Predicting Formula One race results using Bayesian inference on a rank ordered logit model

John Walker

1 Abstract

In this paper we describe a model that can be used to predict the results of Formula One races in a season using results of previous races in that season. It proceeds by assigning a skill parameter to each driver that depends on both the driver and their team. Results of a race are then ordered draws from a specified distribution, with drivers with higher skills tending to be ranked higher in a race. The model is evaluated on how much sense it makes from a practical perspective (do the results it predict make realistic sense?) and how well it can predict new results (posterior predictive inference). The model is found to perform somewhat unsatisfactorily, especially since it predicts with high probability that some unrealistic results will occur.

2 Introduction

In this paper, we attempt to predict the results of a Formula One race using the results of previous races. That is, given the race results of N races, we want to predict the results of an $N + 1$st race. We do this by formulating a method that calculates for each driver i in the $N + 1$st race a skill number θ_i that measures how skilled they are. We predict that the driver with the highest skill parameter will win the race, followed by the driver with the second largest skill parameter, all the way down to the last. We will explain what we precisely mean by this in section 3. For the rest of this section, we will provide background into Formula One and why this problem is interesting, a literature review on similar work already done for this problem, and explain why we chose to use a rank ordered logit model (a type of Bayesian method). Particular emphasis in this paper is put on the 2023 Formula One season, since that season was underway and being watched by the author as the results in this paper were being developed. For this reason, the methods used in this paper are presented using data from the 2023 Formula One season.

2.1 Introduction to Formula One: races, drivers, and constructors

Formula One is a form of Formula racing. Formula means the set of rules that drivers and their cars must abide by, while the One means that it is the highest level of Formula racing [1]. Formula One racing takes place over seasons, labeled by year, taking place over around 8 months, and consisting of roughly 20 grand prix races (for example, in the 2023 season there were 22 grand prix races). Prior to 2021, the only races taking place were grand prix races. In 2021, a new race format was introduced, called a sprint race [1]. Grand prix races consist of drivers racing around a circuit for a set amount of laps, while sprint races are effectively short grand prix races (taking place over 100 km instead of a lap amount) [4]. Sprint races are done in conjunction with a handful of grand prix races, not replacing them. Specifically, grand prix races usually take place on Sunday's (in 2023 all grand prix races took place on a Sunday except

the Las Vegas grand prix which took place on a Saturday), while the sprint race (if it is done in conjunction with that race weekend's grand prix; in 2023 there were 6 sprint races) would take place on Saturday.

In this paper we only focus on grand prix races (usually just called a race instead of grand prix race) and ignore sprint races. The only time sprint races come into our analysis is when we compare our obtained results to driver points. In modern Formula One, points are awarded to the top 10 grand prix race winners, with the 1st place winner receiving 25 points and the 10th place winner receiving 1 point. In sprint races, points are awarded to the top 8 race winners, with the 1st place winner receiving 8 points and the 8th place winner receiving 1 point. Besides sprint races awarding much fewer points compared to grand prixs, they are heavily criticized by drivers for not being the right type of challenge for drivers and engineers [3]. For these reasons, sprint races not being interesting, not having a major impact on points scored, and not being a good measure of skill, we choose to ignore sprint races in coming up with our results.

As mentioned in the previous paragraph, Formula One presents a challenge to both drivers and engineers. The pairing of engineers and drivers must be emphasized. Currently in Formula One (at the time of writing 2023; persisting from the previous few years) there are 10 teams participating in Formula One with 2 drivers competing from each team in each race, resulting in 20 drivers competing in a given race. [7] mentions a quote from retired Formula One driver and 2016 World Championship winner (driver with the most points in 2016) Nico Rosberg that 80% of the success of a driver is attributed to their team and 20% of their success is attributed to their own skill. The team of a driver is usually called a constructor, since they are who builds the car used by the driver. For example, in 2023 Max Verstappen is said to be driving for team/constructor Red Bull.

2.2 Race prediction: why it is interesting in Formula One

Formula One is a type of multicompetitor game [12], meaning that for each game taking place, N competitors participate and are ranked $1, \ldots, N$. This is in contrast to head-to-head games where for each game, 2 competitors compete and each one either wins or loses. The problem of predicting head-to-head games lacks interestingness, since you can expect each competitor to behave roughly the same for the entirety of a season. For example, figure 1 shows the net win/loss amount of Major Leage Baseball (MLB) teams that are part of the American League West division. From this graph, it can be seen that if one team competes poorly at the start of the season, you can expect them to perform with the same amount of poor performance for the rest of the season (especially highlighted by the Oakland Athletics, whose net win/loss decreases roughly linearly over the 2023 season).

In addition to Formula One drivers competing in multicompetitor games, they each belong to a constructor. As remarked in section 2.1, the constructor that a driver belongs to has a large effect on their performance in a race. This introduces a few complications into race prediction. Firstly, that race results do not necessarily reflect the skill of a driver; for example a top tier driver might consistently place in the lower quartile because his car is lesser quality. Second, it is reasonable to expect that the 'skill' of a constructor can vary wildly throughout one season, and the skill of a driver can be expected to remain relatively constant through a season. By 'skill' of a constructor, we really mean the quality of the car they produce for their driver. Constructors are allowed to make changes (usually called upgrades) to their drivers cars, which has the potential to drastically increase or decrease the performance of their cars. In Formula One, it is common for results in the first half of a season to be much different compared to the second half, since 'summer break' occurs roughly halfway through a season, which is roughly a month of no races that many constructors use to modify their cars heavily.

Besides being interesting to model for its mathematical complexity, predicting the outcomes of

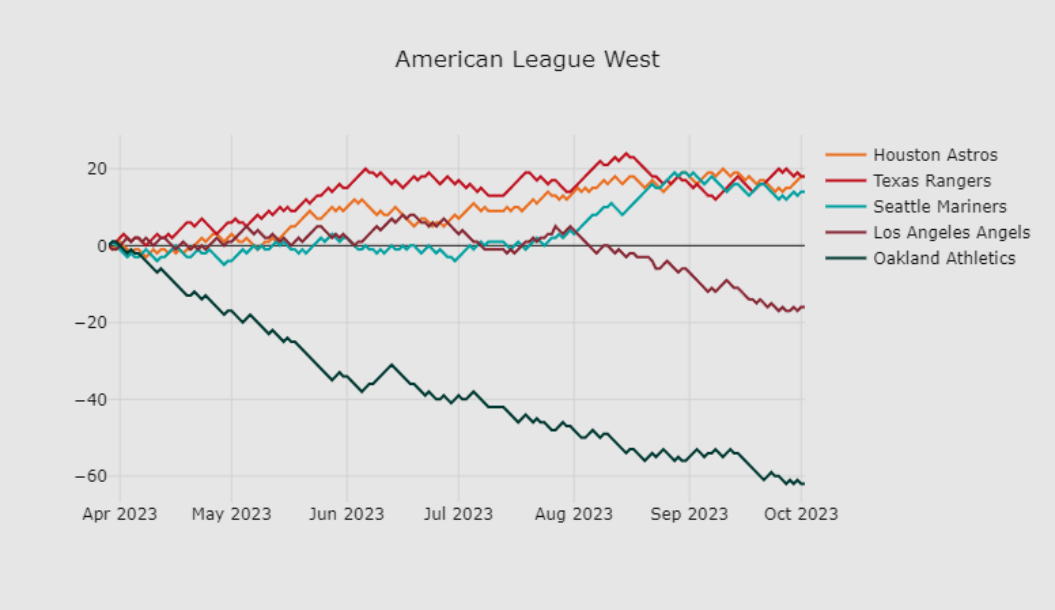

Figure 1: Net win/loss amount of Major Leage Baseball teams that are part of the American League West division. Major League Baseball is an example of a head-to-head sport: teams compete with one other team over sequences of games, where they either win or lose.

races has practical applications. Teams might want to isolate the skill of their drivers from their own constructor skill to determine whether or not they should sign a driver for another contract. Of course, predicting race results without regard to the underlying skills of drivers or constructors has the applicability to gambling, where gamblers are more interested in the outcome of a race instead of how the outcome was decided. For the author, interestingness of predicting race results comes from his own interest: he thinks that it would be neat to be able to predict a race before he watches it and see if his model was correct.

2.3 How Bayesian statistics makes sense for this paper

In this paper, we use a Bayesian model to attempt to predict the results of Formula One races. By using Bayesian statistics, we gain two big advantages and accept one big drawback. One of the advantages and the disadvantage come from the same source, so we begin with the independent advantage: that our model gives easily interpretable results.

Interpretable results comes from the fact that Bayesian models can be used to make inferences based on calculated parameters. We use an example from physics to illustrate this. In [15], a Bayesian model is used to estimate the critical temperature T_c of a superconductor. A critical temperature can be thought of a temperature at which a material changes its physical behavior drastically. Inference on the state of a material above or below a critical temperature isn't really that interesting of a problem to try: we can predict the state of a system by time evolving it using standard physics. The interesting part is determining the critical temperature T_c parameter of the material, since that cannot be calculated using standard physics time evolution techniques.

In this paper, our model returns easily interpretable parameters: ones that represent skill. Without making inferences, and looking only at these parameters, we are able to give a ranking to which driver is better skilled compared to another. Additionally (described in further detail in section 3), we are able to determine skills of constructors, so we can exam driver skill isolated from constructor or determine

which constructor is the most skilled.

The remaining advantage and drawback come from the fact that our model is Bayesian: there is a fair amount of subjectivity. We gain the advantage of easily being able to tell if our results make sense. For example, we can tell that something is wrong if the model predicts that Verstappen in a Haas would still dominate Formula One in 2023. He is an excellent driver, but a Haas is not a championship winning car. This same interpretability leads to the main drawback: people interpret results differently. For example, the author believes that the driver Nico Hulkenberg is a top 10 driver in terms of driver skill, in contrast to his ranking of 16 in points this season. Other fans of Formula One might disagree with such a statement. We believe that the main drawback in this occur in fringe cases such as the Hulkenberg example, while it should be generally agreed that the Verstappen example holds.

3 Methodology

Recall the primary objective of this paper: to predict Formula One race results using past race results. We explain our method for accomplishing this task in section 3.3, which is answered by predicting that the results will just be the ordering of drivers skills. The more interesting results of this paper come from the fact that this inference is made from a Bayesian model, so the parameters used to make such an inference are easily interpretable.

The remainder of this section is organized as follows. In subsection 3.1 we describe the source of the data used in this paper and what parts of it are used and how it is stored in code. In subsection 3.2 we perform a literature review of papers that examine questions similar to the focus of this paper. Particular emphasis is placed on two papers that form the basis of the model used in this paper. In subsection 3.3 we describe the model used in this paper; results of this model are discussed in section 4.

3.1 Data

To be able to make predictions of race results using race results, we need data on race results. Our race result data comes from the `Python` library `fastf1` [5], which obtains its data from the `ergast` API [2]. In analyzing 2023 race data, we noticed one entry in the data had no driver matched to it, which was manually matched to the driver Stroll. Additionally, data on whether a race was a 'street' or 'permanent' was added in manually. By a street race, we mean a race that the laps are done on streets designed for everyday use, while by a permanent race we mean a track designed to be raced on by racecars. In practice, we stored the data in tabular form as follows (a `pandas` dataframe was used for storage):

Abbrev	Pos	GridPos	TeamId	Race	CircType	RaceNum	TeamNum	DrivNum
MAG	10	4	Haas	Miami	1	5	9	13

Figure 2: Layout of data used in code, along with a sample row. This sample reads as follows: driver Magnussen (MAG) finished in position 10, started in position 4, is on team Haas, for the Miami grand prix, this race is a permanent circuit (1 permanent, 0 street), it is the 5th race of the season, the index of the team MAG is on (Haas) is 9, the index of the driver MAG is 13. To keep track of drivers and teams, index numbers are used in code instead of strings.

