Prediction model for the PGA Tour

A Bayesian method

Kári Rögnvaldsson (kar26@hi.is)

Final project in STÆ528M

Instructor: Birgir Hrafnkelsson
November 26, 2021

HÁSKÓLI ÍSLANDS

1 Introduction

> Competitive golf is played mainly on a five-and-a-half-inch course, the space between your ears.
>
> *Bobby Jones*

Golf is a sport played by millions of people around the world. It is a sport circled around competing with yourself to get better every time you play and one needs a very strong mental mindset to be able to succeed. The goal of golf is to put a small golf ball into a hole in as few strokes as possible. A round of golf is traditionally eighteen holes and a professional tournament of golf traditionally consists of four rounds that are played from Thursday thru Sunday. The PGA Tour (Professional Golfers Association) is the largest stage for competitive golf in the world, where elite golfers compete with each other to become the best in the world. Almost every weekend, there is a tournament that takes place on a course somewhere in the world, most often in the United States. In the tournaments, around 120 players bring their best in trying to come out on top.

There are extremely many factors that make a tournament winner, some of the most important are how the course fits the golfer and the player's physical and mental form. In this project, we aim to model golfer's performance and determine winners and top players in tournaments. More specifically, what is the probability that a given player is the winner of a given tournament? What is the probability that he will place in the top 10, top 25, and what is the probability that is cut out of the tournament after two days (misses the cut)? To answer these questions, we will use a hierarchical linear model based on the player.

2 Prevoius work

Some people have been experimenting with golf predictive models, such as Courchene (2018). The model in this study is inspired by that model, but differs in many respects. Others, like Egidi and Gabry (2017) have experimented with hierarchical models for sports, which is very interesting. A description of the Bayesian hierarchical models used and many other techniques are found in Gelman et al. (2021)

3 Data

3.1 The dataset

The project consists of a player based dataset. The dataset is obtained from the following website: `https://www.advancedsportsanalytics.com/pga-raw-data`. It consists of roughly 33,000 data points for almost 500 players on the PGA Tour from 2015–2021. The most important columns of the data are the following:

`player`: Player's full name
`tournament_name`: Name of the tournament that is recorded in the data point.
`course`: The name of the course played in the tournament.
`date`: Date of the tournament.
`pos`: Final position of the player in the tournament.
`hole_par`: The par of the course.
`strokes` Number of strokes for the player in the tournament.
`made_cut`: A value, either 1 or 0, that indicates if a player made the cut.

A "missed cut" is when a player does not qualify high enough the first two days of the tournament.

3.2 Data manipulation

A player's performance relative to the other players that are competing is of most interest and therefore it is not needed to estimate the total number of strokes for each player. There is a massive difference between

courses in difficulty, so we need to account for a course being hard/easy. For each tournament, a statistic that we will here call "standardized strokes gained", is calculated as follows: For every tournament a player plays, the average number of strokes per round is calculated, let us call that S_{ij} where i denotes player i and j denotes tournament j, the quantity SG (strokes gained) is calculated

$$SG_{ij} = \frac{S_{ij} - \mu_j}{s_j},$$

where

$$\mu_j = \frac{1}{n}\sum_k S_{kj} \quad \text{and} \quad s_j^2 = \frac{1}{n-1}\sum_k (S_{kj} - \mu_j)^2,$$

i.e. the S_{ij} terms are normalized to have mean zero and standard deviation 1 over all i for fixed tournament j. This quantity is a measure of a player's performance relative to the field and will be used to predict.

There is also some time element in a player's performance on the field according to the graph in section 3.3. If we denote the day of of tournament j for player i as t^*_{ij}, where t^*_{ij} is the number of days since the first tournament in the dataset was played, we let

$$t_{ij} = 2\frac{t^*_{ij} - t_0}{t^*_{\max}},$$

where $t^*_{\max}$ is the latest date in the data and t_0 is the median value of t^*. The quantity t^*_{ij} has therefore been scaled such that $t_{ij} \in [-1; 1]$.

3.3 Data exploration

When SG_{ij} is plotted for a player as a function of time, we obtain figures like Figures 1. Figure 1 shows the progress for Jordan Spieth, a top class golfer, along with a smooth fit of the data to visualize the trend more. We see that there is a time trend in the data and Jordan Spieth is extremely interesting because of the dip in the trend. He was the top ranked golfer in the world in 2016, then he underperformed from 2018-2020 and in the year 2021, he has performed better again. This is apparent in the smooth curve fitted to the data. That curve is what we want to model. If we look at empirical density of the SG variable on a player basis, we note a left tail in the data (see Figures 2, 3, 4 and 5). This might be because golf is a sport that is extremely hard to play at a high level and very easy to play bad. Therefore, very bad scores are more likely than very good scores and thus, the distributions becomes left tailed.

