BDA - Project

Jan Nyberg, Carl-Victor Schauman

4/12/2022

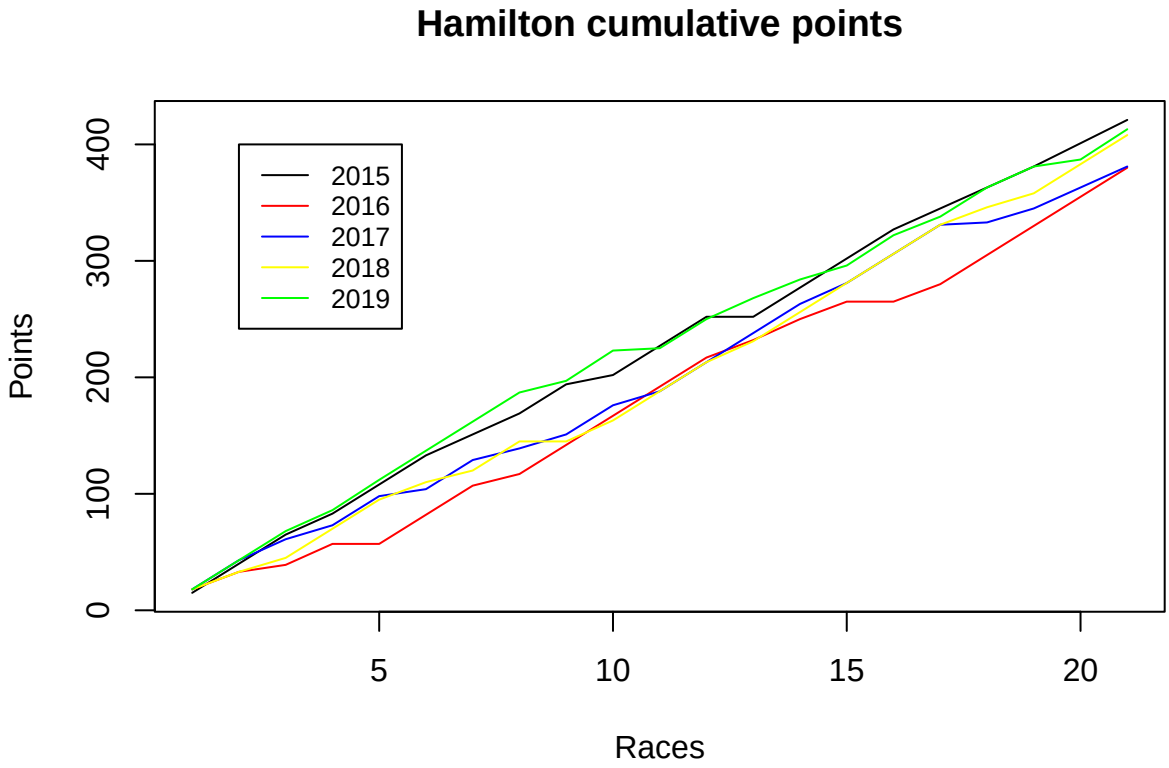
Contents

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> mu;
  real<lower=0> sigma;
  real<lower=0> tau;
  vector[J] mus;
}

model {
  mu ~ normal(0, mu_s);
  tau ~ inv_chi_square(sigma_prior);
  sigma ~ gamma(1,1);
  mus ~ normal(mu, tau);
  for (j in 1:J)
    y[,j] ~ normal(mus[j], sigma);
}

generated quantities {
  vector[J] log_lik[N];
  real ypred;
  real ypred_6;
  ypred = normal_rng(mus[5], sigma);
  ypred_6 = normal_rng(mu, sigma);
  for (j in 1:J){
    for (n in 1:N){
      log_lik[n,j] = normal_lpdf(y[n,j] | mus[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(mean_mu, mean_sigma);
  sigma ~ inv_chi_square(mean_sigma);
  // likelihood
```

```
  y ~ normal(mu, sigma);
}
generated quantities {
  real ypred;

