Bayesian analysis of the positions taken by F1 drivers in each race of the 2022-2024 hybrid era.

Andrzej Janik, Łukasz Gakan

Data Analytics 2024

Automatic Control and Robotics - Computer Science in Control and Management

1. Project formulation

The goal of the project is to conduct a Bayesian analysis of the places held by Formula One drivers in individual races in the hybrid era, spanning 2022-2024 by predicting the positions taken by drivers in races. The hybrid era, characterized by advanced hybrid powertrain technologies, has introduced new challenges and opportunities that significantly affect race results and team strategies. The analysis aims to identify patterns and factors affecting driver performance in this modern era, using Bayesian analysis tools. The project will collect and analyze data on the results of F1 races from 2022-2024, including drivers' starting and finishing positions, weather conditions, times and finishing positions from other sessions i.e qualifications.

1.1. Point of creating model and potential use cases

Bayesian analysis will make it possible to model the relationship between these variables, identifying the key factors affecting drivers' final positions and to predict future results based on current data. This will enable better preparation of teams for future races, as well as a better understanding of the dynamics of competition in the hybrid era of Formula One. The analysis results can be utilized for educating new team members and drivers, helping them better understand the factors affecting their performance and to develop more effective strategies for upcoming races. Outcome can be highly useful for people involved in betting, as it can predict the positions of drivers with greater accuracy, thereby increasing the chances of winning bets. The last use case is that FIA (The Fédération Internationale de l'Automobile) can leverage the model to evaluate the impact of new technical regulations. This can help enhance the overall spectacle for fans by ensuring closer and more exciting races.

1.2. Description of the data

The data for all sessions from the hybrid era (seasons 2022, 2023, 2024) were scraped from the official Formula 1 website [1], and the weather data were obtained using the FastF1 API [2]. After that, the data from each race weekend were appropriately merged and sorted. Below there is the representation of all columns in the prepared dataset:

1. **Year**: The year in which the race took place.
2. **Race**: The name or location of the race.
3. **No**: The car number of the driver.
4. **Driver**: The name of the driver.
5. **Car**: The team or manufacturer of the car.
6. **Pos (race)**: The final position of the driver in the race.
7. **Laps (race)**: The number of laps completed by the driver in the race.
8. **Time/Retired**: The race completion time of the driver or the reason for retirement if the driver did not finish the race.
9. **Points**: The points awarded to the driver for the race.
10. **Pos (qualifying)**: The position of the driver in the qualifying session.
11. **Q1**: The time of the driver in the first qualifying session.
12. **Q2**: The time of the driver in the second qualifying session.
13. **Q3**: The time of the driver in the third qualifying session.
14. **Pos (FP1)**: The position of the driver in the first free practice session.
15. **Time (FP1)**: The time of the driver in the first free practice session.
16. **Gap (FP1)**: The time gap between the driver and the fastest driver in the first free practice session.
17. **Laps (FP1)**: The number of laps completed by the driver in the first free practice session.
18. **Pos (FP2)**: The position of the driver in the second free practice session.
19. **Time (FP2)**: The time of the driver in the second free practice session.
20. **Gap (FP2)**: The time gap between the driver and the fastest driver in the second free practice session.
21. **Laps (FP2)**: The number of laps completed by the driver in the second free practice session.
22. **Pos (FP3)**: The position of the driver in the third free practice session.
23. **Time (FP3)**: The time of the driver in the third free practice session.
24. **Gap (FP3)**: The time gap between the driver and the fastest driver in the third free practice session.
25. **Laps (FP3)**: The number of laps completed by the driver in the third free practice session.
26. **Pos (Sprint)**: The position of the driver in the sprint race (if sprint weekend).
27. **Laps (Sprint)**: The number of laps completed by the driver in the sprint race (if sprint weekend).
28. **Time/Retired (Sprint)**: The sprint race completion time of the driver or the reason for retirement if the driver did not finish the sprint race (if sprint weekend).
29. **Points (Sprint)**: The points awarded to the driver for the sprint race (if sprint weekend).
30. **Pos (Sprint Quali)**: The position of the driver in the sprint qualifying session (if sprint weekend).
31. **SQ1**: The time of the driver in the first sprint qualifying session (if sprint weekend).
32. **SQ2**: The time of the driver in the second sprint qualifying session (if sprint weekend).
33. **SQ3**: The time of the driver in the third sprint qualifying session (if asprint weekend).

34. **Laps (Sprint Quali)**: The number of laps completed by the driver in the sprint qualifying session (if sprint weekend).
35. **AvgAirTemp**: The average air temperature during the race.
36. **AvgHumidity**: The average humidity during the race.
37. **AvgPressure**: The average atmospheric pressure during the race.
38. **TotalRainfall**: The total rainfall during the race.
39. **AvgTrackTemp**: The average track temperature during the race.
40. **AvgWindDirection**: The average wind direction during the race.
41. **AvgWindSpeed**: The average wind speed during the race.

1.3. Libraries Utilized in This Work

In [296...

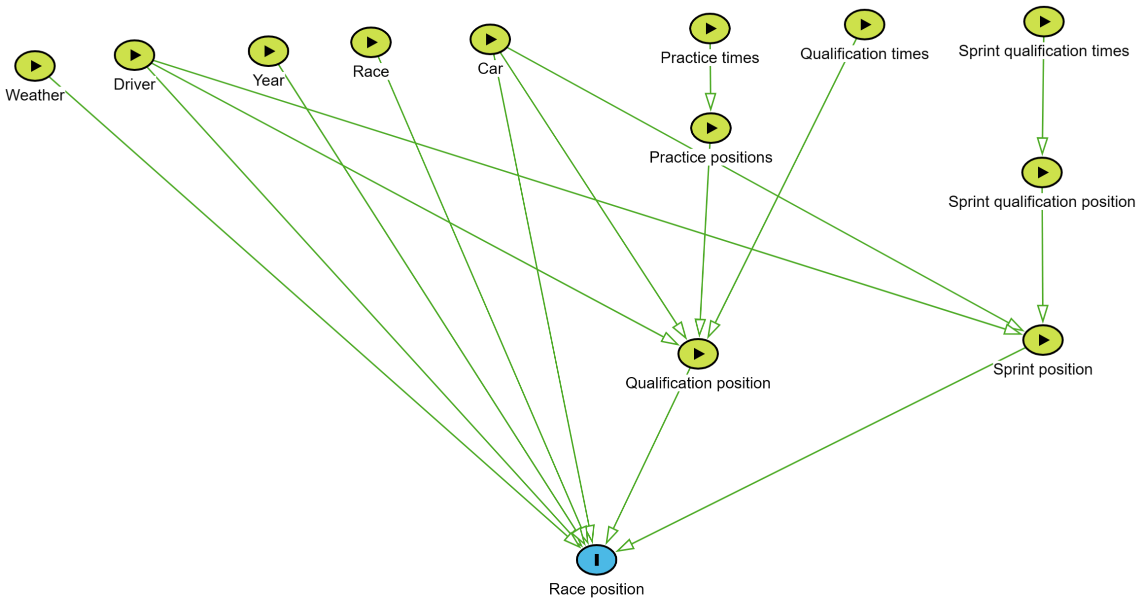
```
from cmdstanpy import CmdStanModel
import arviz as az
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import random
from PIL import Image
from IPython.display import display
```

1.4. DAG

To visualize the relationships between the variables, a Directed Acyclic Graph (DAG) was used.

In [297...

```
DAG = Image.open('DAG.png')
display(DAG)
```



In the models considered later in this paper, only certain variables deemed appropriate were selected. These variables were:

- Weather (track surface temperature and rain)
- Driver
- Year
- Race
- Car (manufacturer responsible for the design)
- Qualifying position

Variables that were excluded either had numerous data gaps (e.g., qualifications Q1, Q2, Q3) or were considered too insignificant to be included (e.g., positions in sprints).

1.5. Confoundings

1. **Pipes**:
   Weather influences Race position.
   Driver influences Race position.
   Driver influences Qualification position, which in turn influences Race position.
   Driver influences Sprint position, which in turn influences Race position.
   Year influences Race position.
   Race influences Race position.
   Car influences Race position.
   Car influences Qualification position, which in turn influences Race position.
   Car influences Sprint position, which in turn influences Race position.
   Practice times influence Practice positions, which in turn influence Qualification position, which further influences Race position.
   Qualification times determine Qualification position.
   Sprint qualification times determine Sprint qualification position, which in turn influences Race position.

2. **Forks**:
   Driver cause changes in Race position, Qualification position and Sprint position,
   Car cause changes in Race position, Qualification position and Sprint position.

3. **Colliders**:
   Colliders are Qualification position and Sprint position which cause changes in next collider - Race position which is our outcome of analysis.

2. Data Preprocessing

2.1. Orginal Data

In [298…

```
original_data = pd.read_csv('final_f1_data.csv')
original_data.head()
```

Out[298…

	Year	Race	No	Driver	Car	Pos (race)	Laps (race)	Time/Retired	Points	(qua
0	2022	Bahrain	16	Charles Leclerc LEC	Ferrari	1	57.0	1:37:33:584	26.0	
1	2022	Bahrain	10	Pierre Gasly GAS	AlphaTauri RBPT	NC	44.0	DNF	0.0	
2	2022	Bahrain	1	Max Verstappen VER	Red Bull Racing RBPT	19	54.0	DNF	0.0	
3	2022	Bahrain	11	Sergio Perez PER	Red Bull Racing RBPT	18	56.0	DNF	0.0	
4	2022	Bahrain	27	Nico Hulkenberg HUL	Aston Martin Aramco Mercedes	17	57.0	+63.829s	0.0	

5 rows × 41 columns

In [299…

```
original_data.describe()
```

Out[299…

	Year	No	Laps (race)	Points	Pos (FP1)	Laps (FP1)	Pos
count	1048.000000	1048.000000	998.000000	998.000000	996.000000	996.000000	805.0(
mean	2022.669847	27.684160	54.092184	5.105210	10.464859	21.462851	10.3!
std	0.674099	23.115599	16.418655	7.259757	5.753720	7.594340	5.7
min	2022.000000	1.000000	0.000000	0.000000	1.000000	1.000000	1.0(
25%	2022.000000	11.000000	50.000000	0.000000	5.000000	19.000000	5.0(
50%	2023.000000	22.000000	57.000000	1.000000	10.000000	23.000000	10.0(
75%	2023.000000	44.000000	65.000000	9.750000	15.000000	26.000000	15.0(
max	2024.000000	99.000000	78.000000	26.000000	20.000000	39.000000	20.0(

In [300…

```
original_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048 entries, 0 to 1047
Data columns (total 41 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Year                   1048 non-null   int64  
 1   Race                   1048 non-null   object 
 2   No                     1048 non-null   int64  
 3   Driver                 1048 non-null   object 
 4   Car                    1048 non-null   object 
 5   Pos (race)             998 non-null    object 
 6   Laps (race)            998 non-null    float64
 7   Time/Retired           998 non-null    object 
 8   Points                 998 non-null    float64
 9   Pos (qualifying)       998 non-null    object 
 10  Q1                     998 non-null    object 
 11  Q2                     744 non-null    object 
 12  Q3                     494 non-null    object 
 13  Pos (FP1)              996 non-null    float64
 14  Time (FP1)             959 non-null    object 
 15  Gap (FP1)              909 non-null    object 
 16  Laps (FP1)             996 non-null    float64
 17  Pos (FP2)              805 non-null    float64
 18  Time (FP2)             793 non-null    object 
 19  Gap (FP2)              752 non-null    object 
 20  Laps (FP2)             805 non-null    float64
 21  Pos (FP3)              777 non-null    float64
 22  Time (FP3)             770 non-null    object 
 23  Gap (FP3)              731 non-null    object 
 24  Laps (FP3)             777 non-null    float64
 25  Pos (Sprint)           219 non-null    object 
 26  Laps (Sprint)          219 non-null    float64
 27  Time/Retired (Sprint)  219 non-null    object 
 28  Points (Sprint)        219 non-null    float64
 29  Pos (Sprint Quali)     160 non-null    object 
 30  SQ1                    160 non-null    object 
 31  SQ2                    117 non-null    object 
 32  SQ3                    79 non-null     object 
 33  Laps (Sprint Quali)    160 non-null    float64
 34  AvgAirTemp             1048 non-null   float64
 35  AvgHumidity            1048 non-null   float64
 36  AvgPressure            1048 non-null   float64
 37  TotalRainfall          1048 non-null   int64  
 38  AvgTrackTemp           1048 non-null   float64
 39  AvgWindDirection       1048 non-null   float64
 40  AvgWindSpeed           1048 non-null   float64
dtypes: float64(17), int64(3), object(21)
memory usage: 335.8+ KB
```

2.2. Preprocessing of the orginal data

1. Selecting relevant and useful columns:

    - Year
    - Race
    - Driver
    - Car
    - Pos (race)

- Points
- Pos (qualifying)
- TotalRainfall
- AvgTrackTemp
- Qualifying position

We rejected the rest of the data because the data contained records that were difficult to analyze (many NaN values in the case of qualifying and training session times), data regarding sprints because they take place relatively rare, and their formula changed.

2. Data Cleaning: Delete records with drivers who participated only in free practice sessions and replacing `NC` and `DQ` Values in the **Pos (race)** and **Pos (qualifying)** columns with the Last possible position

3. Choosing drivers who participated in each of the years considered.

In [301…

```
# Step 1
relevant_columns = ["Year", "Race", "Driver", "Car", "Pos (race)", "Points", "Pc
df_processed = original_data.filter(relevant_columns)
```

In [302…

```
# # Step 2
df_processed = df_processed.dropna(subset=['Pos (race)'])  # Remove rows where '
df_processed['Pos (race)'] = df_processed['Pos (race)'].astype(str)
df_processed['Pos (qualifying)'] = df_processed['Pos (qualifying)'].astype(str)

# Function to replace "NC" and "DQ" values with the next available place
def replace_nc_dq(group, column):
    # Sort by the given column to determine the next place
    positions = sorted([int(pos) for pos in group[column] if pos.isdigit()])
    next_place = max(positions) + 1 if positions else 1
    # Replace "NC" and "DQ" with the next place
    group[column] = group[column].replace(['NC', 'DQ'], next_place)
    return group

for col in ['Pos (race)', 'Pos (qualifying)']:
    df_processed = df_processed.groupby(['Year', 'Race'], group_keys=False).appl
    df_processed[col] = pd.to_numeric(df_processed[col], errors='coerce').fillna
```

In [303…

```
# Step 3
# Filter data for each year
drivers_2022 = set(df_processed[df_processed['Year'] == 2022]['Driver'])
drivers_2023 = set(df_processed[df_processed['Year'] == 2023]['Driver'])
drivers_2024 = set(df_processed[df_processed['Year'] == 2024]['Driver'])
# Find common drivers
common_drivers = drivers_2022 & drivers_2023 & drivers_2024
# Filter rows for these common drivers
df_processed = df_processed[df_processed['Driver'].isin(common_drivers)]
```

In [304…

```
df_processed.head()
```

Out[304…

	Year	Race	Driver	Car	Pos (race)	Points	Pos (qualifying)	TotalRainfall	Avg
0	2022	Bahrain	Charles Leclerc LEC	Ferrari	1	26.0	1	0	
1	2022	Bahrain	Pierre Gasly GAS	AlphaTauri RBPT	20	0.0	10	0	
2	2022	Bahrain	Max Verstappen VER	Red Bull Racing RBPT	19	0.0	2	0	
3	2022	Bahrain	Sergio Perez PER	Red Bull Racing RBPT	18	0.0	4	0	
4	2022	Bahrain	Nico Hulkenberg HUL	Aston Martin Aramco Mercedes	17	0.0	17	0	

In [305…

```
df_processed.to_csv('data_processing/processed_data.csv', index=False)
```

3. Models

The objective of all three models was to determine the positions y of drivers in a Formula 1 races.

$$y \sim Binomial(N, \theta)$$

In Stan, the binomial distribution models the number of successes in a fixed number of trials, each with the same probability of success θ.