Additionally, we chose not to delete entries from our dataset due to a driver not finishing. [14] who performs an analysis similar to the one done in this paper (discussed further in section 3.2) chose to delete entries due to a driver not finishing, since they were more interested in determining driver skill

and had a belief that extraneous things can cause car failures. In this paper, the skill of a driver and their constructor is calculated. We choose to keep these entries, believing that their effect might 'even out' with drivers not finishing due to their own skill or mechanical failurs (constructor skill).

3.2 Literature review

The literature review performed for this paper consisted of two parts. The first being a review of methods used for problems like the one addressed in this paper. Evidently, based on the title of this paper, a Bayesian method was chosen. The second part being a review of what questions and methods other papers used to answer questions similar to the one addressed in this paper using data like the data used in this paper (data related to race results, such as starting grid position, ending grid position, and points earned). We begin with the first part of the literature review, to detail why we chose a Bayesian method, then review the second part.

In searching for methods that predict race/multicompetitor game results (or some result related to this, for example searching for the best athlete in a sport of all time) without doing a full simulation of a race, nearly all were parametric models that draw performances dictating the results of a race. An example of a paper that does not rely on drawing performances to dictate race results is [10]. Instead, this paper essentially performs a generalized least squares fit on race result data to fit skill parameters to constructors and drivers. Based on interest of not doing least squares, the author of this paper (the one writing these words) chose to not use a method such as this.

Parametric models that draw performances dictating results took primarily two forms. The first being maximum likelihood estimation and the second being standard Bayesian inference (sampling from a posterior). Both types of methods begin with the following setup, see [9] or [8] for a fuller description of maximum likelihood methods, and [12] for a fuller description of standard Bayesian inference methods.

The setup is as follows. Over a time of interest (for example, a season) M multicompetitor games are played, each with N competitors, taken from a pool of K overall competitors. Each of the M, N, K can be further decomposed into indices based on the specifics of how the sport goes (for example, index M by times of interest, or N by competitors in a match), but the current specification suffices without being overly complicated by indices. Each competitor i, for $i = 1, \ldots, K$, has an associated skill parameter θ_i (which can be a scalar or vector). Note that the θ_i can be easily interpretable, such as being a direct measure of a competitors skill as in [12], but it need not be. For a given competition j for $j = 1, \ldots, M$, each competitor i_j performing in the competition draws a performance Y_{i_j} for N different i_j's according to some distribution $F(y|\theta_{i_j})$. The result of the competition is then the ordering of the Y_{i_j}'s, the competitor with the largest value of Y_{i_j} winning, down to the smallest value. The probability of a particular outcome of a race is then, ordering the i_j's according to the drawn values of Y_{i_j} (that is, $i_j = 1_j$ corresponds to the largest drawn Y_{i_j} down to $i_j = N_j$ the smallest):

$$P_j(Y_{1_j} > Y_{2_j} > \cdots > Y_{N_j}|\boldsymbol{\theta}) = \int_{-\infty}^{\infty} \int_{-\infty}^{y_{1_j}} \cdots \int_{\infty}^{y_{(N-1)_j}} f(Y_{1_j}, \ldots Y_{N_j}|\boldsymbol{\theta}) dy_{1_j} \ldots dy_{(N-1)_j} \tag{1}$$

where f is the joint cumulative distribution function of the Y_{i_j}'s and $\boldsymbol{\theta} = (\theta_{i_1}, \ldots, \theta_{i_N})$. Note that integration over y_{N_j} does not need to be performed since this is the smallest drawn performance. In practical applications, draws of Y_i are assumed to be independent and the above integral does not need to be computed. Assuming races are independent, the likelihood function of the entire time interval is

then:

$$L = \prod_{j=1}^{M} P_j \tag{2}$$

where the P_j are computed according to equation 1. This point is where both parametric model types diverge in computation. Maximum likelihood methods perform maximum likelihood estimation on equation 2, while standard Bayesian inference methods sample from equation 2. Of course, according to whichever method is used, specifications on skill parameters θ_i and cumulative distribution functions $F(y|\theta_i)$ are given, usually simplifying equation 2 greatly. [12] adopts the name rank ordered logit for models with a likelihood function described by equation 2, which we adopt as well.

We now move onto the second part of the literature review: reviewing papers that study Formula One in particular. All papers found in this literature search were concerned with longtime average behavior. For example, one paper mentioned in the previous part of the literature search [10] was concerned with determining the best driver in Formula One since its inception in 1950. As mentioned before, our paper in particular wants to use data obtained not long ago to predict results right now, and not average over long periods of time. This literature review serves the purpose of obtaining general trends that we should notice our analysis returns.

All papers found that attempt to answer the question whether driver or constructor skill matter the most in determining race results all find that constructors have the largest impact [10] [7] [14]. [8] finds that maximum likelihood based methods for determining driver skill does not match up with points rankings in a season. [8] also finds that while different assumed cumulative distribution functions generally give the same skill rankings, they often differ in their skill rankings in single drivers drastically (for example, one driver was ranked 7th and 8th according to 2 different assumed cumulative distribution functions, while the same driver was ranked 14th on a third; with the remainder of the results being roughly the same).

3.3 Methodology

In this paper we chose to use a variant of standard Bayesian inference applied to a rank ordered logit model, as described in [12], and used in [14]. Specifically, [12] lays out the basic theory of applying a rank ordered logit model to a multicompetitor game to perform Bayesian inference on, while [14] applies a variant of that method to Formula One racing results from the $2014 - 2021$ seasons with the goal of determining the best skilled driver of that era (independent of constructor). Here we use a slight variation of the method used in [14]. In this section we describe our implemented variation, how it differs from that of [12] and [14], and include some code implementation details.

3.3.1 Theory

We are interested in predicting race results in a given season using race results from previous races in that same season. In a given season, there are n total races. For each race, each of the 10 constructors have 2 drivers compete, leading to $N = 20$ drivers competing in each race. Associated with each driver i is a skill parameter θ_i (scalar). For a given race, each driver participating draws a performance Y_i according to a Gumbel distribution:

$$Y_i \sim \text{Gumbel}(\theta_i) \tag{3}$$

$$F_{\text{Gumbel}}(y|\theta) = \exp(-\exp(\theta - y)) \tag{4}$$

where 4 is the cumulative distribution function of the Gumbel distribution. The justification of a Gumbel distribution for performance draws in described in [12] and is used in [14]. Assuming that these performances are drawn independently, it can be shown (by a counting/combinatorics argument) that the probability Y_1 is greater than Y_2 down to Y_N is:

$$P(Y_1 > Y_2 > \cdots > Y_N | \boldsymbol{\theta}) = \frac{\prod_{i=1}^{N-1} \exp(\theta_i)}{\sum_{j=i}^{N} \exp(\theta_j)} \tag{5}$$

where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_N)$. By inspecting the numerator and denominator of 5, it can be shown that this is maximized when $\theta_1 > \theta_2 > \cdots > \theta_N$.

We pause with developing our model here to discuss interpretation. The Y_i draws should be interpreted literally as how well a driver performs in a given race. If he has the highest Y_i draw, he wins: in this model, race outcomes are determined by draws of Y_i. Similarly, the skill parameter θ_i should be interpreted literally as the skill of a driver. The center of a Gumbel distribution is proportional to θ_i, so draws Y_i are drawn around θ_i: we expect drivers with a higher skill to perform better, but he still might lose to a lesser skilled driver. The fact that 5 is maximized when $\theta_1 > \theta_2 > \cdots > \theta_N$ leads to our prediction of a given race: we predict the race will reflect the ordering of the θ_i's. Lastly, compared to the literature review, it can be noted that we are using far fewer indices in our equations. This is not a mistake. In the literature review, we put emphasis on being mathematically correct with every statement. Here we drop indices since doing so makes coding the model easier: to compute 5 for a given race, we search for the drivers that participated in the race, find their driver indices (mentioned in section 3.1), order these indices in order of race results, and compute plainly following the equation. No tricky multi-index work required.

At this point the information on how to proceed from [12] stops, this paper only provides the basis of the rank ordered logit model and leaves execution to users. [14] modifies the model by decomposing driver skill θ_i as follows:

$$\theta_i = \theta_{i;\text{ seas}} + \theta_{i;\text{ cons}} + \theta_{i;\text{ skill}} \tag{6}$$

where $\theta_{i;\text{ seas}}$ is the skill of driver i in a particular season, $\theta_{i;\text{ cons seas}}$ is the skill of the constructor that the driver is on in a particular season, and $\theta_{i\text{ skill}}$ is the overall skill of the driver (over all seasons). Since this paper is focused on only using the results of a single season, in our model we decompose the skill of driver i as:

$$\theta_i = \theta_{i;\text{ dri}} + \theta_{i;\text{ cons}} \tag{7}$$

where $\theta_{i;\text{ dri}}$ is the actual skill of driver i, and $\theta_{i;\text{ cons}}$ is the skill of the constructor he drives for. [14] further decomposes driver skill as follows:

$$\theta_{i;\text{ skill}} = \gamma_{i,0} + \gamma_{i,1} I(\text{wet}) \tag{8}$$

$$\theta_{i;\text{ cons}} = \beta_{i,0} + \gamma_{i,1} I(\text{permanent}) \tag{9}$$

where $I(\text{wet})$ is an indicator function that is 1 if the weather is rainy during the race and 0 if it is dry, and $I(\text{permanent})$ is an indicator function that is 1 if the circuit the race takes place on is permanent (made just for racecars) and 0 if the race is a street race.

We again stop for interpretation. In equation 7 we assume that the skill parameter of a driver linearly depends on the drivers actual skill and their constructors skill. In accordance to our literature review, it makes sense that the largest influences on a particular drivers performance is their actual skill and their constructors skill. Equation 8 reflects the beliefs of the authors of [14] that drivers have different skills whether or not they are racing on a wet road or not, while equation 9 reflects their belief that some cars are better on streets than others. Beliefs that drivers perform differently based whether the track is wet or on a street makes sense, but the implementation of this belief is up to interpretation. The author of this paper (the one being read) disagrees with the authors of [14] that street race skill is reflected in constructor skill, he believes that it is reflected in individual drivers skill; there are reasons why Formula One fans say certain drivers are better at street races and not constructors are.

Taking all combinations of equations 6, 8, and 9 leads to [14] examining a total of 4 submodels: plain equation 6, equations 6 + 8, equations 6 + 9, and equations 6 + 8 + 9, of which they find submodel 6 + 9 (circuit model) to perform the best.

Since this paper is interested in the 2023 season which was relatively dry, and some of the races that did receive rain received it only in parts of the race, we do not implement a submodel that accounts for wet races due to lack of good data. We feel that there has been sufficient data this season to use a street/permanent circuit model, but choose to reflect this in driver skill instead of constructor skill like [14]. We make one additional submodel by ignoring the constructor + driver skill split, and represent the skill of each driver by one plain parameter. In all, we consider the following three submodels described by how driver skill is assigned:

$$\theta_i = \theta_i \tag{10}$$

$$\theta_i = \theta_{i;\,\text{dri}} + \theta_{i;\,\text{cons}} \tag{11}$$

$$\theta_I = \gamma_{i,0} I(\text{street}) + \gamma_{i,1} I(\text{permanent}) \tag{12}$$

where $I(\text{street})$ is an indicator function that is 1 if the race is a street race, and $I(\text{permanent})$ is an indicator if the race is on a permanent track. We call these submodels the single skill, con + dri, and con + dri + cir submodels respectively.

