
 pivot\_wider(names\_from = 'driver\_num', values\_from = c('Driver', 'Position')) %>%  
 mutate(best\_qualifier = ifelse(Position\_1 < Position\_2, Driver\_1, Driver\_2)) %>%  
 ungroup() %>%  
 group\_by(Car) %>%  
 count(best\_qualifier) %>%  
 mutate(percentage = n / sum(n)) %>%  
 filter(!is.na(best\_qualifier)) %>%  
 ggplot(aes(y = fct\_reorder(best\_qualifier, Car), x = percentage, fill = Car)) +  
 geom\_bar(stat = 'identity', position = 'dodge', alpha = 0.5, col = 'black') +  
 theme\_bw() +  
 scale\_x\_continuous(labels = scales::percent\_format()) +  
 facet\_wrap(~ factor(Car,  
 levels = c("Ferrari",  
 "Mercedes",  
 "Haas Ferrari",  
 "Renault",  
 "Scuderia Toro Rosso Honda",  
 "Alfa Romeo Racing Ferrari",   
 "Racing Point BWT Mercedes",  
 "McLaren Renault",  
 "Williams Mercedes",  
 "Red Bull Racing Honda")), ncol = 1, strip.position="top", scales = 'free\_y') +  
 labs(y = 'Driver',  
 x = 'Season-long Qualifying Win Proportion',  
 title = 'Intra-team Qualifying Battles',  
 subtitle = '2019') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 scale\_fill\_manual(values = c("red",   
 "cyan3",   
 "black",   
 "yellow",  
 "darkslateblue",   
 "darkred",   
 "pink",   
 "darkorange",   
 "cornflowerblue",  
 "navy"),  
 breaks = c("Ferrari",  
 "Mercedes",  
 "Haas Ferrari",  
 "Renault",  
 "AlphaTauri Honda",  
 "Alfa Romeo Racing Ferrari",   
 "Racing Point BWT Mercedes",  
 "McLaren Renault",  
 "Williams Mercedes",  
 "Red Bull Racing Honda"))



When we look at the standardized qualifying times, five drivers are clearly faster than the rest of the grid: both Mercedes drivers, both Ferrari drivers, and Max Verstappen.

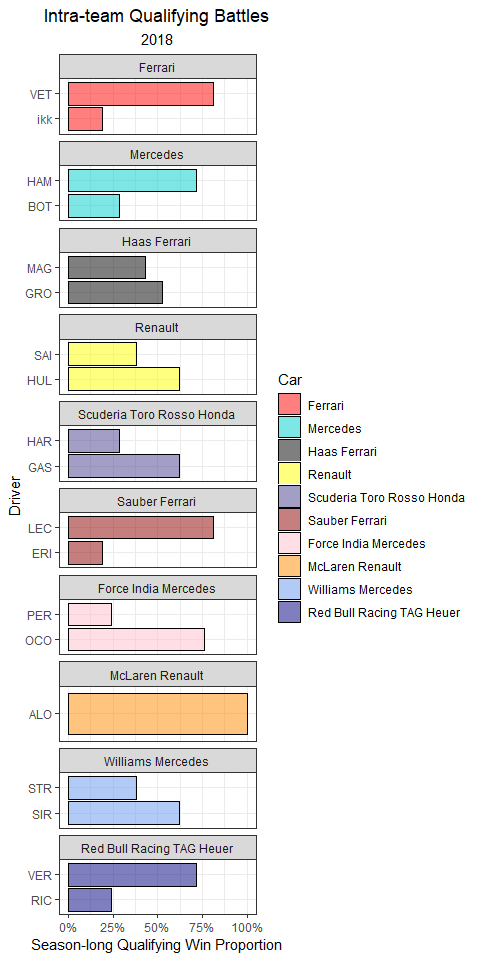
qualifying\_allyears %>%  
 mutate(Driver = ifelse(Driver == 'ikk', 'RAI', Driver)) %>%   
 filter(!is.na(Q\_secs),  
 Year == 2019) %>%  
 group\_by(Race) %>%  
 mutate(track\_mean = mean(Q\_secs, na.rm = T),  
 Time\_std\_track = Q\_secs - track\_mean) %>%   
 ungroup() %>%   
 group\_by(Driver) %>%   
 mutate(driver\_n = n(),  
 mean = mean(Time\_std\_track, na.rm = T)) %>%  
 ungroup() %>%  
 dplyr::filter(driver\_n > 15) %>%   
 ggplot(aes(Time\_std\_track, y = fct\_reorder(Driver, desc(mean)))) +  
 geom\_point(position = position\_jitter(w = 0, h = 0.1), alpha = 0.5) +  
 theme\_bw() +  
 labs(x = 'Standardized Qualifying Time (secs)',  
 y = 'Driver',  
 title = 'Standardized Qualifying Time',  
 subtitle = '2019',  
 caption = '\*Times are subtracted from the average time for each Grand Prix') +  
 geom\_vline(xintercept = 0, col = 'red') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 xlim(-5, 10)



### 5.2.6 2018

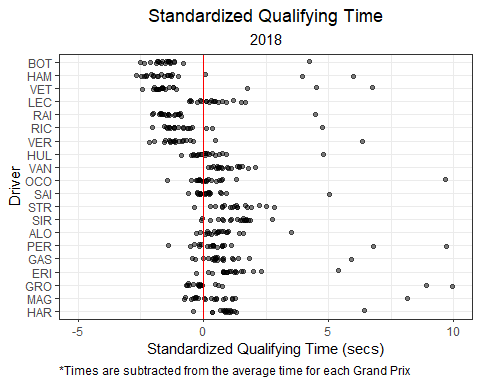
During the 2018 season, Lewis Hamilton and Sebastian Vettel battled closely for pole positions.

quali2018 <- qualifying\_scraper(2018)  
  
quali2018 %>%  
 ungroup() %>%  
 dplyr::select(Race, Position, Car, Driver) %>%  
 mutate(Position = as.integer(Position),  
 Position = ifelse(is.na(Position), 21, Position)) %>%   
 group\_by(Car, Race) %>%   
 arrange(Driver) %>%   
 mutate(driver\_num = 1:n()) %>%   
 pivot\_wider(names\_from = 'driver\_num', values\_from = c('Driver', 'Position')) %>%  
 mutate(best\_qualifier = ifelse(Position\_1 < Position\_2, Driver\_1, Driver\_2)) %>%  
 ungroup() %>%  
 group\_by(Car) %>%  
 count(best\_qualifier) %>%  
 mutate(percentage = n / sum(n)) %>%  
 filter(!is.na(best\_qualifier)) %>%  
 ggplot(aes(y = fct\_reorder(best\_qualifier, Car), x = percentage, fill = Car)) +  
 geom\_bar(stat = 'identity', position = 'dodge', alpha = 0.5, col = 'black') +  
 theme\_bw() +  
 scale\_x\_continuous(labels = scales::percent\_format()) +  
 facet\_wrap(~ factor(Car,  
 levels = c("Ferrari",  
 "Mercedes",  
 "Haas Ferrari",  
 "Renault",  
 "Scuderia Toro Rosso Honda",  
 "Sauber Ferrari",   
 "Force India Mercedes",  
 "McLaren Renault",  
 "Williams Mercedes",  
 "Red Bull Racing TAG Heuer")), ncol = 1, strip.position="top", scales = 'free\_y') +  
 labs(y = 'Driver',  
 x = 'Season-long Qualifying Win Proportion',  
 title = 'Intra-team Qualifying Battles',  
 subtitle = '2018') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 scale\_fill\_manual(values = c("red",   
 "cyan3",   
 "black",   
 "yellow",  
 "darkslateblue",   
 "darkred",   
 "pink",   
 "darkorange",   
 "cornflowerblue",  
 "navy"),  
 breaks = c("Ferrari",  
 "Mercedes",  
 "Haas Ferrari",  
 "Renault",  
 "Scuderia Toro Rosso Honda",  
 "Sauber Ferrari",   
 "Force India Mercedes",  
 "McLaren Renault",  
 "Williams Mercedes",  
 "Red Bull Racing TAG Heuer"))



Every race during the 2018 season featured either Hamilton or Vettel on the front row of the grid. Further, Hamilton and Vettel had average starting positions of 2.48 and 2.62, respectively.

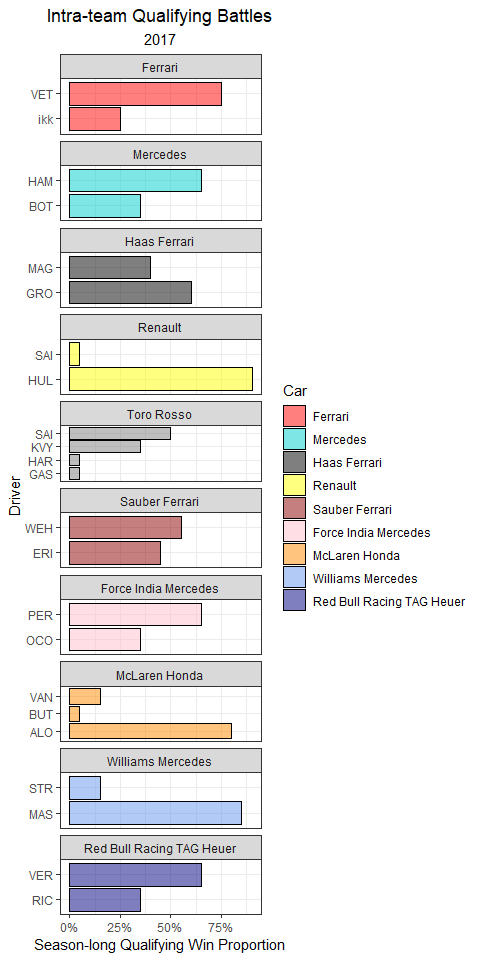
qualifying\_allyears %>%  
 mutate(Driver = ifelse(Driver == 'ikk', 'RAI', Driver)) %>%   
 filter(!is.na(Q\_secs),  
 Year == 2018) %>%  
 group\_by(Race) %>%  
 mutate(track\_mean = mean(Q\_secs, na.rm = T),  
 Time\_std\_track = Q\_secs - track\_mean) %>%   
 ungroup() %>%   
 group\_by(Driver) %>%   
 mutate(driver\_n = n(),  
 mean = mean(Time\_std\_track, na.rm = T)) %>%  
 ungroup() %>%  
 dplyr::filter(driver\_n > 15) %>%   
 ggplot(aes(Time\_std\_track, y = fct\_reorder(Driver, desc(mean)))) +  
 geom\_point(position = position\_jitter(w = 0, h = 0.1), alpha = 0.5) +  
 theme\_bw() +  
 labs(x = 'Standardized Qualifying Time (secs)',  
 y = 'Driver',  
 title = 'Standardized Qualifying Time',  
 subtitle = '2018',  
 caption = '\*Times are subtracted from the average time for each Grand Prix') +  
 geom\_vline(xintercept = 0, col = 'red') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 xlim(-5, 10)



### 5.2.7 2017

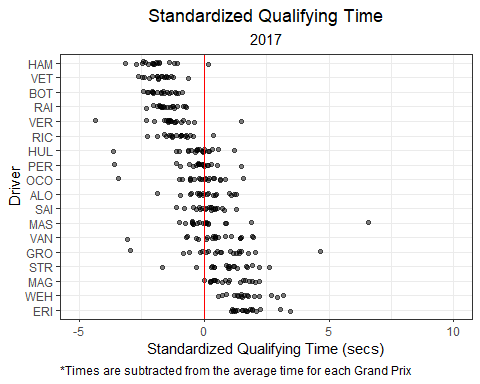
During 2017, only the two teams (Mercedes and Ferrari) qualified on pole. Hamilton achieved pole position in over half the races (11/ 20).

quali2017 <- qualifying\_scraper(2017)  
  
quali2017 %>%  
 ungroup() %>%  
 dplyr::select(Race, Position, Car, Driver) %>%  
 mutate(Position = as.integer(Position),  
 Position = ifelse(is.na(Position), 21, Position)) %>%   
 group\_by(Car, Race) %>%   
 arrange(Driver) %>%   
 mutate(driver\_num = 1:n()) %>%   
 pivot\_wider(names\_from = 'driver\_num', values\_from = c('Driver', 'Position')) %>%  
 mutate(best\_qualifier = ifelse(Position\_1 < Position\_2, Driver\_1, Driver\_2)) %>%  
 ungroup() %>%  
 group\_by(Car) %>%  
 count(best\_qualifier) %>%  
 mutate(percentage = n / sum(n)) %>%  
 filter(!is.na(best\_qualifier)) %>%  
 ggplot(aes(y = fct\_reorder(best\_qualifier, Car), x = percentage, fill = Car)) +  
 geom\_bar(stat = 'identity', position = 'dodge', alpha = 0.5, col = 'black') +  
 theme\_bw() +  
 scale\_x\_continuous(labels = scales::percent\_format()) +  
 facet\_wrap(~ factor(Car,  
 levels = c("Ferrari",  
 "Mercedes",  
 "Haas Ferrari",  
 "Renault",  
 "Toro Rosso",  
 "Sauber Ferrari",   
 "Force India Mercedes",  
 "McLaren Honda",  
 "Williams Mercedes",  
 "Red Bull Racing TAG Heuer")), ncol = 1, strip.position="top", scales = 'free\_y') +  
 labs(y = 'Driver',  
 x = 'Season-long Qualifying Win Proportion',  
 title = 'Intra-team Qualifying Battles',  
 subtitle = '2017') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 scale\_fill\_manual(values = c("red",   
 "cyan3",   
 "black",   
 "yellow",  
 "darkslateblue",   
 "darkred",   
 "pink",   
 "darkorange",   
 "cornflowerblue",  
 "navy"),  
 breaks = c("Ferrari",  
 "Mercedes",  
 "Haas Ferrari",  
 "Renault",  
 "Scuderia Toro Rosso Honda",  
 "Sauber Ferrari",   
 "Force India Mercedes",  
 "McLaren Honda",  
 "Williams Mercedes",  
 "Red Bull Racing TAG Heuer"))



Interestingly, only Bottas and Kimi Raikkonen reached Q3 in every Grand Prix of the season.

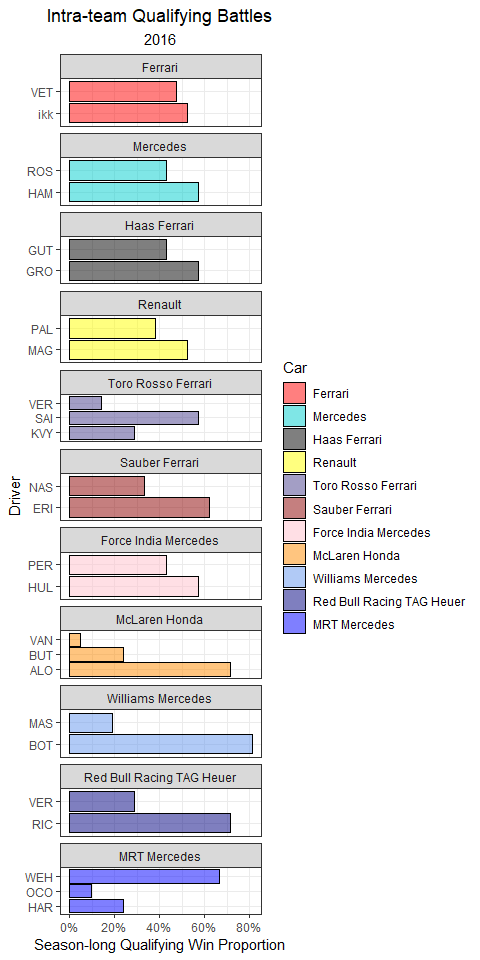
qualifying\_allyears %>%  
 mutate(Driver = ifelse(Driver == 'ikk', 'RAI', Driver)) %>%   
 filter(!is.na(Q\_secs),  
 Year == 2017) %>%  
 group\_by(Race) %>%  
 mutate(track\_mean = mean(Q\_secs, na.rm = T),  
 Time\_std\_track = Q\_secs - track\_mean) %>%   
 ungroup() %>%   
 group\_by(Driver) %>%   
 mutate(driver\_n = n(),  
 mean = mean(Time\_std\_track, na.rm = T)) %>%  
 ungroup() %>%  
 dplyr::filter(driver\_n > 15) %>%   
 ggplot(aes(Time\_std\_track, y = fct\_reorder(Driver, desc(mean)))) +  
 geom\_point(position = position\_jitter(w = 0, h = 0.1), alpha = 0.5) +  
 theme\_bw() +  
 labs(x = 'Standardized Qualifying Time (secs)',  
 y = 'Driver',  
 title = 'Standardized Qualifying Time',  
 subtitle = '2017',  
 caption = '\*Times are subtracted from the average time for each Grand Prix') +  
 geom\_vline(xintercept = 0, col = 'red') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 xlim(-5, 10)



### 5.2.8 2016

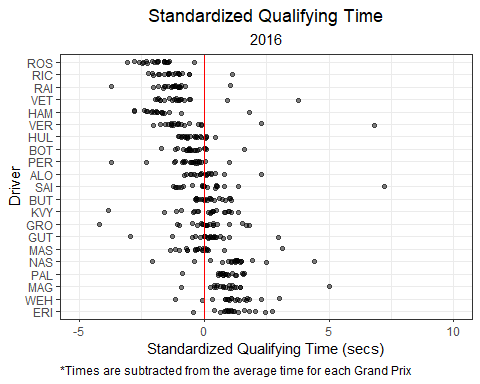
2016 was an extremely tight competition in both races and qualifying for the two Mercedes drivers. Nico Rosberg qualified on the front row in every race of the season, while maintaining an average starting position of 1.62. However, Hamilton still managed to achieve pole more often: 12 poles vs 8 for Rosberg.