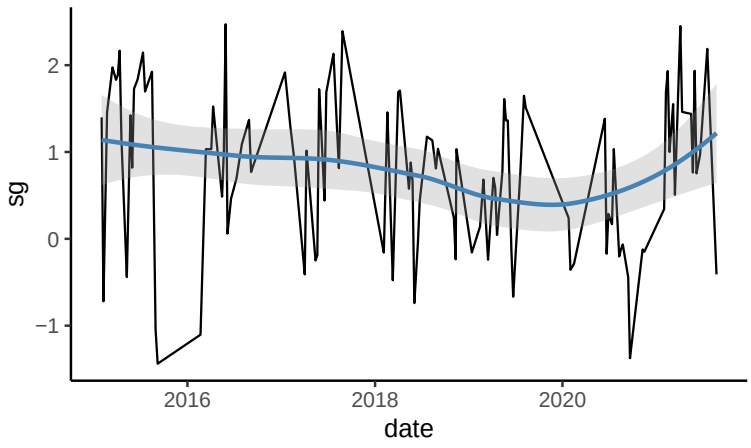

Figure 1: Form trend of Jordan Spieth

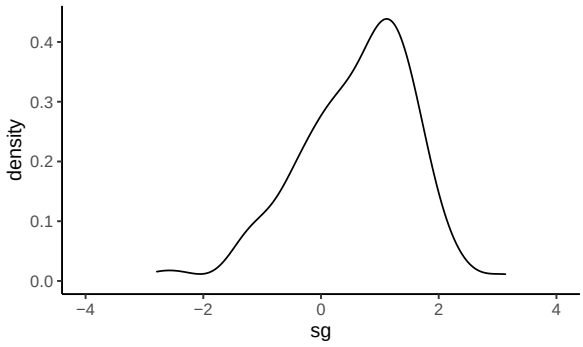

Figure 2: Empirical density for standardized strokes gained for Justin Thomas

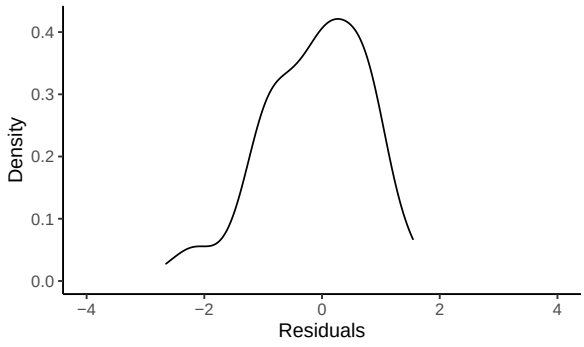

Figure 3: Empirical density for standardized strokes gained for Jordan Spieth

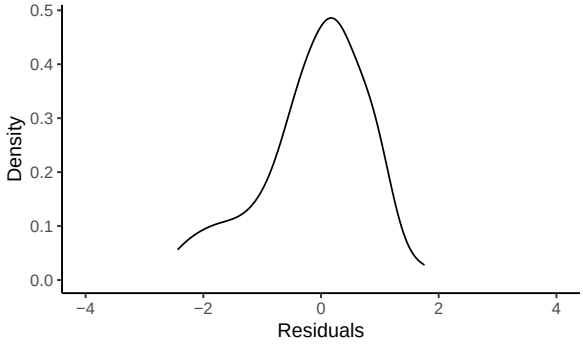

Figure 4: Empirical density for standardized strokes gained for Jon Rahm

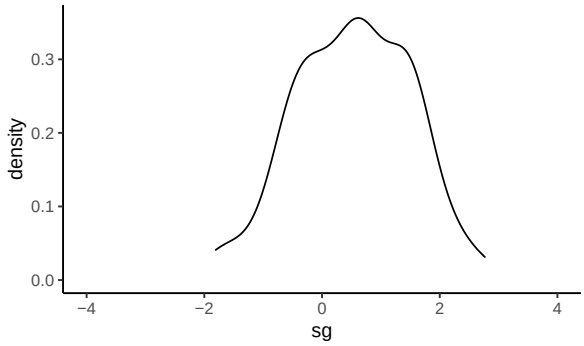

Figure 5: Empirical density for standardized strokes gained for Bryson DeChambeau

3.4 Corrupt data

Some of the tournaments in the date were not fully recorded. From some, it was quite obvious that there were players that played in the tournament that were not in the data, since all or almost all the players in the data made the cut in a specific tournament, which is not realistic. Therefore, the tournaments with over 80% of the players making the cut were removed. This choice was arbitrary, but it is unrealistic that more than 80% of the players make the cut. If too many players in a tournament in the data make the cut it is highly likely that the data does not show the full competition and we thus do not get a correct image of the relative performance. It is usually around the first 70 players that make the cut after two days, of around 120 total players, which makes 58%.

4 Statistical models

In this section, we will present the model that is used in the study.

4.1 Model introduction

As stated in the data section, it is aimed at to predict the number of standardized shots gained of the players competing in a given tournament. Since the data is left-tailed, we model the SG variable as a Mirror-Gumbel variable. To model the time, we introduce a 3rd degree polynomial to describe the evolution of the player in time. Shots gained for player i in tournament j therefore becomes

$$SG_{ij} \sim \text{Mirror-Gumbel}(\alpha_{0,i} + \alpha_{1,i}t + \alpha_{2,i}t^2 + \alpha_{3,i}t^3, e^{\beta_i})$$

where $\alpha_{k,i}$ is parameter k in the 3rd degree polynomial for the expected value of SG for player i and e^{β_i} is the dispersion parameter of the Mirror-Gumbel distribution for player i. We need to use the Mirror-Gumbe since the standard Gumbel distribution is right tailed and this dataset is left tailed. Note that the PDF for the mirrored gumbel distribution is

$$p(y|\mu,\beta) = \frac{1}{\beta}\exp(-(z+\exp(-z)))$$

where $z \coloneqq \frac{\mu-y}{\beta}$. The CDF becomes

$$F(y|\mu,\beta) = e^{e^{-z}}$$

and the inverse CDF (for sampling from the distribution with inverse CDF methods) becomes

$$F^{[-1]}(x|\mu,\beta) = \mu + \beta\log(-\log x).$$

Note that the regular Gumbel distribution is the same but with $z \coloneqq \frac{y-\mu}{\beta}$.

We have 5 variables for each of the players. Since we expect all these parameters to be similarily distributed, we assume

$$\begin{aligned}
&\alpha_{k,i} \sim \mathcal{N}(\mu_{\alpha_k}, \sigma^2_{\alpha_k}),\\
&\beta \sim N(\mu_\beta, \sigma^2_\beta)
\end{aligned}$$

where we assign the weakly informative hyperpriors

$$\begin{aligned}
&\mu_{\alpha_k} \sim \mathcal{N}(0, 5^2),\\
&\mu_\beta \sim \mathcal{N}(0, 5^2),\\
&\sigma^2_{\alpha_i} \sim \text{Inv–}\chi^2(1, 5^2),\\
&\sigma^2_\beta \sim \text{Inv–}\chi^2(1, 5^2).
\end{aligned}$$

The model hierarchy becomes as follows:

Hyperparameters:

$$\begin{aligned}
&\mu_{\alpha_k} \sim \mathcal{N}(0, 5^2),\\
&\mu_\beta \sim \mathcal{N}(0, 5^2),\\
&\sigma^2_{\alpha_i} \sim \text{Inv–}\chi^2(1, 5^2),\\
&\sigma^2_\beta \sim \text{Inv–}\chi^2(1, 5^2).
\end{aligned}$$

Latent level:

$$\begin{aligned}
&\alpha_k|\mu_{\alpha_k}, \sigma_{\alpha_k} \sim \mathcal{N}(\mu_{\alpha_k}, \sigma^2_{\alpha_k})\\
&\beta_i|\mu_\beta, \sigma_\beta \sim \mathcal{N}(\mu_\beta, \sigma^2_\beta)
\end{aligned}$$

Data level:

$$SG_{ij}|\alpha_{0,i}, \alpha_{1,i}, \alpha_{2,i}, \alpha_{3,i}, \beta_i \sim \text{Mirror-Gumbel}(\alpha_{0,i} + \alpha_{1,i}t + \alpha_{2,i}t^2 + \alpha_{3,i}t^3, e^{\beta_i})$$

Now we introduce vector/matrix notation. We let

$$\begin{aligned}
&\boldsymbol{SG} = (SG_{1,1}, SG_{1,2}, \ldots, SG_{1,n_1}, \ldots, SG_{Q,1}, \ldots, SG_{Q,n_Q})^T,\\
&\boldsymbol{\alpha_k} = (\alpha_{k,1}, \ldots, \alpha_{k,Q})^T \quad \text{for} \quad k = 0, 1, 2, 3,\\
&\boldsymbol{A} = \begin{bmatrix}\boldsymbol{\alpha}_0 & \boldsymbol{\alpha}_1 & \boldsymbol{\alpha}_2 & \boldsymbol{\alpha}_3\end{bmatrix}\\
&\boldsymbol{\beta} = (\beta_1, \ldots, \beta_Q)^T.
\end{aligned}$$