3.3.2 Implementation

In section 3.3.1 we described the theory of our model used in this paper. In this section, we describe how we used Bayesian inference to obtain results using this model. [6] summarizes the process of making a Bayesian model for a problem as follows:

1. Define a (joint) probability model over observables y and unobservables θ
2. Condition on observed data y to infer the posterior distribution of a quantity of interest
3. Evaluate the model

Equation 5 defines the joint probability model for our problem. This equation computes the probability that a certain race obtains a given outcome, to compute the probability over an entire season we multiply this probability over all races (assuming independence) like equation 2. Denote this season probability by $P(\mathbf{Y}|\boldsymbol{\theta})$. To complete step 2, we must sample from the posterior distribution $P(\boldsymbol{\theta}|\mathbf{Y})$ given by Bayes's theorem:

$$P(\boldsymbol{\theta}|\mathbf{Y}) \propto P(\mathbf{Y}|\boldsymbol{\theta})P(\boldsymbol{\theta}) \tag{13}$$

where $P(\boldsymbol{\theta})$ is the prior distribution of $\boldsymbol{\theta}$; a distribution that driver skills are drawn from. We will discuss our choice of prior distribution and move onto step 3 in a moment, but we will first discuss our method used to sample from 2.

Sampling of equation 13 was done using the programming language `Stan` [11]. A gentle introduction to the language is provided in [6]. In writing `Stan` code, the user only has to give instructions on how to calculate 2, what data they input for this calculation, and what parameters they want samples of. Technically in `Stan`, the natural logarithm of 13 is required and the prior distribution is implemented automatically, so in practice this consists of looping over the number of races in a season and adding together $\ln(P(Y_1 > \cdots > Y_N|\boldsymbol{\theta})$, where from equation 5:

$$\ln(P(Y_1 > \cdots > Y_N|\boldsymbol{\theta})) = \sum_{i=1}^{N-1} \left(\theta_i - \ln \left(\sum_{j=i}^{N} \exp(\theta_j) \right) \right) \tag{14}$$

$$= \sum_{i=1}^{N-1} (\theta_i - \text{lse}(\theta_i, \ldots, \theta_N)) \tag{15}$$

where lse is the LogSumExp function. What `Stan` then does given instructions on how to compute 13 is sample $\boldsymbol{\theta}$'s from it. It does this by using a variant of (Hamiltonian) Monte Carlo using the No-U-Turn-Sampler [11]. Practically speaking, this is just a variant of standard Monte Carlo as described in Hastings original paper [13], which is used for sampling from probability distributions known up to a multiplicative constant, but tends to work very well on Bayesian problems.

One last part required before being able to run `Stan` is then a choice of prior distribution $P(\boldsymbol{\theta})$. For this we choose to use what is called a weakly informed prior [6], which means we choose a prior distribution that is roughly around (at least in order of magnitude) what we expect the true $\boldsymbol{\theta}$ to be. Inspired by the default choice in prior by the `BRMS` package for Bayesian inference, which is a half student t distribution with 3 degrees of freedom, and inspection on orders of magnitude obtained in [14] which uses a model similar to the one used here, we choose a student t distribution with 3 degrees of freedom as our prior.

We now move onto part 3 of implementation: evaluation of our model. In section 3.3.1 we described 3 different submodels that we will be testing. Our first goal in model evaluation is to select one submodel as best that we use for the remainder of the paper. To do this, we subject our three models to two tests. First is called an $\hat{R}$ test. In `Stan`, samples are drawn using a variation of Monte Carlo. Although Monte Carlo is proven to converge in its draws to a distribution [13], we only run Monte Carlo sampling for a finite time. Specifically, for each submodel, we run Monte Carlo over 4 chains (essentially running one iteration 4 different times), with 1000 warmup iterations (samples drawn here are thrown away; this is just to make sure Monte Carlo finds where it should be), and 2000 samples each; resulting in $4 \cdot 2000 = 8000$ samples. After running, `Stan` calculates what is called an $\hat{R}$ value, which gives a measure of how good the Monte Carlo was performed. A value close to 1 indicates good sampling, while far from 1 indicates bad sampling [11]. If any of our models consistently has a bad $\hat{R}$ value after adjusting how we sample from it, we eliminate the model.

The second test is a cross validation test. This is done using the `R` package `loo` [16], whose precise implementation mechanics can be found in [17]. Here we describe how `loo` works practically for choosing a best submodel, ignoring techincal details. `loo` stands for 'efficient leave one out cross validation', which means that it performs leave one out cross validation (leave one datapoint out of the training data, fit the model again, see how the left out piece fits, repeat), which means that it performs leave

one out cross validation without refitting the model by using an approximation technique. As input, `loo` requires the natural logarithm of the likelihood evaluated at each sample point, which can be done by making use of `Stan`'s `generated quantities` code section and generating 15 for each race. See the code files associated with this paper for an example. As output for the likelihood value of a given model, `loo` returns three quantities:

`elpd_loo` `pareto_k_values` `n_eff`

Where `elpd_loo` gives a measure of how well the model performs with cross validation (higher is better), `pareto_k_values` is a list of as many values as parameters fit in the model that how confident `loo` is in its calculation of its `elpd_loo` value, and `n_eff` is an approximation of how many effective parameters for the model. For us, we are only concerned with the `elpd_loo` and `pareto_k_values`, since each of our models fits a different number of parameters. From this test, we select the model with the best `elpd_loo` result and which also has high confidence in this calculation by having favorable `pareto_k_values`.

At this point in model evaluation, we should have a representative submodel for the entire model. To examine how well this model performs, we subject it to two tests. The first is a subjective test, by asking: does the data make sense? If our model predicts that Sargeant is most likely to win a race, then we can tell that something went wrong. The second is called a visual posterior predictive check. Posterior predictive checks mean using our estimate our samples $\theta^{(m)}$ from our model with input data y to predict new data $\tilde{y}$ [6]. Our visual posterior predictive check follows the one used by [14]. To do this, we use our samples $\theta^{(m)}$ to simulate race result data $\tilde{y}$. Simulating many races, we can examine the finishing positions of drivers and compare these frequencies to their actual frequencies by using a histogram; which is why this is called a visual test.

To be specific on how to form such histograms for a visual posterior predictive check, we begin with our samples $\theta^{(m)}$ obtained through `Stan`. For each race of the season, we find what drivers participated in that race. We then simulate that race N times (for our code, we chose $N = 300$ simulations of each race) as follows. For a given race, take N samples $\theta^{(m)}$. For each of the N samples, draw performances $Y \sim \text{Gumbel}(\theta^{(m)})$ for the participating drivers and order these to determine the outcome of the simulated race. Each race should have N simulations which we perform over the amount of races n in the season (in 2023 there were $n = 22$ races). From these Nn simulations, determine the finishing frequency of each driver from the simulations, and also their actual finishing frequency. Organize these results into a histogram. If our model performs well, we should find that both generated histograms look similar.

4 Analysis

In this section we perform an analysis on the 2023 Formula One season following the methodology in section 3.3. We begin with exploratory data analysis on the starting and finishing positions of drivers in the 2023 season. In this exploratory data analysis, we present our data as graphs. There will be many graphs in this section, so we explain the convention that we are following in their presentation. Graphs that examine individual driver performance will be shown side-by-side with the same graph generated for their teammate. For example, if a graph showing data about Verstappen is to be shown, it will be shown alongside his teammate Perez. Emphasis is put on this since our model attempts to incorporate a separate skill for constructor, which should be shared in this data. To avoid cluttering with too many graphs in the main text, repetetive graphs (graphs that can be generated for each driver) are delegated to the appendix, with a sample or two being shown in the main text.

4.1 Exploratory data analysis

Our exploratory data analysis consisted of generating a graph for each driver that shows his starting position, ending position for each race. Additionally, his averaged finishing position is also displayed. We show the results for the drivers on team Redbull in figure 3, and the drivers on team Haas in figure 4. We do not include figures for other teams in the appendix since these graphs are not the most interesting and can be generated easily. Out of this sample, only Verstappen and potentially Hulkenberg's

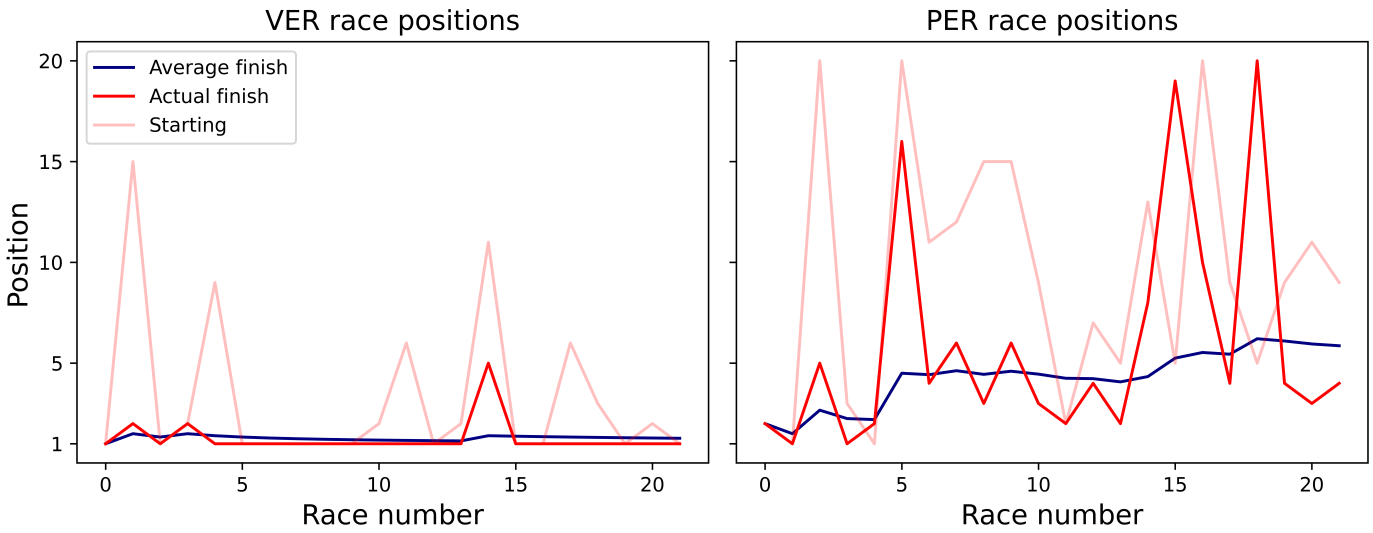

Figure 3: Average, actual, and starting positions over the 2023 season for the drivers on team Redbull, Verstappen and Perez

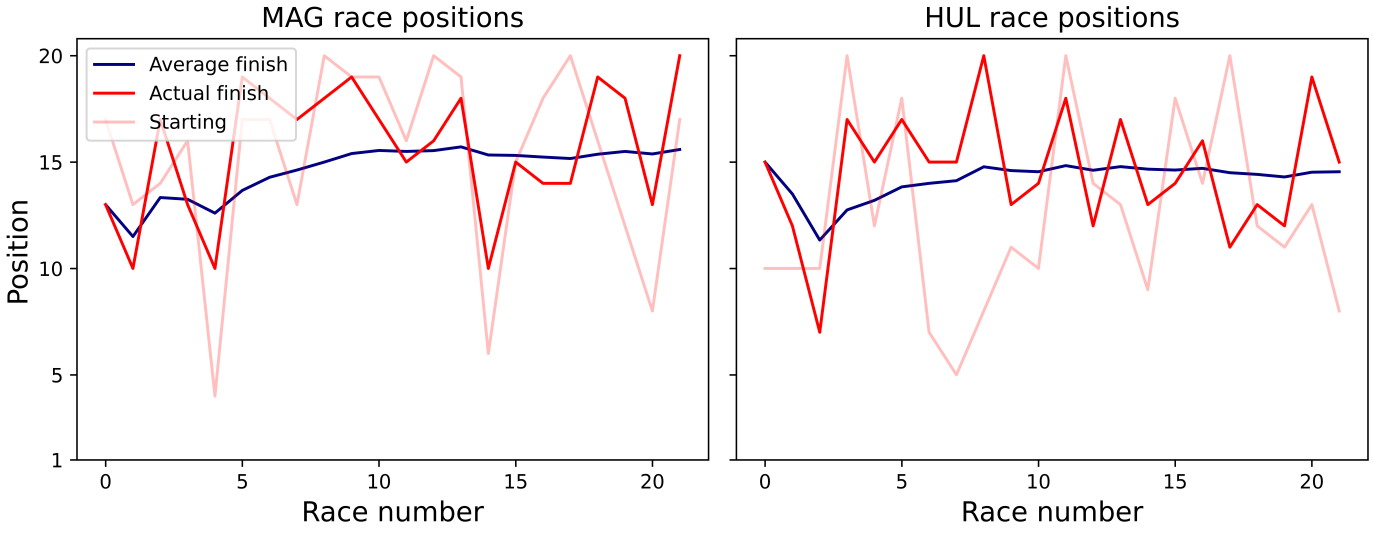

Figure 4: Average, actual, and starting positions over the 2023 season for the drivers on team Haas, Magnussen and Hulkenberg

finishing position seem to be accurately reflected by their average finishing positions. It should be noted

that starting position does not seem to be a very strong indicator of finishing position, affirming to us that leaving this factor out of our model seems like a fine assumption without losing much information.