  // Distribution based on all seasons
  ypred = normal_rng(mu, sigma);
}
```

Running of stan model

Hierarchical model

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

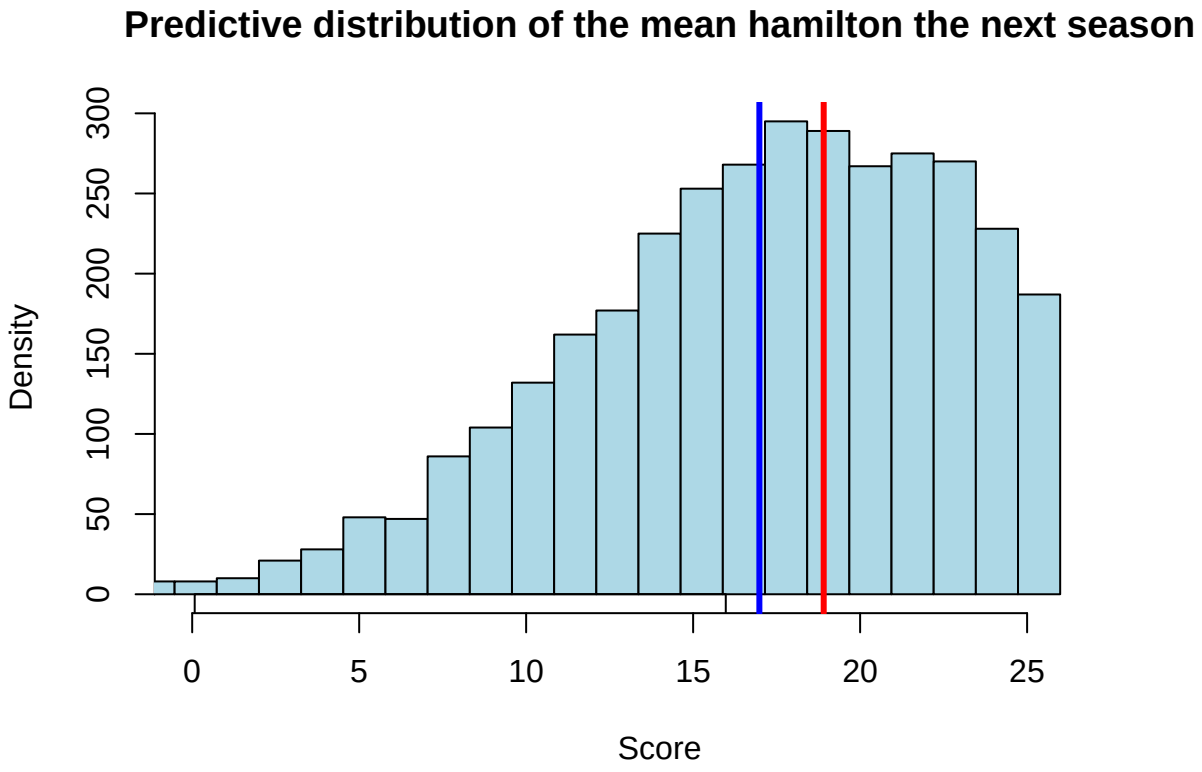
```
hier_data = list(
  y = ham_data,
  N = nrow(ham_data),
  J = ncol(ham_data),
  U = 26,
  mu_s = 20,
  sigma_prior = 7
)