Finally, we set

$$z_{ij} := \frac{(\alpha_{0,i} + t_{ij}\alpha_{1,i} + t_{ij}^2\alpha_{2,i} + t_{ij}^3\alpha_{3,i}) - SG_{ij}}{e^{\beta_i}}$$

This is a variable to ease notation for the mirror Gumbel distribution. Note that this is the negative of the regular z value for the Gumbel distribution. This is because we are mirroring the distribution. The posterior distribution of the parameters now becomes:

$$\begin{aligned}
&p(\boldsymbol{A}, \boldsymbol{\beta}, \mu_{\alpha_k}, \sigma^2_{\alpha_k}, \mu_\beta, \sigma^2_\beta | \boldsymbol{SG}) \\
&\propto p(\boldsymbol{SG}|\boldsymbol{A}, \boldsymbol{\beta}) \left(\prod_{k=0}^{3} p(\boldsymbol{\alpha}_k | \mu_{\alpha_k}, \sigma^2_{\alpha_k}) p(\mu_{\alpha_k}) p(\sigma^2_{\alpha_k}) \right) p(\boldsymbol{\beta}|\mu_\beta, \sigma^2_\beta) p(\mu_\beta) p(\sigma^2_\beta) \\
&\propto \left(\prod_{i=1}^{Q} \prod_{j=1}^{n_i} \frac{1}{e^{\beta_i}} \exp(-z_{ij} - \exp(-z_{ij})) \right) \left(\prod_{k=0}^{3} \exp\left(-\frac{\mu^2_{\alpha_k}}{2 \cdot 5^2} \right) \text{Inv–}\chi^2(\sigma^2_{\alpha_k} | 1, 5^2) \prod_{i=1}^{Q} \frac{1}{\sigma_{\alpha_k}} \exp\left(-\frac{(\alpha_{k,i} - \mu_{\alpha_k})^2}{2\sigma^2_{\alpha_k}} \right) \right) \\
&\times \exp\left(-\frac{\mu^2_\beta}{2 \cdot 5^2} \right) \text{Inv–}\chi^2(\sigma_\beta | 1, 5^2) \prod_{i=1}^{Q} \frac{1}{\sigma_\beta} \exp\left(-\frac{(\beta_i - \mu_\beta)^2}{2\sigma^2_\beta} \right).
\end{aligned}$$

From the posterior density, it is easy to obtain the conditional densities for the Gibbs sampler.

For comparison, we will also look at the model where no time factor is included, i.e. $\alpha_{1,i} = \alpha_{2,i} = \alpha_{3,i} = 0$, let us call that the intercept only model, and a baseline model of the form $SG_{ij} \sim \text{Mirror-Gumbel}(\mu, \beta)$, i.e. a model with no hierarchy.

To summarize, Table 1 shows the parameters along with description of each parameter.

Table 1: A list and description of the parameters of the model

Parameter	**Description**
$\alpha_{k,i}$	Coefficient in the 3rd degree polynomial of the form of player i
β_i	The log of the dispersion of shots gained for player i.
μ_{α_k}	Hyperparameter, mean of μ_i over all i.
$\sigma^2_{\alpha_k}$	Hyperparameter, the variance of μ_i over all i.
μ_β	Hyperparameter, the mean of β_i over all i.
σ^2_β	Hyperparameter, the variance of β_i over all i.

Unfortunately, the $\alpha_{k,i}$ terms are not very interpretable, but $\alpha_{0,i}$ should denote the expected value of SG_{ij} of the player when $t_{ij} = 0$, in the middle of the time interval we are looking at.

4.2 Model implementation

To implement the models, the software Stan was used (Carpenter et al., 2017). Stan is a programming language based on C++ for Bayesian statistics put together by multiple people, most notably Andrew Gelman, professor of statistics at Columbia University. It is a way to but together Bayesian models in a simple, elegant manner (see code in appendix). It uses Hamiltonian Monte Carlo simulation to sample from the posterior distributions, a method that uses Hamiltonian dynamics to generate samples that have lower autocorrelation than one would get in a Metropolis algorithm, which leads to higher acceptance rate and fewer samples needed for computation. For the model, we implemented 4 chains, each including 500 burn-in iterations and 500 sampling iterations, 2000 draws in total.

4.3 Model performance and comparison

To compare the models in the study, the DIC (Deviance information criterion) along with other statistics. To evaluate the fit quality of the full model, a Bayesian p-value was calculated for each player using the

Anderson-Darling statistic:

$$T(y,\theta) = -n - \frac{1}{n}\sum_{i=1}^{n}(2i-1)(\log(w_i) + \log(1-w_{n-i+1}))$$

where the vector $\boldsymbol{w} = (w_1, \ldots, w_{n_j})$ is the vector of ordered values of $F(SG_{ij}|\alpha_{0,i,k} + t_{ij}\alpha_{1,i,k} + t_{ij}^2\alpha_{2,i,k} + t_{ij}^3\alpha_{3,i,k}, \beta_{i,k})$ for player i, $\alpha_{t,i,k}$ is the k-th draw from the posterior distribution of $\alpha_{t,i}$ and $\beta_{i,k}$ is the k-th draw from the posterior distribution of β_i.

To evaluate the predictive performance of the full model, the 12 tournaments that were left out in the fitting (in July 2021) were simulated and the proportion of correct predictions of the winner, top 10, top 25 and made cut were calculated. They were simulated in such a way that for tournament j, SG_{ij} was sampled in each iteration using the posterior draws of $\alpha_{i,k}$ and β_i from the model. The predicted position of a player was the win percentage of the player in the simulations. Only players that were competing in the tournament were available in the simulations. To assess the quality of the predictions, a random model was also simulated for each of the 12 tournaments in a similar fashion. The main difference in the simulations of the models was that there was only one result in the full model's performance (summary over the 2000 simulations), as in practice we would only look at the predicted outcome for our prediction for a tournament. For the random model, the average number of correct guesses was calculated over all of the 2000 simulations. The random model is in this sense interpreted as the expected performance of a random guess

5 Results

In this section, we will describe the results of the model.

5.1 Model comparison

Table 2 shows the model statistics for the model in the study. We see that the effective number of parameters for the full model reduces by around 600, which means that the hierarchy is having an effect. The same goes for the intercept only model. We also notice that the full model performs best in terms of DIC by a margin of around 600, the intercept only model performs second best, and both of these models outperform the baseline model by more than 3000, which is a large margin. For the rest of the results, we will look at the results for the full model (the one using the 3rd degree polynomial fit for time). Table 3 shows the estimates of the hyperparameters of the model (note that β_i is the log of the dispersion of the model).

