**Appendix A – Python Code**

# -\*- coding: utf-8 -\*--

"""

Created on Wed Oct 19 13:18:51 2022

@author: andre

"""

#%% notes

#can only go back to 2018 using the API, 2014-2017 leads to error:

#Cannot load laps, telemetry, weather, and message data because the relevant API is not supported for this session.

#%% installing packages

#pip install "fastf1"

#import fastf1

#installing libraries

import numpy as np

import pandas as pd

import fastf1 as ff1

from fastf1 import plotting

plotting.setup\_mpl()

import matplotlib.pyplot as plt

from matplotlib.collections import LineCollection

from matplotlib import cm

import plotly.express as px

import plotly.graph\_objs as go

from sklearn import linear\_model

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

from patsy import dmatrices

import statsmodels.api as sm

import statsmodels.formula.api as smf

import rpy2.robjects as ro

from rpy2.robjects import pandas2ri

#%%

import os

#%% redirecting OS and caching data for memory

os.getcwd()

#directing to cache folder for data storage

ff1.Cache.enable\_cache("C:\\Users\\andre\\OneDrive\\Desktop\\Thesis\\cache")

#%% getting event schedule by year

schedule18 = ff1.get\_event\_schedule(2018)

schedule19 = ff1.get\_event\_schedule(2019)

schedule20 = ff1.get\_event\_schedule(2020)

schedule21 = ff1.get\_event\_schedule(2021)

schedule22 = ff1.get\_event\_schedule(2022)

#%% loading 2018 data

australia\_race\_18 = ff1.get\_session(2018, 1, 'R')

australia\_race\_18.load()

australia\_race\_18\_laps = australia\_race\_18.laps

australia\_race\_18\_messages = australia\_race\_18.race\_control\_messages

australia\_race\_18\_weather = australia\_race\_18.weather\_data

australia\_race\_18\_results = australia\_race\_18.results

australia\_race\_18\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

australia\_race\_18\_final = pd.merge(australia\_race\_18\_laps, australia\_race\_18\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

australia\_race\_18\_final['TotalTime'] = australia\_race\_18\_final['Time'] - australia\_race\_18.session\_start\_time

australia\_race\_18\_final = australia\_race\_18\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

bahrain\_race\_18 = ff1.get\_session(2018, 2, 'R')

bahrain\_race\_18.load()

bahrain\_race\_18\_laps = bahrain\_race\_18.laps

bahrain\_race\_18\_messages = bahrain\_race\_18.race\_control\_messages

bahrain\_race\_18\_weather = bahrain\_race\_18.weather\_data

bahrain\_race\_18\_results = bahrain\_race\_18.results

bahrain\_race\_18\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

bahrain\_race\_18\_final = pd.merge(bahrain\_race\_18\_laps, bahrain\_race\_18\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

bahrain\_race\_18\_final['TotalTime'] = bahrain\_race\_18\_final['Time'] - bahrain\_race\_18.session\_start\_time

bahrain\_race\_18\_final = bahrain\_race\_18\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

china\_race\_18 = ff1.get\_session(2018, 3, 'R')

china\_race\_18.load()

china\_race\_18\_laps = china\_race\_18.laps

china\_race\_18\_messages = china\_race\_18.race\_control\_messages

china\_race\_18\_weather = china\_race\_18.weather\_data

china\_race\_18\_results = china\_race\_18.results

china\_race\_18\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

china\_race\_18\_final = pd.merge(china\_race\_18\_laps, china\_race\_18\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

china\_race\_18\_final['TotalTime'] = china\_race\_18\_final['Time'] - china\_race\_18.session\_start\_time

china\_race\_18\_final = china\_race\_18\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

azerbaijan\_race\_18 = ff1.get\_session(2018, 4, 'R')

azerbaijan\_race\_18.load()

azerbaijan\_race\_18\_laps = azerbaijan\_race\_18.laps

azerbaijan\_race\_18\_messages = azerbaijan\_race\_18.race\_control\_messages

azerbaijan\_race\_18\_weather = azerbaijan\_race\_18.weather\_data

azerbaijan\_race\_18\_results = azerbaijan\_race\_18.results

azerbaijan\_race\_18\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

azerbaijan\_race\_18\_final = pd.merge(azerbaijan\_race\_18\_laps, azerbaijan\_race\_18\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

azerbaijan\_race\_18\_final['TotalTime'] = azerbaijan\_race\_18\_final['Time'] - azerbaijan\_race\_18.session\_start\_time

azerbaijan\_race\_18\_final = azerbaijan\_race\_18\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

spain\_race\_18 = ff1.get\_session(2018, 5, 'R')

spain\_race\_18.load()

spain\_race\_18\_laps = spain\_race\_18.laps

spain\_race\_18\_messages = spain\_race\_18.race\_control\_messages

spain\_race\_18\_weather = spain\_race\_18.weather\_data

spain\_race\_18\_results = spain\_race\_18.results

spain\_race\_18\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

spain\_race\_18\_final = pd.merge(spain\_race\_18\_laps, spain\_race\_18\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

spain\_race\_18\_final['TotalTime'] = spain\_race\_18\_final['Time'] - spain\_race\_18.session\_start\_time

spain\_race\_18\_final = spain\_race\_18\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

monaco\_race\_18 = ff1.get\_session(2018, 6, 'R')

monaco\_race\_18.load()

monaco\_race\_18\_laps = monaco\_race\_18.laps

monaco\_race\_18\_messages = monaco\_race\_18.race\_control\_messages

monaco\_race\_18\_weather = monaco\_race\_18.weather\_data

monaco\_race\_18\_results = monaco\_race\_18.results

monaco\_race\_18\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

monaco\_race\_18\_final = pd.merge(monaco\_race\_18\_laps, monaco\_race\_18\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

monaco\_race\_18\_final['TotalTime'] = monaco\_race\_18\_final['Time'] - monaco\_race\_18.session\_start\_time

monaco\_race\_18\_final = monaco\_race\_18\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

canada\_race\_18 = ff1.get\_session(2018, 7, 'R')

canada\_race\_18.load()

canada\_race\_18\_laps = canada\_race\_18.laps

canada\_race\_18\_messages = canada\_race\_18.race\_control\_messages

canada\_race\_18\_weather = canada\_race\_18.weather\_data

canada\_race\_18\_results = canada\_race\_18.results

canada\_race\_18\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

canada\_race\_18\_final = pd.merge(canada\_race\_18\_laps, canada\_race\_18\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

canada\_race\_18\_final['TotalTime'] = canada\_race\_18\_final['Time'] - canada\_race\_18.session\_start\_time

canada\_race\_18\_final = canada\_race\_18\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

france\_race\_18 = ff1.get\_session(2018, 8, 'R')

france\_race\_18.load()

france\_race\_18\_laps = france\_race\_18.laps

france\_race\_18\_messages = france\_race\_18.race\_control\_messages

france\_race\_18\_weather = france\_race\_18.weather\_data

france\_race\_18\_results = france\_race\_18.results

france\_race\_18\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

france\_race\_18\_final = pd.merge(france\_race\_18\_laps, france\_race\_18\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

france\_race\_18\_final['TotalTime'] = france\_race\_18\_final['Time'] - france\_race\_18.session\_start\_time

france\_race\_18\_final = france\_race\_18\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

austria\_race\_18 = ff1.get\_session(2018, 9, 'R')

austria\_race\_18.load()

austria\_race\_18\_laps = austria\_race\_18.laps

austria\_race\_18\_messages = austria\_race\_18.race\_control\_messages

austria\_race\_18\_weather = austria\_race\_18.weather\_data

austria\_race\_18\_results = austria\_race\_18.results

austria\_race\_18\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

austria\_race\_18\_final = pd.merge(austria\_race\_18\_laps, austria\_race\_18\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

austria\_race\_18\_final['TotalTime'] = austria\_race\_18\_final['Time'] - austria\_race\_18.session\_start\_time

austria\_race\_18\_final = austria\_race\_18\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

great\_britain\_race\_18 = ff1.get\_session(2018, 10, 'R')

great\_britain\_race\_18.load()

great\_britain\_race\_18\_laps = great\_britain\_race\_18.laps

great\_britain\_race\_18\_messages = great\_britain\_race\_18.race\_control\_messages

great\_britain\_race\_18\_weather = great\_britain\_race\_18.weather\_data

great\_britain\_race\_18\_results = great\_britain\_race\_18.results

great\_britain\_race\_18\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

great\_britain\_race\_18\_final = pd.merge(great\_britain\_race\_18\_laps, great\_britain\_race\_18\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

great\_britain\_race\_18\_final['TotalTime'] = great\_britain\_race\_18\_final['Time'] - great\_britain\_race\_18.session\_start\_time

great\_britain\_race\_18\_final = great\_britain\_race\_18\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

germany\_race\_18 = ff1.get\_session(2018, 11, 'R')

germany\_race\_18.load()

germany\_race\_18\_laps = germany\_race\_18.laps

germany\_race\_18\_messages = germany\_race\_18.race\_control\_messages

germany\_race\_18\_weather = germany\_race\_18.weather\_data

germany\_race\_18\_results = germany\_race\_18.results

germany\_race\_18\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

germany\_race\_18\_final = pd.merge(germany\_race\_18\_laps, germany\_race\_18\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

germany\_race\_18\_final['TotalTime'] = germany\_race\_18\_final['Time'] - germany\_race\_18.session\_start\_time

germany\_race\_18\_final = germany\_race\_18\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

hungary\_race\_18 = ff1.get\_session(2018, 12, 'R')

hungary\_race\_18.load()

hungary\_race\_18\_laps = hungary\_race\_18.laps

hungary\_race\_18\_messages = hungary\_race\_18.race\_control\_messages

hungary\_race\_18\_weather = hungary\_race\_18.weather\_data

hungary\_race\_18\_results = hungary\_race\_18.results

hungary\_race\_18\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

hungary\_race\_18\_final = pd.merge(hungary\_race\_18\_laps, hungary\_race\_18\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

hungary\_race\_18\_final['TotalTime'] = hungary\_race\_18\_final['Time'] - hungary\_race\_18.session\_start\_time

hungary\_race\_18\_final = hungary\_race\_18\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

belgium\_race\_18 = ff1.get\_session(2018, 13, 'R')

belgium\_race\_18.load()

belgium\_race\_18\_laps = belgium\_race\_18.laps

belgium\_race\_18\_messages = belgium\_race\_18.race\_control\_messages

belgium\_race\_18\_weather = belgium\_race\_18.weather\_data

belgium\_race\_18\_results = belgium\_race\_18.results

belgium\_race\_18\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

belgium\_race\_18\_final = pd.merge(belgium\_race\_18\_laps, belgium\_race\_18\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

belgium\_race\_18\_final['TotalTime'] = belgium\_race\_18\_final['Time'] - belgium\_race\_18.session\_start\_time

belgium\_race\_18\_final = belgium\_race\_18\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

italy\_race\_18 = ff1.get\_session(2018, 14, 'R')

italy\_race\_18.load()

italy\_race\_18\_laps = italy\_race\_18.laps

italy\_race\_18\_messages = italy\_race\_18.race\_control\_messages

italy\_race\_18\_weather = italy\_race\_18.weather\_data

italy\_race\_18\_results = italy\_race\_18.results

italy\_race\_18\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

italy\_race\_18\_final = pd.merge(italy\_race\_18\_laps, italy\_race\_18\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

italy\_race\_18\_final['TotalTime'] = italy\_race\_18\_final['Time'] - italy\_race\_18.session\_start\_time

italy\_race\_18\_final = italy\_race\_18\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

singapore\_race\_18 = ff1.get\_session(2018, 15, 'R')

singapore\_race\_18.load()

singapore\_race\_18\_laps = singapore\_race\_18.laps

singapore\_race\_18\_messages = singapore\_race\_18.race\_control\_messages

singapore\_race\_18\_weather = singapore\_race\_18.weather\_data

singapore\_race\_18\_results = singapore\_race\_18.results

singapore\_race\_18\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

singapore\_race\_18\_final = pd.merge(singapore\_race\_18\_laps, singapore\_race\_18\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

singapore\_race\_18\_final['TotalTime'] = singapore\_race\_18\_final['Time'] - singapore\_race\_18.session\_start\_time

singapore\_race\_18\_final = singapore\_race\_18\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

russia\_race\_18 = ff1.get\_session(2018, 16, 'R')

russia\_race\_18.load()

russia\_race\_18\_laps = russia\_race\_18.laps

russia\_race\_18\_messages = russia\_race\_18.race\_control\_messages

russia\_race\_18\_weather = russia\_race\_18.weather\_data

russia\_race\_18\_results = russia\_race\_18.results

russia\_race\_18\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

russia\_race\_18\_final = pd.merge(russia\_race\_18\_laps, russia\_race\_18\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

russia\_race\_18\_final['TotalTime'] = russia\_race\_18\_final['Time'] - russia\_race\_18.session\_start\_time

russia\_race\_18\_final = russia\_race\_18\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

japan\_race\_18 = ff1.get\_session(2018, 17, 'R')

japan\_race\_18.load()

japan\_race\_18\_laps = japan\_race\_18.laps

japan\_race\_18\_messages = japan\_race\_18.race\_control\_messages

japan\_race\_18\_weather = japan\_race\_18.weather\_data

japan\_race\_18\_results = japan\_race\_18.results

japan\_race\_18\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

japan\_race\_18\_final = pd.merge(japan\_race\_18\_laps, japan\_race\_18\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

japan\_race\_18\_final['TotalTime'] = japan\_race\_18\_final['Time'] - japan\_race\_18.session\_start\_time

japan\_race\_18\_final = japan\_race\_18\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

united\_states\_race\_18 = ff1.get\_session(2018, 18, 'R')

united\_states\_race\_18.load()

united\_states\_race\_18\_laps = united\_states\_race\_18.laps

united\_states\_race\_18\_messages = united\_states\_race\_18.race\_control\_messages

united\_states\_race\_18\_weather = united\_states\_race\_18.weather\_data

united\_states\_race\_18\_results = united\_states\_race\_18.results

united\_states\_race\_18\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

united\_states\_race\_18\_final = pd.merge(united\_states\_race\_18\_laps, united\_states\_race\_18\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

united\_states\_race\_18\_final['TotalTime'] = united\_states\_race\_18\_final['Time'] - united\_states\_race\_18.session\_start\_time

united\_states\_race\_18\_final = united\_states\_race\_18\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

mexico\_race\_18 = ff1.get\_session(2018, 19, 'R')

mexico\_race\_18.load()

mexico\_race\_18\_laps = mexico\_race\_18.laps

mexico\_race\_18\_messages = mexico\_race\_18.race\_control\_messages

mexico\_race\_18\_weather = mexico\_race\_18.weather\_data

mexico\_race\_18\_results = mexico\_race\_18.results

mexico\_race\_18\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

mexico\_race\_18\_final = pd.merge(mexico\_race\_18\_laps, mexico\_race\_18\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

mexico\_race\_18\_final['TotalTime'] = mexico\_race\_18\_final['Time'] - mexico\_race\_18.session\_start\_time

mexico\_race\_18\_final = mexico\_race\_18\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

brazil\_race\_18 = ff1.get\_session(2018, 20, 'R')

brazil\_race\_18.load()

brazil\_race\_18\_laps = brazil\_race\_18.laps

brazil\_race\_18\_messages = brazil\_race\_18.race\_control\_messages

brazil\_race\_18\_weather = brazil\_race\_18.weather\_data

brazil\_race\_18\_results = brazil\_race\_18.results

brazil\_race\_18\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

brazil\_race\_18\_final = pd.merge(brazil\_race\_18\_laps, brazil\_race\_18\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

brazil\_race\_18\_final['TotalTime'] = brazil\_race\_18\_final['Time'] - brazil\_race\_18.session\_start\_time

brazil\_race\_18\_final = brazil\_race\_18\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

united\_arab\_emirates\_race\_18 = ff1.get\_session(2018, 21, 'R')

united\_arab\_emirates\_race\_18.load()

united\_arab\_emirates\_race\_18\_laps = united\_arab\_emirates\_race\_18.laps

united\_arab\_emirates\_race\_18\_messages = united\_arab\_emirates\_race\_18.race\_control\_messages

united\_arab\_emirates\_race\_18\_weather = united\_arab\_emirates\_race\_18.weather\_data

united\_arab\_emirates\_race\_18\_results = united\_arab\_emirates\_race\_18.results

united\_arab\_emirates\_race\_18\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

united\_arab\_emirates\_race\_18\_final = pd.merge(united\_arab\_emirates\_race\_18\_laps, united\_arab\_emirates\_race\_18\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

united\_arab\_emirates\_race\_18\_final['TotalTime'] = united\_arab\_emirates\_race\_18\_final['Time'] - united\_arab\_emirates\_race\_18.session\_start\_time

united\_arab\_emirates\_race\_18\_final = united\_arab\_emirates\_race\_18\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

merge2018\_laps = pd.concat([

australia\_race\_18\_laps,

bahrain\_race\_18\_laps,

china\_race\_18\_laps,

azerbaijan\_race\_18\_laps,

spain\_race\_18\_laps,

monaco\_race\_18\_laps,

canada\_race\_18\_laps,

france\_race\_18\_laps,

austria\_race\_18\_laps,

great\_britain\_race\_18\_laps,

germany\_race\_18\_laps,

hungary\_race\_18\_laps,

belgium\_race\_18\_laps,

italy\_race\_18\_laps,

singapore\_race\_18\_laps,

russia\_race\_18\_laps,

japan\_race\_18\_laps,

united\_states\_race\_18\_laps,

mexico\_race\_18\_laps,

brazil\_race\_18\_laps,

united\_arab\_emirates\_race\_18\_laps],

keys = ['Australia',

'Bahrain',

'China',

'Azerbaijan',

'Spain',

'Monaco',

'Canada',

'France',

'Austria',

'GreatBritain',

'Germany',

'Hungary',

'Belgium',

'Italy',

'Singapore',

'Russia',

'Japan',

'UnitedStates',

'Mexico',

'Brazil',

'UnitedArabEmirates'],

names=['RACE'])

merge2018\_messages = pd.concat([

australia\_race\_18\_messages,

bahrain\_race\_18\_messages,

china\_race\_18\_messages,

azerbaijan\_race\_18\_messages,

spain\_race\_18\_messages,

monaco\_race\_18\_messages,

canada\_race\_18\_messages,

france\_race\_18\_messages,

austria\_race\_18\_messages,

great\_britain\_race\_18\_messages,

germany\_race\_18\_messages,

hungary\_race\_18\_messages,

belgium\_race\_18\_messages,

italy\_race\_18\_messages,

singapore\_race\_18\_messages,

russia\_race\_18\_messages,

japan\_race\_18\_messages,

united\_states\_race\_18\_messages,

mexico\_race\_18\_messages,

brazil\_race\_18\_messages,

united\_arab\_emirates\_race\_18\_messages],

keys = ['Australia',

'Bahrain',

'China',

'Azerbaijan',

'Spain',

'Monaco',

'Canada',

'France',

'Austria',

'GreatBritain',

'Germany',

'Hungary',

'Belgium',

'Italy',

'Singapore',

'Russia',

'Japan',

'UnitedStates',

'Mexico',

'Brazil',

'UnitedArabEmirates'],

names=['RACE'])

merge2018\_weather = pd.concat([

australia\_race\_18\_weather,

bahrain\_race\_18\_weather,

china\_race\_18\_weather,

azerbaijan\_race\_18\_weather,

spain\_race\_18\_weather,

monaco\_race\_18\_weather,

canada\_race\_18\_weather,

france\_race\_18\_weather,

austria\_race\_18\_weather,

great\_britain\_race\_18\_weather,

germany\_race\_18\_weather,

hungary\_race\_18\_weather,

belgium\_race\_18\_weather,

italy\_race\_18\_weather,

singapore\_race\_18\_weather,

russia\_race\_18\_weather,

japan\_race\_18\_weather,

united\_states\_race\_18\_weather,

mexico\_race\_18\_weather,

brazil\_race\_18\_weather,

united\_arab\_emirates\_race\_18\_weather],

keys = ['Australia',

'Bahrain',

'China',

'Azerbaijan',

'Spain',

'Monaco',

'Canada',

'France',

'Austria',

'GreatBritain',

'Germany',

'Hungary',

'Belgium',

'Italy',

'Singapore',

'Russia',

'Japan',

'UnitedStates',

'Mexico',

'Brazil',

'UnitedArabEmirates'],

names=['RACE'])

merge2018\_results = pd.concat([

australia\_race\_18\_results,

bahrain\_race\_18\_results,

china\_race\_18\_results,

azerbaijan\_race\_18\_results,

spain\_race\_18\_results,

monaco\_race\_18\_results,

canada\_race\_18\_results,

france\_race\_18\_results,

austria\_race\_18\_results,

great\_britain\_race\_18\_results,

germany\_race\_18\_results,

hungary\_race\_18\_results,

belgium\_race\_18\_results,

italy\_race\_18\_results,

singapore\_race\_18\_results,

russia\_race\_18\_results,

japan\_race\_18\_results,

united\_states\_race\_18\_results,

mexico\_race\_18\_results,

brazil\_race\_18\_results,

united\_arab\_emirates\_race\_18\_results],

keys = ['Australia',

'Bahrain',

'China',

'Azerbaijan',

'Spain',

'Monaco',

'Canada',

'France',

'Austria',

'GreatBritain',

'Germany',

'Hungary',

'Belgium',

'Italy',

'Singapore',

'Russia',

'Japan',

'UnitedStates',

'Mexico',

'Brazil',

'UnitedArabEmirates'],

names=['RACE'])

merge2018\_final = pd.concat([

australia\_race\_18\_final,

bahrain\_race\_18\_final,

china\_race\_18\_final,

azerbaijan\_race\_18\_final,

spain\_race\_18\_final,

monaco\_race\_18\_final,

canada\_race\_18\_final,

france\_race\_18\_final,

austria\_race\_18\_final,

great\_britain\_race\_18\_final,

germany\_race\_18\_final,

hungary\_race\_18\_final,

belgium\_race\_18\_final,

italy\_race\_18\_final,

singapore\_race\_18\_final,

russia\_race\_18\_final,

japan\_race\_18\_final,

united\_states\_race\_18\_final,

mexico\_race\_18\_final,

brazil\_race\_18\_final,

united\_arab\_emirates\_race\_18\_final],

keys = ['Australia',

'Bahrain',

'China',

'Azerbaijan',

'Spain',

'Monaco',

'Canada',

'France',

'Austria',

'GreatBritain',

'Germany',

'Hungary',

'Belgium',

'Italy',

'Singapore',

'Russia',

'Japan',

'UnitedStates',

'Mexico',

'Brazil',

'UnitedArabEmirates'],

names=['RACE'])

merge2018\_laps["Year"] = 2018

merge2018\_messages["Year"] = 2018

merge2018\_weather["Year"] = 2018

merge2018\_results["Year"] = 2018

merge2018\_final["Year"] = 2018

#%% loading 2019 data

australia\_race\_19 = ff1.get\_session(2019, 1, 'R')

australia\_race\_19.load()

australia\_race\_19\_laps = australia\_race\_19.laps

australia\_race\_19\_messages = australia\_race\_19.race\_control\_messages

australia\_race\_19\_weather = australia\_race\_19.weather\_data

australia\_race\_19\_results = australia\_race\_19.results

australia\_race\_19\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

australia\_race\_19\_final = pd.merge(australia\_race\_19\_laps, australia\_race\_19\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

australia\_race\_19\_final['TotalTime'] = australia\_race\_19\_final['Time'] - australia\_race\_19.session\_start\_time

australia\_race\_19\_final = australia\_race\_19\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

bahrain\_race\_19 = ff1.get\_session(2019, 2, 'R')

bahrain\_race\_19.load()

bahrain\_race\_19\_laps = bahrain\_race\_19.laps

bahrain\_race\_19\_messages = bahrain\_race\_19.race\_control\_messages

bahrain\_race\_19\_weather = bahrain\_race\_19.weather\_data

bahrain\_race\_19\_results = bahrain\_race\_19.results

bahrain\_race\_19\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

bahrain\_race\_19\_final = pd.merge(bahrain\_race\_19\_laps, bahrain\_race\_19\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

bahrain\_race\_19\_final['TotalTime'] = bahrain\_race\_19\_final['Time'] - bahrain\_race\_19.session\_start\_time

bahrain\_race\_19\_final = bahrain\_race\_19\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

china\_race\_19 = ff1.get\_session(2019, 3, 'R')

china\_race\_19.load()

china\_race\_19\_laps = china\_race\_19.laps

china\_race\_19\_messages= china\_race\_19.race\_control\_messages

china\_race\_19\_weather = china\_race\_19.weather\_data

china\_race\_19\_results = china\_race\_19.results

china\_race\_19\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

china\_race\_19\_final = pd.merge(china\_race\_19\_laps, china\_race\_19\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

china\_race\_19\_final['TotalTime'] = china\_race\_19\_final['Time'] - china\_race\_19.session\_start\_time

china\_race\_19\_final = china\_race\_19\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

azerbaijan\_race\_19 = ff1.get\_session(2019, 4, 'R')

azerbaijan\_race\_19.load()

azerbaijan\_race\_19\_laps = azerbaijan\_race\_19.laps

azerbaijan\_race\_19\_messages = azerbaijan\_race\_19.race\_control\_messages

azerbaijan\_race\_19\_weather = azerbaijan\_race\_19.weather\_data

azerbaijan\_race\_19\_results = azerbaijan\_race\_19.results

azerbaijan\_race\_19\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

azerbaijan\_race\_19\_final = pd.merge(azerbaijan\_race\_19\_laps, azerbaijan\_race\_19\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

azerbaijan\_race\_19\_final['TotalTime'] = azerbaijan\_race\_19\_final['Time'] - azerbaijan\_race\_19.session\_start\_time

azerbaijan\_race\_19\_final = azerbaijan\_race\_19\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

spain\_race\_19 = ff1.get\_session(2019, 5, 'R')

spain\_race\_19.load()

spain\_race\_19\_laps = spain\_race\_19.laps

spain\_race\_19\_messages = spain\_race\_19.race\_control\_messages

spain\_race\_19\_weather = spain\_race\_19.weather\_data

spain\_race\_19\_results = spain\_race\_19.results

spain\_race\_19\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

spain\_race\_19\_final = pd.merge(spain\_race\_19\_laps, spain\_race\_19\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

spain\_race\_19\_final['TotalTime'] = spain\_race\_19\_final['Time'] - spain\_race\_19.session\_start\_time

spain\_race\_19\_final = spain\_race\_19\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

monaco\_race\_19 = ff1.get\_session(2019, 6, 'R')

monaco\_race\_19.load()

monaco\_race\_19\_laps = monaco\_race\_19.laps

monaco\_race\_19\_messages = monaco\_race\_19.race\_control\_messages

monaco\_race\_19\_weather = monaco\_race\_19.weather\_data

monaco\_race\_19\_results = monaco\_race\_19.results

monaco\_race\_19\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

monaco\_race\_19\_final = pd.merge(monaco\_race\_19\_laps, monaco\_race\_19\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

monaco\_race\_19\_final['TotalTime'] = monaco\_race\_19\_final['Time'] - monaco\_race\_19.session\_start\_time

