BDA - Project

Jan Nyberg, Carl-Victor Schauman

4/12/2022

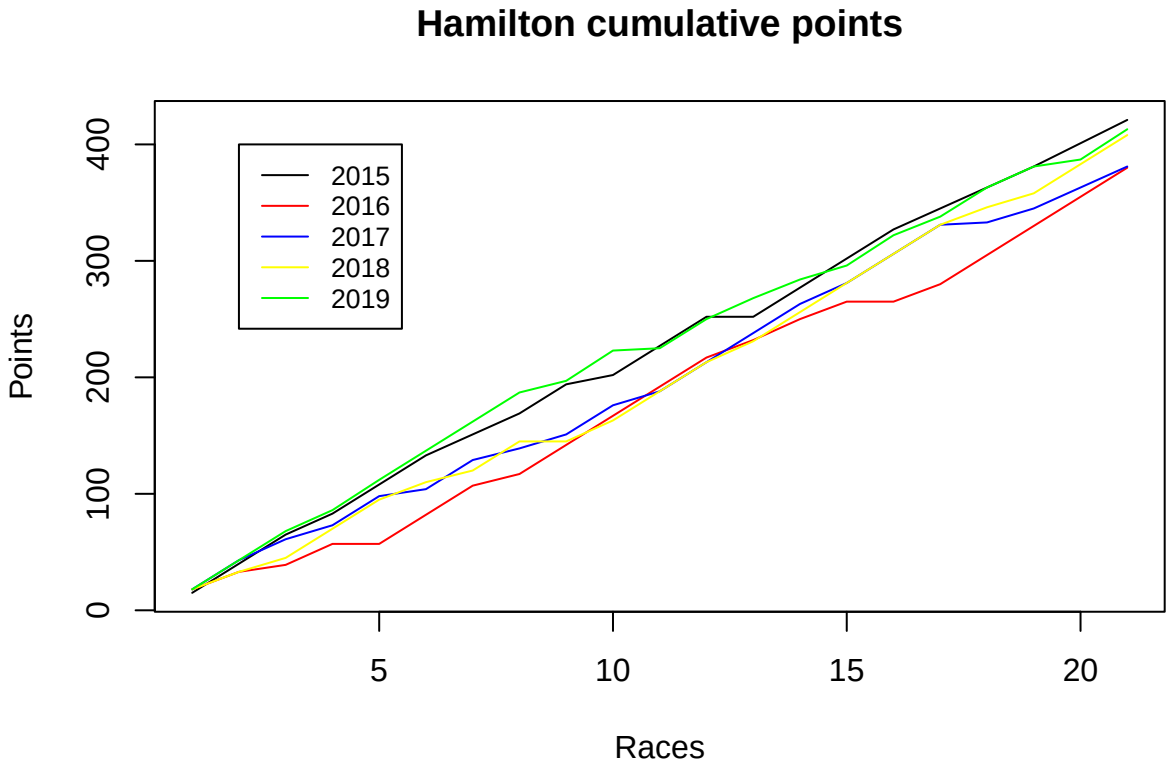
Contents

Introduction

For this project, we are trying to predict Lewis Hamilton's average score in a season based on previous years scores. This is mostly just due to our curiosity if we are able to use this to somehow predict the score. We are modeling his scores from five years, and trying to build a model using it. We want to see what kind of distribution the answer will be and how well it is able to estimate the following year. There are many factors we don't take into consideration, but we hope to see relatively good results and predictions.

Description of the data

The data we use is from Kaggle and can be found here. We took Lewis Hamilton out of the data and chose the years 2015-2019. We selected his scores from all the races from those years. One thing to note with the data is that every year doesn't have an equal amount of races. To account for this we chose to fill in the missing races with the median for the year. This makes the data a bit inaccurate, however, it shouldn't have too big an effect on the data.



Description of models

Priors used

Rstan code

Hierarchical stan model

Below is the code for the hierarchical model for Hamilton's points.

```
data {
  int<lower=0> N;
  int<lower=0> J;
```

```
  vector[J] y[N];
  real<lower=0> mu_s;
  real<lower=0> sigma_prior;
}

parameters {
  real<lower=0> muh;
  real<lower=0> sigma;
  real<lower=0> tau;
  vector[J] mu;
}

model {
  //Hyperpriors
  tau ~ inv_chi_square(sigma_prior);
  muh ~ normal(0, mu_s);

  //priors
  sigma ~ gamma(1,1);
  mu ~ normal(muh, tau);

  for (j in 1:J)
    y[,j] ~ normal(mu[j], sigma);
}

generated quantities {
  vector[J] log_lik[N];
  real ypred;
  real ypred_6;
  ypred = normal_rng(mu[5], sigma);
  ypred_6 = normal_rng(muh, sigma);
  for (j in 1:J){
    for (n in 1:N){
      log_lik[n,j] = normal_lpdf(y[n,j] | mu[j], sigma);
    }
  }
}
```

Non-hierarchical stan model

```
data {
  int<lower=0> N;
  int<lower=0> J;
  vector[N*J] y;
  real mean_mu;
  real<lower=0> mean_sigma;

}
parameters {
  real mu;
  real<lower=0> sigma;
}
model {
```

