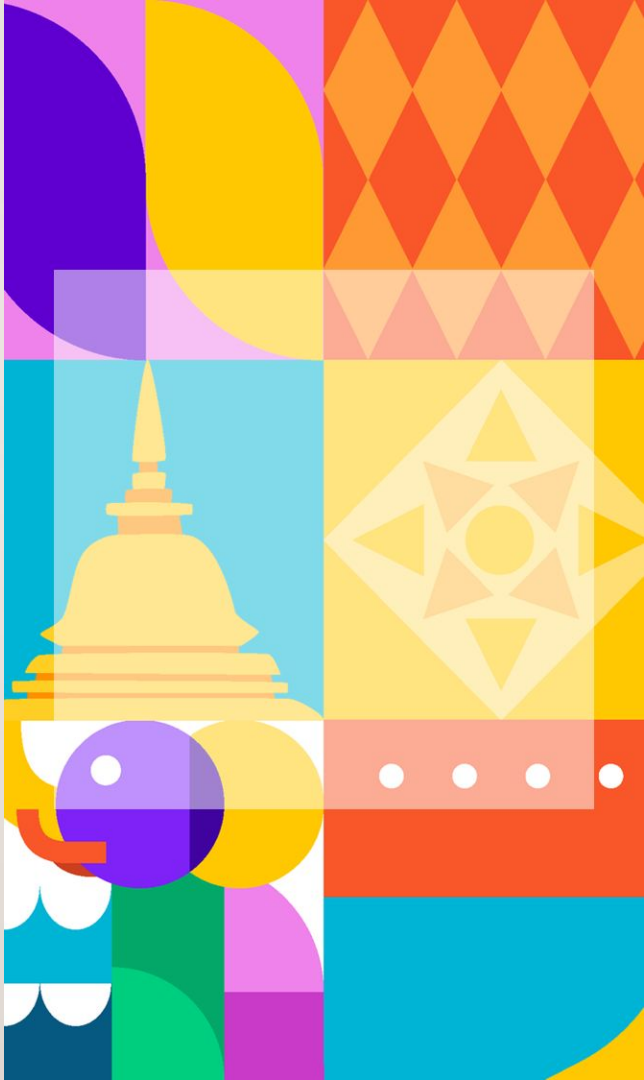


# Visualizing Formula 1 Performance based on F1 Telemetry Data

Dima M Dinama

Dec 15th. 2023 #PyConTH #PyConTH2023

# Introduction

**Dima Maharika Dinama**

- From **Bandung**, Indonesia, 26 years old
- Bachelor of Applied Science, **Mechatronics Engineering**, graduated at 2019
- Previously working as **Computer Vision Engineer** at SiRekap Project (2020), Train computer to read numbers from election form
- Currently working on **boring office stuff** at **Government Institution** (2021 - now)
- Committee of PyCon Indonesia 2019,
  **Lead Organizer** of **PyCon Indonesia** 2020, 2021, and 2023
- **Lightning Talks Speaker** at PyCon ID (2019), PyCon JP, PyCon IN, PyCon TR (2020)
- **Representative Speaker** at PyCon APAC Thailand (2021)

# Agenda

1. Formula 1 Introduction
2. Role of Data in Formula 1
3. FastF1 Plugins
4. Data Processing
5. Visualization Example

# Formula 1

**Formula 1 (F1)** is the **pinnacle of motorsport**, featuring the **world's fastest cars** and **most skilled drivers** competing in a series of **high-speed races**.

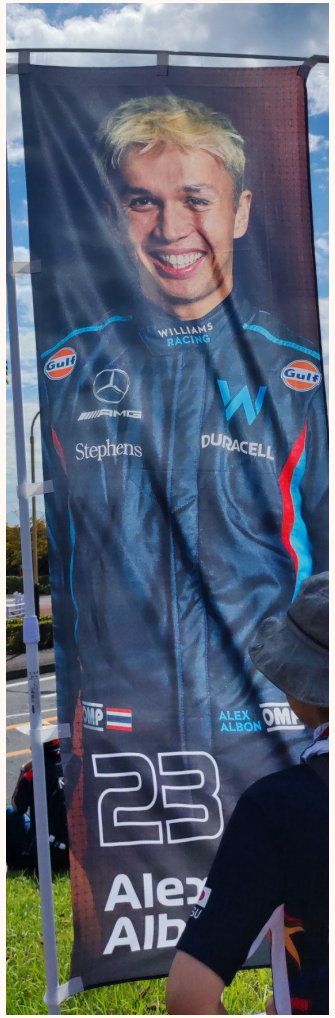
# What Makes Formula 1 Interesting

- **High-Speed Thrills**
  F1 cars racing at over 200 mph (320 km/h), very fast
- **Cutting-Edge Technology**
  F1 showcases the latest advancements in automotive technology
- **Most-Skilled Driver**
  The drivers in Formula 1 are among the most skilled athletes globally
- **Strategic Depth**
  Teams meticulously plan pit stops, tire choices, fuel management, and more
- **Innovation and Evolution**
  Formula 1 is always evolving. Pushing more in every race.
- **Live Experience**
  Watching Formula 1 races live is one of my best life experience

F1 Japanese GP 2022



F1 Japanese GP 2023



# Data and Analysis in Formula 1

- **Basis of Development**
  Car development start with data, and develop car to be faster as more data is acquired
- **Massive amount of Data**
  From brake to engine, F1 cars have hundreds of sensors that send thousand of data each second
- **Strategic-Response**
  Race can go right and wrong, data helps F1 teams choose the best strategy for the race
- **Depth-Analysis**
  Strategist look deep beyond, provided by vast amount of data to be analysed, and every data matters


# FastF1 Plugins

FastF1 is a **python package** for accessing and analyzing Formula 1 results, schedules, timing data and telemetry

- Access to **F1 timing data, telemetry, sessions results and more**
- All data is provided in the form of extended **Pandas DataFrames** to make working with the data easy while having powerful tools available
- Adds custom functions to the Pandas objects specifically to make working with F1 data **quick and simple**
- Integration with **Matplotlib** to facilitate **data visualization**
- Implements caching for all API requests to speed up your scripts

**Installation**

```
pip install fastf1
```

Source:
[Fast F1 Github Pages]

# Data from FastF1

| Topic | Data |
|---|---|
| Event Schedule | event names, countries, locations, dates, scheduled starting times,… (previous and current season including upcoming events) |
| Results | driver names, team names, finishing and grid positions, points, finishing status,… |
| Timing Data | sector times, lap times, pit stops, tyre data and much more |
| Track Status | flags, safety car |
| Session Status | started, finished, finalized |
| Race Control Messages | investigations, penalties, restart announcements,… |
| Telemetry | speed, rpm, gear, normalized track position, … |
| Track Markers | corner numbers, marshall sectors, marshall lights |
| Ergast API | all endpoints that are provided by Ergast |