Table 2: Model statistics for the models in the study

Statistic	**Baseline model**	**Intercept only model**	**Full model**
$D_{\hat{\theta}}$	75952	71578	69664
$D_{\theta_{\text{avg}}}$	75954	72238	70961
Total no. of parameters	2	746	1871
p_D	1.95	659.59	1297.53
DIC	75956	72897	72259

Table 3: Posterior intervals for the hyperparameters of the model

Parameter	Lower 95%	Posterior median	Upper 95%
μ_{α_0}	0.32	0.37	0.42
σ_{α_0}	0.45	0.48	0.52
μ_{α_1}	-0.13	-0.07	0.00
σ_{α_1}	0.42	0.46	0.51
μ_{α_2}	-0.22	-0.15	-0.09
σ_{α_2}	0.45	0.49	0.54
μ_{α_3}	-0.10	0.00	0.09
σ_{α_3}	0.49	0.54	0.61
μ_{β}	-0.27	-0.24	-0.21
σ_{β}	0.28	0.30	0.33

5.2 How well does the theoretical distribution fit the data?

Figure 6 shows the expected value of SG_{ij} w.r.t. time for Jordan Spieth and Figure 7 shows the same for Jon Rahm. Jordan Spieth's curve seems to follow a similar trend to Figure 1 and captures it well. The trend for Jon Rahm is also well described by the model.

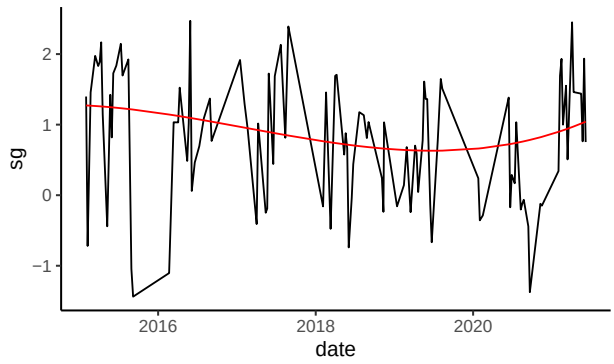

Figure 6: Fitted form of Jordan Spieth

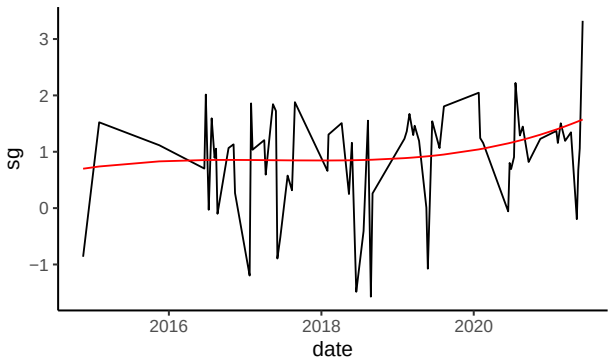

Figure 7: Fitted form of Jon Rahm

Figures 8, 9, 10 and 11 show a representation of the posterior predictive density versus empirical distribution of the residuals of the model for the players we looked at in Section 3.3 (model exploration). We notice that the model seems to fit very well to Thomas's data and well for Rahm, Spieth and DeChambeau. For a full posterior predictive check, one would have to use many samples and smoothing but this method gives an idea of the posterior predictive density.

Figure 12 is a histogram the p-values of the players in the model. Only 1 player has a p-value outside $[0.025; 0.975]$, which indicates that the model describes the data adequately well.

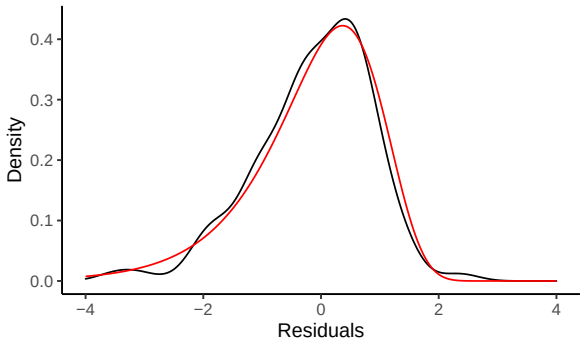

Figure 8: A representative for the posterior predictive distribution for the residuals standardized strokes gained for for Justin Thomas

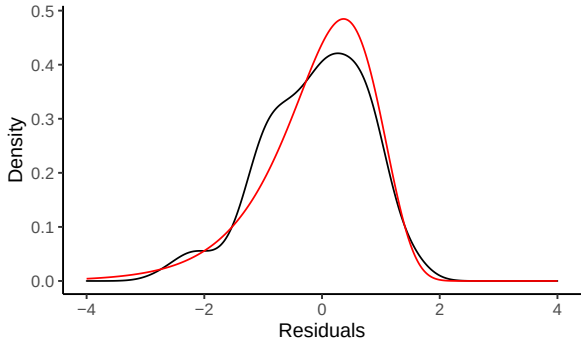

Figure 9: A representative for the posterior predictive distribution for the residuals standardized strokes gained for for Jordan Spieth

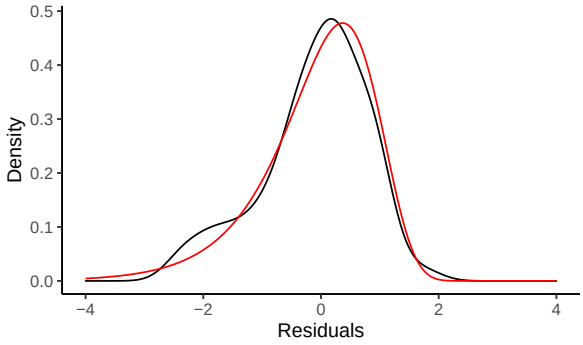

Figure 10: A representative for the posterior predictive distribution for the residuals standardized strokes gained for for Jon Rahm

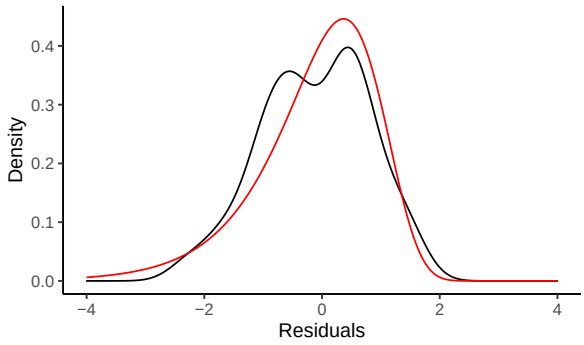

Figure 11: A representative for the posterior predictive distribution for the residuals standardized strokes gained for Bryson DeChambeau

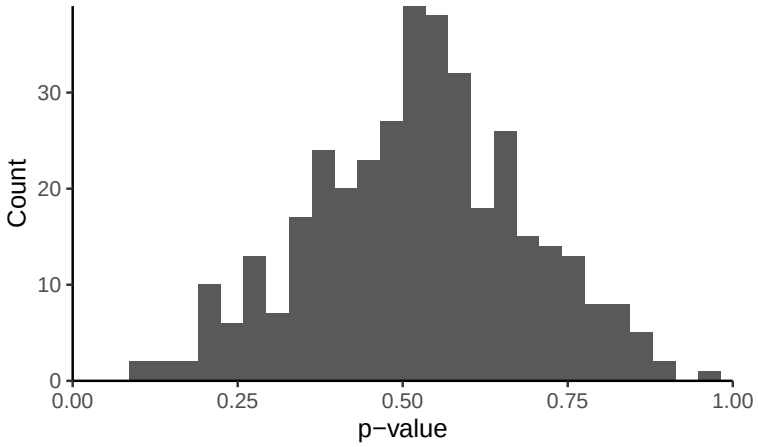

Figure 12: A histogram of the Bayesian p-values for the players in the model.