quali2016 <- qualifying\_scraper(2016)  
  
quali2016 %>%  
 mutate(Car = str\_replace(Car, "-", " ")) %>%   
 ungroup() %>%  
 dplyr::select(Race, Position, Car, Driver) %>%  
 mutate(Position = as.integer(Position),  
 Position = ifelse(is.na(Position), 21, Position)) %>%   
 group\_by(Car, Race) %>%   
 arrange(Driver) %>%   
 mutate(driver\_num = 1:n()) %>%   
 pivot\_wider(names\_from = 'driver\_num', values\_from = c('Driver', 'Position')) %>%  
 mutate(best\_qualifier = ifelse(Position\_1 < Position\_2, Driver\_1, Driver\_2)) %>%  
 ungroup() %>%  
 group\_by(Car) %>%  
 count(best\_qualifier) %>%  
 mutate(percentage = n / sum(n)) %>%  
 filter(!is.na(best\_qualifier)) %>%  
 ggplot(aes(y = fct\_reorder(best\_qualifier, Car), x = percentage, fill = Car)) +  
 geom\_bar(stat = 'identity', position = 'dodge', alpha = 0.5, col = 'black') +  
 theme\_bw() +  
 scale\_x\_continuous(labels = scales::percent\_format()) +  
 facet\_wrap(~ factor(Car,   
 levels = c("Ferrari",  
 "Mercedes",  
 "Haas Ferrari",  
 "Renault",  
 "Toro Rosso Ferrari",  
 "Sauber Ferrari",   
 "Force India Mercedes",  
 "McLaren Honda",  
 "Williams Mercedes",  
 "Red Bull Racing TAG Heuer",  
 "MRT Mercedes")),  
 ncol = 1, strip.position="top", scales = 'free\_y') +  
 labs(y = 'Driver',  
 x = 'Season-long Qualifying Win Proportion',  
 title = 'Intra-team Qualifying Battles',  
 subtitle = '2016') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 scale\_fill\_manual(values = c("red",   
 "cyan3",   
 "black",   
 "yellow",  
 "darkslateblue",   
 "darkred",   
 "pink",   
 "darkorange",   
 "cornflowerblue",  
 "navy",  
 "blue"),  
 breaks = c("Ferrari",  
 "Mercedes",  
 "Haas Ferrari",  
 "Renault",  
 "Toro Rosso Ferrari",  
 "Sauber Ferrari",   
 "Force India Mercedes",  
 "McLaren Honda",  
 "Williams Mercedes",  
 "Red Bull Racing TAG Heuer",  
 "MRT Mercedes"))



The only non-Mercedes pole of 2016 was Daniel Ricciardo in Monaco.

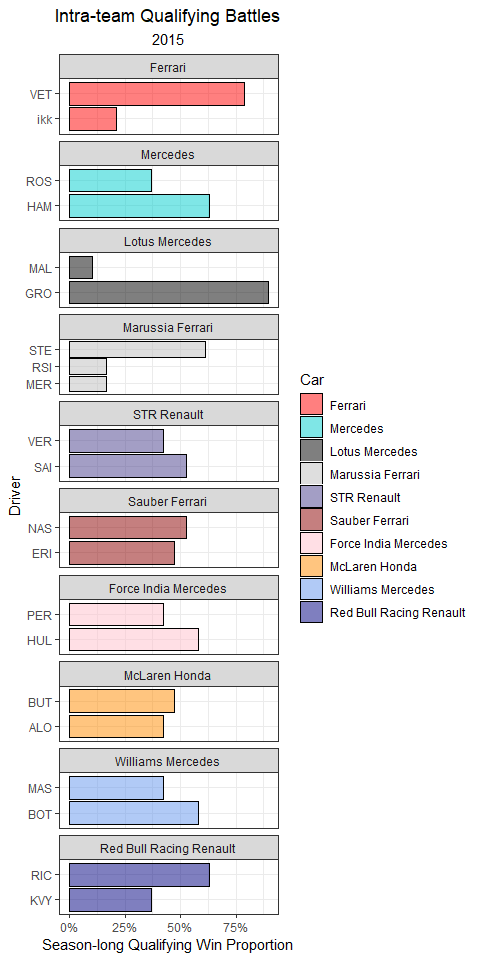
qualifying\_allyears %>%  
 mutate(Driver = ifelse(Driver == 'ikk', 'RAI', Driver)) %>%   
 filter(!is.na(Q\_secs),  
 Year == 2016) %>%  
 group\_by(Race) %>%  
 mutate(track\_mean = mean(Q\_secs, na.rm = T),  
 Time\_std\_track = Q\_secs - track\_mean) %>%   
 ungroup() %>%   
 group\_by(Driver) %>%   
 mutate(driver\_n = n(),  
 mean = mean(Time\_std\_track, na.rm = T)) %>%  
 ungroup() %>%  
 dplyr::filter(driver\_n > 15) %>%   
 ggplot(aes(Time\_std\_track, y = fct\_reorder(Driver, desc(mean)))) +  
 geom\_point(position = position\_jitter(w = 0, h = 0.1), alpha = 0.5) +  
 theme\_bw() +  
 labs(x = 'Standardized Qualifying Time (secs)',  
 y = 'Driver',  
 title = 'Standardized Qualifying Time',  
 subtitle = '2016',  
 caption = '\*Times are subtracted from the average time for each Grand Prix') +  
 geom\_vline(xintercept = 0, col = 'red') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 xlim(-5, 10)



### 5.2.9 2015

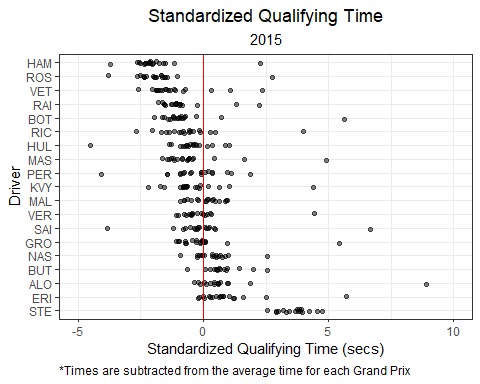
Lewis Hamilton took 11 out of the first 12 poles of the 2015 season, while Rosberg won the final 6 poles.

quali2015 <- qualifying\_scraper(2015)  
  
quali2015 %>%  
 mutate(Car = str\_replace(Car, "-", " ")) %>%   
 ungroup() %>%  
 dplyr::select(Race, Position, Car, Driver) %>%  
 mutate(Position = as.integer(Position),  
 Position = ifelse(is.na(Position), 21, Position)) %>%   
 group\_by(Car, Race) %>%   
 arrange(Driver) %>%   
 mutate(driver\_num = 1:n()) %>%   
 pivot\_wider(names\_from = 'driver\_num', values\_from = c('Driver', 'Position')) %>%  
 mutate(best\_qualifier = ifelse(Position\_1 < Position\_2, Driver\_1, Driver\_2)) %>%  
 ungroup() %>%  
 group\_by(Car) %>%  
 count(best\_qualifier) %>%  
 mutate(percentage = n / sum(n)) %>%  
 filter(!is.na(best\_qualifier)) %>%  
 ggplot(aes(y = fct\_reorder(best\_qualifier, Car), x = percentage, fill = Car)) +  
 geom\_bar(stat = 'identity', position = 'dodge', alpha = 0.5, col = 'black') +  
 theme\_bw() +  
 scale\_x\_continuous(labels = scales::percent\_format()) +  
 facet\_wrap(~ factor(Car,   
 levels =c("Ferrari",  
 "Mercedes",  
 "Lotus Mercedes",  
 "Marussia Ferrari",  
 "STR Renault",  
 "Sauber Ferrari",   
 "Force India Mercedes",  
 "McLaren Honda",  
 "Williams Mercedes",  
 "Red Bull Racing Renault")),  
 ncol = 1, strip.position="top", scales = 'free\_y') +  
 labs(y = 'Driver',  
 x = 'Season-long Qualifying Win Proportion',  
 title = 'Intra-team Qualifying Battles',  
 subtitle = '2015') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 scale\_fill\_manual(values = c("red",   
 "cyan3",   
 "black",   
 "grey",  
 "darkslateblue",   
 "darkred",   
 "pink",   
 "darkorange",   
 "cornflowerblue",  
 "navy",  
 "blue"),  
 breaks = c("Ferrari",  
 "Mercedes",  
 "Lotus Mercedes",  
 "Marussia Ferrari",  
 "STR Renault",  
 "Sauber Ferrari",   
 "Force India Mercedes",  
 "McLaren Honda",  
 "Williams Mercedes",  
 "Red Bull Racing Renault"))



Mercedes was absolutely dominant in qualifying and missed out on only 1 pole to Sebastian Vettel in Singapore.

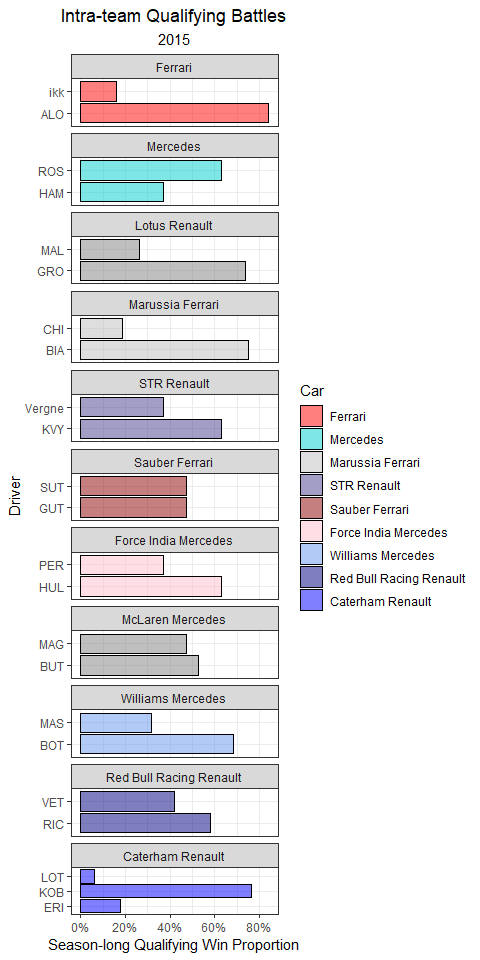
qualifying\_allyears %>%  
 mutate(Driver = ifelse(Driver == 'ikk', 'RAI', Driver)) %>%   
 filter(!is.na(Q\_secs),  
 Year == 2015) %>%  
 group\_by(Race) %>%  
 mutate(track\_mean = mean(Q\_secs, na.rm = T),  
 Time\_std\_track = Q\_secs - track\_mean) %>%   
 ungroup() %>%   
 group\_by(Driver) %>%   
 mutate(driver\_n = n(),  
 mean = mean(Time\_std\_track, na.rm = T)) %>%  
 ungroup() %>%  
 dplyr::filter(driver\_n > 15) %>%   
 ggplot(aes(Time\_std\_track, y = fct\_reorder(Driver, desc(mean)))) +  
 geom\_point(position = position\_jitter(w = 0, h = 0.1), alpha = 0.5) +  
 theme\_bw() +  
 labs(x = 'Standardized Qualifying Time (secs)',  
 y = 'Driver',  
 title = 'Standardized Qualifying Time',  
 subtitle = '2015',  
 caption = '\*Times are subtracted from the average time for each Grand Prix') +  
 geom\_vline(xintercept = 0, col = 'red') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 xlim(-5, 10)



### 5.2.10 2014

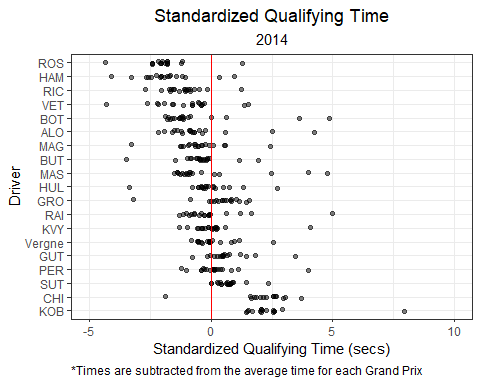
The very consistent Nico Rosberg averaged a 1.68 starting position, won 11 poles, and out-qualified Lewis Hamilton 12:7 in the 2014 season.

quali2014 <- qualifying\_scraper(2014)  
  
quali2014 %>%  
 mutate(Car = str\_replace(Car, "-", " ")) %>%   
 ungroup() %>%  
 dplyr::select(Race, Position, Car, Driver) %>%  
 mutate(Position = as.integer(Position),  
 Position = ifelse(is.na(Position), 21, Position)) %>%   
 group\_by(Car, Race) %>%   
 arrange(Driver) %>%   
 mutate(driver\_num = 1:n()) %>%   
 pivot\_wider(names\_from = 'driver\_num', values\_from = c('Driver', 'Position')) %>%  
 mutate(best\_qualifier = ifelse(Position\_1 < Position\_2, Driver\_1, Driver\_2)) %>%  
 ungroup() %>%  
 group\_by(Car) %>%  
 count(best\_qualifier) %>%  
 mutate(percentage = n / sum(n)) %>%  
 filter(!is.na(best\_qualifier)) %>%  
 ggplot(aes(y = fct\_reorder(best\_qualifier, Car), x = percentage, fill = Car)) +  
 geom\_bar(stat = 'identity', position = 'dodge', alpha = 0.5, col = 'black') +  
 theme\_bw() +  
 scale\_x\_continuous(labels = scales::percent\_format()) +  
 facet\_wrap(~ factor(Car,   
 levels =c("Ferrari",  
 "Mercedes",  
 "Lotus Renault",  
 "Marussia Ferrari",  
 "STR Renault",  
 "Sauber Ferrari",   
 "Force India Mercedes",  
 "McLaren Mercedes",  
 "Williams Mercedes",  
 "Red Bull Racing Renault",  
 "Caterham Renault")),  
 ncol = 1, strip.position="top", scales = 'free\_y') +  
 labs(y = 'Driver',  
 x = 'Season-long Qualifying Win Proportion',  
 title = 'Intra-team Qualifying Battles',  
 subtitle = '2015') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 scale\_fill\_manual(values = c("red",   
 "cyan3",   
 "black",   
 "grey",  
 "darkslateblue",   
 "darkred",   
 "pink",   
 "darkorange",   
 "cornflowerblue",  
 "navy",  
 "blue",  
 "seagreen"),  
 breaks = c("Ferrari",  
 "Mercedes",  
 "Lotus Mercedes",  
 "Marussia Ferrari",  
 "STR Renault",  
 "Sauber Ferrari",   
 "Force India Mercedes",  
 "McLaren Honda",  
 "Williams Mercedes",  
 "Red Bull Racing Renault",  
 "Caterham Renault"))



Mercedes power units claimed every pole of this inaugural season for the turbo-hybrid era.

qualifying\_allyears %>%  
 mutate(Driver = ifelse(Driver == 'ikk', 'RAI', Driver)) %>%   
 filter(!is.na(Q\_secs),  
 Year == 2014) %>%  
 group\_by(Race) %>%  
 mutate(track\_mean = mean(Q\_secs, na.rm = T),  
 Time\_std\_track = Q\_secs - track\_mean) %>%   
 ungroup() %>%   
 group\_by(Driver) %>%   
 mutate(driver\_n = n(),  
 mean = mean(Time\_std\_track, na.rm = T)) %>%  
 ungroup() %>%  
 dplyr::filter(driver\_n > 15) %>%   
 ggplot(aes(Time\_std\_track, y = fct\_reorder(Driver, desc(mean)))) +  
 geom\_point(position = position\_jitter(w = 0, h = 0.1), alpha = 0.5) +  
 theme\_bw() +  
 labs(x = 'Standardized Qualifying Time (secs)',  
 y = 'Driver',  
 title = 'Standardized Qualifying Time',  
 subtitle = '2014',  
 caption = '\*Times are subtracted from the average time for each Grand Prix') +  
 geom\_vline(xintercept = 0, col = 'red') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5),  
 plot.caption = element\_text(hjust = 0)) +  
 xlim(-5, 10)



### 5.2.11 Next Chapter

In the next chapter, we will explore the relationship between qualifying position and race placing.

# 6 Exploratory Data Analysis and Modeling

What determines success in a Grand Prix? Obviously, a multitude of factors do, but which ones specifically, and how much? That’s what I am aiming to answer with this book.

Let’s begin with a mostly uncontroversial statement: All other factors held constant, it is best to start the race in first place! How much does starting position influence a driver’s finishing place? We can explore the data and build a model to answer this.

## 6.1 Plot overall trend: Starting Position vs Finish Position

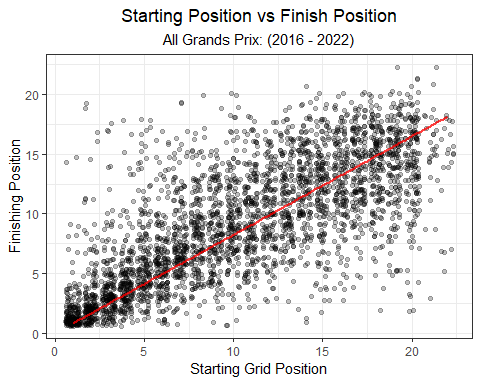
Our dataset contains starting grid position and finishing place for each race from 2016 to 2022. Below, I’ll plot starting grid position vs finishing position for each driver over this seven year period. I’ll include a best-fit line to highlight the overall relationship.