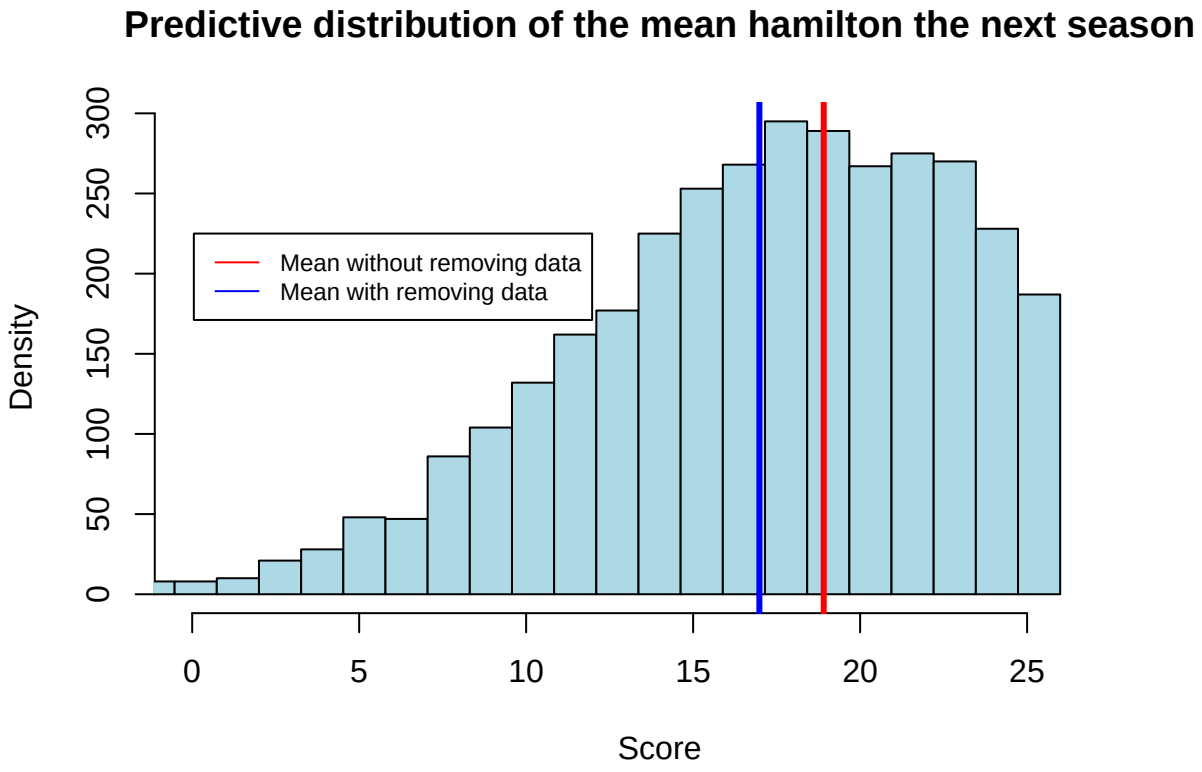
```
  // prior
  mu ~ normal(0, mean_mu);
  sigma ~ inv_chi_square(mean_sigma);
  // likelihood
  y ~ normal(mu, sigma);
}
generated quantities {
  real ypred;
  vector[N*J] log_lik;

  // Distribution based on all seasons
  ypred = normal_rng(mu, sigma);
  for (jn in 1:J*N){
    log_lik[jn] = normal_lpdf(y[jn] | mu, sigma);
  }
}
```