5.3 Posterior simulation - tournament results

The 12 tournaments that were left out in the data were simulated according to the procedure described in section 2. Here we present the results for three of these tournaments for the full model along with a performance estimate of a random guessing model. These tournaments were chosen since the U.S. Open and The Open are two of the four so-called "major" tournaments every year and The Rocket Mortgage Classic was chosen to also showcase a smaller tournament. Table 4 shows the proportion of the correct guesses of the model for the tournaments for winners, top 10, top 25 and made cut. Table 5 shows the same for the random model. We note that the full model performs quite a lot better than the random model in all aspects.

Table 4: Performance of the full model on three tournaments that were left out in the fitting.

Full model	**U.S. Open**	**The Open**	**Rocket Classic**
Correct winner	100%	0%	0%
Correct top 10	40%	50%	0%
Correct top 25	36%	36%	40%
Correct made cut	51%	54%	57%

Table 5: Model performance for a random guess of player position.

Random guess	**U.S. Open**	**The Open**	**Rocket Classic**
Correct winner	0.6%	1.3%	0.7%
Correct top 10	10.3%	10.7%	7.7%
Correct top 25	22.8%	22.8%	20.9%
Correct made cut	50.5%	52.6%	49.7%

6 Discussions and conclusions

Golf is a sport that is very hard to predict, as there are so many players competing at each tournament and because of the multiple factors that make a winner. The goal of this study was to be able to predict for winners and top players in tournaments on the PGA tour. To do that, we have come up with a measure of a player's quality in time and space, by using performance relative to the other players that are playing a given tournament.

By looking at the results, we have discovered that the model fits the data adequately well. The theoretical distributions seem to be reasonable. Of the three models that were compared, the most advanced model did the best in terms of DIC. It is very interesting to see that even though it has quite a lot more parameters than the other two models, it still performs better than the simpler models. It is convincing to look at the polynomial that is fit for the players we looked at in the study along with the Gumbel assumption on the residuals of the model. This task of modeling the performance of a golfer seems to be possible and we are able to capture some factors in a golfers performance with this method.

Some ideas that seemed convincing at first did not work. One of these was a time series based model for the SG variable. This model did not perform well, there was not enough correlation in SG close in time to create a linear model. That model also assumed normality which was not found to be reasonable.

In practice, one would like to use the model at predicting for future tournaments on the PGA Tour, in online betting or such activities (although all gambling is discouraged) or just as a hobby. By looking at the simulations introduced, we saw that the model performs quite better than a random model, but would it help a sports better win bets?. It is good at predicting top 10, top 25 but it seems that it is harder to predict who make the cut.

There are endless possibilities in trying to improve the model. It would be extremely interesting to incorporate a course fit to the player, f.ex. a player that hits it long performs well on a long course, a player that is good at playing in windy areas is good at courses by the sea etc. For this task to be completed, we would need more detailed data on a players strengths and weaknesses, however it would be possible.

References

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of statistical software*, 76(1).

Courchene, M. (2018). Data golf predictive model: methodology. `https://datagolf.com/predictive-model-methodology/`. Accessed: 2021-11-25.

Egidi, L. and Gabry, J. (2017). Bayesian hierarchical models for predicting individual performance in football (soccer). *Journal of Sports Analytics*, 3(14):143–157.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2021). *Bayesian Data Analysis*. Chapman and Hall/CRC, 2nd ed. edition.

7 Appendix

7.1 Data manipulation

```
# manipulate_data.R

library(readr)
library(dplyr)
library(here)
library(tidyr)
library(lubridate)

# It is quite unnatural that the made cut rate is above 80%
golf_dat <- read_csv(here("final_project", "Data" , "pga_raw_data.csv")) %>%
    group_by(`tournament name`, date) %>% filter(mean(made_cut) <= 0.8)

K <- 10

# Make sure at least 10 data points per player
sg_dat <- golf_dat %>%
    group_by(player) %>%
    filter(n() >= 10) %>%
    ungroup() %>%
    rename(tournament = "tournament name") %>%
    group_by(tournament, date) %>%
    mutate(average_strokes = strokes/n_rounds,
           mean_strokes = mean(average_strokes),
           sg = (mean(average_strokes)-average_strokes)/sd(average_strokes)) %>%
    filter(!is.na(sg)) %>%
    ungroup %>%
    arrange(date)

cutoff_date <- sg_dat %>% ungroup %>%
    filter(tournament == "U.S. Open", year(date) == 2021) %>%
    slice(1) %>% pull(date)

cutoff_players <- filter(sg_dat, date < cutoff_date) %>% pull(player) %>% unique

sg_dat <- sg_dat %>% group_by(player) %>%
    select(player, tournament, date, pos, sg) %>%
    filter(player %in% cutoff_players) %>%
    mutate(group = group_indices()) %>%
    mutate(t_star = as.numeric(date-min(date)),
           t = 2*(t_star-(max(t_star)-min(t_star))/2)/
               (max(t_star)-min(t_star))) %>%
    select(c(-t_star)) %>%
    ungroup

sg_dat_cutoff <- sg_dat %>% filter(date < cutoff_date)

# This is the data that the model is fitted to
sg_dat_cutoff %>% write_csv(here("final_project", "Data", "sg_dat_cutoff.csv"))
# This is the full dataset
```

```r
sg_dat %>% write_csv(here("final_project", "Data", "sg_dat.csv"))
```

