Interactive Analysis of Formula 1 Qualifying Data

Norah Kuduk

Advisor: Dr. Matt Higham

Department of Mathematics, Computer Science, and Statistics

St. Lawrence University

May 9, 2025

Table of contents

Introduction

In this project, we focus on analyzing motorsport telemetry data, using data from the 2018 to 2024 Formula 1 (F1) seasons. Formula 1 has a vast amount of data used widely in the sport, making it perfect for visualization and analysis. This project centers on the analysis of F1 qualifying sessions, highlighting comparisons between two drivers in the same race or the same driver across different seasons. We built a Shiny application to allow users to interactively explore qualifying data and create custom visualizations by selecting drivers, seasons, and circuits. As case studies, we look at different qualifying sessions across years with the same driver and across drivers during the same race. For looking at different years, we look at Lando Norris during the São Paulo Grand Prix in 2023 versus in 2024. We look at Charles Leclerc and Oscar Piastri during the 2024 Monaco Grand Prix to compare drivers (and teams). Finally, we look at the 2024 Canadian Grand Prix to observe differentials between Max Verstappen and Sergio Pérez as teammates.

Outline

We begin by exploring what Formula 1 is and how a race weekend is scheduled in Section . We then examine why specifically qualifying sessions are essential for a weekend, why they are ideal for analysis, and what we hope to explore in the data in Section . In Section , we look at the R package `f1dataR` to examine its general usage and why it was instrumental to this project. We will go into more detail about the data used for the app plots in Section . Then,

in Section , we look at how the shiny app was developed, and how a user might utilize it to explore a qualifying session. We then look at two case studies in Section and Section for examples of how the functions can be used to visualize data. We also highlight how contrasting elements like weather or track layout make for engaging qualifying sessions. We also look at a case study in Section to observe the variation between teammates, because they typically have the same machinery. We end with Section to talk about what future work could be done to make the Shiny app more robust, as well as Section , which highlights relevant functions and code used to create the Shiny app.

What is Formula 1?

Formula 1 (F1) is an international motorsport series governed by the Fédération Internationale de l'Automobile (FIA). It involves 20 drivers across 10 "constructors" who compete in 18-24 race weekends across a season. A "constructor" is essentially a team that develops a car for the season and has two drivers who drive for them. The group of drivers/teams for a season is called the "grid," which is named for the grid-like shape that drivers line up in before a race starts.

The combination of drivers and teams means there are two different goals at the end of an F1 season. One is the World Constructor's Championship (WCC), which is given to the team with the most points across both drivers. Points are awarded to the top 10 finishers in a race, with 25 points going to first, and 18, 15, 12, 10, 8, 6, 4, 2, and 1 points awarded for 2nd through 10th place, respectively. The other is the World Driver's Championship (WDC),

which is given to the individual driver with the most points. Though completely separate, it is very uncommon for the driver who wins the World Championship to be from a different team than the one that wins the WCC. Since 2000, there have only been three seasons when the team that won the WCC did not also produce the WDC.

A "typical" race weekend of the F1 season features practice, qualifying, and race sessions, each serving a different purpose. All race weekends are Friday through Sunday, keeping the same schedule to remain as consistent as possible. Fridays involve two "free practice" sessions where they can change car setups at will to allow the teams to gather data to prepare for the race. The first session, Free Practice 1 (FP1), is 60 minutes long, and will typically be focused on getting used to the track and any changes that may have been made from previous seasons, test setups, and gathering data. The second session, aptly named Free Practice 2 (FP2), is also 60 minutes long but typically involves longer race simulations and tire degradation. Saturdays also have two sessions: one final Free Practice 3 (FP3) and qualifying. FP3 is the final chance for setup changes and adjustments and is typically used for short qualifying-style runs. Qualifying is a knockout-style session that sets the starting order for the race day on Sunday.

Because this project focuses on qualifying, we'll go into more detail about what that session looks like. Qualifying, sometimes shortened to "quali," is split into three sessions: Q1, Q2, and Q3. Drivers are allowed to complete as many laps within the section as possible, with their fastest lap counted towards the session's final standings. Q1 is the most extended session, at 18 minutes, and all 20 drivers compete, but the slowest five are eliminated. Q2 is 15 minutes,

where the remaining 15 drivers compete, and another five are eliminated. The top 10 who make it to Q3 will use the session to finalize the order. It is important to note that positions are set at the end of the section, not at the end of the qualifying session, meaning that the bottom five eliminated will always make up starting places 20th-16th. This rule means that even if a car in Q2 goes a slower time than that of the 16th, it will still qualify as either 15th or above. It is possible, then, for a driver to have a faster lap than someone who qualified above them, but it's rare.

The other important part of Formula 1 to mention is the tire compounds. F1's sole tire sponsor is Pirelli, who supplies all the tires for the grid. The tires come in different compounds (soft, medium, hard) and are used for various track conditions and race strategies. The "slick" dry tires are color-coded: red for soft, yellow for medium, and white for hard. There are two "wet" tires for use when it's raining; intermediate tires are green for damp conditions, and full wet tires are blue. For a race weekend, teams are provided with a certain number of sets for each driver and must use at least two compounds during a race. Qualifying laps are almost always run on the soft tire, with the fastest laps typically coming in when drivers use a new soft, as it has the most grip but wears out quickly. Occasionally, you will see drivers on a used set of tires in qualifying to save a set for the race or on a set of mediums if strategy might lead to using more softs later. 2018 is the only year this project utilized the "hypersoft," "ultrasoft," and "supersoft" tires, which are now just all "soft."

This information is accurate through the end of the 2025 season (prior to F1 adding an eleventh team, Cadillac, and subsequently two more drivers to the grid).

Why Qualifying Laps?

A driver's fastest lap from a qualifying session highlights driver skill, car setup efficiency, and team strategy under peak performance conditions, providing a baseline with which to compare. Unlike race laps, which are influenced by tire wear, fuel load, traffic, and strategy, qualifying laps are run under optimal conditions every time because every thousandth of a second counts towards a better grid spot.

We say "optimal conditions" to highlight the idea that any variations in speed are due to driver skill or superior car development rather than unlucky strategy or tire wear. During qualifying, every driver completes laps on low fuel and fresh (or almost fresh) tires, making them the fastest possible. Reducing the variability also allows for more direct comparisons when changing one variable, like the driver or season, while keeping everything else constant because they always attempt a single flying lap under similar conditions. Additionally, because the focus is single flying laps, there are typically no interfering factors that vary lap time, like safety cars, overtaking, or pit stops. An essential rule during qualifying is that cars on a "slow lap" or "outlap" must move off the optimal racing line and out of the way of cars on flying laps to reduce interference. The only interference that might occur during a lap is driver error, mechanical failure, or a safety hazard caused by the environment that would require reduced speed or aborting the lap.

Qualifying also holds a lot of strategic importance to the race weekend, depending on the track. While there are some tracks where qualifying matters less to the race, like Max Verstappen's

incredible 17th to 1st drive at the 2024 São Paulo Grand Prix, there are many examples of where it determined the winner. We will later look at a case study in Monaco, a track infamous for lacking overtaking opportunities, making qualifying all the more critical.

Methods

The f1dataR Package

The `f1dataR` package is an R interface to access Formula 1 data, developed to make use of statistical tools in R with detailed motorsport telemetry and timing information. `f1dataR` leverages both the Jolpica API (formerly known as Ergast) for historical and session metadata, and a wrapper around the FastF1 Python library to extract rich telemetry data directly from F1's live timing systems. Some examples of data that can be accessed through `f1dataR` specifically are session-level data and driver/constructor results. Session-level data includes information on race weekend structure (practice, qualifying, sprint, and race), including start times, session types, and circuit metadata. In contrast, the results data includes finishing positions, grid penalties, retirements, and historical performance going back to the inaugural F1 World Championship in 1950. In addition to this already robust data set, `f1dataR` allows users to pull detailed telemetry data from every timed lap from 2018 through 2025, which includes speed traces, throttle and brake application, gear selection, DRS activation, and car position (X, Y coordinates).

The package also includes some pre-made functions to make life easier for users.