# Pull Race Data  
races2023 <- race\_result\_scraper(2023)  
races2022 <- race\_result\_scraper(2022)  
races2021 <- race\_result\_scraper(2021)  
races2020 <- race\_result\_scraper(2020)  
races2019 <- race\_result\_scraper(2019)  
races2018 <- race\_result\_scraper(2018)  
races2017 <- race\_result\_scraper(2017)  
races2016 <- race\_result\_scraper(2016)  
races2015 <- race\_result\_scraper(2015)  
races2014 <- race\_result\_scraper(2014)  
  
# Combine all race data  
races\_allyears <- rbind(races2023,  
 races2022,  
 races2021,   
 races2020,  
 races2019,  
 races2018,  
 races2017,  
 races2016,  
 races2015,  
 races2014)  
  
# Pull Starting Grid Data  
grid2023 <- starting\_grid\_scraper(2023)  
grid2022 <- starting\_grid\_scraper(2022)  
grid2021 <- starting\_grid\_scraper(2021)  
grid2020 <- starting\_grid\_scraper(2020)  
grid2019 <- starting\_grid\_scraper(2019)  
grid2018 <- starting\_grid\_scraper(2018)  
grid2017 <- starting\_grid\_scraper(2017)  
grid2016 <- starting\_grid\_scraper(2016)  
grid2015 <- starting\_grid\_scraper(2015)  
grid2014 <- starting\_grid\_scraper(2014)  
  
# Combine all starting grid data  
grid\_allyears <- rbind(grid2023,  
 grid2022,  
 grid2021,   
 grid2020,  
 grid2019,  
 grid2018,  
 grid2017,  
 grid2016,  
 grid2015,  
 grid2014)

Here, I’ll merge the grid positions with the race results for all Grands Prix from 2014 to 2023.

## Merge practice and qualifying data  
grids\_and\_races <- races\_allyears %>%  
 left\_join(grid\_allyears %>% dplyr::select(Driver, Race, Circuit, Year, Position, Time\_secs),  
 by = c('Driver', 'Race', 'Circuit', 'Year'),  
 suffix = c("\_race", "\_grid")) %>%  
 filter(!Position\_race %in% c("NC", "DQ")) %>%   
 mutate(Position\_race = as.numeric(Position\_race),   
 Position\_grid = as.numeric(Position\_grid))

grids\_and\_races %>%  
 ggplot(aes(Position\_grid, Position\_race)) +  
 geom\_point(position = position\_jitter(), alpha = 0.25) +  
 geom\_line(stat = 'smooth', se = F, method = 'lm', col = 'red',  
 formula = y ~ 0 + x, size = 1, alpha = 0.75) +  
 theme\_bw() +  
 labs(x = 'Starting Grid Position',  
 y = 'Finishing Position',  
 title = 'Starting Position vs Finish Position',  
 subtitle = 'All Grands Prix: (2016 - 2022)') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5))



### 6.1.1 Overall model

There is certainly a pattern in the data; starting and finishing positions are positively correlated. How much does starting position influence the finishing position of a driver? We need to be as explicit as possible in this research question, because that will guide our model choice and ultimate interpretation of the results. I’ll give it my best shot to clearly state my research question below:

*What effect does starting grid position have on the probability of a finishing position in a Grand Prix?*

What model allows us to answer this question? Because the response variable is ordered (1st place, 2nd place, etc.), a proportional odds model, or ordinal regression model, is an obvious choice.

**What is an Ordinal Regression model?**

An Ordinal Regression model uses a response variable that is an ordered factor. For example, “Low/Medium/High” or “Bad/Good/Great” are both ordered factors with three levels. Finishing Place, “P1, P2, P3, etc.”, is also an ordered factor. If we are interested in understanding the probability of a driver finishing in a given position, an Ordinal Regression is an ideal model choice.

Below, I’ll build an ordinal regression modeol, or a *proportional odds model*, to estimate the effect of starting grid position on the odds of a particular finishing position. This model will utilize all races across the seven year period (2016 to 2022).

library(VGAM)  
  
overall\_model <- vglm(Position\_race ~ Position\_grid,   
 family = cumulative(parallel = T, reverse = F),  
 data = grids\_and\_races)  
  
exp(coef(overall\_model))

## (Intercept):1 (Intercept):2 (Intercept):3 (Intercept):4 (Intercept):5 (Intercept):6   
## 5.251639e-01 1.390002e+00 2.779530e+00 4.946141e+00 8.283053e+00 1.335696e+01   
## (Intercept):7 (Intercept):8 (Intercept):9 (Intercept):10 (Intercept):11 (Intercept):12   
## 2.103747e+01 3.258116e+01 5.030764e+01 7.746446e+01 1.189627e+02 1.837928e+02   
## (Intercept):13 (Intercept):14 (Intercept):15 (Intercept):16 (Intercept):17 (Intercept):18   
## 2.889453e+02 4.695064e+02 8.025183e+02 1.448913e+03 3.072870e+03 7.502012e+03   
## (Intercept):19 (Intercept):20 (Intercept):21 Position\_grid   
## 2.188651e+04 7.977022e+04 2.817311e+05 7.056819e-01

There is a coefficient estimate for each predictor variable (i.e starting grid position). We would interpret these coefficients, after exponentiation, as:

*If a driver starts from the front of the grid (i.e P1), there is a 22% decrease in the odds of finishing at a given position or better.*

Or…

*For drivers who start a race from P3, the odds of finishing P1 vs the top 2 positions are 0.21 lower.*

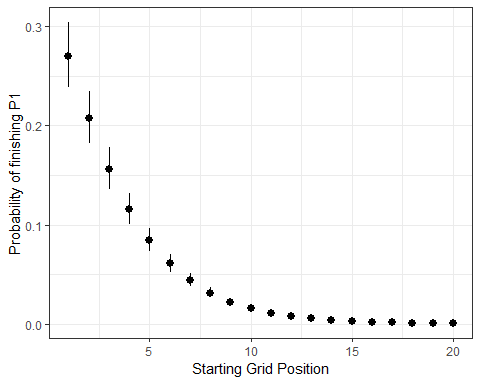
The interpretation of the coefficients are not super intuitive. They make sense, but it takes some effort to sort out.

Another way to articulate this model’s results is by making predictions with the model. Below, I’ll simulate a starting grid, and make predictions for finishing place. Predicted finishing position will include a 95% confidence interval.

# Create a placeholder dataframe with all possible starting positions  
starting\_grid = data.frame(Position\_grid = 1:20)  
  
# Create a placeholder dataframe with names for predictions and standard errors.   
placing\_df <- data.frame(prob = rep('prob', 21)) %>%  
 mutate(y = 1:n()) %>%  
 unite(prob, prob, y, remove = F) %>%  
 mutate(se = rep('se', 21)) %>%  
 unite(se, se, y, remove = F)  
  
# Make Predictions for each Finishing position (1:20), using each Starting Position (1:20)  
predictions <- as.data.frame(predictvglm(overall\_model, newdata = starting\_grid, se.fit = T)$fitted.values)  
  
# rename the columns  
names(predictions) <- placing\_df$prob  
  
# Extract the standard error for each prediction  
standard\_errors <- as.data.frame(predictvglm(overall\_model, newdata = starting\_grid, se.fit = T)$se.fit)  
  
# rename the standard error   
names(standard\_errors) <- placing\_df$se  
  
## As an example, I'll select the predictions and standard error for: Probability(P1 finish | P1 starting grid)  
# Bind the predictions and standard errors together, and convert the odds to probabilities.   
starting\_grid\_coef\_finish1 <- starting\_grid %>%  
 bind\_cols(pr\_1 = predictions$prob\_1,  
 se\_1 = standard\_errors$se\_1) %>%  
 mutate(pr = exp(pr\_1),  
 lower = exp(pr\_1 - se\_1 \* 1.96),  
 upper = exp(pr\_1 + se\_1 \* 1.96)) %>%  
 mutate( pr = pr / (1 + pr),  
 lower = lower / (1 + lower),  
 upper = upper / (1 + upper))

Now that the predictions are made, I’ll plot the Probability of winning a race given each potential starting position, P(P1 finish | a starting grid):

starting\_grid\_coef\_finish1 %>%  
 ggplot(aes(Position\_grid, y = pr, ymin = lower, ymax = upper)) +  
 geom\_pointrange() +  
 theme\_bw() +  
 labs( y = 'Probability of finishing P1',  
 x = 'Starting Grid Position')

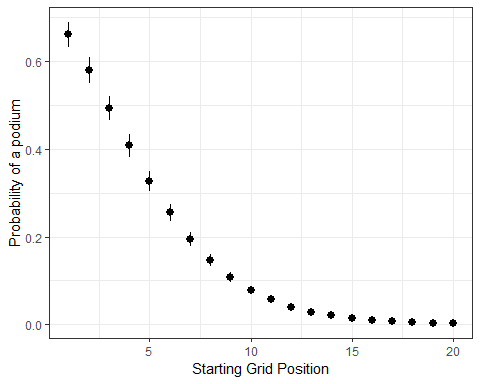


If starting from P1, a driver has a 27% chance of winning the race, on average. If starting from P2, but still on the front row, a driver’s chance of winning drops to 21%.

### 6.1.2 What is the probability of finishing on the podium?

We can use the same approach to estimate the probability of a podium given a starting grid position.

## As an example, I'll select the predictions and standard error for: Probability(Podium | a starting grid)  
# Bind the predictions and standard errors together, and convert the odds to probabilities.   
starting\_grid\_coef\_podium <- starting\_grid %>%  
 bind\_cols(pr\_1 = predictions$prob\_3,  
 se\_1 = standard\_errors$se\_3) %>%  
 mutate(pr = exp(pr\_1),  
 lower = exp(pr\_1 - se\_1 \* 1.96),  
 upper = exp(pr\_1 + se\_1 \* 1.96)) %>%  
 mutate( pr = pr / (1 + pr),  
 lower = lower / (1 + lower),  
 upper = upper / (1 + upper))  
  
starting\_grid\_coef\_podium %>%  
 ggplot(aes(Position\_grid, y = pr, ymin = lower, ymax = upper)) +  
 geom\_pointrange() +  
 theme\_bw() +  
 labs( y = 'Probability of a podium',  
 x = 'Starting Grid Position')



A driver starting from P1 has a 66% chance of finishing on the podium. I like those odds! A driver starting from P5 still has a 1 in 3 chance of finishing on the podium.

#### 6.1.2.1 Prediction Grid

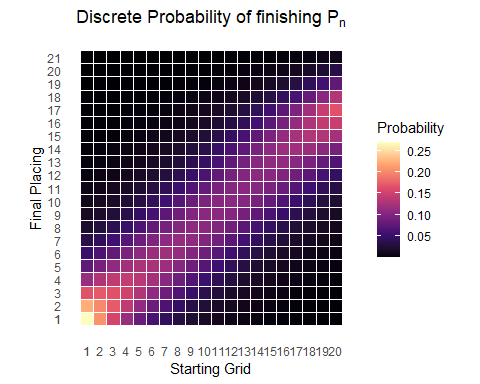
What if we wanted to know the probability of each Finish Position, given each starting position? I’ll plot in a heat map below. Each tile represents the Probability(a finishing position | a starting grid position).

First, I’ll need to re-shape the dataframe to a long format indexed by starting and finish position. I will also add a new column that contains the discrete probability of a given finish. Because the proportional odds model output is converted to a probabilistic statement like Probability(finish P3 or better | a starting position), the discrete probability column will give a statement like Probability(finish P3 | a starting position).

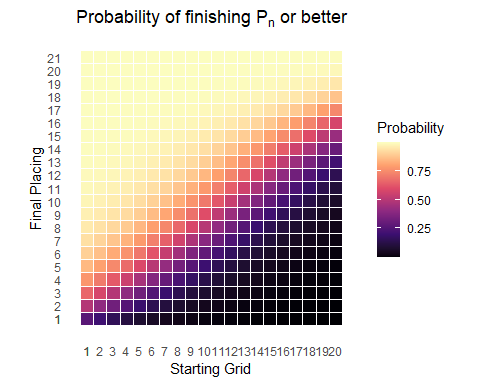
preds\_se\_full <- predictions %>%  
 mutate(starting = 1:n()) %>%  
 pivot\_longer(- starting, names\_to = 'placing', values\_to = 'prob') %>%   
 bind\_cols(standard\_errors %>% mutate(starting = 1:n()) %>%  
 pivot\_longer(- starting, names\_to = 'placing', values\_to = 'se') %>%  
 dplyr::select(se)) %>%  
 mutate(placing = as.numeric(str\_remove(placing, 'prob\_'))) %>%  
 mutate(prob = exp(prob)) %>%  
 mutate(prob = prob / (1 + prob)) %>%  
 group\_by(starting) %>%   
 mutate(discrete\_prob = ifelse(is.na(prob - lag(prob)), prob, (prob - lag(prob)))) %>%  
 ungroup()

Now that the data is re-shaped, I’ll plot the discrete and cumulative probability heatmaps.

# Plot Discrete Probability heatmap  
preds\_se\_full %>%  
 ggplot(aes(starting, placing, fill = discrete\_prob)) +  
 geom\_tile(col = 'white', size = 0.1) +  
 theme\_tufte(base\_family="Helvetica") +  
 scale\_fill\_viridis\_c(option = 'magma') +  
 coord\_equal() +  
 theme(axis.ticks=element\_blank()) +  
 labs(x = 'Starting Grid',  
 y = 'Final Placing',  
 fill = 'Probability',  
 title = expression(Discrete~Probability~of~finishing~P[n])) +  
 scale\_x\_continuous(breaks = c(1:20,1)) +  
 scale\_y\_continuous(breaks = c(1:21,1)) +  
 theme(plot.title = element\_text(hjust = 0.5))

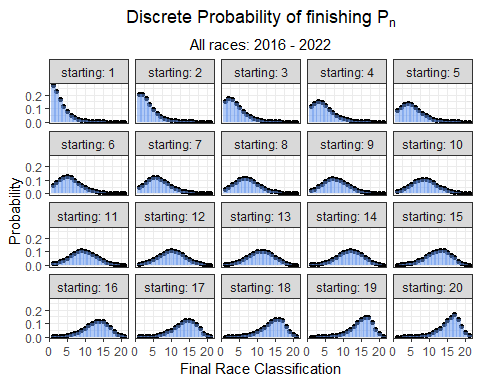


# Plot Cumulative Probability heatmap   
preds\_se\_full %>%  
 ggplot(aes(starting, placing, fill = prob)) +  
 geom\_tile(col = 'white', size = 0.1) +  
 theme\_tufte(base\_family="Helvetica") +  
 scale\_fill\_viridis\_c(option = 'magma') +  
 coord\_equal() +  
 theme(axis.ticks=element\_blank()) +  
 labs(x = 'Starting Grid',  
 y = 'Final Placing',  
 fill = 'Probability',  
 title = expression(Probability~of~finishing~P[n]~or~better)) +  
 scale\_x\_continuous(breaks = c(1:20,1)) +  
 scale\_y\_continuous(breaks = c(1:21,1)) +  
 theme(plot.title = element\_text(hjust = 0.5))



The cumulative probability plot is helpful, but the discrete probability plot is a bit tough to follow. Let’s try plotting this information a different way.

preds\_se\_full %>%  
 mutate(starting\_pos = as.factor(paste0('P',starting)),  
 starting\_pos = fct\_reorder(starting\_pos, starting)) %>%  
 ggplot(aes(placing, discrete\_prob)) +  
 geom\_point() +  
 geom\_segment(aes(x = placing, xend = placing, y = 0, yend = discrete\_prob),  
 size = 2, alpha = 0.5, col = 'cornflowerblue') +  
 theme\_bw() +  
 facet\_wrap(~ starting, labeller = label\_both) +  
 labs(x = 'Final Race Classification',  
 y = 'Probability',  
 title = expression(Discrete~Probability~of~finishing~P[n]),  
 subtitle = 'All races: 2016 - 2022') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5))



The plot above shows the probability of finishing at any position given where the driver starts the race. It is important to keep in mind what this model is built on. These predictions are generated solely from starting position. And, we know there is much more involved in where you finish. Otherwise, why even run the race?! Let’s try expanding this model to include additional variables.