4.2 Results

In this section we present the results obtained following the procedure developed in section 3.3. We divide this presentation into three main parts: model evaulation, where we choose a best submodel to use for all further analysis, estimates, where we list our computed skill estimates, and posterior predictive inference, where we perform a visual posterior predictive inference as detailed in section 3.3.2.

4.2.1 Model evaluation

Recall the three submodels we formulated in section 3.3.1, which were all based on the skill parameter θ_i of driver i:

$$\theta_i = \theta_i \tag{16}$$
$$\theta_i = \theta_{i;\,\text{dri}} + \theta_{i;\,\text{cons}} \tag{17}$$
$$\theta_I = \gamma_{i,0} I(\text{street}) + \gamma_{i,1} I(\text{permanent}) \tag{18}$$

Where we called these submodels the single skill, con + dri, and con + dri + cir submodels respectively. All three models obtained an $\hat{R}$ value close to 1, indicating that `Stan` was able to perform Monte Carlo sampling sufficiently; so all models pass the first test. Our second submodel evaluation consists of using `loo` to perform (approximate) leave one out cross validation. We detailed what the results of `loo` mean practically in section 3.3.2. Running `loo` on these three models returns the following results shown in table 1.

Model	`elpd_loo`	`pareto k`
Single skill	-862.8 ± 14.4	good
Con + dri	-863.7 ± 14.9	good
Con + dri + cir	-864.9 ± 15.8	bad

Table 1: `loo` results on the three submodels examined in this paper. The `n_eff` output of `loo` is omitted since it is uninterpretable between the three models. The `pareto k` column summarizes the `pareto_k_values` result from running `loo`; summarizing if the result says the `elpd_loo` estimation is confident or not.

Since the con + dri + cir model has a bad `pareto k` evaluation (the estimated `elpd_loo` value has low confidence), we eliminate this model from consideration. This leaves the single skill and con + dri submodels to remain. While the single skill model evaluates the best according to cross validation, we choose to use the con + dri model in the rest of our analysis since it is more interpretable and its `elpd_loo` value is not far off the one of the single skill model. For a person only interested in predicting race results and no interpretation, these results suggest the single skill model would work best.

4.2.2 Estimates

From the previous section, we chose the con + dri model as best, which has a skill parameter θ_i for driver i written as follows:

$$\theta_i = \theta_{i;\,\text{dri}} + \theta_{i;\,\text{cons}} \tag{19}$$

Running `Stan` returns many samples (distinguished by an upper index (m)) of $\theta_{i;\text{ dri}}$ and $\theta_{i;\text{ cons}}$. We take as our estimates (estimates are distiguished by a roof notation) of these parameters to be their sample mean:

$$\hat{\theta}_i = \hat{\theta}_{i;\text{ dri}} + \hat{\theta}_{i;\text{ cons}} \tag{20}$$

$$\hat{\theta}_{i;\text{ dri}} = \text{sample mean}(\theta^{(m)}_{i;\text{ dri}}) \tag{21}$$

$$\hat{\theta}_{i;\text{ cons}} = \text{sample mean}(\theta^{(m)}_{i;\text{ con}}) \tag{22}$$

Recall that a sample mean is the result of adding all samples of a quantity and dividing by the amount of that quantity. Associated with the sample mean is the sample standard deviation, which we write in our results following with a $\pm$.

In table 2 we display our estimates of $\hat{\theta}_i$ and $\hat{\theta}_{i;\text{ dri}}$ for the 22 drivers that participated in the 2023 Formula One season. Results are displayed in order of decrementing $\hat{\theta}_{i;\text{ dri}}$. Following this, in table 3 we display our estimates of $\hat{\theta}_{i;\text{ cons}}$ for the 10 constructors that participated in the 2023 Formula One season. Drivers are labeled by the first three letters in their last name. For example, ALO refers to the driver Alonso.

Driver	Total skill	Actual skill	Driver	Total skill	Actual skill
VER	4.16 ± 2.17	3.03 ± 1.23	HUL	-0.55 ± 1.33	-0.12 ± 0.64
ALO	0.84 ± 1.37	0.66 ± 0.68	TSU	-0.42 ± 1.09	-0.12 ± 0.52
HAM	0.89 ± 1.33	0.47 ± 0.64	LEC	0.024 ± 1.311	-0.13 ± 0.63
SAI	0.44 ± 1.31	0.29 ± 0.64	PIA	-0.19 ± 1.30	-0.18 ± 0.63
LAW	-0.056 ± 1.173	0.24 ± 0.61	OCO	-0.30 ± 1.29	-0.20 ± 0.63
NOR	0.18 ± 1.3	0.19 ± 0.63	BOT	-0.51 ± 1.31	-0.20 ± 0.63
ALB	-0.23 ± 1.33	0.14 ± 0.65	MAG	-0.72 ± 1.33	-0.29 ± 0.65
RIC	-0.17 ± 1.34	0.12 ± 0.57	PER	0.78 ± 2.07	-0.35 ± 1.03
GAS	-0.017 ± 1.296	0.085 ± 0.633	STR	-0.25 ± 1.36	-0.44 ± 0.68
RUS	0.36 ± 1.33	-0.062 ± 0.641	SAR	-0.87 ± 1.34	-0.50 ± 0.65
ZHO	-0.42 ± 1.30	-0.11 ± 0.63	DEV	-0.85 ± 1.12	-0.56 ± 0.056

Table 2: Estimated driver skills for the 2023 Formula One season. The column total skill is the sum of the drivers actual skill and the skill of their constructor. The data is organized in descending driver actual skill.

In figure 5 we display graphically our driver actual skill results. Similarly in figure 6 we display graphically our constructor skill results. In figure 7, we display both the actual skill and total skill of all drivers on the same graph, to show the impact of constructor skill. In table 4, we compare our driver skill rankings to the points ranking of the 20 finishing drivers of the 2023 Formula One season. It should be noted that out of the 22 drivers, 19 participated in every race. The driver DEV (Devries) was replaced by RIC (Ricciardo) after participating in 10 of 22 races. Ricciardo then participated in 2 races before breaking his hand, leaving him out of 5 races in which the driver LAW (Lawson) substituted for him in.

4.2.3 Posterior predictive inference

In section 3.3.2 we laid out the instructions on how to perform a visual posterior predictive inference on our model. In summary, this consists of using samples $\theta^{(m)}$ of driver skill to attempt to generate new race result data $\tilde{y}$. The visual part comes from graphically examining predicted race result frequencies

Constructor	Skill
Redbull	1.31 ± 1.05
Mercedes	0.42 ± 0.69
Aston Martin	0.19 ± 0.70
Ferrari	0.16 ± 0.68
Mclaren	-0.005 ± 0.667
Alpine	-0.10 ± 0.66
Alpha Tauri	-0.31 ± 0.67
Alfa Romeo	-0.31 ± 0.67
Williams	-0.37 ± 0.68
Haas	-0.43 ± 0.68

Table 3: Estimated constructor skills for the 2023 Formula One season. The data is organized in descending constructor skill.

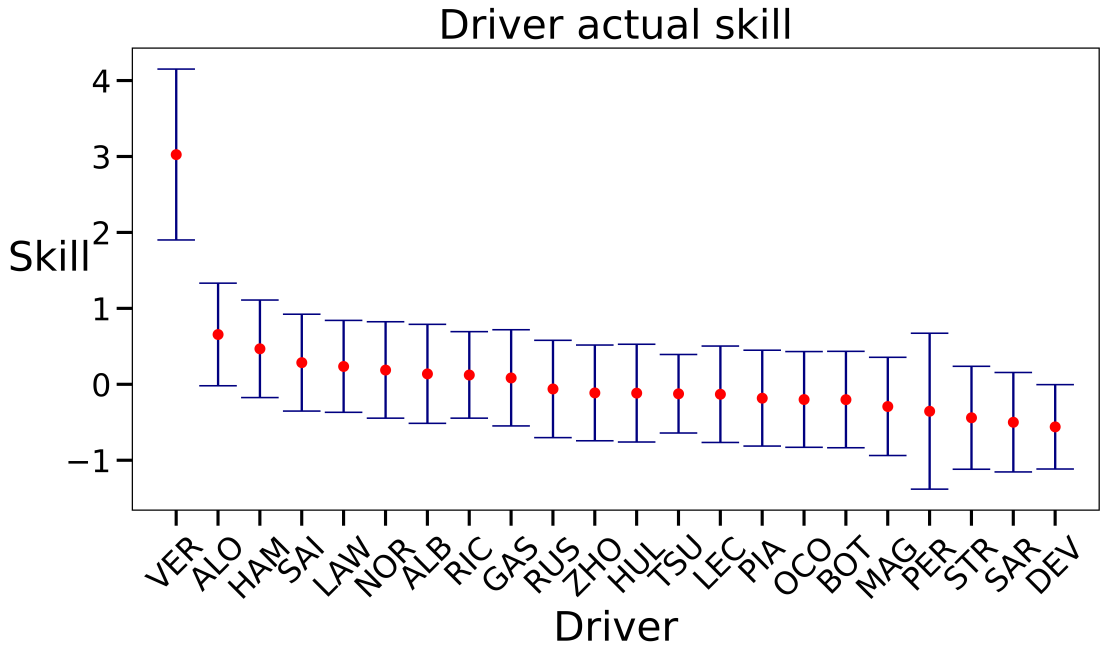

Figure 5: Estimated actual skill of each driver in the 2023 Formula One season. Drivers are listed in descending order of their actual skill.

to actual race result frequencies for each driver. See section 3.3.2 for a complete explanation. In this section we display 2 graphs made from following that method. Graphs for all drivers are displayed in the appendix. In figure 8 we show our posterior predictive inference graph for team Redbull (Verstappen and Perez), and in figure 9 for team Haas (Magnussen and Hulkenberg).

Figure 6: Estimated actual skill of each constructor in the 2023 Formula One season. Constructors are listed in descending order of their skill.