`theme_track()` modifies a `ggplot` object to look more like a track rather than a plot and changes the colors to match Formula 1's official color scheme. `plot_fastest_lap()` is used to simplify race data visualization and allow users to look at interesting data around gear or speed changes around a lap without needing to do any data manipulation themselves.

In this project, we used `f1dataR` to load the fastest qualifying laps for specific drivers, seasons, and races, access the telemetry data to analyze speed and gear changes, and visualize car positioning on the track using X/Y coordinates.

Data

The following section will highlight some of the data used to create the plots for exploring and animating qualifying laps.

The following data is used to display an informative table for lap time exploration and create the lollipop plot. Below, we show an example of lap time data for Lando Norris and Nico Hülkenberg at the 2024 Abu Dhabi Grand Prix.

Table 1: Lap Time Data for Norris and Hülkenberg, 2024 Abu Dhabi Grand Prix.

driver	lap_time	lap_number	compound
NOR	82.595	15	SOFT
HUL	82.886	16	SOFT
NOR	82.949	12	SOFT
HUL	83.040	10	SOFT

NOR	83.098	7	SOFT
HUL	83.492	13	SOFT

Variable Information:

- **driver**: A unique three-letter code that identifies the driver in the session, typically the first three letters of the driver's last name.
- **lap_time**: The final time of the lap in seconds.
- **lap_number**: The order in which the lap was completed (1 is the first lap of the session, etc.).
- **compound**: The tire compound on the car during the lap.

The following determines how the qualifying position relates to the final race position. Many more extenuating factors determine this, but it's nice to have a general idea of what happened in the race. Below, we show the results data for Norris and Hülkenberg at the 2024 Abu Dhabi Grand Prix.

Table 2: Results Data for Norris and Hülkenberg, 2024 Abu Dhabi Grand Prix.

driver_name	code	constructor_id	grid	position
Lando Norris	NOR	mclaren	1	1
Nico Hülkenberg	HUL	haas	7	8

Variable Information:

- **driver_name**: The full first and last name of the driver.
- **code**: A unique three-letter code that identifies the driver in the session, typically the first three letters of the driver's last name.
- **constructor_id**: The team/constructor that the driver drives for.
- **grid**: The place in which the driver qualified in during the session on Saturday.
- **position**: The place the driver finishes following the race on Sunday.

The following data plots the tracks and identifies how the cars should be moving in the animation. Below, we show a snippet of the telemetry data for Norris and Hülkenberg at the 2024 Abu Dhabi Grand Prix.

Table 3: Telemetry Data for Norris and Hülkenberg, 2024 Abu Dhabi Grand Prix

driver	date	time	speed	x	y	lap_number
NOR	2024-12-07 10:01:59	0.000	240.5357	606.7005	2085.295	fastest
NOR	2024-12-07 10:01:59	0.092	242.5071	668.0000	2093.000	fastest
NOR	2024-12-07 10:01:59	0.255	246.0000	777.7058	2105.640	fastest
NOR	2024-12-07 10:01:59	0.352	248.1556	844.0000	2113.000	fastest
NOR	2024-12-07 10:01:59	0.572	253.0444	997.0000	2130.000	fastest
NOR	2024-12-07 10:01:59	0.615	254.0000	1026.8670	2133.309	fastest

Variable Information:

- **date**: The date and time that the session (and lap) occurred.
- **time**: The running time from the start of the lap in seconds.
- **speed**: The car's current speed at the moment in the lap in kph.
- **x**: The horizontal position of the car.
- **y**: The vertical position of the car.
- **lap_number**: The number of the lap being used. By default, the package pulls the fastest lap.

Shiny App

To make our analysis interactive and accessible, we developed a Shiny web application. The app can be found at https://stlawu.shinyapps.io/f1-qualifying-app/. This app has two main modes: one that allows users to compare two drivers in the same qualifying session and one that compares how a single driver performed in different seasons at the same circuit. When first opening the app, there are three main tabs. The first has instructions for a user detailing the different comparison modes and the two separate pages that actually have to do with the data we pulled.

For ease of explanation, the following examples assume the user selected the 'Same Session' option to compare two drivers across a section.

The second page allows users to explore a variety of combinations of results through the 'Data Exploration Page.' First, the user will be able to select a year between 2018 and 2024. The app will then pull the race calendar from that season and populate the race selection with the names. After choosing a race, the app will get all the drivers that participated in the race, preventing issues later on with data not existing if a driver didn't compete for some reason or if there had been a prior mid-season driver change in races. After selecting the four inputs, the app shows a table of all the lap times across drivers, a table with the drivers' final qualifying and race positions, and a lollipop plot highlighting the tire usages across the session. In the lollipop plot, there is typically a group of laps under a certain threshold, with another group tending to be between 12-20 seconds slower. Those laps are typically aborted laps, or 'outlaps,' and aren't necessarily used in the analysis but are still helpful. An aborted lap indicates that a driver made an error at some point in the lap, which caused the lap to no longer be worth the effort of continuing to push. By looking at aborted laps, we can observe driver pace up until the incident, which gives an idea about potential differences in grid order and car potential. Outlaps are helpful to look at when considering the order of laps, in addition to comparing how different drivers prepare their tires and cars to go on a flying lap. If a driver consistently has a faster slow lap than another driver or constructor, you could further observe how the rest of their laps compare.

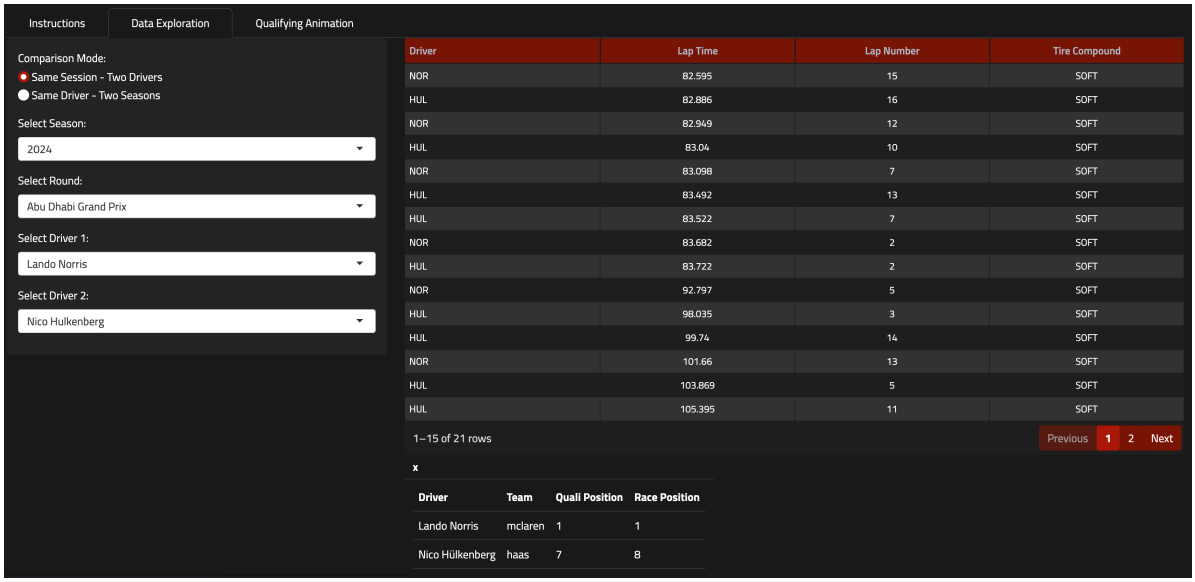

Figure 1: A screenshot of the Shiny app on the data exploration tab after plots are loaded.

The selection process for selecting a single driver across two years is similar, but you would first select the driver, then two seasons that they competed in, and then a race that existed in both seasons and had the driver compete in both. The app will then show the same three tables as before, but with combined data from both seasons.