7.2 Stan models

```
# baseline_model.stan

data {
  int<lower=0> N;
  vector[N] y;
}

parameters {
  real mu;
  real beta;
}

model {
  -y ~ gumbel(-mu, exp(beta));
}
```

```
# intercept_only_model.stan

data {
  int<lower=0> N; // The number of data points
  int<lower=0> Q; // The number of players;

  //Data
  vector[N] y;
  int<lower=1, upper=Q> g[N]; // Player playing given tournament
}

transformed data {
  vector[N] negative_y = -y;
}

parameters {
  vector[Q] mu;
  vector<lower=0>[Q] beta;

  real gamma_mu;
  real<lower=0> tau_mu;

  real gamma_beta;
  real<lower=0> tau_beta;
}

model {
  gamma_mu ~ normal(0, 5);
  tau_mu ~ scaled_inv_chi_square(5, 5);

  gamma_beta ~ normal(3,5);
```

```
  tau_beta ~ scaled_inv_chi_square(5,5);

  mu ~ normal(gamma_mu, tau_mu);
  beta ~ normal(gamma_beta, tau_beta);

  negative_y[1:N] ~ gumbel(-mu[g[1:N]], beta[g[1:N]]);
}
```

```
# full_model.stan

data {
  int<lower=0> N; // The number of data points
  int<lower=0> Q; // The number of players;

  //Data
  vector[N] y;

  // Covariates
  vector[N] t; // Time measured on [-1;1]
  int<lower=1, upper=Q> g[N]; // Player playing given tournament
}

transformed data {
  vector[N] negative_y = -y;

  vector[N] t_squared;
  vector[N] t_cubed;
  for(i in 1:N) {
    t_squared[i] = t[i]*t[i];
    t_cubed[i] = t_squared[i]*t[i];
  }
}

parameters {
  vector[Q] log_beta;

  matrix[Q, 4] alpha;

  vector[4] mu_alpha;
  vector<lower=0>[4] sigma2_alpha;

  real mu_beta;
  real<lower=0> sigma2_beta;
}

model {
  // Hyperpriors
  mu_beta ~ normal(1,5);
  sigma2_beta ~ scaled_inv_chi_square(1,5);

  mu_alpha ~ normal(0,5);
  sigma2_alpha ~ scaled_inv_chi_square(1,5);
```

```
  for(i in 1:Q) {
    alpha[i,] ~ normal(mu_alpha, sqrt(sigma2_alpha));
  }

  log_beta ~ normal(mu_beta, sqrt(sigma2_beta));

  for(n in 1:N) {
    negative_y[n] ~ gumbel(-(alpha[g[n],1] + alpha[g[n],2]*t[n] +
      alpha[g[n],3]*t_squared[n] +  alpha[g[n],4]*t_cubed[n]),
        exp(log_beta[g[n]]));
  }
}
```

7.3 Running the models

```
# run_models.R
library(cmdstanr)
library(dplyr)
library(here)
library(readr)

run_gumbel_model <- function(model, warmup = 500, iters = 500, chains = 4) {
    set_cmdstan_path("~/software/cmdstan")
    sg_dat <- read_csv(here("final_project", "Data", "sg_dat_cutoff.csv"))

    Q <- length(unique(sg_dat$group))

    stan_data <- list(
        N = nrow(sg_dat),
        Q = Q,
        y = sg_dat$sg,
        g = sg_dat$group,
        t= sg_dat$t
    )

    mod <- cmdstan_model(here("final_project", "Models", paste0(model,".stan")))

    fit <- mod$sample(
        data = stan_data,
        show_messages = FALSE,
        chains = chains,
        parallel_chains = chains,
        iter_sampling = iters,
        iter_warmup = warmup,
        max_treedepth = 15,
        init = 0,
        refresh = 10
    )

    fit$save_object(file = here("final_project", "Models", paste0(model, ".rds")))
}
```

```
run_gumbel_model("baseline_model")
run_gumbel_model("intercept_only_model")
run_gumbel_model("full_model")
```