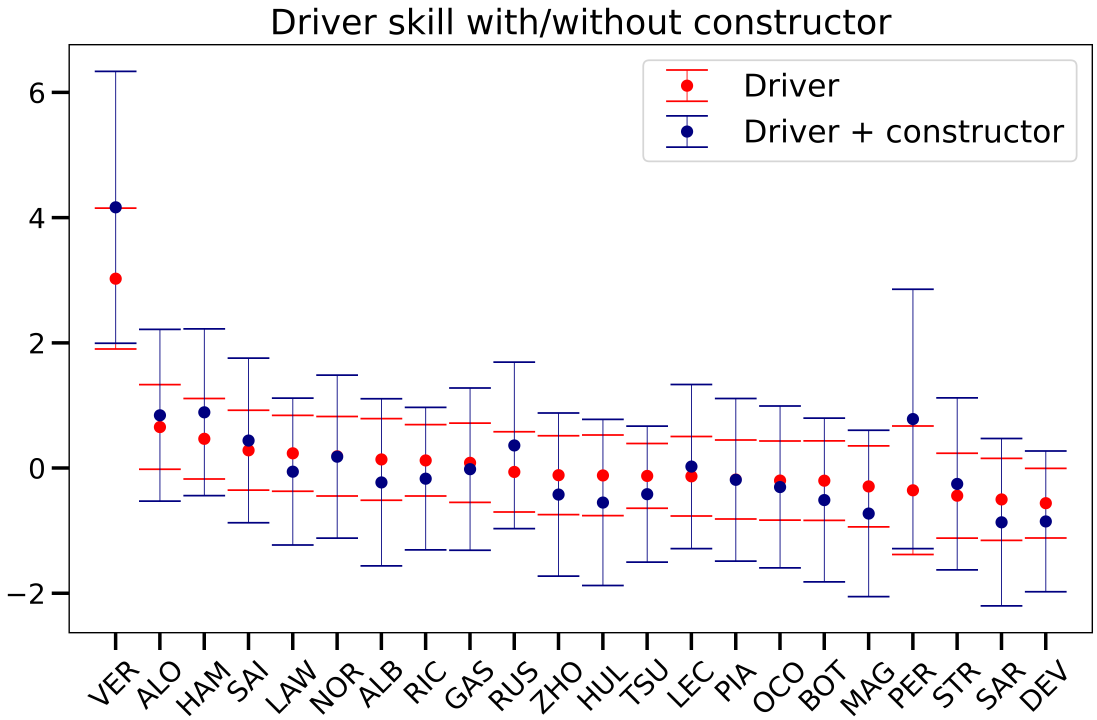

Figure 7: Actual and total skills of each driver in the 2023 Formula One season. Drivers are listed in descending order of their actual skill.

Driver	Points ranking	Total skill ranking	Driver	Points ranking	Skill ranking
VER	1	1	GAS	11	9
PER	2	4	OCO	12	14
HAM	3	2	ALB	13	12
ALO	4	3	TSU	14	15
LEC	5	8	BOT	15	17
NOR	6	7	HUL	16	18
SAI	7	5	RIC	17	10
RUS	8	6	ZHO	18	16
PIA	9	12	MAG	19	19
STR	10	14	SAR	20	20

Table 4: Comparison of computed total skill ranking to points ranking for all 20 season finishing drivers in the 2023 Formula One season

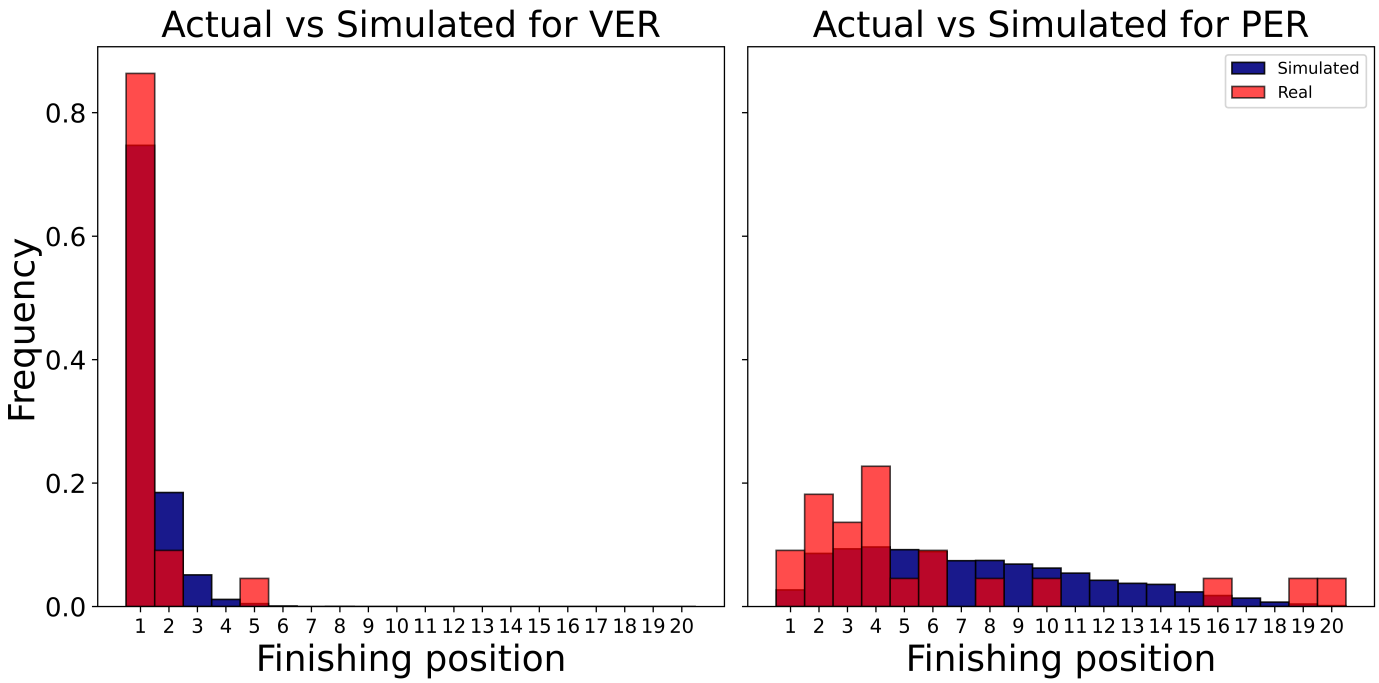

Figure 8: Simulated compared to actual finishing position results for the drivers on team Redbull: Verstappen and Perez

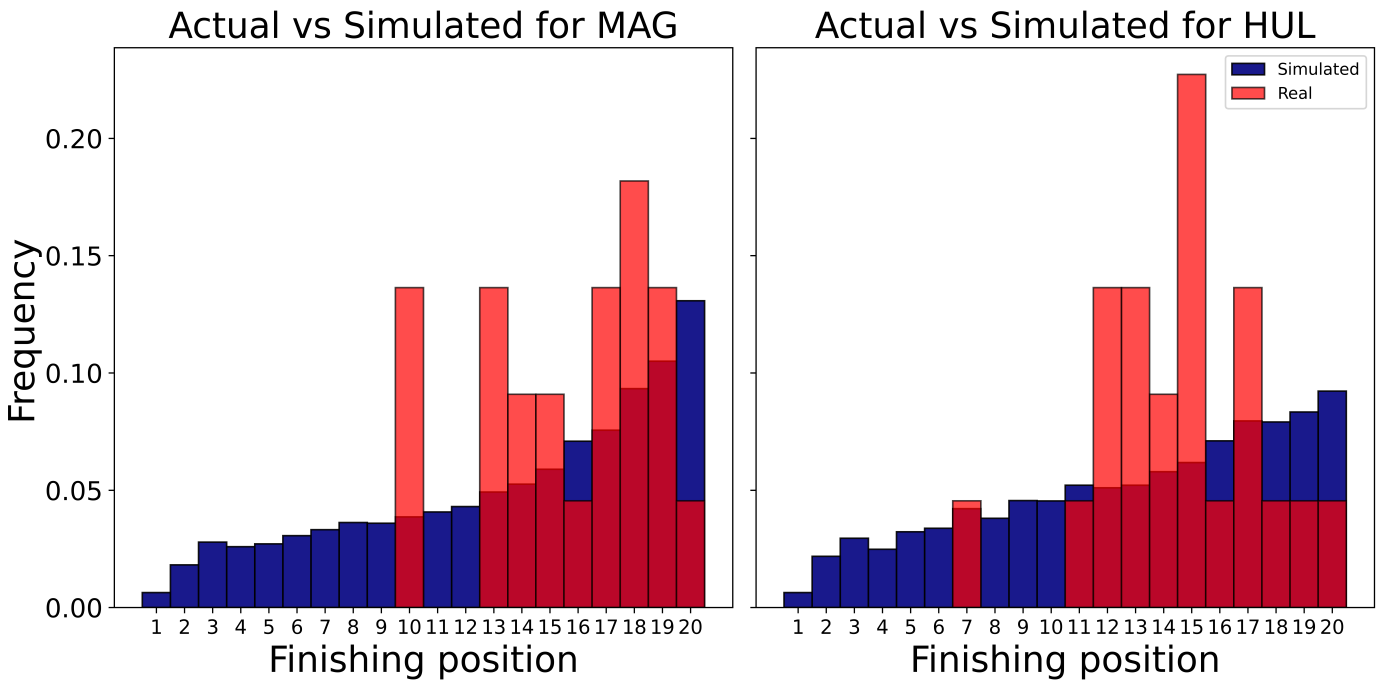

Figure 9: Simulated compared to actual finishing position results for the drivers on team Haas: Magnussen and Hulkenberg

5 Conclusion

Examining figures 5 and 7, it is apparent that we find a similar conclusion to what was found in our literature search: that race results are mainly determined by constructor. This is especially evident in figure 5, where most drivers are estimated to have roughly the same skill. Interestingly, our skill ranking roughly coincides with Formula One points rankings, shown in table 4. This result was not expected since points are only awarded to 10 drivers per race (half of participants) and points awarded decrease swiftly from 25 points to the 1st place winner and 1 point to the 10th place winner.

While these results make our model appear to perform quite well, it exhibits not so good performance in a visual posterior predictive check. Evident from examining finishing position frequencies, drivers tend to finish in the same position quite often. Our race simulation predicts that drivers finish smoothly over nearly all positions. Practically speaking, we do not expect some cars to ever finish in a top 3 place; for example a Haas car. Our race simulation predicts that a Haas has a sizable chance of finishing in a top 3 position. We deem posterior predictive inference to be mostly a failure for this model. While it does sometimes follow actual results in a sense, it does not follow them enough and predicts very wrong results (for example, Haas winning a race).

We do feel that the results in the model have been swayed greatly by team Redbull. It is well known that Redbull has constructed an extremely dominant car in the 2023 season, but they have had only one driver take full advantage of this fact: Verstappen. Perez has admitted to greatly underperform this season, with evidence in figure 3. We feel that the severe underperformance by Perez artificially boosted Verstappen's actual skill estimation, and drug down our skill estimation of team Redbull. As a consequence, the skills of all other drivers are clumped together much closer than they should, since by design $\sum_i \theta_i = 0$ (an average driver should have a skill of 0), which Verstappen's skill of ~ 4 disrupts greatly. As an illustration of this, we can compute the probability that our model predicts Verstappen in a Haas would be Alonso (in his current Aston Martin). This is just equation 5 with $N = 2$ and each drivers $\hat{\theta}_i$ being appropriately read off from tables 2 and 3. This calculation goes out as follows:

$$P(\text{VER in a Haas} > \text{ALO}) = \frac{\exp(3.03 - 0.43)}{\exp(3.03 - 0.43) + \exp(0.84)} \tag{23}$$

$$\doteqdot 0.85 \tag{24}$$

So our model predicts that Verstappen in a Haas would beat Alonso in his current car 85% of the time. This is just something that cannot happen: a Haas cannot win.

Concluding, we feel that our model is okay at predicting Formula One race results. Predictions are done by computing driver skills θ_i and ordering them, since this is the most probable outcome of a race according to [**?**]. There are two immediate changes to the model that the author feels might help the model with prediction. The first is to change the cumulative distribution function from a Gumbel distribution (equation 3) to a much sharper distribution. The reasoning for this comes from performing a visual posterior predictive check, where the model predicts too flat of race outcomes compared to actual outcomes, which are very sharp (drivers tend to place in the position). Second is to partition a season into two time intervals in which constructor skill is estimated: pre and post summer break. Constructors tend to use the month long summer break to make substantial upgrades to their cars. In the 2023 season, a sharp decline in the performance of Aston Martin occured after summer break when Mercedes and Ferrari made medium-sized upgrades to their cars, while Mclaren made substantial upgrades to their cars.