For analyzing Formula 1 race positions from 2022 to 2024, the binomial distribution is ideal as it handles discrete outcomes like drivers' finishing positions. With 20 drivers per race, the number of trials N is 19, and the success probability θ is calculated separately for each model.
This allows the estimation of θ, the likelihood of a driver finishing in specific positions, considering multiple races as repeated trials.

Input values in Stan include N =19 and θ, while the output is the binomial distribution providing insights into driver performance across races. This straightforward model effectively captures race result variability due to factors in the hybrid era.

3.1. Model 1

$$\theta = \alpha_{\text{driver}} + \alpha_{\text{car}}$$

$$\alpha_{\text{driver}} \sim \mathcal{N}(0, \sigma)$$

$$\alpha_{\text{car}} \sim \mathcal{N}(0, \sigma)$$

Model 1 is a simple reference model [5] on which Models 2 and 3 are based.The θ was determined based on the average skills of the driver α_{driver} and the average benefits of the given car constructor α_{car}. Both parameters are represented as normal distributions with a standard deviation of σ. The same σ is used in both cases to ensure that the values of both distributions are similar and have an equal impact on the final evaluation of θ.

Descriptions of the parameters:

- **Parameter** α_{driver}
  The average skills of each driver are assessed by assigning them a value from a normal distribution with a standard deviation σ. Positive values suggest that the driver is below average, while negative values suggest that the driver is above average.

- **Parameter** α_{car}
  The average benefits for each driver from selecting a car by a particular constructor are assessed by assigning each a value derived from a normal distribution with standard deviation σ. Positive values suggest that the constructor's performance is below average, whereas negative values suggest that the constructor's performance is above average.

3.2. Model 2

$$\theta = \alpha_{\text{driver}} + \alpha_{\text{rain}} + \alpha_{\text{car}} + \alpha_{\text{temp}}$$

$$\alpha_{\text{driver}} \sim \mathcal{N}(0, \sigma)$$

$$\alpha_{\text{rain}} \sim \mathcal{N}(0, \sigma_{\text{rain}}) \cdot \text{rainy}$$

$$\alpha_{\text{car}} \sim \mathcal{N}(0, \sigma)$$

$$\alpha_{\text{temp}} \sim \mathcal{N}(0, \sigma_{\text{temp}})$$

Weather conditions are a crucial factor in all types of sports, capable of changing the outcome in ways that can surprise spectators. The goal of Model 2 was to expand the basic model by incorporating weather-related information, such as track temperature and whether it rained during the race.

Descriptions of the new parameters:

- **Parameter** α_{rain}
  Rain in Formula 1 races, though rare, is a factor that can significantly impact the final positions of drivers, revealing which ones handle it better and which ones handle it worse. The variable **rainy** is a boolean variable indicating whether it rained during the race. Depending on whether the value of **rainy** is 0 or 1, an additional normal distribution with a standard deviation of σ_{rain} is included in the calculation of θ, which should be interpreted as the driver's skill in handling the race when it rains.

- **Parameter α_{temp}**
  Track temperature can also have a significant impact on race outcomes. The parameter α_{temp} is modeled as a normal distribution with a standard deviation of σ_{temp}. This parameter accounts for the varying performance of drivers and cars under different temperature conditions, reflecting how well they adapt to changes in track temperature during the race. By including α_{temp}, the model can better evaluate a driver's skill and car performance in different thermal environments, providing a more comprehensive understanding of factors influencing race results.

3.3. Model 3

$$\theta = \alpha_{\text{driver}} + \alpha_{\text{rain}} + \alpha_{\text{car}} + \alpha_{\text{quals}}$$

$$\alpha_{\text{driver}} \sim \mathcal{N}(0, \sigma)$$

$$\alpha_{\text{rain}} \sim \mathcal{N}(0, \sigma_{\text{rain}}) \cdot \text{rainy}$$

$$\alpha_{\text{car}} \sim \mathcal{N}(0, \sigma)$$

$$\alpha_{\text{quals}} \sim \mathcal{N}(0, \sigma_{\text{quals}}) \cdot \text{passingRatio}$$

In Formula 1 racing, before each race in which points are earned to determine the season standings, there are qualifying sessions consisting of three rounds: Q1, Q2, and Q3. Based on the results of these sessions, each driver receives their starting position. Overtaking during a Formula 1 race is difficult and relatively rare, except in unpredictable situations (such as rain, where it becomes more frequent).
The position obtained in the qualifications is very important for the race because it is much easier for drivers in the leading positions to maintain their lead. The goal of this model is to account for these positions and their impact.

Descriptions of the new parameters:

- **Parameter α_{quals}**
  The parameter α_{quals} is designed to incorporate the influence of the driver's starting position on the final race outcome. It is modeled as a normal distribution with a standard deviation of σ_{quals}, scaled by the variable **passingRatio**, which represents the relative difficulty of overtaking during the race. A higher passing ratio indicates a greater likelihood of overtaking, reducing the impact of the qualifying position on the race result. Conversely, a lower passing ratio emphasizes the importance of a good starting position, as overtaking is more challenging. By including α_{quals} in the model, we can better understand how the qualifying performance and the race dynamics interact to influence the driver's final standing.

4. Priors

4.1. Explanation why particular priors for parameters were selected

- **Parameter** α_{driver}
  The skill of each driver is modeled using a normal distribution $\mathcal{N}(0, \sigma)$. The choice of a normal distribution is based on the assumption that driver skills are symmetrically distributed around an average level, with some drivers performing better and others worse. The standard deviation σ reflects the variability in driver skills.

- **Parameter** α_{car}
  The performance benefit provided by each car constructor is also modeled using a normal distribution $\mathcal{N}(0, \sigma)$. This assumption is based on the belief that car performance, like driver skills, varies around an average level. The same σ as for drivers ensures that the impact of cars and drivers on race outcomes is comparable.

- **Parameter** α_{rain}
  The influence of rain is modeled with $\mathcal{N}(0, \sigma_{\text{rain}}) \cdot \text{rainy}$. The normal distribution reflects the variability in drivers' abilities to handle wet conditions, with the boolean variable rainy accounting for whether it rained. This captures the rare but impactful nature of rain on race results.

- **Parameter** α_{temp}
  Track temperature effects are modeled with $\mathcal{N}(0, \sigma_{\text{temp}})$. The normal distribution is chosen to capture the varying impact of temperature on driver and car performance. The standard deviation σ_{temp} represents the extent of this variability.

- **Parameter** α_{quals}
  The effect of qualifying position is modeled with $\mathcal{N}(0, \sigma_{\text{quals}})\cdot$ passingRatio. The normal distribution represents the variability in the importance of starting positions, with the passingRatio scaling this effect based on how easy it is to overtake in a race. This accounts for the significant role of qualifying performance in determining race outcomes.

4.2. Prior predictive checks for parameters

Predictive checks involve generating simulated data from the model using the distributions of the parameters and comparing it to the observed data. This helps to assess the model's ability to replicate real-world outcomes.

Procedure

1. Generate Simulated Data: Using the distributions of the parameters α_{driver}, α_{car}, α_{rain}, α_{temp}, and α_{quals}, generate simulated race results.
2. Analyze Observed Data: Compare the distribution of simulated race positions to the observed positions from the actual race data.
3. Diagnostic Metrics: Use metrics such as the predictive p-value, which measures the proportion of simulated data that is more extreme than the observed data. Values near 0.5 indicate good model fit.

Interpretation

- If the simulated data closely matches the observed data, this suggests that the model's parameters are appropriately capturing the underlying processes influencing race outcomes.
- Discrepancies between simulated and observed data can indicate areas where the model might be improved, such as by including additional factors or refining the priors.

4.3. Pior predictive checks for measurements

Prior predictive checks involve generating simulated data from the model using the prior distributions of the parameters, before observing any data. This helps to assess whether the chosen priors produce plausible outcomes.

Procedure

1. Generate Simulated Data: Using the prior distributions for α_{driver}, α_{car}, α_{rain}, α_{temp}, and α_{quals}, generate simulated race results.
2. Compare to Plausible Outcomes: Compare the distribution of simulated race positions to what is considered plausible based on domain knowledge of Formula 1 racing.
3. Diagnostic Metrics: Use metrics such as the range and mean of the simulated race positions to ensure they fall within expected bounds (e.g., race positions between 1 and 20).

Interpretation

- If the simulated data from the priors results in implausible outcomes (e.g., all drivers finishing in the top positions or all in the bottom positions), this suggests that the priors may need to be adjusted.
- Plausible prior predictive checks ensure that the priors are reasonable and do not introduce bias or unrealistic assumptions into the model.

4.4. Explanation how prior parameters were selected

The selection of prior parameters involves choosing the mean and standard deviation for the normal distributions used for each parameter. The following considerations were made:

- **Domain Knowledge**: Historical data and expert knowledge about the typical performance of drivers and cars, as well as the impact of rain, temperature, and qualifying positions, were used to inform the priors.
- **Centering and Variability**: Priors were centered around zero (e.g., $\mathcal{N}(0, \sigma)$) to reflect that, on average, drivers and cars are expected to be neither exceptionally good nor bad. The standard deviations (σ, σ_{rain}, σ_{temp}, σ_{quals}) were chosen to reflect realistic levels of variability observed in past race data.
- **Prior Predictive Checks**: Iterative adjustments were made to the prior parameters based on the outcomes of prior predictive checks, ensuring that the priors produced plausible simulated data before any real data was incorporated.

After thorough analysis and iterative prior predictive checks, the following values were selected for the standard deviations of the normal distributions:

- $\sigma = 1$
  This value reflects the typical variability in driver skills and car performance, ensuring a balanced impact of both factors on the race outcomes.

- $\sigma_{\text{rain}} = 2$
  A higher value was chosen to capture the significant and variable impact of rain on race outcomes, reflecting the increased difficulty and unpredictability of wet conditions.

- $\sigma_{\text{temp}} = 0.5$
  This value represents the variability in performance due to different track temperatures, accounting for how well drivers and cars adapt to varying thermal environments.

- $\sigma_{\text{quals}} = 1$
  This value captures the importance of qualifying positions on race outcomes, scaled by the passing ratio to account for the difficulty of overtaking during races.

By carefully selecting and validating the prior parameters through these steps, the model is grounded in realistic assumptions and is better equipped to make accurate predictions about race outcomes in the hybrid era of Formula 1 racing.

4.5. Prior for model 1

In [306…

```
df = pd.read_csv('data_processing/processed_data.csv')
unique_drivers = df['Driver'].nunique()
data = {
    'D': unique_drivers,
}

model = CmdStanModel(stan_file='stan/model1_prior.stan')
prior_model_1 = model.sample(data=data, iter_sampling=1000, chains=1, fixed_para
```

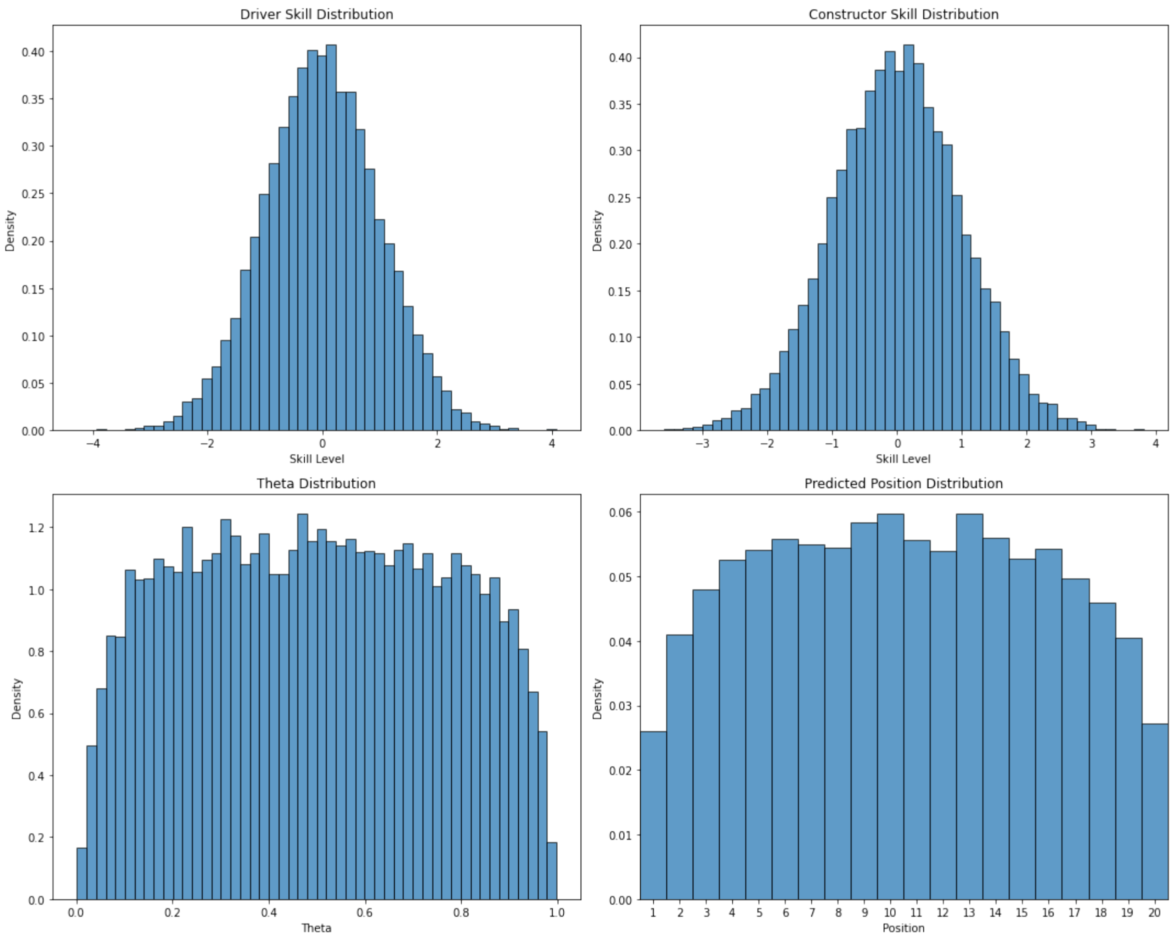
```
INFO:cmdstanpy:found newer exe file, not recompiling
INFO:cmdstanpy:CmdStan start processing
chain 1 |██████████| 00:00 Sampling completed
```

```
INFO:cmdstanpy:CmdStan done processing.
```