Source: Fast F1 Docs

# Data from FastF1

## Lap Timing Data

- **Time** (pandas.Timedelta): Session time when the lap time was set (end of lap)
- **Driver** (string): Three letter driver identifier
- **DriverNumber** (str): Driver number
- **LapTime** (pandas.Timedelta): Recorded lap time. Officially deleted lap times will *not* be deleted here. Deleting laps is currently not supported.
- **LapNumber** (float): Recorded lap number
- **Stint** (float): Stint number
- **PitOutTime** (pandas.Timedelta): Session time when car exited the pit
- **PitInTime** (pandas.Timedelta): Session time when car entered the pit
- **Sector1Time** (pandas.Timedelta): Sector 1 recorded time
- **Sector2Time** (pandas.Timedelta): Sector 2 recorded time
- **Sector3Time** (pandas.Timedelta): Sector 3 recorded time
- **Sector1SessionTime** (pandas.Timedelta): Session time when the Sector 1 time was set
- **Sector2SessionTime** (pandas.Timedelta): Session time when the Sector 2 time was set
- **Sector3SessionTime** (pandas.Timedelta): Session time when the Sector 3 time was set
- **SpeedI1** (float): Speedtrap sector 1 [km/h]
- **SpeedI2** (float): Speedtrap sector 2 [km/h]
- **SpeedFL** (float): Speedtrap at finish line [km/h]
- **SpeedST** (float): Speedtrap on longest straight (Not sure) [km/h]
- **IsPersonalBest** (bool): Flag that indicates whether this lap is the official personal best lap of a driver. If any lap of a driver is quicker than their respective personal best lap, this means that the quicker lap is invalid and not counted. For example, this can happen if the track limits were exceeded.
- **Compound** (str): Tyres event specific compound name: SOFT, MEDIUM, HARD, INTERMEDIATE, WET (the actual underlying compounds C1 to C5 are not differentiated).
- **TyreLife** (float): Laps driven on this tire (includes laps in other sessions for used sets of tires)
- **FreshTyre** (bool): Tyre had TyreLife=0 at stint start, i.e. was a new tire
- **Team** (str): Team name
- **LapStartTime** (pandas.Timedelta): Session time at the start of the lap
- **LapStartDate** (pandas.Timestamp): Timestamp at the start of the lap
- **TrackStatus** (str): A string that contains track status numbers for all track status that occurred during this lap. The meaning of the track status numbers is explained in `fastf1.api.track_status_data()`. For filtering laps by track status, you may want to use `Laps.pick_track_status()`.
- **Position** (float): Position of the driver at the end of each lap. This value is NaN for FP1, FP2, FP3, Sprint Shootout, and Qualifying as well as for crash laps.
- **Deleted** (Optional[bool]): Indicates that a lap was deleted by the stewards, for example because of a track limits violation. This data is only available when race control messages are loaded.
- **DeletedReason** (str): Gives the reason for a lap time deletion. This data is only available when race control messages are loaded.



Source: [Fast F1 Docs]

# Data from FastF1

## Telemetry Data

- **Car data:**
  - *Speed* (float): Car speed [km/h]
  - *RPM* (int): Car RPM
  - *nGear* (int): Car gear number
  - *Throttle* (float): 0-100 Throttle pedal pressure [%]
  - *Brake* (bool): Brakes are applied or not.
  - *DRS* (int): DRS indicator (See `fastf1.api.car_data()` for more info)
- **Position data:**
  - *X* (float): X position [1/10 m]
  - *Y* (float): Y position [1/10 m]
  - *Z* (float): Z position [1/10 m]
  - *Status* (string): Flag - OffTrack/OnTrack
- **For both of the above:**
  - *Time* (timedelta): Time (0 is start of the data slice)
  - *SessionTime* (timedelta): Time elapsed since the start of the session
  - *Date* (datetime): The full date + time at which this sample was created
  - *Source* (str): Flag indicating how this sample was created:

Source: Fast F1 Docs

# Code Example

1. Position Change during Race
2. Driver Race Performance
3. Driver Lap Performance
4. Driver Lap Comparison

# Position Change during Race

Try to see drivers position from the first lap until the last lap throughout the race.

**Data we need** :

- Race session
- Drivers in the session
- Drivers position in each lap throughout the session

We can get all of those data from **‘Lap Timing Data’**

Then, we will **plot the position of each driver** throughout the race

# Position Change during Race

Import the Libraries needed, load the session, and create figures

```
import fastf1.plotting
import matplotlib.pyplot as plt

# Setup fastf1 matplotlib plotting modules
fastf1.plotting.setup_mpl(misc_mpl_mods=False)

# Load the session we choose
session = fastf1.get_session(2023, 'Abu Dhabi', 'R')
session.load(telemetry=False, weather=False)

# Define the figure size we will use
fig, ax = plt.subplots(figsize=(10.0, 4.9))
```

# Position Change during Race

Plot the position for each drivers

```
# For each driver, get their three letter abbreviation (e.g. 'HAM')
# by simply using the value of the first lap, get their color
# and then plot their position over the number of laps.

# Loop through each driver
for drv in session.drivers:
    drv_laps = session.laps.pick_driver(drv)
    # Get driver abbreviation and color
    abb = drv_laps['Driver'].iloc[0]
    color = fastf1.plotting.driver_color(abb)
    # Plot the lap position throughout the race
    ax.plot(drv_laps['LapNumber'], drv_laps['Position'],
            label=abb, color=color)
```

# Position Change during Race

Plot the position for each drivers

```
# For each driver, get their three letter abbreviation (e.g. 'VER')
# by simply using the value of the first lap, get their color
# and then plot their position over the number of laps.

# Loop through each driver
for drv in session.drivers:
    drv_laps = session.laps.pick_driver(drv)
    # Get driver abbreviation and color
    abb = drv_laps['Driver'].iloc[0]
    color = fastf1.plotting.driver_color(abb)
    # Plot the lap position throughout the race
    ax.plot(drv_laps['LapNumber'], drv_laps['Position'],
            label=abb, color=color)
```

