Assignment3_Final

March 24, 2025

1 F1 Interactive Dashboard Tutorial

Author: Anuj Kadakia

Description: F1 Overview with multiple visualizations The goal of this project is twofold:

1. **To deepen my understanding** of interactive visualizations
2. **To create a clear, worked example** that others can learn from

1.1 1. Visualization Techniques (25%)

This dashboard integrates **five distinct chart types** to explore race and driver trends in Formula 1. Each was chosen to highlight a different part of the dataset:

1.1.1 Line Chart – Driver Points Over Time

Displays how selected drivers performed across multiple seasons.
This chart is controlled by user filters: driver name, nationality, and year range. Ideal for analyzing long-term performance.

1.1.2 Bar Chart – Drivers by Nationality

This static chart gives a view of the diversity of F1 competitors, sorted by nationality.

1.1.3 Bar Chart – Car Constructor Win Counts

Highlights the most successful car constructors across all seasons. This helps correlate team success with individual performance.

1.1.4 Scatter Plot – Race Winners by Year and Circuit

A visual breakdown of who won which races, where, and when.
Color-coded by driver and arranged with race year on the x-axis and Grand Prix circuit on the y-axis. Also filterable.

1.1.5 Geographic Heat Map – F1 Drivers by Country of Origin

Visualizes the count of drivers by country using a global choropleth map.
This adds geographical storytelling and works with the nationality bar chart.

1.1.6 Visual Interactivity

Where beneficial, there are three main filters used. The first is the ability to filter by year, as the data spans a range of 1950-2023. Then, there is an option to pick driver names, based on interest. This allows you to select groupings of drivers. Lastly you can use a dropdown to pick specific nationalities, such as if the interest was to analyze performance amongst drivers from Great Britain, specifically. With the different interactive, updating filters, it is possible to take a look at unique slices of the data, in addition to the entirety.

Together, these charts complement each other: - Time-series + geographic + categorical perspectives - Static and interactive modes - Macro trends + micro breakdowns

As a whole, the visuals work to paint a picture of drivers, constructors, and races across the years

1.2 2. Visualization Library (25%)

I considered different potential library choices, such as Matplotlib, Seaborn, and Plotly. I could have produced similar visuals with each of the libraries but decided to explore Plotly further.

This dashboard was built using:

- **Plotly Express**: A declarative, open-source visualization library built by Plotly inc. It integrates directly with pandas DataFrames and supports interactive, publish-ready charts in a few lines of code.
- **ipywidgets**: An interactive widget library designed for Jupyter environments. It allows real-time UI control of plots via sliders, dropdowns, and more.

1.2.1 Why These Libraries?

Feature	Plotly + ipywidgets
Interactive	Fully interactive & browser-native
Easy to Learn	Few lines of code per chart
Jupyter-Friendly	Seamless notebook support
Declarative	Clean syntax, fast layout

This approach avoids the potential issues with a web app framework (e.g., Dash or Streamlit) while keeping everything in a single notebook. Also, Plotly supports over 40+ different visualizations.

1.2.2 How to install

Use: pip install plotly ipywidgets

1.3 3. Demonstration (50%)

2 Import Libraries

The following libraries will be used for data cleaning and visualizations

[1]:
```
import pandas as pd
import plotly.express as px
```

```
import plotly.graph_objects as go
import ipywidgets as widgets
from IPython.display import display, clear_output
```

Two CSV files were used:

- `drivers_updated.csv`: Driver-level data by season including position, points, nationality, and constructor.
- `winners.csv`: Race-level data including Grand Prix names, race dates, and winning drivers.