In [307…

```
driver_skill_flat = prior_model_1.stan_variable('driver_skill').flatten()
constructor_skill_flat = prior_model_1.stan_variable('constructor_skill').flatte
position_flat = prior_model_1.stan_variable('position').flatten() + 1
theta_flat = prior_model_1.stan_variable('theta').flatten()

fig, axs = plt.subplots(2, 2, figsize=(15, 12))

axs[0, 0].hist(driver_skill_flat, bins=50, edgecolor='black', alpha=0.7, density
axs[0, 0].set_title('Driver Skill Distribution')
axs[0, 0].set_xlabel('Skill Level')
axs[0, 0].set_ylabel('Density')
```

```python
axs[0, 1].hist(constructor_skill_flat, bins=50, edgecolor='black', alpha=0.7, de
axs[0, 1].set_title('Constructor Skill Distribution')
axs[0, 1].set_xlabel('Skill Level')
axs[0, 1].set_ylabel('Density')

axs[1, 0].hist(theta_flat, bins=50, edgecolor='black', alpha=0.7, density=True)
axs[1, 0].set_title('Theta Distribution')
axs[1, 0].set_xlabel('Theta')
axs[1, 0].set_ylabel('Density')

axs[1, 1].hist(position_flat, bins=np.arange(1, 22) - 0.5, edgecolor='black', al
axs[1, 1].set_title('Predicted Position Distribution')
axs[1, 1].set_xlabel('Position')
axs[1, 1].set_ylabel('Density')
axs[1, 1].set_xticks(range(1, 21))
axs[1, 1].set_xlim([0.5, 20.5])

plt.tight_layout()
plt.show()
```



- The symmetry around zero in the top histograms indicates that there is no inherent bias in the priors towards higher or lower values for driver skill or car performance.
- The standard deviation of 1 for both σ_{driver} and σ_{car} suggests moderate variability, meaning that while most values are close to zero, there is a reasonable chance of observing more extreme values.
- The bottom histograms highlight the range of θ values that can be expected before observing any data, demonstrating the combined variability introduced by both

driver and car factors.

4.6. Prior for model 2

In [308…

```
df = pd.read_csv('data_processing/processed_data.csv')
unique_drivers = df['Driver'].nunique()
rainy = [random.choice([0, 1]) for _ in range(unique_drivers)]
data = {
    'D': unique_drivers,
    'rainy': rainy
}

model = CmdStanModel(stan_file='stan/model2_prior.stan')
prior_model_2 = model.sample(data=data, iter_sampling=1000, chains=1, fixed_para
```

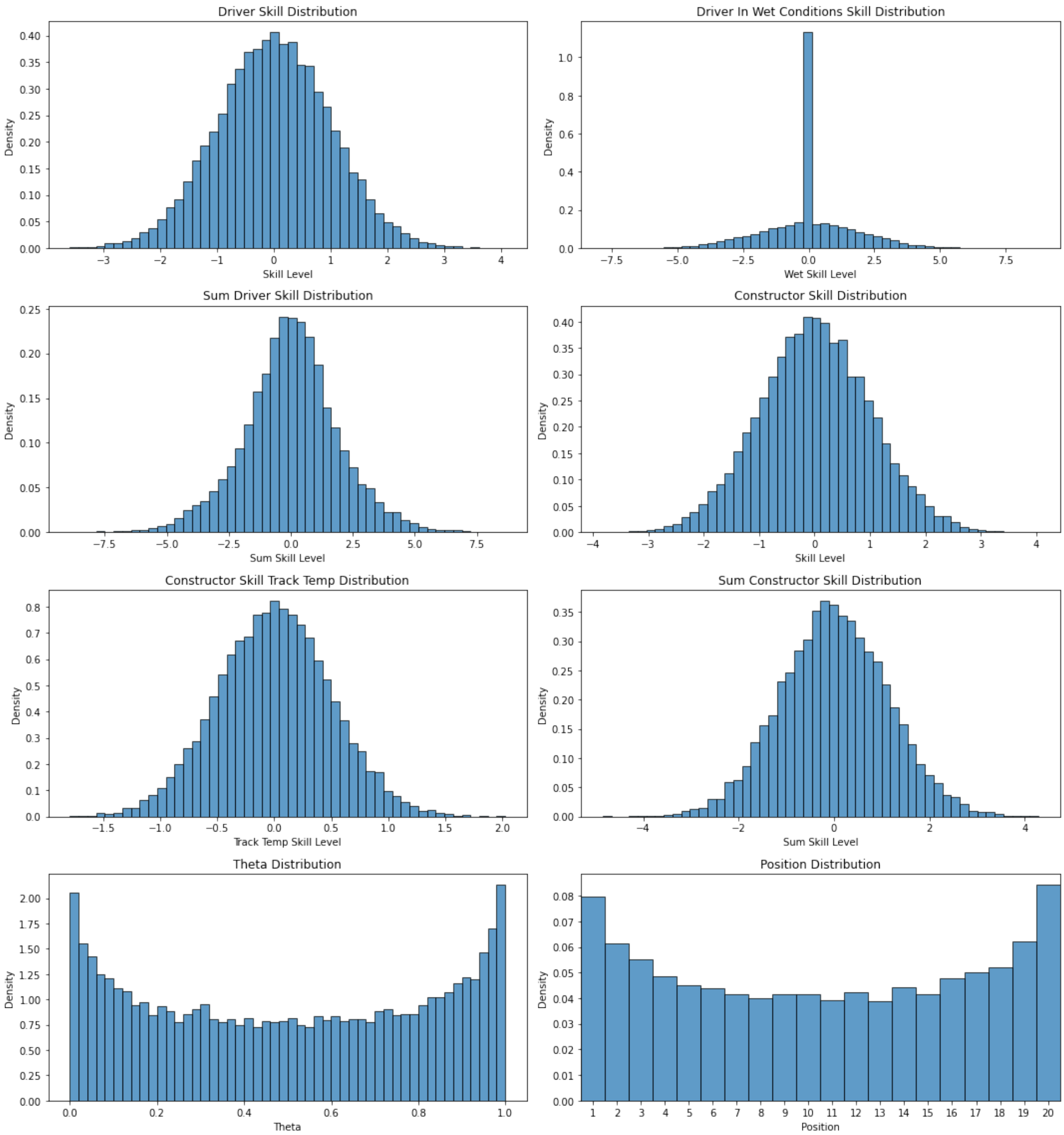
```
INFO:cmdstanpy:found newer exe file, not recompiling
INFO:cmdstanpy:CmdStan start processing
chain 1 |          | 00:00 Sampling completed
```

```
INFO:cmdstanpy:CmdStan done processing.
```

In [309…

```
driver_skill_flat = prior_model_2.stan_variable('driver_skill').flatten()
driver_skill_wet_flat = prior_model_2.stan_variable('driver_skill_wet').flatten(
driver_skill_sum_flat = prior_model_2.stan_variable('driver_skill_sum').flatten(
constructor_skill_flat = prior_model_2.stan_variable('constructor_skill').flatte
constructor_skill_flat_track_temp = prior_model_2.stan_variable('constructor_ski
constructor_skill_flat_sum = prior_model_2.stan_variable('constructor_skill_sum'
position_flat = prior_model_2.stan_variable('position').flatten() + 1
theta_flat = prior_model_2.stan_variable('theta').flatten()


fig, axs = plt.subplots(4, 2, figsize=(15, 16))

axs[0, 0].hist(driver_skill_flat, bins=50, edgecolor='black', alpha=0.7, density
axs[0, 0].set_title('Driver Skill Distribution')
axs[0, 0].set_xlabel('Skill Level')
axs[0, 0].set_ylabel('Density')

axs[0, 1].hist(driver_skill_wet_flat, bins=50, edgecolor='black', alpha=0.7, den
axs[0, 1].set_title('Driver In Wet Conditions Skill Distribution')
axs[0, 1].set_xlabel('Wet Skill Level')
axs[0, 1].set_ylabel('Density')

axs[1, 0].hist(driver_skill_sum_flat, bins=50, edgecolor='black', alpha=0.7, den
axs[1, 0].set_title('Sum Driver Skill Distribution')
axs[1, 0].set_xlabel('Sum Skill Level')
axs[1, 0].set_ylabel('Density')

axs[1, 1].hist(constructor_skill_flat, bins=50, edgecolor='black', alpha=0.7, de
axs[1, 1].set_title('Constructor Skill Distribution')
axs[1, 1].set_xlabel('Skill Level')
axs[1, 1].set_ylabel('Density')

axs[2, 0].hist(constructor_skill_flat_track_temp, bins=50, edgecolor='black', al
axs[2, 0].set_title('Constructor Skill Track Temp Distribution')
axs[2, 0].set_xlabel('Track Temp Skill Level')
axs[2, 0].set_ylabel('Density')
```

```
axs[2, 1].hist(constructor_skill_flat_sum, bins=50, edgecolor='black', alpha=0.7
axs[2, 1].set_title('Sum Constructor Skill Distribution')
axs[2, 1].set_xlabel('Sum Skill Level')
axs[2, 1].set_ylabel('Density')

axs[3, 0].hist(theta_flat, bins=50, edgecolor='black', alpha=0.7, density=True)
axs[3, 0].set_title('Theta Distribution')
axs[3, 0].set_xlabel('Theta')
axs[3, 0].set_ylabel('Density')

axs[3, 1].hist(position_flat, bins=np.arange(1, 22) - 0.5, edgecolor='black', al
axs[3, 1].set_title('Position Distribution')
axs[3, 1].set_xlabel('Position')
axs[3, 1].set_ylabel('Density')
axs[3, 1].set_xticks(range(1, 21))
axs[3, 1].set_xlim([0.5, 20.5])

plt.tight_layout()
plt.show()
```



The introduction of weather parameters, specifically related to weather and track temperature, in the second model caused an interesting shift in the distribution of positions. In the first model, the highest probabilities were centered in the middle, whereas in the second model, the highest probabilities were for the first and last positions. From the remaining charts, it can be observed that the main reason for this is rain, which makes both overtaking and failing to finish the race (for example, due to an accident caused by slipping) more likely.

4.7. Prior for model 3

In [310…

```
df = pd.read_csv('data_processing/processed_data.csv')
unique_drivers = df['Driver'].nunique()
rainy = [random.choice([0, 1]) for _ in range(unique_drivers)]
qualifying_position = [random.randint(1, 20) for _ in range(unique_drivers)]
data = {
    'D': unique_drivers,
    'rainy': rainy,
    'qualifying_position': qualifying_position
}

model = CmdStanModel(stan_file='stan/model3_prior.stan')
prior_model_3 = model.sample(data=data, iter_sampling=1000, chains=1, fixed_para
```

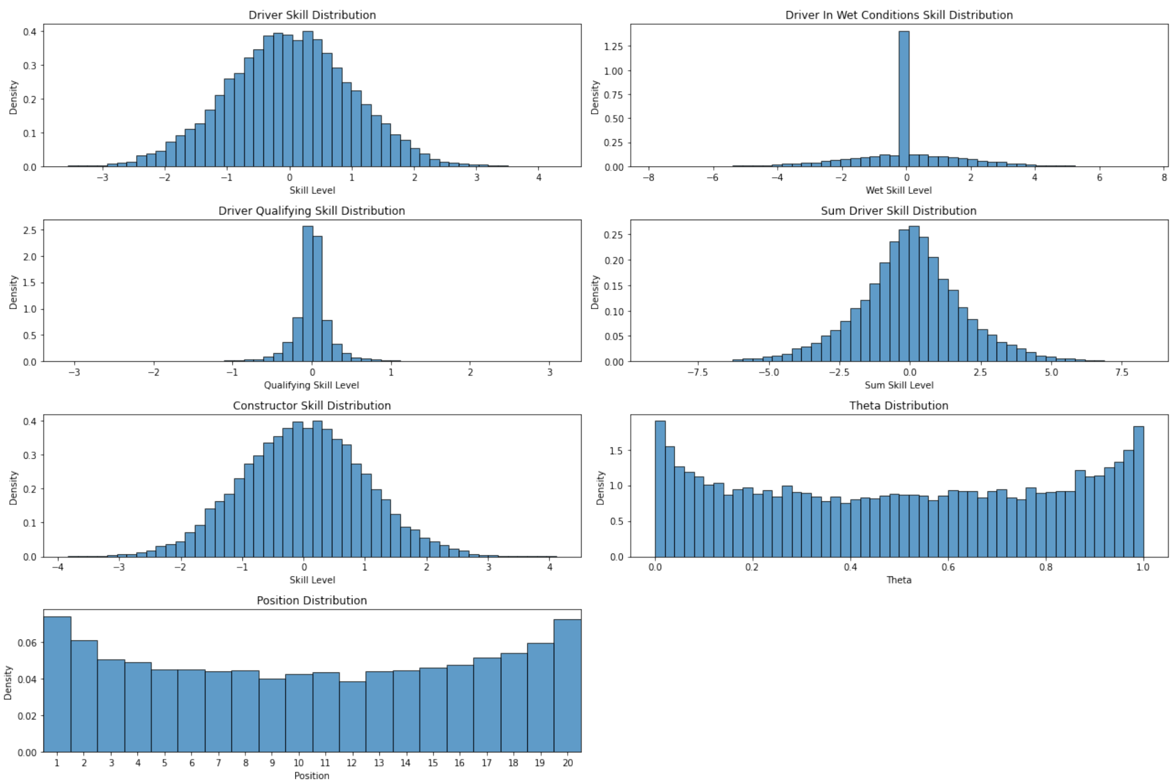
```
INFO:cmdstanpy:found newer exe file, not recompiling
INFO:cmdstanpy:CmdStan start processing
chain 1 |          | 00:00 Sampling completed
```

```
INFO:cmdstanpy:CmdStan done processing.
```