The third page allows users to carry over their selections from the exploration or pick new inputs to animate a qualifying lap comparison. The animation runs on a constant loop, meaning it takes as long as the slowest lap time is compared, then starts over. The only additional input users can select is the 'Smoothing Factor.' The smoothing factor affects the caption of the animation, which shows the current time, as well as the speed of each respective car at that fraction of a second. Because F1, especially qualifying, is measured in thousandths of seconds, the telemetry data can have between 10 and 20 rows of data (speed, position, etc.) for

a single second, making it difficult to read if the speed was left in its raw state. The smoothing factor takes the average speed over that number of frames and uses that as the caption instead. We see a trade-off, with higher factors making the plot more readable but less accurate than a smaller factor. The small diamond point represents the start/finish line of the circuit. In the static plots of the case studies, there is an arrow indicating the direction the drivers are moving in.

Figure 2: A screenshot of the Shiny app on the animation tab.

Case Studies

The following section covers three different case studies, each examining a qualifying session that was interesting to watch, or showcased a large difference. The first, Section , showcases

how weather affects lap times by looking at lap times during the São Paulo Grand Prix in 2023 (dry) and 2024 (wet) for Lando Norris. The second and third case studies both look at scenarios where we compare drivers during the same session. Section looks at lap times from the drivers who qualified first and second, to observe how important the tenths of a second are around a track like Monaco. Section takes lap times from two drivers from the same team to highlight how even with the same machinery, driver ability plays a large part in pace and overall performance.

2023 vs 2024 São Paulo Grand Prix

The first case study will compare laps from the 2023 and 2024 São Paulo Grand Prix. These two qualifying sessions highlight the difference between wet and dry weather driving conditions, which might create the most significant differential in lap times. This season comparison highlights how wet weather conditions in 2023 led to significantly slower lap times compared to the dry conditions in 2024, as the on-track positions for Lando Norris between the two seasons are drastically far from each other.

It is important to note that there is some variance in lap times from year to year due to car and driver development, but keeping the driver (and constructor) the same seeks to mitigate some of the variance. We will see more examples later in this section about how qualifying lap times vary between teammates and constructors.

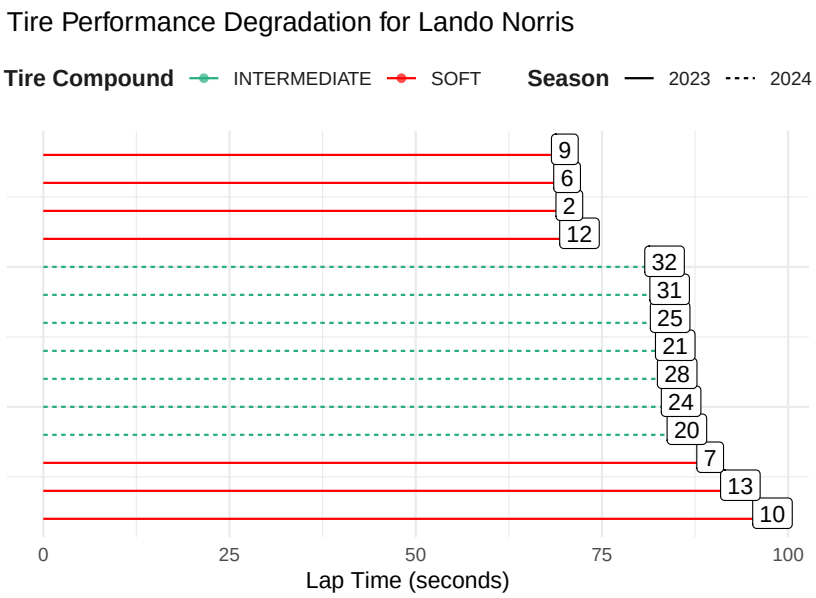

Figure 3: Plot of the seven fastest lap times and tire degradation across the two seasons.

The plot above shows the seven fastest times from Lando Norris during the 2023 and 2024 qualifying sessions. We can see that the laps from 2024 were driven on intermediate tires, which are one of the two types of tires used in wet conditions. Unlike the soft, medium, or hard tires, intermediates and wet tires have additional tread on them to help displace water that sits on the track. While this makes the cars safer to drive in wet conditions, it makes them considerably slower.

Table 4: Qualifying results for Lando Norris.

driver_code	season	constructor_id	lap_time	lap_number	compound	grid
NOR	2024	mclaren	83.405	32	INTERMEDIATE	1
NOR	2023	mclaren	70.021	9	SOFT	6

Figure 4: Snapshot of a late point in the lap where cars are very far apart (about 69 seconds in).

In this plot, we see that the positioning of Norris between the two seasons is vastly different. In the dry conditions of 2023, Norris is almost completed with his lap, while in 2024 he still has three turns to go. Wet versus dry conditions typically cause the largest gaps in qualifying due to how difficult wet conditions are to drive in.

2024 Monaco Grand Prix

Monaco is likely the most famous Formula 1 race that still exists on the calendar today, especially as many of its historic counterparts have been removed due to safety concerns based on how old the tracks are. Since it was added to the calendar, the track layout through Monte

Carlo's streets has remained unchanged. It is the only track that sits below the FIA's minimum length due to its history (and the inability to make it any longer). What has changed since 1950, when the race started being on the calendar annually, is the size of Formula 1 cars. Alongside its history, Monaco is not infamous for being one of the least interesting F1 races because it is almost impossible for cars to overtake each other on track. A driver's Saturday qualifying result is even more critical, as "Monaco is won on Saturday" holds accurate in most seasons because the driver on pole (fastest qualifying) tends to win the race.

This case study compares Oscar Piastri and Charles Leclerc's fastest qualifying laps from 2024. The two drivers, from different constructors, both nearing the front of the constructor's championship, can show how important it is to find every last thousandth of pace in a lap.

With a first-lap crash and a safety car allowing drivers to change tires immediately without fear of losing strategy to another team, the 2024 Monaco Grand Prix essentially consisted of every driver going around in circles for the duration of the race. The finishing order was almost identical to the starting order (barring a few DNFs). While not so interesting to watch live, it highlights how important qualifying is to a driver's success, especially on particular tracks.

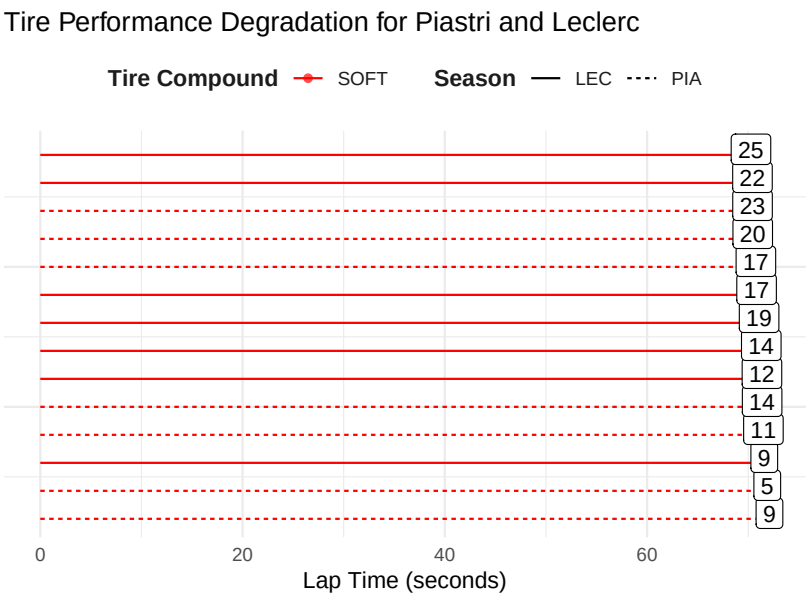

Figure 5: Plot of the seven fastest lap times and tire degredation across the two drivers

The plot above shows the seven fastest times from Piastri and Leclerc during the 2024 Monaco qualifying session. Unlike the previous example, all of the fastest lap times are incredibly close to one another, highlighting that top-performing cars and drivers can have the smallest of margins between them. Though, it's important to note that Leclerc had two laps faster than Piastri's fastest, meaning that even without his final lap, Leclerc would've remained on pole. We can also see from this plot that because both drivers had lap numbers in the 20s, its likely they both made it through to Q3.