monaco\_race\_19\_final = monaco\_race\_19\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

canada\_race\_19 = ff1.get\_session(2019, 7, 'R')

canada\_race\_19.load()

canada\_race\_19\_laps = canada\_race\_19.laps

canada\_race\_19\_messages = canada\_race\_19.race\_control\_messages

canada\_race\_19\_weather = canada\_race\_19.weather\_data

canada\_race\_19\_results = canada\_race\_19.results

canada\_race\_19\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

canada\_race\_19\_final = pd.merge(canada\_race\_19\_laps, canada\_race\_19\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

canada\_race\_19\_final['TotalTime'] = canada\_race\_19\_final['Time'] - canada\_race\_19.session\_start\_time

canada\_race\_19\_final = canada\_race\_19\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

france\_race\_19 = ff1.get\_session(2019, 8, 'R')

france\_race\_19.load()

france\_race\_19\_laps = france\_race\_19.laps

france\_race\_19\_messages = france\_race\_19.race\_control\_messages

france\_race\_19\_weather = france\_race\_19.weather\_data

france\_race\_19\_results = france\_race\_19.results

france\_race\_19\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

france\_race\_19\_final = pd.merge(france\_race\_19\_laps, france\_race\_19\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

france\_race\_19\_final['TotalTime'] = france\_race\_19\_final['Time'] - france\_race\_19.session\_start\_time

france\_race\_19\_final = france\_race\_19\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

austria\_race\_19 = ff1.get\_session(2019, 9, 'R')

austria\_race\_19.load()

austria\_race\_19\_laps = austria\_race\_19.laps

austria\_race\_19\_messages = austria\_race\_19.race\_control\_messages

austria\_race\_19\_weather = austria\_race\_19.weather\_data

austria\_race\_19\_results = austria\_race\_19.results

austria\_race\_19\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

austria\_race\_19\_final = pd.merge(austria\_race\_19\_laps, austria\_race\_19\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

austria\_race\_19\_final['TotalTime'] = austria\_race\_19\_final['Time'] - austria\_race\_19.session\_start\_time

austria\_race\_19\_final = austria\_race\_19\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

great\_britain\_race\_19 = ff1.get\_session(2019, 10, 'R')

great\_britain\_race\_19.load()

great\_britain\_race\_19\_laps = great\_britain\_race\_19.laps

great\_britain\_race\_19\_messages = great\_britain\_race\_19.race\_control\_messages

great\_britain\_race\_19\_weather = great\_britain\_race\_19.weather\_data

great\_britain\_race\_19\_results = great\_britain\_race\_19.results

great\_britain\_race\_19\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

great\_britain\_race\_19\_final = pd.merge(great\_britain\_race\_19\_laps, great\_britain\_race\_19\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

great\_britain\_race\_19\_final['TotalTime'] = great\_britain\_race\_19\_final['Time'] - great\_britain\_race\_19.session\_start\_time

great\_britain\_race\_19\_final = great\_britain\_race\_19\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

germany\_race\_19 = ff1.get\_session(2019, 11, 'R')

germany\_race\_19.load()

germany\_race\_19\_laps = germany\_race\_19.laps

germany\_race\_19\_messages = germany\_race\_19.race\_control\_messages

germany\_race\_19\_weather = germany\_race\_19.weather\_data

germany\_race\_19\_results = germany\_race\_19.results

germany\_race\_19\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

germany\_race\_19\_final = pd.merge(germany\_race\_19\_laps, germany\_race\_19\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

germany\_race\_19\_final['TotalTime'] = germany\_race\_19\_final['Time'] - germany\_race\_19.session\_start\_time

germany\_race\_19\_final = germany\_race\_19\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

hungary\_race\_19 = ff1.get\_session(2019, 12, 'R')

hungary\_race\_19.load()

hungary\_race\_19\_laps = hungary\_race\_19.laps

hungary\_race\_19\_messages = hungary\_race\_19.race\_control\_messages

hungary\_race\_19\_weather = hungary\_race\_19.weather\_data

hungary\_race\_19\_results = hungary\_race\_19.results

hungary\_race\_19\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

hungary\_race\_19\_final = pd.merge(hungary\_race\_19\_laps, hungary\_race\_19\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

hungary\_race\_19\_final['TotalTime'] = hungary\_race\_19\_final['Time'] - hungary\_race\_19.session\_start\_time

hungary\_race\_19\_final = hungary\_race\_19\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

belgium\_race\_19 = ff1.get\_session(2019, 13, 'R')

belgium\_race\_19.load()

belgium\_race\_19\_laps = belgium\_race\_19.laps

belgium\_race\_19\_messages = belgium\_race\_19.race\_control\_messages

belgium\_race\_19\_weather = belgium\_race\_19.weather\_data

belgium\_race\_19\_results = belgium\_race\_19.results

belgium\_race\_19\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

belgium\_race\_19\_final = pd.merge(belgium\_race\_19\_laps, belgium\_race\_19\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

belgium\_race\_19\_final['TotalTime'] = belgium\_race\_19\_final['Time'] - belgium\_race\_19.session\_start\_time

belgium\_race\_19\_final = belgium\_race\_19\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

italy\_race\_19 = ff1.get\_session(2019, 14, 'R')

italy\_race\_19.load()

italy\_race\_19\_laps = italy\_race\_19.laps

italy\_race\_19\_messages = italy\_race\_19.race\_control\_messages

italy\_race\_19\_weather = italy\_race\_19.weather\_data

italy\_race\_19\_results = italy\_race\_19.results

italy\_race\_19\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

italy\_race\_19\_final = pd.merge(italy\_race\_19\_laps, italy\_race\_19\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

italy\_race\_19\_final['TotalTime'] = italy\_race\_19\_final['Time'] - italy\_race\_19.session\_start\_time

italy\_race\_19\_final = italy\_race\_19\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

singapore\_race\_19 = ff1.get\_session(2019, 15, 'R')

singapore\_race\_19.load()

singapore\_race\_19\_laps = singapore\_race\_19.laps

singapore\_race\_19\_messages = singapore\_race\_19.race\_control\_messages

singapore\_race\_19\_weather = singapore\_race\_19.weather\_data

singapore\_race\_19\_results = singapore\_race\_19.results

singapore\_race\_19\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

singapore\_race\_19\_final = pd.merge(singapore\_race\_19\_laps, singapore\_race\_19\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

singapore\_race\_19\_final['TotalTime'] = singapore\_race\_19\_final['Time'] - singapore\_race\_19.session\_start\_time

singapore\_race\_19\_final = singapore\_race\_19\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

russia\_race\_19 = ff1.get\_session(2018, 16, 'R')

russia\_race\_19.load()

russia\_race\_19\_laps = russia\_race\_19.laps

russia\_race\_19\_messages = russia\_race\_19.race\_control\_messages

russia\_race\_19\_weather = russia\_race\_19.weather\_data

russia\_race\_19\_results = russia\_race\_19.results

russia\_race\_19\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

russia\_race\_19\_final = pd.merge(russia\_race\_19\_laps, russia\_race\_19\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

russia\_race\_19\_final['TotalTime'] = russia\_race\_19\_final['Time'] - russia\_race\_19.session\_start\_time

russia\_race\_19\_final = russia\_race\_19\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

japan\_race\_19 = ff1.get\_session(2019, 17, 'R')

japan\_race\_19.load()

japan\_race\_19\_laps = japan\_race\_19.laps

japan\_race\_19\_messages = japan\_race\_19.race\_control\_messages

japan\_race\_19\_weather = japan\_race\_19.weather\_data

japan\_race\_19\_results = japan\_race\_19.results

japan\_race\_19\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

japan\_race\_19\_final = pd.merge(japan\_race\_19\_laps, japan\_race\_19\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

japan\_race\_19\_final['TotalTime'] = japan\_race\_19\_final['Time'] - japan\_race\_19.session\_start\_time

japan\_race\_19\_final = japan\_race\_19\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

mexico\_race\_19 = ff1.get\_session(2019, 18, 'R')

mexico\_race\_19.load()

mexico\_race\_19\_laps = mexico\_race\_19.laps

mexico\_race\_19\_messages = mexico\_race\_19.race\_control\_messages

mexico\_race\_19\_weather = mexico\_race\_19.weather\_data

mexico\_race\_19\_results = mexico\_race\_19.results

mexico\_race\_19\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

mexico\_race\_19\_final = pd.merge(mexico\_race\_19\_laps, mexico\_race\_19\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

mexico\_race\_19\_final['TotalTime'] = mexico\_race\_19\_final['Time'] - mexico\_race\_19.session\_start\_time

mexico\_race\_19\_final = mexico\_race\_19\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

united\_states\_race\_19 = ff1.get\_session(2019, 19, 'R')

united\_states\_race\_19.load()

united\_states\_race\_19\_laps = united\_states\_race\_19.laps

united\_states\_race\_19\_messages = united\_states\_race\_19.race\_control\_messages

united\_states\_race\_19\_weather = united\_states\_race\_19.weather\_data

united\_states\_race\_19\_results = united\_states\_race\_19.results

united\_states\_race\_19\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

united\_states\_race\_19\_final = pd.merge(united\_states\_race\_19\_laps, united\_states\_race\_19\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

united\_states\_race\_19\_final['TotalTime'] = united\_states\_race\_19\_final['Time'] - united\_states\_race\_19.session\_start\_time

united\_states\_race\_19\_final = united\_states\_race\_19\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

brazil\_race\_19 = ff1.get\_session(2019, 20, 'R')

brazil\_race\_19.load()

brazil\_race\_19\_laps = brazil\_race\_19.laps

brazil\_race\_19\_messages = brazil\_race\_19.race\_control\_messages

brazil\_race\_19\_weather = brazil\_race\_19.weather\_data

brazil\_race\_19\_results = brazil\_race\_19.results

brazil\_race\_19\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

brazil\_race\_19\_final = pd.merge(brazil\_race\_19\_laps, brazil\_race\_19\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

brazil\_race\_19\_final['TotalTime'] = brazil\_race\_19\_final['Time'] - brazil\_race\_19.session\_start\_time

brazil\_race\_19\_final = brazil\_race\_19\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

abu\_dhabi\_race\_19 = ff1.get\_session(2019, 21, 'R')

abu\_dhabi\_race\_19.load()

abu\_dhabi\_race\_19\_laps = abu\_dhabi\_race\_19.laps

abu\_dhabi\_race\_19\_messages = abu\_dhabi\_race\_19.race\_control\_messages

abu\_dhabi\_race\_19\_weather = abu\_dhabi\_race\_19.weather\_data

abu\_dhabi\_race\_19\_results = abu\_dhabi\_race\_19.results

abu\_dhabi\_race\_19\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

abu\_dhabi\_race\_19\_final = pd.merge(abu\_dhabi\_race\_19\_laps, abu\_dhabi\_race\_19\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

abu\_dhabi\_race\_19\_final['TotalTime'] = abu\_dhabi\_race\_19\_final['Time'] - abu\_dhabi\_race\_19.session\_start\_time

abu\_dhabi\_race\_19\_final = abu\_dhabi\_race\_19\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

merge2019\_laps = pd.concat([

australia\_race\_19\_laps,

bahrain\_race\_19\_laps,

china\_race\_19\_laps,

azerbaijan\_race\_19\_laps,

spain\_race\_19\_laps,

monaco\_race\_19\_laps,

canada\_race\_19\_laps,

france\_race\_19\_laps,

austria\_race\_19\_laps,

great\_britain\_race\_19\_laps,

germany\_race\_19\_laps,

hungary\_race\_19\_laps,

belgium\_race\_19\_laps,

italy\_race\_19\_laps,

singapore\_race\_19\_laps,

russia\_race\_19\_laps,

japan\_race\_19\_laps,

mexico\_race\_19\_laps,

united\_states\_race\_19\_laps,

brazil\_race\_19\_laps,

abu\_dhabi\_race\_19\_laps],

keys = ['Australia',

'Bahrain',

'China',

'Azerbaijan',

'Spain',

'Monaco',

'Canada',

'France',

'Austria',

'GreatBritain',

'Germany',

'Hungary',

'Belgium',

'Italy',

'Singapore',

'Russia',

'Japan',

'Mexico',

'UnitedStates',

'Brazil',

'AbuDhabi'],

names=['RACE'])

merge2019\_messages = pd.concat([

australia\_race\_19\_messages,

bahrain\_race\_19\_messages,

china\_race\_19\_messages,

azerbaijan\_race\_19\_messages,

spain\_race\_19\_messages,

monaco\_race\_19\_messages,

canada\_race\_19\_messages,

france\_race\_19\_messages,

austria\_race\_19\_messages,

great\_britain\_race\_19\_messages,

germany\_race\_19\_messages,

hungary\_race\_19\_messages,

belgium\_race\_19\_messages,

italy\_race\_19\_messages,

singapore\_race\_19\_messages,

russia\_race\_19\_messages,

japan\_race\_19\_messages,

mexico\_race\_19\_messages,

united\_states\_race\_19\_messages,

brazil\_race\_19\_messages,

abu\_dhabi\_race\_19\_messages],

keys = ['Australia',

'Bahrain',

'China',

'Azerbaijan',

'Spain',

'Monaco',

'Canada',

'France',

'Austria',

'GreatBritain',

'Germany',

'Hungary',

'Belgium',

'Italy',

'Singapore',

'Russia',

'Japan',

'Mexico',

'UnitedStates',

'Brazil',

'AbuDhabi'],

names=['RACE'])

merge2019\_weather = pd.concat([

australia\_race\_19\_weather,

bahrain\_race\_19\_weather,

china\_race\_19\_weather,

azerbaijan\_race\_19\_weather,

spain\_race\_19\_weather,

monaco\_race\_19\_weather,

canada\_race\_19\_weather,

france\_race\_19\_weather,

austria\_race\_19\_weather,

great\_britain\_race\_19\_weather,

germany\_race\_19\_weather,

hungary\_race\_19\_weather,

belgium\_race\_19\_weather,

italy\_race\_19\_weather,

singapore\_race\_19\_weather,

russia\_race\_19\_weather,

japan\_race\_19\_weather,

mexico\_race\_19\_weather,

united\_states\_race\_19\_weather,

brazil\_race\_19\_weather,

abu\_dhabi\_race\_19\_weather],

keys = ['Australia',

'Bahrain',

'China',

'Azerbaijan',

'Spain',

'Monaco',

'Canada',

'France',

'Austria',

'GreatBritain',

'Germany',

'Hungary',

'Belgium',

'Italy',

'Singapore',

'Russia',

'Japan',

'Mexico',

'UnitedStates',

'Brazil',

'AbuDhabi'],

names=['RACE'])

merge2019\_results = pd.concat([

australia\_race\_19\_results ,

bahrain\_race\_19\_results ,

china\_race\_19\_results ,

azerbaijan\_race\_19\_results ,

spain\_race\_19\_results ,

monaco\_race\_19\_results ,

canada\_race\_19\_results ,

france\_race\_19\_results ,

austria\_race\_19\_results ,

great\_britain\_race\_19\_results ,

germany\_race\_19\_results ,

hungary\_race\_19\_results ,

belgium\_race\_19\_results ,

italy\_race\_19\_results ,

singapore\_race\_19\_results,

russia\_race\_19\_results ,

japan\_race\_19\_results ,

mexico\_race\_19\_results ,

united\_states\_race\_19\_results ,

brazil\_race\_19\_results ,

abu\_dhabi\_race\_19\_results],

keys = ['Australia',

'Bahrain',

'China',

'Azerbaijan',

'Spain',

'Monaco',

'Canada',

'France',

'Austria',

'GreatBritain',

'Germany',

'Hungary',

'Belgium',

'Italy',

'Singapore',

'Russia',

'Japan',

'Mexico',

'UnitedStates',

'Brazil',

'AbuDhabi'],

names=['RACE'])

merge2019\_final = pd.concat([

australia\_race\_19\_final ,

bahrain\_race\_19\_final ,

china\_race\_19\_final ,

azerbaijan\_race\_19\_final ,

spain\_race\_19\_final ,

monaco\_race\_19\_final ,

canada\_race\_19\_final ,

france\_race\_19\_final ,

austria\_race\_19\_final ,

great\_britain\_race\_19\_final ,

germany\_race\_19\_final ,

hungary\_race\_19\_final ,

belgium\_race\_19\_final ,

italy\_race\_19\_final ,

singapore\_race\_19\_final,

russia\_race\_19\_final ,

japan\_race\_19\_final ,

mexico\_race\_19\_final ,

united\_states\_race\_19\_final ,

brazil\_race\_19\_final ,

abu\_dhabi\_race\_19\_final],

keys = ['Australia',

'Bahrain',

'China',

'Azerbaijan',

'Spain',

'Monaco',

'Canada',

'France',

'Austria',

'GreatBritain',

'Germany',

'Hungary',

'Belgium',

'Italy',

'Singapore',

'Russia',

'Japan',

'Mexico',

'UnitedStates',

'Brazil',

'AbuDhabi'],

names=['RACE'])

merge2019\_laps["Year"] = 2019

merge2019\_messages["Year"] = 2019

merge2019\_weather["Year"] = 2019

merge2019\_results["Year"] = 2019

merge2019\_final["Year"] = 2019

#%% loading 2020 data

austria\_race\_20a = ff1.get\_session(2020, 1, 'R')

austria\_race\_20a.load()

austria\_race\_20a\_laps = austria\_race\_20a.laps

austria\_race\_20a\_messages = austria\_race\_20a.race\_control\_messages

austria\_race\_20a\_weather = austria\_race\_20a.weather\_data

austria\_race\_20a\_results = austria\_race\_20a.results

austria\_race\_20a\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

austria\_race\_20a\_final = pd.merge(austria\_race\_20a\_laps, austria\_race\_20a\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

austria\_race\_20a\_final['TotalTime'] = austria\_race\_20a\_final['Time'] - austria\_race\_20a.session\_start\_time

austria\_race\_20a\_final = austria\_race\_20a\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

austria\_race\_20b = ff1.get\_session(2020, 2, 'R')

austria\_race\_20b.load()

austria\_race\_20b\_laps = austria\_race\_20b.laps

austria\_race\_20b\_messages = austria\_race\_20b.race\_control\_messages

austria\_race\_20b\_weather = austria\_race\_20b.weather\_data

austria\_race\_20b\_results = austria\_race\_20b.results

austria\_race\_20b\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

austria\_race\_20b\_final = pd.merge(austria\_race\_20b\_laps, austria\_race\_20b\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

austria\_race\_20b\_final['TotalTime'] = austria\_race\_20b\_final['Time'] - austria\_race\_20b.session\_start\_time

austria\_race\_20b\_final = austria\_race\_20b\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

hungary\_race\_20 = ff1.get\_session(2020, 3, 'R')

hungary\_race\_20.load()

hungary\_race\_20\_laps = hungary\_race\_20.laps

hungary\_race\_20\_messages = hungary\_race\_20.race\_control\_messages

hungary\_race\_20\_weather = hungary\_race\_20.weather\_data

hungary\_race\_20\_results = hungary\_race\_20.results

hungary\_race\_20\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

hungary\_race\_20\_final = pd.merge(hungary\_race\_20\_laps, hungary\_race\_20\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

hungary\_race\_20\_final['TotalTime'] = hungary\_race\_20\_final['Time'] - hungary\_race\_20.session\_start\_time

hungary\_race\_20\_final = hungary\_race\_20\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

great\_britain\_race\_20a = ff1.get\_session(2020, 4, 'R')

great\_britain\_race\_20a.load()

great\_britain\_race\_20a\_laps = great\_britain\_race\_20a.laps

great\_britain\_race\_20a\_messages = great\_britain\_race\_20a.race\_control\_messages

great\_britain\_race\_20a\_weather = great\_britain\_race\_20a.weather\_data

great\_britain\_race\_20a\_results = great\_britain\_race\_20a.results

great\_britain\_race\_20a\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

great\_britain\_race\_20a\_final = pd.merge(great\_britain\_race\_20a\_laps, great\_britain\_race\_20a\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

great\_britain\_race\_20a\_final['TotalTime'] = great\_britain\_race\_20a\_final['Time'] - great\_britain\_race\_20a.session\_start\_time

great\_britain\_race\_20a\_final = great\_britain\_race\_20a\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

great\_britain\_race\_20b = ff1.get\_session(2020, 5, 'R')

great\_britain\_race\_20b.load()

great\_britain\_race\_20b\_laps = great\_britain\_race\_20b.laps

great\_britain\_race\_20b\_messages = great\_britain\_race\_20b.race\_control\_messages

great\_britain\_race\_20b\_weather = great\_britain\_race\_20b.weather\_data

great\_britain\_race\_20b\_results = great\_britain\_race\_20b.results

great\_britain\_race\_20b\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

great\_britain\_race\_20b\_final = pd.merge(great\_britain\_race\_20b\_laps, great\_britain\_race\_20b\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

great\_britain\_race\_20b\_final['TotalTime'] = great\_britain\_race\_20b\_final['Time'] - great\_britain\_race\_20b.session\_start\_time

great\_britain\_race\_20b\_final = great\_britain\_race\_20b\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

spain\_race\_20 = ff1.get\_session(2020, 6, 'R')

spain\_race\_20.load()

spain\_race\_20\_laps = spain\_race\_20.laps

spain\_race\_20\_messages = spain\_race\_20.race\_control\_messages

spain\_race\_20\_weather = spain\_race\_20.weather\_data

spain\_race\_20\_results = spain\_race\_20.results

spain\_race\_20\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

spain\_race\_20\_final = pd.merge(spain\_race\_20\_laps, spain\_race\_20\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

spain\_race\_20\_final['TotalTime'] = spain\_race\_20\_final['Time'] - spain\_race\_20.session\_start\_time

spain\_race\_20\_final = spain\_race\_20\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

belgium\_race\_20 = ff1.get\_session(2020, 7, 'R')

belgium\_race\_20.load()

belgium\_race\_20\_laps = belgium\_race\_20.laps

belgium\_race\_20\_messages = belgium\_race\_20.race\_control\_messages

belgium\_race\_20\_weather = belgium\_race\_20.weather\_data

belgium\_race\_20\_results = belgium\_race\_20.results

belgium\_race\_20\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

belgium\_race\_20\_final = pd.merge(belgium\_race\_20\_laps, belgium\_race\_20\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

belgium\_race\_20\_final['TotalTime'] = belgium\_race\_20\_final['Time'] - belgium\_race\_20.session\_start\_time

belgium\_race\_20\_final = belgium\_race\_20\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

italy\_race\_20a = ff1.get\_session(2020, 8, 'R')

italy\_race\_20a.load()

italy\_race\_20a\_laps = italy\_race\_20a.laps

italy\_race\_20a\_messages = italy\_race\_20a.race\_control\_messages

italy\_race\_20a\_weather = italy\_race\_20a.weather\_data

italy\_race\_20a\_results = italy\_race\_20a.results

italy\_race\_20a\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

italy\_race\_20a\_final = pd.merge(italy\_race\_20a\_laps, italy\_race\_20a\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

italy\_race\_20a\_final['TotalTime'] = italy\_race\_20a\_final['Time'] - italy\_race\_20a.session\_start\_time

italy\_race\_20a\_final = italy\_race\_20a\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

italy\_race\_20b = ff1.get\_session(2020, 9, 'R')

italy\_race\_20b.load()

italy\_race\_20b\_laps = italy\_race\_20b.laps

italy\_race\_20b\_messages = italy\_race\_20b.race\_control\_messages

italy\_race\_20b\_weather = italy\_race\_20b.weather\_data

italy\_race\_20b\_results = italy\_race\_20b.results

italy\_race\_20b\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

italy\_race\_20b\_final = pd.merge(italy\_race\_20b\_laps, italy\_race\_20b\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

italy\_race\_20b\_final['TotalTime'] = italy\_race\_20b\_final['Time'] - italy\_race\_20b.session\_start\_time

italy\_race\_20b\_final = italy\_race\_20b\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

russia\_race\_20 = ff1.get\_session(2020, 10, 'R')

russia\_race\_20.load()

russia\_race\_20\_laps = russia\_race\_20.laps

russia\_race\_20\_messages = russia\_race\_20.race\_control\_messages

russia\_race\_20\_weather = russia\_race\_20.weather\_data

russia\_race\_20\_results = russia\_race\_20.results

russia\_race\_20\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

russia\_race\_20\_final = pd.merge(russia\_race\_20\_laps, russia\_race\_20\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

russia\_race\_20\_final['TotalTime'] = russia\_race\_20\_final['Time'] - russia\_race\_20.session\_start\_time

russia\_race\_20\_final = russia\_race\_20\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

germany\_race\_20 = ff1.get\_session(2020, 11, 'R')

germany\_race\_20.load()

germany\_race\_20\_laps = germany\_race\_20.laps

germany\_race\_20\_messages = germany\_race\_20.race\_control\_messages

germany\_race\_20\_weather = germany\_race\_20.weather\_data

germany\_race\_20\_results = germany\_race\_20.results

germany\_race\_20\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

germany\_race\_20\_final = pd.merge(germany\_race\_20\_laps, germany\_race\_20\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

germany\_race\_20\_final['TotalTime'] = germany\_race\_20\_final['Time'] - germany\_race\_20.session\_start\_time

germany\_race\_20\_final = germany\_race\_20\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

portugal\_race\_20 = ff1.get\_session(2020, 12, 'R')

portugal\_race\_20.load()

portugal\_race\_20\_laps = portugal\_race\_20.laps

portugal\_race\_20\_messages = portugal\_race\_20.race\_control\_messages

portugal\_race\_20\_weather = portugal\_race\_20.weather\_data

portugal\_race\_20\_results = portugal\_race\_20.results

portugal\_race\_20\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

portugal\_race\_20\_final = pd.merge(portugal\_race\_20\_laps, portugal\_race\_20\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

portugal\_race\_20\_final['TotalTime'] = portugal\_race\_20\_final['Time'] - portugal\_race\_20.session\_start\_time

portugal\_race\_20\_final = portugal\_race\_20\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

italy\_race\_20c = ff1.get\_session(2020, 13, 'R')

italy\_race\_20c.load()

italy\_race\_20c\_laps = italy\_race\_20b.laps

italy\_race\_20c\_messages = italy\_race\_20b.race\_control\_messages

italy\_race\_20c\_weather = italy\_race\_20b.weather\_data

italy\_race\_20c\_results = italy\_race\_20b.results

italy\_race\_20c\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

italy\_race\_20c\_final = pd.merge(italy\_race\_20c\_laps, italy\_race\_20c\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

italy\_race\_20c\_final['TotalTime'] = italy\_race\_20c\_final['Time'] - italy\_race\_20c.session\_start\_time

italy\_race\_20c\_final = italy\_race\_20c\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

turkey\_race\_20 = ff1.get\_session(2020, 14, 'R')

turkey\_race\_20.load()

turkey\_race\_20\_laps = turkey\_race\_20.laps

turkey\_race\_20\_messages = turkey\_race\_20.race\_control\_messages

turkey\_race\_20\_weather = turkey\_race\_20.weather\_data

turkey\_race\_20\_results = turkey\_race\_20.results

turkey\_race\_20\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

turkey\_race\_20\_final = pd.merge(turkey\_race\_20\_laps, turkey\_race\_20\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

turkey\_race\_20\_final['TotalTime'] = turkey\_race\_20\_final['Time'] - turkey\_race\_20.session\_start\_time