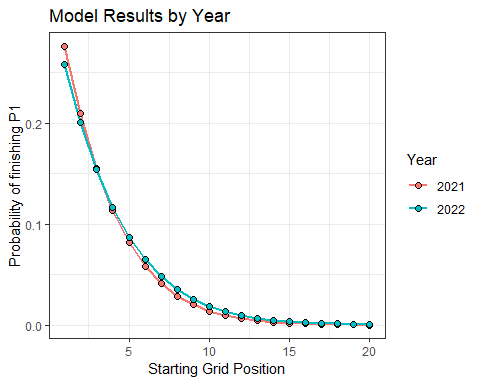
### 6.1.3 Yearly Differences

Does this relationship between starting and finishing position vary over time? 2022 ushered in a completely different set of regulations that enabled cars to follow more closely and overtake more easily. Is this represented in the data? We will now look into yearly differences in starting grid position coefficient!

model16 <- vglm(Position\_race ~ Position\_grid,   
 family = cumulative(parallel = T, reverse = F),  
 data = grids\_and\_races %>%  
 filter(Year == 2016))  
  
model17 <- vglm(Position\_race ~ Position\_grid,   
 family = cumulative(parallel = T, reverse = F),  
 data = grids\_and\_races %>%  
 filter(Year == 2017))  
  
model18 <- vglm(Position\_race ~ Position\_grid,   
 family = cumulative(parallel = T, reverse = F),  
 data = grids\_and\_races %>%  
 filter(Year == 2018))  
  
model19 <- vglm(Position\_race ~ Position\_grid,   
 family = cumulative(parallel = T, reverse = F),  
 data = grids\_and\_races %>%  
 filter(Year == 2019))  
  
model20 <- vglm(Position\_race ~ Position\_grid,   
 family = cumulative(parallel = T, reverse = F),  
 data = grids\_and\_races %>%  
 filter(Year == 2020))  
  
model21 <- vglm(Position\_race ~ Position\_grid,   
 family = cumulative(parallel = T, reverse = F),  
 data = grids\_and\_races %>%  
 filter(Year == 2021))  
  
model22 <- vglm(Position\_race ~ Position\_grid,   
 family = cumulative(parallel = T, reverse = F),  
 data = grids\_and\_races %>%  
 filter(Year == 2022))  
  
model23 <- vglm(Position\_race ~ Position\_grid,   
 family = cumulative(parallel = T, reverse = F),  
 data = grids\_and\_races %>%  
 filter(Year == 2023))

## 2016  
preds\_2016 <- data.frame(predictvglm(model16, newdata = starting\_grid, se.fit = T)$fitted.values)  
  
# rename the columns  
names(preds\_2016) <- placing\_df$prob  
  
preds\_2016 <- preds\_2016 %>%   
 mutate(Year = 2016) %>%  
 bind\_cols(starting\_grid)  
  
# Extract the standard error for each prediction  
standard\_errors\_2016 <- as.data.frame(predictvglm(model16, newdata = starting\_grid, se.fit = T)$se.fit)  
  
# rename the standard error   
names(standard\_errors\_2016) <- placing\_df$se  
  
## 2017  
preds\_2017 <- data.frame(predictvglm(model17, newdata = starting\_grid, se.fit = T)$fitted.values)  
  
# rename the columns  
names(preds\_2017) <- placing\_df$prob[1:17]  
  
preds\_2017 <- preds\_2017 %>%   
 mutate(Year = 2017) %>%  
 bind\_cols(starting\_grid)  
  
# Extract the standard error for each prediction  
standard\_errors\_2017 <- as.data.frame(predictvglm(model17, newdata = starting\_grid, se.fit = T)$se.fit) %>%   
 mutate(Year = 2017) %>%  
 bind\_cols(starting\_grid)  
  
# rename the standard error   
names(standard\_errors\_2017) <- placing\_df$se[1:18]  
  
## 2018  
preds\_2018 <- data.frame(predictvglm(model18, newdata = starting\_grid, se.fit = T)$fitted.values)  
  
# rename the columns  
names(preds\_2018) <- placing\_df$prob[1:19]  
  
preds\_2018 <- preds\_2018 %>%   
 mutate(Year = 2018) %>%  
 bind\_cols(starting\_grid)  
  
# Extract the standard error for each prediction  
standard\_errors\_2018 <- as.data.frame(predictvglm(model18, newdata = starting\_grid, se.fit = T)$se.fit) %>%   
 mutate(Year = 2018) %>%  
 bind\_cols(starting\_grid)  
  
# rename the standard error   
names(standard\_errors\_2018) <- placing\_df$se[1:20]  
  
## 2019  
preds\_2019 <- data.frame(predictvglm(model19, newdata = starting\_grid, se.fit = T)$fitted.values)  
  
# rename the columns  
names(preds\_2019) <- placing\_df$prob[1:19]  
  
preds\_2019 <- preds\_2019 %>%   
 mutate(Year = 2019) %>%  
 bind\_cols(starting\_grid)  
  
# Extract the standard error for each prediction  
standard\_errors\_2019 <- as.data.frame(predictvglm(model19, newdata = starting\_grid, se.fit = T)$se.fit) %>%   
 mutate(Year = 2019) %>%  
 bind\_cols(starting\_grid)  
  
# rename the standard error   
names(standard\_errors\_2019) <- placing\_df$se[1:20]  
  
## 2020  
preds\_2020 <- data.frame(predictvglm(model20, newdata = starting\_grid, se.fit = T)$fitted.values)  
  
# rename the columns  
names(preds\_2020) <- placing\_df$prob[1:18]  
  
preds\_2020 <- preds\_2020 %>%   
 mutate(Year = 2020) %>%  
 bind\_cols(starting\_grid)  
  
# Extract the standard error for each prediction  
standard\_errors\_2020 <- as.data.frame(predictvglm(model20, newdata = starting\_grid, se.fit = T)$se.fit) %>%   
 mutate(Year = 2020) %>%  
 bind\_cols(starting\_grid)  
  
# rename the standard error   
names(standard\_errors\_2020) <- placing\_df$se[1:19]  
  
## 2021  
preds\_2021 <- data.frame(predictvglm(model21, newdata = starting\_grid, se.fit = T)$fitted.values)  
  
# rename the columns  
names(preds\_2021) <- placing\_df$prob[1:19]  
  
preds\_2021 <- preds\_2021 %>%   
 mutate(Year = 2021) %>%  
 bind\_cols(starting\_grid)  
  
# Extract the standard error for each prediction  
standard\_errors\_2021 <- as.data.frame(predictvglm(model21, newdata = starting\_grid, se.fit = T)$se.fit) %>%   
 mutate(Year = 2021) %>%  
 bind\_cols(starting\_grid)  
  
# rename the standard error   
names(standard\_errors\_2021) <- placing\_df$se[1:20]  
  
## 2022  
preds\_2022 <- data.frame(predictvglm(model22, newdata = starting\_grid, se.fit = T)$fitted.values)  
  
# rename the columns  
names(preds\_2022) <- placing\_df$prob[1:19]  
  
preds\_2022 <- preds\_2022 %>%   
 mutate(Year = 2022) %>%  
 bind\_cols(starting\_grid)  
  
# Extract the standard error for each prediction  
standard\_errors\_2022 <- as.data.frame(predictvglm(model22, newdata = starting\_grid, se.fit = T)$se.fit) %>%   
 mutate(Year = 2022) %>%  
 bind\_cols(starting\_grid)  
  
# rename the standard error   
names(standard\_errors\_2022) <- placing\_df$se[1:20]  
  
## 2023  
preds\_2023 <- data.frame(predictvglm(model23, newdata = starting\_grid, se.fit = T)$fitted.values)  
  
# rename the columns  
names(preds\_2023) <- placing\_df$prob[1:19]  
  
preds\_2023 <- preds\_2023 %>%   
 mutate(Year = 2023) %>%  
 bind\_cols(starting\_grid)  
  
# Extract the standard error for each prediction  
standard\_errors\_2023 <- as.data.frame(predictvglm(model23, newdata = starting\_grid, se.fit = T)$se.fit) %>%   
 mutate(Year = 2023) %>%  
 bind\_cols(starting\_grid)  
  
# rename the standard error   
names(standard\_errors\_2023) <- placing\_df$se[1:20]

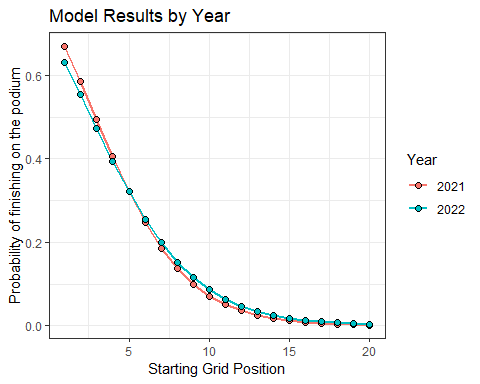
rbind(preds\_2016[, c('prob\_1', 'Year', 'Position\_grid')], preds\_2017[, c('prob\_1', 'Year', 'Position\_grid')], preds\_2018[, c('prob\_1', 'Year', 'Position\_grid')],  
 preds\_2019[, c('prob\_1', 'Year', 'Position\_grid')],  
 preds\_2020[, c('prob\_1', 'Year', 'Position\_grid')], preds\_2021[, c('prob\_1', 'Year', 'Position\_grid')],   
 preds\_2022[, c('prob\_1', 'Year', 'Position\_grid')], preds\_2023[, c('prob\_1', 'Year', 'Position\_grid')]) %>%   
 mutate(pr = exp(prob\_1),  
 Year = factor(Year)) %>%   
 mutate( pr = pr / (1 + pr)) %>%   
 filter(Year %in% c('2021', '2022')) %>%   
 ggplot(aes(Position\_grid, y = pr, group = Year, col = Year)) +  
 geom\_line(size = 1) +   
 geom\_point(size = 2, pch = 21, col = 'black', aes(fill = Year)) +  
 theme\_bw() +  
 labs( y = 'Probability of finishing P1',  
 x = 'Starting Grid Position',  
 title = 'Model Results by Year')



It looks like drivers in 2022 had a better chance of winning a race from 5th or worse position on the starting grid. In 2021, drivers who started from the first two rows of the grid had a better chance of winning than they did in 2022. Are these differences due to the new regulations, or is this simply capturing the fact that Verstappen won from 7th, 10th, and 14th?

What about podiums?

rbind(preds\_2016[, c('prob\_3', 'Year', 'Position\_grid')], preds\_2017[, c('prob\_3', 'Year', 'Position\_grid')], preds\_2018[, c('prob\_3', 'Year', 'Position\_grid')],  
 preds\_2019[, c('prob\_3', 'Year', 'Position\_grid')],  
 preds\_2020[, c('prob\_3', 'Year', 'Position\_grid')], preds\_2021[, c('prob\_3', 'Year', 'Position\_grid')],   
 preds\_2022[, c('prob\_3', 'Year', 'Position\_grid')],   
 preds\_2023[, c('prob\_3', 'Year', 'Position\_grid')]) %>%  
 mutate(pr = exp(prob\_3),  
 Year = factor(Year)) %>%  
 mutate( pr = pr / (1 + pr)) %>%  
 filter(Year %in% c(2021, 2022)) %>%   
 ggplot(aes(Position\_grid, y = pr, group = Year, col = Year)) +  
 geom\_line(size = 1) +   
 geom\_point(size = 2, pch = 21, col = 'black', aes(fill = Year)) +  
 theme\_bw() +  
 labs( y = 'Probability of finishing on the podium',  
 x = 'Starting Grid Position',  
 title = 'Model Results by Year')



The podium probabilities are actually more compelling. Drivers starting from 5th or lower have an even greater chance of finishing on the podium in 2022.

## 6.2 Explore Circuit Specific Models

It is generally accepted among racing teams, drivers, and enthusiasts that overtaking (or passing opponents) is easier at some circuits compared to others. Let’s explore whether there are indeed differences in our model coefficients across different circuits.

### 6.2.1 Bahrain

The Bahrain Grand Prix is the first race of the 2023 season. What type of finish might we expect, if only using starting grid position as a predictor?

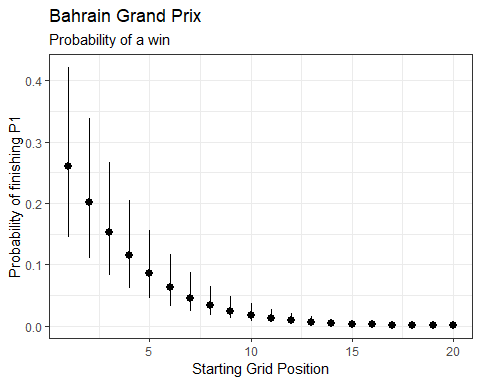
bahrain\_model <- vglm(Position\_race ~ Position\_grid,   
 family = cumulative(parallel = T, reverse = F),  
 data = grids\_and\_races %>%  
 filter(Race == 'bahrain'))  
  
summary(bahrain\_model)

##   
## Call:  
## vglm(formula = Position\_race ~ Position\_grid, family = cumulative(parallel = T,   
## reverse = F), data = grids\_and\_races %>% filter(Race == "bahrain"))  
##   
## Coefficients:   
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept):1 -0.71682 0.38544 -1.860 0.06292 .   
## (Intercept):2 0.25694 0.31651 0.812 0.41691   
## (Intercept):3 0.97556 0.30294 3.220 0.00128 \*\*   
## (Intercept):4 1.56822 0.30841 5.085 3.68e-07 \*\*\*  
## (Intercept):5 2.07456 0.32182 6.446 1.15e-10 \*\*\*  
## (Intercept):6 2.53923 0.33940 7.481 7.35e-14 \*\*\*  
## (Intercept):7 2.97706 0.35938 8.284 < 2e-16 \*\*\*  
## (Intercept):8 3.39893 0.38081 8.926 < 2e-16 \*\*\*  
## (Intercept):9 3.81682 0.40337 9.462 < 2e-16 \*\*\*  
## (Intercept):10 4.23980 0.42686 9.933 < 2e-16 \*\*\*  
## (Intercept):11 4.63579 0.44890 10.327 < 2e-16 \*\*\*  
## (Intercept):12 5.07800 0.47318 10.732 < 2e-16 \*\*\*  
## (Intercept):13 5.54374 0.49812 11.129 < 2e-16 \*\*\*  
## (Intercept):14 6.03512 0.52391 11.519 < 2e-16 \*\*\*  
## (Intercept):15 6.51782 0.54970 11.857 < 2e-16 \*\*\*  
## (Intercept):16 7.12033 0.58570 12.157 < 2e-16 \*\*\*  
## (Intercept):17 8.11628 0.67346 12.052 < 2e-16 \*\*\*  
## (Intercept):18 9.22555 0.87402 10.555 < 2e-16 \*\*\*  
## Position\_grid -0.33144 0.03226 -10.274 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Number of linear predictors: 18   
##   
## Residual deviance: 874.4459 on 3077 degrees of freedom  
##   
## Log-likelihood: -437.223 on 3077 degrees of freedom  
##   
## Number of Fisher scoring iterations: 7   
##   
## Warning: Hauck-Donner effect detected in the following estimate(s):  
## '(Intercept):17', '(Intercept):18'  
##   
##   
## Exponentiated coefficients:  
## Position\_grid   
## 0.7178892

Make Predictions for the grid

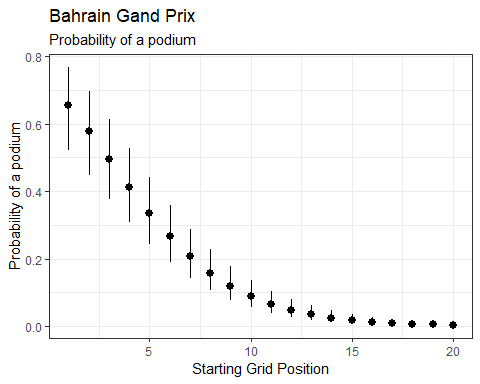
# Make Predictions for each Finishing position (1:20), using each Starting Position (1:20)  
predictions\_bahrain <- as.data.frame(predictvglm(bahrain\_model, newdata = starting\_grid, se.fit = T)$fitted.values)  
  
# rename the columns  
names(predictions\_bahrain) <- placing\_df$prob[1:18]  
  
# Extract the standard error for each prediction  
standard\_errors\_bahrain <- as.data.frame(predictvglm(bahrain\_model, newdata = starting\_grid, se.fit = T)$se.fit)  
  
# rename the standard error   
names(standard\_errors\_bahrain) <- placing\_df$se[1:18]

starting\_grid %>%  
 bind\_cols(pr\_1 = predictions\_bahrain$prob\_1,  
 se\_1 = standard\_errors\_bahrain$se\_1) %>%  
 mutate(pr = exp(pr\_1),  
 lower = exp(pr\_1 - se\_1 \* 1.96),  
 upper = exp(pr\_1 + se\_1 \* 1.96)) %>%  
 mutate( pr = pr / (1 + pr),  
 lower = lower / (1 + lower),  
 upper = upper / (1 + upper)) %>%  
 ggplot(aes(Position\_grid, y = pr, ymin = lower, ymax = upper)) +  
 geom\_pointrange() +  
 theme\_bw() +  
 labs( y = 'Probability of finishing P1',  
 x = 'Starting Grid Position',  
 title = 'Bahrain Grand Prix',  
 subtitle = 'Probability of a win')



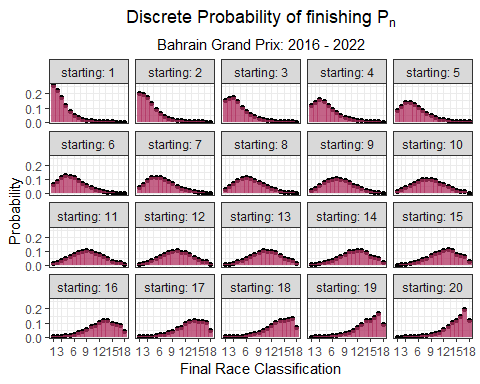
What’s the probability of a podium?