In [311…

```
driver_skill_flat = prior_model_3.stan_variable('driver_skill').flatten()
driver_skill_wet_flat = prior_model_3.stan_variable('driver_skill_wet').flatten(
driver_skill_qualifying = prior_model_3.stan_variable('driver_skill_qualifying')
driver_skill_sum_flat = prior_model_3.stan_variable('driver_skill_sum').flatten(
constructor_skill_flat = prior_model_3.stan_variable('constructor_skill').flatte
position_flat = prior_model_3.stan_variable('position').flatten() + 1
theta_flat = prior_model_3.stan_variable('theta').flatten()

fig, axs = plt.subplots(4, 2, figsize=(18, 12))

axs[0, 0].hist(driver_skill_flat, bins=50, edgecolor='black', alpha=0.7, density
axs[0, 0].set_title('Driver Skill Distribution')
axs[0, 0].set_xlabel('Skill Level')
axs[0, 0].set_ylabel('Density')

axs[0, 1].hist(driver_skill_wet_flat, bins=50, edgecolor='black', alpha=0.7, den
axs[0, 1].set_title('Driver In Wet Conditions Skill Distribution')
axs[0, 1].set_xlabel('Wet Skill Level')
axs[0, 1].set_ylabel('Density')

axs[1, 0].hist(driver_skill_qualifying, bins=50, edgecolor='black', alpha=0.7, d
axs[1, 0].set_title('Driver Qualifying Skill Distribution')
axs[1, 0].set_xlabel('Qualifying Skill Level')
axs[1, 0].set_ylabel('Density')

axs[1, 1].hist(driver_skill_sum_flat, bins=50, edgecolor='black', alpha=0.7, den
axs[1, 1].set_title('Sum Driver Skill Distribution')
axs[1, 1].set_xlabel('Sum Skill Level')
axs[1, 1].set_ylabel('Density')
```

```
axs[2, 0].hist(constructor_skill_flat, bins=50, edgecolor='black', alpha=0.7, de
axs[2, 0].set_title('Constructor Skill Distribution')
axs[2, 0].set_xlabel('Skill Level')
axs[2, 0].set_ylabel('Density')

axs[2, 1].hist(theta_flat, bins=50, edgecolor='black', alpha=0.7, density=True)
axs[2, 1].set_title('Theta Distribution')
axs[2, 1].set_xlabel('Theta')
axs[2, 1].set_ylabel('Density')

axs[3, 0].hist(position_flat, bins=np.arange(1, 22) - 0.5, edgecolor='black', al
axs[3, 0].set_title('Position Distribution')
axs[3, 0].set_xlabel('Position')
axs[3, 0].set_ylabel('Density')
axs[3, 0].set_xticks(range(1, 21))
axs[3, 0].set_xlim([0.5, 20.5])

fig.delaxes(axs[3, 1])
plt.tight_layout()
plt.show()
```



- The histograms for the combined distributions indicate a broader range of possible values for θ, reflecting the realistic complexity and variability in race outcomes when all influential factors are considered.
- The parameter defining the position positively influenced the determination of the position. The values of θ and the predicted positions aligned, considering not only whether it was raining but also the starting positions of each driver in the race.

5. Posterior analysis (model 1)

5.1. Issues with the sampling

There were no issues. All samplings were conducted without any obstacles.

5.2. Analysis of samples from the posterior predictive distribution

In [312…

```
df = pd.read_csv('data_processing/processed_data.csv')
df = df.dropna(subset=['Driver', 'Car'])
unique_drivers = df['Driver'].nunique()
unique_constructors = df['Car'].nunique()
drivers_indices = pd.factorize(df['Driver'])[0] + 1
constructors_indices = pd.factorize(df['Car'])[0] + 1
df['Pos (race)'] = df['Pos (race)'] - 1
positions = df['Pos (race)'].values
data = {
    'N': len(df),
    'D': unique_drivers,
    'C': unique_constructors,
    'drivers': drivers_indices,
    'constructors': constructors_indices,
    'position': positions
}
model = CmdStanModel(stan_file='stan/model1_posterior.stan')
posterior_model_1 = model.sample(data=data, iter_warmup=1, iter_sampling=1000, c
```

```
INFO:cmdstanpy:found newer exe file, not recompiling
INFO:cmdstanpy:CmdStan start processing
chain 1 |          | 00:01 Sampling completed
```

```
INFO:cmdstanpy:CmdStan done processing.
```

In [313…

```
def plot_comparision_between_simulated_and_observed_for_drivers(model, df):
    df = pd.read_csv('data_processing/processed_data.csv')
    n_rows, n_cols = 6, 3
    fig, axes = plt.subplots(n_rows, n_cols, figsize=(8*n_cols, 5*n_rows))
    n_bins = np.arange(22) - 0.5

    drivers_names = df['Driver'].unique().tolist()

    for driver_index, driver_name in enumerate(drivers_names):
        specified_driver = df['Driver'].eq(driver_name)
        results = df[specified_driver]

        row_idx = driver_index // n_cols
        col_idx = driver_index % n_cols

        axes[row_idx, col_idx].hist((results['Pos (race)']).tolist(), bins=n_bin

        results_idx = results.index
        simulated_positions = model.T[results_idx].flatten() + 1
        axes[row_idx, col_idx].hist(simulated_positions, bins=n_bins, rwidth=1,
        axes[row_idx, col_idx].set_xticks(range(22))
        axes[row_idx, col_idx].set_xlim([0, 21])
        axes[row_idx, col_idx].set_yticks([])
        axes[row_idx, col_idx].set_title(driver_name.split()[-1] + ' Finishing P
        axes[row_idx, col_idx].legend()
        axes[row_idx, col_idx].set_xlabel('Position')
```

```
    fig.tight_layout()
    plt.show()
```

In [314…

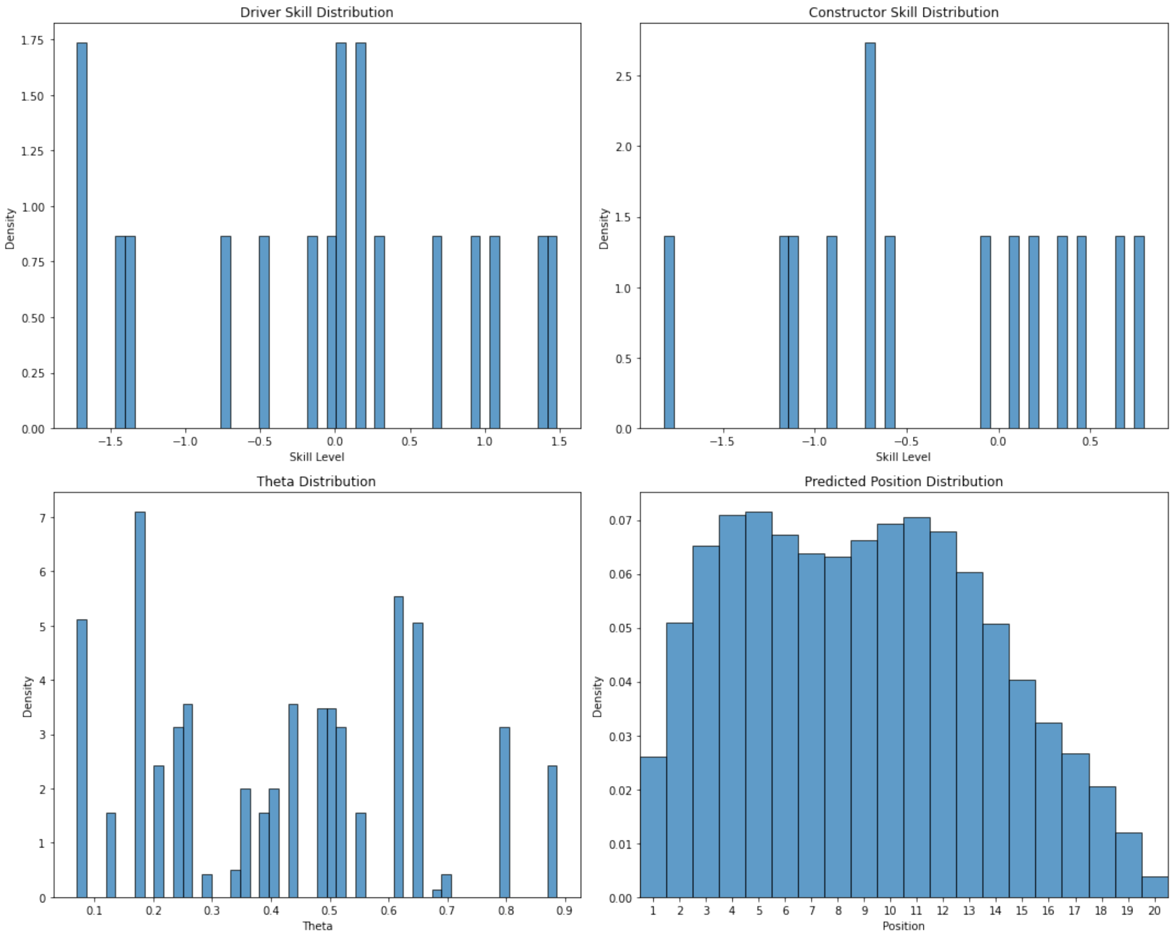
```
positions_predicted_model1 = posterior_model_1.stan_variable('position_predicted
plot_comparision_between_simulated_and_observed_for_drivers(positions_predicted_
```

The posterior predictive distributions from Model 1 generally align with the observed race results, indicating that the model captures the central tendencies and variability of driver performance adequately. However, there are instances where the simulated distributions exhibit biases, either overestimating or underestimating the drivers' performance. The model captures the overall spread of the data, but occasionally fails to account for extreme values and distribution shapes in certain cases. Overall, Model 1 provides a reasonable fit, but there is room for improvement in capturing more nuanced aspects of the race outcomes.

5.3. Analysis of parameter marginal distributions

In [315…

```
driver_skill_flat = posterior_model_1.stan_variable('driver_skill').flatten()
constructor_skill_flat = posterior_model_1.stan_variable('constructor_skill').fl
position_flat = posterior_model_1.stan_variable('position_predicted').flatten()
theta_flat = posterior_model_1.stan_variable('theta').flatten()

fig, axs = plt.subplots(2, 2, figsize=(15, 12))

axs[0, 0].hist(driver_skill_flat, bins=50, edgecolor='black', alpha=0.7, density
axs[0, 0].set_title('Driver Skill Distribution')
axs[0, 0].set_xlabel('Skill Level')
axs[0, 0].set_ylabel('Density')

axs[0, 1].hist(constructor_skill_flat, bins=50, edgecolor='black', alpha=0.7, de
axs[0, 1].set_title('Constructor Skill Distribution')
axs[0, 1].set_xlabel('Skill Level')
axs[0, 1].set_ylabel('Density')

axs[1, 0].hist(theta_flat, bins=50, edgecolor='black', alpha=0.7, density=True)
axs[1, 0].set_title('Theta Distribution')
axs[1, 0].set_xlabel('Theta')
axs[1, 0].set_ylabel('Density')

axs[1, 1].hist(position_flat, bins=np.arange(1, 22) - 0.5, edgecolor='black', al
axs[1, 1].set_title('Predicted Position Distribution')
axs[1, 1].set_xlabel('Position')
axs[1, 1].set_ylabel('Density')
axs[1, 1].set_xticks(range(1, 21))
axs[1, 1].set_xlim([0.5, 20.5])

plt.tight_layout()
plt.show()
```



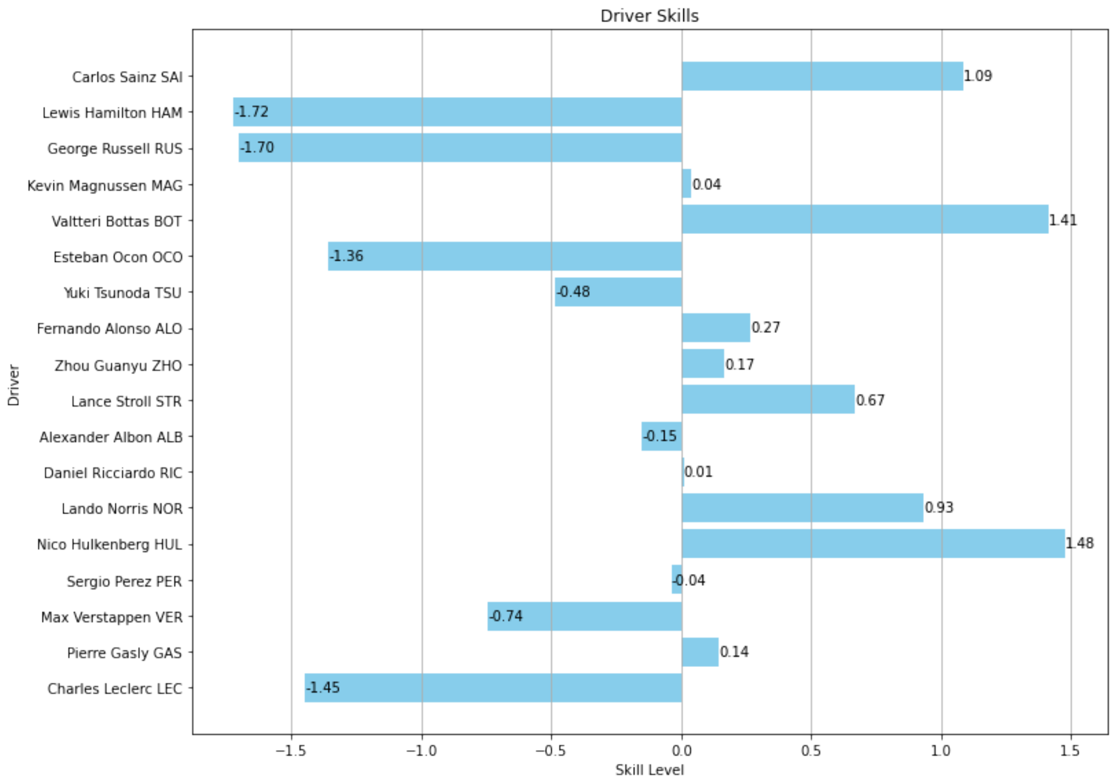
These distributions reflect the estimated variability and uncertainty in the model parameters, such as driver skill and car performance.
The peaks and spreads in the histograms indicate how the parameters are distributed across different races, with some parameters showing more concentrated distributions and others exhibiting broader variability.
Overall, the distributions provide insight into the range and tendencies of the model parameters, highlighting where the model finds consistency and where there is more uncertainty or variability in driver and car effects.