6 Code

The code used for this paper is available at `https://github.com/Meromorphics/f1project`.

References

[1] Drivers, teams, cars, circuits and more - everything you need to know about formula 1. https://www.formula1.com/en/latest/article.drivers-teams-cars-circuits-and-more-everything-you-need-to-know-about.7iQfL3Rivf1comzdqV5jwc.html. Accessed: 2023-12-14.

[2] Ergast developer api. https://ergast.com/mrd/.

[3] F1 drivers still not won over by sprint races but most prefer reduced practice. https://www.racefans.net/2023/11/02/f1-drivers-still-not-won-over-by-sprint-races-but-most-prefer-reduced-practice/. Accessed: 2023-12-14.

[4] F1 sprint races in 2023: How does it work and when is it happening? https://www.autosport.com/f1/news/f1-sprint-races-in-2023-how-does-it-work-and-when-is-it-happening/10488355/. Accessed: 2023-12-14.

[5] Fastf1. https://docs.fastf1.dev/.

[6] Getting started with bayesian statistics using stan and python. https://bob-carpenter.github.io/stan-getting-started/stan-getting-started.html#pragmatic-bayesian-statistics. Accessed: 2023-12-14.

[7] How to win in formula one: is it the driver or the car? https://thecorrespondent.com/642/how-to-win-in-formula-one-is-it-the-driver-or-the-car. Accessed: 2023-12-14.

[8] Aaron Anderson. Maximum likelihood ranking in racing sports. *Applied Economics*, 46:1778–1787, 02 2014.

[9] Aaron Anderson. A monte carlo comparison of alternative methods of maximum likelihood ranking in racing sports. *Journal of Applied Statistics*, 42:1–17, 02 2015.

[10] Andrew Bell, James Smith, Clive Sabel, and Kelvyn Jones. Formula for success: Multilevel modelling of formula one driver and constructor performance, 1950-2014. *Journal of Quantitative Analysis in Sports*, 12:99–112, 06 2016.

[11] Bob Carpenter, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of statistical software*, 76(1), 2017.

[12] Mark Glickman and Jonathan Hennessy. A stochastic rank ordered logit model for rating multi-competitor games and sports. *Journal of Quantitative Analysis in Sports*, 11, 01 2015.

[13] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57:97–109, 1970.

[14] Erik-Jan Kesteren and Tom Bergkamp. Bayesian analysis of formula one race results: Disentangling driver skill and constructor advantage, 03 2022.

[15] Thanh Dung Le, Rita Noumeir, Huu Luong Quach, Ji Hyung Kim, Jung Ho Kim, and Ho Min Kim. Critical temperature prediction for a superconductor: A variational bayesian neural network approach. *IEEE Transactions on Applied Superconductivity*, 30(4):1–5, 2020.

[16] Aki Vehtari, Jonah Gabry, Mans Magnusson, Yuling Yao, Paul-Christian Bürkner, Topi Paananen, and Andrew Gelman. loo: Efficient leave-one-out cross-validation and waic for bayesian models, 2023. R package version 2.6.0.

[17] Aki Vehtari, Andrew Gelman, and Jonah Gabry. Practical bayesian model evaluation using leave-one-out cross-validation and waic. *Statistics and Computing*, 27:1413–1432, 2017.

7 Appendix

In this appendix, we include all visual posterior predictive inference graphs. See sections 3.3.2 and 4.2.3 for more details.

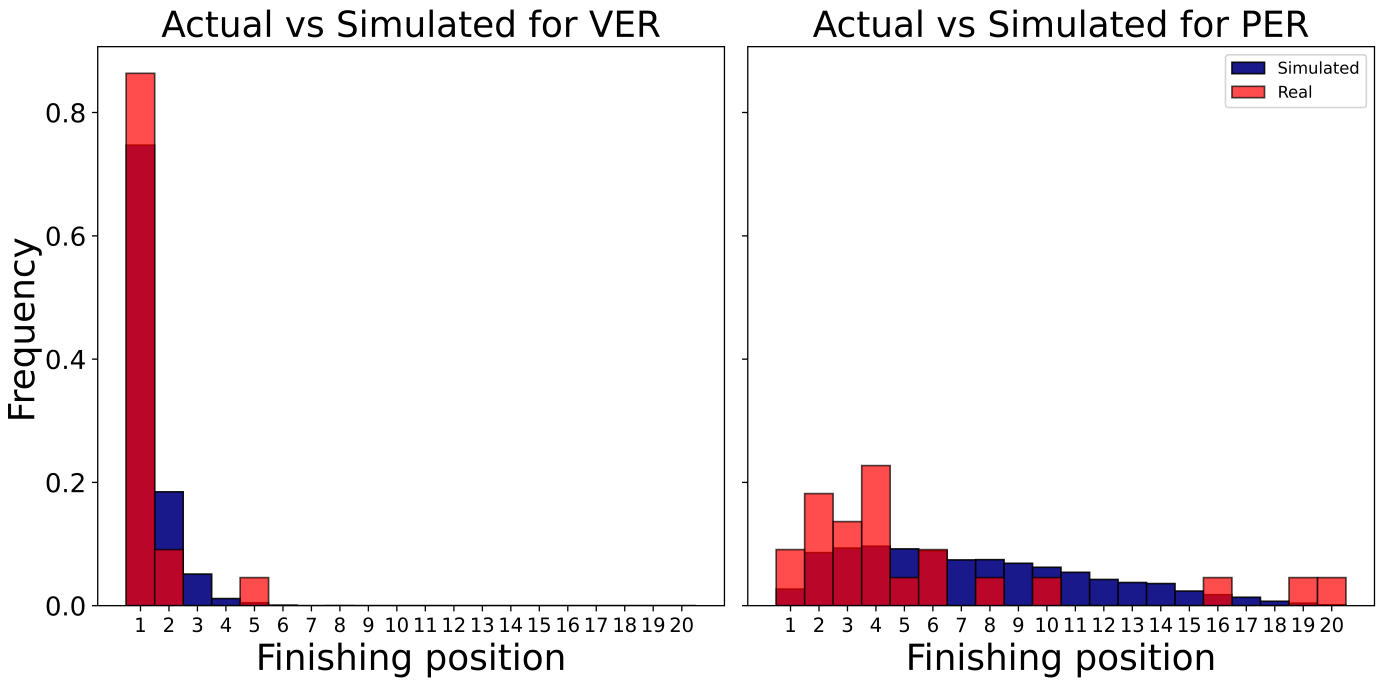

Figure 10: Simulated compared to actual finishing position results for the drivers on team Redbull: Verstappen and Perez

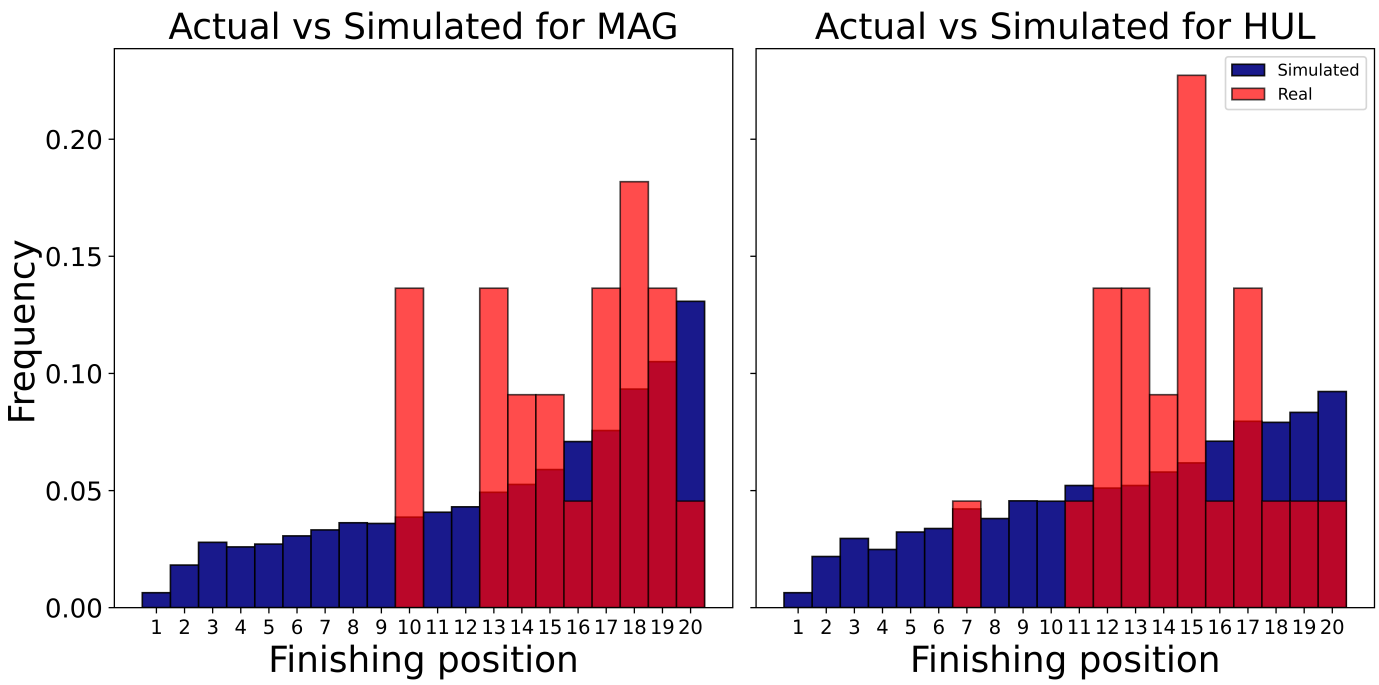

Figure 11: Simulated compared to actual finishing position results for the drivers on team Haas: Magnussen and Hulkenberg

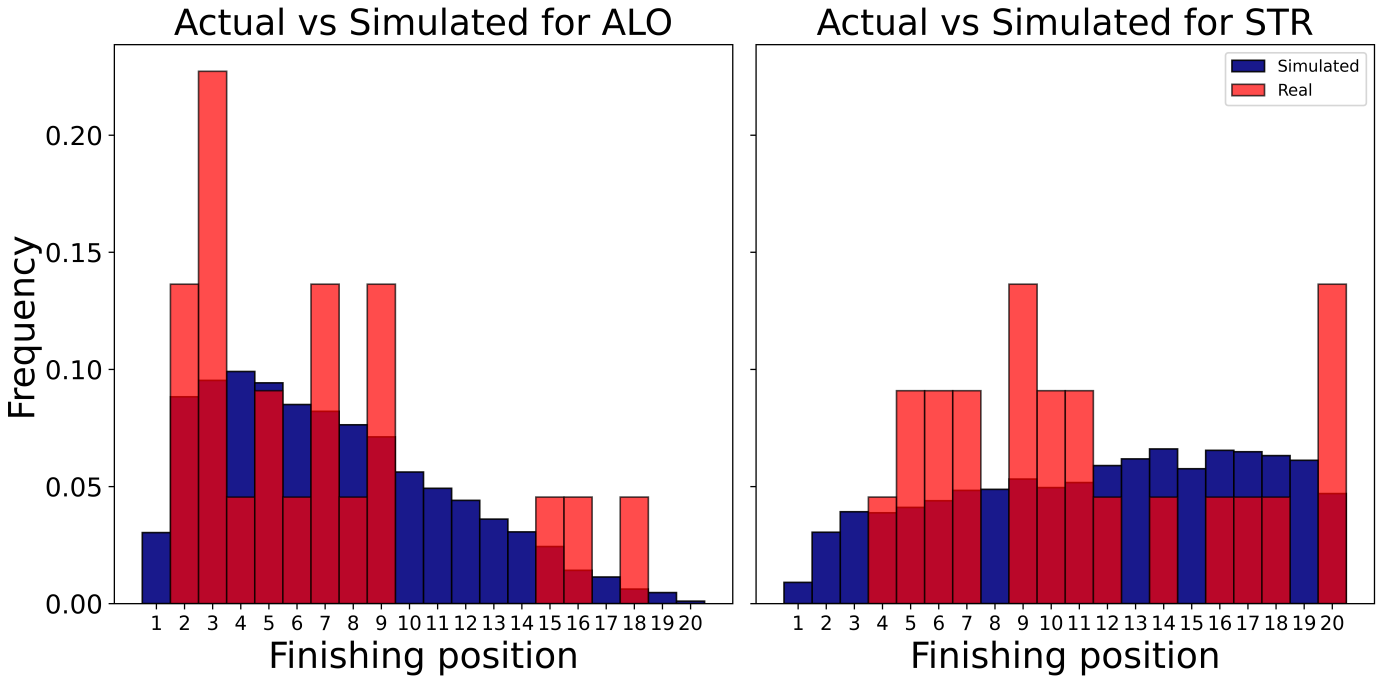

Figure 12: Simulated compared to actual finishing position results for the drivers on team Aston Martin: Alonso and Stroll