Table 5: Qualifying results for Piastri and Leclerc at the 2024 Monaco Grand Prix

driver_name	driver_code	constructor_id	lap_time	lap_number	compound	grid
Charles Leclerc	LEC	ferrari	70.270	25	SOFT	1

driver_name	driver_code	constructor_id	lap_time	lap_number	compound	grid
Oscar Piastri	PIA	mclaren	70.424	23	SOFT	2

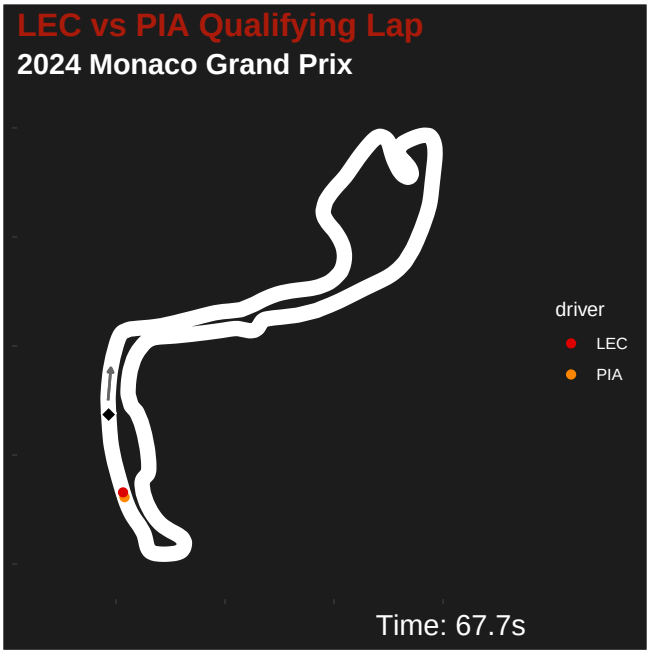

Figure 6: Snapshot of a late point in the lap (about 67 seconds in).

We can see from the table and the plot that Leclerc and Piastri were incredibly close, even nearing the end of the lap, with only two tenths of a second separating them as they crossed the line. While not the smallest margin for pole in F1 history (that goes to identical times down to the thousandth), it is still an incredibly small margin. But, like most time-based sports, being close just isn't good enough, and because of those fractions of a second, it was almost impossible for Piastri to win the race on Sunday.

2024 Canadian Grand Prix

Finally, we will observe two drivers from the reigning Constructors' Champions for the following case study. To provide some context, Max Verstappen, Sergio Pérez, and Red Bull Racing were coming off one of the most dominant seasons in F1 history, with Verstappen winning the WDC by 290 points (575 points to teammate Pérez's 285 in second) and Red Bull winning the WCC by over 400. If you took away all of Pérez's points from Red Bull, they still would have won by 166 points. After that, everyone entered the 2024 season expecting the same thing. Unfortunately for Red Bull, and fortunately for anyone who likes to watch more than one team win, Red Bull did not dominate the 2024 season, partially because of the gap in teammate performance. Max Verstappen and Sergio Pérez are qualifying on opposite ends of the grid in the same car.

The 2024 Canadian Grand Prix is one such example. In equal machinery, Verstappen consistently extracts more pace from the car than his teammate. While Max was fighting for pole position in Q3, Pérez was eliminated in Q1. This case study highlights how even with the same car, the pace differential can be huge depending on the driver's ability.

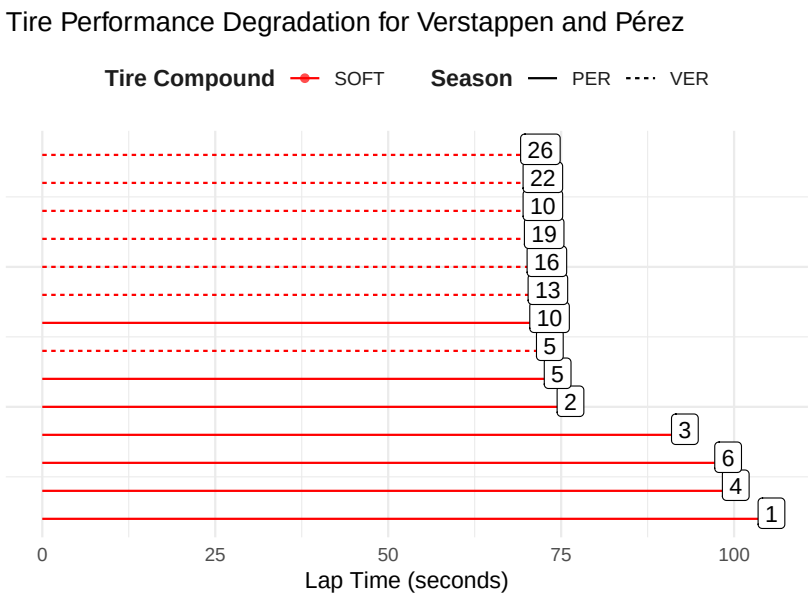

Figure 7: Plot of the seven fastest lap times and tire degredation across the two drivers

The plot above shows the seven fastest times from Verstappen and Pérez during the 2024 Canada qualifying session. We can see a much larger difference in the fastest laps of the two drivers here, with Verstappen having six laps faster than Pérez's fastest lap. This is a much larger gap than the previous example. We can also see that Pérez's fastest lap came on lap number 10, indicating he was eliminated much earlier.

Table 6: Qualifying results for Verstappen and Perez at the 2024 Canadian Grand Prix

driver_name	driver_code	constructor_id	lap_time	lap_number	compound	grid
Max Verstappen	VER	red_bull	72.000	26	SOFT	2
Sergio Pérez	PER	red_bull	73.326	10	SOFT	16

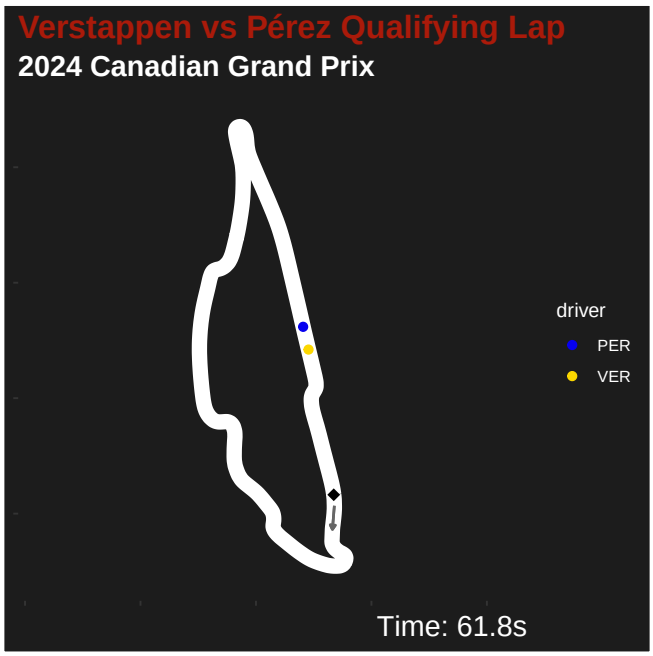

Figure 8: Snapshot of a late point in the lap (about 62 seconds in).

From the results summary and the plot, we see a separation that is a middle ground between the two previous case studies. While not as drastic as the weather separation, the difference in lap times is considerably larger than the one we saw in Monaco. A difference of 1.2 seconds between teammates though is an incredibly large gap, especially as Verstappen is continuing to lead the driver's championship, meaning that the car is likely not the issue.

Conclusion

This project highlights how qualifying laps provide an interesting platform to experiment with performance analysis in F1. The influence of weather, the varying importance of qualifying

depending on track features, and differentials in driver performance are only some of the many case studies possible to examine.

Future work would include allowing users to customize what lap number they would like to animate, as there are interesting things to be analyzed from laps that are not the driver's fastest lap. For one, if there were rain at the end of a session in Q3, we would be unable to visualize what the pole lap would look like because it would likely not be the fastest. There are also the rare instances we mentioned earlier where hazards like a red or yellow flag force a driver to slow down on a flying lap. In this case, it would be ideal to observe how close the pace was up until the flags came out.