7.4 Data analysis

```
# simulate_tournament.R

inv_gumbel <- function(x, mu,beta) {
    mu + beta*log(-log(x))
}

# Full model
simulate_tournament <- function(tournament_name, year) {
    tournament_dat <- read_csv(here("final_project", "Data", "sg_dat.csv")) %>%
        filter(tournament == tournament_name, year(date) == year)

    mod <- read_rds(here("final_project", "Models", "gumbel_time_model.rds"))
    m <- mod$draws() %>% as_draws_df
    rm(mod)

    alpha_draws <- m %>% spread_draws(alpha[Q,i]) %>%
        pivot_wider(names_from = i,
                    names_prefix = "alpha_",
                    values_from = alpha) %>%
        ungroup %>%
        rename(group = Q) %>%
        select(group, matches("alpha"), .draw)
    beta_draws <- m %>% spread_draws(log_beta[n]) %>%
        mutate(beta = exp(log_beta)) %>%
        rename(group = n) %>%
        select(group,beta, .draw)

    parameter_draws <- left_join(alpha_draws, beta_draws, by=c("group", ".draw"))

    cut <- length(tournament_dat$pos[!is.na(tournament_dat$pos)])

    tournament_dat %>%
        left_join(parameter_draws, by="group") %>%
        mutate(rng = runif(n()),
               mu = alpha_1 + t*alpha_2 + t^2*alpha_3 + t^3*alpha_4,
               predicted = inv_gumbel(rng, mu, beta),
               uniform_dist = (sg-inv_gumbel(0.025,mu,beta))/
                   (inv_gumbel(0.975, mu,beta)-inv_gumbel(0.025,mu,beta))) %>%
        group_by(.draw) %>%
        arrange(desc(predicted)) %>%
        mutate(predicted_pos = 1:n()) %>%
        ungroup() %>%
        group_by(player) %>%
        summarise(win_prob = mean(predicted_pos == 1),
                  top_10_prob = mean(predicted_pos <= 10),
                  top_25_prob = mean(predicted_pos <= 25),
```

```
                  made_cut_prob = mean(predicted_pos <= cut),
                  predicted_sg = mean(mu)) %>%
        left_join(tournament_dat, by="player") %>%
        ungroup() %>%
        arrange(desc(win_prob)) %>%
        mutate(predicted_pos = c(1:cut, rep(NA, n()-cut))) %>%
        arrange(predicted_pos) %>%
        select(player, win_prob, top_10_prob, top_25_prob, made_cut_prob,
               tournament, pos, predicted_pos, sg, predicted_sg)
}

# Random model
simulate_uniform <- function(tournament_name, year) {
    tournament_dat <- read_csv(here("final_project", "Data", "sg_dat.csv")) %>%
        filter(tournament == tournament_name, year(date) == year)

    total_winners <- 0
    total_top_10 <- 0
    total_top_25 <- 0
    total_made_cut <- 0

    cut <- length(tournament_dat$pos[!is.na(tournament_dat$pos)])

    L <- 2000
    for(i in 1:L) {
        simulation <- tournament_dat %>% slice(sample(n())) %>%
            mutate(predicted_pos = 1:n())
        total_winners <- total_winners + nrow(filter(simulation, pos == 1,
                                                     predicted_pos == 1))
        total_top_10 <- total_top_10 + nrow(filter(simulation, pos <= 10,
                                                   predicted_pos <= 10))
        total_top_25 <- total_top_25 + nrow(filter(simulation, pos <= 25,
                                                   predicted_pos <= 25))
        total_made_cut <- total_made_cut + nrow(filter(simulation,
                                                       pos <= cut,
                                                       predicted_pos <= cut))
    }

    tibble(prop_winners = total_winners/L,
         prop_top_10 = total_top_10/(10*L),
         prop_top_25 = total_top_25/(25*L),
         prop_made_cut = total_made_cut/(cut*L))
}
```

```
# analyze_models.R

library(posterior)
library(tidyr)
library(readr)
library(dplyr)
library(here)
library(lubridate)
library(rstan)
```

```
library(tidybayes)
library(latex2exp)

source(here("final_project", "Scripts", "simulate_tournament_gumbel.R"))
theme_set(theme_classic(base_size = 12) + theme(legend.position = "bottom"))

# Reading in the model
gumbel_time_model <- read_rds(here("final_project", "Models",
                                   "gumbel_time_model.rds"))
gumbel_time_m <- gumbel_time_model$draws() %>% as_draws_df
rm(gumbel_time_model)

# Calculating parameter estimates
posterior_alpha_estimates <- gumbel_time_m %>% spread_draws(alpha[Q,i]) %>%
    group_by(Q,i) %>%
    summarise(lower_95 = quantile(alpha, 0.025),
              median = quantile(alpha,0.5),
              upper_95 = quantile(alpha,0.975)) %>%
    select(Q,i,median) %>%
    pivot_wider(names_from = i,
                names_prefix = "alpha_",
                values_from = median) %>%
    rename(group = Q) %>%
    ungroup

posterior_beta_estimates <- gumbel_time_m %>% spread_draws(log_beta[group]) %>%
    mutate(beta = exp(log_beta)) %>%
    group_by(group) %>%
    summarise(beta = quantile(beta, 0.5))%>%
    ungroup

hyperparameter_estimates <- gumbel_time_m %>% select(matches("mu_alpha"),
                                                     matches("sigma2_alpha"),
                                                     mu_beta, sigma2_beta) %>%
    rename(sigma_alpha_1 = "sigma2_alpha[1]",
           sigma_alpha_2 = "sigma2_alpha[2]",
           sigma_alpha_3 = "sigma2_alpha[3]",
           sigma_alpha_4 = "sigma2_alpha[4]",
           mu_alpha_1 = "mu_alpha[1]",
           mu_alpha_2 = "mu_alpha[2]",
           mu_alpha_3 = "mu_alpha[3]",
           mu_alpha_4 = "mu_alpha[4]",
           sigma_beta = sigma2_beta) %>%
    as_tibble() %>%
    mutate(sigma_alpha_1 = sqrt(sigma_alpha_1),
           sigma_alpha_2 = sqrt(sigma_alpha_2),
           sigma_alpha_3 = sqrt(sigma_alpha_3),
           sigma_alpha_4 = sqrt(sigma_alpha_4),
           sigma_beta = sqrt(sigma_beta)) %>%
    mutate(draw = 1:n()) %>%
    pivot_longer(c(-draw), names_to="parameter") %>%
    group_by(parameter) %>%
    summarise(lower_95 = quantile(value, 0.025),
```

```
              median = quantile(value,0.5),
              upper_95 = quantile(value,0.975)) %>%
    slice(c(1,6,2,7,3,8,4,9,5,10)) %>%
    mutate(parameter = paste0("$\\",parameter, "}$"),
           parameter = gsub("alpha", "{\\\\alpha", parameter),
           parameter = gsub("beta", "{\\\\beta", parameter)) %>%
    rename(Parameter = parameter,
           "Lower 95\\%" = lower_95,
           "Posterior median" = median,
           "Upper 95\\%" = upper_95)

kable(hyperparameter_estimates)

alpha_draws <- gumbel_time_m %>% spread_draws(alpha[group,i]) %>%
    pivot_wider(names_from = i,
                names_prefix = "alpha_",
                values_from = alpha) %>%
    ungroup %>%
    select(c(-.chain, -.iteration))
beta_draws <- gumbel_time_m %>% spread_draws(log_beta[group]) %>%
    mutate(beta = exp(log_beta)) %>%
    select(c(-.chain,-.iteration,-log_beta))

parameter_draws <- left_join(alpha_draws, beta_draws, by=c("group",".draw"))

sg_dat <- read_csv(here("final_project", "Data", "sg_dat.csv"))
sg_dat_cutoff <- read_csv(here("final_project", "Data", "sg_dat_cutoff.csv"))

mirror_gumbel_pdf <- function(y, mu,beta) {
    z = (mu-y)/beta
    1/beta*exp(-z-exp(-z))
}

mirror_gumbel_cdf <- function(y, mu, beta) {
    z = (mu-y)/beta
    exp(-exp(-z))
}

inverse_mirror_gumbel <- function(p, mu, beta) {
    beta*log(-log(p))+mu
}

# Note that predicted is calculated according to scaling by
# adding beta*log(log(2))
player_predictions <- sg_dat_cutoff %>%
    left_join(posterior_alpha_estimates, by=c("group")) %>%
    left_join(posterior_beta_estimates, by=c("group")) %>%
    mutate(mu = alpha_1 + alpha_2*t + alpha_3*t^2 + alpha_4*t^3,
           predicted = mu+beta*log(log(2)),
           quantile = mirror_gumbel_cdf(sg, mu, beta),
           residuals = sg-predicted)

dens <- function(fitted_player, show_fit = T) {
```

```
    index <- sg_dat %>% filter(player == fitted_player) %>% pull(group) %>%
        unique()
    player_density <- tibble(t=seq(-4,4,0.01),
                             density=mirror_gumbel_pdf(t,
                                      -log(log(2)),