This data was collected from Kaggle

[2]:
```
drivers_df = pd.read_csv('drivers_updated.csv')
winners_df = pd.read_csv('winners.csv')
```

[3]:
```
# Defines basis to track driver points over time through a line chart
def line_chart(df):
    fig = px.line(df, x='year', y='PTS', color='Driver', title='Driver Points␣
 ↪Over Time')
    fig.update_layout(legend_title_text='Driver', height=400, margin=dict(t=50,␣
 ↪l=25, r=25, b=25))
    fig.show()
```

[4]:
```
# Gives initial parameters to visualize bar chart for drivers by nationality
def nationality_bar(df):
    nat_counts = df['Nationality'].value_counts().reset_index()
    nat_counts.columns = ['Nationality', 'Count']
    fig = px.bar(nat_counts, x='Nationality', y='Count', title='Drivers by␣
 ↪Nationality')
    fig.update_layout(height=400, margin=dict(t=50, l=25, r=25, b=25))
    fig.show()
```

[5]:
```
# Determines visual for bar chart to show the constructor wins achieved␣
 ↪relative to one another
def constructor_bar(df):
    car_counts = df['Car'].value_counts().reset_index()
    car_counts.columns = ['Constructor', 'Wins']
    fig = px.bar(car_counts, x='Constructor', y='Wins', title='Car Constructor␣
 ↪Win Counts')
    fig.update_layout(height=400, margin=dict(t=50, l=25, r=25, b=25))
    fig.show()
```

[6]:
```
# Scatter plot visualization to compare circuit winners based on driver over␣
 ↪time
def race_scatter(df):
    df['Date'] = pd.to_datetime(df['Date'])
    df['Year'] = df['Date'].dt.year
    fig = px.scatter(
```

```
        df,
        x='Year',
        y='Grand Prix',
        color='Winner',
        title='Race Winners by Year and Circuit',
        labels={'Year': 'Race Year', 'Grand Prix': 'Circuit', 'Winner':␣
 ↪'Driver'},
        hover_data=['Winner'],
        height=800
    )
    fig.update_layout(margin=dict(t=50, l=25, r=25, b=25))
    fig.show()
```

[7]:
```
# Geographical visualization to show heat map of drivers based on the country␣
 ↪of origin
def geo_map(df):
    nationality_to_country = {
        'BRA': 'Brazil', 'SWE': 'Sweden', 'GBR': 'United Kingdom', 'ITA':␣
 ↪'Italy',
        'FRA': 'France', 'GER': 'Germany', 'USA': 'United States', 'AUS':␣
 ↪'Australia',
        'CAN': 'Canada', 'JPN': 'Japan', 'ESP': 'Spain', 'NED': 'Netherlands',
        'ARG': 'Argentina', 'FIN': 'Finland', 'AUT': 'Austria', 'NZL': 'New␣
 ↪Zealand',
        'BEL': 'Belgium', 'SUI': 'Switzerland', 'MEX': 'Mexico', 'RSA': 'South␣
 ↪Africa',
        'DEN': 'Denmark', 'RUS': 'Russia', 'MON': 'Monaco', 'POL': 'Poland',
        'THA': 'Thailand', 'VEN': 'Venezuela', 'COL': 'Colombia', 'IND':␣
 ↪'India',
        'POR': 'Portugal', 'IRL': 'Ireland', 'RHO': 'Rhodesia', 'CHN': 'China',
        'HUN': 'Hungary', 'CHI': 'Chile', 'MAS': 'Malaysia', 'INA': 'Indonesia',
        'RAF': 'French Equatorial Africa'
    }
    df_counts = df['Nationality'].value_counts().reset_index()
    df_counts.columns = ['Nationality', 'Count']
    df_counts['Country'] = df_counts['Nationality'].map(nationality_to_country)
    fig = px.choropleth(df_counts,
                        locations="Country",
                        locationmode="country names",
                        color="Count",
                        hover_name="Country",
                        color_continuous_scale="Oranges",
                        title="F1 Drivers by Country of Origin (Heat Map)")
    fig.update_layout(margin=dict(t=50, l=25, r=25, b=25))
    fig.show()
```

[8]:
```
# Allows year to be filtered based on total dataset
year_slider = widgets.IntRangeSlider(
    value=[1950, 2023],
    min=1950,
    max=2023,
    step=1,
    description='Year Range:',
    continuous_update=False
)

# Can select different groupings of drivers, including all at once or none
driver_dropdown = widgets.SelectMultiple(
    options=sorted(drivers_df['Driver'].unique()),
    description='Drivers:',
    value=['Nino  Farina '],
    layout=widgets.Layout(width='50%')
)

# Starts as all, can select specific nationalities
nationality_dropdown = widgets.Dropdown(
    options=['All'] + sorted(drivers_df['Nationality'].unique()),
    value='All',
    description='Nationality:'
)
```

3 Plotting the data

With the interactive filters created and the visuals defined, now we can put it together and take a look at the different visualizations. This will help us understand the F1 datasets to be able to gather insight of the performance and statistics over time.