turkey\_race\_20\_final = turkey\_race\_20\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

bahrain\_race\_20a = ff1.get\_session(2020, 15, 'R')

bahrain\_race\_20a.load()

bahrain\_race\_20a\_laps = bahrain\_race\_20a.laps

bahrain\_race\_20a\_messages = bahrain\_race\_20a.race\_control\_messages

bahrain\_race\_20a\_weather = bahrain\_race\_20a.weather\_data

bahrain\_race\_20a\_results = bahrain\_race\_20a.results

bahrain\_race\_20a\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

bahrain\_race\_20a\_final = pd.merge(bahrain\_race\_20a\_laps, bahrain\_race\_20a\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

bahrain\_race\_20a\_final['TotalTime'] = bahrain\_race\_20a\_final['Time'] - bahrain\_race\_20a.session\_start\_time

bahrain\_race\_20a\_final = bahrain\_race\_20a\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

bahrain\_race\_20b = ff1.get\_session(2020, 16, 'R')

bahrain\_race\_20b.load()

bahrain\_race\_20b\_laps = bahrain\_race\_20b.laps

bahrain\_race\_20b\_messages = bahrain\_race\_20b.race\_control\_messages

bahrain\_race\_20b\_weather = bahrain\_race\_20b.weather\_data

bahrain\_race\_20b\_results = bahrain\_race\_20b.results

bahrain\_race\_20b\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

bahrain\_race\_20b\_final = pd.merge(bahrain\_race\_20b\_laps, bahrain\_race\_20b\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

bahrain\_race\_20b\_final['TotalTime'] = bahrain\_race\_20b\_final['Time'] - bahrain\_race\_20b.session\_start\_time

bahrain\_race\_20b\_final = bahrain\_race\_20b\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

abu\_dhabi\_race\_20 = ff1.get\_session(2020, 17, 'R')

abu\_dhabi\_race\_20.load()

abu\_dhabi\_race\_20\_laps = abu\_dhabi\_race\_20.laps

abu\_dhabi\_race\_20\_messages = abu\_dhabi\_race\_20.race\_control\_messages

abu\_dhabi\_race\_20\_weather = abu\_dhabi\_race\_20.weather\_data

abu\_dhabi\_race\_20\_results = abu\_dhabi\_race\_20.results

abu\_dhabi\_race\_20\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

abu\_dhabi\_race\_20\_final = pd.merge(abu\_dhabi\_race\_20\_laps, abu\_dhabi\_race\_20\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

abu\_dhabi\_race\_20\_final['TotalTime'] = abu\_dhabi\_race\_20\_final['Time'] - abu\_dhabi\_race\_20.session\_start\_time

abu\_dhabi\_race\_20\_final = abu\_dhabi\_race\_20\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

merge2020\_laps = pd.concat([

austria\_race\_20a\_laps,

austria\_race\_20b\_laps,

hungary\_race\_20\_laps,

great\_britain\_race\_20a\_laps,

great\_britain\_race\_20b\_laps,

spain\_race\_20\_laps,

belgium\_race\_20\_laps,

italy\_race\_20a\_laps,

italy\_race\_20b\_laps,

russia\_race\_20\_laps,

germany\_race\_20\_laps,

portugal\_race\_20\_laps,

italy\_race\_20c\_laps,

turkey\_race\_20\_laps,

bahrain\_race\_20a\_laps,

bahrain\_race\_20b\_laps,

abu\_dhabi\_race\_20\_laps],

keys = ['AustriaA',

'AustriaB',

'Hungary',

'GreatBritainA',

'GreatBritainB',

'Spain',

'Belgium',

'ItalyA',

'ItalyB',

'Russia',

'Germany',

'Portugal',

'ItalyC',

'Turkey',

'BahrainA',

'BahrainB',

'AbuDhabi'],

names=['RACE'])

merge2020\_messages = pd.concat([

austria\_race\_20a\_messages ,

austria\_race\_20b\_messages,

hungary\_race\_20\_messages ,

great\_britain\_race\_20a\_messages ,

great\_britain\_race\_20b\_messages ,

spain\_race\_20\_messages ,

belgium\_race\_20\_messages ,

italy\_race\_20a\_messages ,

italy\_race\_20b\_messages ,

russia\_race\_20\_messages ,

germany\_race\_20\_messages ,

portugal\_race\_20\_messages ,

italy\_race\_20c\_messages ,

turkey\_race\_20\_messages ,

bahrain\_race\_20a\_messages ,

bahrain\_race\_20b\_messages ,

abu\_dhabi\_race\_20\_messages ],

keys = ['AustriaA',

'AustriaB',

'Hungary',

'GreatBritainA',

'GreatBritainB',

'Spain',

'Belgium',

'ItalyA',

'ItalyB',

'Russia',

'Germany',

'Portugal',

'ItalyC',

'Turkey',

'BahrainA',

'BahrainB',

'AbuDhabi'],

names=['RACE'])

merge2020\_weather = pd.concat([

austria\_race\_20a\_weather,

austria\_race\_20b\_weather,

hungary\_race\_20\_weather,

great\_britain\_race\_20a\_weather,

great\_britain\_race\_20b\_weather,

spain\_race\_20\_weather,

belgium\_race\_20\_weather,

italy\_race\_20a\_weather,

italy\_race\_20b\_weather,

russia\_race\_20\_weather,

germany\_race\_20\_weather,

portugal\_race\_20\_weather,

italy\_race\_20c\_weather,

turkey\_race\_20\_weather,

bahrain\_race\_20a\_weather,

bahrain\_race\_20b\_weather,

abu\_dhabi\_race\_20\_weather],

keys = ['AustriaA',

'AustriaB',

'Hungary',

'GreatBritainA',

'GreatBritainB',

'Spain',

'Belgium',

'ItalyA',

'ItalyB',

'Russia',

'Germany',

'Portugal',

'ItalyC',

'Turkey',

'BahrainA',

'BahrainB',

'AbuDhabi'],

names=['RACE'])

merge2020\_results = pd.concat([

austria\_race\_20a\_results ,

austria\_race\_20b\_results ,

hungary\_race\_20\_results ,

great\_britain\_race\_20a\_results ,

great\_britain\_race\_20b\_results ,

spain\_race\_20\_results ,

belgium\_race\_20\_results ,

italy\_race\_20a\_results ,

italy\_race\_20b\_results ,

russia\_race\_20\_results ,

germany\_race\_20\_results ,

portugal\_race\_20\_results ,

italy\_race\_20c\_results ,

turkey\_race\_20\_results ,

bahrain\_race\_20a\_results ,

bahrain\_race\_20b\_results ,

abu\_dhabi\_race\_20\_results ],

keys = ['AustriaA',

'AustriaB',

'Hungary',

'GreatBritainA',

'GreatBritainB',

'Spain',

'Belgium',

'ItalyA',

'ItalyB',

'Russia',

'Germany',

'Portugal',

'ItalyC',

'Turkey',

'BahrainA',

'BahrainB',

'AbuDhabi'],

names=['RACE'])

merge2020\_final = pd.concat([

austria\_race\_20a\_final ,

austria\_race\_20b\_final ,

hungary\_race\_20\_final ,

great\_britain\_race\_20a\_final ,

great\_britain\_race\_20b\_final ,

spain\_race\_20\_final ,

belgium\_race\_20\_final ,

italy\_race\_20a\_final ,

italy\_race\_20b\_final ,

russia\_race\_20\_final ,

germany\_race\_20\_final ,

portugal\_race\_20\_final ,

italy\_race\_20c\_final ,

turkey\_race\_20\_final ,

bahrain\_race\_20a\_final ,

bahrain\_race\_20b\_final ,

abu\_dhabi\_race\_20\_final ],

keys = ['AustriaA',

'AustriaB',

'Hungary',

'GreatBritainA',

'GreatBritainB',

'Spain',

'Belgium',

'ItalyA',

'ItalyB',

'Russia',

'Germany',

'Portugal',

'ItalyC',

'Turkey',

'BahrainA',

'BahrainB',

'AbuDhabi'],

names=['RACE'])

merge2020\_laps["Year"] = 2020

merge2020\_messages["Year"] = 2020

merge2020\_weather["Year"] = 2020

merge2020\_results["Year"] = 2020

merge2020\_final["Year"] = 2020

#%% loading 2021 data

bahrain\_race\_21 = ff1.get\_session(2021, 1, 'R')

bahrain\_race\_21.load()

bahrain\_race\_21\_laps = bahrain\_race\_21.laps

bahrain\_race\_21\_messages= bahrain\_race\_21.race\_control\_messages

bahrain\_race\_21\_weather= bahrain\_race\_21.weather\_data

bahrain\_race\_21\_results = bahrain\_race\_21.results

bahrain\_race\_21\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

bahrain\_race\_21\_final = pd.merge(bahrain\_race\_21\_laps, bahrain\_race\_21\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

bahrain\_race\_21\_final['TotalTime'] = bahrain\_race\_21\_final['Time'] - bahrain\_race\_21.session\_start\_time

bahrain\_race\_21\_final = bahrain\_race\_21\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

italy\_race\_21a = ff1.get\_session(2021, 2, 'R')

italy\_race\_21a.load()

italy\_race\_21a\_laps = italy\_race\_21a.laps

italy\_race\_21a\_messages= italy\_race\_21a.race\_control\_messages

italy\_race\_21a\_weather= italy\_race\_21a.weather\_data

italy\_race\_21a\_results= italy\_race\_21a.results

italy\_race\_21a\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

italy\_race\_21a\_final = pd.merge(italy\_race\_21a\_laps, italy\_race\_21a\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

italy\_race\_21a\_final['TotalTime'] = italy\_race\_21a\_final['Time'] - italy\_race\_21a.session\_start\_time

italy\_race\_21a\_final = italy\_race\_21a\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

portugal\_race\_21 = ff1.get\_session(2021, 3, 'R')

portugal\_race\_21.load()

portugal\_race\_21\_laps = portugal\_race\_21.laps

portugal\_race\_21\_messages= portugal\_race\_21.race\_control\_messages

portugal\_race\_21\_weather= portugal\_race\_21.weather\_data

portugal\_race\_21\_results = portugal\_race\_21.results

portugal\_race\_21\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

portugal\_race\_21\_final = pd.merge(portugal\_race\_21\_laps, portugal\_race\_21\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

portugal\_race\_21\_final['TotalTime'] = portugal\_race\_21\_final['Time'] - portugal\_race\_21.session\_start\_time

portugal\_race\_21\_final = portugal\_race\_21\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

spain\_race\_21 = ff1.get\_session(2021, 4, 'R')

spain\_race\_21.load()

spain\_race\_21\_laps = spain\_race\_21.laps

spain\_race\_21\_messages= spain\_race\_21.race\_control\_messages

spain\_race\_21\_weather= spain\_race\_21.weather\_data

spain\_race\_21\_results= spain\_race\_21.results

spain\_race\_21\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

spain\_race\_21\_final = pd.merge(spain\_race\_21\_laps, spain\_race\_21\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

spain\_race\_21\_final['TotalTime'] = spain\_race\_21\_final['Time'] - spain\_race\_21.session\_start\_time

spain\_race\_21\_final = spain\_race\_21\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

monaco\_race\_21 = ff1.get\_session(2021, 5, 'R')

monaco\_race\_21.load()

monaco\_race\_21\_laps = monaco\_race\_21.laps

monaco\_race\_21\_messages= monaco\_race\_21.race\_control\_messages

monaco\_race\_21\_weather= monaco\_race\_21.weather\_data

monaco\_race\_21\_results= monaco\_race\_21.results

monaco\_race\_21\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

monaco\_race\_21\_final = pd.merge(monaco\_race\_21\_laps, monaco\_race\_21\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

monaco\_race\_21\_final['TotalTime'] = monaco\_race\_21\_final['Time'] - monaco\_race\_21.session\_start\_time

monaco\_race\_21\_final = monaco\_race\_21\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

azerbaijan\_race\_21 = ff1.get\_session(2021, 6, 'R')

azerbaijan\_race\_21.load()

azerbaijan\_race\_21\_laps = azerbaijan\_race\_21.laps

azerbaijan\_race\_21\_messages= azerbaijan\_race\_21.race\_control\_messages

azerbaijan\_race\_21\_weather= azerbaijan\_race\_21.weather\_data

azerbaijan\_race\_21\_results= azerbaijan\_race\_21.results

azerbaijan\_race\_21\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

azerbaijan\_race\_21\_final = pd.merge(azerbaijan\_race\_21\_laps, azerbaijan\_race\_21\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

azerbaijan\_race\_21\_final['TotalTime'] = azerbaijan\_race\_21\_final['Time'] - azerbaijan\_race\_21.session\_start\_time

azerbaijan\_race\_21\_final = azerbaijan\_race\_21\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

france\_race\_21 = ff1.get\_session(2021, 7, 'R')

france\_race\_21.load()

france\_race\_21\_laps = france\_race\_21.laps

france\_race\_21\_messages= france\_race\_21.race\_control\_messages

france\_race\_21\_weather= france\_race\_21.weather\_data

france\_race\_21\_results = france\_race\_21.results

france\_race\_21\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

france\_race\_21\_final = pd.merge(france\_race\_21\_laps, france\_race\_21\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

france\_race\_21\_final['TotalTime'] = france\_race\_21\_final['Time'] - france\_race\_21.session\_start\_time

france\_race\_21\_final = france\_race\_21\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

austria\_race\_21a = ff1.get\_session(2021, 8, 'R')

austria\_race\_21a.load()

austria\_race\_21a\_laps = austria\_race\_21a.laps

austria\_race\_21a\_messages= austria\_race\_21a.race\_control\_messages

austria\_race\_21a\_weather= austria\_race\_21a.weather\_data

austria\_race\_21a\_results = austria\_race\_21a.results

austria\_race\_21a\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

austria\_race\_21a\_final = pd.merge(austria\_race\_21a\_laps, austria\_race\_21a\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

austria\_race\_21a\_final['TotalTime'] = austria\_race\_21a\_final['Time'] - austria\_race\_21a.session\_start\_time

austria\_race\_21a\_final = austria\_race\_21a\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

austria\_race\_21b = ff1.get\_session(2021, 9, 'R')

austria\_race\_21b.load()

austria\_race\_21b\_laps = austria\_race\_21b.laps

austria\_race\_21b\_messages= austria\_race\_21b.race\_control\_messages

austria\_race\_21b\_weather= austria\_race\_21b.weather\_data

austria\_race\_21b\_results = austria\_race\_21b.results

austria\_race\_21b\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

austria\_race\_21b\_final = pd.merge(austria\_race\_21b\_laps, austria\_race\_21b\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

austria\_race\_21b\_final['TotalTime'] = austria\_race\_21b\_final['Time'] - austria\_race\_21b.session\_start\_time

austria\_race\_21b\_final = austria\_race\_21b\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

great\_britain\_race\_21 = ff1.get\_session(2021, 10, 'R')

great\_britain\_race\_21.load()

great\_britain\_race\_21\_laps = great\_britain\_race\_21.laps

great\_britain\_race\_21\_messages= great\_britain\_race\_21.race\_control\_messages

great\_britain\_race\_21\_weather= great\_britain\_race\_21.weather\_data

great\_britain\_race\_21\_results = great\_britain\_race\_21.results

great\_britain\_race\_21\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

great\_britain\_race\_21\_final = pd.merge(great\_britain\_race\_21\_laps, great\_britain\_race\_21\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

great\_britain\_race\_21\_final['TotalTime'] = great\_britain\_race\_21\_final['Time'] - great\_britain\_race\_21.session\_start\_time

great\_britain\_race\_21\_final = great\_britain\_race\_21\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

hungary\_race\_21 = ff1.get\_session(2021, 11, 'R')

hungary\_race\_21.load()

hungary\_race\_21\_laps = hungary\_race\_21.laps

hungary\_race\_21\_messages= hungary\_race\_21.race\_control\_messages

hungary\_race\_21\_weather= hungary\_race\_21.weather\_data

hungary\_race\_21\_results = hungary\_race\_21.results

hungary\_race\_21\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

hungary\_race\_21\_final = pd.merge(hungary\_race\_21\_laps, hungary\_race\_21\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

hungary\_race\_21\_final['TotalTime'] = hungary\_race\_21\_final['Time'] - hungary\_race\_21.session\_start\_time

hungary\_race\_21\_final = hungary\_race\_21\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

belgium\_race\_21 = ff1.get\_session(2021, 12, 'R')

belgium\_race\_21.load()

belgium\_race\_21\_laps = belgium\_race\_21.laps

belgium\_race\_21\_messages= belgium\_race\_21.race\_control\_messages

belgium\_race\_21\_weather= belgium\_race\_21.weather\_data

belgium\_race\_21\_results = belgium\_race\_21.results

belgium\_race\_21\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

belgium\_race\_21\_final = pd.merge(belgium\_race\_21\_laps, belgium\_race\_21\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

belgium\_race\_21\_final['TotalTime'] = belgium\_race\_21\_final['Time'] - belgium\_race\_21.session\_start\_time

belgium\_race\_21\_final = belgium\_race\_21\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

netherlands\_race\_21 = ff1.get\_session(2021, 13, 'R')

netherlands\_race\_21 .load()

netherlands\_race\_21\_laps = netherlands\_race\_21.laps

netherlands\_race\_21\_messages= netherlands\_race\_21.race\_control\_messages

netherlands\_race\_21\_weather= netherlands\_race\_21.weather\_data

netherlands\_race\_21\_results = netherlands\_race\_21.results

netherlands\_race\_21\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

netherlands\_race\_21\_final = pd.merge(netherlands\_race\_21\_laps, netherlands\_race\_21\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

netherlands\_race\_21\_final['TotalTime'] = netherlands\_race\_21\_final['Time'] - netherlands\_race\_21.session\_start\_time

netherlands\_race\_21\_final = netherlands\_race\_21\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

italy\_race\_21b = ff1.get\_session(2021, 14, 'R')

italy\_race\_21b.load()

italy\_race\_21b\_laps = italy\_race\_21b.laps

italy\_race\_21b\_messages= italy\_race\_21b.race\_control\_messages

italy\_race\_21b\_weather= italy\_race\_21b.weather\_data

italy\_race\_21b\_results = italy\_race\_21b.results

italy\_race\_21b\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

italy\_race\_21b\_final = pd.merge(italy\_race\_21b\_laps, italy\_race\_21b\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

italy\_race\_21b\_final['TotalTime'] = italy\_race\_21b\_final['Time'] - italy\_race\_21b.session\_start\_time

italy\_race\_21b\_final = italy\_race\_21b\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

russia\_race\_21 = ff1.get\_session(2021, 15, 'R')

russia\_race\_21.load()

russia\_race\_21\_laps = russia\_race\_21.laps

russia\_race\_21\_messages= russia\_race\_21.race\_control\_messages

russia\_race\_21\_weather= russia\_race\_21.weather\_data

russia\_race\_21\_results = russia\_race\_21.results

russia\_race\_21\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

russia\_race\_21\_final = pd.merge(russia\_race\_21\_laps, russia\_race\_21\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

russia\_race\_21\_final['TotalTime'] = russia\_race\_21\_final['Time'] - russia\_race\_21.session\_start\_time

russia\_race\_21\_final = russia\_race\_21\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

turkey\_race\_21 = ff1.get\_session(2021, 16, 'R')

turkey\_race\_21.load()

turkey\_race\_21\_laps = turkey\_race\_21.laps

turkey\_race\_21\_messages= turkey\_race\_21.race\_control\_messages

turkey\_race\_21\_weather= turkey\_race\_21.weather\_data

turkey\_race\_21\_results = turkey\_race\_21.results

turkey\_race\_21\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

turkey\_race\_21\_final = pd.merge(turkey\_race\_21\_laps, turkey\_race\_21\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

turkey\_race\_21\_final['TotalTime'] = turkey\_race\_21\_final['Time'] - turkey\_race\_21.session\_start\_time

turkey\_race\_21\_final = turkey\_race\_21\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

united\_states\_race\_21 = ff1.get\_session(2021, 17, 'R')

united\_states\_race\_21.load()

united\_states\_race\_21\_laps = united\_states\_race\_21.laps

united\_states\_race\_21\_messages= united\_states\_race\_21.race\_control\_messages

united\_states\_race\_21\_weather= united\_states\_race\_21.weather\_data

united\_states\_race\_21\_results = united\_states\_race\_21.results

united\_states\_race\_21\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

united\_states\_race\_21\_final = pd.merge(united\_states\_race\_21\_laps, united\_states\_race\_21\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

united\_states\_race\_21\_final['TotalTime'] = united\_states\_race\_21\_final['Time'] - united\_states\_race\_21.session\_start\_time

united\_states\_race\_21\_final = united\_states\_race\_21\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

mexico\_race\_21 = ff1.get\_session(2021, 18, 'R')

mexico\_race\_21.load()

mexico\_race\_21\_laps = mexico\_race\_21.laps

mexico\_race\_21\_messages= mexico\_race\_21.race\_control\_messages

mexico\_race\_21\_weather= mexico\_race\_21.weather\_data

mexico\_race\_21\_results = mexico\_race\_21.results

mexico\_race\_21\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

mexico\_race\_21\_final = pd.merge(mexico\_race\_21\_laps, mexico\_race\_21\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

mexico\_race\_21\_final['TotalTime'] = mexico\_race\_21\_final['Time'] - mexico\_race\_21.session\_start\_time

mexico\_race\_21\_final = mexico\_race\_21\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

brazil\_race\_21 = ff1.get\_session(2021, 19, 'R')

brazil\_race\_21.load()

brazil\_race\_21\_laps = brazil\_race\_21.laps

brazil\_race\_21\_messages= brazil\_race\_21.race\_control\_messages

brazil\_race\_21\_weather= brazil\_race\_21.weather\_data

brazil\_race\_21\_results = brazil\_race\_21.results

brazil\_race\_21\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

brazil\_race\_21\_final = pd.merge(brazil\_race\_21\_laps, brazil\_race\_21\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

brazil\_race\_21\_final['TotalTime'] = brazil\_race\_21\_final['Time'] - brazil\_race\_21.session\_start\_time

brazil\_race\_21\_final = brazil\_race\_21\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

qatar\_race\_21 = ff1.get\_session(2021, 20, 'R')

qatar\_race\_21.load()

qatar\_race\_21\_laps = qatar\_race\_21.laps

qatar\_race\_21\_messages= qatar\_race\_21.race\_control\_messages

qatar\_race\_21\_weather= qatar\_race\_21.weather\_data

qatar\_race\_21\_results = qatar\_race\_21.results

qatar\_race\_21\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

qatar\_race\_21\_final = pd.merge(qatar\_race\_21\_laps, qatar\_race\_21\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

qatar\_race\_21\_final['TotalTime'] = qatar\_race\_21\_final['Time'] - qatar\_race\_21.session\_start\_time

qatar\_race\_21\_final = qatar\_race\_21\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

saudi\_arabia\_race\_21 = ff1.get\_session(2021, 21, 'R')

saudi\_arabia\_race\_21.load()

saudi\_arabia\_race\_21\_laps = saudi\_arabia\_race\_21.laps

saudi\_arabia\_race\_21\_messages= saudi\_arabia\_race\_21.race\_control\_messages

saudi\_arabia\_race\_21\_weather= saudi\_arabia\_race\_21.weather\_data

saudi\_arabia\_race\_21\_results= saudi\_arabia\_race\_21.results

saudi\_arabia\_race\_21\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

saudi\_arabia\_race\_21\_final = pd.merge(saudi\_arabia\_race\_21\_laps, saudi\_arabia\_race\_21\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

saudi\_arabia\_race\_21\_final['TotalTime'] = saudi\_arabia\_race\_21\_final['Time'] - saudi\_arabia\_race\_21.session\_start\_time

saudi\_arabia\_race\_21\_final = saudi\_arabia\_race\_21\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

abu\_dhabi\_race\_21 = ff1.get\_session(2021, 22, 'R')

abu\_dhabi\_race\_21.load()

abu\_dhabi\_race\_21\_laps = abu\_dhabi\_race\_21.laps

abu\_dhabi\_race\_21\_messages= abu\_dhabi\_race\_21.race\_control\_messages

abu\_dhabi\_race\_21\_weather= abu\_dhabi\_race\_21.weather\_data

abu\_dhabi\_race\_21\_results= abu\_dhabi\_race\_21.results

abu\_dhabi\_race\_21\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

abu\_dhabi\_race\_21\_final = pd.merge(abu\_dhabi\_race\_21\_laps, abu\_dhabi\_race\_21\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

abu\_dhabi\_race\_21\_final['TotalTime'] = abu\_dhabi\_race\_21\_final['Time'] - abu\_dhabi\_race\_21.session\_start\_time

abu\_dhabi\_race\_21\_final = abu\_dhabi\_race\_21\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

merge2021\_laps = pd.concat([

bahrain\_race\_21\_laps,

italy\_race\_21a\_laps,

portugal\_race\_21\_laps,

spain\_race\_21\_laps,

monaco\_race\_21\_laps,

azerbaijan\_race\_21\_laps,

france\_race\_21\_laps,

austria\_race\_21a\_laps,

austria\_race\_21b\_laps,

great\_britain\_race\_21\_laps,

hungary\_race\_21\_laps,

belgium\_race\_21\_laps,

netherlands\_race\_21\_laps,

italy\_race\_21b\_laps,

russia\_race\_21\_laps,

turkey\_race\_21\_laps,

united\_states\_race\_21\_laps,

mexico\_race\_21\_laps,

brazil\_race\_21\_laps,

qatar\_race\_21\_laps,

saudi\_arabia\_race\_21\_laps,

abu\_dhabi\_race\_21\_laps],

keys = ['Bahrain',

'ItalyA',

'Portugal',

'Spain',

'Monaco',

'Azerbaijan',

'France',

'AustriaA',

'AustriaB',

'GreatBritain',

'Hungary',

'Belgium',

'Netherlands',

'ItalyB',

'Russia',

'Turkey',

'UnitedStates',

'Mexico',

'Brazil',

'Qatar',

'SaudiArabia',

'AbuDhabi'],

names=['RACE'])

merge2021\_messages = pd.concat([

bahrain\_race\_21\_messages,

italy\_race\_21a\_messages,

portugal\_race\_21\_messages,

spain\_race\_21\_messages,

monaco\_race\_21\_messages,

azerbaijan\_race\_21\_messages,

france\_race\_21\_messages,

austria\_race\_21a\_messages,

austria\_race\_21b\_messages,

great\_britain\_race\_21\_messages,

hungary\_race\_21\_messages,

belgium\_race\_21\_messages,

netherlands\_race\_21\_messages,

italy\_race\_21b\_messages,

russia\_race\_21\_messages,

turkey\_race\_21\_messages,

united\_states\_race\_21\_messages,

mexico\_race\_21\_messages,

brazil\_race\_21\_messages,

qatar\_race\_21\_messages,

saudi\_arabia\_race\_21\_messages,

abu\_dhabi\_race\_21\_messages],

keys = ['Bahrain',

'ItalyA',

'Portugal',

'Spain',

'Monaco',

'Azerbaijan',

'France',

'AustriaA',

'AustriaB',

'GreatBritain',

'Hungary',

'Belgium',

'Netherlands',

'ItalyB',

'Russia',

'Turkey',

'UnitedStates',

'Mexico',

'Brazil',

'Qatar',

'SaudiArabia',

'AbuDhabi'],

names=['RACE'])