# Position Change during Race

## Example of Lap Timing Data

```
session.laps.pick_driver('VER')
```

| | Time | Driver | DriverNumber | LapTime | LapNumber | Stint | PitOutTime | PitInTime | Sector1Time | Sector2Time | ... | FreshTyre | Team | LapStartTime | LapStartDate |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 days 01:03:57.047000 | VER | 1 | 0 days 00:01:32.190000 | 1.0 | 1.0 | NaT | NaT | NaT | 0 days 00:00:38.769000 | ... | True | Red Bull Racing | 0 days 01:02:24.519000 | 2023-11-26 13:03:25.644 |
| 1 | 0 days 01:05:27.757000 | VER | 1 | 0 days 00:01:30.710000 | 2.0 | 1.0 | NaT | NaT | 0 days 00:00:18.377000 | 0 days 00:00:38.691000 | ... | True | Red Bull Racing | 0 days 01:03:57.047000 | 2023-11-26 13:04:58.172 |
| 2 | 0 days 01:06:58.165000 | VER | 1 | 0 days 00:01:30.408000 | 3.0 | 1.0 | NaT | NaT | 0 days 00:00:18.549000 | 0 days 00:00:38.725000 | ... | True | Red Bull Racing | 0 days 01:05:27.757000 | 2023-11-26 13:06:28.882 |
| 3 | 0 days 01:08:28.881000 | VER | 1 | 0 days 00:01:30.716000 | 4.0 | 1.0 | NaT | NaT | 0 days 00:00:18.562000 | 0 days 00:00:38.810000 | ... | True | Red Bull Racing | 0 days 01:06:58.165000 | 2023-11-26 13:07:59.290 |
| 4 | 0 days 01:09:59.433000 | VER | 1 | 0 days 00:01:30.552000 | 5.0 | 1.0 | NaT | NaT | 0 days 00:00:18.599000 | 0 days 00:00:38.714000 | ... | True | Red Bull Racing | 0 days 01:08:28.881000 | 2023-11-26 13:09:30.006 |
| 5 | 0 days 01:11:29.873000 | VER | 1 | 0 days 00:01:30.440000 | 6.0 | 1.0 | NaT | NaT | 0 days 00:00:18.445000 | 0 days 00:00:38.767000 | ... | True | Red Bull Racing | 0 days 01:09:59.433000 | 2023-11-26 13:11:00.558 |
| 6 | 0 days 01:13:00.402000 | VER | 1 | 0 days 00:01:30.529000 | 7.0 | 1.0 | NaT | NaT | 0 days 00:00:18.491000 | 0 days 00:00:38.692000 | ... | True | Red Bull Racing | 0 days 01:11:29.873000 | 2023-11-26 13:12:30.998 |
| 7 | 0 days 01:14:30.918000 | VER | 1 | 0 days 00:01:30.516000 | 8.0 | 1.0 | NaT | NaT | 0 days 00:00:18.368000 | 0 days 00:00:38.719000 | ... | True | Red Bull Racing | 0 days 01:13:00.402000 | 2023-11-26 13:14:01.527 |

# Position Change during Race

## Example of Lap Timing Data

```
session.laps.pick_driver('VER').iloc[0]
```

| | |
|---|---|
| Time | 0 days 01:03:57.047000 |
| Driver | VER |
| DriverNumber | 1 |
| LapTime | 0 days 00:01:32.190000 |
| LapNumber | 1.0 |
| Stint | 1.0 |
| PitOutTime | NaT |
| PitInTime | NaT |
| Sector1Time | NaT |
| Sector2Time | 0 days 00:00:38.769000 |
| Sector3Time | 0 days 00:00:33.004000 |
| Sector1SessionTime | NaT |
| Sector2SessionTime | 0 days 01:03:24.128000 |
| Sector3SessionTime | 0 days 01:03:57.112000 |
| SpeedI1 | 284.0 |
| SpeedI2 | 293.0 |
| SpeedFL | 213.0 |
| SpeedST | 304.0 |
| IsPersonalBest | False |
| Compound | MEDIUM |
| TyreLife | 1.0 |
| FreshTyre | True |
| Team | Red Bull Racing |
| LapStartTime | 0 days 01:02:24.519000 |
| LapStartDate | 2023-11-26 13:03:25.644000 |
| TrackStatus | 1 |
| Position | 1.0 |
| Deleted | False |
| DeletedReason | |
| FastF1Generated | False |
| IsAccurate | False |

Name: 0, dtype: object

# Position Change during Race

Beautify the plot figure

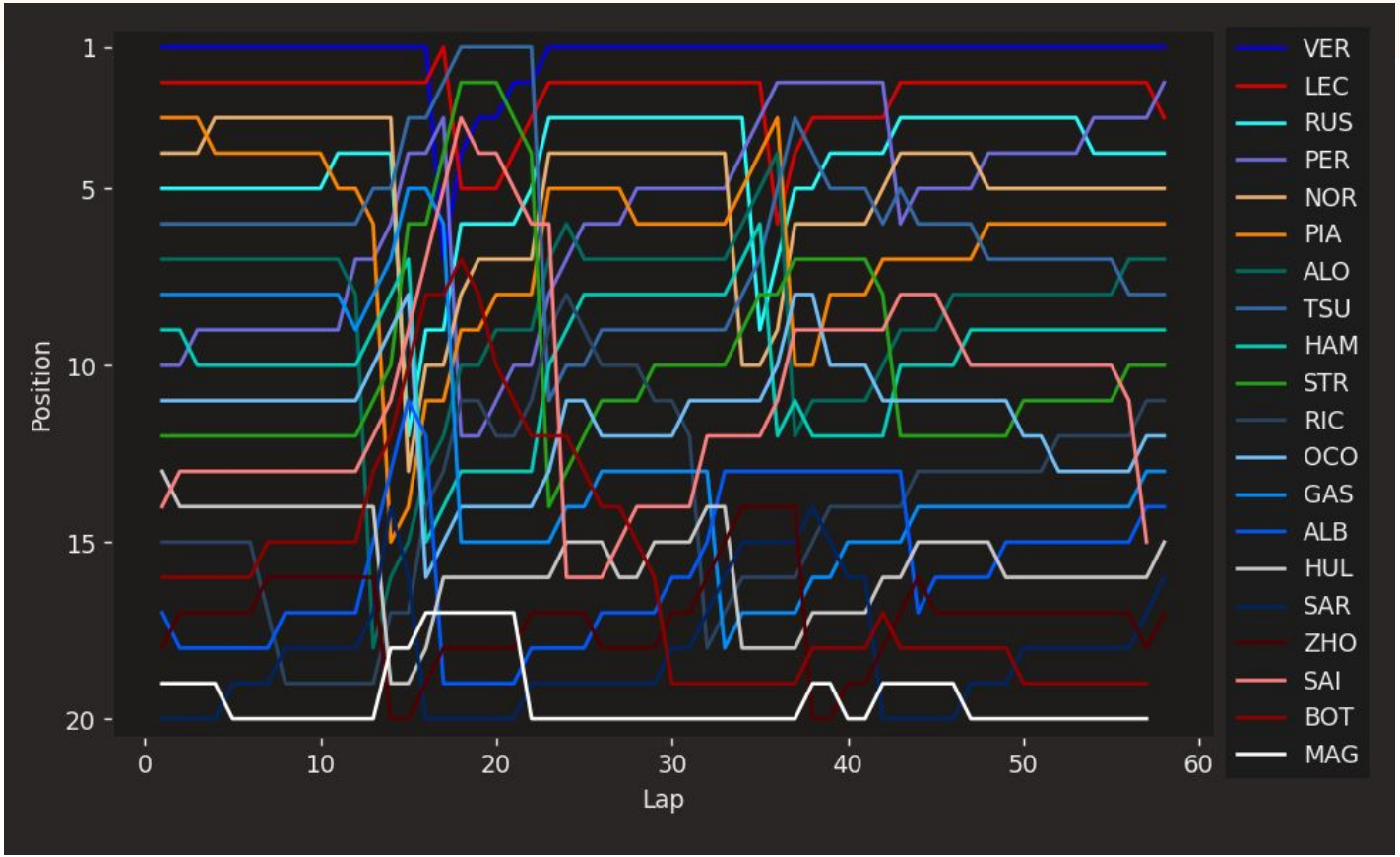
```
# Make the plot looks better by setting y limit
# Add dabel to the axis
ax.set_ylim([20.5, 0.5])
ax.set_yticks([1, 5, 10, 15, 20])
ax.set_xlabel('Lap')
ax.set_ylabel('Position')

# Add legend
ax.legend(bbox_to_anchor=(1.0, 1.02))
plt.tight_layout()

plt.show()
```