3.0.1 Line Chart – Driver Points Over Time

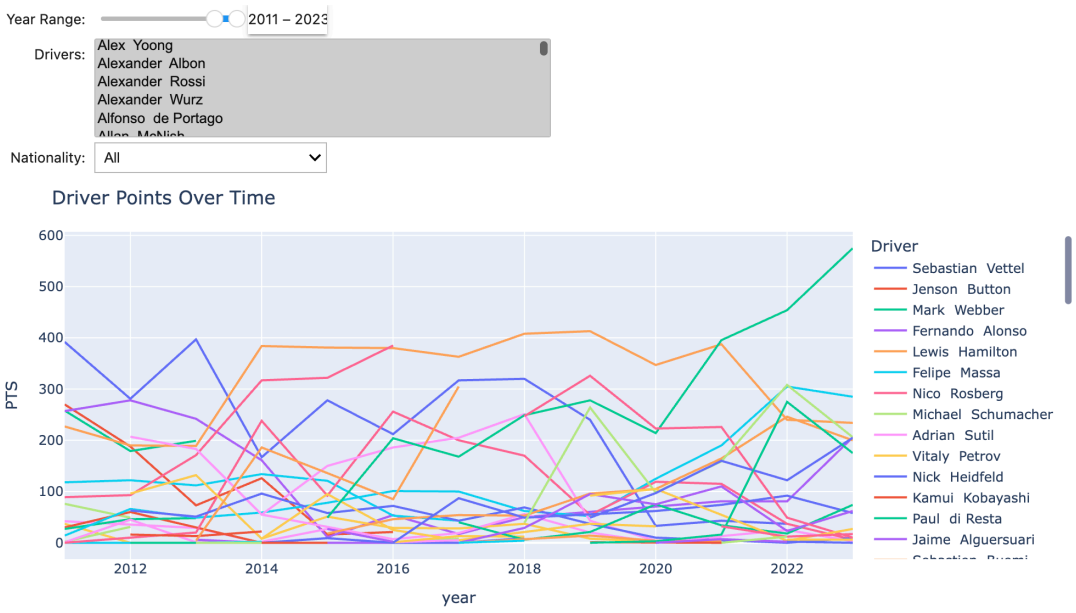
This chart uses a **line graph** to show how drivers accumulate points across F1 seasons.
It helps highlight consistency, peaks, and trends among selected drivers.
- **x-axis**: Year
- **y-axis**: Points (PTS)
- **Color-coded**: Driver name
This visualization is interactive and updates based on driver, year, and nationality filters.

[9]:
```
def interactive_driver_points():
    # Builds an interactive widget to filter and plot driver points over time
    def update(year_range, selected_drivers, nationality_filter):
        clear_output(wait=True)
        filtered_df = drivers_df[(drivers_df['year'] >= year_range[0]) &␣
↪(drivers_df['year'] <= year_range[1])]
        if nationality_filter != 'All':
```

```
            filtered_df = filtered_df[filtered_df['Nationality'] ==␣
↪nationality_filter]
        filtered_df = filtered_df[filtered_df['Driver'].isin(selected_drivers)]
        line_chart(filtered_df)
    # Displays the interactive controls and links them to the updating feature
    display(widgets.VBox([
        year_slider,
        driver_dropdown,
        nationality_dropdown,
        widgets.interactive_output(update, {
            'year_range': year_slider,
            'selected_drivers': driver_dropdown,
            'nationality_filter': nationality_dropdown
        })
    ]))
```

[10]:
```
interactive_driver_points()
```

```
VBox(children=(IntRangeSlider(value=(1950, 2023), continuous_update=False,␣
↪description='Year Range:', max=2023…
```