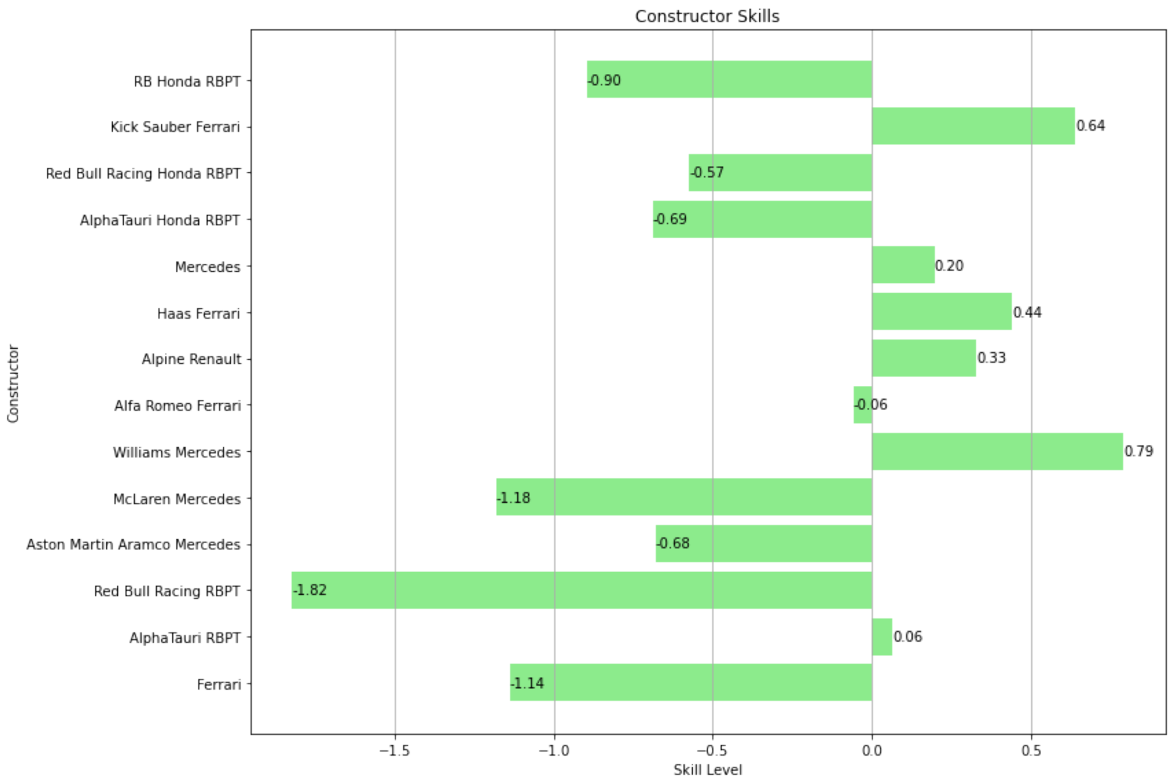
5.4. Results for individual drivers and constructors

In [316…

```
driver_skill_model1 = posterior_model_1.stan_variable('driver_skill')
constructor_skill_model1 = posterior_model_1.stan_variable('constructor_skill')

unique_driver_names = df['Driver'].unique()
unique_constructor_names = df['Car'].unique()

mean_driver_skill_model1 = np.mean(driver_skill_model1, axis=0)
mean_constructor_skill_model1 = np.mean(constructor_skill_model1, axis=0)

fig, axs = plt.subplots(2, 1, figsize=(12, 16))

bars = axs[0].barh(unique_driver_names, mean_driver_skill_model1, color='skyblue
axs[0].set_xlabel('Skill Level')
axs[0].set_ylabel('Driver')
axs[0].set_title('Driver Skills')
axs[0].grid(axis='x')
for bar in bars:
```

```
    axs[0].text(bar.get_width(), bar.get_y() + bar.get_height()/2, f'{bar.get_wi
                va='center', ha='left', color='black')


bars = axs[1].barh(unique_constructor_names, mean_constructor_skill_model1, colo
axs[1].set_xlabel('Skill Level')
axs[1].set_ylabel('Constructor')
axs[1].set_title('Constructor Skills')
axs[1].grid(axis='x')
for bar in bars:
    axs[1].text(bar.get_width(), bar.get_y() + bar.get_height()/2, f'{bar.get_wi
                va='center', ha='left', color='black')

plt.tight_layout()
plt.show()
```





As expected, several drivers (Lewis Hamilton, George Russell, Charles Leclerc) have a highest skill coefficient. A slightly lower coefficient has Max Verstappen that is the three-time world champion, the absolute dominator in the current era, in his case the model performed mediocre, because despite winning more than 60% of the races between 2022 and 2024 he occasionally finished 5th-7th and at the end of the grid, which had a big impact on the model's prediction. The highest coefficients for the constructor were obtained by Red Bull Racing RBPT, McLaren Mercedes and Ferrari, which is true, in the case of Mercedes the coefficient is low, which is also positive, because the drivers of this team achieved high positions because of their skills.

6. Posterior analysis (model 2)

6.1. Issues with the sampling

There were no issues. All samplings were conducted without any obstacles.

6.2. Analysis of samples from the posterior predictive distribution

In [317…

```
df = pd.read_csv('data_processing/processed_data.csv')
unique_drivers = df['Driver'].nunique()
unique_constructors = df['Car'].nunique()
drivers_indices = pd.factorize(df['Driver'])[0] + 1
constructors_indices = pd.factorize(df['Car'])[0] + 1
df['Pos (race)'] = pd.to_numeric(df['Pos (race)'], errors='coerce')
df['Pos (race)'] = df['Pos (race)'].astype(int) - 1
positions = df['Pos (race)'].values
rainy = df['TotalRainfall'].astype(bool).astype(int).values
data = {
    'N': len(df),
    'D': unique_drivers,
    'C': unique_constructors,
    'drivers': drivers_indices,
    'constructors': constructors_indices,
    'position': positions,
    'rainy': rainy,
}
model = CmdStanModel(stan_file='stan/model2_posterior.stan')
posterior_model_2 = model.sample(data=data, iter_warmup=1, iter_sampling=1000, c
```

```
INFO:cmdstanpy:found newer exe file, not recompiling
INFO:cmdstanpy:CmdStan start processing
chain 1 |██████████| 00:01 Sampling completed
```

```
INFO:cmdstanpy:CmdStan done processing.
```

In [318…

```
positions_predicted_model2 = posterior_model_2.stan_variable('position_predicted
plot_comparision_between_simulated_and_observed_for_drivers(positions_predicted_
```

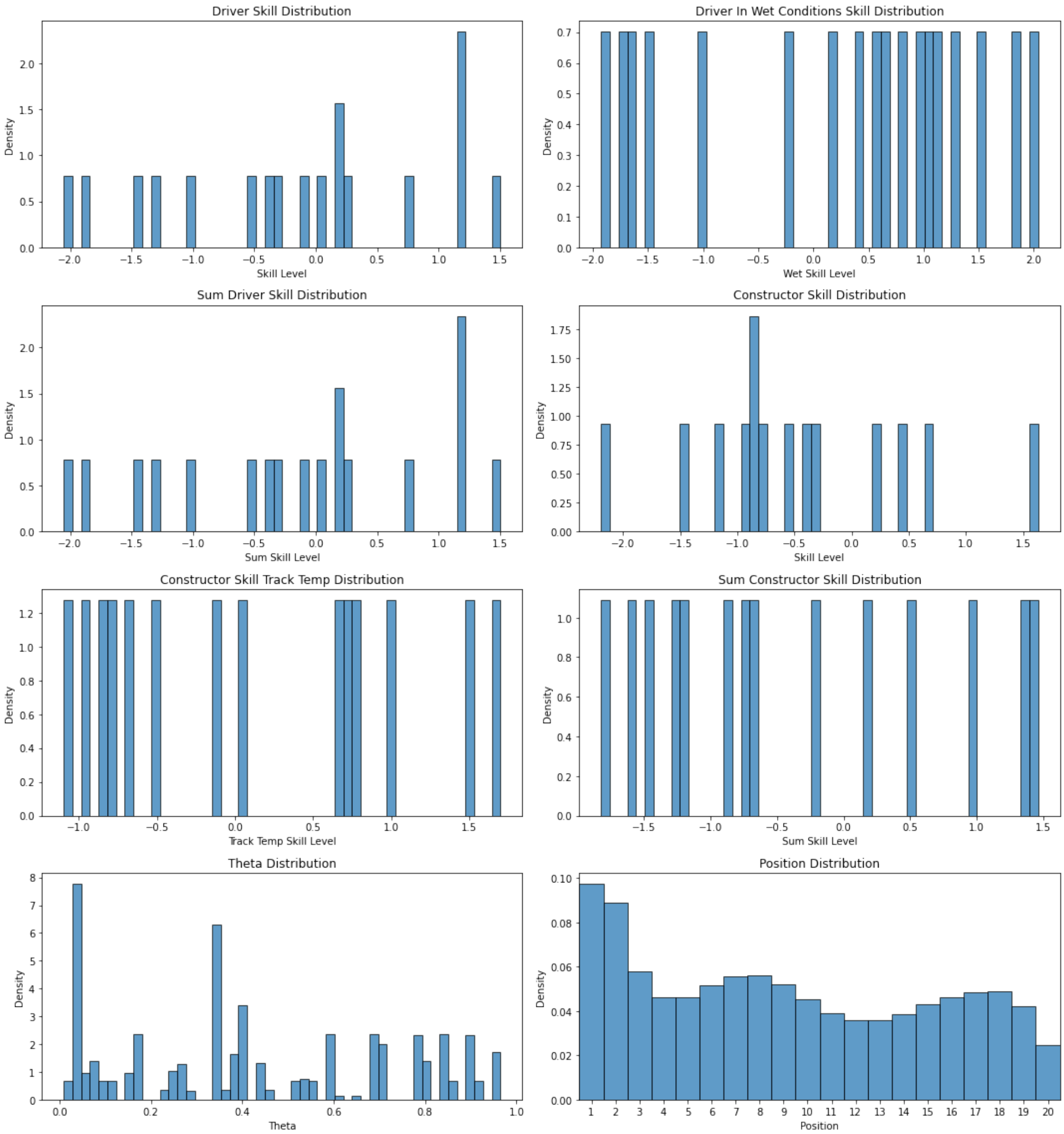
The simulated distributions (in blue) capture the trends and variability seen in the observed data (in red), though there are noticeable mismatches in specific bins. Some races exhibit good agreement between observed and simulated distributions, while others show discrepancies, indicating potential areas where the model might underperform or require refinement. Overall, the model reasonably approximates driver finishing positions, but further tuning or additional factors might be needed to enhance its predictive accuracy

6.3. Analysis of parameter marginal distributions

In [319…

```
driver_skill_flat = posterior_model_2.stan_variable('driver_skill').flatten()
driver_skill_wet_flat = posterior_model_2.stan_variable('driver_skill_wet').flat
```

```
driver_skill_sum_flat = posterior_model_2.stan_variable('driver_skill_sum').flat
constructor_skill_flat = posterior_model_2.stan_variable('constructor_skill').fl
constructor_skill_flat_track_temp = posterior_model_2.stan_variable('constructor
constructor_skill_flat_sum = posterior_model_2.stan_variable('constructor_skill_
position_flat = posterior_model_2.stan_variable('position_predicted').flatten()
theta_flat = posterior_model_2.stan_variable('theta').flatten()


fig, axs = plt.subplots(4, 2, figsize=(15, 16))

axs[0, 0].hist(driver_skill_flat, bins=50, edgecolor='black', alpha=0.7, density
axs[0, 0].set_title('Driver Skill Distribution')
axs[0, 0].set_xlabel('Skill Level')
axs[0, 0].set_ylabel('Density')

axs[0, 1].hist(driver_skill_wet_flat, bins=50, edgecolor='black', alpha=0.7, den
axs[0, 1].set_title('Driver In Wet Conditions Skill Distribution')
axs[0, 1].set_xlabel('Wet Skill Level')
axs[0, 1].set_ylabel('Density')

axs[1, 0].hist(driver_skill_sum_flat, bins=50, edgecolor='black', alpha=0.7, den
axs[1, 0].set_title('Sum Driver Skill Distribution')
axs[1, 0].set_xlabel('Sum Skill Level')
axs[1, 0].set_ylabel('Density')

axs[1, 1].hist(constructor_skill_flat, bins=50, edgecolor='black', alpha=0.7, de
axs[1, 1].set_title('Constructor Skill Distribution')
axs[1, 1].set_xlabel('Skill Level')
axs[1, 1].set_ylabel('Density')

axs[2, 0].hist(constructor_skill_flat_track_temp, bins=50, edgecolor='black', al
axs[2, 0].set_title('Constructor Skill Track Temp Distribution')
axs[2, 0].set_xlabel('Track Temp Skill Level')
axs[2, 0].set_ylabel('Density')

axs[2, 1].hist(constructor_skill_flat_sum, bins=50, edgecolor='black', alpha=0.7
axs[2, 1].set_title('Sum Constructor Skill Distribution')
axs[2, 1].set_xlabel('Sum Skill Level')
axs[2, 1].set_ylabel('Density')

axs[3, 0].hist(theta_flat, bins=50, edgecolor='black', alpha=0.7, density=True)
axs[3, 0].set_title('Theta Distribution')
axs[3, 0].set_xlabel('Theta')
axs[3, 0].set_ylabel('Density')

axs[3, 1].hist(position_flat, bins=np.arange(1, 22) - 0.5, edgecolor='black', al
axs[3, 1].set_title('Position Distribution')
axs[3, 1].set_xlabel('Position')
axs[3, 1].set_ylabel('Density')
axs[3, 1].set_xticks(range(1, 21))
axs[3, 1].set_xlim([0.5, 20.5])

plt.tight_layout()
plt.show()
```



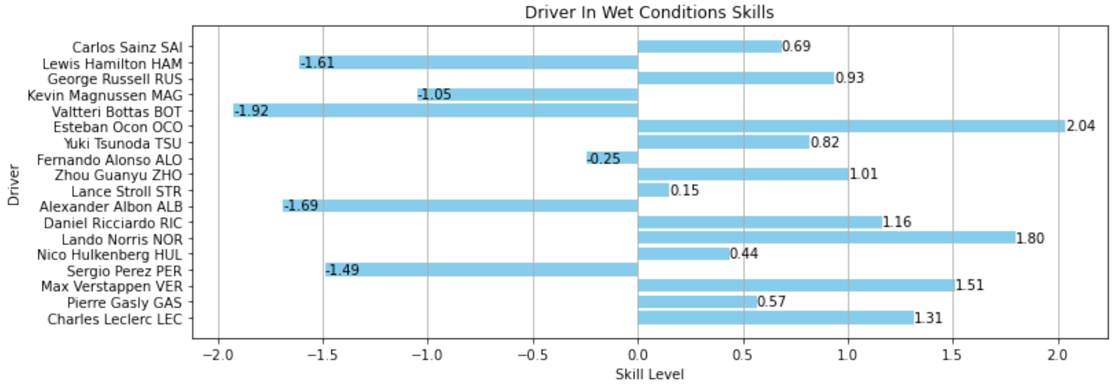
Incorporating basic model parameters associated with track temperature and rainfall has expanded the range of θ values and significantly altered the position distribution. Analysis indicates that this adjustment has positively impacted the model's performance.