starting\_grid %>%  
 bind\_cols(pr\_3 = predictions\_bahrain$prob\_3,  
 se\_3 = standard\_errors\_bahrain$se\_3) %>%  
 mutate(pr = exp(pr\_3),  
 lower = exp(pr\_3 - se\_3 \* 1.96),  
 upper = exp(pr\_3 + se\_3 \* 1.96)) %>%  
 mutate( pr = pr / (1 + pr),  
 lower = lower / (1 + lower),  
 upper = upper / (1 + upper)) %>%  
 ggplot(aes(Position\_grid, y = pr, ymin = lower, ymax = upper)) +  
 geom\_pointrange() +  
 theme\_bw() +  
 labs( y = 'Probability of a podium',  
 x = 'Starting Grid Position',  
 title = 'Bahrain Gand Prix',  
 subtitle = 'Probability of a podium')



And now, I’ll construct probability distributions for each finishing position given a starting a positin at the Bahrain GP.

bahrain\_preds\_se\_full <- predictions\_bahrain %>%  
 mutate(starting = 1:n()) %>%  
 pivot\_longer(- starting, names\_to = 'placing', values\_to = 'prob') %>%   
 bind\_cols(standard\_errors\_bahrain %>% mutate(starting = 1:n()) %>%  
 pivot\_longer(- starting, names\_to = 'placing', values\_to = 'se') %>%  
 dplyr::select(se)) %>%  
 mutate(placing = as.numeric(str\_remove(placing, 'prob\_'))) %>%  
 mutate(prob = exp(prob)) %>%  
 mutate(prob = prob / (1 + prob)) %>%  
 group\_by(starting) %>%   
 mutate(discrete\_prob = ifelse(is.na(prob - lag(prob)), prob, (prob - lag(prob)))) %>%  
 ungroup()  
  
bahrain\_preds\_se\_full %>%  
 mutate(starting\_pos = as.factor(paste0('P',starting)),  
 starting\_pos = fct\_reorder(starting\_pos, starting)) %>%  
 ggplot(aes(placing, discrete\_prob)) +  
 geom\_point() +  
 geom\_segment(aes(x = placing, xend = placing, y = 0, yend = discrete\_prob),  
 size = 2, alpha = 0.75, col = 'maroon') +  
 theme\_bw() +  
 facet\_wrap(~ starting, labeller = label\_both) +  
 labs(x = 'Final Race Classification',  
 y = 'Probability',  
 title = expression(Discrete~Probability~of~finishing~P[n]),  
 subtitle = 'Bahrain Grand Prix: 2016 - 2022') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5)) +  
 scale\_x\_continuous(breaks = c(1, 3, 6, 9, 12, 15, 18), labels = c(1, 3, 6, 9, 12, 15, 18))



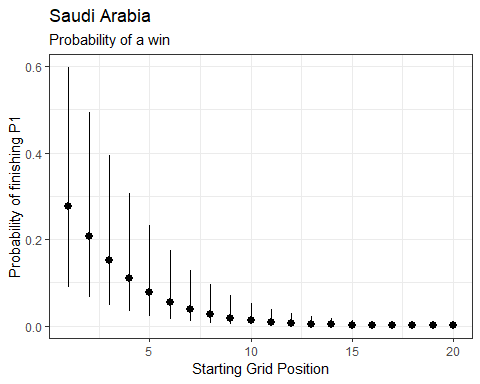
### 6.2.2 Saudi Arabia

The second race of the 2023 season takes place in Saudi Arabia at the Jeddah Corniche Circuit. Jeddah is the fastest street circuit in Formula 1. How does it compare to other street circuits?

# Fit model  
saudiarabia\_model <- vglm(Position\_race ~ Position\_grid,   
 family = cumulative(parallel = T, reverse = F),  
 data = grids\_and\_races %>%  
 filter(Race == 'saudi-arabia'))  
  
# Make Predictions for each Finishing position (1:20), using each Starting Position (1:20)  
predictions\_saudiarabia <- as.data.frame(predictvglm(saudiarabia\_model, newdata = starting\_grid, se.fit = T)$fitted.values)  
  
# rename the columns  
names(predictions\_saudiarabia) <- placing\_df$prob[1:17]  
  
# Extract the standard error for each prediction  
standard\_errors\_saudiarabia <- as.data.frame(predictvglm(saudiarabia\_model, newdata = starting\_grid, se.fit = T)$se.fit)  
  
# rename the standard error   
names(standard\_errors\_saudiarabia) <- placing\_df$se[1:17]

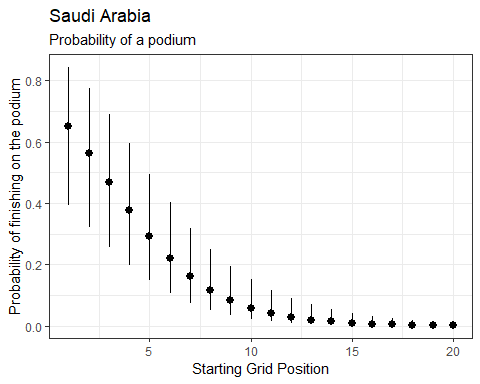
Plot Saudi Arabian Grand Prix probability of a win

starting\_grid %>%  
 bind\_cols(pr\_1 = predictions\_saudiarabia$prob\_1,  
 se\_1 = standard\_errors\_saudiarabia$se\_1) %>%  
 mutate(pr = exp(pr\_1),  
 lower = exp(pr\_1 - se\_1 \* 1.96),  
 upper = exp(pr\_1 + se\_1 \* 1.96)) %>%  
 mutate( pr = pr / (1 + pr),  
 lower = lower / (1 + lower),  
 upper = upper / (1 + upper)) %>%  
 ggplot(aes(Position\_grid, y = pr, ymin = lower, ymax = upper)) +  
 geom\_pointrange() +  
 theme\_bw() +  
 labs( y = 'Probability of finishing P1',  
 x = 'Starting Grid Position',  
 title = 'Saudi Arabia',  
 subtitle = 'Probability of a win')

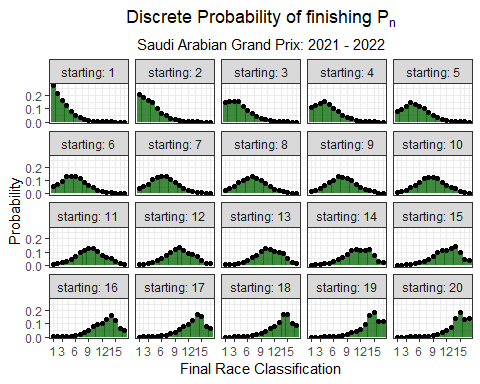


Plot Saudi Arabian Grand Prix probability of a podium.

starting\_grid %>%  
 bind\_cols(pr\_3 = predictions\_saudiarabia$prob\_3,  
 se\_3 = standard\_errors\_saudiarabia$se\_3) %>%  
 mutate(pr = exp(pr\_3),  
 lower = exp(pr\_3 - se\_3 \* 1.96),  
 upper = exp(pr\_3 + se\_3 \* 1.96)) %>%  
 mutate( pr = pr / (1 + pr),  
 lower = lower / (1 + lower),  
 upper = upper / (1 + upper)) %>%  
 ggplot(aes(Position\_grid, y = pr, ymin = lower, ymax = upper)) +  
 geom\_pointrange() +  
 theme\_bw() +  
 labs( y = 'Probability of finishing on the podium',  
 x = 'Starting Grid Position',  
 title = 'Saudi Arabia',  
 subtitle = 'Probability of a podium')



saudiarabia\_preds\_se\_full <- predictions\_saudiarabia %>%  
 mutate(starting = 1:n()) %>%  
 pivot\_longer(- starting, names\_to = 'placing', values\_to = 'prob') %>%   
 bind\_cols(standard\_errors\_saudiarabia %>% mutate(starting = 1:n()) %>%  
 pivot\_longer(- starting, names\_to = 'placing', values\_to = 'se') %>%  
 dplyr::select(se)) %>%  
 mutate(placing = as.numeric(str\_remove(placing, 'prob\_'))) %>%  
 mutate(prob = exp(prob)) %>%  
 mutate(prob = prob / (1 + prob)) %>%  
 group\_by(starting) %>%   
 mutate(discrete\_prob = ifelse(is.na(prob - lag(prob)), prob, (prob - lag(prob)))) %>%  
 ungroup()  
  
saudiarabia\_preds\_se\_full %>%  
 mutate(starting\_pos = as.factor(paste0('P',starting)),  
 starting\_pos = fct\_reorder(starting\_pos, starting)) %>%  
 ggplot(aes(placing, discrete\_prob)) +  
 geom\_segment(aes(x = placing, xend = placing, y = 0, yend = discrete\_prob),  
 size = 2, alpha = 0.75, col = 'darkgreen') +  
 geom\_point() +  
 theme\_bw() +  
 facet\_wrap(~ starting, labeller = label\_both) +  
 labs(x = 'Final Race Classification',  
 y = 'Probability',  
 title = expression(Discrete~Probability~of~finishing~P[n]),  
 subtitle = 'Saudi Arabian Grand Prix: 2021 - 2022') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5)) +  
 scale\_x\_continuous(breaks = c(1, 3, 6, 9, 12, 15, 18), labels = c(1, 3, 6, 9, 12, 15, 18))



There are only two races to construct this model, so there’s tremendous variability in the estimated probabilities. That being said, we can still compare to a much slower street circuit like Monaco.

### 6.2.3 Monaco

monaco\_model <- vglm(Position\_race ~ Position\_grid,   
 family = cumulative(parallel = T, reverse = F),  
 data = grids\_and\_races %>%  
 filter(Race == 'monaco'))  
  
summary(monaco\_model)

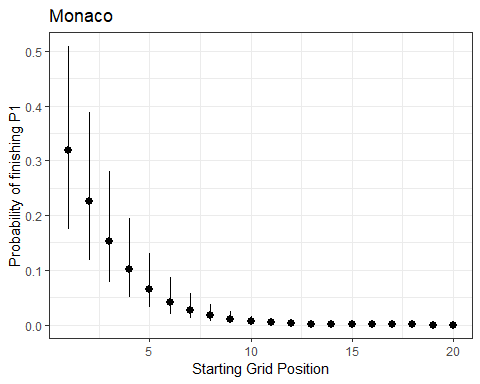
##   
## Call:  
## vglm(formula = Position\_race ~ Position\_grid, family = cumulative(parallel = T,   
## reverse = F), data = grids\_and\_races %>% filter(Race == "monaco"))  
##   
## Coefficients:   
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept):1 -0.28789 0.42256 -0.681 0.4957   
## (Intercept):2 0.76160 0.35184 2.165 0.0304 \*   
## (Intercept):3 1.56028 0.34291 4.550 5.36e-06 \*\*\*  
## (Intercept):4 2.28028 0.35665 6.394 1.62e-10 \*\*\*  
## (Intercept):5 2.93310 0.38156 7.687 1.50e-14 \*\*\*  
## (Intercept):6 3.54044 0.41230 8.587 < 2e-16 \*\*\*  
## (Intercept):7 4.16225 0.44900 9.270 < 2e-16 \*\*\*  
## (Intercept):8 4.78580 0.48898 9.787 < 2e-16 \*\*\*  
## (Intercept):9 5.35326 0.52673 10.163 < 2e-16 \*\*\*  
## (Intercept):10 5.83811 0.55934 10.437 < 2e-16 \*\*\*  
## (Intercept):11 6.32467 0.59207 10.682 < 2e-16 \*\*\*  
## (Intercept):12 6.89611 0.63004 10.946 < 2e-16 \*\*\*  
## (Intercept):13 7.51153 0.66946 11.220 < 2e-16 \*\*\*  
## (Intercept):14 8.15972 0.70863 11.515 < 2e-16 \*\*\*  
## (Intercept):15 8.81674 0.74604 11.818 < 2e-16 \*\*\*  
## (Intercept):16 9.38034 0.77793 12.058 < 2e-16 \*\*\*  
## (Intercept):17 10.16416 0.82962 12.252 < 2e-16 \*\*\*  
## (Intercept):18 11.11774 0.93077 11.945 < 2e-16 \*\*\*  
## Position\_grid -0.47330 0.04255 -11.124 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Number of linear predictors: 18   
##   
## Residual deviance: 700.7868 on 2699 degrees of freedom  
##   
## Log-likelihood: -350.3934 on 2699 degrees of freedom  
##   
## Number of Fisher scoring iterations: 7   
##   
## Warning: Hauck-Donner effect detected in the following estimate(s):  
## '(Intercept):18'  
##   
##   
## Exponentiated coefficients:  
## Position\_grid   
## 0.6229455

Make Predictions for the grid

# Make Predictions for each Finishing position (1:20), using each Starting Position (1:20)  
predictions\_monaco <- as.data.frame(predictvglm(monaco\_model, newdata = starting\_grid, se.fit = T)$fitted.values)  
  
# rename the columns  
names(predictions\_monaco) <- placing\_df$prob[1:18]  
  
# Extract the standard error for each prediction  
standard\_errors\_monaco <- as.data.frame(predictvglm(monaco\_model, newdata = starting\_grid, se.fit = T)$se.fit)  
  
# rename the standard error   
names(standard\_errors\_monaco) <- placing\_df$se[1:18]

Now, I’ll the Probability(P1 finish | a starting grid) at Monaco:

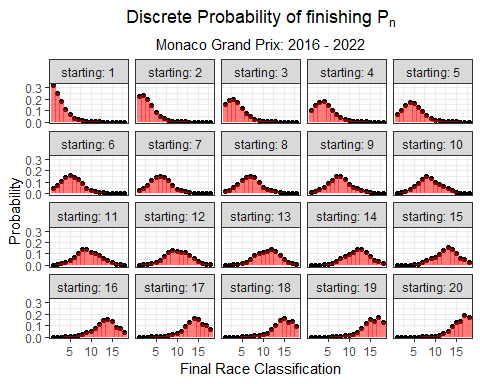
starting\_grid %>%  
 bind\_cols(pr\_1 = predictions\_monaco$prob\_1,  
 se\_1 = standard\_errors\_monaco$se\_1) %>%  
 mutate(pr = exp(pr\_1),  
 lower = exp(pr\_1 - se\_1 \* 1.96),  
 upper = exp(pr\_1 + se\_1 \* 1.96)) %>%  
 mutate( pr = pr / (1 + pr),  
 lower = lower / (1 + lower),  
 upper = upper / (1 + upper)) %>%  
 ggplot(aes(Position\_grid, y = pr, ymin = lower, ymax = upper)) +  
 geom\_pointrange() +  
 theme\_bw() +  
 labs( y = 'Probability of finishing P1',  
 x = 'Starting Grid Position',  
 title = 'Monaco')



monaco\_preds\_se\_full <- predictions\_monaco %>%  
 mutate(starting = 1:n()) %>%  
 pivot\_longer(- starting, names\_to = 'placing', values\_to = 'prob') %>%   
 bind\_cols(standard\_errors\_monaco %>% mutate(starting = 1:n()) %>%  
 pivot\_longer(- starting, names\_to = 'placing', values\_to = 'se') %>%  
 dplyr::select(se)) %>%  
 mutate(placing = as.numeric(str\_remove(placing, 'prob\_'))) %>%  
 mutate(prob = exp(prob)) %>%  
 mutate(prob = prob / (1 + prob)) %>%  
 group\_by(starting) %>%   
 mutate(discrete\_prob = ifelse(is.na(prob - lag(prob)), prob, (prob - lag(prob)))) %>%  
 ungroup()

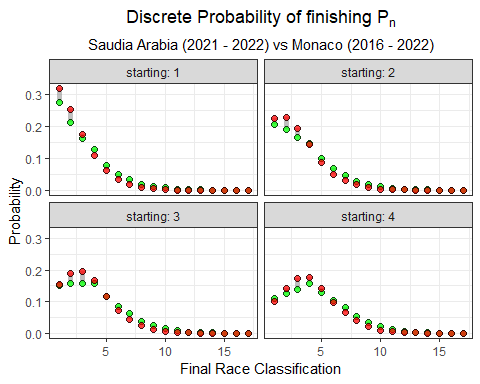
Plot the discrete probability distributions.

monaco\_preds\_se\_full %>%  
 mutate(starting\_pos = as.factor(paste0('P',starting)),  
 starting\_pos = fct\_reorder(starting\_pos, starting)) %>%  
 ggplot(aes(placing, discrete\_prob)) +  
 geom\_point() +  
 geom\_segment(aes(x = placing, xend = placing, y = 0, yend = discrete\_prob),  
 size = 2, alpha = 0.5, col = 'red') +  
 theme\_bw() +  
 facet\_wrap(~ starting, labeller = label\_both) +  
 labs(x = 'Final Race Classification',  
 y = 'Probability',  
 title = expression(Discrete~Probability~of~finishing~P[n]),  
 subtitle = 'Monaco Grand Prix: 2016 - 2022') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5))



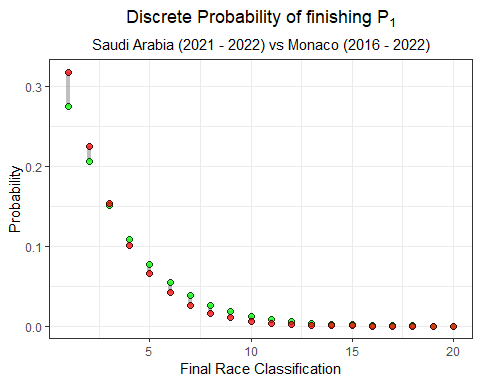
Let’s compare the fastest street circuit (Jeddah) to the slowest (Monaco).