# Position Change during Race

Plot result



# Driver Race Performance

Try to see drivers laptime from the first lap until the last lap throughout the race.

**Data we need** :

- Race session
- Drivers in the session
- Drivers LapTime in each lap throughout the session

We can get all of those data from **‘Lap Timing Data’**

Then, we will **plot the laptime** throughout the race

# Driver Race Performance

Import the Libraries needed, load the session, and create figures

```
import fastf1.plotting
import matplotlib.pyplot as plt

# Setup fastf1 matplotlib plotting modules
fastf1.plotting.setup_mpl(misc_mpl_mods=False)

# Load the session we choose
session = fastf1.get_session(2023, 'Abu Dhabi', 'R')
session.load(telemetry=False, weather=False)

# Define the figure size we will use
fig, ax = plt.subplots(figsize=(10.0, 4.9))
```

# Driver Race Performance

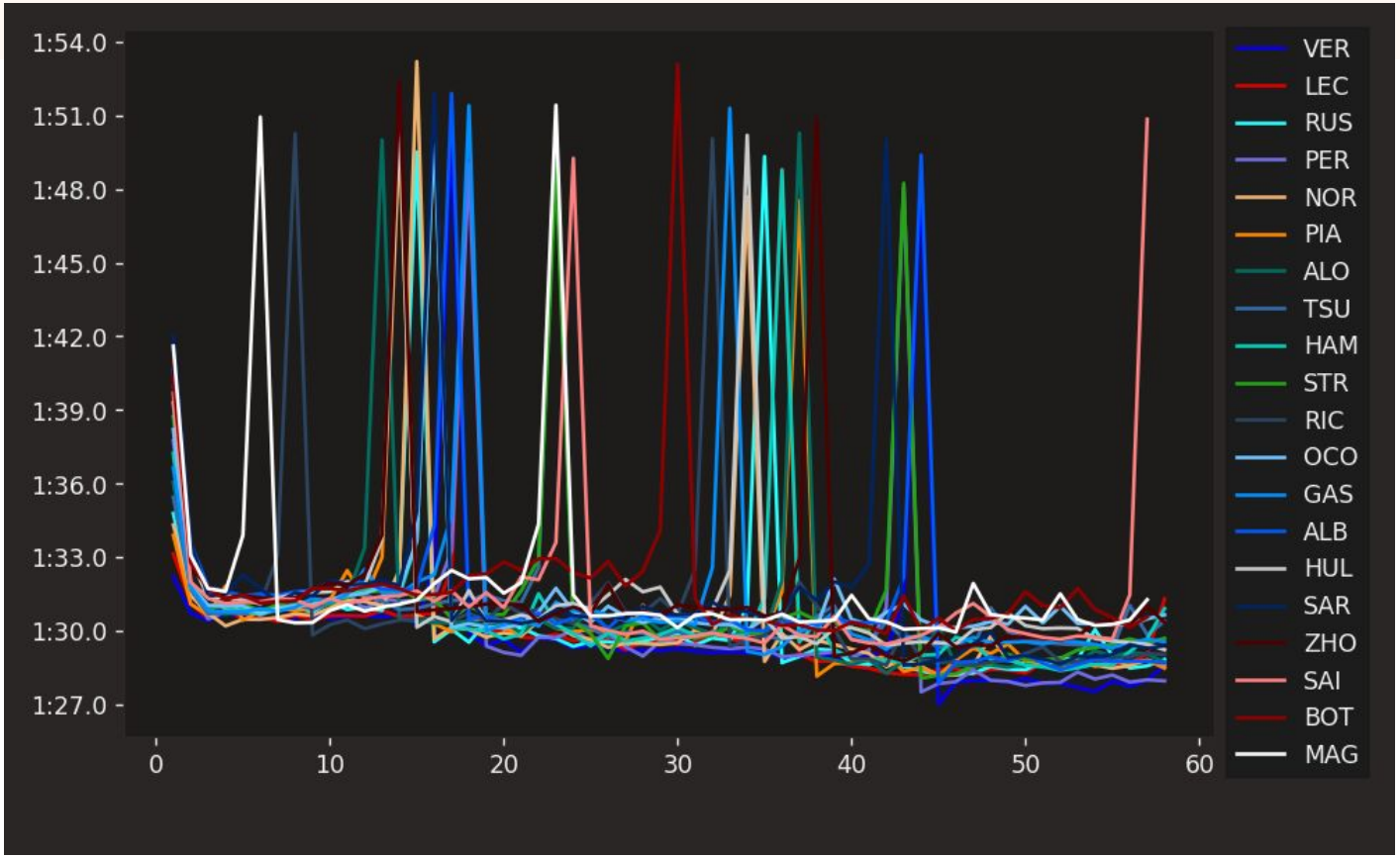
Plot the Laptime for each drivers

```
# For each driver, get their three letter abbreviation (e.g. 'VER')
# by simply using the value of the first lap, get their color
# and then plot their position over the number of laps.

# Loop through each driver
for drv in session.drivers:
    drv_laps = session.laps.pick_driver(drv)
    # Get driver abbreviation and color
    abb = drv_laps['Driver'].iloc[0]
    color = fastf1.plotting.driver_color(abb)
    # Plot the lap position throughout the race
    ax.plot(drv_laps['LapNumber'], drv_laps['LapTime'],
            label=abb, color=color)
```