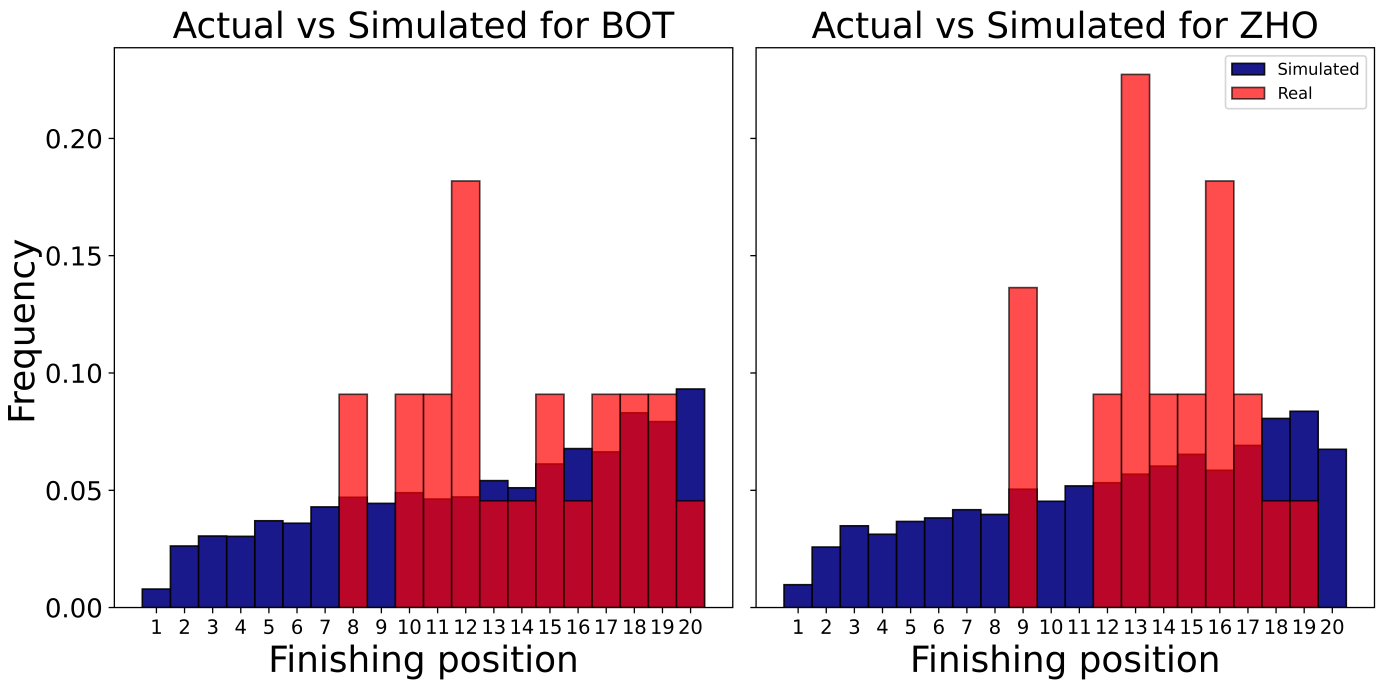

Figure 13: Simulated compared to actual finishing position results for the drivers on team Alfa Romeo: Bottas and Zhou

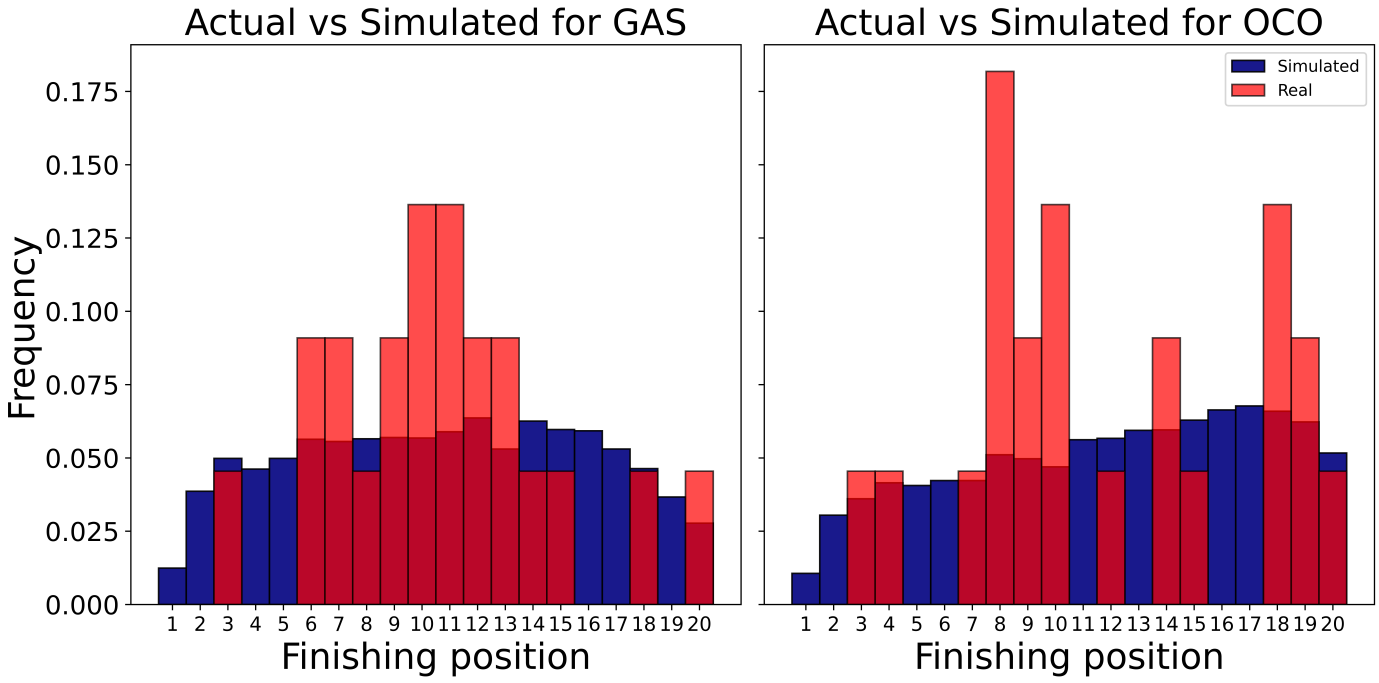

Figure 14: Simulated compared to actual finishing position results for the drivers on team Alpine: Gasly and Ocon

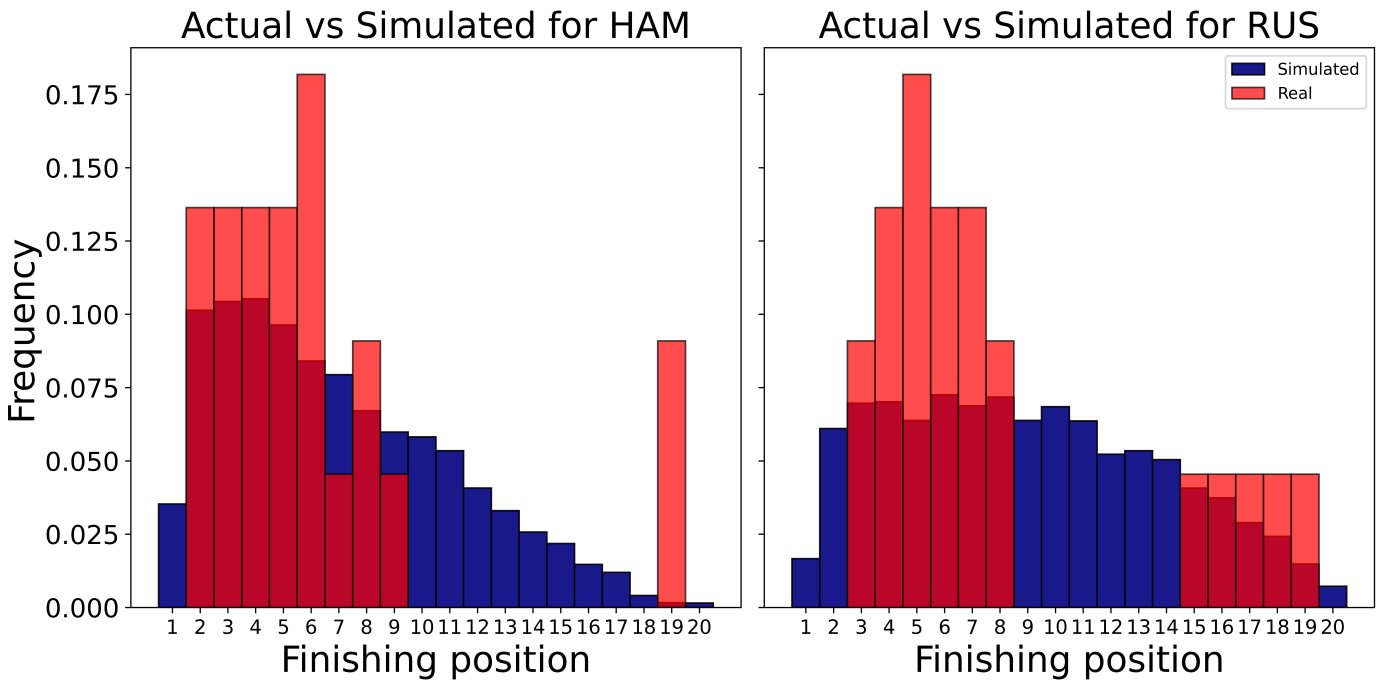

Figure 15: Simulated compared to actual finishing position results for the drivers on team Mercedes: Hamilton and Russell