merge2021\_weather = pd.concat([

bahrain\_race\_21\_weather,

italy\_race\_21a\_weather,

portugal\_race\_21\_weather,

spain\_race\_21\_weather,

monaco\_race\_21\_weather,

azerbaijan\_race\_21\_weather,

france\_race\_21\_weather,

austria\_race\_21a\_weather,

austria\_race\_21b\_weather,

great\_britain\_race\_21\_weather,

hungary\_race\_21\_weather,

belgium\_race\_21\_weather,

netherlands\_race\_21\_weather,

italy\_race\_21b\_weather,

russia\_race\_21\_weather,

turkey\_race\_21\_weather,

united\_states\_race\_21\_weather,

mexico\_race\_21\_weather,

brazil\_race\_21\_weather,

qatar\_race\_21\_weather,

saudi\_arabia\_race\_21\_weather,

abu\_dhabi\_race\_21\_weather],

keys = ['Bahrain',

'ItalyA',

'Portugal',

'Spain',

'Monaco',

'Azerbaijan',

'France',

'AustriaA',

'AustriaB',

'GreatBritain',

'Hungary',

'Belgium',

'Netherlands',

'ItalyB',

'Russia',

'Turkey',

'UnitedStates',

'Mexico',

'Brazil',

'Qatar',

'SaudiArabia',

'AbuDhabi'],

names=['RACE'])

merge2021\_results = pd.concat([

bahrain\_race\_21\_results,

italy\_race\_21a\_results,

portugal\_race\_21\_results,

spain\_race\_21\_results,

monaco\_race\_21\_results,

azerbaijan\_race\_21\_results,

france\_race\_21\_results,

austria\_race\_21a\_results,

austria\_race\_21b\_results,

great\_britain\_race\_21\_results,

hungary\_race\_21\_results,

belgium\_race\_21\_results,

netherlands\_race\_21\_results,

italy\_race\_21b\_results,

russia\_race\_21\_results,

turkey\_race\_21\_results,

united\_states\_race\_21\_results,

mexico\_race\_21\_results,

brazil\_race\_21\_results,

qatar\_race\_21\_results,

saudi\_arabia\_race\_21\_results,

abu\_dhabi\_race\_21\_results],

keys = ['Bahrain',

'ItalyA',

'Portugal',

'Spain',

'Monaco',

'Azerbaijan',

'France',

'AustriaA',

'AustriaB',

'GreatBritain',

'Hungary',

'Belgium',

'Netherlands',

'ItalyB',

'Russia',

'Turkey',

'UnitedStates',

'Mexico',

'Brazil',

'Qatar',

'SaudiArabia',

'AbuDhabi'],

names=['RACE'])

merge2021\_final = pd.concat([

bahrain\_race\_21\_final,

italy\_race\_21a\_final,

portugal\_race\_21\_final,

spain\_race\_21\_final,

monaco\_race\_21\_final,

azerbaijan\_race\_21\_final,

france\_race\_21\_final,

austria\_race\_21a\_final,

austria\_race\_21b\_final,

great\_britain\_race\_21\_final,

hungary\_race\_21\_final,

belgium\_race\_21\_final,

netherlands\_race\_21\_final,

italy\_race\_21b\_final,

russia\_race\_21\_final,

turkey\_race\_21\_final,

united\_states\_race\_21\_final,

mexico\_race\_21\_final,

brazil\_race\_21\_final,

qatar\_race\_21\_final,

saudi\_arabia\_race\_21\_final,

abu\_dhabi\_race\_21\_final],

keys = ['Bahrain',

'ItalyA',

'Portugal',

'Spain',

'Monaco',

'Azerbaijan',

'France',

'AustriaA',

'AustriaB',

'GreatBritain',

'Hungary',

'Belgium',

'Netherlands',

'ItalyB',

'Russia',

'Turkey',

'UnitedStates',

'Mexico',

'Brazil',

'Qatar',

'SaudiArabia',

'AbuDhabi'],

names=['RACE'])

merge2021\_laps["Year"] = 2021

merge2021\_messages["Year"] = 2021

merge2021\_weather["Year"] = 2021

merge2021\_results["Year"] = 2021

merge2021\_final["Year"] = 2021

#%% loading 2022 data

bahrain\_race\_22 = ff1.get\_session(2022, 1, 'R')

bahrain\_race\_22.load()

bahrain\_race\_22\_laps = bahrain\_race\_22.laps

bahrain\_race\_22\_messages = bahrain\_race\_22.race\_control\_messages

bahrain\_race\_22\_weather= bahrain\_race\_22.weather\_data

bahrain\_race\_22\_results= bahrain\_race\_22.results

bahrain\_race\_22\_results.columns

bahrain\_race\_22\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

bahrain\_race\_22\_final = pd.merge(bahrain\_race\_22\_laps, bahrain\_race\_22\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

bahrain\_race\_22\_final['TotalTime'] = bahrain\_race\_22\_final['Time'] - bahrain\_race\_22.session\_start\_time

bahrain\_race\_22\_final = bahrain\_race\_22\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

saudi\_arabia\_race\_22 = ff1.get\_session(2022, 2, 'R')

saudi\_arabia\_race\_22.load()

saudi\_arabia\_race\_22\_laps = saudi\_arabia\_race\_22.laps

saudi\_arabia\_race\_22\_messages = saudi\_arabia\_race\_22.race\_control\_messages

saudi\_arabia\_race\_22\_weather= saudi\_arabia\_race\_22.weather\_data

saudi\_arabia\_race\_22\_results= saudi\_arabia\_race\_22.results

saudi\_arabia\_race\_22\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

saudi\_arabia\_race\_22\_final = pd.merge(saudi\_arabia\_race\_22\_laps, saudi\_arabia\_race\_22\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

saudi\_arabia\_race\_22\_final['TotalTime'] = saudi\_arabia\_race\_22\_final['Time'] - saudi\_arabia\_race\_22.session\_start\_time

saudi\_arabia\_race\_22\_final = saudi\_arabia\_race\_22\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

australia\_race\_22 = ff1.get\_session(2022, 3, 'R')

australia\_race\_22.load()

australia\_race\_22\_laps = australia\_race\_22.laps

australia\_race\_22\_messages = australia\_race\_22.race\_control\_messages

australia\_race\_22\_weather= australia\_race\_22.weather\_data

australia\_race\_22\_results= australia\_race\_22.results

australia\_race\_22\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

australia\_race\_22\_final = pd.merge(australia\_race\_22\_laps, australia\_race\_22\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

australia\_race\_22\_final['TotalTime'] = australia\_race\_22\_final['Time'] - australia\_race\_22.session\_start\_time

australia\_race\_22\_final = australia\_race\_22\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

italy\_race\_22a = ff1.get\_session(2022, 4, 'R')

italy\_race\_22a.load()

italy\_race\_22a\_laps = italy\_race\_22a.laps

italy\_race\_22a\_messages = italy\_race\_22a.race\_control\_messages

italy\_race\_22a\_weather= italy\_race\_22a.weather\_data

italy\_race\_22a\_results= italy\_race\_22a.results

italy\_race\_22a\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

italy\_race\_22a\_final = pd.merge(italy\_race\_22a\_laps, italy\_race\_22a\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

italy\_race\_22a\_final['TotalTime'] = italy\_race\_22a\_final['Time'] - italy\_race\_22a.session\_start\_time

italy\_race\_22a\_final = italy\_race\_22a\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

united\_states\_race\_22a = ff1.get\_session(2022, 5, 'R')

united\_states\_race\_22a.load()

united\_states\_race\_22a\_laps = united\_states\_race\_22a.laps

united\_states\_race\_22a\_messages= united\_states\_race\_22a.race\_control\_messages

united\_states\_race\_22a\_weather= united\_states\_race\_22a.weather\_data

united\_states\_race\_22a\_results= united\_states\_race\_22a.results

united\_states\_race\_22a\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

united\_states\_race\_22a\_final = pd.merge(united\_states\_race\_22a\_laps, united\_states\_race\_22a\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

united\_states\_race\_22a\_final['TotalTime'] = united\_states\_race\_22a\_final['Time'] - united\_states\_race\_22a.session\_start\_time

united\_states\_race\_22a\_final = united\_states\_race\_22a\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

spain\_race\_22 = ff1.get\_session(2022, 6, 'R')

spain\_race\_22.load()

spain\_race\_22\_laps = spain\_race\_22.laps

spain\_race\_22\_messages= spain\_race\_22.race\_control\_messages

spain\_race\_22\_weather= spain\_race\_22.weather\_data

spain\_race\_22\_results= spain\_race\_22.results

spain\_race\_22\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

spain\_race\_22\_final = pd.merge(spain\_race\_22\_laps, spain\_race\_22\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

spain\_race\_22\_final['TotalTime'] = spain\_race\_22\_final['Time'] - spain\_race\_22.session\_start\_time

spain\_race\_22\_final = spain\_race\_22\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

monaco\_race\_22 = ff1.get\_session(2022, 7, 'R')

monaco\_race\_22.load()

monaco\_race\_22\_laps = monaco\_race\_22.laps

monaco\_race\_22\_messages= monaco\_race\_22.race\_control\_messages

monaco\_race\_22\_weather= monaco\_race\_22.weather\_data

monaco\_race\_22\_results= monaco\_race\_22.results

monaco\_race\_22\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

monaco\_race\_22\_final = pd.merge(monaco\_race\_22\_laps, monaco\_race\_22\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

monaco\_race\_22\_final['TotalTime'] = monaco\_race\_22\_final['Time'] - monaco\_race\_22.session\_start\_time

monaco\_race\_22\_final = monaco\_race\_22\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

azerbaijan\_race\_22 = ff1.get\_session(2022, 8, 'R')

azerbaijan\_race\_22.load()

azerbaijan\_race\_22\_laps = azerbaijan\_race\_22.laps

azerbaijan\_race\_22\_messages= azerbaijan\_race\_22.race\_control\_messages

azerbaijan\_race\_22\_weather= azerbaijan\_race\_22.weather\_data

azerbaijan\_race\_22\_results= azerbaijan\_race\_22.results

azerbaijan\_race\_22\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

azerbaijan\_race\_22\_final = pd.merge(azerbaijan\_race\_22\_laps, azerbaijan\_race\_22\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

azerbaijan\_race\_22\_final['TotalTime'] = azerbaijan\_race\_22\_final['Time'] - azerbaijan\_race\_22.session\_start\_time

azerbaijan\_race\_22\_final = azerbaijan\_race\_22\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

canada\_race\_22 = ff1.get\_session(2022, 9, 'R')

canada\_race\_22.load()

canada\_race\_22\_laps = canada\_race\_22.laps

canada\_race\_22\_messages= canada\_race\_22.race\_control\_messages

canada\_race\_22\_weather= canada\_race\_22.weather\_data

canada\_race\_22\_results= canada\_race\_22.results

canada\_race\_22\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

canada\_race\_22\_final = pd.merge(canada\_race\_22\_laps, canada\_race\_22\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

canada\_race\_22\_final['TotalTime'] = canada\_race\_22\_final['Time'] - canada\_race\_22.session\_start\_time

canada\_race\_22\_final = canada\_race\_22\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

great\_britain\_race\_22 = ff1.get\_session(2022, 10, 'R')

great\_britain\_race\_22.load()

great\_britain\_race\_22\_laps = azerbaijan\_race\_22.laps

great\_britain\_race\_22\_messages= azerbaijan\_race\_22.race\_control\_messages

great\_britain\_race\_22\_weather= azerbaijan\_race\_22.weather\_data

great\_britain\_race\_22\_results= azerbaijan\_race\_22.results

great\_britain\_race\_22\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

great\_britain\_race\_22\_final = pd.merge(great\_britain\_race\_22\_laps, great\_britain\_race\_22\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

great\_britain\_race\_22\_final['TotalTime'] = great\_britain\_race\_22\_final['Time'] - great\_britain\_race\_22.session\_start\_time

great\_britain\_race\_22\_final = great\_britain\_race\_22\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

austria\_race\_22 = ff1.get\_session(2022, 11, 'R')

austria\_race\_22.load()

austria\_race\_22\_laps = austria\_race\_22.laps

austria\_race\_22\_messages= austria\_race\_22.race\_control\_messages

austria\_race\_22\_weather= austria\_race\_22.weather\_data

austria\_race\_22\_results= austria\_race\_22.results

austria\_race\_22\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

austria\_race\_22\_final = pd.merge(austria\_race\_22\_laps, austria\_race\_22\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

austria\_race\_22\_final['TotalTime'] = austria\_race\_22\_final['Time'] - austria\_race\_22.session\_start\_time

austria\_race\_22\_final = austria\_race\_22\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

france\_race\_22 = ff1.get\_session(2022, 12, 'R')

france\_race\_22.load()

france\_race\_22\_laps = france\_race\_22.laps

france\_race\_22\_messages= france\_race\_22.race\_control\_messages

france\_race\_22\_weather= france\_race\_22.weather\_data

france\_race\_22\_results= france\_race\_22.results

france\_race\_22\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

france\_race\_22\_final = pd.merge(france\_race\_22\_laps, france\_race\_22\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

france\_race\_22\_final['TotalTime'] = france\_race\_22\_final['Time'] - france\_race\_22.session\_start\_time

france\_race\_22\_final = france\_race\_22\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

hungary\_race\_22 = ff1.get\_session(2022, 13, 'R')

hungary\_race\_22.load()

hungary\_race\_22\_laps = hungary\_race\_22.laps

hungary\_race\_22\_messages= hungary\_race\_22.race\_control\_messages

hungary\_race\_22\_weather= hungary\_race\_22.weather\_data

hungary\_race\_22\_results= hungary\_race\_22.results

hungary\_race\_22\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

hungary\_race\_22\_final = pd.merge(hungary\_race\_22\_laps, hungary\_race\_22\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

hungary\_race\_22\_final['TotalTime'] = hungary\_race\_22\_final['Time'] - hungary\_race\_22.session\_start\_time

hungary\_race\_22\_final = hungary\_race\_22\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

belgium\_race\_22 = ff1.get\_session(2022, 14, 'R')

belgium\_race\_22.load()

belgium\_race\_22\_laps = belgium\_race\_22.laps

belgium\_race\_22\_messages= belgium\_race\_22.race\_control\_messages

belgium\_race\_22\_weather= belgium\_race\_22.weather\_data

belgium\_race\_22\_results= belgium\_race\_22.results

belgium\_race\_22\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

belgium\_race\_22\_final = pd.merge(belgium\_race\_22\_laps, belgium\_race\_22\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

belgium\_race\_22\_final['TotalTime'] = belgium\_race\_22\_final['Time'] - belgium\_race\_22.session\_start\_time

belgium\_race\_22\_final = belgium\_race\_22\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

netherlands\_race\_22 = ff1.get\_session(2022, 15, 'R')

netherlands\_race\_22.load()

netherlands\_race\_22\_laps = netherlands\_race\_22.laps

netherlands\_race\_22\_messages= netherlands\_race\_22.race\_control\_messages

netherlands\_race\_22\_weather= netherlands\_race\_22.weather\_data

netherlands\_race\_22\_results= netherlands\_race\_22.results

netherlands\_race\_22\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

netherlands\_race\_22\_final = pd.merge(netherlands\_race\_22\_laps, netherlands\_race\_22\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

netherlands\_race\_22\_final['TotalTime'] = netherlands\_race\_22\_final['Time'] - netherlands\_race\_22.session\_start\_time

netherlands\_race\_22\_final = netherlands\_race\_22\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

italy\_race\_22b = ff1.get\_session(2022, 16, 'R')

italy\_race\_22b.load()

italy\_race\_22b\_laps = italy\_race\_22b.laps

italy\_race\_22b\_messages= italy\_race\_22b.race\_control\_messages

italy\_race\_22b\_weather= italy\_race\_22b.weather\_data

italy\_race\_22b\_results= italy\_race\_22b.results

italy\_race\_22b\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

italy\_race\_22b\_final = pd.merge(italy\_race\_22b\_laps, italy\_race\_22b\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

italy\_race\_22b\_final['TotalTime'] = italy\_race\_22b\_final['Time'] - italy\_race\_22b.session\_start\_time

italy\_race\_22b\_final = italy\_race\_22b\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

singapore\_race\_22 = ff1.get\_session(2022, 17, 'R')

singapore\_race\_22.load()

singapore\_race\_22\_laps = singapore\_race\_22.laps

singapore\_race\_22\_messages= singapore\_race\_22.race\_control\_messages

singapore\_race\_22\_weather= singapore\_race\_22.weather\_data

singapore\_race\_22\_results= singapore\_race\_22.results

singapore\_race\_22\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

singapore\_race\_22\_final = pd.merge(singapore\_race\_22\_laps, singapore\_race\_22\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

singapore\_race\_22\_final['TotalTime'] = singapore\_race\_22\_final['Time'] - singapore\_race\_22.session\_start\_time

singapore\_race\_22\_final = singapore\_race\_22\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

japan\_race\_22 = ff1.get\_session(2022, 18, 'R')

japan\_race\_22.load()

japan\_race\_22\_laps = japan\_race\_22.laps

japan\_race\_22\_messages= japan\_race\_22.race\_control\_messages

japan\_race\_22\_weather = japan\_race\_22.weather\_data

japan\_race\_22\_results= japan\_race\_22.results

japan\_race\_22\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

japan\_race\_22\_final = pd.merge(japan\_race\_22\_laps, japan\_race\_22\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

japan\_race\_22\_final['TotalTime'] = japan\_race\_22\_final['Time'] - japan\_race\_22.session\_start\_time

japan\_race\_22\_final = japan\_race\_22\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

united\_states\_race\_22b = ff1.get\_session(2022, 19, 'R')

united\_states\_race\_22b.load()

united\_states\_race\_22b\_laps = united\_states\_race\_22b.laps

united\_states\_race\_22b\_messages= united\_states\_race\_22b.race\_control\_messages

united\_states\_race\_22b\_weather= united\_states\_race\_22b.weather\_data

united\_states\_race\_22b\_results= united\_states\_race\_22b.results

united\_states\_race\_22b\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

united\_states\_race\_22b\_final = pd.merge(united\_states\_race\_22b\_laps, united\_states\_race\_22b\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

united\_states\_race\_22b\_final['TotalTime'] = united\_states\_race\_22b\_final['Time'] - united\_states\_race\_22b.session\_start\_time

united\_states\_race\_22b\_final = united\_states\_race\_22b\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

mexico\_race\_22 = ff1.get\_session(2022, 20, 'R')

mexico\_race\_22.load()

mexico\_race\_22\_laps = mexico\_race\_22.laps

mexico\_race\_22\_messages= mexico\_race\_22.race\_control\_messages

mexico\_race\_22\_weather = mexico\_race\_22.weather\_data

mexico\_race\_22\_results= mexico\_race\_22.results

mexico\_race\_22\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

mexico\_race\_22\_final = pd.merge(mexico\_race\_22\_laps, mexico\_race\_22\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

mexico\_race\_22\_final['TotalTime'] = mexico\_race\_22\_final['Time'] - mexico\_race\_22.session\_start\_time

mexico\_race\_22\_final = mexico\_race\_22\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

brazil\_race\_22 = ff1.get\_session(2022, 21, 'R')

brazil\_race\_22.load()

brazil\_race\_22\_laps = brazil\_race\_22.laps

brazil\_race\_22\_messages= brazil\_race\_22.race\_control\_messages

brazil\_race\_22\_weather = brazil\_race\_22.weather\_data

brazil\_race\_22\_results= brazil\_race\_22.results

brazil\_race\_22\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

brazil\_race\_22\_final = pd.merge(brazil\_race\_22\_laps, brazil\_race\_22\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

brazil\_race\_22\_final['TotalTime'] = brazil\_race\_22\_final['Time'] - brazil\_race\_22.session\_start\_time

brazil\_race\_22\_final = brazil\_race\_22\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

abu\_dhabi\_race\_22 = ff1.get\_session(2022, 22, 'R')

abu\_dhabi\_race\_22.load()

abu\_dhabi\_race\_22\_laps = abu\_dhabi\_race\_22.laps

abu\_dhabi\_race\_22\_messages= abu\_dhabi\_race\_22.race\_control\_messages

abu\_dhabi\_race\_22\_weather = abu\_dhabi\_race\_22.weather\_data

abu\_dhabi\_race\_22\_results= abu\_dhabi\_race\_22.results

abu\_dhabi\_race\_22\_results.rename(columns={'Abbreviation': 'Driver'}, inplace=True)

abu\_dhabi\_race\_22\_final = pd.merge(abu\_dhabi\_race\_22\_laps, abu\_dhabi\_race\_22\_results[['Driver', 'GridPosition', 'Position', 'Points']], on='Driver', how='left')

abu\_dhabi\_race\_22\_final['TotalTime'] = abu\_dhabi\_race\_22\_final['Time'] - abu\_dhabi\_race\_22.session\_start\_time

abu\_dhabi\_race\_22\_final = abu\_dhabi\_race\_22\_final.groupby('Driver').tail(1000).sort\_values(by='TotalTime')

merge2022\_laps = pd.concat([

bahrain\_race\_22\_laps,

saudi\_arabia\_race\_22\_laps,

australia\_race\_22\_laps,

italy\_race\_22a\_laps,

united\_states\_race\_22a\_laps,

spain\_race\_22\_laps,

monaco\_race\_22\_laps,

azerbaijan\_race\_22\_laps,

canada\_race\_22\_laps,

great\_britain\_race\_22\_laps,

austria\_race\_22\_laps,

france\_race\_22\_laps,

hungary\_race\_22\_laps,

belgium\_race\_22\_laps,

netherlands\_race\_22\_laps,

italy\_race\_22b\_laps,

singapore\_race\_22\_laps,

japan\_race\_22\_laps,

united\_states\_race\_22b\_laps,

mexico\_race\_22\_laps,

brazil\_race\_22\_laps,

abu\_dhabi\_race\_22\_laps],

keys = ['Bahrain',

'SaudiArabia',

'Australia',

'ItalyA',

'UnitedStatesA',

'Spain',

'Monaco',

'Azerbaijan',

'Canada',

'GreatBritain',

'Austria',

'France',

'Hungary',

'Belgium',

'Netherlands',

'ItalyB',

'Singapore',

'Japan',

'UnitedStatesB',

'Mexico',

'Brazil',

'AbuDhabi'],

names=['RACE'])

merge2022\_messages = pd.concat([

bahrain\_race\_22\_messages,

saudi\_arabia\_race\_22\_messages,

australia\_race\_22\_messages,

italy\_race\_22a\_messages,

united\_states\_race\_22a\_messages,

spain\_race\_22\_messages,

monaco\_race\_22\_messages,

azerbaijan\_race\_22\_messages,

canada\_race\_22\_messages,

great\_britain\_race\_22\_messages,

austria\_race\_22\_messages,

france\_race\_22\_messages,

hungary\_race\_22\_messages,

belgium\_race\_22\_messages,

netherlands\_race\_22\_messages,

italy\_race\_22b\_messages,

singapore\_race\_22\_messages,

japan\_race\_22\_messages,

united\_states\_race\_22b\_messages,

mexico\_race\_22\_messages,

brazil\_race\_22\_messages,

abu\_dhabi\_race\_22\_messages],

keys = ['Bahrain',

'SaudiArabia',

'Australia',

'ItalyA',

'UnitedStatesA',

'Spain',

'Monaco',

'Azerbaijan',

'Canada',

'GreatBritain',

'Austria',

'France',

'Hungary',

'Belgium',

'Netherlands',

'ItalyB',

'Singapore',

'Japan',

'UnitedStatesB',

'Mexico',

'Brazil',

'Abu Dhabi'],

names=['RACE'])

merge2022\_weather = pd.concat([

bahrain\_race\_22\_weather,

saudi\_arabia\_race\_22\_weather,

australia\_race\_22\_weather,

italy\_race\_22a\_weather,

united\_states\_race\_22a\_weather,

spain\_race\_22\_weather,

monaco\_race\_22\_weather,

azerbaijan\_race\_22\_weather,

canada\_race\_22\_weather,

great\_britain\_race\_22\_weather,

austria\_race\_22\_weather,

france\_race\_22\_weather,

hungary\_race\_22\_weather,

belgium\_race\_22\_weather,

netherlands\_race\_22\_weather,

italy\_race\_22b\_weather,

singapore\_race\_22\_weather,

japan\_race\_22\_weather,

united\_states\_race\_22b\_weather,

mexico\_race\_22\_weather,

brazil\_race\_22\_weather,

abu\_dhabi\_race\_22\_weather],

keys = ['Bahrain',

'SaudiArabia',

'Australia',

'ItalyA',

'UnitedStatesA',

'Spain',

'Monaco',

'Azerbaijan',

'Canada',

'GreatBritain',

'Austria',

'France',

'Hungary',

'Belgium',

'Netherlands',

'ItalyB',

'Singapore',

'Japan',

'UnitedStatesB',

'Mexico',

'Brazil',

'AbuDhabi'],

names=['RACE'])

merge2022\_results = pd.concat([

bahrain\_race\_22\_results,

saudi\_arabia\_race\_22\_results,

australia\_race\_22\_results,

italy\_race\_22a\_results,

united\_states\_race\_22a\_results,

spain\_race\_22\_results,

monaco\_race\_22\_results,

azerbaijan\_race\_22\_results,

canada\_race\_22\_results,

great\_britain\_race\_22\_results,

austria\_race\_22\_results,

france\_race\_22\_results,

hungary\_race\_22\_results,

belgium\_race\_22\_results,

netherlands\_race\_22\_results,

italy\_race\_22b\_results,

singapore\_race\_22\_results,

japan\_race\_22\_results,

united\_states\_race\_22b\_results,

mexico\_race\_22\_results,

brazil\_race\_22\_results,

abu\_dhabi\_race\_22\_results],

keys = ['Bahrain',

'SaudiArabia',

'Australia',

'ItalyA',

'UnitedStatesA',

'Spain',

'Monaco',

'Azerbaijan',

'Canada',

'GreatBritain',

'Austria',

'France',

'Hungary',

'Belgium',

'Netherlands',

'ItalyB',

'Singapore',

'Japan',

'UnitedStatesB',

'Mexico',

'Brazil',

'AbuDhabi'],

names=['RACE'])

merge2022\_final = pd.concat([

bahrain\_race\_22\_final,

saudi\_arabia\_race\_22\_final,

australia\_race\_22\_final,

italy\_race\_22a\_final,

united\_states\_race\_22a\_final,

spain\_race\_22\_final,

monaco\_race\_22\_final,

azerbaijan\_race\_22\_final,

canada\_race\_22\_final,

great\_britain\_race\_22\_final,

austria\_race\_22\_final,

france\_race\_22\_final,

hungary\_race\_22\_final,

belgium\_race\_22\_final,

netherlands\_race\_22\_final,

italy\_race\_22b\_final,

singapore\_race\_22\_final,

japan\_race\_22\_final,

united\_states\_race\_22b\_final,

mexico\_race\_22\_final,

brazil\_race\_22\_final,

abu\_dhabi\_race\_22\_final],

keys = ['Bahrain',

'SaudiArabia',

'Australia',

'ItalyA',

'UnitedStatesA',

'Spain',

'Monaco',

'Azerbaijan',

'Canada',

'GreatBritain',

'Austria',

'France',

'Hungary',

'Belgium',

'Netherlands',

'ItalyB',

'Singapore',

'Japan',

'UnitedStatesB',

'Mexico',

'Brazil',

'AbuDhabi'],

names=['RACE'])

merge2022\_laps["Year"] = 2022

merge2022\_messages["Year"] = 2022

merge2022\_weather["Year"] = 2022

merge2022\_results["Year"] = 2022

merge2022\_final["Year"] = 2022

#%% merging the yearly data

merge\_total\_laps = pd.concat([

merge2018\_laps,

merge2019\_laps,

merge2020\_laps,

merge2021\_laps,

merge2022\_laps])

merge\_total\_messages = pd.concat([

merge2018\_messages,

merge2019\_messages,

merge2020\_messages,

merge2021\_messages,

merge2022\_messages])

merge\_total\_weather = pd.concat([

merge2018\_weather,

merge2019\_weather,

merge2020\_weather,

merge2021\_weather,

merge2022\_weather])

merge\_total\_results = pd.concat([

merge2018\_results,

merge2019\_results,

merge2020\_results,

merge2021\_results,

merge2022\_results])

merge\_total\_final = pd.concat([

merge2018\_final,

merge2019\_final,

merge2020\_final,

merge2021\_final,

merge2022\_final])

merge\_total\_laps.insert(0, "NewID", range(1, 1 + len(merge\_total\_laps)))

merge\_total\_laps.drop(["Year"], axis=1)

merge\_total\_laps.loc[58170:71068, ["Year"]] = 2022

merge\_total\_messages.insert(0, "NewID", range(1, 1 + len(merge\_total\_messages)))

merge\_total\_messages.drop(["Year"], axis=1)

merge\_total\_messages.loc[58170:71068, ["Year"]] = 2022

merge\_total\_weather.insert(0, "NewID", range(1, 1 + len(merge\_total\_weather)))

merge\_total\_weather.drop(["Year"], axis=1)

merge\_total\_weather.loc[58170:71068, ["Year"]] = 2022

merge\_total\_results.insert(0, "NewID", range(1, 1 + len(merge\_total\_results)))

merge\_total\_results.drop(["Year"], axis=1)

merge\_total\_results.loc[58170:71068, ["Year"]] = 2022

merge\_total\_final.insert(0, "NewID", range(1, 1 + len(merge\_total\_final)))

merge\_total\_final.drop(["Year"], axis=1)

merge\_total\_final.loc[58170:71068, ["Year"]] = 2022

#%% exporting the data to csv

merge\_total\_laps.to\_csv("C:\\Users\\andre\\OneDrive\\Desktop\\totalLaps.csv", na\_rep='NA')

merge\_total\_messages.to\_csv("C:\\Users\\andre\\OneDrive\\Desktop\\messages.csv", na\_rep='NA')

merge\_total\_weather.to\_csv("C:\\Users\\andre\\OneDrive\\Desktop\\weather.csv", na\_rep='NA')

merge\_total\_results.to\_csv("C:\\Users\\andre\\OneDrive\\Desktop\\results.csv", na\_rep='NA')

merge\_total\_final.to\_csv("C:\\Users\\andre\\OneDrive\\Desktop\\final.csv", na\_rep='NA')