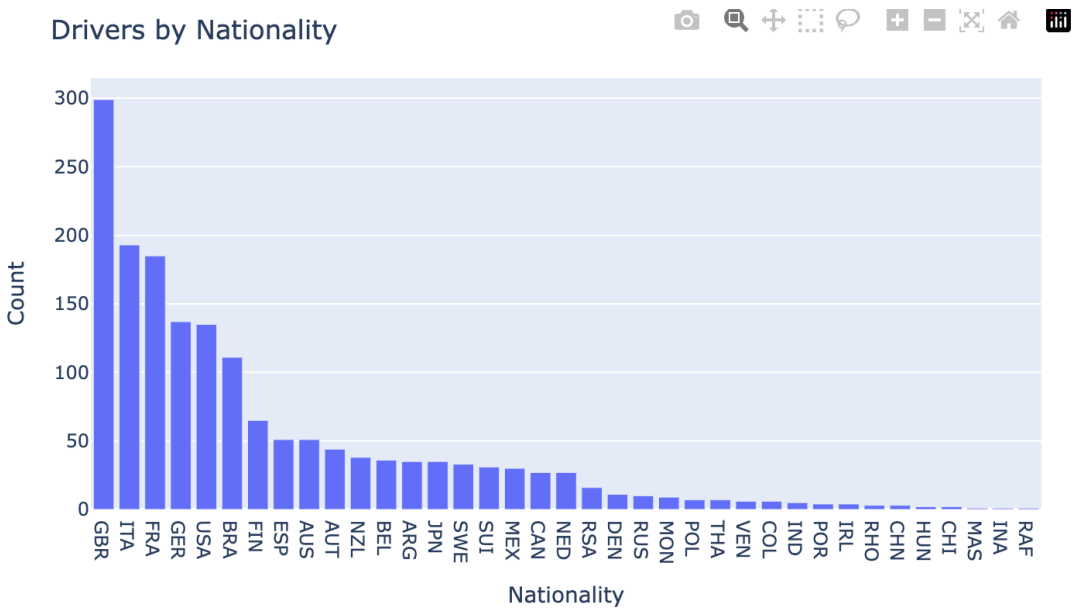


3.0.2 Bar Chart – Drivers by Nationality

This bar chart aggregates the number of drivers per nationality.
It provides a quick overview of the global distribution of F1 drivers over time.
- **x-axis**: Nationality
- **y-axis**: Number of Drivers
It's a static chart, useful for contextualizing diversity in the dataset.

[11]:
```
def static_nationality_bar():
    nationality_bar(drivers_df)
```

[12]:
```
static_nationality_bar()
```

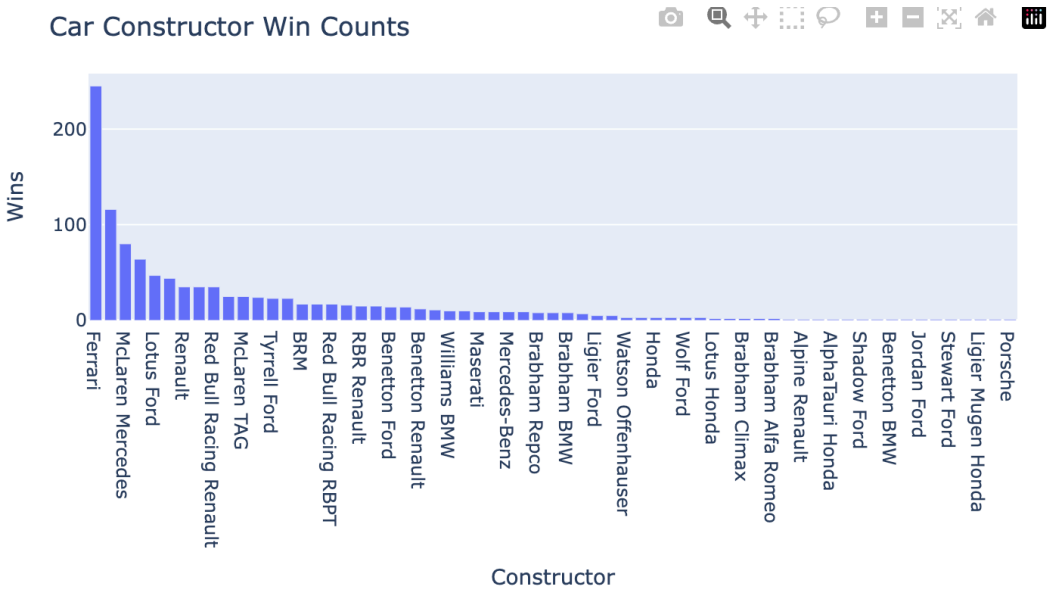


3.0.3 Bar Chart – Constructor Win Counts

This chart tallies how many races each car constructor has won.
It provides a macro-level view of team performance across seasons.
- **x-axis**: Constructor - **y-axis**: Total wins
This is a static chart and complements individual driver stats.