Another feature to add would be the ability to pause, play, and speed up or slow down the animation, depending on what users are trying to observe from the plot. If someone were interested in just one corner, it would not be efficient or pleasant to stare at the circles just to catch the millisecond they're at the apex.

Data plays a crucial role in Formula 1, and visual analytics can offer fans a deeper understanding of qualifying performance. Interactive tools like Shiny help make F1 data more accessible, insightful, and enjoyable for the average viewer.

Acknowledgements

I would like to thank Santiago Casanova and Philip Bulsink for their work and maintenance on the `f1dataR` package. This project wouldn't have been possible without the ability to easily

retrieve data from various sources across the web and the helper functions for plotting and adjusting track ratios to look more realistic. Having the package update as races and sessions happen is no easy feat, and immediately having access to the data within days of the session was invaluable to doing up-to-date exploration during the 2024 season.

I would also like to thank my advisor, Dr. Matt Higham, for his guidance and support throughout the duration of this project. It wouldn't have been possible without his knowledge and expertise. Hopefully, this project has convinced you to watch a full race soon!

Appendix

R functions

The R functions created and written for `f1animateR` can be found in the GitHub repository at https://github.com/norah-kuduk/StatsSYE/tree/master/R.

Shiny App Code

Code for the Shiny Application. The app can be found at https://stlawu.shinyapps.io/f1-qualifying-app/.

```
library(tidyverse)
library(shiny)
library(dplyr)
library(ggplot2)
library(stringr)
```

```r
library(f1dataR)
library(f1animateR)
library(knitr)
library(kableExtra)
library(gganimate)
library(reactable)
library(ggrepel)
library(tidyr)
library(bslib)
library(reactablefmtr)
library(shinycssloaders)
library(showtext)
library(gifski)

## setup_fastf1()
## devtools::install_github("https://github.com/norah-kuduk/StatsSYE")
## use_virtualenv("f1dataR_env")

# Create and activate environment if it doesn't exist
if (!virtualenv_exists("f1env")) {
  virtualenv_create("f1env")
  virtualenv_install("f1env", packages = c("fastf1"))
}
use_virtualenv("f1env", required = TRUE)

# Set a cache directory that works in ShinyApps.io
cache_dir <- file.path(tempdir(), "fastf1_cache")
dir.create(cache_dir, showWarnings = FALSE, recursive = TRUE)

# Enable FastF1 cache via reticulate
fastf1 <- import("fastf1", delay_load = TRUE)
fastf1$Cache$enable_cache(cache_dir)

cat("FastF1 cache directory set to:", cache_dir, "\n")

font <- font_link("Titillium Web",
                  href="https://fonts.googleapis.com/css2?family=Patrick+Hand&family=Titilliu

# custom theme
f1_theme <- bs_theme(
  bootswatch = "darkly",  # dark theme base
  bg = "#1C1C1C",          # background color
  fg = "#ffffff",           # default text color
  primary = "#aa1a0a",    # red highlight color (matches plot title)
  secondary = "#aa1a0b",    # secondary text color
  base_font = font,
  code_font = font_google("Fira Code"),
```

```
  "nav-link-hover-color" = "#aa1a0a",  # hover color for tabs
  "nav-link-active-color" = "#aa1a0a",   # active tab color
  "link-color" = "#aa1a0a" # link color
)

options(spinner.image = "f1_loading.gif",
        spinner.image.width = "150px",
        spinner.image.height = "150px")

font_add_google("Titillium Web", "titillium")
showtext_auto()

ui <- fluidPage(
  theme = f1_theme,

  titlePanel(div(
    "F1 Qualifying Analysis",
    actionButton("restart_app", "Restart App", class = "btn-danger", style = "float: right; m
  )),

  tabsetPanel(
    #### Tab 1: Instructions and Important Information ####
    tabPanel("Instructions",
             fluidPage(
               titlePanel("How to Use This App"),

               h3("Overview"),
               p("This app allows users to analyze F1 qualifying performance by comparing tel

               h3("Different Comparison Modes"),
               tags$ul(
                 tags$li(strong("Same Session:"), " Compare two drivers in the same qualifyi
                 tags$li(strong("Same Driver, Different Years:"), " Analyze how a single driv
               ),

               h3("The Data Exploration Page"),
               tags$ul(
                 tags$li("Table of all the lap times across drivers/seasons."),
                 tags$li("Tire degradation analysis."),
                 tags$li("Final qualifying and race position tables."),
               ),

               h3("How to Use Animation Feature"),
               tags$ol(
                 tags$li("Choose whether or not to carry over data from exploration page."),
                 tags$li("Select the desired comparison mode."),
                 tags$li("Choose the season(s) and round you want to analyze."),
```

```
            tags$li("Select the drivers (or driver, if comparing across seasons)."),
            tags$li("Click 'Animate Qualifying' to visualize lap performance."),
            tags$li("Use 'Clear Animation' to reset.")
          ),

          h3("Important Notes"),
          tags$ul(
            tags$li("Please be patient, there is a lot of data that gets loaded when thi
            tags$li("Animations loop through each driver or season's fastest lap from th
            tags$li("If you want to change any of the inputs for your animation, hit the
          ),

          h3("References"),
          p("This app utilizes the `f1dataR` package for fetching Formula 1 data, includ
          tags$a(href="https://cran.r-project.org/package=f1dataR", "f1dataR on CRAN", t
        )

),

#### Tab 2: Data Exploration ####
tabPanel("Data Exploration",
         sidebarLayout(
           sidebarPanel(
             radioButtons("compare_mode", "Comparison Mode:",
                          choices = c("Same Session - Two Drivers" = "same_session",
                                      "Same Driver - Two Seasons" = "diff_seasons")),

             conditionalPanel(
               condition = "input.compare_mode == 'same_session'",
               selectInput("season_1", "Select Season:", choices = 2018:2024, selected =
               uiOutput("round_select"),
               uiOutput("driver_select_1"),
               uiOutput("driver_select_2"),
             ),

             conditionalPanel(
               condition = "input.compare_mode == 'diff_seasons'",
               uiOutput("driver_same"),
               uiOutput("season_a"),
               uiOutput("season_b"),
               uiOutput("round_overlap")
             )
           ),

           mainPanel(
             withSpinner(
               reactableOutput("quali_comparison")
```

```
                ),
                withSpinner(
                  tableOutput("quali_race_table")
                ),
                withSpinner(
                  plotOutput("tire_degradation_plot")
                ),
              )
            )),

    #### Tab 3: Animation ####
    tabPanel("Qualifying Animation",
             sidebarLayout(
               sidebarPanel(
                 checkboxInput("carry_over", "Use Selections from Data Exploration", value =

                 radioButtons("anim_compare_mode", "Comparison Mode:",
                              choices = c("Same Session - Two Drivers" = "same_session",
                                          "Same Driver - Two Seasons" = "diff_seasons")),

                 conditionalPanel(
                   condition = "input.anim_compare_mode == 'same_session'",
                   selectInput("anim_season_1", "Select Season:", choices = 2018:2024, selec
                   uiOutput("anim_round_select"),
                   uiOutput("anim_driver_select_1"),
                   uiOutput("anim_driver_select_2"),
                   sliderInput("smoothing_factor", "Smoothing Factor:",
                               min = 0, max = 60,
                               value = 16),
                   actionButton("animate_btn", "Animate Qualifying"),
                   actionButton("stop_btn", "Clear Animation")

                 ),

                 conditionalPanel(