saudiarabia\_preds\_se\_full %>%  
 dplyr::select(starting, placing, discrete\_prob) %>%  
 left\_join(monaco\_preds\_se\_full %>%  
 dplyr::select(starting, placing, discrete\_prob), by = c("starting", "placing"), suffix = c("\_saudiarabia", "\_monaco")) %>%  
 mutate(starting\_pos = as.factor(paste0('P',starting)),  
 starting\_pos = fct\_reorder(starting\_pos, starting),  
 delta = ifelse(discrete\_prob\_monaco > discrete\_prob\_saudiarabia, 'monaco', 'saudiarabia')) %>%  
 filter(starting\_pos %in% c('P1', 'P2', 'P3', 'P4')) %>%  
 ggplot() +  
 geom\_segment(aes(x = placing, xend = placing, y = discrete\_prob\_saudiarabia, yend = discrete\_prob\_monaco, col = delta),  
 size = 2, alpha = 0.5) +  
 geom\_point(size = 2, pch = 21, col = 'black', aes(placing, discrete\_prob\_saudiarabia, fill = 'Saudi Arabia'), alpha = 0.75, show.legend = F) +  
 geom\_point(size = 2, pch = 21, col = 'black', aes(placing, discrete\_prob\_monaco, fill = 'Monaco'), alpha = 0.75, show.legend = F) +  
 scale\_color\_manual("", values = c('Saudi Arabia' = 'green', 'Monaco' = 'red')) +  
 scale\_fill\_manual("", values = c('Saudi Arabia' = 'green', 'Monaco' = 'red')) +  
 theme\_bw() +  
 facet\_wrap(~ starting, labeller = label\_both) +  
 labs(x = 'Final Race Classification',  
 y = 'Probability',  
 title = expression(Discrete~Probability~of~finishing~P[n]),  
 subtitle = 'Saudia Arabia (2021 - 2022) vs Monaco (2016 - 2022)') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5))



So, nothing too interesting for the first 2 rows of the starting grid. Is it statistically easier to win in Jeddah from further down the grid?

saudiarabia\_preds\_se\_full %>%  
 dplyr::select(starting, placing, discrete\_prob) %>%  
 left\_join(monaco\_preds\_se\_full %>%  
 dplyr::select(starting, placing, discrete\_prob), by = c("starting", "placing"), suffix = c("\_saudiarabia", "\_monaco")) %>%  
 mutate(starting\_pos = as.factor(paste0('P',starting)),  
 starting\_pos = fct\_reorder(starting\_pos, starting),  
 delta = ifelse(discrete\_prob\_monaco > discrete\_prob\_saudiarabia, 'monaco', 'saudiarabia')) %>%  
 filter(placing == 1) %>%  
 ggplot() +  
 geom\_segment(aes(x = starting, xend = starting, y = discrete\_prob\_saudiarabia, yend = discrete\_prob\_monaco, col = delta),  
 size = 1.5, alpha = 0.5) +  
 geom\_point(size = 2, pch = 21, col = 'black', aes(starting, discrete\_prob\_saudiarabia), fill = 'green', alpha = 0.75) +  
 geom\_point(size = 2, pch = 21, col = 'black', aes(starting, discrete\_prob\_monaco), fill = 'red', alpha = 0.75) +  
 theme\_bw() +  
 scale\_color\_manual("", values = c('Saudi Arabia' = 'green', 'Monaco' = 'red')) +  
 labs(x = 'Final Race Classification',  
 y = 'Probability',  
 title = expression(Discrete~Probability~of~finishing~P[1]),  
 subtitle = 'Saudi Arabia (2021 - 2022) vs Monaco (2016 - 2022)') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5))



Maybe slightly. It looks like there is just a much stronger advantage to starting at the front of the grid in Monaco.

### 6.2.4 Brazil

The Sao Paulo Grand Prix is perhaps the easiest circuit to overtake on. Let’s compare the model predictions and probabilities for Brazil to those of Monaco.

brazil\_model <- vglm(Position\_race ~ Position\_grid,   
 family = cumulative(parallel = T, reverse = F),  
 data = grids\_and\_races %>%  
 filter(Race == 'brazil'))  
  
summary(brazil\_model)

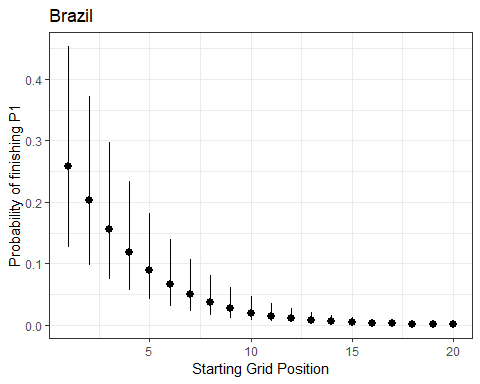
##   
## Call:  
## vglm(formula = Position\_race ~ Position\_grid, family = cumulative(parallel = T,   
## reverse = F), data = grids\_and\_races %>% filter(Race == "brazil"))  
##   
## Coefficients:   
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept):1 -0.73953 0.45966 -1.609 0.107645   
## (Intercept):2 0.24697 0.37789 0.654 0.513401   
## (Intercept):3 0.92998 0.36331 2.560 0.010476 \*   
## (Intercept):4 1.39449 0.36752 3.794 0.000148 \*\*\*  
## (Intercept):5 1.87117 0.38103 4.911 9.07e-07 \*\*\*  
## (Intercept):6 2.31922 0.40025 5.794 6.85e-09 \*\*\*  
## (Intercept):7 2.74110 0.42263 6.486 8.83e-11 \*\*\*  
## (Intercept):8 3.16937 0.44836 7.069 1.56e-12 \*\*\*  
## (Intercept):9 3.59978 0.47607 7.561 3.99e-14 \*\*\*  
## (Intercept):10 4.02870 0.50443 7.987 1.39e-15 \*\*\*  
## (Intercept):11 4.45699 0.53261 8.368 < 2e-16 \*\*\*  
## (Intercept):12 4.86320 0.55869 8.705 < 2e-16 \*\*\*  
## (Intercept):13 5.28447 0.58499 9.033 < 2e-16 \*\*\*  
## (Intercept):14 5.78672 0.61592 9.395 < 2e-16 \*\*\*  
## (Intercept):15 6.29784 0.64839 9.713 < 2e-16 \*\*\*  
## (Intercept):16 6.97457 0.69829 9.988 < 2e-16 \*\*\*  
## (Intercept):17 7.75148 0.78010 9.936 < 2e-16 \*\*\*  
## (Intercept):18 9.20953 1.11599 8.252 < 2e-16 \*\*\*  
## Position\_grid -0.31650 0.03799 -8.330 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Number of linear predictors: 18   
##   
## Residual deviance: 594.0966 on 2087 degrees of freedom  
##   
## Log-likelihood: -297.0483 on 2087 degrees of freedom  
##   
## Number of Fisher scoring iterations: 6   
##   
## Warning: Hauck-Donner effect detected in the following estimate(s):  
## '(Intercept):17', '(Intercept):18'  
##   
##   
## Exponentiated coefficients:  
## Position\_grid   
## 0.728696

Make Predictions for the grid

# Make Predictions for each Finishing position (1:20), using each Starting Position (1:20)  
predictions\_brazil<- as.data.frame(predictvglm(brazil\_model, newdata = starting\_grid, se.fit = T)$fitted.values)  
  
# rename the columns  
names(predictions\_brazil) <- placing\_df$prob[1:18]  
  
# Extract the standard error for each prediction  
standard\_errors\_brazil <- as.data.frame(predictvglm(brazil\_model, newdata = starting\_grid, se.fit = T)$se.fit)  
  
# rename the standard error   
names(standard\_errors\_brazil) <- placing\_df$se[1:18]

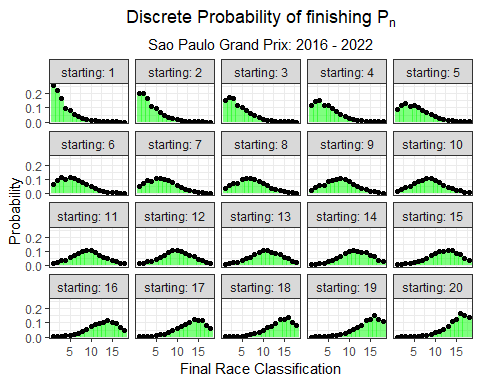
Now, I’ll the Probability(P1 finish | a starting grid) at Monaco:

starting\_grid %>%  
 bind\_cols(pr\_1 = predictions\_brazil$prob\_1,  
 se\_1 = standard\_errors\_brazil$se\_1) %>%  
 mutate(pr = exp(pr\_1),  
 lower = exp(pr\_1 - se\_1 \* 1.96),  
 upper = exp(pr\_1 + se\_1 \* 1.96)) %>%  
 mutate( pr = pr / (1 + pr),  
 lower = lower / (1 + lower),  
 upper = upper / (1 + upper)) %>%  
 ggplot(aes(Position\_grid, y = pr, ymin = lower, ymax = upper)) +  
 geom\_pointrange() +  
 theme\_bw() +  
 labs( y = 'Probability of finishing P1',  
 x = 'Starting Grid Position',  
 title = 'Brazil')



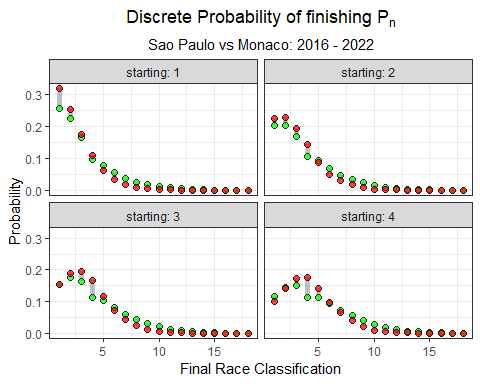
Plot the discrete probability distributions.

brazil\_preds\_se\_full <- predictions\_brazil %>%  
 mutate(starting = 1:n()) %>%  
 pivot\_longer(- starting, names\_to = 'placing', values\_to = 'prob') %>%   
 bind\_cols(standard\_errors\_brazil %>% mutate(starting = 1:n()) %>%  
 pivot\_longer(- starting, names\_to = 'placing', values\_to = 'se') %>%  
 dplyr::select(se)) %>%  
 mutate(placing = as.numeric(str\_remove(placing, 'prob\_'))) %>%  
 mutate(prob = exp(prob)) %>%  
 mutate(prob = prob / (1 + prob)) %>%  
 group\_by(starting) %>%   
 mutate(discrete\_prob = ifelse(is.na(prob - lag(prob)), prob, (prob - lag(prob)))) %>%  
 ungroup()  
  
  
brazil\_preds\_se\_full %>%  
 mutate(starting\_pos = as.factor(paste0('P',starting)),  
 starting\_pos = fct\_reorder(starting\_pos, starting)) %>%  
 ggplot(aes(placing, discrete\_prob)) +  
 geom\_segment(aes(x = placing, xend = placing, y = 0, yend = discrete\_prob),  
 size = 2, alpha = 0.5, col = 'green') +  
 geom\_point() +  
 theme\_bw() +  
 facet\_wrap(~ starting, labeller = label\_both) +  
 labs(x = 'Final Race Classification',  
 y = 'Probability',  
 title = expression(Discrete~Probability~of~finishing~P[n]),  
 subtitle = 'Sao Paulo Grand Prix: 2016 - 2022') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5))



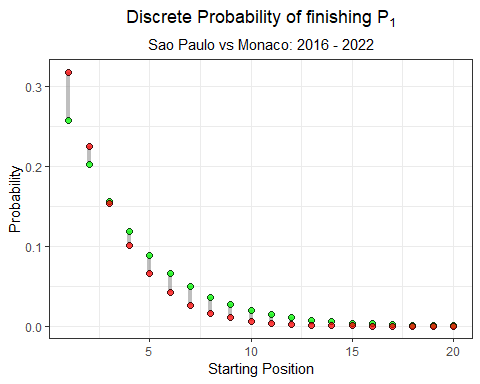
How does Brazil and Monaco compare?

brazil\_preds\_se\_full %>%  
 dplyr::select(starting, placing, discrete\_prob) %>%  
 left\_join(monaco\_preds\_se\_full %>%  
 dplyr::select(starting, placing, discrete\_prob), by = c("starting", "placing"), suffix = c("\_brazil", "\_monaco")) %>%  
 mutate(starting\_pos = as.factor(paste0('P',starting)),  
 starting\_pos = fct\_reorder(starting\_pos, starting),  
 delta = ifelse(discrete\_prob\_monaco > discrete\_prob\_brazil, 'monaco', 'brazil')) %>%  
 filter(starting\_pos %in% c('P1', 'P2', 'P3', 'P4')) %>%  
 ggplot() +  
 geom\_segment(aes(x = placing, xend = placing, y = discrete\_prob\_brazil, yend = discrete\_prob\_monaco, col = delta),  
 size = 2, alpha = 0.5) +  
 geom\_point(size = 2, pch = 21, col = 'black', aes(placing, discrete\_prob\_brazil, fill = 'Brazil'), alpha = 0.75, show.legend = F) +  
 geom\_point(size = 2, pch = 21, col = 'black', aes(placing, discrete\_prob\_monaco, fill = 'Monaco'), alpha = 0.75, show.legend = F) +  
 scale\_color\_manual("", values = c('Brazil' = 'green', 'Monaco' = 'red')) +  
 scale\_fill\_manual("", values = c('Brazil' = 'green', 'Monaco' = 'red')) +  
 theme\_bw() +  
 facet\_wrap(~ starting, labeller = label\_both) +  
 labs(x = 'Final Race Classification',  
 y = 'Probability',  
 title = expression(Discrete~Probability~of~finishing~P[n]),  
 subtitle = 'Sao Paulo vs Monaco: 2016 - 2022') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5))



Or, let’s compare the probability of finishing P1 given any starting grid position?

brazil\_preds\_se\_full %>%  
 dplyr::select(starting, placing, discrete\_prob) %>%  
 left\_join(monaco\_preds\_se\_full %>%  
 dplyr::select(starting, placing, discrete\_prob), by = c("starting", "placing"), suffix = c("\_brazil", "\_monaco")) %>%  
 mutate(starting\_pos = as.factor(paste0('P',starting)),  
 starting\_pos = fct\_reorder(starting\_pos, starting),  
 delta = ifelse(discrete\_prob\_monaco > discrete\_prob\_brazil, 'monaco', 'brazil')) %>%  
 filter(placing == 1) %>%  
 ggplot() +  
 geom\_segment(aes(x = starting, xend = starting, y = discrete\_prob\_brazil, yend = discrete\_prob\_monaco, col = delta),  
 size = 1.5, alpha = 0.5) +  
 geom\_point(size = 2, pch = 21, col = 'black', aes(starting, discrete\_prob\_brazil), fill = 'green', alpha = 0.75) +  
 geom\_point(size = 2, pch = 21, col = 'black', aes(starting, discrete\_prob\_monaco), fill = 'red', alpha = 0.75) +  
 theme\_bw() +  
 scale\_color\_manual("", values = c('Brazil' = 'green', 'Monaco' = 'red')) +  
 labs(x = 'Starting Position',  
 y = 'Probability',  
 title = expression(Discrete~Probability~of~finishing~P[1]),  
 subtitle = 'Sao Paulo vs Monaco: 2016 - 2022') +  
 theme(plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5))



There’s a distinct advantage in starting the Monaco Grand Prix from the front row, in terms of a driver’s probability of winning the race. A driver has a higher likelihood of winning from any other row on the gird of the Sao Paulo Grand Prix.

Stated plainly… Qualifying on the front row in Monaco is very important! In Sao Paulo, a driver can make up positions.

# 7 DNF (Did Not Finish)

There are lots of different reasons a car DNFs from a race. Mechanical failure, crashes, or precautionary considerations can all lead to a car retiring early from a race. DNFs are almost always unexpected, and can have substantial influence on the outcome of the race for other drivers (see Abu Dhabi 2021).