[13]:
```
def static_constructor_bar():
    constructor_bar(winners_df)
```

[14]:
```
static_constructor_bar()
```



3.0.4 Scatter Plot – Race Winners by Year and Circuit

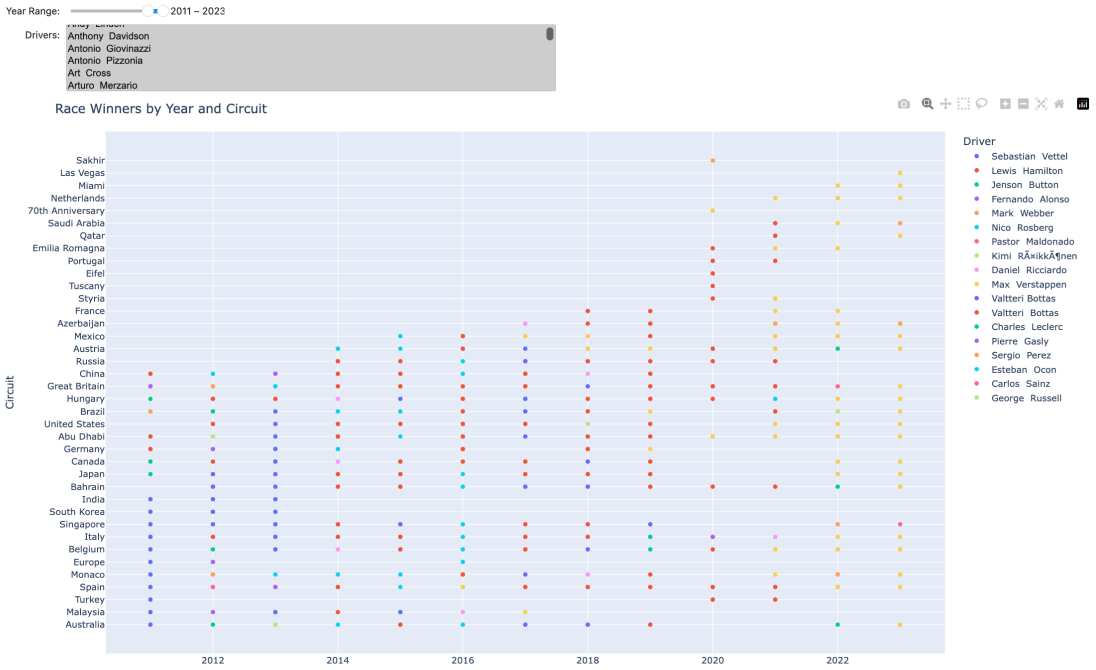
A scatter plot showing which drivers won races, and at which circuits, year by year.
- **x-axis**: Year of the race
- **y-axis**: Grand Prix circuit
- **Color**: Driver (Winner)

This chart is interactive and ideal for identifying patterns in race outcomes, such as repeat winners at specific circuits.

[15]:
```
def interactive_race_scatter():
    # Builds an interactive widget to filter and plot driver wins over time
    def update(year_range, selected_drivers):
        clear_output(wait=True)
        winners_filtered = winners_df.copy()
        winners_filtered['Date'] = pd.to_datetime(winners_filtered['Date'])
        winners_filtered['Year'] = winners_filtered['Date'].dt.year
        winners_filtered = winners_filtered[(winners_filtered['Year'] >=␣
↪year_range[0]) & (winners_filtered['Year'] <= year_range[1])]
        winners_filtered = winners_filtered[winners_filtered['Winner'].
↪isin(selected_drivers)]
        race_scatter(winners_filtered)
    # Displays the interactive controls and links them to the updating feature
    display(widgets.VBox([
        year_slider,
        driver_dropdown,
        widgets.interactive_output(update, {
            'year_range': year_slider,
            'selected_drivers': driver_dropdown
        })
    ]))
```