# Driver Race Performance

Plot result



# Driver Race Performance

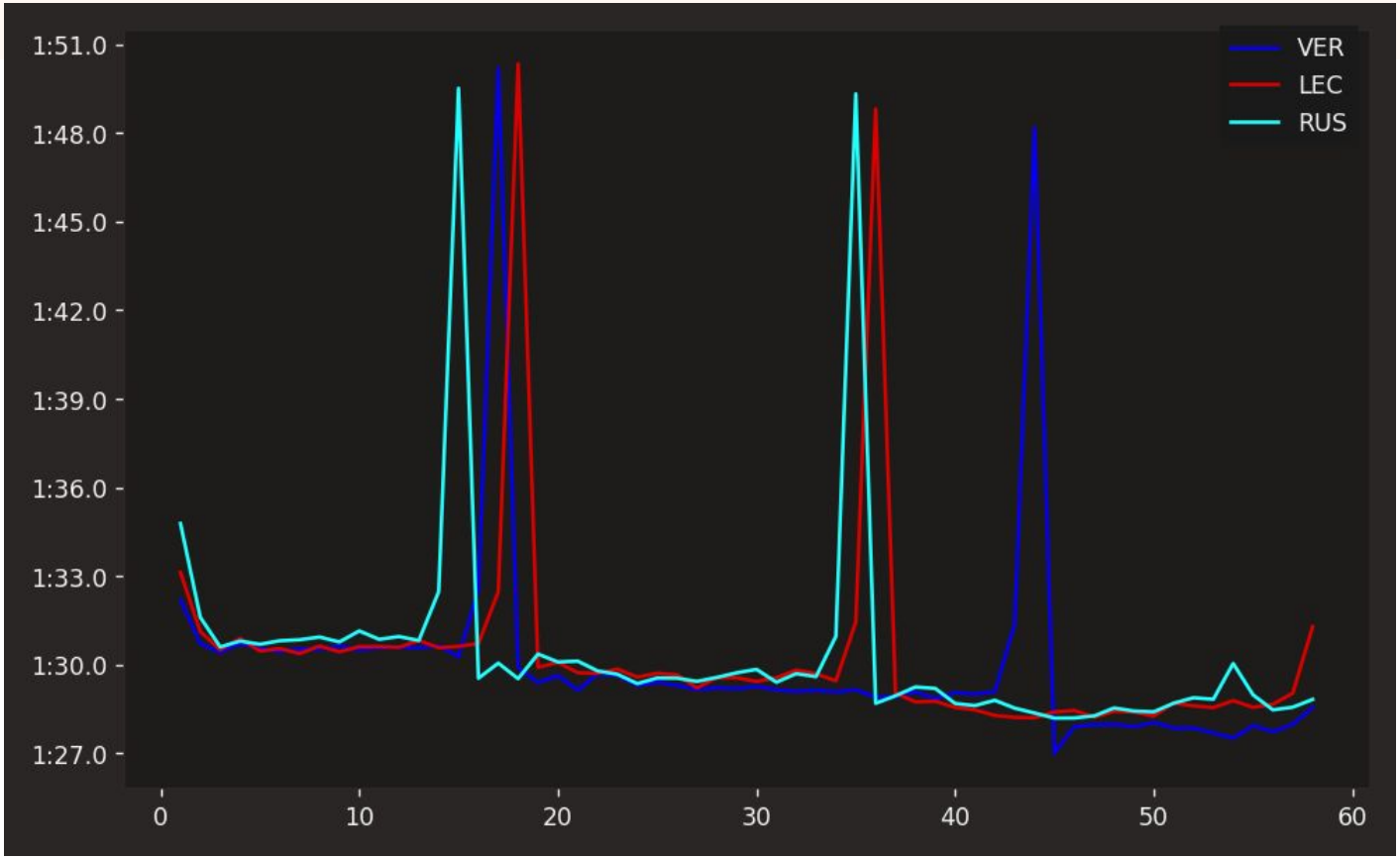
Plot the Laptime for 3 drivers only

```
# If we show all of the drivers, the plot is too crowded
# Let’s try for only three drivers

# Loop through each driver
for drv in ['VER','LEC','RUS']:
    drv_laps = session.laps.pick_driver(drv)
    # Get driver abbreviation and color
    abb = drv_laps['Driver'].iloc[0]
    color = fastf1.plotting.driver_color(abb)
    # Plot the lap position throughout the race
    ax.plot(drv_laps['LapNumber'], drv_laps['LapTime'],
            label=abb, color=color)
```

# Driver Race Performance

Plot result



# Driver Race Performance

Let's see what tyre are used

```
# Create list of stints and count how many laps each stint used
stints = session.laps[["Driver", "Stint", "Compound", "LapNumber"]]
stints = stints.groupby(["Driver", "Stint", "Compound"])
stints = stints.count().reset_index()
# Rename the columns
stints = stints.rename(columns={"LapNumber": "StintLength"})
# See the data
print(stints.loc[stints['Driver'].isin(['VER','LEC','RUS'])])
```

| | Driver | Stint | Compound | StintLength |
|---|---|---|---|---|
| 17 | LEC | 1.0 | MEDIUM | 17 |
| 18 | LEC | 2.0 | HARD | 18 |
| 19 | LEC | 3.0 | HARD | 23 |
| 37 | RUS | 1.0 | MEDIUM | 14 |
| 38 | RUS | 2.0 | HARD | 20 |
| 39 | RUS | 3.0 | HARD | 24 |
| 51 | VER | 1.0 | MEDIUM | 16 |
| 52 | VER | 2.0 | HARD | 27 |
| 53 | VER | 3.0 | HARD | 15 |

# Driver Race Performance

Plot the Laptime for 3 drivers only

```
# Create figure with two plots
fig, ax = plt.subplots(2, 1, gridspec_kw={'height_ratios': [4,1]})

# Loop through each driver
for drv in ['VER','LEC','RUS']:
    drv_laps = session.laps.pick_driver(drv)
    # Get driver abbreviation and color
    abb = drv_laps['Driver'].iloc[0]
    color = fastf1.plotting.driver_color(abb)
    # Plot the lap position throughout the race
    ax[0].plot(drv_laps['LapNumber'], drv_laps['LapTime'],
            label=abb, color=color)
```