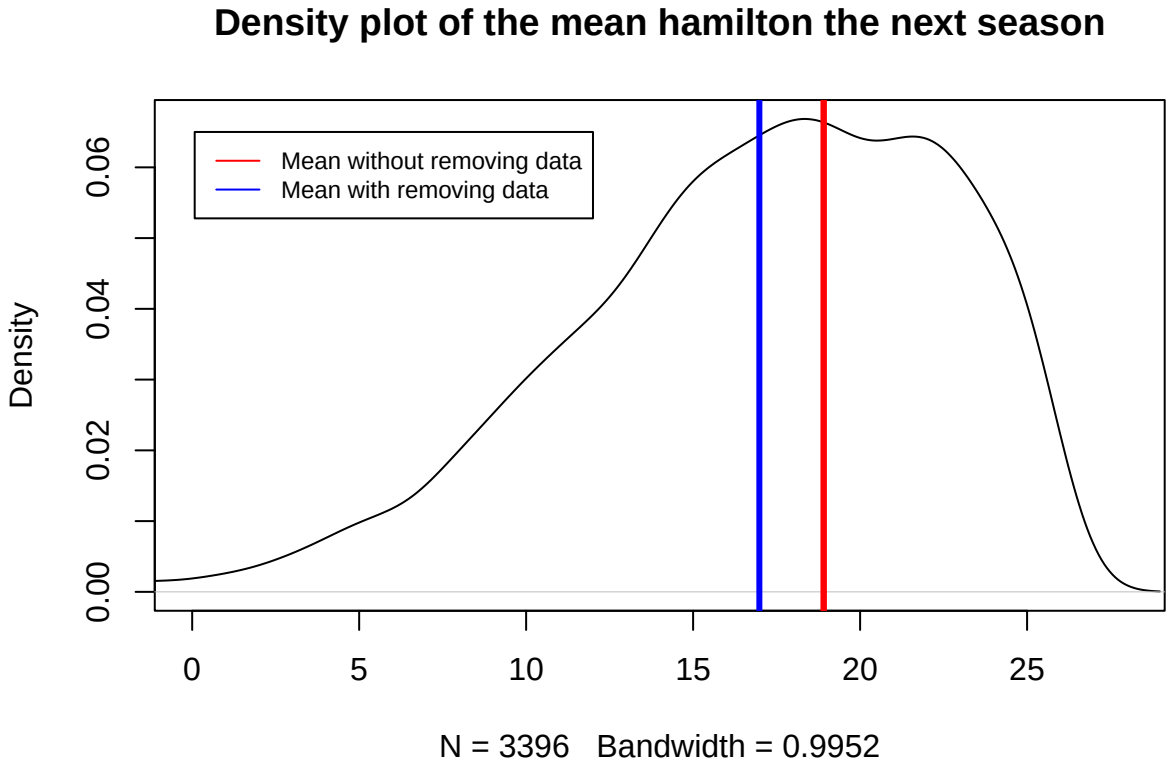
Running of stan model

Hierarchical model

Below is the hierarchical model run with the corresponding histogram with the data.

```
hier_data = list(
  y = ham_data,
  N = nrow(ham_data),
  J = ncol(ham_data),
  mu_s = 20,
  sigma_prior = 7
)

hier_fit = sampling(
  hier_ham,
  data = hier_data,
  chains = 4,
  iter = 2000,
  warmup = 1000,
  refresh = 0
)
```