What is the probability that a car will DNF? Well, in an attempt to answer that question, one could simply calculate the proportion of DNFs from 2014 to 2022. That calculation would reveal that 17.5% of cars DNF from a race:

mean(races\_allyears\_dnf$DNF, na.rm = T)

## [1] 0.1719205

A more statistically sound way to approach this question is to use a logistic regression model. The most basic logistic regression model would be an intercept-only model. Notice that the transformed intercept is identical to the manually calcualted proportion of 17.5%.

dnf.model.0 <- glm(DNF ~ 1, data = races\_allyears\_dnf,  
 family = binomial(link = 'logit'))  
  
# Transform the coefficient to probability scale  
plogis(coef(dnf.model.0))

## (Intercept)   
## 0.1719205

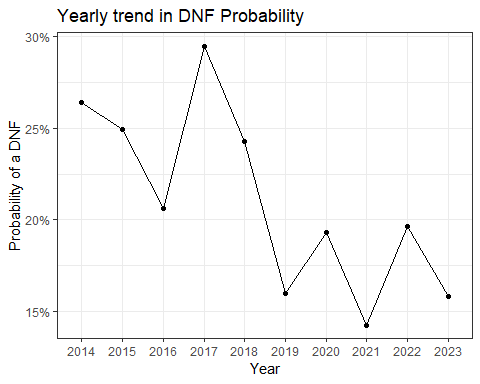
Does this proportion vary over time? One would think! I would assume that as regulation era evolves, the engineers will build increasingly dependable cars. Let’s see if the data backs this up:

dnf.model.1 <- glm(DNF ~ 0 + Year, data = races\_allyears\_dnf,  
 family = binomial(link = 'logit'))  
  
# Transform the coefficient to probability scale  
data.frame(Probability = plogis(coef(dnf.model.1)))

## Probability  
## Year2014 0.2088452  
## Year2015 0.1994681  
## Year2016 0.1709957  
## Year2017 0.2275000  
## Year2018 0.1952381  
## Year2019 0.1380952  
## Year2020 0.1617647  
## Year2021 0.1250000  
## Year2022 0.1642857  
## Year2023 0.1366743

It looks like the data support this to some degree. We can plot this data to make it a little more clear:

tidy(dnf.model.1, exp = T) %>%  
 mutate(term = str\_remove(term, "Year")) %>%   
 ggplot(aes(term, estimate)) +  
 geom\_line(group = 1) +  
 geom\_point() +   
 theme\_bw() +  
 labs(y = 'Probability of a DNF',  
 x = 'Year',  
 title = 'Yearly trend in DNF Probability') +  
scale\_y\_continuous(labels = scales::percent\_format())



If we were to look at the number of DNFs by track, some locations stand out. For instance, Baku, Australia, and the U.S. Grand Prix all have DNF percentages of at least 20% since 2014.

races\_allyears\_dnf %>%  
 group\_by(Circuit) %>%  
 summarize(`% of DNFs` = round(mean(DNF, na.rm = T), 3)) %>%  
 ungroup() %>%  
 arrange(desc(`% of DNFs`)) %>%  
 mutate(`% of DNFs` = scales::percent(`% of DNFs`)) %>%   
 gt()

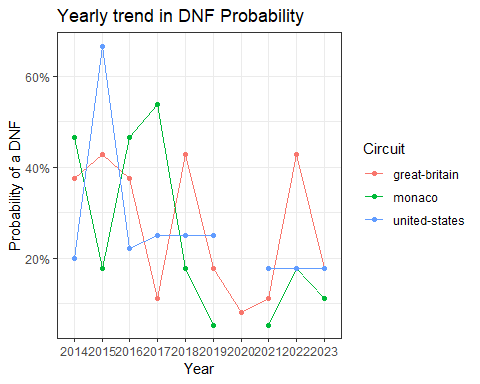
| Circuit | % of DNFs |
| --- | --- |
| australia | 26.10% |
| azerbaijan | 24.20% |
| germany | 22.10% |
| singapore | 21.50% |
| malaysia | 20.50% |
| saudi-arabia | 20.00% |
| united-states | 20.00% |
| canada | 19.50% |
| great-britain | 19.20% |
| austria | 19.10% |
| monaco | 18.50% |
| italy | 18.30% |
| europe | 18.20% |
| bahrain | 17.90% |
| russia | 16.60% |
| belgium | 16.20% |
| brazil | 15.60% |
| mexico | 15.40% |
| hungary | 15.20% |
| las-vegas | 15.00% |
| styria | 15.00% |
| abu-dhabi | 13.40% |
| netherlands | 13.30% |
| japan | 12.80% |
| france | 12.50% |
| miami | 12.50% |
| spain | 12.30% |
| china | 12.10% |
| qatar | 10.00% |
| turkey | 10.00% |
| portugal | 5.00% |

We can add an interaction term to this model, and make a yearly estimate for each circuit in our data.

dnf.model.2 <- glm(DNF ~ 0 + Year:Circuit, data = races\_allyears\_dnf,  
 family = binomial(link = 'logit'))

Here’s a comparison of this updated model’s output across a few different circuits:

tidy(dnf.model.2, exp = T) %>%  
 mutate(term = str\_remove(term, "Year")) %>%  
 separate(term, sep = ':Circuit', c("Year", "Circuit")) %>%  
 filter(Circuit %in% c('monaco', 'united-states', 'great-britain')) %>%   
 ggplot(aes(Year, estimate, group = Circuit, col = Circuit)) +  
 geom\_line() +  
 geom\_point() +   
 theme\_bw() +  
 labs(y = 'Probability of a DNF',  
 x = 'Year',  
 title = 'Yearly trend in DNF Probability') +  
scale\_y\_continuous(labels = scales::percent\_format())

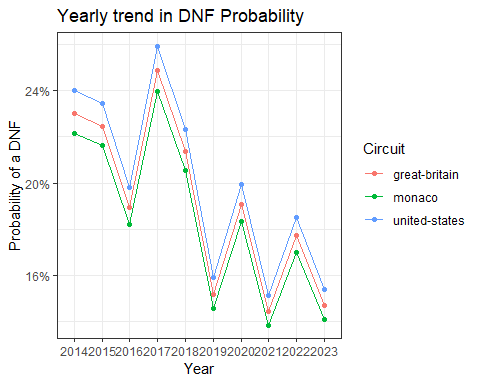


An alternative to fitting an estimate for each race-year combination is to model both the year and track estimates, but not as an interaction term.

dnf.model.3 <- glm(DNF ~ 0 + Year + Circuit, data = races\_allyears\_dnf,  
 family = binomial(link = 'logit'))

As we can see in the figure below, this model assumes a consistent yearly effect and adjusts the probability estimate up or down based on the circuit. So, for that rainy 2015 U.S. Grand Prix, this model gives a more *sober* result.

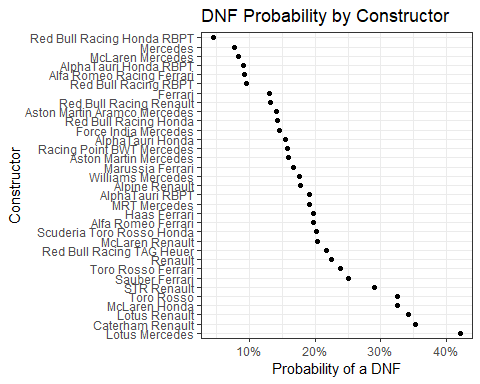
emmeans(dnf.model.3, ~ Circuit + Year, type = 'response') %>%   
 as.data.frame() %>%   
 filter(Circuit %in% c('monaco', 'united-states', 'great-britain')) %>%   
 ggplot(aes(Year, prob, group = Circuit, col = Circuit)) +  
 geom\_line() +  
 geom\_point() +   
 theme\_bw() +  
 labs(y = 'Probability of a DNF',  
 x = 'Year',  
 title = 'Yearly trend in DNF Probability') +  
scale\_y\_continuous(labels = scales::percent\_format())



We could use a similar approach to estimate the probability of a DNF by constructor. One could hypothesize that the car with the best power unit, Mercedes, could run the engine at a more conservative mode, thus reducing the risk of failure. Let’s see if the data shows any significant shifts in DNF probability.

dnf.model.4 <- glm(DNF ~ 0 + Car, data = races\_allyears\_dnf,  
 family = binomial(link = 'logit'))

emmeans(dnf.model.4, ~ Car, type = 'response') %>%   
 as.data.frame() %>%   
 ggplot(aes(prob, fct\_reorder(Car, desc(prob)))) +  
 geom\_line() +  
 geom\_point() +   
 theme\_bw() +  
 labs(x = 'Probability of a DNF',  
 y = 'Constructor',  
 title = 'DNF Probability by Constructor') +  
scale\_x\_continuous(labels = scales::percent\_format())

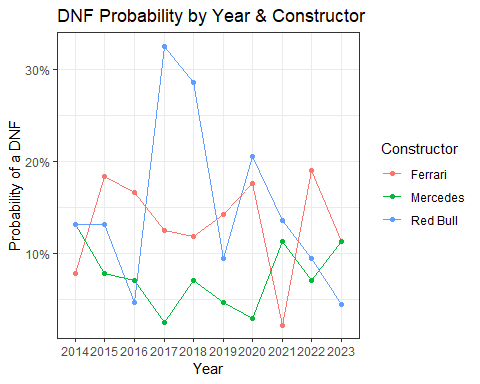


There certainly appears to be some variability in DNF probability across the various constructors. During the Turbo-hybrid era, Mercedes unquestionably had the best power unit (overall). And according to this data, Mercedes also had the lowest probability of a DNF from 2014 to 2022 (Red Bull was quite reliable in 2023). In fact, McLaren using a Mercedes power unit had the third lowest DNF probability. Does this vary by year?

dnf.model.5 <- glm(DNF ~ 0 + Car:Year, data = races\_allyears\_dnf,  
 family = binomial(link = 'logit'))

After fitting the model, I’ll focus on the top 3 teams: Mercedes, Ferrari, and Red Bull:

emmeans(dnf.model.5, ~ Car:Year, type = 'response') %>%   
 as.data.frame() %>%  
 filter(!is.na(Year),  
 !is.na(prob),  
 Car %in% c('Mercedes', 'Ferrari',   
 'Red Bull Racing Honda RBPT', 'Red Bull Racing RBPT', 'Red Bull Racing Renault', 'Red Bull Racing Honda', 'Red Bull Racing TAG Heuer')) %>%   
 mutate(constructor = case\_when(  
 Car != 'Mercedes' & Car != 'Ferrari' ~ 'Red Bull',   
 Car == 'Mercedes' ~ 'Mercedes',   
 Car == 'Ferrari' ~ 'Ferrari')) %>%   
 ggplot(aes(Year, prob, group = constructor, col = constructor)) +  
 geom\_path() +  
 geom\_point() +   
 theme\_bw() +  
 labs(y = 'Probability of a DNF',  
 col = 'Constructor',  
 title = 'DNF Probability by Year & Constructor') +  
scale\_y\_continuous(labels = scales::percent\_format())



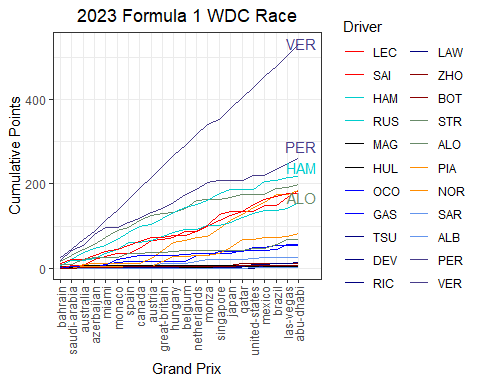
# 8 The WCC (World Constructor Championship) and WDC (World Drivers Championship)

Over the course of a season, teams and drivers compete for the WCC (World Constructor Championship) and WDC (World Drivers Championship), respectively. The WCC is awarded to the team with the highest total points scored and the WDC is awarded to the driver with the highest total points. During some seasons, the champions are crowned early (i.e. 2022), while other championships may not be decided until the final lap of the season (i.e. 2021).

During this chapter, we will take a look at championship fights.

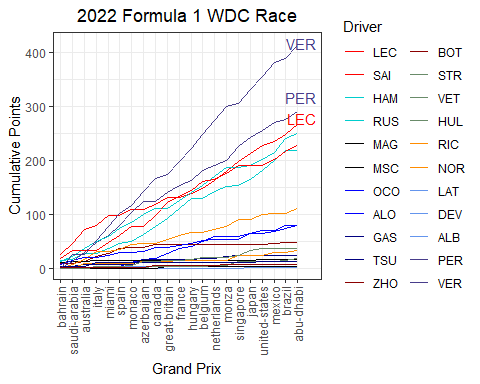
By starting in 2023, we see one of the most dominant seasons ever by a Driver. Max won 19 of 22 races and finished on the podium in all bu Singapore. Lewis Hamilton and Sergio Perez fought for second place in the WDC. And, Fernando Alonso made a welcome return to the front of the grid.

# Pull race results  
races2023 <- race\_result\_scraper(2023)  
  
races2023 %>%  
 mutate(rowid = 1:n()) %>%  
 group\_by(Driver) %>%   
 mutate(cumulativePoints = cumsum(Points)) %>%  
 ungroup() %>%   
 ggplot(aes(fct\_reorder(Race, rowid), cumulativePoints,  
 group = Driver, col = Driver)) +  
 geom\_line() +  
 theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1),  
 plot.title = element\_text(hjust = 0.5)) +  
 labs(x = 'Grand Prix',  
 y = 'Cumulative Points',  
 title = '2023 Formula 1 WDC Race') +  
 scale\_color\_manual(values = c("red", "red",   
 "cyan3", "cyan3",   
 "black", "black",  
 "blue", "blue",  
 "navy", "navy", "navy", "navy",  
 "darkred", "darkred",   
 "darkseagreen4", "darkseagreen4",  
 "darkorange", "darkorange",  
 "cornflowerblue", "cornflowerblue",   
 "darkslateblue", "darkslateblue"),  
 breaks = c("LEC", "SAI",  
 "HAM", "RUS",   
 "MAG", "HUL",   
 "OCO", "GAS",  
 "TSU", "DEV", "RIC", "LAW",  
 "ZHO", "BOT",  
 "STR", "ALO",   
 "PIA", "NOR",  
 "SAR", "ALB",  
 "PER", "VER"  
 )) +  
 geom\_text\_repel(data = ~ subset(.x, Driver %in% c('VER', 'PER', 'HAM', 'ALO') & Race == 'abu-dhabi'),  
 aes(label = Driver),  
 nudge\_x = 2, nudge\_y = 3, show.legend = F)



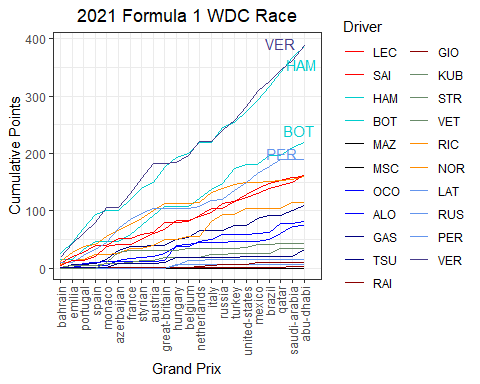
Here’s a quick look at the 2022 WDC fight. This year wasn’t exactly a nail-biting affair! If we look at the entire plot, the championship appears to be over before it even began. However, in reality Verstappen mathematically clinched the championship after the Japanese Grand Prix.

# Pull race results  
races2022 <- race\_result\_scraper(2022)  
  
races2022 %>%  
 mutate(rowid = 1:n()) %>%  
 group\_by(Driver) %>%   
 mutate(cumulativePoints = cumsum(Points)) %>%  
 ungroup() %>%   
 ggplot(aes(fct\_reorder(Race, rowid), cumulativePoints,  
 group = Driver, col = Driver)) +  
 geom\_line() +  
 theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1),  
 plot.title = element\_text(hjust = 0.5)) +  
 labs(x = 'Grand Prix',  
 y = 'Cumulative Points',  
 title = '2022 Formula 1 WDC Race') +  
 scale\_color\_manual(values = c("red", "red",   
 "cyan3", "cyan3",   
 "black", "black",  
 "blue", "blue",  
 "navy", "navy",  
 "darkred", "darkred",   
 "darkseagreen4", "darkseagreen4", "darkseagreen4",  
 "darkorange", "darkorange",  
 "cornflowerblue", "cornflowerblue", "cornflowerblue",  
 "darkslateblue", "darkslateblue"),  
 breaks = c("LEC", "SAI",  
 "HAM", "RUS",   
 "MAG", "MSC",   
 "OCO", "ALO",  
 "GAS", "TSU",   
 "ZHO", "BOT",  
 "STR", "VET", "HUL",   
 "RIC", "NOR",  
 "LAT", "DEV", "ALB",  
 "PER", "VER"  
 )) +  
 geom\_text\_repel(data = ~ subset(.x, Driver %in% c('VER', 'PER', 'LEC') & Race == 'abu-dhabi'),  
 aes(label = Driver),  
 nudge\_x = 2, nudge\_y = 3, show.legend = F)



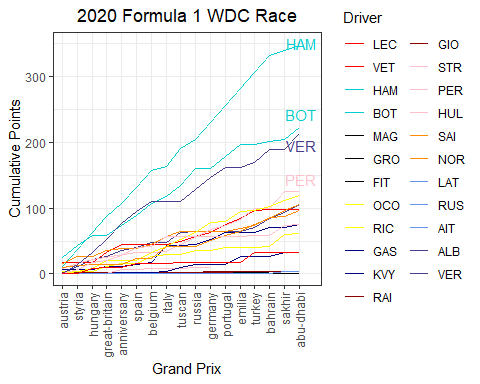
The 2021 Championship fight was not settled until the final lap of the season.