# Driver Race Performance

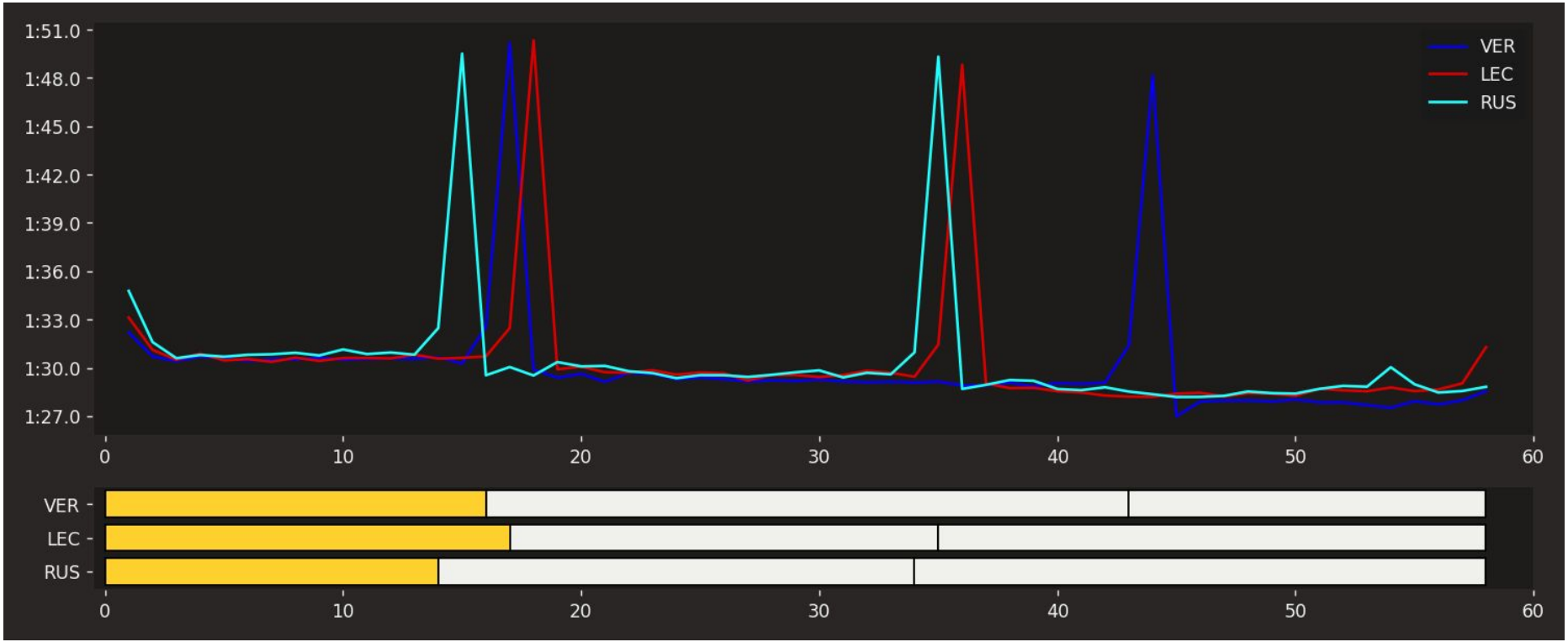
Create bar graph for stints used

```
# Loop through the drivers
for driver in ['VER','LEC','RUS']:
    driver_stints = stints.loc[stints["Driver"] == driver]

    previous_stint_end = 0
    for idx, row in driver_stints.iterrows():
        # each row contains the compound name and stint length
        # we can use these information to draw horizontal bars
        ax[1].barh(
            y=driver,
            width=row["StintLength"],
            left=previous_stint_end,
            color=fastf1.plotting.COMPOUND_COLORS[row["Compound"]],
            edgecolor="black",
            fill=True
        )
        previous_stint_end += row["StintLength"]
```

# Driver Race Performance

Figure result of Laptimes Plot and Stints Used Bar



# Driver Lap Performance

Try to see and compare drivers speeds from their best lap in a session.

**Data we need** :

- Race session
- Drivers best lap in the session
- Drivers Laptime and Telemetry for the lap

We can get all of those data from **‘Telemetry Data’**

Then, we will **plot the data** of **speed**, **throttle**, and **delta time** of the drivers throughout their fastest lap.

# Driver Lap Performance

Import the Libraries needed, load the session, and select driver

```
import fastf1.plotting
import matplotlib.pyplot as plt

# Setup fastf1 matplotlib plotting modules
fastf1.plotting.setup_mpl(misc_mpl_mods=False)

# Load the session we choose
session = fastf1.get_session(2023, 'Monaco', 'Q')
session.load()

# Select the fastest lap of driver we want to compare
ver_lap = session.laps.pick_driver('VER').pick_fastest()
alo_lap = session.laps.pick_driver('ALO').pick_fastest()
```

# Driver Lap Performance

## Get the telemetry

```
# Get the telemetry data for each driver
ver_tel = ver_lap.get_car_data().add_distance()
alo_tel = alo_lap.get_car_data().add_distance()
```

| | Date | RPM | Speed | nGear | Throttle | Brake | DRS | Source | Time | SessionTime | Distance |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2023-05-27 15:10:04.325 | 11158 | 282 | 7 | 100 | False | 12 | car | 0 days 00:00:00.299000 | 0 days 01:22:56.205000 | 23.421667 |
| 1 | 2023-05-27 15:10:04.485 | 11315 | 285 | 7 | 100 | False | 12 | car | 0 days 00:00:00.459000 | 0 days 01:22:56.365000 | 36.088333 |
| 2 | 2023-05-27 15:10:04.845 | 11350 | 288 | 7 | 100 | False | 12 | car | 0 days 00:00:00.819000 | 0 days 01:22:56.725000 | 64.888333 |
| 3 | 2023-05-27 15:10:05.005 | 11371 | 290 | 7 | 91 | False | 12 | car | 0 days 00:00:00.979000 | 0 days 01:22:56.885000 | 77.777222 |
| 4 | 2023-05-27 15:10:05.165 | 11197 | 289 | 7 | 0 | True | 8 | car | 0 days 00:00:01.139000 | 0 days 01:22:57.045000 | 90.621667 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 257 | 2023-05-27 15:11:14.605 | 10796 | 262 | 6 | 100 | False | 12 | car | 0 days 00:01:10.579000 | 0 days 01:24:06.485000 | 3216.240556 |
| 258 | 2023-05-27 15:11:14.805 | 10517 | 267 | 7 | 100 | False | 12 | car | 0 days 00:01:10.779000 | 0 days 01:24:06.685000 | 3231.073889 |
| 259 | 2023-05-27 15:11:14.965 | 10767 | 267 | 7 | 100 | False | 12 | car | 0 days 00:01:10.939000 | 0 days 01:24:06.845000 | 3242.940556 |
| 260 | 2023-05-27 15:11:15.205 | 10890 | 274 | 7 | 100 | False | 12 | car | 0 days 00:01:11.179000 | 0 days 01:24:07.085000 | 3261.207222 |
| 261 | 2023-05-27 15:11:15.365 | 10984 | 275 | 7 | 100 | False | 12 | car | 0 days 00:01:11.339000 | 0 days 01:24:07.245000 | 3273.429444 |