                   condition = "input.anim_compare_mode == 'diff_seasons'",
                   uiOutput("anim_driver_same"),
                   uiOutput("anim_season_a"),
                   uiOutput("anim_season_b"),
                   uiOutput("anim_round_overlap"),
                   sliderInput("smoothing_factor", "Smoothing Factor:",
                               min = 0, max = 60,
                               value = 16),
                   actionButton("animate_btn", "Animate Qualifying"),
                   actionButton("stop_btn", "Clear Animation")
                 )
               ),
```

```
              mainPanel(
                withSpinner(
                  plotOutput("animation_output")
                )
              )
            ))
  )
)

server <- function(input, output, session) {
  #### Functions ####

  # fetch qualifying data
  fetch_quali_data <- function(season, round) {
    tryCatch(
      {
        results <- load_session_laps(season, round, "Q")
        if (is.null(results) || nrow(results) == 0) return(NULL)
        results
      },
      error = function(e) NULL
    )
  }

  fetch_quali_data_seasons <- function(season_a, season_b, round) {
    tryCatch(
      {
        results_a <- load_session_laps(season_a, round, "Q") |> mutate(season = season_a)
        results_b <- load_session_laps(season_b, round, "Q") |> mutate(season = season_b)
        if (is.null(results_a) || is.null(results_b) || nrow(results_a) == 0 || nrow(results_
          return(NULL)
        } else {
          results <- bind_rows(results_a, results_b)
          return(results)
        }

      },
      error = function(e) NULL
    )
  }
  #### Reactive ####

  observeEvent(input$restart_app, {
    session$reload()
  })
```

```
# fetch schedule data for a given season
schedule_data <- reactive({
  req(input$season_1)
  load_schedule(input$season_1)
})

seasons_schedule_data <- reactive({
  req(input$season_a, input$season_b)
  schedule_a <- load_schedule(input$season_a) |> mutate(season = input$season_a)
  schedule_b <- load_schedule(input$season_b) |> mutate(season = input$season_b)
  bind_rows(schedule_a, schedule_b)
})

# fetch qualifying drivers/teams
quali_data_1 <- reactive({
  # browser()
  req(input$season_1, input$round_1)
  get_session_drivers_and_teams(input$season_1, input$round_1, "Q") |>
    mutate(driver_code = abbreviation) |>
    select(-abbreviation)
})

quali_data_diff_seasons <- reactive({
  req(input$season_a, input$season_b, input$round_constant, input$driver_same)
  year_1 <- get_session_drivers_and_teams(input$season_a, input$round_constant, "Q") |>
    mutate(driver_code = abbreviation) |>
    select(-abbreviation)

  year_2 <- get_session_drivers_and_teams(input$season_b, input$round_constant, "Q") |>
    mutate(driver_code = abbreviation) |>
    select(-abbreviation)

  bind_rows(year_1, year_2)
})

# get quali and race results for comparison
quali_race_results <- reactive({
  req(input$season_1, input$round_1, input$driver_1, input$driver_2)
  schedule <- schedule_data()
  round_num <- schedule |>
    filter(race_name == input$round_1) |>
    mutate(round = as.integer(round)) |>
    pull(round)

  full_results2 <- load_results(input$season_1, round_num)
  driver_data <- load_drivers(input$season_1)
```

```
  inner_join(full_results2, driver_data, by = "driver_id")
})

# get quali and race results for comparison (diff seasons)
seasons_quali_race_results <- reactive({
  req(input$season_a, input$season_b, input$round_constant)

  schedule <- seasons_schedule_data()

  rounds <- schedule |>
    filter(race_name == input$round_constant) |>
    mutate(round = as.integer(round))

  round_num_a <- rounds |>
    filter(season == input$season_a) |>
    pull(round)

  round_num_b <- rounds |>
    filter(season == input$season_b) |>
    pull(round)

  full_results_a <- load_results(input$season_a, round_num_a) |> mutate(season = input$seas
  full_results_b <- load_results(input$season_b, round_num_b) |> mutate(season = input$seas

  full_results <- bind_rows(full_results_a, full_results_b)

  driver_data <- load_drivers(input$season_a)

  inner_join(full_results, driver_data, by = "driver_id")
})


# get all drivers from 2018 to 2024 that drove in more than 1 season
all_drivers <- reactive({
  seasons <- 2018:2024

  # Fetch and combine drivers from all seasons
  driver_data <- bind_rows(lapply(seasons, function(season) {
    load_drivers(season = season) |>
      mutate(season = season)
  }))

  driver_data |>
    unite("Driver", c("given_name", "family_name"), sep = " ") |>
    group_by(code) |>
    mutate(n_seasons = n()) |>
    filter(n_seasons > 1)
```

```
})

# get all seasons that a driver participated in (2018 to 2024)
valid_seasons <- reactive({
  req(input$driver_same)

  driver_data <- all_drivers()

  driver_data |> filter(code == input$driver_same) |> pull(season)
})

# find races that only existed in those two seasons (has to be by name)
round_overlap <- reactive({
  req(input$season_a, input$season_b)

  season_a <- input$season_a
  season_b <- input$season_b

  seasons <- c(season_a, season_b)

  race_data <- bind_rows(lapply(seasons, function(season) {
    load_schedule(season = season)
  }))

  race_data |> group_by(race_name) |>
    mutate(num_races = n()) |>
    filter(num_races > 1) |>
    distinct(race_name)
})

#### Comparison Panel ####

###### SAME SESSION LOADING ######
output$round_select <- renderUI({
  data <- schedule_data()
  data <- data |> mutate(round = as.integer(round))

  selectInput("round_1", "Select Round: ", choices = data$race_name)
})

# populate driver selection for same session comparison
output$driver_select_1 <- renderUI({
  data <- quali_data_1()
  driver_choices <- setNames(data$driver_code, data$name)
  selectInput("driver_1", "Select Driver 1: ", choices = driver_choices)
})
```

```
output$driver_select_2 <- renderUI({
  data <- quali_data_1()
  driver_choices <- setNames(data$driver_code, data$name)
  selectInput("driver_2", "Select Driver 2: ", choices = driver_choices)
})

###### SAME DRIVER LOADING ######

output$driver_same <- renderUI({
  driver_data <- all_drivers()
  driver_choices <- setNames(driver_data$code, driver_data$Driver)
  selectInput("driver_same", "Select Driver: ", choices = driver_choices, selected = "Lewis
})

output$season_a <- renderUI({
  data <- valid_seasons()

  selectInput("season_a", "Select First Season: ", choices = data)
})

output$season_b <- renderUI({
  data <- valid_seasons()

  selectInput("season_b", "Select Second Season: ", choices = data)
})

output$round_overlap <- renderUI({
  data <- round_overlap()

  selectInput("round_constant", "Select Round: ", choices = data$race_name)
})

##### Output Tables #####

# display comparison table (two drivers)
output$quali_comparison <- renderReactable({
  if (input$compare_mode == "same_session") {
    req(input$driver_1, input$driver_2)

    data <- quali_data_1()
    req(data)

    comparison <- data |>
      filter(driver_code == input$driver_1 | driver_code == input$driver_2)

    quali_laps <- fetch_quali_data(input$season_1, input$round_1) |>
      filter(driver == input$driver_1 | driver == input$driver_2) |>
```

```
      filter(lap_time > 0) |>
      select(driver, lap_time, lap_number, compound) |>
      arrange(lap_time)

    reactable(
      quali_laps,
      striped = TRUE,
      highlight = TRUE,
      bordered = TRUE,
      defaultPageSize = 15,
      theme = f1_reactable(),
      columns = list(
        driver = colDef(name = "Driver", align = "left"),
        lap_time = colDef(name = "Lap Time", align = "center"),
        lap_number = colDef(name = "Lap Number", align = "center"),
        compound = colDef(name = "Tire Compound", align = "center")
      )
    )
  } else {
    req(input$driver_same, input$season_a, input$season_b, input$round_constant)

    data <- quali_data_diff_seasons()

    comparison <- data |> filter(driver_code == input$driver_same)

    quali_laps_a <- fetch_quali_data(input$season_a, input$round_constant) |>
      filter(driver == input$driver_same) |>
      filter(lap_time > 0) |>
      select(driver, lap_time, lap_number, compound) |>
      mutate(season = input$season_a)