#%% pit only dataframe and reducing dimensionality

#making RACE a column

merge\_total\_final.reset\_index(inplace=True)

#calculating gap

merge\_total\_final['Gap'] = merge\_total\_final['TotalTime'] - merge\_total\_final['TotalTime'].shift()

merge\_total\_final['GapSeconds'] = merge\_total\_final['Gap'].dt.total\_seconds()

merge\_total\_final['GapSeconds'] = (merge\_total\_final.GapSeconds.clip(lower=0))

#making pit stop only dataframe and removing the first lap

merge\_total\_final\_pit\_only = merge\_total\_final[pd.notnull(merge\_total\_final['PitInTime'])]

merge\_total\_final\_pit\_only = merge\_total\_final\_pit\_only[merge\_total\_final\_pit\_only['LapNumber'] != 1]

#reducing dimensionality

merge\_total\_final = merge\_total\_final[[

'DriverNumber', 'LapNumber', 'Compound', 'TyreLife', 'FreshTyre', 'Stint', 'Team', 'Year',

'RACE', 'Driver', 'TrackStatus', 'GridPosition', 'Position', 'Points', 'TotalTime', 'GapSeconds']]

merge\_total\_final\_pit\_only = merge\_total\_final\_pit\_only[[

'DriverNumber', 'LapNumber', 'Compound', 'TyreLife', 'FreshTyre', 'Stint', 'Team', 'Year',

'RACE', 'Driver', 'TrackStatus', 'GridPosition', 'Position', 'Points', 'TotalTime', 'GapSeconds']]

merge\_total\_final\_finishers\_only = merge\_total\_final.loc[merge\_total\_final['Points'] != 0]

#transforming string data into categorical/factor https://pbpython.com/categorical-encoding.html

merge\_total\_final\_pit\_only['Team'] = merge\_total\_final\_pit\_only['Team'].astype('category')

merge\_total\_final\_pit\_only["TeamCat"] = merge\_total\_final\_pit\_only["Team"].cat.codes

merge\_total\_final\_pit\_only['Driver'] = merge\_total\_final\_pit\_only['Driver'].astype('category')

merge\_total\_final\_pit\_only["DriverCat"] = merge\_total\_final\_pit\_only["Driver"].cat.codes

merge\_total\_final\_pit\_only['DriverNumber'] = merge\_total\_final\_pit\_only['DriverNumber'].astype('category')

merge\_total\_final\_pit\_only["DriverNumberCat"] = merge\_total\_final\_pit\_only["DriverNumber"].cat.codes

merge\_total\_final\_pit\_only['TrackStatus'] = merge\_total\_final\_pit\_only['TrackStatus'].astype('category')

merge\_total\_final\_pit\_only["TrackStatusCat"] = merge\_total\_final\_pit\_only["TrackStatus"].cat.codes

merge\_total\_final\_pit\_only['Compound'] = merge\_total\_final\_pit\_only['Compound'].astype('category')

merge\_total\_final\_pit\_only["CompoundCat"] = merge\_total\_final\_pit\_only["Compound"].cat.codes

merge\_total\_final\_pit\_only = merge\_total\_final\_pit\_only[[

'DriverNumberCat', 'LapNumber', 'CompoundCat', 'TyreLife', 'FreshTyre', 'Stint', 'TeamCat', 'Year',

'RACE', 'DriverCat', 'TrackStatusCat', 'GridPosition', 'Position', 'Points', 'TotalTime', 'GapSeconds']]

merge\_total\_final\_pit\_only['TotalTime'] = merge\_total\_final\_pit\_only['TotalTime'].dt.total\_seconds()

#%%

#aggregating the data

pit\_in\_time\_lap\_number = merge\_total\_final\_pit\_only.groupby(['Year', 'RACE', 'TeamCat', 'DriverCat']).agg({'LapNumber': ['mean', 'min', 'max', 'count']}).reset\_index()

pit\_in\_time\_lap\_number = pit\_in\_time\_lap\_number.droplevel(0, axis=1)

pit\_in\_time\_lap\_number.reset\_index(inplace=True)

pit\_in\_time\_lap\_number = pit\_in\_time\_lap\_number.astype({'count': 'float'})

pit\_in\_time\_lap\_number.columns = ['Index', 'Year', 'Race', 'Team', 'Driver', 'Mean', 'Min', 'Max', 'NumberOfStops']

#visualizing the normalized value of pit stop frequency

pit\_in\_time\_lap\_number.NumberOfStops.value\_counts(normalize=True).plot(kind='bar')

#visualizing the actual number of pit stop frequency

pit\_in\_time\_lap\_number.NumberOfStops.value\_counts(bins=8).plot(kind='bar')

#analyze tyre life when pit stop is made

pit\_in\_time\_tyre\_life = merge\_total\_final\_pit\_only.groupby(['Year', 'RACE', 'TeamCat', 'DriverCat']).agg({'TyreLife': lambda x: x.tolist(),'LapNumber': ['mean', 'min', 'max', 'count']}).reset\_index()

pit\_in\_time\_tyre\_life = pit\_in\_time\_tyre\_life.droplevel(0, axis=1)

pit\_in\_time\_tyre\_life.reset\_index(inplace=True)

pit\_in\_time\_tyre\_life = pit\_in\_time\_tyre\_life.astype({'count': 'float'})

pit\_in\_time\_tyre\_life.columns = ['Index', 'Year', 'Race', 'Team', 'Driver', 'TyreLife', 'Mean', 'Min', 'Max', 'NumberOfStops']

pit\_in\_time\_tyre\_life.to\_csv("C:\\Users\\andre\\OneDrive\\Desktop\\pit\_in\_time\_tyre\_lifer.csv", na\_rep='NA')

#pit in time plot

pit\_in\_time\_tyre\_life.TyreLife.value\_counts(normalize=False).plot(kind='bar')

pit\_in\_time\_tyre\_life.TyreLife.value\_counts(normalize=True).plot(kind='line')

#%%

#dataframe showing the tyre life when a pitstop was made

pit\_stop\_tyre\_life = pd.DataFrame(pit\_in\_time\_tyre\_life.TyreLife.tolist(), index=pit\_in\_time\_tyre\_life.index)

#combine pitStop\_TyreLife and pit\_in\_time\_tyre\_life into same dataframe

pit\_stop\_df = pd.concat([pit\_in\_time\_tyre\_life, pit\_stop\_tyre\_life], axis=1)

#%%performing the poisson calculation

#https://timeseriesreasoning.com/contents/poisson-regression-model/

poisson\_results\_compound = sm.GLM(merge\_total\_final\_pit\_only['LapNumber'],

merge\_total\_final\_pit\_only['CompoundCat'],

family=sm.families.Poisson()).fit()

poisson\_results\_driver\_number = sm.GLM(merge\_total\_final\_pit\_only['LapNumber'],

merge\_total\_final\_pit\_only['DriverNumberCat'],

family=sm.families.Poisson()).fit()

poisson\_results\_tyre\_life = sm.GLM(merge\_total\_final\_pit\_only['LapNumber'],

merge\_total\_final\_pit\_only['TyreLife'],

family=sm.families.Poisson()).fit()

poisson\_results\_fresh\_tyre = sm.GLM(merge\_total\_final\_pit\_only['LapNumber'],

pd.get\_dummies(merge\_total\_final\_pit\_only['FreshTyre'], dtype = float),

family=sm.families.Poisson()).fit()

poisson\_results\_stint = sm.GLM(merge\_total\_final\_pit\_only['LapNumber'],

merge\_total\_final\_pit\_only['Stint'],

family=sm.families.Poisson()).fit()

poisson\_results\_team = sm.GLM(merge\_total\_final\_pit\_only['LapNumber'],

merge\_total\_final\_pit\_only['TeamCat'],

family=sm.families.Poisson()).fit()

poisson\_results\_driver = sm.GLM(merge\_total\_final\_pit\_only['LapNumber'],

merge\_total\_final\_pit\_only['DriverCat'],

family=sm.families.Poisson()).fit()

poisson\_results\_track\_status = sm.GLM(merge\_total\_final\_pit\_only['LapNumber'],

merge\_total\_final\_pit\_only['TrackStatusCat'],

family=sm.families.Poisson()).fit()

poisson\_results\_total\_time = sm.GLM(merge\_total\_final\_pit\_only['LapNumber'],

merge\_total\_final\_pit\_only['TotalTime'],

family=sm.families.Poisson()).fit()

poisson\_results\_gap = sm.GLM(merge\_total\_final\_pit\_only['LapNumber'],

merge\_total\_final\_pit\_only['GapSeconds'],

family=sm.families.Poisson()).fit()

#viewing poisson results per variable

print(poisson\_results\_compound.summary())

print(poisson\_results\_tyre\_life.summary())

print(poisson\_results\_total\_time.summary())

print(poisson\_results\_stint.summary())

print(poisson\_results\_gap.summary())

print(poisson\_results\_fresh\_tyre.summary())

print(poisson\_results\_team.summary())

print(poisson\_results\_driver.summary())

print(poisson\_results\_driver\_number.summary())

print(poisson\_results\_track\_status.summary())

#further research https://bookdown.org/roback/bookdown-BeyondMLR/ch-poissonreg.html

#%% random forest

merge\_total\_final\_rf = merge\_total\_final\_pit\_only[[

'DriverNumberCat', 'LapNumber', 'CompoundCat', 'TyreLife', 'FreshTyre', 'Stint', 'TeamCat', 'Year',

'DriverCat', 'TrackStatusCat', 'GridPosition', 'Position', 'Points', 'TotalTime', 'GapSeconds']]

y = merge\_total\_final\_rf['LapNumber']

X = merge\_total\_final\_rf.drop(['LapNumber'], axis = 1)

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=44)

from sklearn.ensemble import RandomForestClassifier

rf\_model = RandomForestClassifier(n\_estimators=50, max\_features="auto", random\_state=44)

rf\_model.fit(X\_train, y\_train)

predictions = rf\_model.predict(X\_test)

predictions

importances = rf\_model.feature\_importances\_

columns = X.columns

i = 0

while i < len(columns):

print(f" The importance of feature ' {columns[i]}' is {round(importances[i] \* 100, 3)}%.")

i += 1

accuracy\_score(y\_test, predictions)

confusion\_matrix(y\_test, predictions)

#%%#calculate mean/variance of poisson\_results

merge\_total\_final\_pit\_only['LapNumber'].var() #46.567

merge\_total\_final\_pit\_only['LapNumber'].mean() #14.833

#visualize poisson to justify model choice

fig = merge\_total\_final\_pit\_only['LapNumber'].value\_counts(sort=False).plot(kind='bar')

#group dataset by tyre type and get mean and variance to test for heterogeneity

#if variance > mean in group, look at negative binomial

merge\_total\_final\_pit\_only['CompoundCat'].mean() #0.479

merge\_total\_final\_pit\_only['CompoundCat'].var() #0.297

#%%#frequency graph compound tyres

#sort numerically

merge\_total\_final\_pit\_only['CompoundCat'].value\_counts()[merge\_total\_final\_pit\_only.CompoundCat.unique()].plot(kind='bar')

#%% Building Consul’s Generalized Poison regression model, know as GP-1:

# =============================================================================

# gen\_poisson\_gp1\_cumulative = sm.GeneralizedPoisson(merge\_total\_final\_pit\_only['LapNumber'], ###################

# merge\_total\_final\_pit\_only['Cumulative'],

# p=1)

# gen\_poisson\_gp1\_cumulative\_results = gen\_poisson\_gp1\_cumulative.fit()

# print(gen\_poisson\_gp1\_cumulative\_results.summary())

#

# gen\_poisson\_gp1\_cumulative\_lap = sm.GeneralizedPoisson(merge\_total\_final\_pit\_only['LapNumber'], ###################

# merge\_total\_final\_pit\_only['CumulativeLap'],

# p=1)

# gen\_poisson\_gp1\_cumulative\_lap\_results = gen\_poisson\_gp1\_cumulative\_lap.fit()

# print(gen\_poisson\_gp1\_cumulative\_lap\_results.summary())

#

# gen\_poisson\_gp1\_cumulative\_pit = sm.GeneralizedPoisson(merge\_total\_final\_pit\_only['LapNumber'], ###################

# merge\_total\_final\_pit\_only['CumulativePit'],

# p=1)

# gen\_poisson\_gp1\_cumulative\_pit\_results = gen\_poisson\_gp1\_cumulative\_pit.fit()

# print(gen\_poisson\_gp1\_cumulative\_pit\_results.summary())

#

# gen\_poisson\_gp1\_gap = sm.GeneralizedPoisson(merge\_total\_final\_pit\_only['LapNumber'], ###################

# merge\_total\_final\_pit\_only['Gap'],

# p=1)

# gen\_poisson\_gp1\_gap\_results = gen\_poisson\_gp1\_gap.fit()

# print(gen\_poisson\_gp1\_gap\_results.summary())

#

# gen\_poisson\_gp1\_track\_status = sm.GeneralizedPoisson(merge\_total\_final\_pit\_only['LapNumber'], ###################

# merge\_total\_final\_pit\_only['TrackStatusCat'],

# p=1)

# gen\_poisson\_gp1\_track\_status\_results = gen\_poisson\_gp1\_gap.fit()

# print(gen\_poisson\_gp1\_track\_status\_results.summary())

#

# gen\_poisson\_gp1\_compound = sm.GeneralizedPoisson(merge\_total\_final\_pit\_only['LapNumber'], ###################

# merge\_total\_final\_pit\_only['CompoundCat'],

# p=1)

# gen\_poisson\_gp1\_compound\_results = gen\_poisson\_gp1\_compound.fit()

# print(gen\_poisson\_gp1\_compound\_results.summary())

#

# gen\_poisson\_gp1\_pit\_time = sm.GeneralizedPoisson(merge\_total\_final\_pit\_only['LapNumber'], ###################

# merge\_total\_final\_pit\_only['PitTime'],

# p=1)

# gen\_poisson\_gp1\_pit\_time\_results = gen\_poisson\_gp1\_pit\_time.fit()

# print(gen\_poisson\_gp1\_pit\_time\_results.summary())

#

# gen\_poisson\_gp1\_pit\_out = sm.GeneralizedPoisson(merge\_total\_final\_pit\_only['LapNumber'], ###################

# merge\_total\_final\_pit\_only['PitOutTimeSeconds'],

# p=1)

# gen\_poisson\_gp1\_pit\_out\_results = gen\_poisson\_gp1\_pit\_out.fit()

# print(gen\_poisson\_gp1\_pit\_out\_results.summary())

#

# gen\_poisson\_gp1\_pit\_in = sm.GeneralizedPoisson(merge\_total\_final\_pit\_only['LapNumber'], ###################

# merge\_total\_final\_pit\_only['PitInTimeSeconds'],

# p=1)

# gen\_poisson\_gp1\_pit\_in\_results = gen\_poisson\_gp1\_pit\_in.fit()

# print(gen\_poisson\_gp1\_pit\_in\_results.summary())

#

# =============================================================================

gen\_poisson\_gp1\_stint = sm.GeneralizedPoisson(merge\_total\_final\_pit\_only['LapNumber'],

merge\_total\_final\_pit\_only['Stint'],

p=1)

gen\_poisson\_gp1\_stint\_results = gen\_poisson\_gp1\_stint.fit()

print(gen\_poisson\_gp1\_stint\_results.summary())

gen\_poisson\_gp1\_fresh\_tyre = sm.GeneralizedPoisson(merge\_total\_final\_pit\_only['LapNumber'],

pd.get\_dummies(merge\_total\_final\_pit\_only['FreshTyre'], dtype = float), p=1)

gen\_poisson\_gp1\_fresh\_tyre\_results = gen\_poisson\_gp1\_fresh\_tyre.fit()

print(gen\_poisson\_gp1\_fresh\_tyre\_results.summary())

gen\_poisson\_gp1\_tyre\_life = sm.GeneralizedPoisson(merge\_total\_final\_pit\_only['LapNumber'],

merge\_total\_final\_pit\_only['TyreLife'],

p=1)

gen\_poisson\_gp1\_tyre\_life\_results = gen\_poisson\_gp1\_tyre\_life.fit()

print(gen\_poisson\_gp1\_tyre\_life\_results.summary())

gen\_poisson\_gp1\_team = sm.GeneralizedPoisson(merge\_total\_final\_pit\_only['LapNumber'],

merge\_total\_final\_pit\_only['TeamCat'],

p=1)

gen\_poisson\_gp1\_team\_results = gen\_poisson\_gp1\_team.fit()

print(gen\_poisson\_gp1\_team\_results.summary())

gen\_poisson\_gp1\_driver = sm.GeneralizedPoisson(merge\_total\_final\_pit\_only['LapNumber'],

merge\_total\_final\_pit\_only['DriverCat'],

p=1)

gen\_poisson\_gp1\_driver\_results = gen\_poisson\_gp1\_driver.fit()

print(gen\_poisson\_gp1\_driver\_results.summary())

gen\_poisson\_gp1\_driver\_number = sm.GeneralizedPoisson(merge\_total\_final\_pit\_only['LapNumber'],

merge\_total\_final\_pit\_only['DriverNumberCat'],

p=1)

gen\_poisson\_gp1\_driver\_number\_results = gen\_poisson\_gp1\_driver\_number.fit()

print(gen\_poisson\_gp1\_driver\_number\_results.summary())

#%% negative binomial time, holla

#https://timeseriesreasoning.com/contents/negative-binomial-regression-model/

merge\_total\_final\_pit\_only['FreshTyre'] = merge\_total\_final\_pit\_only['FreshTyre'].astype('category')

merge\_total\_final\_pit\_only["FreshTyreCat"] = merge\_total\_final\_pit\_only["FreshTyre"].cat.codes

#create the training and testing data sets

mask = np.random.rand(len(merge\_total\_final\_pit\_only)) < 0.8

merge\_total\_final\_pit\_only\_train = merge\_total\_final\_pit\_only[mask]

merge\_total\_final\_pit\_only\_test = merge\_total\_final\_pit\_only[~mask]

#Setup the regression expression in patsy notation. We are telling patsy that BB\_COUNT is our dependent variable and it depends on the regression variables: DAY, DAY\_OF\_WEEK, MONTH, HIGH\_T, LOW\_T and PRECIP

expr = """LapNumber ~ DriverNumberCat + CompoundCat + TyreLife + FreshTyreCat+

Stint + GapSeconds + TeamCat + DriverCat + TrackStatusCat + Points + TotalTime"""

#Set up the X and y matrices for the training and testing data sets

y\_train, X\_train = dmatrices(expr, merge\_total\_final\_pit\_only\_train, return\_type='dataframe')

y\_test, X\_test = dmatrices(expr, merge\_total\_final\_pit\_only\_test, return\_type='dataframe')

poisson\_training\_results = sm.GLM(y\_train, X\_train, family=sm.families.Poisson()).fit()

print(poisson\_training\_results.summary())

#Add the λ vector as a new column called 'BB\_LAMBDA' to the Data Frame of the training data set

merge\_total\_final\_pit\_only\_train['LapNumber'] = poisson\_training\_results.mu

print(poisson\_training\_results.mu)

print(len(poisson\_training\_results.mu))

#add a derived column called 'AUX\_OLS\_DEP' to the pandas Data Frame. This new column will store the values of the dependent variable of the OLS regression

merge\_total\_final\_pit\_only\_train['AUX\_OLS\_DEP'] = merge\_total\_final\_pit\_only\_train.apply(lambda x: ((x['LapNumber'] - x['LapNumber'])\*\*2 - x['LapNumber']) / x['LapNumber'], axis=1)

#use patsy to form the model specification for the OLSR

ols\_expr = """AUX\_OLS\_DEP ~ LapNumber - 1"""

#Configure and fit the OLSR model

aux\_olsr\_results = smf.ols(ols\_expr, merge\_total\_final\_pit\_only\_train).fit()

print(aux\_olsr\_results.params) #-.06 (why negative, should be positive?)

aux\_olsr\_results.tvalues #test significance of alpha -14.670

#using t-value calculator (https://goodcalculators.com/student-t-value-calculator/))

#positive .06 with 36 degrees of freedom (from len of poisson training.mu) = 2-tailed 1.94 and right tailed 1.59

#should 35 be degrees of freedom (len - 1) or should we use total dataset (47 - 1?)