# Driver Lap Performance

## Plot the telemetry data

```
# Create a figure contain 3 plots
fig, ax = plt.subplots(3, 1, gridspec_kw={'height_ratios': [3,1,1]},figsize=(12.0,5.0))
# Give title
plt.suptitle(f"Fastest Lap Comparison \n "
             f"{session.event['EventName']} {session.event.year} Qualifying")

# Plot 1, comparing speed throughout the lap
ax[0].plot(ver_tel['Distance'], ver_tel['Speed'], color=rbr_color, label='VER')
ax[0].plot(alo_tel['Distance'], alo_tel['Speed'], color=amr_color, label='ALO')
ax[0].set_ylabel('Speed in km/h')
ax[0].legend()

# Plot 2, comparing throttle throughout the lap
ax[1].plot(ver_tel['Distance'], ver_tel['Throttle'], color=rbr_color, label='VER')
ax[1].plot(alo_tel['Distance'], alo_tel['Throttle'], color=amr_color, label='ALO')
ax[1].set_ylabel('Throttle')
ax[1].legend()
```

# Driver Lap Performance

Plot the telemetry data

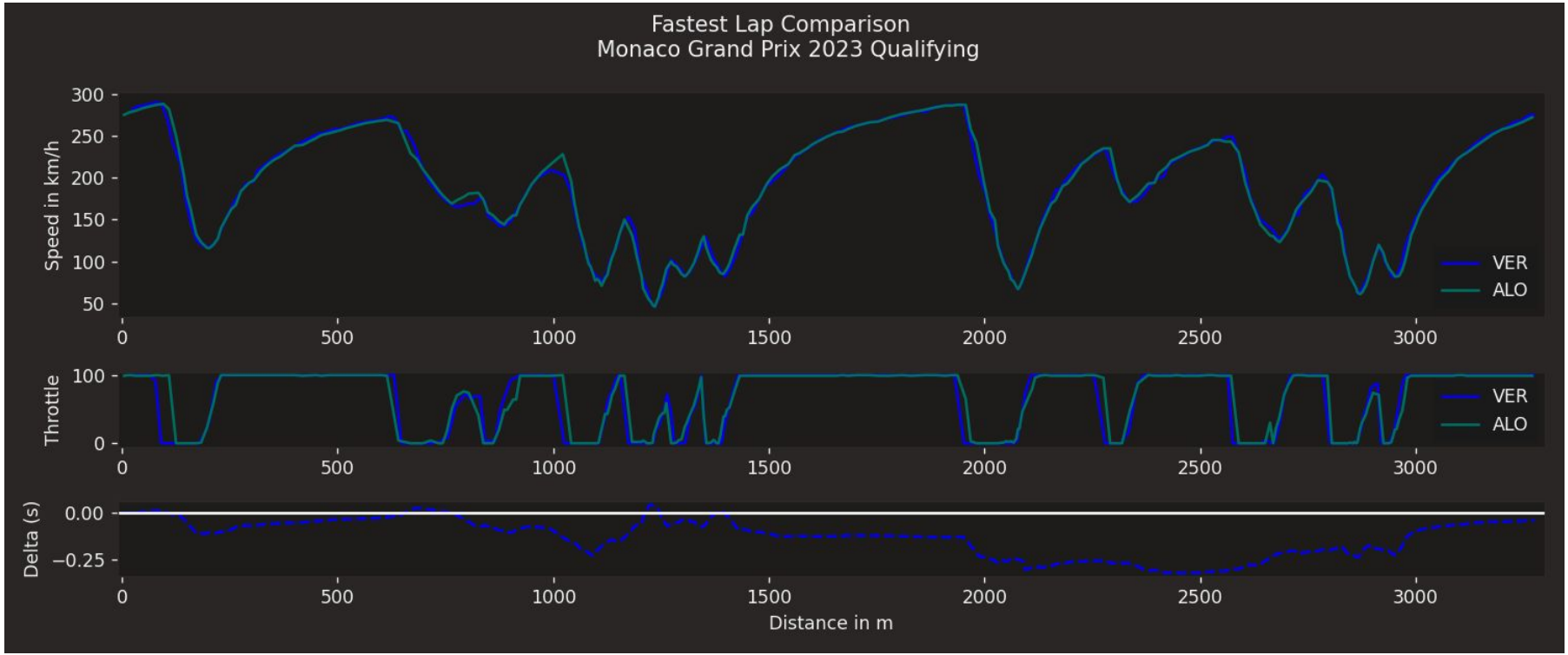
```
# Extract delta time between the drivers
delta_time, ref_tel, compare_tel = fastf1.utils.delta_time(ver_lap, alo_lap)

# Plot 3, comparing the delta time of the drivers
ax[2].plot(ref_tel['Distance'], delta_time, 'b--')
ax[2].axhline(0,color='white')
ax[2].set_ylabel('Delta (s)')

# Setup the figure layout
ax[2].set_xlabel('Distance in m')
ax[0].set_xlim([-10.5, 3300])
ax[1].set_xlim([-10.5, 3300])
ax[2].set_xlim([-10.5, 3300])
plt.tight_layout()
```

# Driver Lap Performance

## Plotting Results



# Driver Lap Comparison

Import the Libraries needed and load the session

```
import fastf1.plotting
import matplotlib.pyplot as plt
from matplotlib.collections import LineCollection
from matplotlib import cm
import numpy as np

# Setup fastf1 matplotlib plotting modules
fastf1.plotting.setup_mpl(misc_mpl_mods=False)

# Load the session we choose
session = fastf1.get_session(2023, 'Monaco', 'Q')
session.load()
```

# Driver Lap Comparison

Select driver and combine telemetry

```
# Select the fastest lap of driver we want to compare
ver_lap = session.laps.pick_driver('VER').pick_fastest()
alo_lap = session.laps.pick_driver('ALO').pick_fastest()

# Get the telemetry data for each driver
ver_tel = ver_lap.get_telemetry().add_distance()
alo_tel = alo_lap.get_telemetry().add_distance()

# Add driver column
ver_tel['Driver'] = 'VER'
alo_tel['Driver'] = 'ALO'

# Append the telemetry
all_tel = ver_tel._append(alo_tel)
```

# Driver Lap Comparison

Create Mini-Sector

```
# We want 21 mini-sectors (this can be adjusted up and down)
num_minisectors = 7*3
# Grab the maximum value of distance that is known in the telemetry
total_distance = max(all_tel['Distance'])
# Generate equally sized mini-sectors
minisector_length = total_distance / num_minisectors

# Initiate minisector variable, with 0 (meters) as a starting point.
minisectors = [0]
# Add multiples of minisector_length to the minisectors
for i in range(0, (num_minisectors - 1)):
    minisectors.append(minisector_length * (i + 1))
```