    quali_laps_b <- fetch_quali_data(input$season_b, input$round_constant) |>
      filter(driver == input$driver_same) |>
      filter(lap_time > 0) |>
      select(driver, lap_time, lap_number, compound) |>
      mutate(season = input$season_b)

    data <- bind_rows(quali_laps_a, quali_laps_b) |>
      unique() |>
      arrange(lap_time)

    reactable(
      data,
      striped = TRUE,
      highlight = TRUE,
      bordered = TRUE,
      defaultPageSize = 15,
```

```
      theme = f1_reactable(),
      columns = list(
        driver = colDef(name = "Driver", align = "left"),
        season = colDef(name = "Season", align = "center"),
        lap_time = colDef(name = "Lap Time", align = "center"),
        lap_number = colDef(name = "Lap Number", align = "center"),
        compound = colDef(name = "Tire Compound", align = "center")
      )
    )
  }
})

output$tire_degradation_plot <- renderPlot({
  if (input$compare_mode == "same_session") {

    req(input$driver_1, input$driver_2, input$season_1)

    quali_laps <- fetch_quali_data(input$season_1, input$round_1) |>
      filter(driver == input$driver_1 | driver == input$driver_2) |>
      filter(lap_time > 0) |>
      select(driver, lap_time, lap_number, compound) |>
      arrange(desc(lap_time)) |>
      mutate(index = row_number())

    ggplot(quali_laps, aes(x = index, y = lap_time)) +
      geom_segment(aes(x = index, xend = index, y = 0, yend = lap_time, color = compound,
      geom_point(size = 4, aes(color = compound), alpha = 0.7) +
      geom_label(aes(label = lap_number), nudge_x = 0.2) +
      scale_color_manual(values = c("SOFT" = "red", "MEDIUM" = "gold", "HARD" = "white", "
                                    "HYPERSOFT" = "lightpink", "ULTRASOFT" = "purple", "SU
      coord_flip() +
      labs(
        title = "Tire Performance Degradation",
        y = "Lap Time (seconds)",
        color = "Tire Compound",
        linetype = "Driver"
      ) +
      theme_track() +
      theme(legend.position = "top", legend.text = element_text(color = "#1C1C1C", size =
            legend.title = element_text(color = "#1C1C1C", face = "bold", size = 16),
            legend.box.background = element_rect(color="white"),
            axis.title.y = element_blank(), axis.text.y = element_blank(),
            axis.ticks.y = element_blank())

  } else {
    req(input$season_a, input$season_b, input$round_constant, input$driver_same)
```

```
    quali_laps <- fetch_quali_data_seasons(input$season_a, input$season_b, input$round_cons
      filter(driver == input$driver_same) |>
      filter(lap_time > 0) |>
      select(driver, season, lap_time, lap_number, compound) |>
      arrange(desc(lap_time)) |>
      unique() |>
      mutate(index = row_number())

    ggplot(quali_laps, aes(x = index, y = lap_time)) +
      geom_segment(aes(x = index, xend = index, y = 0, yend = lap_time, color = compound, l
      geom_point(size = 4, aes(color = compound), alpha = 0.7) +
      geom_label(aes(label = lap_number), nudge_x = 0.2) +
      scale_color_manual(values = c("SOFT" = "red", "MEDIUM" = "gold", "HARD" = "white", "W
                                    "HYPERSOFT" = "lightpink", "ULTRASOFT" = "purple", "SUP
      coord_flip() +
      labs(
        title = "Tire Performance Degradation",
        y = "Lap Time (seconds)",
        color = "Tire Compound",
        linetype = "Season"
      ) +
      theme_track() +
      theme(legend.position = "top", legend.text = element_text(color = "#1C1C1C", size = 1
            legend.title = element_text(color = "#1C1C1C", face = "bold", size = 16),
            legend.box.background = element_rect(color="white"),
            axis.title.y = element_blank(), axis.text.y = element_blank(),
            axis.ticks.y = element_blank())
  }
}, height = 600, width = 600)

output$quali_race_table <- renderTable({
  if (input$compare_mode == "same_session") {

    # browser()
    data <- quali_race_results()

    data <- data |> filter(code == input$driver_1 | code == input$driver_2) |>
      unite("driver", c(given_name, family_name), sep = " ") |>
      rename("Driver" = driver, "Team" = constructor_id, "Quali Position" = grid, "Race Pos
      select(Driver, Team, `Quali Position`, `Race Position`)

    kable(data, format = "html") |> kable_styling("striped", full_width = FALSE)

  } else {

    data <- seasons_quali_race_results()
```

```
    data <- data |> filter(code == input$driver_same) |>
      unite("driver", c(given_name, family_name), sep = " ") |>
      rename("Season" = season, "Driver" = driver, "Team" = constructor_id, "Quali Positio
      select(Season, Driver, Team, `Quali Position`, `Race Position`) |>
      unique()

    kable(data, format = "html") |> kable_styling("striped", full_width = FALSE)

  }
}, sanitize.text.function = function(x) x)


#### Animation Panel ####

# Observe the 'carry_over' check box and update inputs when it's checked
observe({
  req(input$carry_over)  # Only execute when carry_over is checked

  # Ensure inputs exist before updating
  req(input$compare_mode)

  # If comparison mode is "same_session", update corresponding inputs
  if (input$compare_mode == "same_session") {
    req(input$season_1, input$driver_1, input$driver_2, input$round_1)

    updateRadioButtons(session, "anim_compare_mode", selected = input$compare_mode)
    updateSelectInput(session, "anim_season_1", selected = input$season_1)
  } else {
    req(input$season_a, input$season_b, input$driver_same, input$round_constant)

    updateRadioButtons(session, "anim_compare_mode", selected = input$compare_mode)
  }
})

##### SAME SESSION #####

# Define reactive variables for the schedule and qualifying data
anim_schedule_data <- reactive({
  req(input$anim_season_1)
  load_schedule(input$anim_season_1)
})

anim_quali_data_1 <- reactive({
  req(input$anim_season_1, input$round_1)
  get_session_drivers_and_teams(input$anim_season_1, input$round_1, "Q") |>
    mutate(driver_code = abbreviation) |>
    select(-abbreviation)
```

```
})

output$anim_round_select <- renderUI({
  data <- anim_schedule_data()

  if (input$carry_over) {
    # Use selection from Data Exploration panel
    selectInput("round_1", "Select Round:",
                choices = schedule_data()$race_name,
                selected = input$round_1)  # Set selected to match the carry_over data
  } else {
    # Default behavior: use current data
    selectInput("round_1", "Select Round:",
                choices = data$race_name)
  }
})

output$anim_driver_select_1 <- renderUI({
  data <- anim_quali_data_1()

  if (input$carry_over) {
    selectInput("driver_1", "Select Driver 1:",
                choices = setNames(data$driver_code, data$name),
                selected = input$driver_1)  # Set selected to carry over value
  } else {
    selectInput("driver_1", "Select Driver 1:",
                choices = setNames(data$driver_code, data$name))
  }
})

# Update the driver selection UI reactively for Driver 2
output$anim_driver_select_2 <- renderUI({
  data <- anim_quali_data_1()

  if (input$carry_over) {
    selectInput("driver_2", "Select Driver 2:",
                choices = setNames(data$driver_code, data$name),
                selected = input$driver_2)  # Set selected to carry over value
  } else {
    selectInput("driver_2", "Select Driver 2:",
                choices = setNames(data$driver_code, data$name))
  }
})

##### SAME DRIVER #####

output$anim_driver_same <- renderUI({
```

```
  driver_data <- all_drivers()
  driver_choices <- setNames(driver_data$code, driver_data$Driver)

  selected_driver <- isolate(input$driver_same)

  selectInput("driver_same", "Select Driver:",
              choices = driver_choices,
              selected = if (input$carry_over) selected_driver else NULL)
})

output$anim_season_a <- renderUI({
  data <- valid_seasons()

  selected_season_a <- isolate(input$season_a)

  selectInput("season_a", "Select First Season:",
              choices = data,
              selected = if (input$carry_over) selected_season_a else NULL)
})

output$anim_season_b <- renderUI({
  data <- valid_seasons()

  selected_season_b <- isolate(input$season_b)

  selectInput("season_b", "Select Second Season: ",
              choices = data,