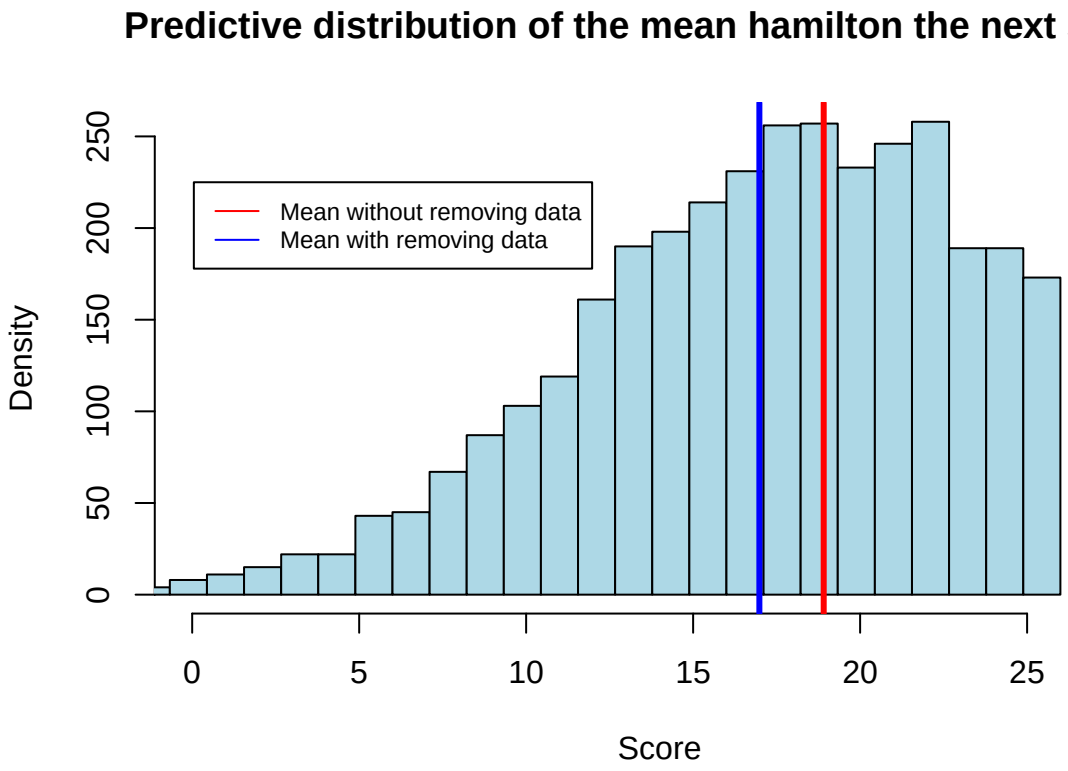


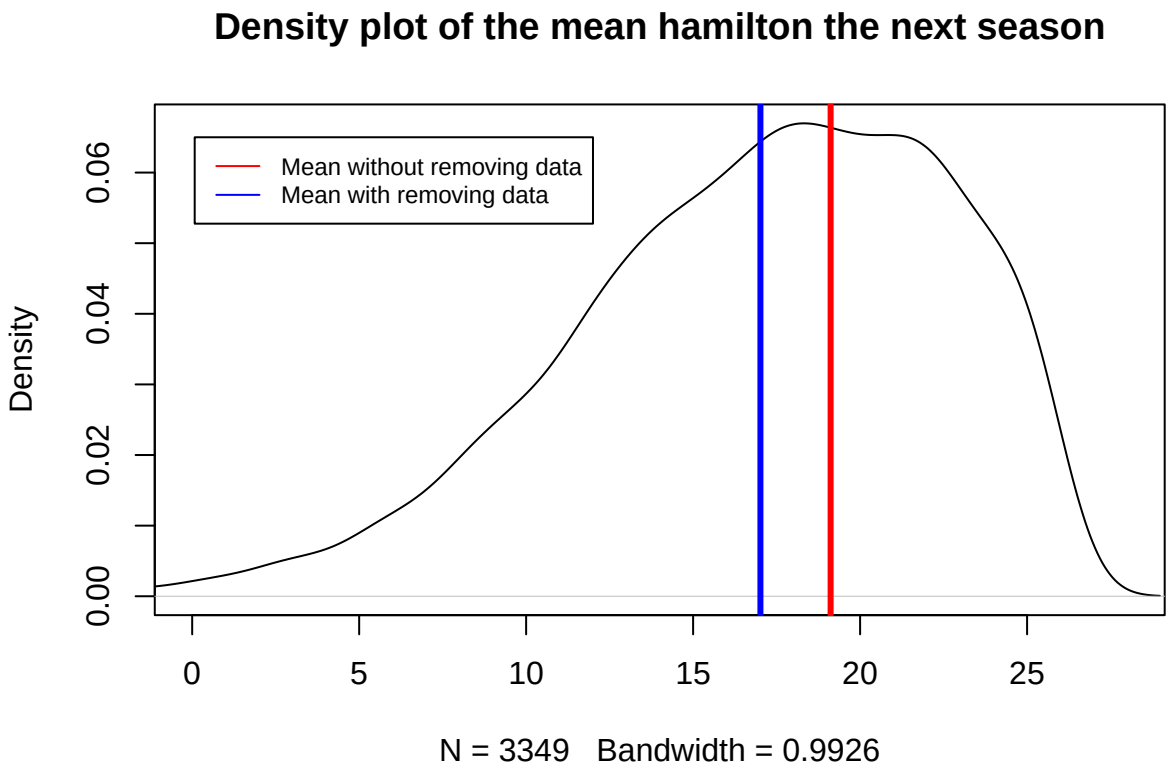


Nonhierarcial model (Pooled model)

```
pool_data = list(
  y = unlist(ham_data),
  N = nrow(ham_data),
```

```
  J = ncol(ham_data),
  mean_mu = 20,
  mean_sigma = 7
)

pool_fit = sampling(
  pool_ham,
  data = pool_data,
  chains = 4,
  iter = 2000,
  warmup = 1000,
  refresh = 0
)
```





Since the values of these normal distributions, go beyond the max points, i.e. 26, we have limited them a bit. We still plot the mean of both the limited and unlimited data. As can be seen, there isn't a lot of difference, however, over several races, this difference can be quite large. Below is also the histogram as a density plot.

Convergence diagnostics

The $\widehat{R}$ for our fits are as follows:

- Hierarchical model: 1.01
- Pooled model: 1

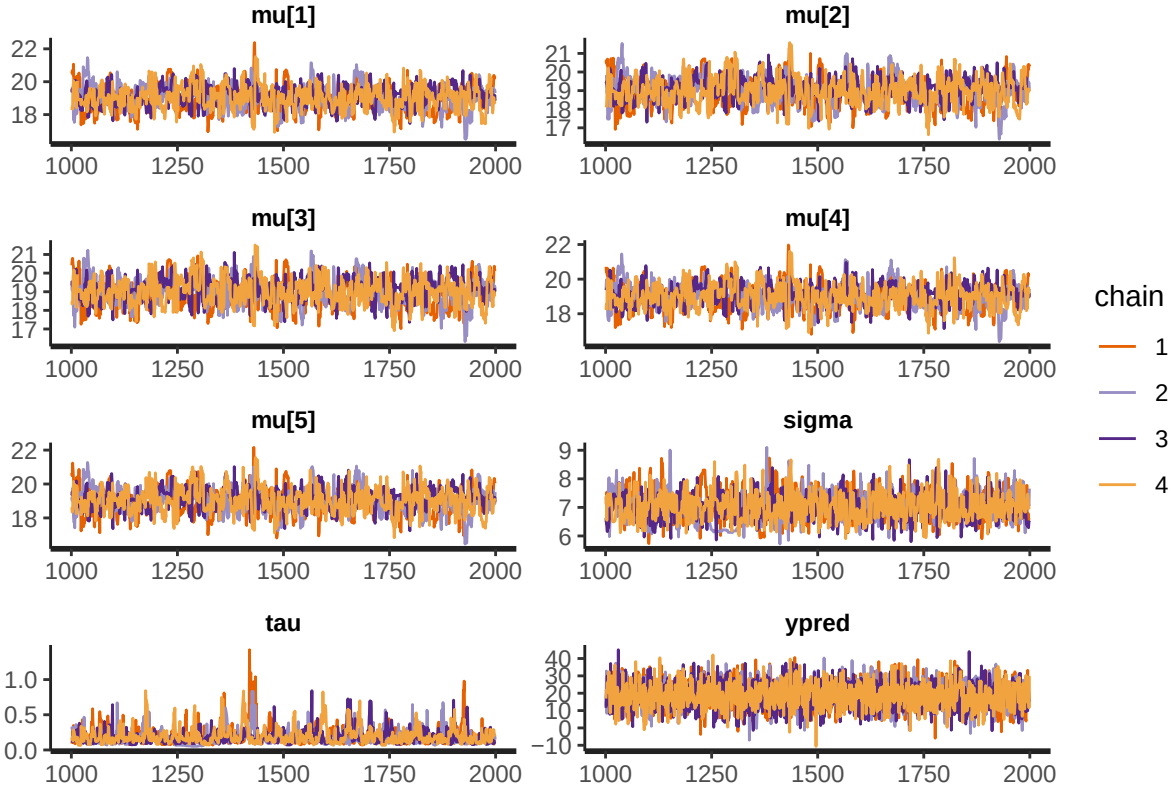
Since these $\widehat{R}$ values are under 1.05, the chains have most likely mixed well.