# Pull race results  
races2021 <- race\_result\_scraper(2021)  
  
races2021 %>%  
 mutate(Driver = ifelse(Driver == 'ikk', 'RAI', Driver)) %>%   
 mutate(rowid = 1:n()) %>%  
 group\_by(Driver) %>%   
 mutate(cumulativePoints = cumsum(Points)) %>%  
 ungroup() %>%   
 ggplot(aes(fct\_reorder(Race, rowid), cumulativePoints,  
 group = Driver, col = Driver)) +  
 geom\_line() +  
 theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1),  
 plot.title = element\_text(hjust = 0.5)) +  
 labs(x = 'Grand Prix',  
 y = 'Cumulative Points',  
 title = '2021 Formula 1 WDC Race') +  
 scale\_color\_manual(values = c("red", "red",   
 "cyan3", "cyan3",   
 "black", "black",  
 "blue", "blue",  
 "navy", "navy",  
 "darkred", "darkred",   
 "darkseagreen4", "darkseagreen4", "darkseagreen4",  
 "darkorange", "darkorange",  
 "cornflowerblue", "cornflowerblue", "cornflowerblue",  
 "darkslateblue", "darkslateblue"),  
 breaks = c("LEC", "SAI",  
 "HAM", "BOT",   
 "MAZ", "MSC",   
 "OCO", "ALO",  
 "GAS", "TSU",   
 "RAI", "GIO", "KUB",  
 "STR", "VET",   
 "RIC", "NOR",  
 "LAT", "RUS",  
 "PER", "VER"  
 )) +  
 geom\_text\_repel(data = ~ subset(.x, Driver %in% c('VER', 'HAM', 'BOT', 'PER') & Race == 'abu-dhabi'),  
 aes(label = Driver),  
 nudge\_x = 1.5, nudge\_y = 3, show.legend = F)



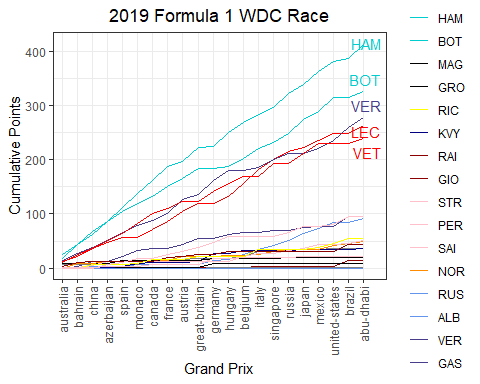
The 2020 Championship fight …

# Pull race results  
races2020 <- race\_result\_scraper(2020)  
  
races2020 %>%  
 mutate(Driver = ifelse(Driver == 'ikk', 'RAI', Driver)) %>%   
 mutate(rowid = 1:n()) %>%  
 group\_by(Driver) %>%   
 mutate(cumulativePoints = cumsum(Points)) %>%  
 ungroup() %>%   
 ggplot(aes(fct\_reorder(Race, rowid), cumulativePoints,  
 group = Driver, col = Driver)) +  
 geom\_line() +  
 theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1),  
 plot.title = element\_text(hjust = 0.5)) +  
 labs(x = 'Grand Prix',  
 y = 'Cumulative Points',  
 title = '2020 Formula 1 WDC Race') +  
 geom\_text\_repel(data = ~ subset(.x, Driver %in% c('VER', 'HAM', 'BOT', 'PER') & Race == 'abu-dhabi'),  
 aes(label = Driver),  
 nudge\_x = 1.5, nudge\_y = 3, show.legend = F) +  
 scale\_color\_manual(values = c("red", "red",   
 "cyan3", "cyan3",   
 "black", "black", "black",  
 "yellow", "yellow",  
 "navy", "navy",  
 "darkred", "darkred",   
 "pink", "pink", "pink",  
 "darkorange", "darkorange",  
 "cornflowerblue", "cornflowerblue", "cornflowerblue",  
 "darkslateblue", "darkslateblue"),  
 breaks = c("LEC", "VET",  
 "HAM", "BOT",  
 "MAG", "GRO", "FIT",  
 "OCO", "RIC",  
 "GAS", "KVY",   
 "RAI", "GIO",  
 "STR", "PER", "HUL",   
 "SAI", "NOR",  
 "LAT", "RUS", "AIT",  
 "ALB", "VER"  
 ))



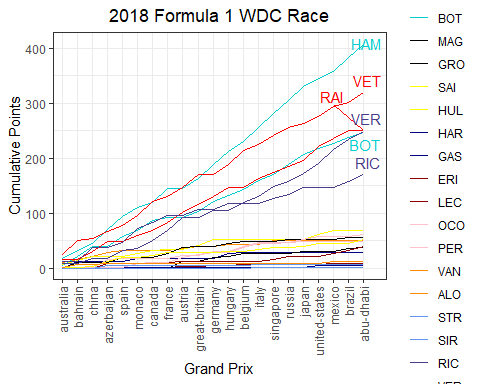
The 2019 Championship fight …

# Pull race results  
races2019 <- race\_result\_scraper(2019)  
  
races2019 %>%  
 mutate(Driver = ifelse(Driver == 'ikk', 'RAI', Driver)) %>%   
 mutate(rowid = 1:n()) %>%  
 group\_by(Driver) %>%   
 mutate(cumulativePoints = cumsum(Points)) %>%  
 ungroup() %>%   
 ggplot(aes(fct\_reorder(Race, rowid), cumulativePoints,  
 group = Driver, col = Driver)) +  
 geom\_line() +  
 theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1),  
 plot.title = element\_text(hjust = 0.5)) +  
 labs(x = 'Grand Prix',  
 y = 'Cumulative Points',  
 title = '2019 Formula 1 WDC Race') +  
 geom\_text\_repel(data = ~ subset(.x, Driver %in% c('VER', 'HAM', 'BOT', 'LEC', 'VET') & Race == 'abu-dhabi'),  
 aes(label = Driver),  
 nudge\_x = 1.5, nudge\_y = 1.5, show.legend = F) +  
 scale\_color\_manual(values = c("red", "red",   
 "cyan3", "cyan3",   
 "black", "black",   
 "yellow", "yellow",  
 "navy",   
 "darkred", "darkred",   
 "pink", "pink", "pink",  
 "darkorange", "darkorange",  
 "cornflowerblue", "cornflowerblue",  
 "darkslateblue", "darkslateblue", "darkslateblue"),  
 breaks = c("LEC", "VET",  
 "HAM", "BOT",  
 "MAG", "GRO",  
 "OCO", "RIC",  
 "KVY",  
 "RAI", "GIO",  
 "STR", "PER",   
 "SAI", "NOR",  
 "LAT", "RUS",  
 "ALB", "VER", "GAS"  
 ))



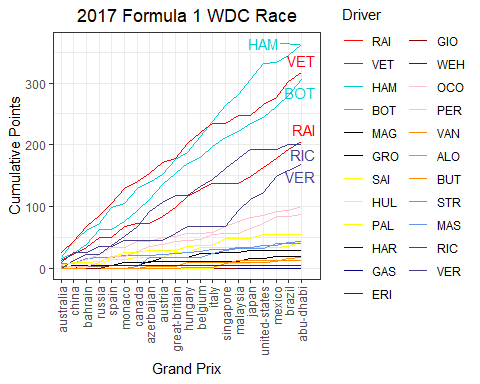
The 2018 Championship fight …

# Pull race results  
races2018 <- race\_result\_scraper(2018)  
  
races2018 %>%  
 mutate(Driver = ifelse(Driver == 'ikk', 'RAI', Driver)) %>%   
 mutate(rowid = 1:n()) %>%  
 group\_by(Driver) %>%   
 mutate(cumulativePoints = cumsum(Points)) %>%  
 ungroup() %>%   
 ggplot(aes(fct\_reorder(Race, rowid), cumulativePoints,  
 group = Driver, col = Driver)) +  
 geom\_line() +  
 theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1),  
 plot.title = element\_text(hjust = 0.5)) +  
 labs(x = 'Grand Prix',  
 y = 'Cumulative Points',  
 title = '2018 Formula 1 WDC Race') +  
 geom\_text\_repel(data = ~ subset(.x, Driver %in% c('VER', 'HAM', 'BOT', 'RAI', 'VET', 'RIC') & Race == 'abu-dhabi'),  
 aes(label = Driver),  
 nudge\_x = 1.5, nudge\_y = 1, show.legend = F) +   
 scale\_color\_manual(values = c("red", "red",   
 "cyan3", "cyan3",   
 "black", "black",   
 "yellow", "yellow",  
 "navy", "navy",  
 "darkred", "darkred",   
 "pink", "pink",  
 "darkorange", "darkorange",  
 "cornflowerblue", "cornflowerblue",  
 "darkslateblue", "darkslateblue"),  
 breaks = c("RAI", "VET",  
 "HAM", "BOT",  
 "MAG", "GRO",  
 "SAI", "HUL",  
 "HAR", "GAS",  
 "ERI", "LEC",  
 "OCO", "PER",   
 "VAN", "ALO",  
 "STR", "SIR",  
 "RIC", "VER"  
 ))



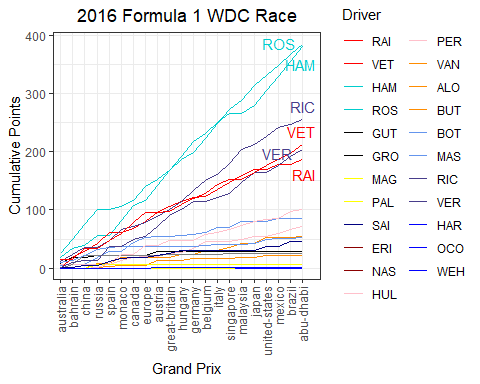
The 2017 Championship fight …

# Pull race results  
races2017 <- race\_result\_scraper(2017)  
  
races2017 %>%  
 mutate(Driver = ifelse(Driver == 'ikk', 'RAI', Driver),  
 Driver = ifelse(Driver == 'Resta', 'DIV', Driver)) %>%   
 mutate(rowid = 1:n()) %>%  
 group\_by(Driver) %>%   
 mutate(cumulativePoints = cumsum(Points)) %>%  
 ungroup() %>%   
 ggplot(aes(fct\_reorder(Race, rowid), cumulativePoints,  
 group = Driver, col = Driver)) +  
 geom\_line() +  
 theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1),  
 plot.title = element\_text(hjust = 0.5)) +  
 labs(x = 'Grand Prix',  
 y = 'Cumulative Points',  
 title = '2017 Formula 1 WDC Race') + geom\_text\_repel(data = ~ subset(.x, Driver %in% c('RIC', 'HAM', 'BOT', 'RAI', 'VET', 'VER') & Race == 'abu-dhabi'),  
 aes(label = Driver),  
 nudge\_x = 1.5, nudge\_y = 1, show.legend = F) +   
 scale\_color\_manual(values = c("red", "red",   
 "cyan3", "cyan3",   
 "black", "black",   
 "yellow", "yellow", "yellow",  
 "navy", "navy", "navy",   
 "darkred", "darkred", "darkred",   
 "pink", "pink",   
 "darkorange", "darkorange", "darkorange",  
 "cornflowerblue", "cornflowerblue", "cornflowerblue",  
 "darkslateblue", "darkslateblue"),  
 breaks = c("RAI", "VET",  
 "HAM", "BOT",  
 "MAG", "GRO",  
 "SAI", "HUL", "PAL",  
 "HAR", "GAS", "KYV",  
 "ERI", "GIO", "WEH",  
 "OCO", "PER",   
 "VAN", "ALO", "BUT",  
 "STR", "MAS", "DIR",  
 "RIC", "VER"  
 ))



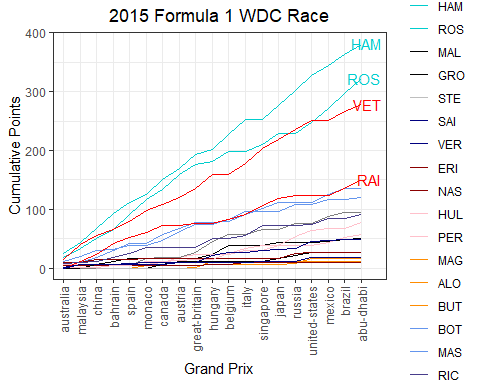
The 2016 Championship fight …

# Pull race results  
races2016 <- race\_result\_scraper(2016)  
  
races2016 %>%  
 mutate(Driver = ifelse(Driver == 'ikk', 'RAI', Driver),  
 Driver = ifelse(Driver == 'Resta', 'DIV', Driver)) %>%   
 mutate(rowid = 1:n()) %>%  
 group\_by(Driver) %>%   
 mutate(cumulativePoints = cumsum(Points)) %>%  
 ungroup() %>%   
 ggplot(aes(fct\_reorder(Race, rowid), cumulativePoints,  
 group = Driver, col = Driver)) +  
 geom\_line() +  
 theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1),  
 plot.title = element\_text(hjust = 0.5)) +  
 labs(x = 'Grand Prix',  
 y = 'Cumulative Points',  
 title = '2016 Formula 1 WDC Race') + geom\_text\_repel(data = ~ subset(.x, Driver %in% c('RIC', 'HAM', 'ROS', 'RAI', 'VET', 'VER') & Race == 'abu-dhabi'),  
 aes(label = Driver),  
 nudge\_x = 1.5, nudge\_y = 1, show.legend = F) +   
 scale\_color\_manual(values = c("red", "red",   
 "cyan3", "cyan3",   
 "black", "black",   
 "yellow", "yellow",   
 "navy", "navy",  
 "darkred", "darkred",   
 "pink", "pink",   
 "darkorange", "darkorange", "darkorange",  
 "cornflowerblue", "cornflowerblue",   
 "darkslateblue", "darkslateblue",  
 "blue", "blue", "blue"),  
 breaks = c("RAI", "VET",  
 "HAM", "ROS",  
 "GUT", "GRO",  
 "MAG", "PAL",   
 "SAI", "KYV",  
 "ERI", "NAS",   
 "HUL", "PER",   
 "VAN", "ALO", "BUT",  
 "BOT", "MAS",  
 "RIC", "VER",  
 "HAR", "OCO", "WEH"  
 ))



The 2015 Championship fight …

# Pull race results  
races2015 <- race\_result\_scraper(2015)  
  
races2015 %>%  
 mutate(Driver = ifelse(Driver == 'ikk', 'RAI', Driver),  
 Driver = ifelse(Driver == 'Resta', 'DIV', Driver)) %>%   
 mutate(rowid = 1:n()) %>%  
 group\_by(Driver) %>%   
 mutate(cumulativePoints = cumsum(Points)) %>%  
 ungroup() %>%   
 ggplot(aes(fct\_reorder(Race, rowid), cumulativePoints,  
 group = Driver, col = Driver)) +  
 geom\_line() +  
 theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1),  
 plot.title = element\_text(hjust = 0.5)) +  
 labs(x = 'Grand Prix',  
 y = 'Cumulative Points',  
 title = '2015 Formula 1 WDC Race') + geom\_text\_repel(data = ~ subset(.x, Driver %in% c('HAM', 'ROS', 'RAI', 'VET') & Race == 'abu-dhabi'),  
 aes(label = Driver),  
 nudge\_x = 1.5, nudge\_y = 1, show.legend = F) +   
 scale\_color\_manual(values = c("red", "red",   
 "cyan3", "cyan3",   
 "black", "black",   
 "grey", "grey", "grey",  
 "navy", "navy",  
 "darkred", "darkred",   
 "pink", "pink",   
 "darkorange", "darkorange", "darkorange",  
 "cornflowerblue", "cornflowerblue",   
 "darkslateblue", "darkslateblue",  
 "blue", "blue", "blue"),  
 breaks = c("RAI", "VET",  
 "HAM", "ROS",  
 "MAL", "GRO",  
 "STE", "WEH", "ROS",  
 "SAI", "VER",  
 "ERI", "NAS",   
 "HUL", "PER",   
 "MAG", "ALO", "BUT",  
 "BOT", "MAS",  
 "RIC", "KYV",  
 "HAR", "OCO", "WEH"  
 ))



The 2014 Championship fight …

# Pull race results  
races2014 <- race\_result\_scraper(2014)  
  
races2014 %>%  
 mutate(Driver = ifelse(Driver == 'ikk', 'RAI', Driver),  
 Driver = ifelse(Driver == 'Resta', 'DIV', Driver)) %>%   
 mutate(rowid = 1:n()) %>%  
 group\_by(Driver) %>%   
 mutate(cumulativePoints = cumsum(Points)) %>%  
 ungroup() %>%   
 ggplot(aes(fct\_reorder(Race, rowid), cumulativePoints,  
 group = Driver, col = Driver)) +  
 geom\_line() +  
 theme\_bw() +   
 theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1),  
 plot.title = element\_text(hjust = 0.5)) +  
 labs(x = 'Grand Prix',  
 y = 'Cumulative Points',  
 title = '2014 Formula 1 WDC Race') + geom\_text\_repel(data = ~ subset(.x, Driver %in% c('HAM', 'ROS', 'RIC', 'VET') & Race == 'abu-dhabi'),  
 aes(label = Driver),  
 nudge\_x = 1.5, nudge\_y = 1, show.legend = F) +   
 scale\_color\_manual(values = c("red", "red",   
 "cyan3", "cyan3",   
 "black", "black",   
 "grey", "grey", "grey",  
 "navy", "navy",  
 "darkred", "darkred",   
 "pink", "pink",   
 "darkorange", "darkorange",   
 "cornflowerblue", "cornflowerblue",   
 "darkslateblue", "darkslateblue",  
 "seagreen", "seagreen", "seagreen", "seagreen"),  
 breaks = c("RAI", "ALO",  
 "HAM", "ROS",  
 "MAL", "GRO",  
 "CHI", "BIA", "ROS",  
 "KYV", "VER",  
 "SUT", "GUT",   
 "HUL", "PER",   
 "MAG", "BUT",  
 "BOT", "MAS",  
 "RIC", "VET",  
 "ERI", "STE", "KOB", "LOT"  
 ))