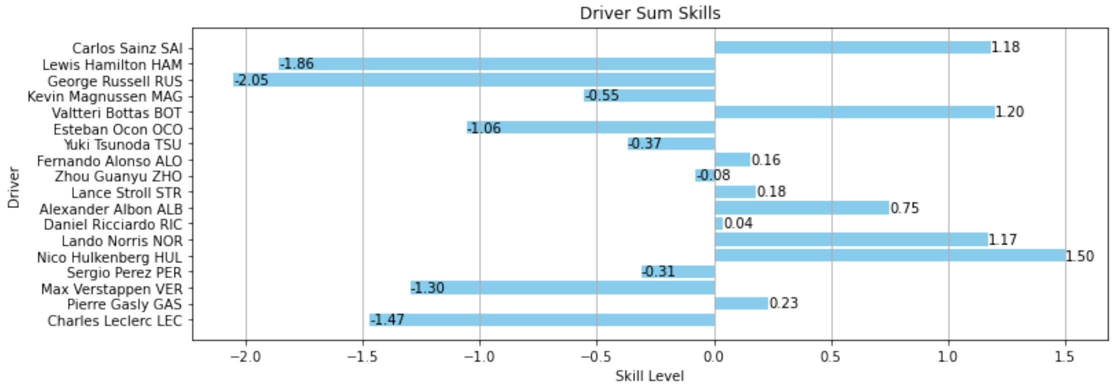
6.4. Results for individual drivers and constructors

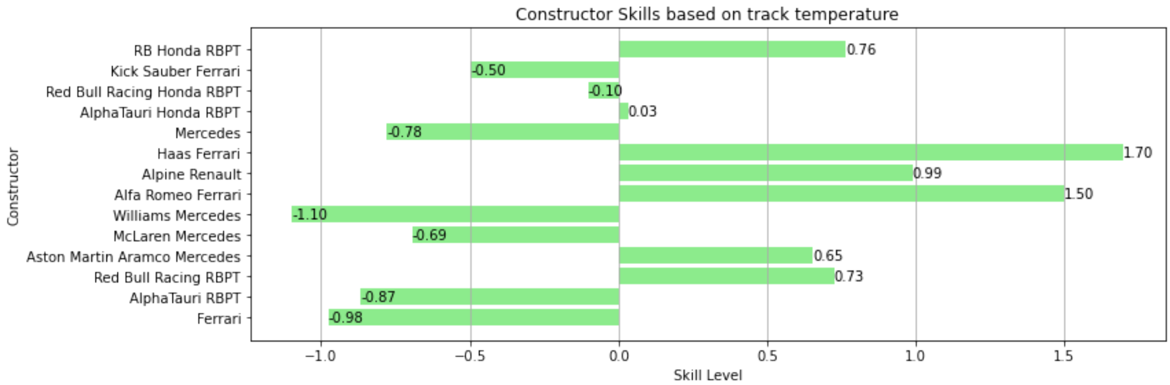
In [320…

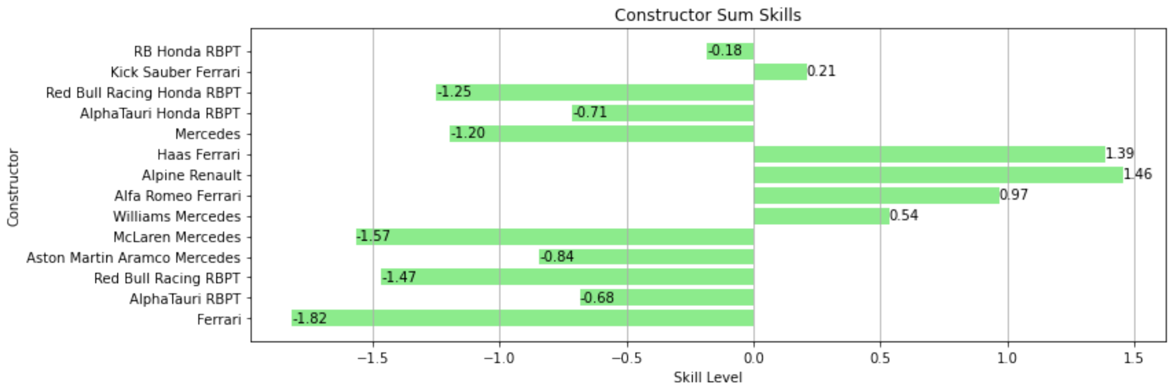
```
driver_skill_sum_wet_model2 = posterior_model_2.stan_variable('driver_skill_wet'
driver_skill_model2_sum = posterior_model_2.stan_variable('driver_skill_sum')
constructor_skill_model2_track_temp = posterior_model_2.stan_variable('construct
constructor_skill_model2_sum = posterior_model_2.stan_variable('constructor_skil

unique_driver_names = df['Driver'].unique()
unique_constructor_names = df['Car'].unique()

mean_driver_skill_wet_model2 = np.mean(driver_skill_sum_wet_model2, axis=0)
mean_driver_skill_model2_sum = np.mean(driver_skill_model2_sum, axis=0)
mean_constructor_skill_model2_track_temp = np.mean(constructor_skill_model2_trac
mean_constructor_skill_model2_sum = np.mean(constructor_skill_model2_sum, axis=0
```

```
fig, axs = plt.subplots(4, 1, figsize=(12, 16))

bars = axs[0].barh(unique_driver_names, mean_driver_skill_wet_model2, color='sky
axs[0].set_xlabel('Skill Level')
axs[0].set_ylabel('Driver')
axs[0].set_title('Driver In Wet Conditions Skills')
axs[0].grid(axis='x')
for bar in bars:
    axs[0].text(bar.get_width(), bar.get_y() + bar.get_height()/2, f'{bar.get_wi
                va='center', ha='left', color='black')


bars = axs[1].barh(unique_driver_names, mean_driver_skill_model2_sum, color='sky
axs[1].set_xlabel('Skill Level')
axs[1].set_ylabel('Driver')
axs[1].set_title('Driver Sum Skills')
axs[1].grid(axis='x')
for bar in bars:
    axs[1].text(bar.get_width(), bar.get_y() + bar.get_height()/2, f'{bar.get_wi
                va='center', ha='left', color='black')


bars = axs[2].barh(unique_constructor_names, mean_constructor_skill_model2_track
axs[2].set_xlabel('Skill Level')
axs[2].set_ylabel('Constructor')
axs[2].set_title('Constructor Skills based on track temperature')
axs[2].grid(axis='x')
for bar in bars:
    axs[2].text(bar.get_width(), bar.get_y() + bar.get_height()/2, f'{bar.get_wi
                va='center', ha='left', color='black')


bars = axs[3].barh(unique_constructor_names, mean_constructor_skill_model2_sum,
axs[3].set_xlabel('Skill Level')
axs[3].set_ylabel('Constructor')
axs[3].set_title('Constructor Sum Skills')
axs[3].grid(axis='x')
for bar in bars:
    axs[3].text(bar.get_width(), bar.get_y() + bar.get_height()/2, f'{bar.get_wi
                va='center', ha='left', color='black')

plt.tight_layout()
plt.show()
```









As with the first model, it appears that the same drivers as Leclerc, Russel, Hamilton, Verstappen, Ocon have the highest skill factor. It is noteworthy that only Hamilton of these drivers has a high skill coefficient in the rain, which means that the rest perform best in dry track conditions. For constructors, the situation is similar, the same three teams are the best (Ferrari, Red Bull Racing RBPT and McLaren Mercedes), but when we take into account the constructor's adaptation to the track temperature Mercedes has by far the better coefficient for the second model.

7. Posterior analysis (model 3)

7.1. Issues with the sampling

There were no issues. All samplings were conducted without any obstacles.

7.2. Analysis of samples from the posterior predictive distribution

In [321…

```
df = pd.read_csv('data_processing/processed_data.csv')
unique_drivers = df['Driver'].nunique()
unique_constructors = df['Car'].nunique()
drivers_indices = pd.factorize(df['Driver'])[0] + 1
constructors_indices = pd.factorize(df['Car'])[0] + 1
df['Pos (race)'] = df['Pos (race)'] - 1
positions = df['Pos (race)'].values
rainy = df['TotalRainfall'].astype(bool).values.astype(int)
avg_track_temp = df['AvgTrackTemp'].values
qualifying_positions = df['Pos (qualifying)'].values
data = {
    'N': len(df),
    'D': unique_drivers,
    'C': unique_constructors,
    'drivers': drivers_indices,
    'constructors': constructors_indices,
    'position': positions,
    'rainy': rainy,
    'qualifying_position': qualifying_positions
}
model = CmdStanModel(stan_file='stan/model3_posterior.stan')
posterior_model_3 = model.sample(data=data, iter_warmup=1, iter_sampling=1000, c
```

```
INFO:cmdstanpy:found newer exe file, not recompiling
INFO:cmdstanpy:CmdStan start processing
chain 1 |██████████| 00:02 Sampling completed
```

```
INFO:cmdstanpy:CmdStan done processing.
```

In [322…

```
positions_predicted_model3 = posterior_model_3.stan_variable('position_predicted
plot_comparision_between_simulated_and_observed_for_drivers(positions_predicted_
```

The posterior predictive distributions from Model 3 show improved alignment with the observed race results compared to Model 1 and 2, capturing both central tendencies and variability more accurately.

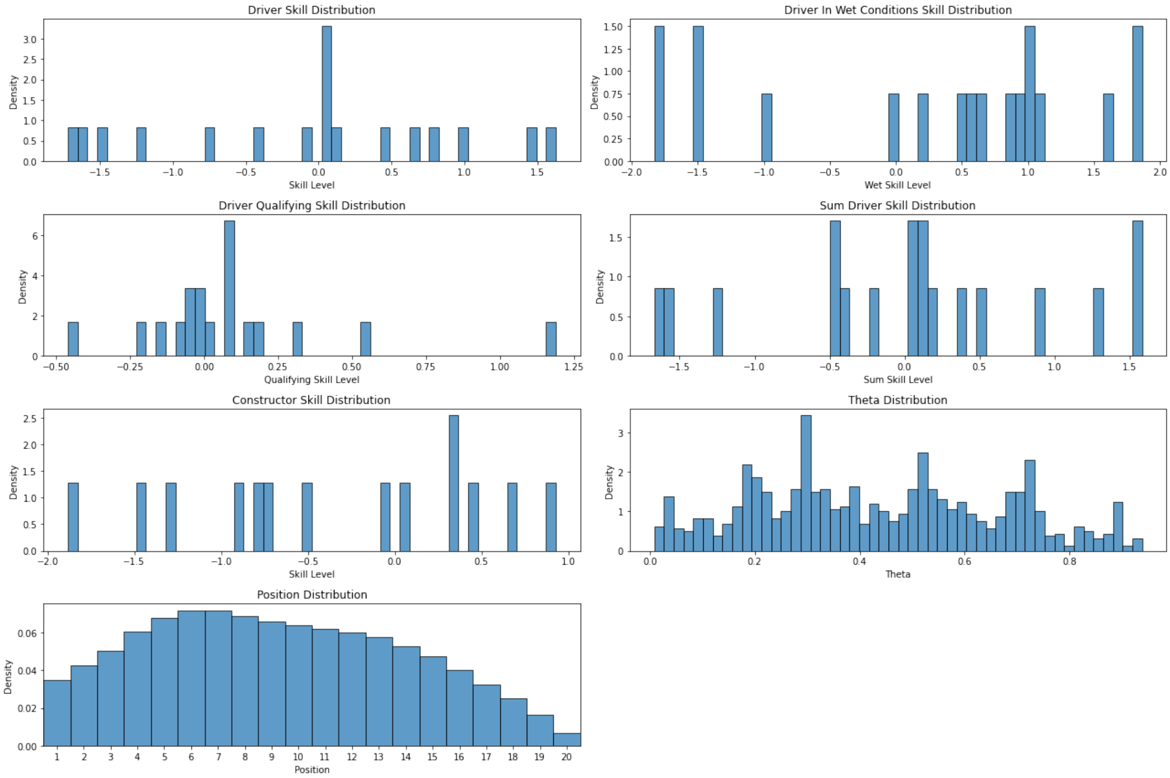
This model, which incorporates qualifying positions, accounts for key factors affecting race outcomes, resulting in better simulation of peak positions and spread. However, while Model 3 generally reduces bias, there are still occasional discrepancies in capturing extreme values and the precise shapes of some distributions.

Overall, Model 3 offers a more comprehensive fit, reflecting the importance of qualifying performance in determining race results.