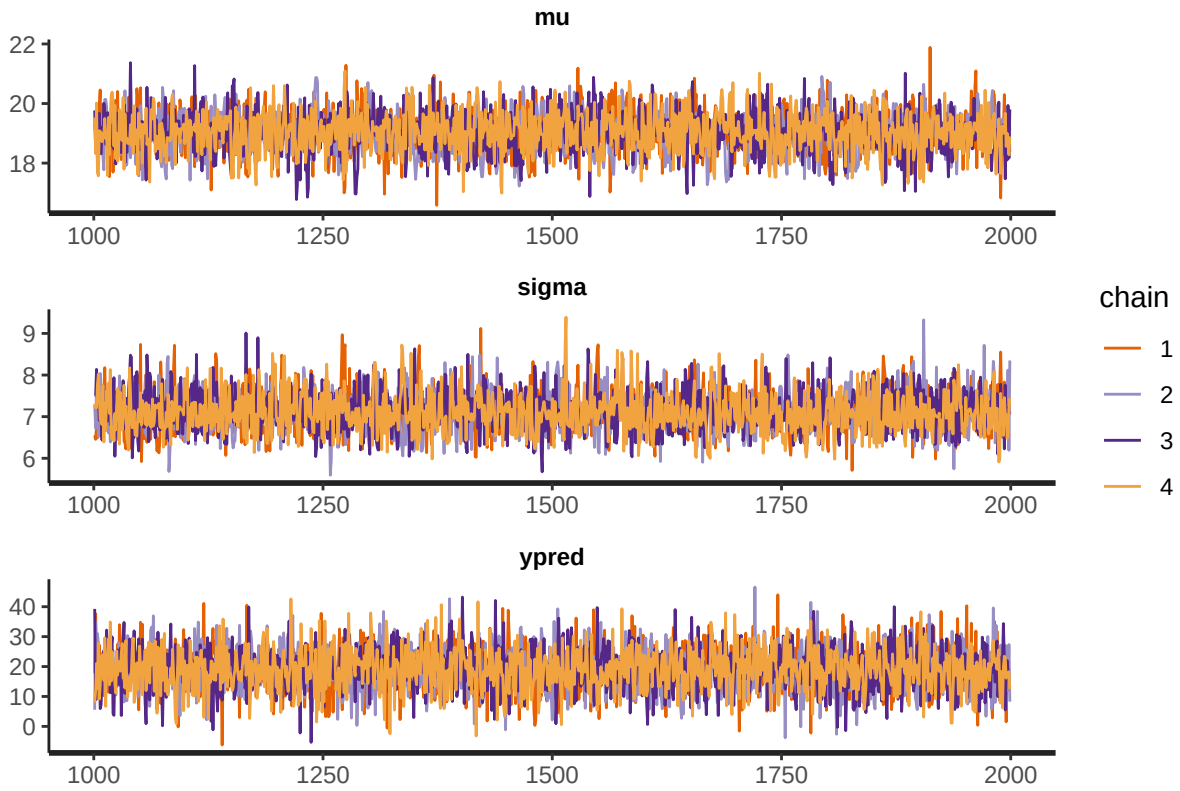
Another convergence diagnostic we can look at is the ESS value we get out of the fits.

- Bulk ESS of the hierarchical model: 521.86
- Tail ESS of the hierarchical model: 638.22
- Bulk ESS of the pooled model: 3697.33
- Tail ESS of the pooled model: 3214.33

These ESS values measure the cruse effective sample since for the bulk and tail quantities. A value over 100 is good and all of our values are over it.

Below are the traceplots for the chains. As can be seen, they seem to converge well.