# Driver Lap Comparison

Pick fastest driver for each mini-sector

```
# Add mini-sector to telemetry
all_tel['Minisector'] = all_tel['Distance'].apply(
                lambda dist: (int((dist // minisector_length) + 1)))

# Create average speed
average_speed = all_tel.groupby(['Minisector',
                'Driver'])['Speed'].mean().reset_index()

# Select the driver with the highest average speed
fastest_driver = average_speed.loc[average_speed.groupby(
                ['Minisector'])['Speed'].idxmax()]

# Get rid of the speed column and rename the driver column
fastest_driver = fastest_driver[['Minisector',
                'Driver']].rename(columns={'Driver': 'Fastest_driver'})
```

# Driver Lap Comparison

Create mini-sector map

```
# Join the fastest driver per minisector with the full telemetry
all_tel = all_tel.merge(fastest_driver, on=['Minisector'])
# Order the data by distance to make matploblib does not get confused
all_tel = all_tel.sort_values(by=['Distance'])

# Convert driver name to integer
all_tel.loc[all_tel['Fastest_driver'] == 'VER', 'Fastest_driver_int'] = 1
all_tel.loc[all_tel['Fastest_driver'] == 'ALO', 'Fastest_driver_int'] = 2
x = np.array(all_tel['X'].values)
y = np.array(all_tel['Y'].values)

# Prepare for the map
points = np.array([x, y]).T.reshape(-1, 1, 2)
segments = np.concatenate([points[:-1], points[1:]], axis=1)
fastest_driver_array = all_tel['Fastest_driver_int'].to_numpy().astype(float)
```

# Driver Lap Comparison

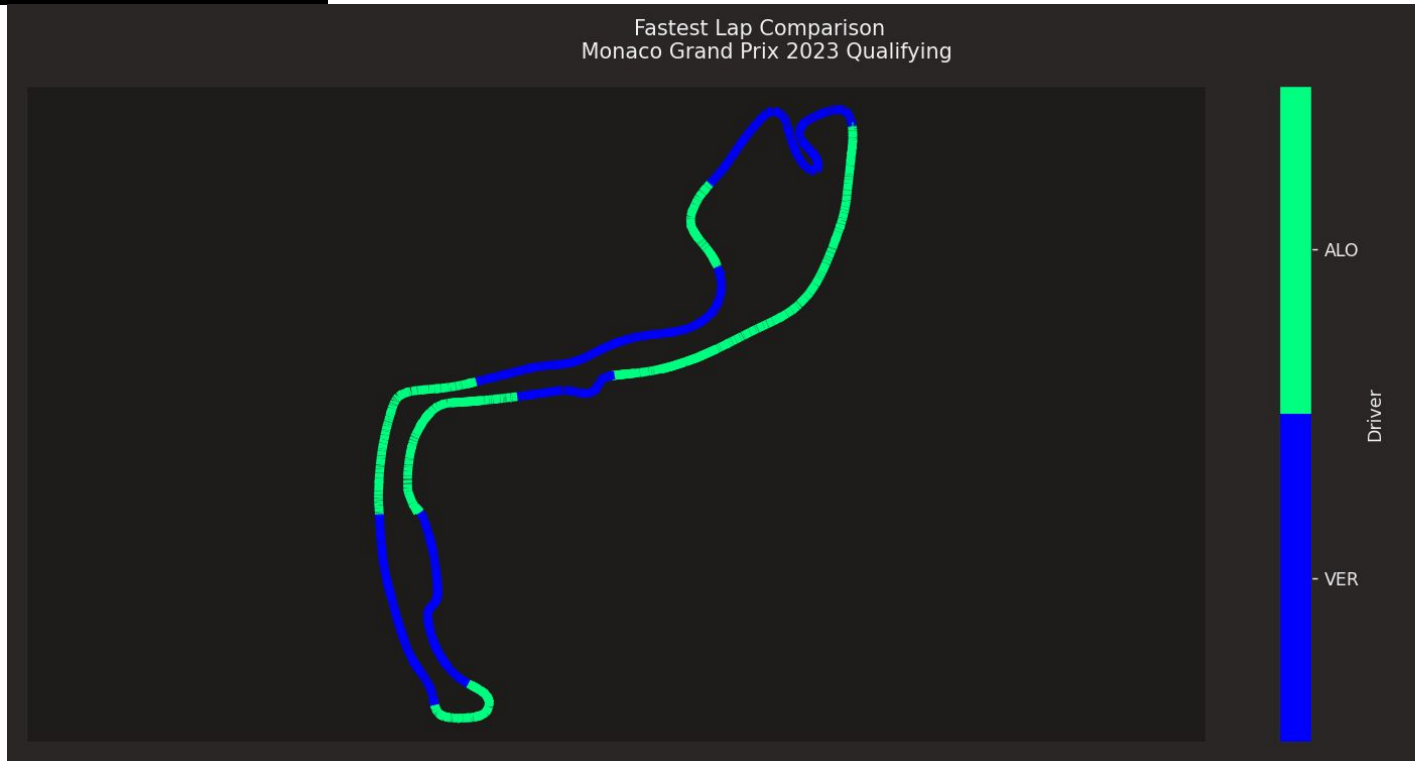
## Create Mini-Sector Map

```
cmap = plt.get_cmap('winter',2)
lc_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1),
cmap=cmap)
lc_comp.set_array(fastest_driver_array)
lc_comp.set_linewidth(5)
plt.rcParams['figure.figsize'] = [12, 6]

plt.gca().add_collection(lc_comp)
plt.axis('equal')
plt.tick_params(labelleft=False, left=False, labelbottom=False,
bottom=False)
cbar = plt.colorbar(mappable=lc_comp, label='Driver',
boundaries=np.arange(1,4))
cbar.set_ticks(np.arange(1.5, 3.5))
cbar.set_ticklabels(['VER','ALO'])
```

# Driver Lap Comparison

## Mini-Sector Map Result



# Resources

- Fast F1 Github Pages
- Fast F1 Documentation
- Fast F1 Example Gallery
- Towards F1 Data Analysis @Twitter
- F1 Speed Indonesia @Twitter
- Towards F1 Data Analysis – Jesper @Medium
- Raul Garcia @Medium

# Let’s Come to Indonesia!


PYCON APAC 2024
Yogyakarta, Indonesia
October 25-27, 2024

#PyConTH #PyConTH2023


# Thanks!
# Terima Kasih!
# ขอบคุณ

**Dima M Dinama**

Follow Me : @dmaharika
GitHub, Twitter/X, Instagram, LinkedIn, Facebook

**Slides available at**
https://dinama.id/slides/pyconth2023.pdf