                                      posterior_beta_estimates$beta[index]))
    p <- player_predictions %>% filter(player == fitted_player) %>%
        ggplot() + coord_cartesian(xlim=c(-4,4)) +
        geom_density(aes(x=residuals)) +
        xlab("Residuals") +
        ylab("Density")

    if(show_fit) p <- p+geom_line(data=player_density, aes(x=t,y=density), color="red")
    p
}

dens("Justin Thomas")
ggsave(here("final_project", "Report", "Figures",
            "posterior_predictive_thomas.pdf"), width=5,height=3, device="pdf")
dens("Jon Rahm")
ggsave(here("final_project", "Report", "Figures",
            "posterior_predictive_rahm.pdf"), width=5,height=3, device="pdf")

dens("Jon Rahm", show_fit = F)
ggsave(here("final_project", "Report", "Figures",
            "rahm_empirical_density.pdf"), width=5, height=3, device="pdf")

dens("Jordan Spieth", show_fit = F)
ggsave(here("final_project", "Report", "Figures",
            "empirical_density_spieth.pdf"), width=5, height=3, device="pdf")

dens("Dustin Johnson")
dens("Jordan Spieth")
ggsave(here("final_project", "Report", "Figures",
            "posterior_predictive_spieth.pdf"), width=5, height=3, device="pdf")

dens("Bryson DeChambeau")
ggsave(here("final_project", "Report", "Figures",
            "posterior_predictive_dechambeau.pdf"), width=5,height=3, device="pdf")
dens("Sandy Lyle")

player_predictions %>% filter(player == "Justin Thomas") %>%
    ggplot(aes(x=residuals)) + geom_density() +
    geom_line()

p <- player_predictions %>% filter(player == "Jordan Spieth") %>%
    ggplot() + geom_line(aes(x=date,y=sg)) +
    geom_line(aes(x=date,y=predicted), color='red')
ggsave(here("final_project", "Report", "Figures", "spieth_fitted_form.pdf"),
       p, width=5, height=3, device="pdf")

p <- player_predictions %>% filter(player == "Bryson DeChambeau") %>%
    ggplot() + geom_line(aes(x=date,y=sg))# +
```

```
    #geom_line(aes(x=date,y=predicted), color='red')
ggsave(here("final_project", "Report", "Figures", "dechambeau_sg.pdf"),
       p, width=5, height=3, device="pdf")

ggsave(here("final_project", "Report", "Figures", "dechambeau_fitted_form.pdf"),
       p, width=5, height=3, device="pdf")

p <- player_predictions %>% filter(player == "Jon Rahm") %>%
    ggplot() + geom_line(aes(x=date,y=sg))
ggsave(here("final_project", "Report", "Figures", "rahm_sg.pdf"),
       p, width=5, height=3, device="pdf")

ggsave(here("final_project", "Report", "Figures", "rahm_fitted_form.pdf"),
       p, width=5, height=3, device="pdf")

mirror_gumbel_pdf <- function(y, mu,beta) {
    z = (mu-y)/beta
    1/beta*exp(-z-exp(-z))
}

mirror_gumbel_cdf <- function(y, mu, beta) {
    z = (mu-y)/beta
    exp(-exp(-z))
}

# Calculating the DIC etc.
model_summary <- function(m, dat) {
    posterior_alpha_estimates <- m %>% spread_draws(alpha[Q,i]) %>%
        group_by(Q,i) %>%
        summarise(lower_95 = quantile(alpha, 0.025),
                  median = quantile(alpha,0.5),
                  upper_95 = quantile(alpha,0.975)) %>%
        select(Q,i,median) %>%
        pivot_wider(names_from = i,
                    names_prefix = "alpha_",
                    values_from = median) %>%
        rename(group = Q) %>%
        ungroup

    posterior_beta_estimates <- m %>% spread_draws(beta[n]) %>%
        group_by(n) %>%
        summarise(beta = quantile(beta,0.5)) %>%
        rename(group = n)

    D_theta_hat <- dat %>% left_join(posterior_alpha_estimates, by="group") %>%
        left_join(posterior_beta_estimates, by="group") %>%
        mutate(log_density = log(mirror_gumbel_pdf(sg, alpha_1 +
                                                       t*alpha_2 +
                                                       t^2*alpha_3 +
                                                       t^3*alpha_4, beta))) %>%
        summarise(D = -2*sum(log_density)) %>% pull(D)

    alpha_draws <- m %>% spread_draws(alpha[Q,i]) %>%
```

```
        pivot_wider(names_from = i,
                    names_prefix = "alpha_",
                    values_from = alpha) %>%
        ungroup
    beta_draws <- m %>% select(matches("beta")) %>% as.matrix

    D_theta <- c()

    for(i in 1:nrow(m)) {
        if(i %% 100 == 0) print(i)
        current_alpha_draw <- alpha_draws %>% filter(.draw == i)
        D_theta[i] <- dat %>% mutate(alpha_1 = current_alpha_draw$alpha_1[group],
                                     alpha_2 = current_alpha_draw$alpha_2[group],
                                     alpha_3 = current_alpha_draw$alpha_3[group],
                                     alpha_4 = current_alpha_draw$alpha_4[group],
                                     beta = beta_draws[i,group]) %>%
            mutate(log_density = log(mirror_gumbel_pdf(sg, alpha_1 +
                                                       t*alpha_2 +
                                                       t^2*alpha_3 +
                                                       t^3*alpha_4, beta))) %>%
            summarise(D = -2*sum(log_density)) %>% pull(D)
    }

    D_theta_avg <- mean(D_theta)
    p_D <- D_theta_avg - D_theta_hat
    DIC <- D_theta_hat + 2*p_D

    list(posterior_alpha_estimates = posterior_alpha_estimates,
         posterior_beta_estimate = posterior_beta_estimates,
         D_theta_hat = D_theta_hat,
         D_theta_avg = D_theta_avg,
         p_D = p_D,
         DIC = DIC)
}

model_summary(gumbel_time_m, sg_dat_cutoff)


# Simulating the leftover tournaments
cutoff_date <- max(sg_dat_cutoff$date)

tournaments_left <- unique(filter(sg_dat, date > cutoff_date)$tournament)

tournament_simulations <- tibble()

for(tournament in tournaments_left) {
    tournament_simulations <- tournament_simulations %>%
        bind_rows(simulate_tournament(tournament, 2021))
}

tournament_simulations %>% write_csv(here("final_project", "Data",
                                          "tournament_simulations.csv"))
```

```
tournament_simulations <- read_csv(here("final_project", "Data",
                                        "tournament_simulations.csv"))

model_performance <- tournament_simulations %>% group_by(tournament) %>%
    summarise(correct_winner = sum(pos == 1 & predicted_pos == 1, na.rm=T),
              correct_top_10 = sum(pos <= 10 & predicted_pos <= 10, na.rm=T)/10,
              prop_top_10 = 10/n(),
              correct_top_25 = sum(pos <= 25 & predicted_pos <= 25, na.rm=T)/25,
              prop_25 = 25/n(),
              correct_made_cut = sum(!is.na(pos) & !is.na(predicted_pos))/
                  max(pos, na.rm=T),
              prop_cut = max(pos,na.rm=T)/n())



# Calculating p values
anderson_darling <- function(y, mu, beta) {
    w = c(mirror_gumbel_cdf(y,mu,beta))
    sort(w)
    -length(y) - mean((2*(1:length(y))-1)*(log(w) + log(1-rev(w))))
}
L <- 2000

discrepancy <- tibble()

for(i in 1:L) {
    if(i %% 100 == 0) print(i)
    alpha_draw <- alpha_draws %>% filter(.draw == i) %>%
        select(group, matches("alpha"))
    beta_draw <- beta_draws %>% filter(.draw == i) %>% ungroup %>%
        select(group, beta)

    draw_predictions <- sg_dat_cutoff %>% left_join(alpha_draw, by="group") %>%
        left_join(beta_draw, by="group") %>%
        mutate(mu = alpha_1 + alpha_2*t + alpha_3*t^2 + alpha_4*t^4,
               y_rep = inverse_mirror_gumbel(runif(n()), mu, beta)) %>%
        group_by(player) %>%
        summarise(discrepancy = anderson_darling(y_rep, mu, beta) >=
                      anderson_darling(sg, mu, beta))

    discrepancy <- discrepancy %>% bind_rows(draw_predictions)
}

p_values <- discrepancy %>% group_by(player) %>%
    summarise(p_value = mean(discrepancy))

p_values %>% ggplot() + geom_histogram(aes(x=p_value), bins=30) +
    xlab("p-value") +
    ylab("Count") +
    scale_y_continuous(expand=c(0,0)) +
    scale_x_continuous(expand=c(0,0), limits=c(0,1)) +
    theme(plot.margin=unit(c(0.2,0.5,0.2,0.2),"cm"))
```

```
ggsave(here("final_project", "Report", "Figures", "p_values.pdf"),
       width=5, height=3, device="pdf")
```