Posterior predictive checks

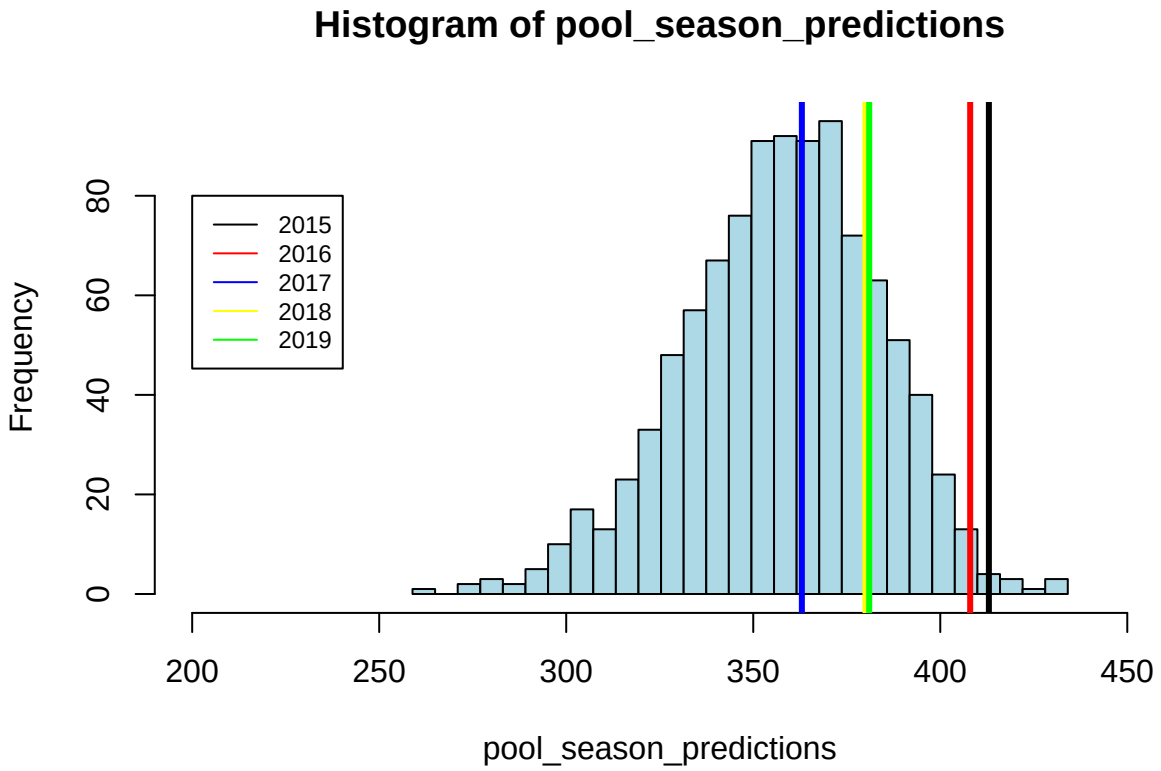
Predictive performance assessment

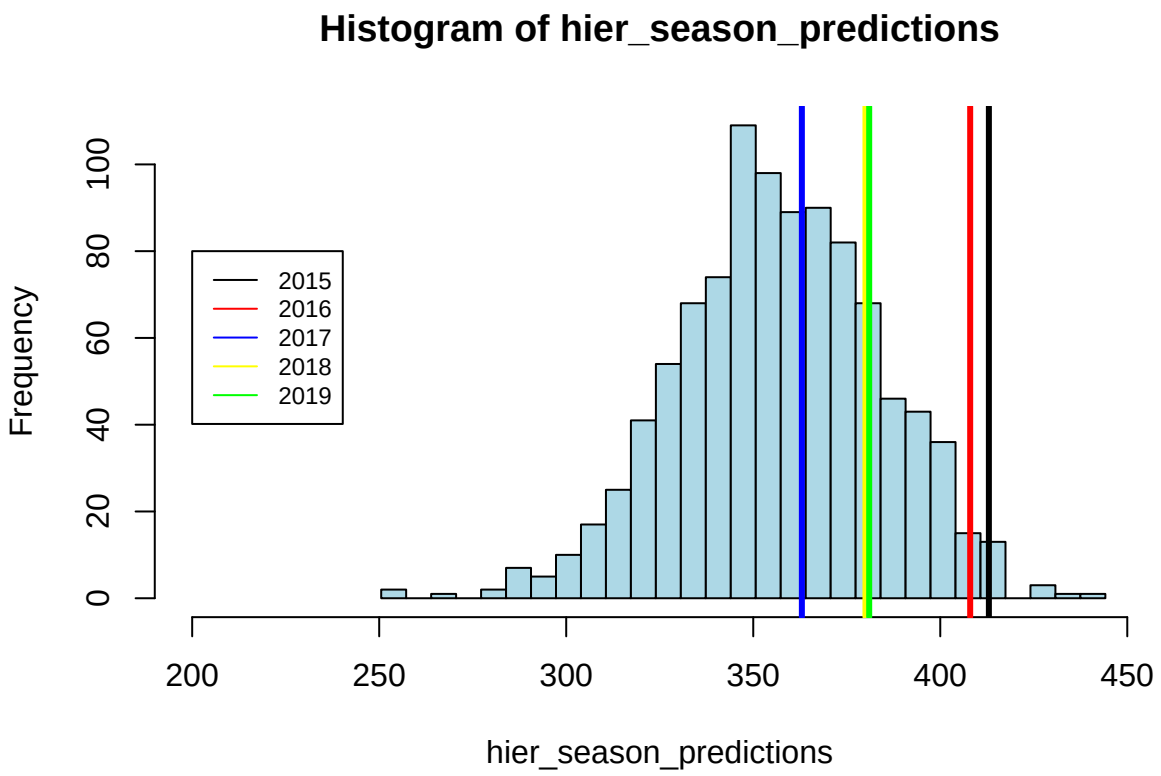
To assess the performance of the models we simulate 1000 seasons with the help of our models and compare the outcomes with the real world data.

Year	Driver	Points
2015	Hamilton	413
2016	Hamilton	408
2017	Hamilton	363
2018	Hamilton	380
2019	Hamilton	381

```
# Code for simulating 1000 seasons

pool_season_predictions = c()
hier_season_predictions = c()

for(i in 1:1000) {
  dens = ham_pool_extracted_density
  N <- 21
  newx <- sample(x = dens$x, N,
       prob = dens$y, replace=TRUE)
    + rnorm(N, 0, dens$bw)
  pool_season_predictions <- append(pool_season_predictions, sum(newx))
```

```
}

for(i in 1:1000) {
  dens = ham_extracted_density
  N <- 21
  newx <- sample(x = dens$x, N,
       prob = dens$y, replace=TRUE)
    + rnorm(N, 0, dens$bw)
  hier_season_predictions <- append(hier_season_predictions, sum(newx))
}
```