[16]:
```
interactive_race_scatter()
```

```
VBox(children=(IntRangeSlider(value=(1950, 2023), continuous_update=False,␣
↪description='Year Range:', max=2023…
```

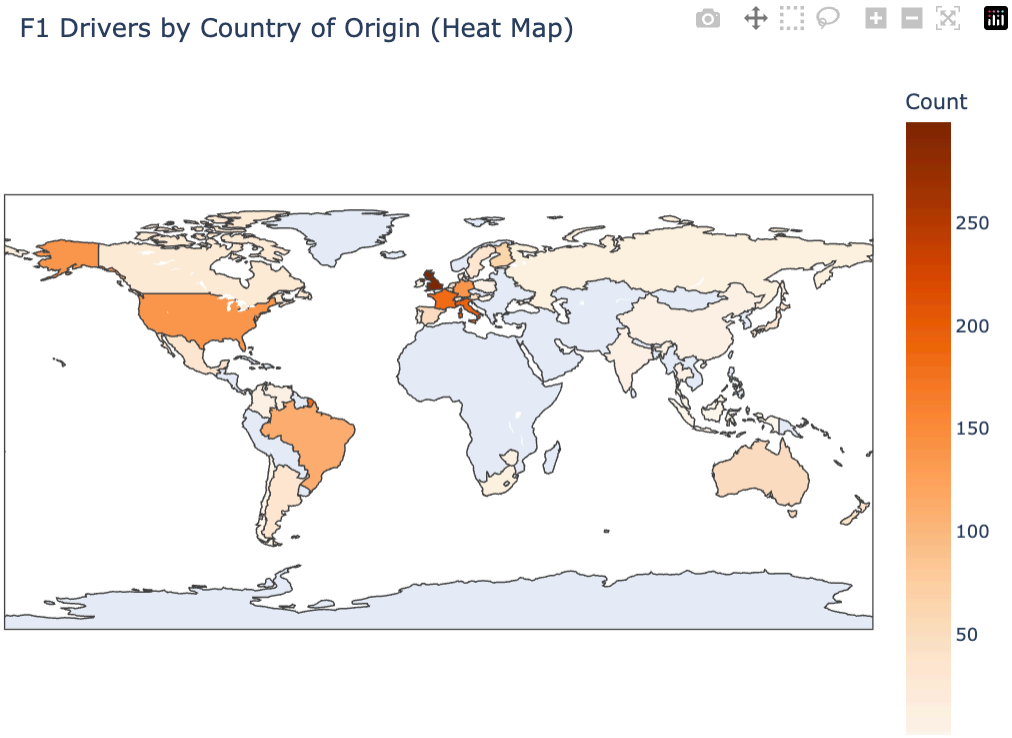


3.0.5 Geographic Heat Map – F1 Drivers by Country of Origin

This map shows the number of F1 drivers from each country, shaded using a color gradient. It works in tandem with the nationality bar chart, but this is more effective. It is easy to group by Global region/Continent - Darker shades represent more drivers from that country
- Uses nationality-to-country mapping to convert codes to readable names
This geographic chart helps tie the data to real-world geography and improves the diversity story visually.

[17]:
```
def static_geo_map():
    geo_map(drivers_df)
```

[18]:
```
static_geo_map()
```



4 Record Dependencies

Below are the dependencies for this notebook

[19]:
```
%load_ext watermark
%watermark -v -m -p pandas,numpy,ipywidgets,plotly
```

```
Python implementation: CPython
Python version       : 3.10.11
IPython version      : 8.13.2

pandas    : 2.0.2
numpy     : 1.24.3
ipywidgets: 8.0.6
plotly    : 6.0.0

Compiler    : GCC 11.3.0
OS          : Linux
Release     : 6.5.0-1020-aws
Machine     : x86_64
Processor   : x86_64
CPU cores   : 32
Architecture: 64bit
```