7.3. Analysis of parameter marginal distributions

In [323…

```
driver_skill_flat = posterior_model_3.stan_variable('driver_skill').flatten()
driver_skill_wet_flat = posterior_model_3.stan_variable('driver_skill_wet').flat
driver_skill_qualifying = posterior_model_3.stan_variable('driver_skill_qualifyi
driver_skill_sum_flat = posterior_model_3.stan_variable('driver_skill_sum').flat
constructor_skill_flat = posterior_model_3.stan_variable('constructor_skill').fl
position_flat = posterior_model_3.stan_variable('position_predicted').flatten()
theta_flat = posterior_model_3.stan_variable('theta').flatten()

fig, axs = plt.subplots(4, 2, figsize=(18, 12))

axs[0, 0].hist(driver_skill_flat, bins=50, edgecolor='black', alpha=0.7, density
axs[0, 0].set_title('Driver Skill Distribution')
axs[0, 0].set_xlabel('Skill Level')
axs[0, 0].set_ylabel('Density')

axs[0, 1].hist(driver_skill_wet_flat, bins=50, edgecolor='black', alpha=0.7, den
axs[0, 1].set_title('Driver In Wet Conditions Skill Distribution')
axs[0, 1].set_xlabel('Wet Skill Level')
axs[0, 1].set_ylabel('Density')

axs[1, 0].hist(driver_skill_qualifying, bins=50, edgecolor='black', alpha=0.7, d
axs[1, 0].set_title('Driver Qualifying Skill Distribution')
axs[1, 0].set_xlabel('Qualifying Skill Level')
axs[1, 0].set_ylabel('Density')

axs[1, 1].hist(driver_skill_sum_flat, bins=50, edgecolor='black', alpha=0.7, den
axs[1, 1].set_title('Sum Driver Skill Distribution')
axs[1, 1].set_xlabel('Sum Skill Level')
axs[1, 1].set_ylabel('Density')

axs[2, 0].hist(constructor_skill_flat, bins=50, edgecolor='black', alpha=0.7, de
axs[2, 0].set_title('Constructor Skill Distribution')
axs[2, 0].set_xlabel('Skill Level')
axs[2, 0].set_ylabel('Density')

axs[2, 1].hist(theta_flat, bins=50, edgecolor='black', alpha=0.7, density=True)
axs[2, 1].set_title('Theta Distribution')
axs[2, 1].set_xlabel('Theta')
axs[2, 1].set_ylabel('Density')

axs[3, 0].hist(position_flat, bins=np.arange(1, 22) - 0.5, edgecolor='black', al
axs[3, 0].set_title('Position Distribution')
axs[3, 0].set_xlabel('Position')
axs[3, 0].set_ylabel('Density')
axs[3, 0].set_xticks(range(1, 21))
axs[3, 0].set_xlim([0.5, 20.5])

fig.delaxes(axs[3, 1])
plt.tight_layout()
plt.show()
```



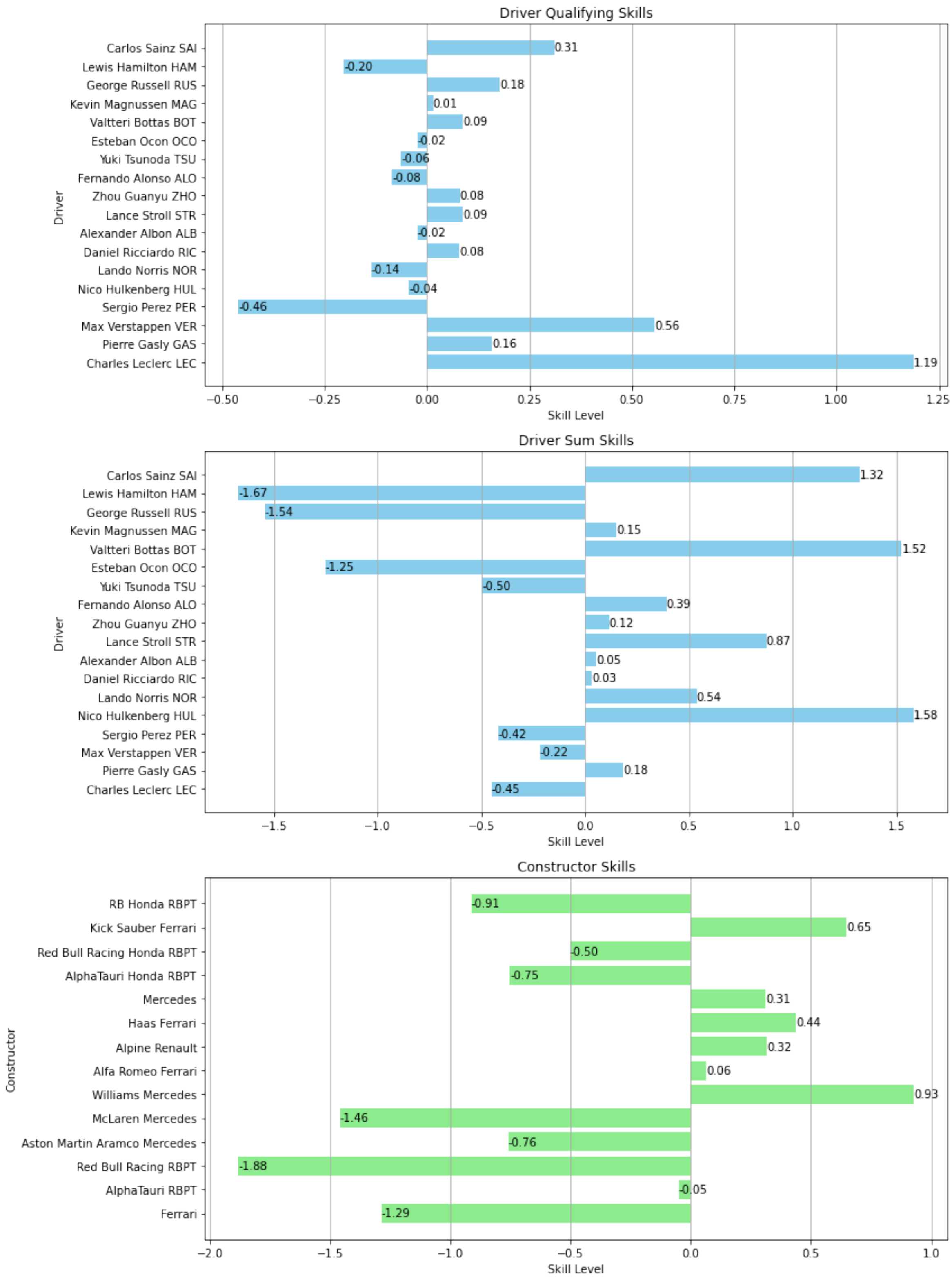
The inclusion of qualifying performance and the passing ratio has resulted in a more varied set of parameter distributions. Parameters related to qualifying, such as α_{quals}, display notable peaks and broader distributions, suggesting significant variability in the model's consideration of qualifying positions across races. This model provides a refined perspective, enhancing our understanding of the complex interplay between a driver's starting position and other race factors, thereby offering a more detailed assessment of their impact on race outcomes.

7.4. Results for individual drivers and constructors

In [324…

```
driver_skill_model3 = posterior_model_3.stan_variable('driver_skill_sum')
driver_skill_model3_qualifying = posterior_model_3.stan_variable('driver_skill_q
constructor_skill_model3 = posterior_model_3.stan_variable('constructor_skill')
unique_driver_names = df['Driver'].unique()
unique_constructor_names = df['Car'].unique()
mean_driver_skill_model3 = np.mean(driver_skill_model3, axis=0)
mean_driver_skill_model3_qualifying = np.mean(driver_skill_model3_qualifying, ax
mean_constructor_skill_model3 = np.mean(constructor_skill_model3, axis=0)

fig, axs = plt.subplots(3, 1, figsize=(12, 16))

bars = axs[0].barh(unique_driver_names, mean_driver_skill_model3_qualifying, col
axs[0].set_xlabel('Skill Level')
axs[0].set_ylabel('Driver')
axs[0].set_title('Driver Qualifying Skills')
axs[0].grid(axis='x')
for bar in bars:
    axs[0].text(bar.get_width(), bar.get_y() + bar.get_height()/2, f'{bar.get_wi
                va='center', ha='left', color='black')


bars = axs[1].barh(unique_driver_names, mean_driver_skill_model3, color='skyblue
axs[1].set_xlabel('Skill Level')
```

```
axs[1].set_ylabel('Driver')
axs[1].set_title('Driver Sum Skills')
axs[1].grid(axis='x')
for bar in bars:
    axs[1].text(bar.get_width(), bar.get_y() + bar.get_height()/2, f'{bar.get_wi
                va='center', ha='left', color='black')

bars = axs[2].barh(unique_constructor_names, mean_constructor_skill_model3, colc
axs[2].set_xlabel('Skill Level')
axs[2].set_ylabel('Constructor')
axs[2].set_title('Constructor Skills')
axs[2].grid(axis='x')
for bar in bars:
    axs[2].text(bar.get_width(), bar.get_y() + bar.get_height()/2, f'{bar.get_wi
                va='center', ha='left', color='black')

plt.tight_layout()
plt.show()
```



Model 3 contains additional information regarding drivers' performance in qualifying. As before, Russel and Hamitlon have high skill coefficients, but for Verstappen and Leclerc it is lower than in earlier models because of a qualifying coefficient. For the constructors, again the same 3 teams are the best (Red Bull Racing RBPT, McLaren Mercedes and Ferrari), and the coefficient for Mercedes is similar to that in Model 1, as we did not take into account adaptation to track temperature.

8. Model comaprison

8.1. General comparison

In [325…

```
def plot_position_comparison(first_model_to_compare, second_model_to_compare, df
    n_rows, n_cols = 6, 3
    fig, axes = plt.subplots(n_rows, n_cols, figsize=(8*n_cols, 5*n_rows))

    n_bins = np.arange(22) - 0.5
    drivers_names = df['Driver'].unique().tolist()

    for driver_index, driver_name in enumerate(drivers_names):
        specified_driver = df['Driver'].eq(driver_name)
        results = df[specified_driver]

        row_idx = driver_index // n_cols
        col_idx = driver_index % n_cols

        axes[row_idx, col_idx].hist((results['Pos (race)']).tolist(), bins=n_bin

        results_idx = results.index
        simulated_positions_model1 = first_model_to_compare.T[results_idx].flatt
        simulated_positions_model2 = second_model_to_compare.T[results_idx].flat

        if np.array_equal(first_model_to_compare, positions_predicted_model1):
            model1_label = 'Simulated Model 1'
        elif np.array_equal(first_model_to_compare, positions_predicted_model2):
            model1_label = 'Simulated Model 2'
        elif np.array_equal(first_model_to_compare, positions_predicted_model3):
            model1_label = 'Simulated Model 3'

        if np.array_equal(second_model_to_compare, positions_predicted_model1):
            model2_label = 'Simulated Model 1'
        elif np.array_equal(second_model_to_compare, positions_predicted_model2)
            model2_label = 'Simulated Model 2'
        elif np.array_equal(second_model_to_compare, positions_predicted_model3)
            model2_label = 'Simulated Model 3'

        axes[row_idx, col_idx].hist(simulated_positions_model1, bins=n_bins, rwi
        axes[row_idx, col_idx].hist(simulated_positions_model2, bins=n_bins, rwi
        axes[row_idx, col_idx].set_xticks(range(22))
        axes[row_idx, col_idx].set_xlim([0, 21])
        axes[row_idx, col_idx].set_yticks([])
        axes[row_idx, col_idx].set_title(driver_name.split()[-1] + ' Finishing P
        axes[row_idx, col_idx].legend()
        axes[row_idx, col_idx].set_xlabel('Position')

    fig.tight_layout()
    plt.show()
```

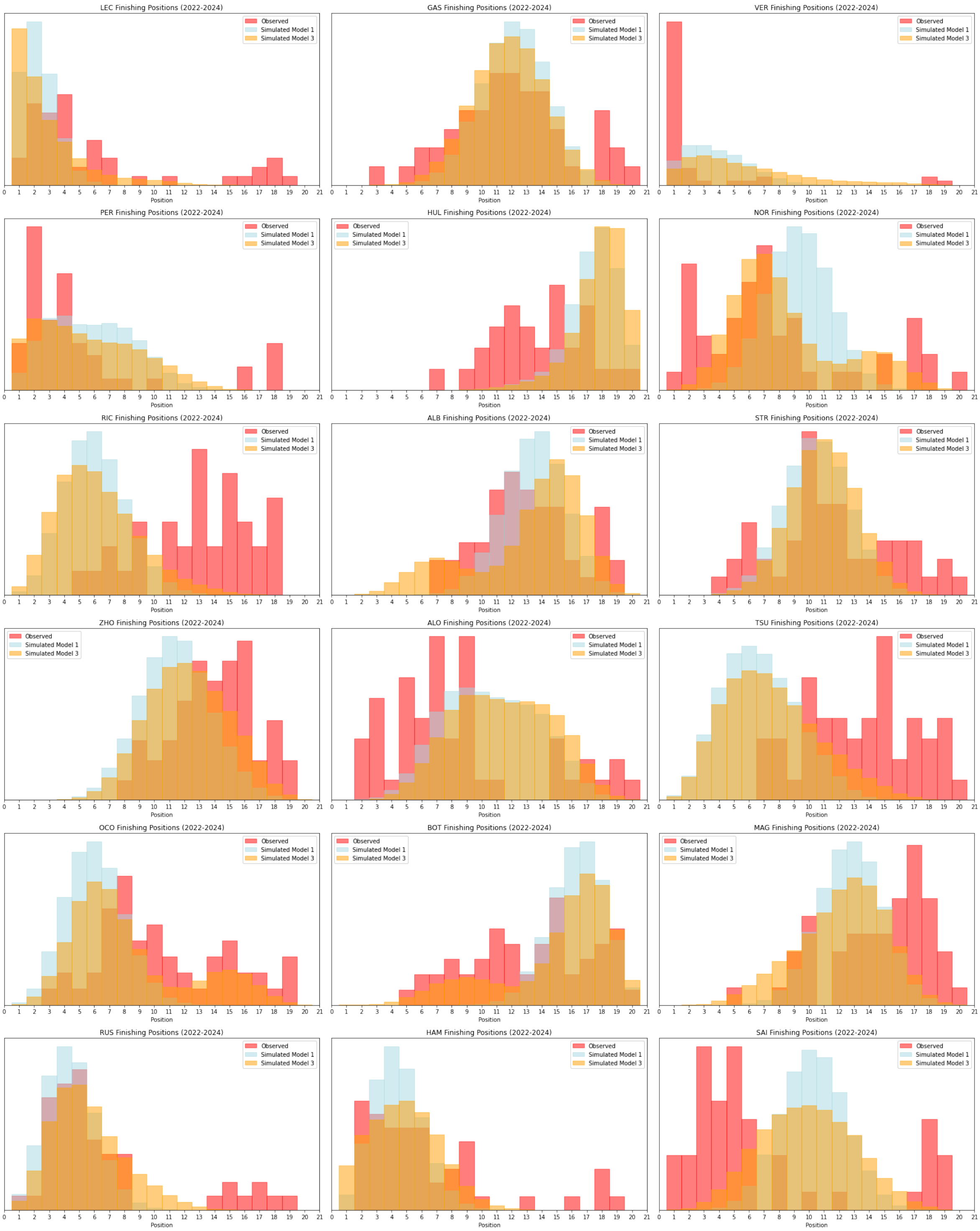
In [326…

```
df = pd.read_csv('data_processing/processed_data.csv')
plot_position_comparison(positions_predicted_model1, positions_predicted_model2,
```

Comparing models 1 and 2, it can be observed that model 2 has a slight advantage in most cases. Unfortunately, both models struggle with drivers who usually achieve high placements but occasionally finish in lower positions, as seen with LEC, VER, RUS, and HAM.