#either way, t-values are above 1.9 and 1.5

#train the NB2 model on the training data set

nb2\_training\_results = sm.GLM(y\_train, X\_train,family=sm.families.NegativeBinomial(alpha=-aux\_olsr\_results.params[0])).fit()

print(nb2\_training\_results.summary())

#make some predictions using our trained NB2 model

nb2\_predictions = nb2\_training\_results.get\_prediction(X\_test)

predictions\_summary\_frame = nb2\_predictions.summary\_frame()

print(predictions\_summary\_frame)

#plot the predicted counts versus the actual counts for the test data

predicted\_counts=predictions\_summary\_frame['mean']

actual\_counts = y\_test['LapNumber']

fig = plt.figure()

fig.suptitle('Predicted versus actual lap number when pit stop made')

predicted, = plt.plot(X\_test.index, predicted\_counts, 'go-', label='Predicted counts')

actual, = plt.plot(X\_test.index, actual\_counts, 'ro-', label='Actual counts')

plt.legend(handles=[predicted, actual])

plt.show()

#%%#Build Famoye's Restricted Generalized Poison regression model, know as GP-2

# =============================================================================

# gen\_poisson\_gp2 = sm.GeneralizedPoisson(y\_train, X\_train, p=2)

#

# #Fit the model

# gen\_poisson\_gp2\_results = gen\_poisson\_gp2.fit()

#

# #print the results

# print(gen\_poisson\_gp2\_results.summary())

# =============================================================================

#

#%% writing pit only dataframe to csv

merge\_total\_final\_pit\_only.to\_csv("C:\\Users\\andre\\OneDrive\\Desktop\\final\_pit\_only.csv", na\_rep='NA')

abu\_dhabi\_race\_19\_laps.head()

**Appendix B – Python Course Visualization**

# -\*- coding: utf-8 -\*-

"""

Created on Tue Apr 18 10:07:55 2023

@author: andre

"""

#%% visualizing 2018 fastest lap

#graphing the track, with the gear shifts used by the fastest driver

#not relevant to the thesis, but that was what the example used

#https://theoehrly.github.io/Fast-F1/examples\_gallery/plot\_gear\_shifts\_on\_track.html#sphx-glr-examples-gallery-plot-gear-shifts-on-track-py

#the first two races of 2018 are not available. here is the code for testing

#neither are austria2020b or belgium2021

#test = ff1.get\_session(2018, 2, 'R')

#test.load()

#lap = china\_race\_18.laps.pick\_fastest()

#tel = lap.get\_telemetry()

#China - 2018

lapChina18 = china\_race\_18.laps.pick\_fastest()

tel = lapChina18.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapChina18['Driver']} - {china\_race\_18.event['EventName']} {china\_race\_18.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Azerbaijan - 2018

lapAzerbaijan18 = azerbaijan\_race\_18.laps.pick\_fastest()

tel = lapAzerbaijan18.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapAzerbaijan18['Driver']} - {azerbaijan\_race\_18.event['EventName']} {azerbaijan\_race\_18.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Spain - 2018

lapSpain18 = spain\_race\_18.laps.pick\_fastest()

tel = lapSpain18.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapSpain18['Driver']} - {spain\_race\_18.event['EventName']} {spain\_race\_18.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Monaco - 2018

lapMonaco18 = monaco\_race\_18.laps.pick\_fastest()

tel = lapMonaco18.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapMonaco18['Driver']} - {monaco\_race\_18.event['EventName']} {monaco\_race\_18.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Canada - 2018

lapCanada18 = canada\_race\_18.laps.pick\_fastest()

tel = lapCanada18.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapCanada18['Driver']} - {canada\_race\_18.event['EventName']} {canada\_race\_18.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#France - 2018

lapFrance18 = france\_race\_18.laps.pick\_fastest()

tel = lapFrance18.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapFrance18['Driver']} - {france\_race\_18.event['EventName']} {france\_race\_18.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Austria - 2018

lapAustria18 = austria\_race\_18.laps.pick\_fastest()

tel = lapAustria18.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapAustria18['Driver']} - {austria\_race\_18.event['EventName']} {austria\_race\_18.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Great Britain - 2018

lapGB18 = great\_britain\_race\_18.laps.pick\_fastest()

tel = lapGB18.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapGB18['Driver']} - {great\_britain\_race\_18.event['EventName']} {great\_britain\_race\_18.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Germany - 2018

lapGermany18 = germany\_race\_18.laps.pick\_fastest()

tel = lapGermany18.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapGermany18['Driver']} - {germany\_race\_18.event['EventName']} {germany\_race\_18.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Hungary - 2018

lapHungary18 = hungary\_race\_18.laps.pick\_fastest()

tel = lapHungary18.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapHungary18['Driver']} - {hungary\_race\_18.event['EventName']} {hungary\_race\_18.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Belgium - 2018

lapBelgium18 = belgium\_race\_18.laps.pick\_fastest()

tel = lapBelgium18.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapBelgium18['Driver']} - {belgium\_race\_18.event['EventName']} {belgium\_race\_18.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Italy - 2018

lapItaly18 = italy\_race\_18.laps.pick\_fastest()

tel = lapItaly18.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapItaly18['Driver']} - {italy\_race\_18.event['EventName']} {italy\_race\_18.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Singapore - 2018

lapSingapore18 = singapore\_race\_18.laps.pick\_fastest()

tel = lapSingapore18.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapSingapore18['Driver']} - {singapore\_race\_18.event['EventName']} {singapore\_race\_18.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Russia - 2018

lapRussia18 = russia\_race\_18.laps.pick\_fastest()

tel = lapRussia18.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapRussia18['Driver']} - {russia\_race\_18.event['EventName']} {russia\_race\_18.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Japan - 2018

lapJapan18 = japan\_race\_18.laps.pick\_fastest()

tel = lapJapan18.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapJapan18['Driver']} - {japan\_race\_18.event['EventName']} {japan\_race\_18.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#United States - 2018

lapUS18 = united\_states\_race\_18.laps.pick\_fastest()

tel = lapUS18.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapUS18['Driver']} - {united\_states\_race\_18.event['EventName']} {united\_states\_race\_18.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Mexico - 2018

lapMexico18 = mexico\_race\_18.laps.pick\_fastest()

tel = lapMexico18.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapMexico18['Driver']} - {mexico\_race\_18.event['EventName']} {mexico\_race\_18.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Brazil - 2018

lapBrazil18 = brazil\_race\_18.laps.pick\_fastest()

tel = lapBrazil18.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapBrazil18['Driver']} - {brazil\_race\_18.event['EventName']} {brazil\_race\_18.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#UAE - 2018

lapUAE18 = united\_arab\_emirates\_race\_18.laps.pick\_fastest()

tel = lapUAE18.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapUAE18['Driver']} - {united\_arab\_emirates\_race\_18.event['EventName']} {united\_arab\_emirates\_race\_18.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#%% visualizing 2019 fastest lap

#Australia - 2019

lapAustralia19 = australia\_race\_19.laps.pick\_fastest()

tel = lapAustralia19.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapAustralia19['Driver']} - {australia\_race\_19.event['EventName']} {australia\_race\_19.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Bahrain - 2019

lapBahrain19 = bahrain\_race\_19.laps.pick\_fastest()

tel = lapBahrain19.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapBahrain19['Driver']} - {bahrain\_race\_19.event['EventName']} {bahrain\_race\_19.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#China - 2019

lapChina19 = china\_race\_19.laps.pick\_fastest()

tel = lapChina19.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapChina19['Driver']} - {china\_race\_19.event['EventName']} {china\_race\_19.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Azerbaijan - 2019

lapAzerbaijan19 = azerbaijan\_race\_19.laps.pick\_fastest()

tel = lapAzerbaijan19.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapAzerbaijan19['Driver']} - {azerbaijan\_race\_19.event['EventName']} {azerbaijan\_race\_19.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Spain - 2019

lapSpain19 = spain\_race\_19.laps.pick\_fastest()

tel = lapSpain19.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapSpain19['Driver']} - {spain\_race\_19.event['EventName']} {spain\_race\_19.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Monaco - 2019

lapMonaco19 = monaco\_race\_19.laps.pick\_fastest()

tel = lapMonaco19.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapMonaco19['Driver']} - {monaco\_race\_19.event['EventName']} {monaco\_race\_19.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Canada - 2019

lapCanada19 = canada\_race\_19.laps.pick\_fastest()

tel = lapCanada19.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapCanada19['Driver']} - {canada\_race\_19.event['EventName']} {canada\_race\_19.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#France - 2019

lapFrance19 = france\_race\_19.laps.pick\_fastest()

tel = lapFrance19.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapFrance19['Driver']} - {france\_race\_19.event['EventName']} {france\_race\_19.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Austria - 2019

lapAustria19 = austria\_race\_19.laps.pick\_fastest()

tel = lapAustria19.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapAustria19['Driver']} - {austria\_race\_19.event['EventName']} {austria\_race\_19.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Great Britain - 2019

lapGB19 = great\_britain\_race\_19.laps.pick\_fastest()

tel = lapGB19.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapGB19['Driver']} - {great\_britain\_race\_19.event['EventName']} {great\_britain\_race\_19.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Germany - 2019

lapGermany19 = germany\_race\_19.laps.pick\_fastest()

tel = lapGermany19.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapGermany19['Driver']} - {germany\_race\_19.event['EventName']} {germany\_race\_19.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Hungary - 2019

lapHungary19 = hungary\_race\_19.laps.pick\_fastest()

tel = lapHungary19.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapHungary19['Driver']} - {hungary\_race\_19.event['EventName']} {hungary\_race\_19.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Belgium - 2019

lapBelgium19 = belgium\_race\_19.laps.pick\_fastest()

tel = lapBelgium19.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapBelgium19['Driver']} - {belgium\_race\_19.event['EventName']} {belgium\_race\_19.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Italy - 2019

lapItaly19 = italy\_race\_19.laps.pick\_fastest()

tel = lapItaly19.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapItaly19['Driver']} - {italy\_race\_19.event['EventName']} {italy\_race\_19.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Singapore - 2019

lapSingapore19 = singapore\_race\_19.laps.pick\_fastest()

tel = lapSingapore19.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapSingapore19['Driver']} - {singapore\_race\_19.event['EventName']} {singapore\_race\_19.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Russia - 2019

lapRussia19 = russia\_race\_19.laps.pick\_fastest()

tel = lapRussia19.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapRussia19['Driver']} - {russia\_race\_19.event['EventName']} {russia\_race\_19.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Japan - 2019

lapJapan19 = japan\_race\_19.laps.pick\_fastest()

tel = lapJapan19.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapJapan19['Driver']} - {japan\_race\_19.event['EventName']} {japan\_race\_19.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Mexico - 2019

lapMexico19 = mexico\_race\_19.laps.pick\_fastest()

tel = lapMexico19.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapMexico19['Driver']} - {mexico\_race\_19.event['EventName']} {mexico\_race\_19.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#United States - 2019

lapUS19 = united\_states\_race\_19.laps.pick\_fastest()

tel = lapUS19.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapUS19['Driver']} - {united\_states\_race\_19.event['EventName']} {united\_states\_race\_19.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Brazil - 2019

lapBrazil19 = brazil\_race\_19.laps.pick\_fastest()

tel = lapBrazil19.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapBrazil19['Driver']} - {brazil\_race\_19.event['EventName']} {brazil\_race\_19.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Abu Dhabi - 2019

lapAbu19 = abu\_dhabi\_race\_19.laps.pick\_fastest()

tel = lapAbu19.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapAbu19['Driver']} - {abu\_dhabi\_race\_19.event['EventName']} {abu\_dhabi\_race\_19.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#%% visualizing 2020 fastest lap

#Austria 2020 - First Race

lapAustria20a = austria\_race\_20a.laps.pick\_fastest()

tel = lapAustria20a.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapAustria20a['Driver']} - {austria\_race\_20a.event['EventName']} {austria\_race\_20a.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Great Britain 2020 - First Race

lapGB20a = great\_britain\_race\_20a.laps.pick\_fastest()

tel = lapGB20a.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapGB20a['Driver']} - {great\_britain\_race\_20a.event['EventName']} {great\_britain\_race\_20a.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Great Britain 2020 - Second Race

lapGB20b = great\_britain\_race\_20b.laps.pick\_fastest()

tel = lapGB20b.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapGB20b['Driver']} - {great\_britain\_race\_20b.event['EventName']} {great\_britain\_race\_20b.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Belgium 2020

lapBelgium20= belgium\_race\_20.laps.pick\_fastest()

tel = lapBelgium20.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapBelgium20['Driver']} - {belgium\_race\_20.event['EventName']} {belgium\_race\_20.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Russia 2020

lapRussia20= russia\_race\_20.laps.pick\_fastest()

tel = lapRussia20.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapRussia20['Driver']} - {russia\_race\_20.event['EventName']} {russia\_race\_20.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Gemany 2020

lapGemany20= germany\_race\_20.laps.pick\_fastest()

tel = lapGemany20.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapGemany20['Driver']} - {germany\_race\_20.event['EventName']} {germany\_race\_20.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Portugal 2020

lapPortugal20= portugal\_race\_20.laps.pick\_fastest()

tel = lapPortugal20.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapPortugal20['Driver']} - {portugal\_race\_20.event['EventName']} {portugal\_race\_20.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Italy 2020 - first race

lapItaly20a = italy\_race\_20a.laps.pick\_fastest()

tel = lapItaly20a.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapItaly20a['Driver']} - {italy\_race\_20a.event['EventName']} {italy\_race\_20a.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Italy 2020 - second race

lapItaly20b = italy\_race\_20b.laps.pick\_fastest()

tel = lapItaly20b.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapItaly20b['Driver']} - {italy\_race\_20b.event['EventName']} {italy\_race\_20b.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Italy 2020 - third race

lapItaly20c = italy\_race\_20c.laps.pick\_fastest()

tel = lapItaly20c.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapItaly20c['Driver']} - {italy\_race\_20c.event['EventName']} {italy\_race\_20c.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Turkey 2020

lapTurkey20 = turkey\_race\_20.laps.pick\_fastest()

tel = lapTurkey20.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapTurkey20['Driver']} - {turkey\_race\_20.event['EventName']} {turkey\_race\_20.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Bahrain 2020 - first race

lapBahrain20a = bahrain\_race\_20a.laps.pick\_fastest()

tel = lapBahrain20a.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapBahrain20a['Driver']} - {bahrain\_race\_20a.event['EventName']} {bahrain\_race\_20a.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Bahrain 2020 - second race

lapBahrain20b = bahrain\_race\_20b.laps.pick\_fastest()

tel = lapBahrain20b.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapBahrain20b['Driver']} - {bahrain\_race\_20b.event['EventName']} {bahrain\_race\_20b.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Abu Dhabi 2020

lapAbu20 = abu\_dhabi\_race\_20.laps.pick\_fastest()

tel = lapAbu20.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapAbu20['Driver']} - {abu\_dhabi\_race\_20.event['EventName']} {abu\_dhabi\_race\_20.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#%% visualizing 2021 fastest lap

#Bahrain 2021

lapBahrain21 = bahrain\_race\_21.laps.pick\_fastest()

tel = lapBahrain21.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapBahrain21['Driver']} - {bahrain\_race\_21.event['EventName']} {bahrain\_race\_21.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Italy 2021 - first race

lapItaly21a = italy\_race\_21a.laps.pick\_fastest()

tel = lapItaly21a.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapItaly21a['Driver']} - {italy\_race\_21a.event['EventName']} {italy\_race\_21a.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Portugal 2021

lapPortugal21 = portugal\_race\_21.laps.pick\_fastest()

tel = lapPortugal21.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapPortugal21['Driver']} - {portugal\_race\_21.event['EventName']} {portugal\_race\_21.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Spain 2021

lapSpain21 = spain\_race\_21.laps.pick\_fastest()

tel = lapSpain21.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapSpain21['Driver']} - {spain\_race\_21.event['EventName']} {spain\_race\_21.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Monaco 2021

lapMonaco21 = monaco\_race\_21.laps.pick\_fastest()

tel = lapMonaco21.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapMonaco21['Driver']} - {monaco\_race\_21.event['EventName']} {monaco\_race\_21.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Azerbaijan 2021

lapAzerbaijan21 = azerbaijan\_race\_21.laps.pick\_fastest()

tel = lapAzerbaijan21.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapAzerbaijan21['Driver']} - {azerbaijan\_race\_21.event['EventName']} {azerbaijan\_race\_21.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#France 2021

lapFrance21 = france\_race\_21.laps.pick\_fastest()

tel = lapFrance21.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapFrance21['Driver']} - {france\_race\_21.event['EventName']} {france\_race\_21.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Austria 2021 - first race

lapAustria21a = austria\_race\_21a.laps.pick\_fastest()

tel = lapAustria21a.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapAustria21a['Driver']} - {austria\_race\_21a.event['EventName']} {austria\_race\_21a.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Austria 2021 - second race

lapAustria21b = austria\_race\_21b.laps.pick\_fastest()

tel = lapAustria21b.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapAustria21b['Driver']} - {austria\_race\_21b.event['EventName']} {austria\_race\_21b.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Great Britain 2021

lapGB21= great\_britain\_race\_21.laps.pick\_fastest()

tel = lapGB21.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapGB21['Driver']} - {great\_britain\_race\_21.event['EventName']} {great\_britain\_race\_21.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Hungary 2021

lapHungary21= hungary\_race\_21.laps.pick\_fastest()

tel = lapHungary21.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapHungary21['Driver']} - {hungary\_race\_21.event['EventName']} {hungary\_race\_21.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Netherlands 2021

lapNetherlands21= netherlands\_race\_21.laps.pick\_fastest()

tel = lapNetherlands21.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapNetherlands21['Driver']} - {netherlands\_race\_21.event['EventName']} {netherlands\_race\_21.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Italy 2021 - second race

lapItaly21b= italy\_race\_21b.laps.pick\_fastest()

tel = lapItaly21b.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapItaly21b['Driver']} - {italy\_race\_21b.event['EventName']} {italy\_race\_21b.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Russia 2021

lapRussia21= russia\_race\_21.laps.pick\_fastest()

tel = lapRussia21.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapRussia21['Driver']} - {russia\_race\_21.event['EventName']} {russia\_race\_21.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Turkey 2021

lapTurkey21= turkey\_race\_21.laps.pick\_fastest()

tel = lapTurkey21.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapTurkey21['Driver']} - {turkey\_race\_21.event['EventName']} {turkey\_race\_21.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#United States 2021

lapUS21= united\_states\_race\_21.laps.pick\_fastest()

tel = lapUS21.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapUS21['Driver']} - {united\_states\_race\_21.event['EventName']} {united\_states\_race\_21.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Mexico 2021

lapMexico21= mexico\_race\_21.laps.pick\_fastest()

tel = lapMexico21.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapMexico21['Driver']} - {mexico\_race\_21.event['EventName']} {mexico\_race\_21.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Brazil 2021

lapBrazil21= brazil\_race\_21.laps.pick\_fastest()

tel = lapBrazil21.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapBrazil21['Driver']} - {brazil\_race\_21.event['EventName']} {brazil\_race\_21.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Qatar 2021

lapQatar21= qatar\_race\_21.laps.pick\_fastest()

tel = lapQatar21.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapQatar21['Driver']} - {qatar\_race\_21.event['EventName']} {qatar\_race\_21.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Saudi Arabia 2021

lapSaudi21= saudi\_arabia\_race\_21.laps.pick\_fastest()

tel = lapSaudi21.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapSaudi21['Driver']} - {saudi\_arabia\_race\_21.event['EventName']} {saudi\_arabia\_race\_21.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Abu 2021

lapAbu21= abu\_dhabi\_race\_21.laps.pick\_fastest()

tel = lapAbu21.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapAbu21['Driver']} - {abu\_dhabi\_race\_21.event['EventName']} {abu\_dhabi\_race\_21.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#%% visualizing 2022 fastest lap

#Bahrain 2022

lapBahrain22 = bahrain\_race\_22.laps.pick\_fastest()

tel = lapBahrain22.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapBahrain22['Driver']} - {bahrain\_race\_22.event['EventName']} {bahrain\_race\_22.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Saudi Arabia 2022

lapSaudi22 = saudi\_arabia\_race\_22.laps.pick\_fastest()

tel = lapSaudi22.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapSaudi22['Driver']} - {saudi\_arabia\_race\_22.event['EventName']} {saudi\_arabia\_race\_22.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Australia 2022

lapAustralia22 = australia\_race\_22.laps.pick\_fastest()

tel = lapAustralia22.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapAustralia22['Driver']} - {australia\_race\_22.event['EventName']} {australia\_race\_22.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Italy 2022

lapItaly22a = italy\_race\_22a.laps.pick\_fastest()

tel = lapItaly22a.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapItaly22a['Driver']} - {italy\_race\_22a.event['EventName']} {italy\_race\_22a.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#US 2022 - first race

lapUS22a = united\_states\_race\_22a.laps.pick\_fastest()

tel = lapUS22a.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapUS22a['Driver']} - {united\_states\_race\_22a.event['EventName']} {united\_states\_race\_22a.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Spain 2022

lapSpain22 = spain\_race\_22.laps.pick\_fastest()

tel = lapSpain22.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapSpain22['Driver']} - {spain\_race\_22.event['EventName']} {spain\_race\_22.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Monaco 2022

lapMonaco22 = monaco\_race\_22.laps.pick\_fastest()

tel = lapMonaco22.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapMonaco22['Driver']} - {monaco\_race\_22.event['EventName']} {monaco\_race\_22.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Azerbaijan 2022

lapAzerbaijan22 = azerbaijan\_race\_22.laps.pick\_fastest()

tel = lapAzerbaijan22.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapAzerbaijan22['Driver']} - {azerbaijan\_race\_22.event['EventName']} {azerbaijan\_race\_22.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Canada 2022

lapCanada22 = canada\_race\_22.laps.pick\_fastest()

tel = lapCanada22.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapCanada22['Driver']} - {canada\_race\_22.event['EventName']} {canada\_race\_22.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Great Britain 2022

lapGB22 = great\_britain\_race\_22.laps.pick\_fastest()

tel = lapGB22.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapGB22['Driver']} - {great\_britain\_race\_22.event['EventName']} {great\_britain\_race\_22.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Austria 2022

lapAustria22 = austria\_race\_22.laps.pick\_fastest()

tel = lapAustria22.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapAustria22['Driver']} - {austria\_race\_22.event['EventName']} {austria\_race\_22.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#France 2022

lapFrance22 = france\_race\_22.laps.pick\_fastest()

tel = lapFrance22.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapFrance22['Driver']} - {france\_race\_22.event['EventName']} {france\_race\_22.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Hungary 2022

lapHungary22 = hungary\_race\_22.laps.pick\_fastest()

tel = lapHungary22.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapHungary22['Driver']} - {hungary\_race\_22.event['EventName']} {hungary\_race\_22.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Belgium 2022

lapBelgium22 = belgium\_race\_22.laps.pick\_fastest()

tel = lapBelgium22.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapBelgium22['Driver']} - {belgium\_race\_22.event['EventName']} {belgium\_race\_22.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Netherlands 2022

lapNetherlands22 = netherlands\_race\_22.laps.pick\_fastest()

tel = lapNetherlands22.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapNetherlands22['Driver']} - {netherlands\_race\_22.event['EventName']} {netherlands\_race\_22.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Italy 2022

lapItaly22b = italy\_race\_22b.laps.pick\_fastest()

tel = lapItaly22b.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapItaly22b['Driver']} - {italy\_race\_22b.event['EventName']} {italy\_race\_22b.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Singapore 2022

lapSingapore22 = singapore\_race\_22.laps.pick\_fastest()

tel = lapSingapore22.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapSingapore22['Driver']} - {singapore\_race\_22.event['EventName']} {singapore\_race\_22.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Japan 2022

lapJapan22 = japan\_race\_22.laps.pick\_fastest()

tel = lapJapan22.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapJapan22['Driver']} - {japan\_race\_22.event['EventName']} {japan\_race\_22.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#US 2022 - second race

lapUS22b = united\_states\_race\_22b.laps.pick\_fastest()

tel = lapUS22b.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapUS22b['Driver']} - {united\_states\_race\_22b.event['EventName']} {united\_states\_race\_22b.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#Mexico 2022

lapMexico22 = mexico\_race\_22.laps.pick\_fastest()

tel = lapMexico22.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapMexico22['Driver']} - {mexico\_race\_22.event['EventName']} {mexico\_race\_22.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#brazil 2022

lapBrazil22 = brazil\_race\_22.laps.pick\_fastest()

tel = lapBrazil22.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapBrazil22['Driver']} - {brazil\_race\_22.event['EventName']} {brazil\_race\_22.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#abu dhabi 2022

lapAbu22 = abu\_dhabi\_race\_22.laps.pick\_fastest()

tel = lapAbu22.get\_telemetry()

x = np.array(tel['X'].values)

y = np.array(tel['Y'].values)

points = np.array([x, y]).T.reshape(-1, 1, 2)

segments = np.concatenate([points[:-1], points[1:]], axis=1)

gear = tel['nGear'].to\_numpy().astype(float)

cmap = cm.get\_cmap('Paired')

lc\_comp = LineCollection(segments, norm=plt.Normalize(1, cmap.N+1), cmap=cmap)

lc\_comp.set\_array(gear)

lc\_comp.set\_linewidth(4)

plt.gca().add\_collection(lc\_comp)

plt.axis('equal')

plt.tick\_params(labelleft=False, left=False, labelbottom=False, bottom=False)

title = plt.suptitle(

f"Fastest Lap Gear Shift Visualization\n"

f"{lapAbu22['Driver']} - {abu\_dhabi\_race\_22.event['EventName']} {abu\_dhabi\_race\_22.event.year}"

)

cbar = plt.colorbar(mappable=lc\_comp, label="Gear", boundaries=np.arange(1, 10))

cbar.set\_ticks(np.arange(1.5, 9.5))

cbar.set\_ticklabels(np.arange(1, 9))

plt.show()

#%% visualizing 2018 tyre

#visualize race strategy

#https://medium.com/towards-formula-1-analysis/visualizing-formula-1-race-strategies-in-python-using-fastf1-pandas-and-matplotlib-95fe6b3298fa

#find a way to numerically analyze to see where divergences occur (driver, team, course, etc)

#is there a way to do this using the mergeTotal dataframe? don't want excessive code

#figure out why the color doesn't work in the inner for loop

# =============================================================================

# australia\_race\_18\_laps,

# bahrain\_race\_18\_laps,

# china\_race\_18\_laps,

# azerbaijan\_race\_18\_laps,

# spain\_race\_18\_laps,

# monaco\_race\_18\_laps,

# canada\_race\_18\_laps,

# france\_race\_18\_laps,

# austria\_race\_18\_laps,

# great\_britain\_race\_18\_laps,

# germany\_race\_18\_laps,

# hungary\_race\_18\_laps,

# belgium\_race\_18\_laps,

# italy\_race\_18\_laps,

# singapore\_race\_18\_laps,

# russia\_race\_18\_laps,

# japan\_race\_18\_laps,

# united\_states\_race\_18\_laps,

# mexico\_race\_18\_laps,

# brazil\_race\_18\_laps,

# united\_arab\_emirates\_race\_18\_laps

# =============================================================================

compound\_colors = {

'SOFT': 'YELLOW',

'MEDIUM': 'WHITE',

'HARD': 'BLUE',

'INTERMEDIATE': 'GREY',

'WET': 'BLACK'

}

driver\_stints = australia\_race\_18\_laps[['Team', 'Driver', 'DriverNumber', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, australia\_race\_18\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in australia\_race\_18\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Australia 2018")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = bahrain\_race\_18\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, bahrain\_race\_18\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in bahrain\_race\_18\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Bahrain 2018")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = china\_race\_18\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, china\_race\_18\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in china\_race\_18\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("China 2018")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = azerbaijan\_race\_18\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, azerbaijan\_race\_18\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in azerbaijan\_race\_18\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Azerbaijan 2018")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = spain\_race\_18\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, spain\_race\_18\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in spain\_race\_18\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

#plt.title(f'Race strategy - {circuit} {year}')

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = monaco\_race\_18\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, monaco\_race\_18\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in monaco\_race\_18\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Monaco 2018")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = canada\_race\_18\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, canada\_race\_18\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in canada\_race\_18\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Canada 2018")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = france\_race\_18\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, france\_race\_18\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in france\_race\_18\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("France 2018")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = austria\_race\_18\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, austria\_race\_18\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in austria\_race\_18\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Austria 2018")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = great\_britain\_race\_18\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, great\_britain\_race\_18\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in great\_britain\_race\_18\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Great Britain 2018")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = germany\_race\_18\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, germany\_race\_18\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in germany\_race\_18\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Germany 2018")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = hungary\_race\_18\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, hungary\_race\_18\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in hungary\_race\_18\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Hungary 2018")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = belgium\_race\_18\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, belgium\_race\_18\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in belgium\_race\_18\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Belgium 2018")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = italy\_race\_18\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, italy\_race\_18\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in italy\_race\_18\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Italy 2018")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = singapore\_race\_18\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, singapore\_race\_18\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in singapore\_race\_18\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Singapore 2018")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = russia\_race\_18\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, russia\_race\_18\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in russia\_race\_18\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Russia 2018")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = japan\_race\_18\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, japan\_race\_18\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in japan\_race\_18\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Japan 2018")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = united\_states\_race\_18\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, united\_states\_race\_18\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in united\_states\_race\_18\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("USA 2018")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = mexico\_race\_18\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, mexico\_race\_18\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in mexico\_race\_18\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Mexico 2018")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = brazil\_race\_18\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, brazil\_race\_18\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in brazil\_race\_18\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Brazil 2018")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = united\_arab\_emirates\_race\_18\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, united\_arab\_emirates\_race\_18\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in united\_arab\_emirates\_race\_18\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("United Arab Emirates 2018")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