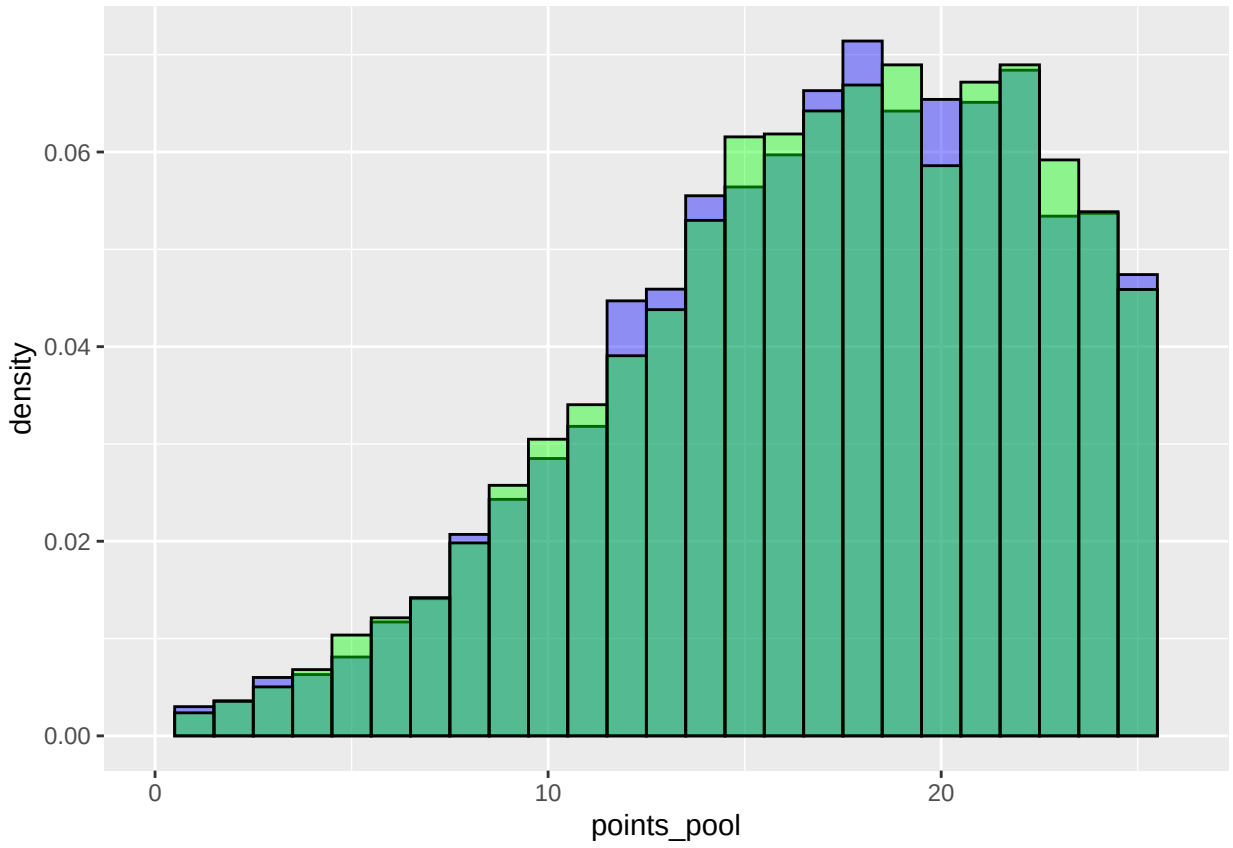


When comparing the simulated data with the actual real world data we can see that it is in the right ball park, but slightly of.

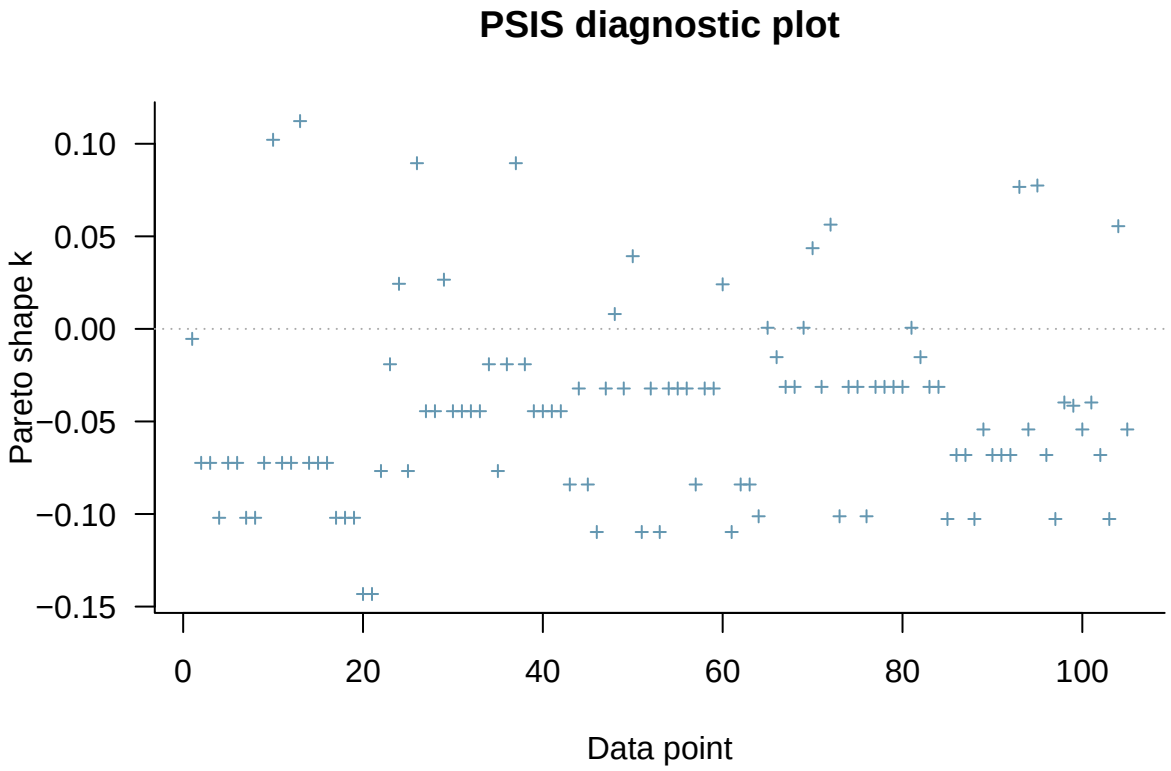
Sensitivity analysis

Model comparison

Our models compared to each other in one graph looks like this. As we can see, they are relatively similar. To check performance and comparing them, we will also use PSIS-LOO values.