              selected = if (input$carry_over) selected_season_b else NULL)
})

output$anim_round_overlap <- renderUI({
  req(input$season_a, input$season_b)

  data <- round_overlap()

  selected_round <- isolate(input$round_constant)

  selectInput("round_constant", "Select Round: ",
              choices = data$race_name,
              selected = if (input$carry_over) selected_round else NULL)
})


# reactive variable to store telemetry data
plot_data <- reactiveVal(NULL)
track_data <- reactiveVal(NULL)
animation_running <- reactiveVal(FALSE)
```

```
##### Output #####

# animate button clicked
observeEvent(input$animate_btn, {
  showSpinner()

  if (input$anim_compare_mode == "same_session") {
    animation_running(TRUE)

    # fetch telemetry data
    req(input$driver_1, input$driver_2, input$season_1, input$round_1)

    driver1_data <- get_quali_telemetry(laps = "fastest", season = input$season_1, round =
    driver2_data <- get_quali_telemetry(laps = "fastest", season = input$season_1, round =

    # smooth the track data
    track_data(smooth_track(driver1_data))

    # assign driver labels
    driver1_data <- driver1_data |> mutate(driver = input$driver_1)
    driver2_data <- driver2_data |> mutate(driver = input$driver_2)

    # combine telemetry data
    combined_data <- bind_data_driver(driver1_data, driver2_data, drivers = c(input$driver_

    # smooth speed values

    smooth_speed(plot_data = combined_data, grouping = "driver", input$smoothing_factor)

    # only proceed if animation is still running
    if (animation_running()) {
      plot_data(combined_data)
    }
  } else {
    animation_running(TRUE)

    # fetch telemetry data
    req(input$driver_same, input$season_a, input$season_b, input$round_constant)

    driver_seasona_data <- get_quali_telemetry(laps = "fastest", season = input$season_a, r
    driver_seasonb_data <- get_quali_telemetry(laps = "fastest", season = input$season_b, r

    # smooth the track data
    track_data(smooth_track(driver_seasona_data))

    # assign season labels
```

```
    driver_seasona_data <- driver_seasona_data |> mutate(season = input$season_a)
    driver_seasonb_data <- driver_seasonb_data |> mutate(season = input$season_b)

    # combine telemetry data
    combined_data <- bind_rows(driver_seasona_data, driver_seasonb_data) |> mutate(season =

    smooth_speed(plot_data = combined_data, grouping = "season", input$smoothing_factor)

    # only proceed if animation is still running
    if (animation_running()) {
      plot_data(combined_data)
    }
  }
  hideSpinner()

})

driver_colors <- reactive({
  req(input$driver_1, input$driver_2, input$season_1, input$round_1)

  driver1_team <- tolower(get_team_by_driver(
    input$driver_1, season = input$season_1, round = input$round_1, short = TRUE
  ))
  driver2_team <- tolower(get_team_by_driver(
    input$driver_2, season = input$season_1, round = input$round_1, short = TRUE
  ))

  if (driver1_team == driver2_team) {
    driver1_color <- constructor_color(driver1_team)
    driver2_color <- "black"
  } else {
    driver1_color <- constructor_color(driver1_team)
    driver2_color <- constructor_color(driver2_team)
  }

  setNames(c(driver1_color, driver2_color), c(input$driver_1, input$driver_2))
})

observeEvent(input$stop_btn, {
  animation_running(FALSE)  # stop animation
  plot_data(NULL)  # clear data to stop rendering
})

season_colors <- reactive({
  req(input$season_a, input$season_b, input$driver_same, input$round_constant)

  seasona_team <- tolower(get_team_by_driver(
```

```
    input$driver_same, season = input$season_a, round = input$round_constant, short = TRUE
  ))
  seasonb_team <- tolower(get_team_by_driver(
    input$driver_same, season = input$season_b, round = input$round_constant, short = TRUE
  ))

  if (seasona_team == seasonb_team) {
    seasona_color <- constructor_color(seasona_team)
    seasonb_color <- "black"
  } else {
    seasona_color <- constructor_color(seasona_team)
    seasonb_color <- constructor_color(seasonb_team)
  }

  setNames(c(seasona_color, seasonb_color), c(input$season_a, input$season_b))
})

# generate animated plot
output$animation_output <- renderImage({
  showSpinner()
  req(input$anim_compare_mode)

  if (input$anim_compare_mode == "same_session") {
    req(plot_data(), track_data(), animation_running())  # Ensure animation is running

    t_df <- track_data()
    plot_data <- plot_data()

    plot_data <- smooth_speed(plot_data, "driver", input$smoothing_factor)

    start_coord <- t_df |> slice(1)

    driver_colors <- driver_colors()

    static_plot <- ggplot() +
      geom_path(data = t_df, aes(x = x2, y = y2, group = 1),
                linewidth = 8, color = "white") +
      geom_point(data = start_coord, aes(x = x2, y = y2),
                 color = "black", shape = 18, size = 4) +
      geom_point(data = plot_data(), aes(x = x, y = y, group = driver, color = driver),
                 size = 3) +
      scale_color_manual(values = driver_colors) +
      theme_track() +
      theme(plot.title = element_text(family = "titillium", face = "bold", size = 20, colo
            plot.subtitle = element_text(face = "bold", size = 16, color = "white"),
            plot.caption = element_text(face = "plain", size = 16, color = "white")) +
      labs(title = paste(input$driver_1, "vs.", input$driver_2, "Qualifying Lap"),
```

```
            subtitle = paste(input$season_1, input$round_1),
            x = NULL, y = NULL)

    # apply track correction
    corrected_plot <- corrected_track(static_plot, plot_data())

    animated_plot <- corrected_plot +
      transition_reveal(along = time) +
      ease_aes('linear') +
      labs(
        caption = paste(
          "Time: {sprintf('%.3f', frame_along)} s\n",
          input$driver_1, " Speed: {plot_data$speed_smooth[which.min(abs(plot_data$time[plo
          input$driver_2," Speed: {plot_data$speed_smooth[which.min(abs(plot_data$time[plot
        )
      )

    outfile <- tempfile(fileext = ".gif")
    animation <- animate(animated_plot, width = 700, height = 700, fps = 10, renderer = gi
    anim_save(outfile, animation = animation)
  }

  if (input$anim_compare_mode == "diff_seasons") {
    req(plot_data(), track_data(), animation_running())  # Ensure animation is running

    t_df <- track_data()
    plot_data <- plot_data()

    plot_data <- smooth_speed(plot_data, "season", input$smoothing_factor)

    start_coord <- t_df |> slice(1)

    season_colors <- season_colors()

    static_plot <- ggplot() +
      geom_path(data = t_df, aes(x = x2, y = y2, group = 1),
                linewidth = 8, color = "white") +
      geom_point(data = start_coord, aes(x = x2, y = y2),
                 color = "black", shape = 18, size = 4) +
      geom_point(data = plot_data(), aes(x = x, y = y, group = season, color = season),
                 size = 3) +
      scale_color_manual(values = season_colors) +
      theme_track() +
      theme(plot.title = element_text(family = "titillium", face = "bold", size = 20, colo
            plot.subtitle = element_text(face = "bold", size = 16, color = "white"),
            plot.caption = element_text(face = "plain", size = 16, color = "white")) +
      labs(title = paste(input$season_a, "vs.", input$season_b, "Qualifying Lap"),
```

```
            subtitle = paste(input$driver_same, input$round_constant),
            x = NULL, y = NULL)

      # apply track correction
      corrected_plot <- corrected_track(static_plot, plot_data())

      animated_plot <- corrected_plot +
        transition_reveal(along = time) +
        ease_aes('linear') +
        labs(
          caption = paste(
            "Time: {sprintf('%.3f', frame_along)} s\n",
            input$season_a, " Speed: {plot_data$speed_smooth[which.min(abs(plot_data$time[plo
            input$season_b," Speed: {plot_data$speed_smooth[which.min(abs(plot_data$time[plot
          )
        )

      outfile <- tempfile(fileext = ".gif")
      animation <- animate(animated_plot, width = 700, height = 700, fps = 10, renderer = gi
      anim_save(outfile, animation = animation)
    }
    hideSpinner()

    # return a list containing the file name
    list(src = outfile, contentType = 'image/gif')

  }, deleteFile = TRUE)
}

shinyApp(ui, server)
```