In [327…

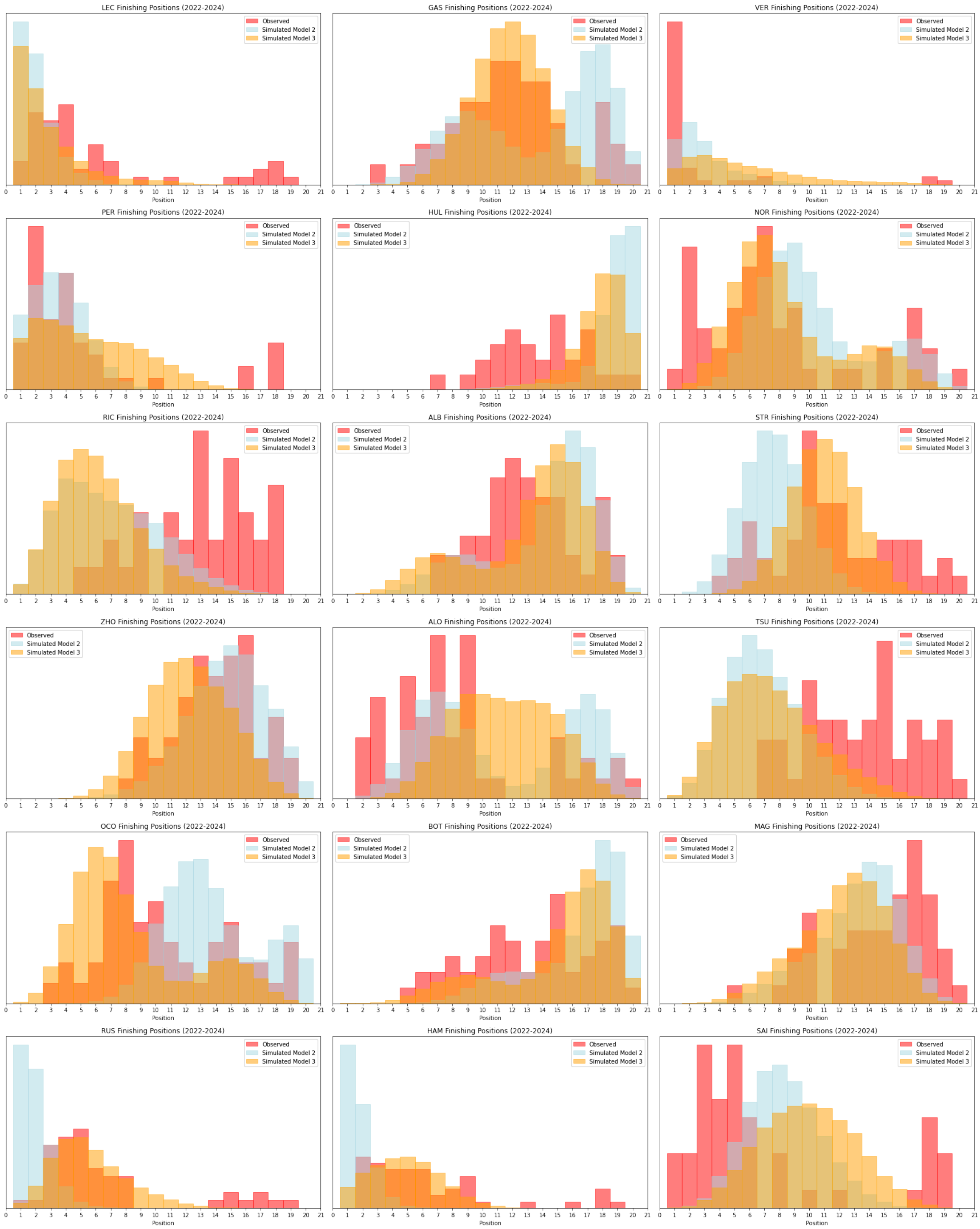
```
plot_position_comparison(positions_predicted_model1, positions_predicted_model3,
```



Although Models 1 and 3 mostly overlap, a slight advantage of Model 3 can be observed. The values are often more stretched along the X-axis, which contains positions, and this mostly better reflects the behavior of the data.

In [328…

```
plot_position_comparison(positions_predicted_model2, positions_predicted_model3,
```



When comparing Models 2 and 3, there are instances where each model performs better or worse, often with overlapping results. However, Model 3 appears to be superior because it avoids significant errors that Model 2 makes, such as for drivers RUS and HAM.

8.2. Comparison of individual drivers and constructors

8.2.1 Drivers

In [329…

```
unique_driver_names = df['Driver'].unique()
models = ['Model 1', 'Model 2', 'Model 3']

fig, axs = plt.subplots(6, 3, figsize=(18, 20))
```

```python
for i, driver in enumerate(unique_driver_names):
    value_model1 = mean_driver_skill_model1[i]
    value_model2 = mean_driver_skill_model2_sum[i]
    value_model3 = mean_driver_skill_model3[i]

    axs[i // 3, i % 3].barh(models, [value_model1, value_model2, value_model3],
                            color=['skyblue', 'lightgreen', 'salmon'])
    axs[i // 3, i % 3].set_ylabel('Skill Level')
    axs[i // 3, i % 3].set_title(f'{driver}')
    axs[i // 3, i % 3].grid(axis='y')

plt.tight_layout()
plt.show()
```

For most drivers, the skill coefficients calculated using the three models are approximately equal. We can observe larger differences for Leclerc, Verstappen, Ricciardo, Stroll, Norris, Magnussen. Most likely, they result from the use of additional parameters such as weather or qualification data.

8.2.2 Constructors

In [330…

```
unique_constructor_names = df['Car'].unique()
models = ['Model 1', 'Model 2', 'Model 3']

fig, axs = plt.subplots(7, 2, figsize=(18, 20))
for i, constructor in enumerate(unique_constructor_names):
    value_model1 = mean_constructor_skill_model1[i]
    value_model2 = mean_constructor_skill_model2_sum[i]
    value_model3 = mean_constructor_skill_model3[i]

    axs[i // 2, i % 2].barh(models, [value_model1, value_model2, value_model3],
                            color=['skyblue', 'lightgreen', 'salmon'])
    axs[i // 2, i % 2].set_ylabel('Skill Level')
    axs[i // 2, i % 2].set_title(f'{constructor}')
    axs[i // 2, i % 2].grid(axis='y')

plt.tight_layout()
plt.show()
```

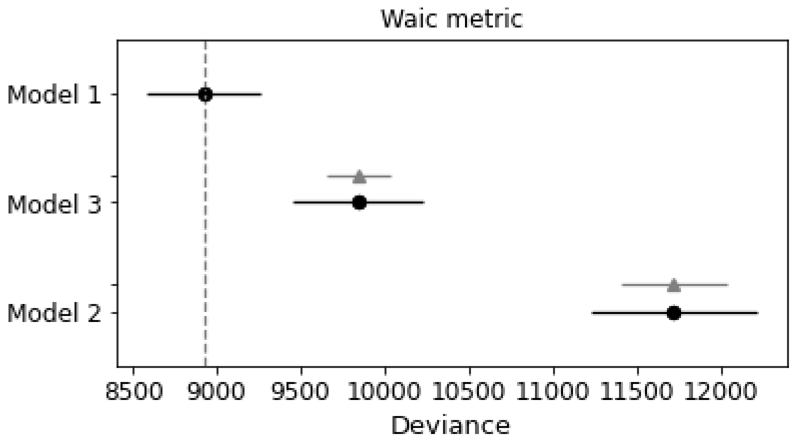
The constructor coefficients calculated in models 1 and 2 are approximately equal. However, differences can be noticed compared to model 2, because in this model an adaptation factor to track temperature has been added.

8.3. WAIC and PSIS-LOO results

8.3.1 WAIC

In [331…

```
comp_dict = {'Model 1': posterior_model_1, 'Model 2': posterior_model_2, 'Model
comp_waic = az.compare(comp_dict, ic='waic', scale='deviance')
az.plot_compare(comp_waic)
plt.title("Waic metric")
plt.show()
```



In [332…

```
comp_waic
```

Out[332…

	rank	waic	p_waic	d_waic	weight	se	dse
Model 1	0	8927.545420	2.713238e-27	0.000000	0.490992	338.905364	0.000000
Model 3	1	9844.382604	3.357910e-27	916.837184	0.233588	388.514099	193.457074
Model 2	2	11721.272168	5.968140e-27	2793.726748	0.275420	494.925636	314.714588

No errors occurred when comparing models using the Waic criterion. The black dot on the plot represents the WAIC value for each model, lower value indicate better predictive performance. The gray triangle represents the standard error of the WAIC estimate, this show the uncertainity around the WAIC value, smaller standard error indicated more precise estimates. The horizontal black line through black dot represents the confidence interval for the WAIC estimate. This interval shows the range within the true WAIC value is expected to lie.

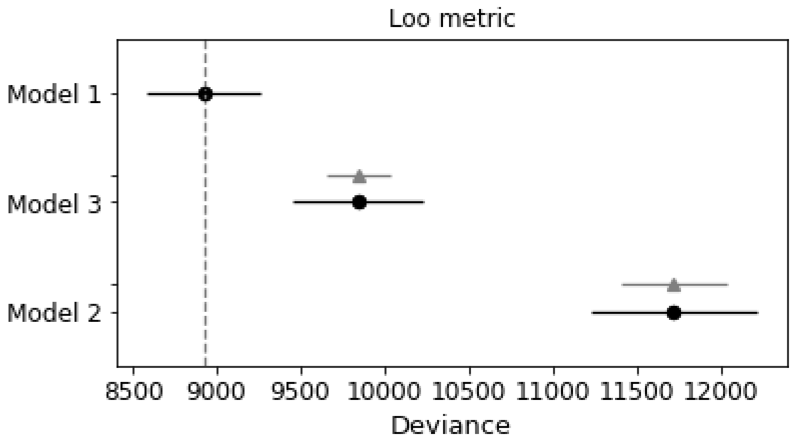
Model 1 should be considered the primary model due to its best WAIC value (waic column in dataframe above) and higher relative weight, indicating the best predictive accuracy and model fit. Model 3 is the second-best, can be considered as an alternative, but its significantly higher WAIC suggests that it is less optimal compared to Model 1. Model 2 is the least credbile indicated by the highest WAIC. There is no overlap between Model 1 and 3 based on Waic.

8.3.2 PSIS-LOO

In [333...

```
comp_dict = {'Model 1': posterior_model_1, 'Model 2': posterior_model_2, 'Model
comp_loo = az.compare(comp_dict, ic='loo', scale='deviance')
az.plot_compare(comp_loo)
plt.title("Loo metric")
plt.show()
```

```
/usr/local/lib/python3.12/site-packages/arviz/stats/stats.py:811: UserWarning: Es
timated shape parameter of Pareto distribution is greater than 0.7 for one or mor
e samples. You should consider using a more robust model, this is because importa
nce sampling is less likely to work well if the marginal posterior and LOO poster
ior are very different. This is more likely to happen with a non-robust model and
highly influential observations.
  warnings.warn(
/usr/local/lib/python3.12/site-packages/arviz/stats/stats.py:811: UserWarning: Es
timated shape parameter of Pareto distribution is greater than 0.7 for one or mor
e samples. You should consider using a more robust model, this is because importa
nce sampling is less likely to work well if the marginal posterior and LOO poster
ior are very different. This is more likely to happen with a non-robust model and
highly influential observations.
  warnings.warn(
/usr/local/lib/python3.12/site-packages/arviz/stats/stats.py:811: UserWarning: Es
timated shape parameter of Pareto distribution is greater than 0.7 for one or mor
e samples. You should consider using a more robust model, this is because importa
nce sampling is less likely to work well if the marginal posterior and LOO poster
ior are very different. This is more likely to happen with a non-robust model and
highly influential observations.
  warnings.warn(
```



In [334...

```
comp_loo
```

Out[334…

	rank	loo	p_loo	d_loo	weight	se	dse	warn
Model 1	0	8927.545420	0.0	0.000000	0.490992	338.905364	0.000000	T
Model 3	1	9844.382604	0.0	916.837184	0.233588	388.514099	193.457074	T
Model 2	2	11721.272168	0.0	2793.726748	0.275420	494.925636	314.714588	T

When comparing models using the Loo criterion, an error occurred suggests that current models may not be robust. Non-robust models are more sensitive to individual observations, and making LOO estimates less reliable.

The black dot on the plot represents the LOO value for each model, lower value indicate better predictive performance. The gray triangle represents the standard error of the LOO estimate, this show the uncertainity around the LOO value, smaller standard error indicated more precise estimates. The horizontal black line through black dot represents the confidence interval for the LOO estimate. This interval shows the range within the true LOO value is expected to lie.

Model 1 is the best performing model with the lowest LOO metric value (loo column in dataframe above) and the highest weight, indicating it is the most likely model. Model 3 is the second-best model but has a significantly higher LOO metric value and standard error compared to Model 1, reflecting a worse fit. Model 2 has the highest LOO and standard error. Despite having a weight similiar to Model 3, the high LOO value suggests it is the least credible model. There is no overlap between Model 1 and 3 based on Loo criterion.

Overall about models

Overall, the WAIC-based comparision aligns with the LOO results, reaffirming that Model 1 is the best model in terms of both fit and predictive accuracy. Model 1 is the simplest model, it contains the least number of parameters (only driver and constrcutor skill) and that means is less prone to overfitting, making it more reliable for prediction on new data by capturing the underlying patterns in the data more effectively. Model 2 is more complicated including additional paramaters regarding the dependence of driver skill on weather and constructor skill on track surface temperature. Based on our observations, track temperature has a very negative effect on a prediction. Due to the poor efect of track temperature on the prediction, we decided to remove it in model 3 and instead add a coefficient representing the driver's skill in qualifying which improved the model significantly compared to model 3, but is is still worse than model 1. After consideration, we agree with the results derived from the information criteria.

9. References

1. Formula 1. (n.d.). Official Formula 1 website. Retrieved June 15, 2024, from https://www.formula1.com/

2. The Formula 1 API Development Team (n.d.). Formula 1 API documentation. Retrieved June 15, 2024, from https://docs.fastf1.dev/

3. Stan Development Team. (n.d.). Stan documentation. Retrieved June 15, 2024, from https://mc-stan.org/users/documentation/

4. The Jupyter Development Team. (n.d.). Jupyter notebook documentation. Retrieved June 15, 2024, from https://docs.jupyter.org/en/latest/

5. van Kesteren, E.-J., & Bergkamp, T. (2023). Bayesian analysis of Formula One race results: disentangling driver skill and constructor advantage.

6. Python Software Foundation. (n.d.). Python documentation. Retrieved June 15, 2024, from https://docs.python.org/3.11