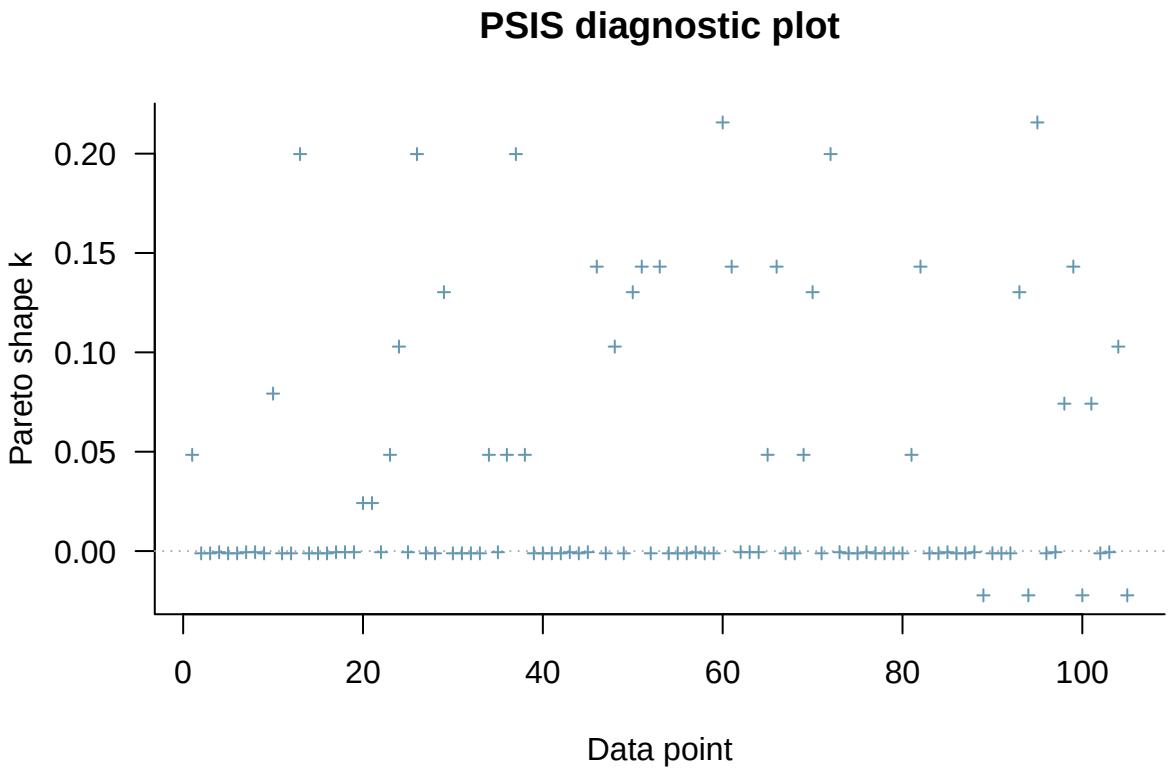
Hierarchical analysis



	Estimate	SE
elpd_loo	-357.767184	8.4583729
p_loo	2.224619	0.5271362
looic	715.534367	16.9167458

As can be seen, all of our k values are under 0.5, which means they are all good, and our PSIS-LOO estimate can be considered reliable.

Nonhierarchical analysis



	Estimate	SE
elpd_loo	-357.773141	8.2947834
p_loo	2.269913	0.5461605
looic	715.546282	16.5895668

Like in the previous one, all of our k values are under 0.5, which means they are all good, and our PSIS-LOO estimate can be considered reliable.

```
knitr::kable(loo_compare(pool_loo, hier_loo))
```

	elpd_diff	se_diff	elpd_loo	se_elpd_loo	p_loo	se_p_loo	looic	se_looic
model2	0.0000000	0.0000000	-357.7672	8.458373	2.224619	0.5271362	715.5344	16.91675
model1	-0.0059572	0.1675774	-357.7731	8.294783	2.269913	0.5461605	715.5463	16.58957

Here we can once again see that the values are very close to each other, and that both models perform quite

similarly to each other. This wasn't a surprise considering the first graph in this section, which shows that both are very similar.

Discussion of issues and potential improvements

Our problem to begin with is almost impossible to model correctly since F1 is dependent on so many different parameters. Who you are driving for, the current regulations and the engineering of the car are all important factors that the drivers can not simply affect in any way. The choice to model the data based on a normal distribution might also have been a bad idea since our rstan model is also predicting out values that are negative and larger than 26 (max points from one race). Lastly the data used had minor inconsistencies with number of races each season. 2015 had 2 less races and 2017 1 less. To make it easier for us we filled in the missing values with the mean score of the season.

In our project we decided to look at the results of an individual driver, but it might have also been a good idea to look at results of teams instead. However by looking at team results we just get different inconsistencies such as who is driving for the team and what their current form is. An other idea for improvements is to look at more parameters than just the points scored e.g. positions in qualifying and point difference between teammates.

Conclusion what was learned from the data analysis

Self-reflection