#%% visualizing 2019 tyre

driver\_stints = australia\_race\_19\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, australia\_race\_19\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in australia\_race\_19\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Australia 2019")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = bahrain\_race\_19\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, bahrain\_race\_19\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in bahrain\_race\_19\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Bahrain 2019")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = china\_race\_19\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, china\_race\_19\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in china\_race\_19\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("China 2019")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = azerbaijan\_race\_19\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, azerbaijan\_race\_19\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in azerbaijan\_race\_19\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Azerbaijan 2019")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = spain\_race\_19\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, spain\_race\_19\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in spain\_race\_19\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Spain 2019")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = monaco\_race\_19\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, monaco\_race\_19\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in monaco\_race\_19\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Monaco 2019")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = canada\_race\_19\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, canada\_race\_19\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in canada\_race\_19\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Canada 2019")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = france\_race\_19\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, france\_race\_19\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in france\_race\_19\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("France 2019")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = austria\_race\_19\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, austria\_race\_19\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in austria\_race\_19\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Austria 2019")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = great\_britain\_race\_19\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, great\_britain\_race\_19\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in great\_britain\_race\_19\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Great Britain 2019")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = germany\_race\_19\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, germany\_race\_19\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in germany\_race\_19\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Germany 2019")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = hungary\_race\_19\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, hungary\_race\_19\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in hungary\_race\_19\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Hungary 2019")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = belgium\_race\_19\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, belgium\_race\_19\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in belgium\_race\_19\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Belgium 2019")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = italy\_race\_19\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, italy\_race\_19\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in italy\_race\_19\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Italy 2019")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = singapore\_race\_19\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, singapore\_race\_19\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in singapore\_race\_19\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Singapore 2019")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = russia\_race\_19\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, russia\_race\_19\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in russia\_race\_19\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Russia 2019")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = japan\_race\_19\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, japan\_race\_19\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in japan\_race\_19\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Japan 2019")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = united\_states\_race\_19\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, united\_states\_race\_19\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in united\_states\_race\_19\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("United States 2019")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = mexico\_race\_19\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, mexico\_race\_19\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in mexico\_race\_19\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Mexico 2019")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = brazil\_race\_19\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, brazil\_race\_19\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in brazil\_race\_19\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Brazil 2019")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = abu\_dhabi\_race\_19\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, abu\_dhabi\_race\_19\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in abu\_dhabi\_race\_19\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Abu Dhabi 2019")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

#%% visualizing 2020 tyre

driver\_stints = austria\_race\_20a\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, austria\_race\_20a\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in austria\_race\_20a\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Austria #1 2020")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = austria\_race\_20b\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, austria\_race\_20b\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in austria\_race\_20b\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Austria #2 2020")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = hungary\_race\_20\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, hungary\_race\_20\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in hungary\_race\_20\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Hungary 2020")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = great\_britain\_race\_20a\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, great\_britain\_race\_20a\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in great\_britain\_race\_20a\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Great Britain #1 2020")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = great\_britain\_race\_20b\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, great\_britain\_race\_20b\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in great\_britain\_race\_20b\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Great Britain #2 2020")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = spain\_race\_20\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, spain\_race\_20\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in spain\_race\_20\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Spain 2020")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = belgium\_race\_20\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, belgium\_race\_20\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in belgium\_race\_20\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Belgium 2020")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = italy\_race\_20a\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, italy\_race\_20a\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in italy\_race\_20a\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Italy #1 2020")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = italy\_race\_20b\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, italy\_race\_20b\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in italy\_race\_20b\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Italy #2 2020")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = italy\_race\_20c\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, italy\_race\_20c\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in italy\_race\_20c\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Italy #3 2020")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = russia\_race\_20\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, russia\_race\_20\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in russia\_race\_20\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Russia 2020")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = germany\_race\_20\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, germany\_race\_20\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in germany\_race\_20\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Germany 2020")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = portugal\_race\_20\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, portugal\_race\_20\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in portugal\_race\_20\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Portugal 2020")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = abu\_dhabi\_race\_20\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, abu\_dhabi\_race\_20\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in abu\_dhabi\_race\_20\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Abu Dhabi 2020")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = bahrain\_race\_20a\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, bahrain\_race\_20a\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in bahrain\_race\_20a\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Bahrain #1 2020")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = bahrain\_race\_20b\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, bahrain\_race\_20b\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in bahrain\_race\_20b\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Bahrain #2 2020")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

#%% visualizing 2021 tyre

driver\_stints = abu\_dhabi\_race\_21\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, abu\_dhabi\_race\_21\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in abu\_dhabi\_race\_21\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Abu Dhabi 2021")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = austria\_race\_21a\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, austria\_race\_21a\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in austria\_race\_21a\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Austria #1 2021")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = austria\_race\_21b\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, austria\_race\_21b\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in austria\_race\_21b\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Austria #2 2021")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = azerbaijan\_race\_21\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, azerbaijan\_race\_21\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in azerbaijan\_race\_21\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Azerbaijan 2021")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = bahrain\_race\_21\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, bahrain\_race\_21\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in bahrain\_race\_21\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Bahrain 2021")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = belgium\_race\_21\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, belgium\_race\_21\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in belgium\_race\_21\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Belgium 2021")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = brazil\_race\_21\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, brazil\_race\_21\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in brazil\_race\_21\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Brazil 2021")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = france\_race\_21\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, france\_race\_21\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in france\_race\_21\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("France 2021")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = great\_britain\_race\_21\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, great\_britain\_race\_21\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in great\_britain\_race\_21\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Great Britain 2021")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = hungary\_race\_21\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, hungary\_race\_21\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in hungary\_race\_21\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Hungary 2021")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = italy\_race\_21a\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, italy\_race\_21a\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in italy\_race\_21a\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Italy #1 2021")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = italy\_race\_21b\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, italy\_race\_21b\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in italy\_race\_21b\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Italy #2 2021")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = mexico\_race\_21\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, mexico\_race\_21\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in mexico\_race\_21\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Mexico 2021")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = monaco\_race\_21\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, monaco\_race\_21\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in monaco\_race\_21\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Monaco 2021")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = netherlands\_race\_21\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, netherlands\_race\_21\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in netherlands\_race\_21\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Netherlands 2021")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = portugal\_race\_21\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, portugal\_race\_21\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in portugal\_race\_21\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Portugal 2021")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = qatar\_race\_21\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, qatar\_race\_21\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in qatar\_race\_21\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Qatar 2021")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = russia\_race\_21\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, russia\_race\_21\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in russia\_race\_21\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Russia 2021")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = saudi\_arabia\_race\_21\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, saudi\_arabia\_race\_21\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in saudi\_arabia\_race\_21\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Saudi Arabia 2021")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = spain\_race\_21\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, spain\_race\_21\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in spain\_race\_21\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Spain 2021")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = turkey\_race\_21\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, turkey\_race\_21\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in turkey\_race\_21\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Turkey 2021")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = united\_states\_race\_21\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, united\_states\_race\_21\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in united\_states\_race\_21\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("USA 2021")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

#%% visualizing 2022 tyre

driver\_stints = abu\_dhabi\_race\_22\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, abu\_dhabi\_race\_22\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in abu\_dhabi\_race\_22\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Abu Dhabi 2022")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = austria\_race\_22\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, austria\_race\_22\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in austria\_race\_22\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Austria 2022")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = azerbaijan\_race\_22\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, azerbaijan\_race\_22\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in azerbaijan\_race\_22\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Azerbaijan 2022")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = bahrain\_race\_22\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, bahrain\_race\_22\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in bahrain\_race\_22\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Bahrain 2022")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = belgium\_race\_22\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, belgium\_race\_22\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in belgium\_race\_22\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Belgium 2022")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = brazil\_race\_22\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, brazil\_race\_22\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in brazil\_race\_22\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Brazil 2022")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = france\_race\_22\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, france\_race\_22\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in france\_race\_22\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("France 2022")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = great\_britain\_race\_22\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, great\_britain\_race\_22\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in great\_britain\_race\_22\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Great Britain 2022")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = hungary\_race\_22\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, hungary\_race\_22\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in hungary\_race\_22\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Hungary 2022")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = italy\_race\_22a\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, italy\_race\_22a\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in italy\_race\_22a\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Italy #1 2022")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = italy\_race\_22b\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, italy\_race\_22b\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in italy\_race\_22b\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Italy #2 2022")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = mexico\_race\_22\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, mexico\_race\_22\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in mexico\_race\_22\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Mexico 2022")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = monaco\_race\_22\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, monaco\_race\_22\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in monaco\_race\_22\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Monaco 2022")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = netherlands\_race\_22\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, netherlands\_race\_22\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in netherlands\_race\_22\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Netherlands 2022")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = singapore\_race\_22\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, singapore\_race\_22\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in singapore\_race\_22\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Singapore 2022")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = japan\_race\_22\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, japan\_race\_22\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in japan\_race\_22\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Japan 2022")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = canada\_race\_22\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, canada\_race\_22\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in canada\_race\_22\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Canada 2022")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = saudi\_arabia\_race\_22\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, saudi\_arabia\_race\_22\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in saudi\_arabia\_race\_22\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Saudi Arabia 2022")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = spain\_race\_22\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, spain\_race\_22\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in spain\_race\_22\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Spain 2022")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = australia\_race\_22\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, australia\_race\_22\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in australia\_race\_22\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("Australia 2022")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = united\_states\_race\_22a\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, united\_states\_race\_22a\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in united\_states\_race\_22a\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("USA #1 2022")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

driver\_stints = united\_states\_race\_22b\_laps[['Team', 'Driver', 'Stint', 'Compound', 'LapNumber']].groupby(

['Team', 'Driver', 'Stint', 'Compound']

).count().reset\_index()

test = [driver\_stints, united\_states\_race\_22b\_results]

test = pd.concat(test)

driver\_stints = test.rename(columns={'LapNumber': 'StintLength'})

driver\_stints = driver\_stints.sort\_values(by=['Position'])

plt.rcParams["figure.figsize"] = [15, 10]

plt.rcParams["figure.autolayout"] = True

fig, ax = plt.subplots()

for driver in united\_states\_race\_22b\_results['Abbreviation']:

stints = driver\_stints.loc[driver\_stints['Driver'] == driver]

previous\_stint\_end = 0

for \_, stint in stints.iterrows():

plt.barh(

[driver],

stint['StintLength'],

left=previous\_stint\_end,

#color=compound\_colors[stint['Compound']],

edgecolor = "black"

)

previous\_stint\_end = previous\_stint\_end + stint['StintLength']

# Set title

plt.title("USA #2 2022")

# Set x-label

plt.xlabel('Lap')

# Invert y-axis

plt.gca().invert\_yaxis()

# Remove frame from plot

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.spines['left'].set\_visible(False)

plt.show()

**Appendix C – R Code**

---

title: "thesis\_qPoisson"

author: "Andrew Estes"

date: "2023-04-30"

output: pdf\_document

---

```{r, warning=FALSE, message=FALSE, echo=FALSE}

library(tidyverse)

library(readxl)

library(lubridate)

library(pscl)

library(lmtest)

library(geepack)

library(flexmix)

library(lme4)

library(bpr)

library(bayesplot)

library(tidybayes)

library(broom.mixed)

library(rstanarm)

library(bayesrules)

#importing data

original.df <- read.csv("final\_pit\_only.csv")

df <- original.df

head(df)

```

```{r}

cols <- c("DriverNumberCat", "CompoundCat", "FreshTyre", "Stint", "TeamCat", "Year", "DriverCat", "TrackStatusCat", "GridPosition", "Position", "RaceCat", "FreshTyreCat")

df[cols] <- lapply(df[cols], factor) ## as.factor() could also be used

head(df)

```

```{r}

#https://cran.r-project.org/web/packages/pscl/vignettes/countreg.pdf

library(sandwich)

pois <- glm(LapNumber ~ ., data = df, family = poisson)

summary(pois)

coeftest(pois, vcov = sandwich) #inference function wald test; sandwich covariance matrix estimator

qpois <- glm(LapNumber ~ ., data = df, family = quasipoisson)

summary(qpois)

nbin <- MASS::glm.nb(LapNumber ~ ., data = df)

summary(nbin)

#cannot do hurdle or zip to to non-zero minimum count

```

```{r}

m.ex <- geeglm(LapNumber ~ .,

data = df,

id = interaction(cols, ),

family = poisson,

corstr = "exchangeable")

flex <- flexmix(LapNumber ~ .,

data = df,

k = 19,

model = FLXMRglm(family = "poisson"))

```

```{r}

fit\_bpr <- sample\_bpr(LapNumber ~ ., data = df, iter=100000)

#Running MH sampler with a gaussian prior distribution.

#Chains initialized at the maximum likelihood estimates.

#Error: copy into submatrix: incompatible matrix dimensions: 236x1 and 165x1

#https://cran.r-project.org/web/packages/bpr/bpr.pdf

```

```{r}

ggplot(df, aes(x = LapNumber)) +

geom\_histogram(color = "white", breaks = seq(0, 100, by = 10))

# Simulate the Normal model

equality\_normal\_sim <- stan\_glm(LapNumber ~ .,

data = df,

family = gaussian,

prior\_intercept = normal(7, 1.5),

prior = normal(0, 2.5, autoscale = TRUE),

prior\_aux = exponential(1, autoscale = TRUE),

chains = 4, iter = 2 \* 1000, seed = 84735)

# Posterior predictive check

pp\_check(equality\_normal\_sim, plotfun = "hist", nreps = 5) +

geom\_vline(xintercept = 0) +

xlab("laws")

pp\_check

equality\_model\_prior <- stan\_glm(LapNumber ~ .,

data = df,

family = poisson,

prior\_intercept = normal(2, 0.5),

prior = normal(0, 2.5, autoscale = TRUE),

chains = 4, iter = 2 \* 1000, seed = 84735,

prior\_PD = TRUE)

#https://www.bayesrulesbook.com/chapter-12.html

prior\_summary(equality\_model\_prior)

#df %>%

## add\_fitted\_draws(equality\_model\_prior, n = 100) %>%

# ggplot(aes(x = TyreLife, y = LapNumber, color = TeamCat)) +

# geom\_line(aes(y = .value, group = paste(historical, .draw))) +

# ylim(0, 100)

equality\_model <- update(equality\_model\_prior, prior\_PD = FALSE)

#checking predictive power

mcmc\_trace(equality\_model)

mcmc\_dens\_overlay(equality\_model)

mcmc\_acf(equality\_model)

set.seed(1)

pp\_check(equality\_model, plotfun = "hist", nreps = 5) +

xlab("laws")

pp\_check(equality\_model) +

xlab("laws")

#df %>%

# add\_fitted\_draws(equality\_model, n = 50) %>%

# ggplot(aes(x = TyreLife, y = LapNumber, color = Team)) +

# geom\_line(aes(y = .value, group = paste(historical, .draw)),

# alpha = .1) +

# geom\_point(data = df, size = 0.1)

tidy(equality\_model, conf.int = TRUE, conf.level = 0.80)

# Simulate posterior predictive models for each state

set.seed(84735)

poisson\_predictions <- posterior\_predict(equality\_model, newdata = df)

# Plot the posterior predictive models for each state

ppc\_intervals\_grouped(df$LapNumber, yrep = poisson\_predictions,

x = df$TyreLife,

group = df$RACE,

prob = 0.5, prob\_outer = 0.95,

facet\_args = list(scales = "fixed"))

prediction\_summary(equality\_model, df)

set.seed(84735)

poisson\_cv <- prediction\_summary\_cv(model = equality.\_model,

data = df, k = 10)

poisson\_cv$cv

```

```{r}

#https://www.bayesrulesbook.com/chapter-12.html

df %>%

group\_by(cut(TyreLife,3), FreshTyre) %>%

summarize(mean = mean(LapNumber), var = var(LapNumber))

books\_negbin\_sim <- stan\_glm(

LapNumber ~ .,

data = df, family = neg\_binomial\_2,

prior\_intercept = normal(0, 2.5, autoscale = TRUE),

prior = normal(0, 2.5, autoscale = TRUE),

prior\_aux = exponential(1, autoscale = TRUE),

chains = 4, iter = 2 \* 1000, seed = 84735)

# Check out the priors

prior\_summary(books\_negbin\_sim)

pp\_check(books\_negbin\_sim) +

xlim(0, 75) +

xlab("books")

tidy(books\_negbin\_sim, conf.int = TRUE, conf.level = 0.80)

```

**Appendix D – R Shiny Visualization Code**

#

# This is a Shiny web application. You can run the application by clicking

# the 'Run App' button above.

#

# Find out more about building applications with Shiny here:

#

#    http://shiny.rstudio.com/

#

setwd("C:\\Users\\andre\\OneDrive\\Desktop")

#loading the libraries and setting up the dataframes and variables

#library(shiny)

library(plotly)

library(readxl)

library(tidyverse)

library(magrittr)

library(jpeg)

library(officer)

#setwd("C:\\Users\\andre\\OneDrive\\Desktop\\thesis\_test")

z.orig.laps <- read.csv("laps.csv")

laps  <- z.orig.laps

z.orig.weather <- read.csv("weather.csv")

weather  <- z.orig.weather

z.orig.messages <- read.csv("messages.csv")

messages  <- z.orig.messages

z.orig.results <- read.csv("results.csv")

results  <- z.orig.results

#documentation <- read\_docx("documentation.docx")

#word1 <- a(href="documentation.docx")

linebreaks <- function(n){HTML(strrep(br(), n))}

courses <- c(

  "Australia","Austria", "Bahrain","China", "Azerbaijan", "Spain", "Monaco", "Canada",

  "France","Austria","GreatBritain","Germany","Hungary","Belgium","Italy",

  "Singapore","Russia","Japan","UnitedStates","Mexico","Brazil","UnitedArabEmirates",

  "AbuDhabi","Portugal","Turkey","Netherlands","Qatar","SaudiArabia")

courses <- lapply(courses, sort)

#manipulating the message/flag data

flags <- messages %>%

  subset(Flag == "YELLOW" |

           Flag == "DOUBLE YELLOW" |

           Flag == "RED")

#manipulating the pitstop data

pitInOnly <- laps[complete.cases(laps$PitInTime),]

pitInVariables <- as.list(colnames(pitInOnly))

pit\_by <- pitInOnly %>%

  group\_by(RACE, Year, Team, Driver) %>%

  summarize()

#maniuplating the results data

points\_by <- results %>%

  group\_by(RACE, Year, TeamName, TeamColor, FullName, Points) %>%

  summarize() %>%

  mutate(Driver = sub(".\* ", "", FullName)) %>%

  mutate(Driver = substr(Driver, 1, 3)) %>%

  mutate(Team = TeamName)

#manipulating the weather data

weatherAir\_scale <- scale(weather$AirTemp)

weatherTrack\_scale <- scale(weather$TrackTemp)

weatherHumidity\_scale <- scale(weather$Humidity)

weatherPressure\_scale <- scale(weather$Pressure)

weatherWind\_scale <- scale(weather$WindSpeed)

weather\_scale <- weather %>%

  mutate(AirScale = weatherAir\_scale) %>%

  mutate(TrackScale = weatherTrack\_scale) %>%

  mutate(HumidityScale = weatherHumidity\_scale) %>%

  mutate(PressureScale = weatherPressure\_scale) %>%

  mutate(WindScale = weatherWind\_scale)

by\_race <- weather\_scale %>%

  group\_by(RACE, Year) %>%

  summarise(

    meanAir = round(mean(AirTemp), 1),

    minAir = min(AirTemp),

    maxAir = max(AirTemp),

    rangeAir = maxAir - minAir,

    meanTrack = round(mean(TrackTemp), 1),

    minTrack = min(TrackTemp),

    maxTrack = max(TrackTemp),

    rangeTrack = maxTrack - minTrack,

    meanHumidity = round(mean(Humidity), 1),

    minHumidity = min(Humidity),

    maxHumidity = max(Humidity),

    rangeHumidity = maxHumidity - minHumidity,

    meanPressure = round(mean(Pressure), 1),

    minPressure = min(Pressure),

    maxPressure = max(Pressure),

    rangePressure = maxPressure - minPressure,

    meanWind = round(mean(WindSpeed), 1),

    minWind = min(WindSpeed),

    maxWind = max(WindSpeed),

    rangeWind = maxWind - minWind

  )

weather\_table <- by\_race %>%

  select(Year, meanAir, meanTrack, meanHumidity, meanWind, meanPressure)

by\_race\_range <- data.frame(

  AirTemperatureMean = round(range(by\_race$meanAir), 2),

  AirTemperatureRange = round(range(by\_race$rangeAir), 2),

  TrackTemperatureMean = round(range(by\_race$meanTrack), 2),

  TrackTemperatureRange = round(range(by\_race$rangeTrack), 2),

  HumidityMean = round(range(by\_race$meanHumidity), 2),

  HumidityRange = round(range(by\_race$rangeHumidity), 2),

  AirPressureMean = round(range(by\_race$meanPressure), 2),

  AirPressureRange = round(range(by\_race$rangePressure), 2),

  WindSpeedMean = round(range(by\_race$meanWind), 2),

  WindSpeedRange = round(range(by\_race$rangeWind), 2)

)

totalweather <- weather %>%

  mutate(RACE = "RACE") %>%

  mutate(Year = "YEAR") %>%

  mutate(meanAir = round(mean(AirTemp)), 2) %>%

  mutate(meanTemp = round(mean(TrackTemp)), 2) %>%

  mutate(meanHumidity = round(mean(Humidity)), 2) %>%

  mutate(meanWind = round(mean(WindSpeed)), 2) %>%

  mutate(meanPressure = round(mean(Pressure)), 2) %>%

select(RACE, Year, meanAir, meanTemp, meanHumidity, meanWind, meanPressure)

# Define UI for application

ui <- fluidPage(

  sidebarLayout(

    sidebarPanel(

                # radioButtons("Histograms", "Histogram types",

                #              c("Courses" = "courses",

                #                "LapNumber" = "LapNumber",

                #                "Flag" = "Flag")),

                  selectInput(inputId = "courses",

                              label = h3("Select a Race"),

                              choices = courses,

                              multiple = FALSE,

                              selected = "Australia"),

                  sliderInput("Year", "Year:",

                              min = 2018, max = 2022,

                              value = c(2019, 2021),

                              step = 1),

                  # selectInput(inputId =  "var1",

                  #             label = h3("Histogram of Which Lap Number Pit Stops Occur"),

                  #             choices = "LapNumber",

                  #             selected = "LapNumber"),

                  #

                  # selectInput(inputId =  "var2",

                  #             label = h3("Histogram of Flag Frequency"),

                  #             choices = "Flag",

                  #             selected = "Flag"),

                  selectInput(inputId =  "var4",

                              label = h3("Point Accumulation by Team or Driver"),

                              choices = c("Team", "Driver"),

                              selected = "Team\*"),

                  selectInput(inputId =  "var3",

                            label = h3("Pit Stops Frequency by Team or Driver"),

                            choices = c("Team", "Driver"),

                            selected = "Team"),

                  selectInput("dataset",

                              label = h3("Dataset Preview"),

                              choices = c("results", "messages", "weather", "laps")),

                  linebreaks(0),

                  downloadButton("downloadData", "Download Dataset",

                                 style="color: #fff; background-color: #337ab7; border-color: #2e6da4"),

                  # linebreaks(3),

                  #

                  # selectInput("documentation",

                  #             label = h3("Documentation"),

                  #             choices = c("documentation")),

                  #

                  # downloadButton('downloadDocumentation', 'Download Documentation',

                  #                style="color: #fff; background-color: #337ab7; border-color: #2e6da4"),

                  linebreaks(10),

                  word1,

                  a("Download Documentation", href="documentation.docx"),

                  linebreaks(2),

                  fileInput(inputId = "file1",

                            label = "Attach Feedback")

    ),

    mainPanel(

            tabsetPanel(type = "tabs",

                 tabPanel("Course Visualization", fluidRow(

                        column(width=12,

                               imageOutput("img1")))),

                 tabPanel("Lap # When Pit Stop",

                          fluidRow(

                            #column(10, plotlyOutput("CoursesPlot")),

                            column(10, plotlyOutput("TotalPlot"))

                            )),

                 tabPanel("Tyre Life When Pit Stop",

                          fluidRow(

                            column(10, plotlyOutput("TyrePlot")),

                            column(10, plotlyOutput("CoursesPlot"))

                          )),

                 tabPanel("Weather",

                          fluidRow(

                            dataTableOutput("weather"),

                            dataTableOutput("totalweather")

                          )),

                 tabPanel("Flag", plotlyOutput("FlagPlot")),

                 tabPanel("Points", plotlyOutput("PointPlot")),

                 tabPanel("Pit", plotlyOutput("PitPlot")),

                tabPanel("Preview Datasets", tableOutput("dataset")),

                 #tabPanel("Documentation", textOutput("documentation"))

        )

      )

    )

)

# Define server logic

server <- function(input, output) {

#https://stackoverflow.com/questions/40861908/shiny-r-implement-slider-input

  output$CoursesPlot <- renderPlotly({

    p <- pitInOnly %>%

      subset(LapNumber!=1) %>%

      mutate(

        courses = input$courses) %>%

      filter(

        RACE == courses) %>%

      filter(

        Year >= input$Year[1] & Year <= input$Year[2]

      ) %>%

      ggplot(aes(x = LapNumber,

                 fill = as.factor(Year))) +

        geom\_histogram() +

      xlim(0, 60) +

      ggtitle((req(input$courses)))

      ggplotly(p)

  })

  output$TyrePlot <- renderPlotly({

    p1 <- pitInOnly %>%

      subset(LapNumber!=1) %>%

      mutate(

        courses = input$courses) %>%

      filter(

        RACE == courses) %>%

      filter(

        Year >= input$Year[1] & Year <= input$Year[2]

      ) %>%

      ggplot(aes(x = TyreLife,

                 fill = as.factor(Year))) +

      geom\_histogram() +

      xlim(0, 60) +

      ggtitle((req(input$courses)))

    ggplotly(p1)

  })

  output$FlagPlot <- renderPlotly({

    p2 <- flags %>%

      mutate(

        courses = input$courses) %>%

      filter(

        RACE == courses) %>%

      filter(

        Year >= input$Year[1] & Year <= input$Year[2]

      ) %>%

      ggplot(aes(

                 x = Flag,

                 fill = Flag)) +

      geom\_histogram(stat="count",

                     position = "dodge",

                     bins = 1) +

      scale\_fill\_manual(values = c("DOUBLE YELLOW" = "black",

                                   "RED" = "red",

                                   "YELLOW" = "yellow")) +

      scale\_color\_manual(values = c("DOUBLE YELLOW" = "yellow",

                                    "RED" = "red",

                                    "YELLOW" = "yellow")) +

      ggtitle("Quantification of Hazardous Flags by Course and Year")

    ggplotly(p2)

  })

  output$PitPlot <- renderPlotly({

    p3 <- pit\_by %>%

      mutate(

        courses = input$courses) %>%

      filter(

        RACE == courses) %>%

      filter(

        Year >= input$Year[1] & Year <= input$Year[2]

      ) %>%

      ggplot(aes(y = .data[[input$var3]],

                 fill= as.factor(Year))) +

      geom\_bar() +

      ggtitle((req(input$courses)))

    ggplotly(p3)

  })

  output$PointPlot <- renderPlotly({

    p4 <- points\_by %>%

      mutate(

        courses = input$courses) %>%

      filter(

        RACE == courses) %>%

      filter(

        Year >= input$Year[1] & Year <= input$Year[2]

      ) %>%

      ggplot(aes(x = Points,

                 y = .data[[input$var4]],

                 fill=TeamName)) +

      geom\_col() +

      ggtitle("Points Accumulated by Team or Driver By Course and Year")

    ggplotly(p4)

  })

  output$weather <- renderDataTable({by\_race %>%

      mutate(

        courses = input$courses) %>%

      filter(

        RACE == courses) %>%

      select(Year, meanAir, meanTrack, meanHumidity, meanWind, meanPressure)

  })

  output$totalweather <- renderDataTable({

    head(totalweather, 1)

  })

  re1 <- reactive({

    input$file1

    })

  output$Invoice <- renderImage({

    re1()

    })

#  output$img1 <- renderImage({

#    list(src="C:\\Users\\andre\\OneDrive\\Desktop\\F1\_Tracks\\Total\_noDuplicate\\France.png")

#  })

  output$img1 <- renderImage({

    if(input$courses == "Abu Dhabi"){

      list(src="AbuDhabi.png", height = "500", width="800")