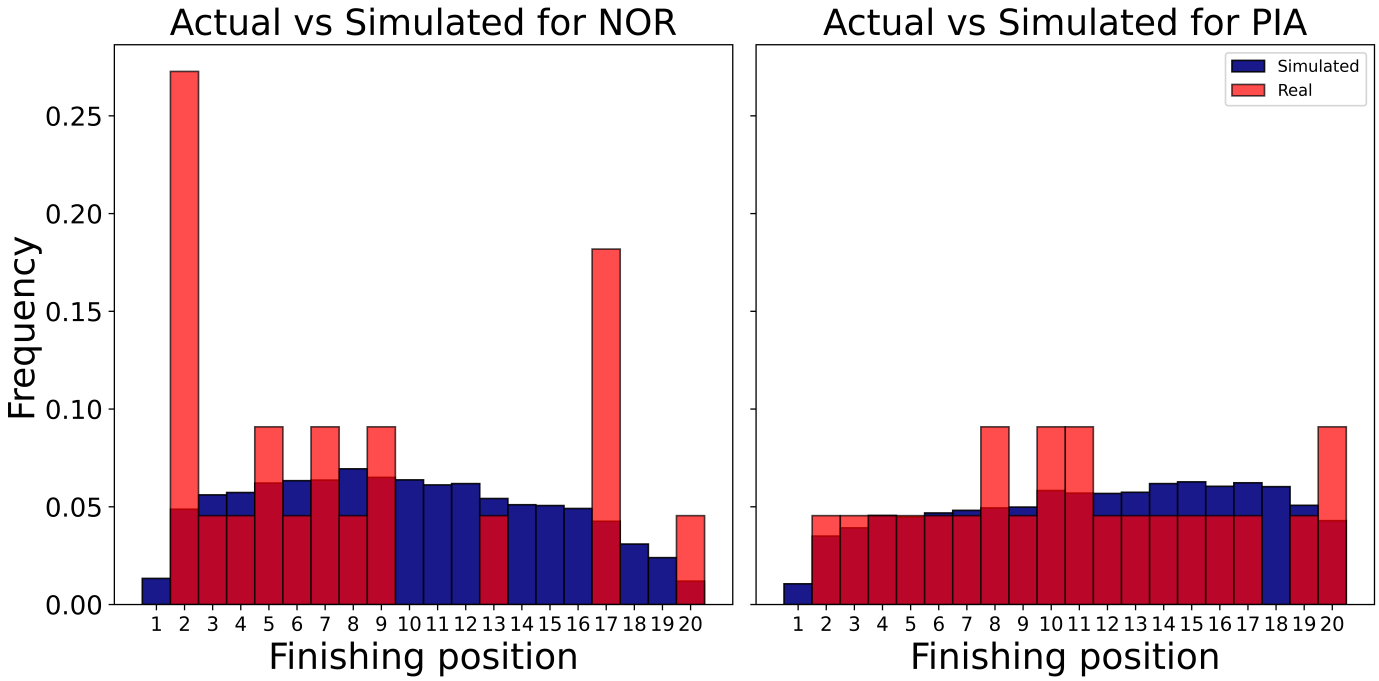

Figure 16: Simulated compared to actual finishing position results for the drivers on team Mclaren: Norris and Piastri

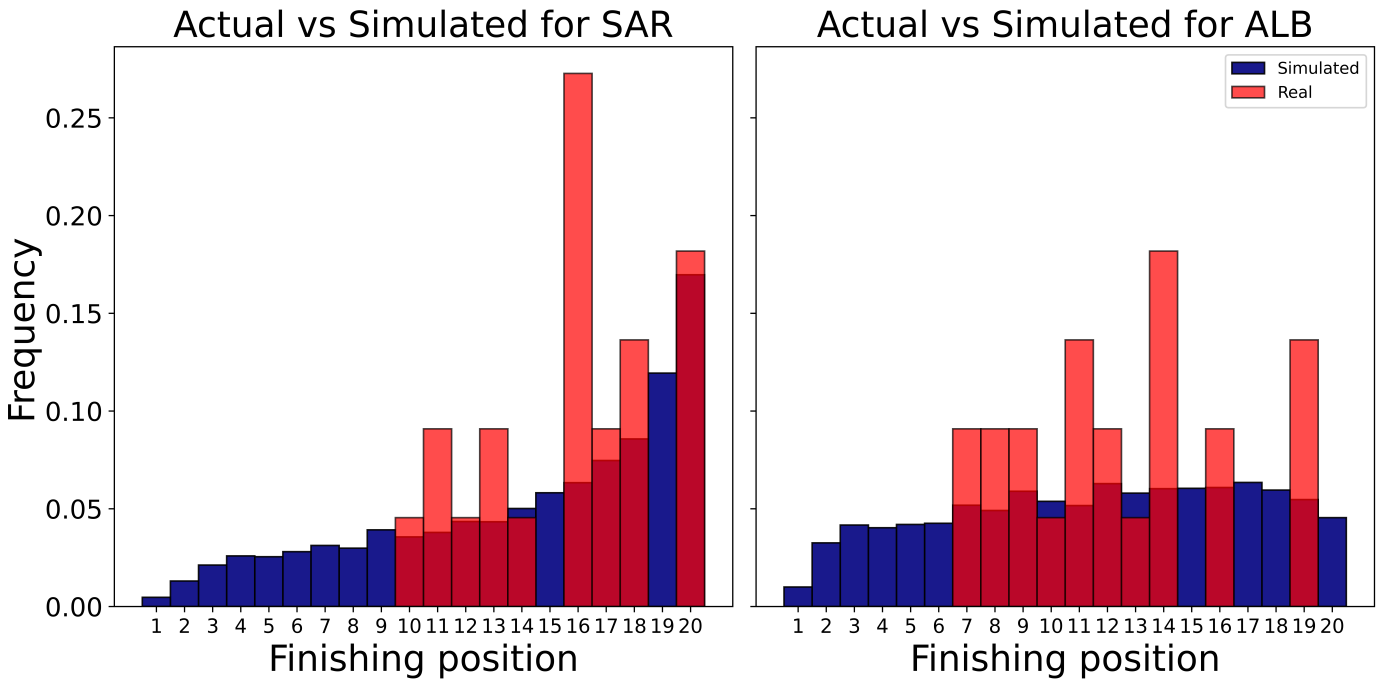

Figure 17: Simulated compared to actual finishing position results for the drivers on team Williams: Sargeant and Albon

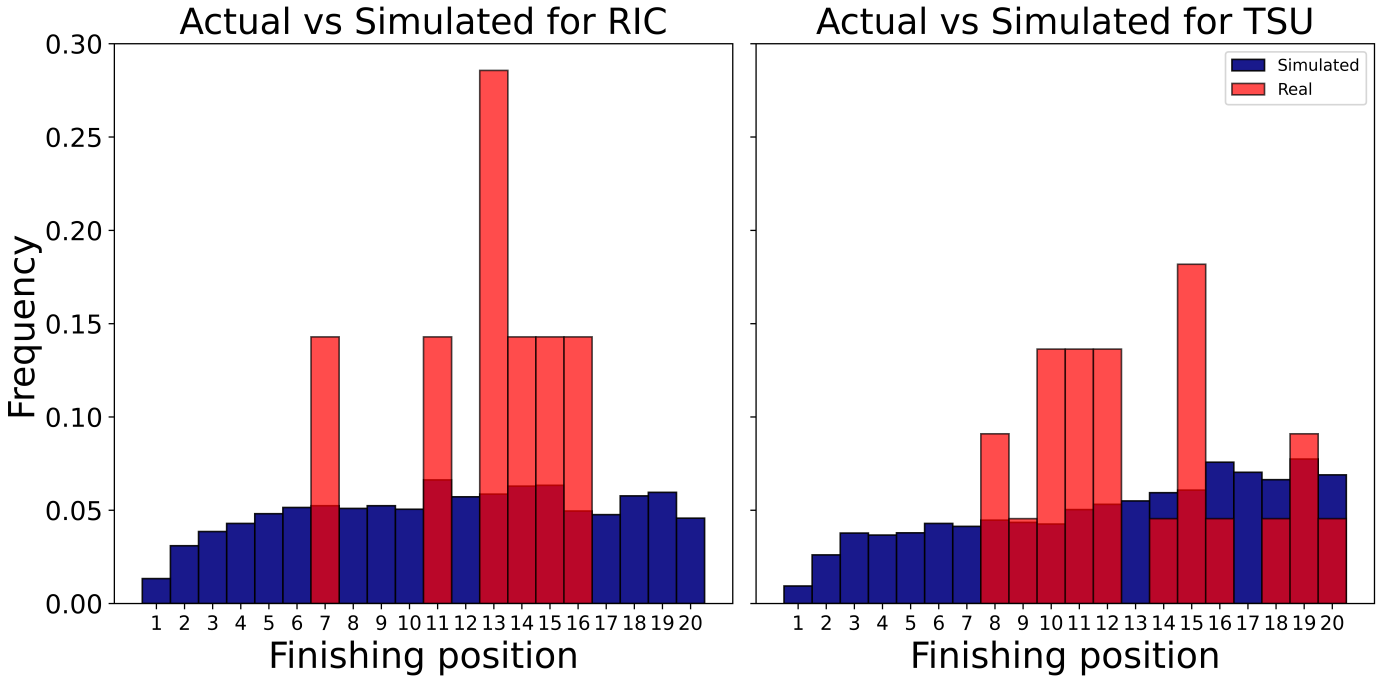

Figure 18: Simulated compared to actual finishing position results for season finishing drivers on team Alpha Tauri: Ricciardo and Tsunoda

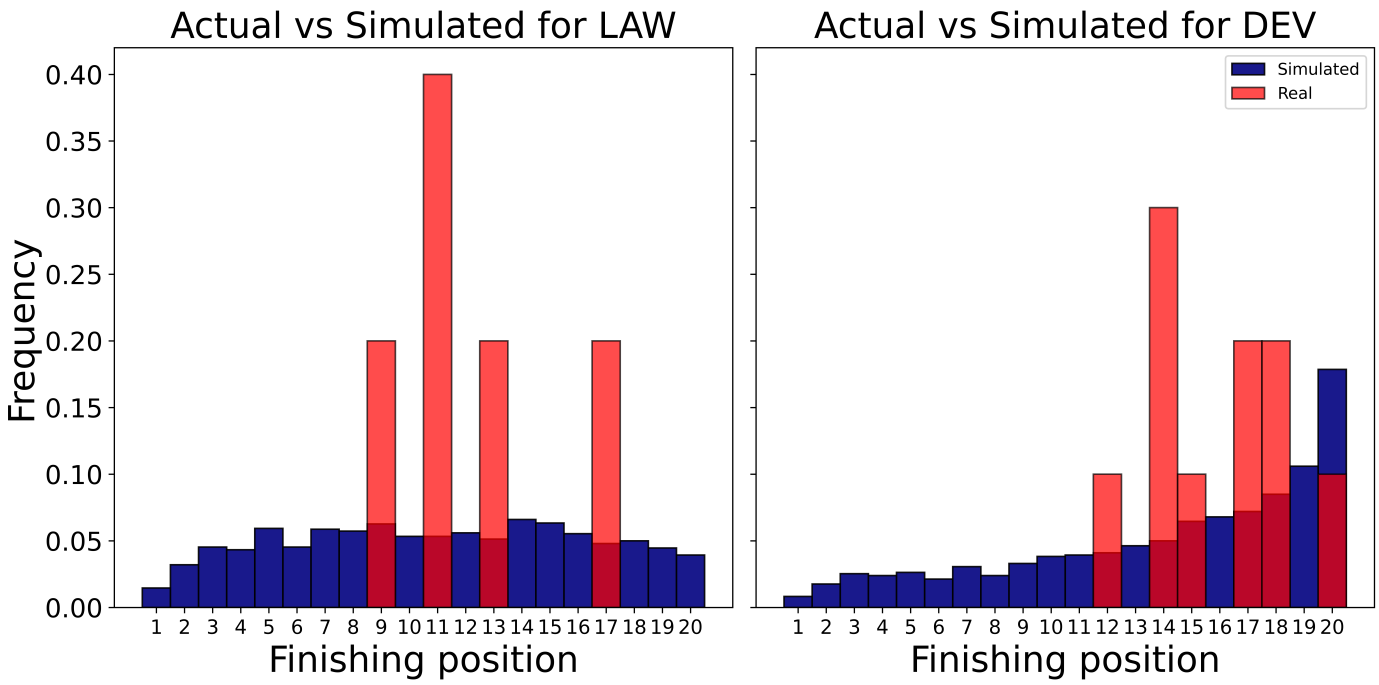

Figure 19: Simulated compared to actual finishing position results for the temporary drivers on team Alpha Tauri: Lawson and Devries