    } else if(input$courses == "Australia"){

      list(src="Australian.png", height = "500", width="800")

    } else if(input$courses == "Austria"){

      list(src="Austria.png",height = "500", width="800")

    } else if(input$courses == "Azerbaijan"){

      list(src="Azerbaijan.png",height = "500", width="800")

    } else if(input$courses == "Bahrain"){

      list(src="Bahrain.png",height = "500", width="800")

    } else if(input$courses == "Belgium"){

      list(src="Belgium.png",height = "500", width="800")

    } else if(input$courses == "Brazil"){

      list(src="Brazil",height = "500", width="800")

    } else if(input$courses == "Canada"){

      list(src="Canada",height = "500", width="800")

    } else if(input$courses == "China"){

      list(src="China.png",height = "500", width="800")

    } else if(input$courses == "France"){

      list(src="France.png",height = "500", width="800")

    } else if(input$courses == "Germany"){

      list(src="Germany.png",height = "500", width="800")

    } else if(input$courses == "Great Britain"){

      list(src="GreatBritain.png",height = "500", width="800")

    } else if(input$courses == "Hungary"){

      list(src="Hungary.png",height = "500", width="800")

    } else if(input$courses == "Italy"){

      list(src="Italy.png",height = "500", width="800")

    } else if(input$courses == "Japan"){

      list(src="Japan.png",height = "500", width="800")

    } else if(input$courses == "Mexico"){

      list(src="Mexico.png",height = "500", width="800")

    } else if(input$courses == "Monaco"){

      list(src="Monaco.png",height = "500", width="800")

    } else if(input$courses == "Netherlands"){

      list(src="Netherlands.png",height = "500", width="800")

    } else if(input$courses == "Portugal"){

      list(src="Portugal.png",height = "500", width="800")

    } else if(input$courses == "Qatar"){

      list(src="Qatar.png",height = "500", width="800")

    } else if(input$courses == "Russia"){

      list(src="Russia.png",height = "500", width="800")

    } else if(input$courses == "SaudiArabia"){

      list(src="SaudiArabia.png",height = "500", width="800")

    } else if(input$courses == "Singapore"){

      list(src="Singapore.png",height = "500", width="800")

    } else if(input$courses == "Spain"){

      list(src="Spain.png",height = "500", width="800")

    } else if(input$courses == "Turkey"){

      list(src="Turkey.png",height = "500", width="800")

    } else if(input$courses == "United Arab Emirates"){

      list(src="UnitedArabEmirates.png",height = "500", width="800")

    } else {

      list(src="UnitedStates.png", height = "500", width="800")

    }

  })

  #https://shiny.rstudio.com/gallery/file-download.html

  datasetInput <- reactive({

    switch(input$dataset,

           "results" = results,

           "laps" = lap,

           "weather" = weather,

           "messages" = messages)

  })

  output$dataset <- renderTable({

    head(datasetInput())

  })

  output$downloadData <- downloadHandler(

    filename = function() {

      paste(input$dataset, ".csv", sep = "")

    },

    content = function(file) {

      write.csv(datasetInput(), file, row.names = FALSE)

    }

  )

  output$TotalPlot <- renderPlotly({

    p5 <- pitInOnly %>%

      subset(LapNumber!=1) %>%

      # filter(

      #   Year >= input$Year[1] & Year <= input$Year[2]

      # ) %>%

      ggplot(aes(x = LapNumber

      )) +

      geom\_histogram(

        binwidth = 1

      ) +

      ggtitle("Lap Number When Driver Came in for PitStop Across All Courses")

    ggplotly(p5)

  })

  output$TotalTyrePlot <- renderPlotly({

    p6 <- pitInOnly %>%

      subset(LapNumber!=1) %>%

      filter(

        Year >= input$Year[1] & Year <= input$Year[2]

      ) %>%

      ggplot(aes(x = TyreLife)) +

      geom\_histogram(

        binwidth = 1

      ) +

      ggtitle("Lap Number When Driver Came in for PitStop Across All Courses")

    ggplotly(p6)

  })

#"C:\\Users\\andre\\OneDrive\\Desktop\\thesis\_test\\documentation.docx"

#https://www.rdocumentation.org/packages/ReporteRs/versions/0.8.10/topics/writeDoc

#https://stackoverflow.com/questions/59023362/download-existing-docx-object-from-r-shiny-app

  # output$downloadDocumentation <- downloadHandler(

  #   filename = function() {

  #     paste(documentation, ".docx", sep = "")

  #   },

  #   content = function(file) {

  #     writeDoc(documentation, file)

  #       print(target = file)

  #   }

  # )

  #

  #

  # output$Documentation <- renderText({

  #   'test'

  # })

  #

}

# Run the application

shinyApp(ui = ui, server = server)

**Appendix E – Variable Description**

For the “Laps” dataframe

**Time** (pandas.Timedelta): Session time when the lap time was set (end of lap) **DriverNumber** (str): Driver number

**LapTime** (pandas.Timedelta): Recorded lap time

**LapNumber** (int): Recorded lap 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. This can happen if the track limits were exceeded, for example.

**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

**Stint** (int): Stint number

**LapStartTime** (pandas.Timedelta): Session time at the start of the lap

**Team** (str): Team name

**Driver** (string): Three letter driver identifier

**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()*](https://theoehrly.github.io/Fast-F1/api.html#fastf1.api.track_status_data). For filtering laps by track status, you may want to use [*Laps.pick\_track\_status()*](https://theoehrly.github.io/Fast-F1/core.html#fastf1.core.Laps.pick_track_status)

**IsAccurate** (bool): Indicates that the lap start and end time are synced correctly with other laps. Do not confuse this with the accuracy of the lap time or sector times. They are always considered to be accurate if they exist! If this value is True, the lap has passed as basic accuracy check for timing data. This does not guarantee accuracy but laps marked as inaccurate need to be handled with caution. They might contain errors which can not be spotted easily. Laps need to satisfy the following criteria to be marked as accurate:

o   not an inlap or outlap

o   set under green or yellow flag (the api sometimes has issues with data from SC/VSC laps)

o   is not the first lap after a safety car period (issues with SC/VSC might still appear on the first lap after it has ended)

o   has a value for lap time and all sector times

o   the sum of the sector times matches the lap time (If this were to ever occur, it would also be logged separately as a data integrity error. You usually don’t need to worry about this.)

**LapStartDate** (pandas.Timestamp): Timestamp at the start of the lap

For the “race\_control\_messages” dataframe

* Utc: Message timestamp
* Category (str): Type of message, “Other”, “Flag”, “Drs”, “CarEvent”
* Message (str): Content of message
* Status (str): Status of context, e.g. “DISABLED” for disabling DRS
* Flag (str): Type of flag being waved “GREEN”, “RED”, “YELLOW”, “CLEAR”, “CHEQUERED”
* Scope (str): Scope of message “Track”, “Sector”, “Driver”
* Sector (int): Affected track sector for sector-scoped **messages**
* RacingNumber (str): Affected driver for CarEvent **messages**

For the “results” dataframe

* **DriverNumber** | **str** | The number associated with this driver in this session (usually the drivers permanent number)
* **BroadcastName** | **str** | First letter of the drivers first name plus the drivers full last name in all capital letters. (e.g. ‘P GASLY’)
* **FullName** | **str** | The drivers full name (e.g. “Pierre Gasly”)
* **Abbreviation** | **str** | The drivers three letter abbreviation (e.g. “GAS”)
* **TeamName** | **str** | The team name (short version without title sponsors)
* **TeamColor** | **str** | The color commonly associated with this team (hex value)
* **FirstName** | **str** | The drivers first name
* **LastName** | **str** | The drivers last name
* **Position** | **float** | The drivers finishing position (values only given if session is ‘Race’, ‘Qualifying’ or ‘Sprint Qualifying’)
* **GridPosition** | **float** | The drivers starting position (values only given if session is ‘Race’ or ‘Sprint Qualifying’)
* **Q1** | **pd.Timedelta** | The drivers best Q1 time (values only given if session is ‘Qualifying’)
* **Q2** | **pd.Timedelta** | The drivers best Q2 time (values only given if session is ‘Qualifying’)
* **Q3** | **pd.Timedelta** | The drivers best Q3 time (values only given if session is ‘Qualifying’)
* **Time** | **pd.Timedelta** | The drivers total race time (values only given if session is ‘Race’ or ‘Sprint Qualifying’ and the driver was not more than one lap behind the leader)
* **Status** | **str** | A status message to indicate if and how the driver finished the race or to indicate the cause of a DNF. Possible values include but are not limited to ‘Finished’, ‘+ 1 Lap’, ‘Crash’, ‘Gearbox’, … (values only given if session is ‘Race’ or ‘Sprint Qualifying’)
* **Points**| **float** | The number of points received by each driver for their finishing **result**.

For the “weather” dataframe

* Time (datetime.timedelta): session timestamp (time only)
* AirTemp (float): Air temperature [°C]
* Humidity (float): Relative humidity [%]
* Pressure (float): Air pressure [mbar]
* Rainfall (bool): Shows if there is rainfall
* TrackTemp (float): Track temperature [°C]
* WindDirection (int): Wind direction [°] (0°-359°)
* WindSpeed (float): Wind speed [km/h]

**Appendix F – R Visualizations**

---

title: "thesis\_visualizations"

author: "Andrew Estes"

date: '2022-11-06'

output: pdf\_document

---

```{r, echo=FALSE}

#normalize pit stop based off percentages by year (different # races/yr)

#why are pit stops so frequent on lap 1/2

#pit stop times by driver and team

#scatterplot tyre life vs lap #

#factor messages$message and histogram type

#factor messages$flag

#factor messages$category

#break into year and by course

#do weather shiny app and data by course/year

```

```{r, echo=FALSE, warning=FALSE, error=FALSE,message=FALSE}

library(readxl)

library(tidyverse)

library(dplyr)

library(lubridate)

library(colorspace)

library(patchwork)

library(forecast)

library(zoo)

library(xts)

library(janitor)

library(knitr)

library(TTR)

library(viridis)

library(randtests)

library(gridExtra)

library(plotly)

library(shiny)

library(knitr)

z.orig.laps <- read.csv("laps.csv")

laps  <- z.orig.laps

z.orig.weather <- read.csv("weather.csv")

weather  <- z.orig.weather

z.orig.messages <- read.csv("messages.csv")

messages  <- z.orig.messages

```

```{r, echo=FALSE}

pitInOnly <- laps[complete.cases(laps$PitInTime),]

ggplot(pitInOnly,

       aes(

         x = LapNumber

       )) +

  geom\_histogram(

    binwidth = 1

  ) +

  ggtitle("Pit Stop In, Cumulative, All Laps")

#graphing the pitstop entrances excluding the first lap

pitInExclude1 <- subset(pitInOnly, LapNumber!=1) %>%

  ggplot(aes(

         x = LapNumber

       )) +

  geom\_histogram(

    binwidth = 1

  ) +

  ggtitle("Pit Stop In, Cumulative, Excluding Lap 1")

pitInExclude1

#graphing the pitstop exits, excluding the first lap, only for 2018

pitInExclude1.2018 <- pitInOnly %>%

  subset(LapNumber!=1) %>%

  filter(Year == 2018) %>%

    ggplot(aes(

         x = LapNumber

       )) +

  geom\_histogram(

    binwidth = 1

  ) +

  ggtitle("Pit Stop In, 2018, Excluding Lap 1")

pitInExclude1.2018

#graphing the pitstop exits, excluding the first lap, only for 2019

pitInExclude1.2019 <- pitInOnly %>%

  subset(LapNumber!=1) %>%

  filter(Year == 2019) %>%

    ggplot(aes(

         x = LapNumber

       )) +

  geom\_histogram(

    binwidth = 1

  ) +

  ggtitle("Pit Stop In, 2019, Excluding Lap 1")

pitInExclude1.2019

#graphing the pitstop exits, excluding the first lap, only for 2020

pitInExclude1.2020 <- pitInOnly %>%

  subset(LapNumber!=1) %>%

  filter(Year == 2020) %>%

    ggplot(aes(

         x = LapNumber

       )) +

  geom\_histogram(

    binwidth = 1

  ) +

  ggtitle("Pit Stop In, 2020, Excluding Lap 1")

pitInExclude1.2020

#graphing the pitstop exits, excluding the first lap, only for 2021

pitInExclude1.2021 <- pitInOnly %>%

  subset(LapNumber!=1) %>%

  filter(Year == 2021) %>%

    ggplot(aes(

         x = LapNumber

       )) +

  geom\_histogram(

    binwidth = 1

  ) +

  ggtitle("Pit Stop In, 2021, Excluding Lap 1")

pitInExclude1.2021

#graphing the pitstop exits, excluding the first lap, only for 2022

pitInExclude1.2022 <- pitInOnly %>%

  subset(LapNumber!=1) %>%

  filter(Year == 2022) %>%

    ggplot(aes(

         x = LapNumber

       )) +

  geom\_histogram(

    binwidth = 1

  ) +

  ggtitle("Pit Stop In, 2022, Excluding Lap 1")

pitInExclude1.2022

```

```{r, echo=FALSE}

pitOutOnly <- laps[complete.cases(laps$PitOutTime),]

ggplot(pitOutOnly,

       aes(

         x = LapNumber

       )) +

  geom\_histogram(

    binwidth = 1

  ) +

  ggtitle("Pit Stop Exit, Cumulative, All Laps")

#graphing the pitstop exits excluding the first lap

pitOutExclude1 <- subset(pitOutOnly, LapNumber!=1) %>%

  ggplot(aes(

         x = LapNumber

       )) +

  geom\_histogram(

    binwidth = 1

  ) +

  ggtitle("Pit Stop Exit, Cumulative, Excluding Lap 1")

pitOutExclude1

#graphing the pitstop exits, excluding the first lap, only for 2018

pitoutExclude1.2018 <- pitOutOnly %>%

  subset(LapNumber!=1) %>%

  filter(Year == 2018) %>%

    ggplot(aes(

         x = LapNumber

       )) +

  geom\_histogram(

    binwidth = 1

  ) +

  ggtitle("Pit Stop Exit, 2018, Excluding Lap 1")

pitoutExclude1.2018

#graphing the pitstop exits, excluding the first lap, only for 2019

pitoutExclude1.2019 <- pitOutOnly %>%

  subset(LapNumber!=1) %>%

  filter(Year == 2019) %>%

    ggplot(aes(

         x = LapNumber

       )) +

  geom\_histogram(

    binwidth = 1

  ) +

  ggtitle("Pit Stop Exit, 2019, Excluding Lap 1")

pitoutExclude1.2019

#graphing the pitstop exits, excluding the first lap, only for 2020

pitoutExclude1.2020 <- pitOutOnly %>%

  subset(LapNumber!=1) %>%

  filter(Year == 2020) %>%

    ggplot(aes(

         x = LapNumber

       )) +

  geom\_histogram(

    binwidth = 1

  ) +

  ggtitle("Pit Stop Exit, 2020, Excluding Lap 1")

pitoutExclude1.2020

#graphing the pitstop exits, excluding the first lap, only for 2021

pitoutExclude1.2021 <- pitOutOnly %>%

  subset(LapNumber!=1) %>%

  filter(Year == 2021) %>%

    ggplot(aes(

         x = LapNumber

       )) +

  geom\_histogram(

    binwidth = 1

  ) +

  ggtitle("Pit Stop Exit, 2021, Excluding Lap 1")

pitoutExclude1.2021

#graphing the pitstop exits, excluding the first lap, only for 2022

pitoutExclude1.2022 <- pitOutOnly %>%

  subset(LapNumber!=1) %>%

  filter(Year == 2022) %>%

    ggplot(aes(

         x = LapNumber

       )) +

  geom\_histogram(

    binwidth = 1

  ) +

  ggtitle("Pit Stop Exit, 2022, Excluding Lap 1")

pitoutExclude1.2022

```

```{r, echo=FALSE, eval=FALSE}

#

# This is a Shiny web application. You can run the application by clicking

# the 'Run App' button above.

#

# Find out more about building applications with Shiny here:

#

#    http://shiny.rstudio.com/

#

library(shiny)

library(plotly)

courses <- c(

  "Australia", "Bahrain","China", "Azerbaijan", "Spain", "Monaco", "Canada",

  "France","Austria","Great Britain","Germany","Hungary","Belgium","Italy",

  "Singapore","Russia","Japan","United States","Mexico","Brazil","United Arab Emirates",

  "Abu Dhabi","Portugal","Turkey","Netherlands","Qatar","Saudi Arabia")

pitInVariables <- as.list(colnames(pitInOnly))

# Define UI for application

ui <- fluidPage(

  titlePanel("Pit In Lap Number Histogram by Course"),

  sidebarLayout(

    sidebarPanel("Sidebar Panel",

                 selectInput( inputId = "courses",

                              label = h3("Select a Course"),

                              choices = courses,

                              selected = "Australia"),

                 selectInput(inputId =  "var1",

                             label = h3("Choose A Variable"),

                             choices = pitInVariables,

                             selected = "LapNumber"),

                 sliderInput("Year", "Year:",

                             min = 2018, max = 2022,

                             value = c(2019, 2021),

                             step = 1)

    ),

    mainPanel(

      h2("Histogram of Pit Stop By Courses"),

#      plotOutput("CoursesPlot2"),

      plotlyOutput("CoursesPlot"),

      dataTableOutput("data")

          )

  )

)

# Define server logic

server <- function(input, output) {

#https://stackoverflow.com/questions/40861908/shiny-r-implement-slider-input

#  data.react <- reactive({

#    if(!is.null(input$Year)){

#      data.react <- pitInOnly %>%

#        filter(Year >= input$year[1] & Year <= input$Year[2])

#    }

#    data.react

#  })\*

#  output$CoursesPlot2 <- renderPlot({

#    plot(data=data.react(), x=Year, y=LapNumber)

#  })

  output$CoursesPlot <- renderPlotly({

    p <- pitInOnly %>%

      mutate(

        courses = input$courses) %>%

      filter(

        RACE == courses) %>%

      filter(

        Year >= input$Year[1] & Year <= input$Year[2]

      ) %>%

      ggplot(aes(x = .data[[input$var1]])) +

        geom\_histogram()

      ggplotly(p)

  })

  output$data <- renderDataTable({pitInOnly %>%

      mutate(

        courses = input$courses) %>%

      filter(

        RACE == courses)

  })

}

# Run the application

shinyApp(ui = ui, server = server)

```

```{r, echo=FALSE, warning=FALSE, error=FALSE,message=FALSE}

#standardize/normalize the weather data

weatherAir\_scale <- scale(weather$AirTemp)

weatherTrack\_scale <- scale(weather$TrackTemp)

weatherHumidity\_scale <- scale(weather$Humidity)

weatherPressure\_scale <- scale(weather$Pressure)

weatherWind\_scale <- scale(weather$WindSpeed)

weather\_scale <- weather %>%

  mutate(AirScale = weatherAir\_scale) %>%

  mutate(TrackScale = weatherTrack\_scale) %>%

  mutate(HumidityScale = weatherHumidity\_scale) %>%

  mutate(PressureScale = weatherPressure\_scale) %>%

  mutate(WindScale = weatherWind\_scale)

#grouping by race and finding the averages and ranges per track/year

by\_race <- weather\_scale %>%

  group\_by(RACE, Year) %>%

  summarise(

    meanAir = mean(AirTemp),

    minAir = min(AirTemp),

    maxAir = max(AirTemp),

    rangeAir = maxAir - minAir,

    meanTrack = mean(TrackTemp),

    minTrack = min(TrackTemp),

    maxTrack = max(TrackTemp),

    rangeTrack = maxTrack - minTrack,

    meanHumidity = mean(Humidity),

    minHumidity = min(Humidity),

    maxHumidity = max(Humidity),

    rangeHumidity = maxHumidity - minHumidity,

    meanPressure = mean(Pressure),

    minPressure = min(Pressure),

    maxPressure = max(Pressure),

    rangePressure = maxPressure - minPressure,

    meanWind = mean(WindSpeed),

    minWind = min(WindSpeed),

    maxWind = max(WindSpeed),

    rangeWind = maxWind - minWind

  )

#showing the range of the mean all data across all races & years

#showing the range of the range for all data across all races & years

by\_race\_range <- data.frame(

  AirTemperatureMean = round(range(by\_race$meanAir), 2),

  AirTemperatureRange = round(range(by\_race$rangeAir), 2),

  TrackTemperatureMean = round(range(by\_race$meanTrack), 2),

  TrackTemperatureRange = round(range(by\_race$rangeTrack), 2),

  HumidityMean = round(range(by\_race$meanHumidity), 2),

  HumidityRange = round(range(by\_race$rangeHumidiy), 2),

  AirPressureMean = round(range(by\_race$meanPressure), 2),

  AirPressureRange = round(range(by\_race$rangePressure), 2),

  WindSpeedMean = round(range(by\_race$meanWind), 2),

  WindSpeedRange = round(range(by\_race$rangeWind), 2)

  )

kable(by\_race\_range)

#the range columns show the smallest and largest fluctuations in a race across all tracks and years

#the mean shows the smallest and largest averages across all tracks and years

```

```{r}

tyre.lm <- lm()

#PCA or CLUSTER for correlation/regression variable importance for laps pit in time

```

**Appendix G – Fastest Lap Course Visualizations**

A screen shot of a computer screen

Description automatically generated

A map of a race track

Description automatically generated

A race track with different colors

Description automatically generated

A map of a race track

Description automatically generated

A screen shot of a race track

Description automatically generated

A race track with a colorful line

Description automatically generated

A map of a race track

Description automatically generated

A race track with a map of the race track

Description automatically generatedA screen shot of a race track

Description automatically generatedA map of a race track

Description automatically generatedA race track with a black background

Description automatically generatedA computer screen shot of a race track

Description automatically generatedA race track with colorful ribbons

Description automatically generatedA race track with a black background

Description automatically generatedA race track with a black background

Description automatically generatedA race track with a black background

Description automatically generatedA computer screen shot of a race track

Description automatically generatedA race track with a black background

Description automatically generatedA graph of a race track

Description automatically generatedA graph of a race track

Description automatically generatedA map of a race track

Description automatically generatedA race track with a black background

Description automatically generatedA race track with a black background

Description automatically generatedA colorful rope in the shape of a bird

Description automatically generatedA race track with a black background

Description automatically generatedA computer screen shot of a line

Description automatically generatedA race track with a graph

Description automatically generatedA race track with many colors

Description automatically generatedA race track with a map of different colors

Description automatically generatedA map of a race track

Description automatically generatedA map of a race car

Description automatically generatedA colorful race track with text

Description automatically generated

**Appendix H – Initial Structure**

Structure for when a pit stop occurs

For 1-SB, determine SB% - competitor B makes change

* 1 - SB → Competitor B stays status quo
* SB → Competitor B reacts/changes plan based upon Driver A pit stop

Determine Pa - relative power, support from other parties

* 1 - Pa → probability B wins
* Pa → probability A wins

A diagram of a car

Description automatically generated

Structure for when a pit stop does not occur

Determine Q - probability that status quo will continue

* Q → Status Quo
* 1 - Q → Status Quo changes

For 1-Q, determine T - probability of beneficial change

* T → probability of improved place for A
* 1 - T → probability of worse place for A

A diagram of a driver

Description automatically generated

**Appendix I – Course Scheduling**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **2018** | **2019** | **2020** | **2021** | **2022** |
| Australia | Australia | Austria | Bahrain | Bahrain |
| Bahrain | Bahrain | Austria | Italy | Saudi Arabia |
| China | China | Hungary | Portugal | Australia |
| Azerbaijan | Azerbaijan | Great Britain | Spain | Italy |
| Spain | Spain | Great Britain | Monaco | United States |
| Monaco | Monaco | Spain | Azerbaijan | Spain |
| Canada | Canada | Belgium | France | Monaco |
| France | France | Italy | Austria | Azerbaijan |
| Austria | Austria | Italy | Austria | Canada |
| Great Britain | Great Britain | Russia | Great Britain | Great Britain |
| Germany | Germany | Germany | Hungary | Austria |
| Hungary | Hungary | Portugal | Belgium | France |
| Belgium | Belgium | Italy | Netherlands | Hungary |
| Italy | Italy | Turkey | Italy | Belgium |
| Singapore | Singapore | Bahrain | Russia | Netherlands |
| Russia | Russia | Bahrain | Turkey | Italy |
| Japan | Japan | Abu Dhabi | United States | Singapore |
| United States | Mexico |  | Mexico | Japan |
| Mexico | United States |  | Brazil | United States |
| Brazil | Brazil |  | Qatar | Mexico |
| United Arab Emirates | Abu Dhabi |  | Saudi Arabia | Brazil |
|  |  |  | Abu Dhabi | Abu Dhabi |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Course*** | ***2018*** | ***2019*** | ***2020*** | ***2021*** | ***2022*** | **Sub-Total** |
| *Australia* | 1 | 1 | 0 | 0 | 1 | **3** |
| *Bahrain* | 1 | 1 | 2 | 1 | 1 | **6** |
| *China* | 1 | 1 | 0 | 0 | 0 | **2** |
| *Azerbaijan* | 1 | 1 | 0 | 1 | 1 | **4** |
| *Spain* | 1 | 1 | 1 | 1 | 1 | **5** |
| *Monaco* | 1 | 1 | 0 | 1 | 1 | **4** |
| *Canada* | 1 | 1 | 0 | 0 | 1 | **3** |
| *France* | 1 | 1 | 0 | 1 | 1 | **4** |
| *Austria* | 1 | 1 | 2 | 2 | 1 | **7** |
| *Great Britain* | 1 | 1 | 2 | 1 | 1 | **6** |
| *Germany* | 1 | 1 | 1 | 0 | 0 | **3** |
| *Hungary* | 1 | 1 | 1 | 1 | 1 | **5** |
| *Belgium* | 1 | 1 | 1 | 1 | 1 | **5** |
| *Italy* | 1 | 1 | 3 | 2 | 2 | **9** |
| *Singapore* | 1 | 1 | 0 | 0 | 1 | **3** |
| *Russia* | 1 | 1 | 1 | 1 | 0 | **4** |
| *Japan* | 1 | 1 | 0 | 0 | 1 | **3** |
| *United States* | 1 | 1 | 0 | 1 | 2 | **5** |
| *Mexico* | 1 | 1 | 0 | 1 | 1 | **4** |
| *Brazil* | 1 | 1 | 0 | 1 | 1 | **4** |
| *United Arab Emirates* | 1 | 0 | 0 | 0 | 0 | **1** |
| *Abu Dhabi* | 0 | 1 | 1 | 1 | 1 | **4** |
| *Portugal* | 0 | 0 | 1 | 1 | 0 | **2** |
| *Turkey* | 0 | 0 | 1 | 1 | 0 | **2** |
| *Netherlands* | 0 | 0 | 0 | 1 | 1 | **2** |
| *Qatar* | 0 | 0 | 0 | 1 | 0 | **1** |
| *Saudi Arabia* | 0 | 0 | 0 | 1 | 1 | **2** |
| **Sub-Total** | **21** | **21** | **17** | **22** | **22** | **103** |