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

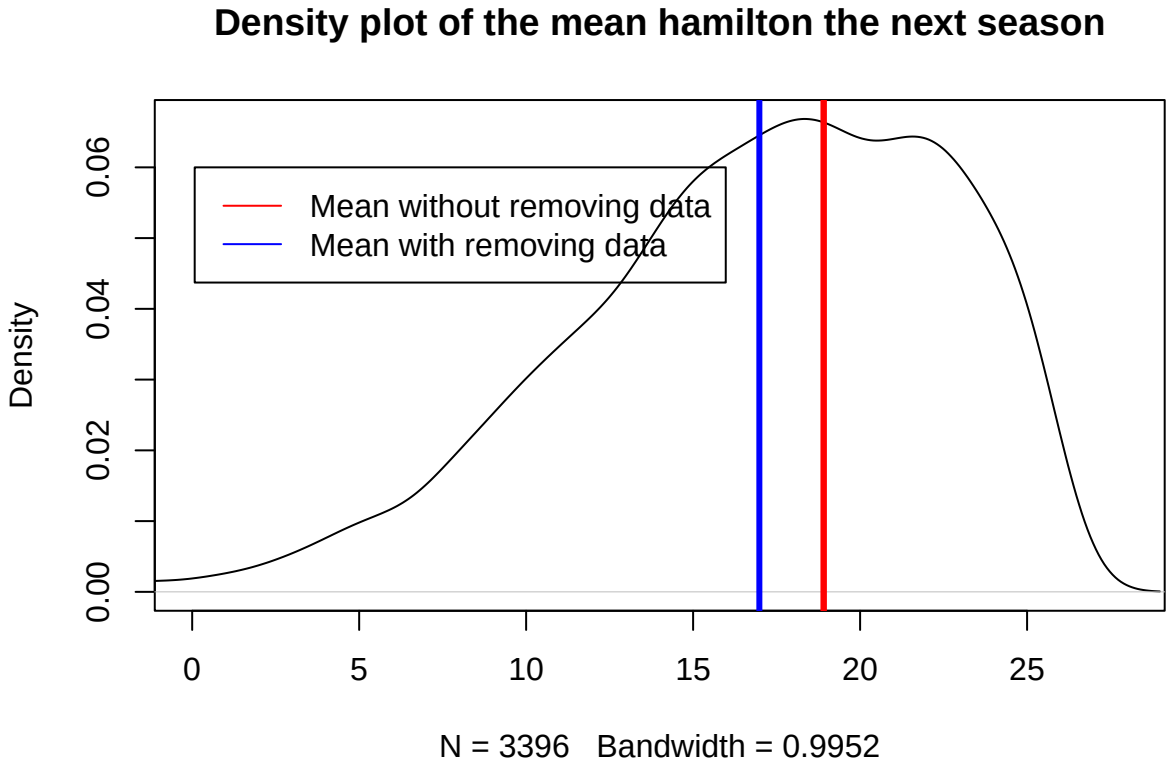
```
## Warning: There were 99 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#bulk-ess
```

```
## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quant
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#tail-ess
```

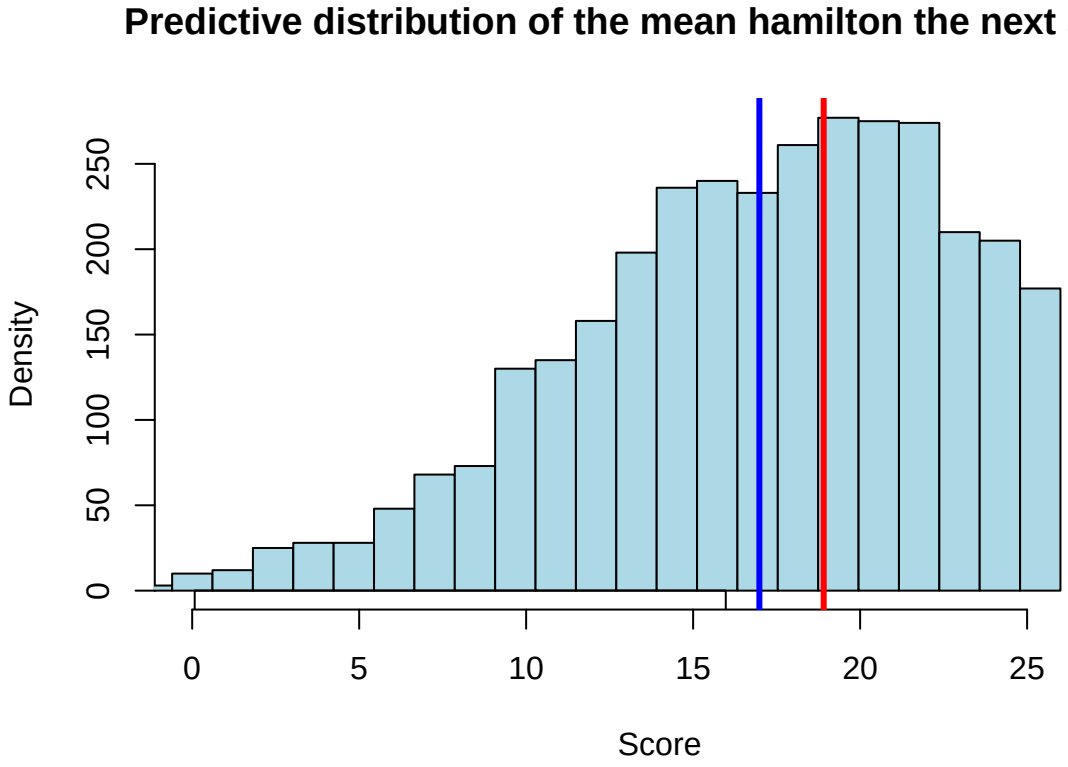


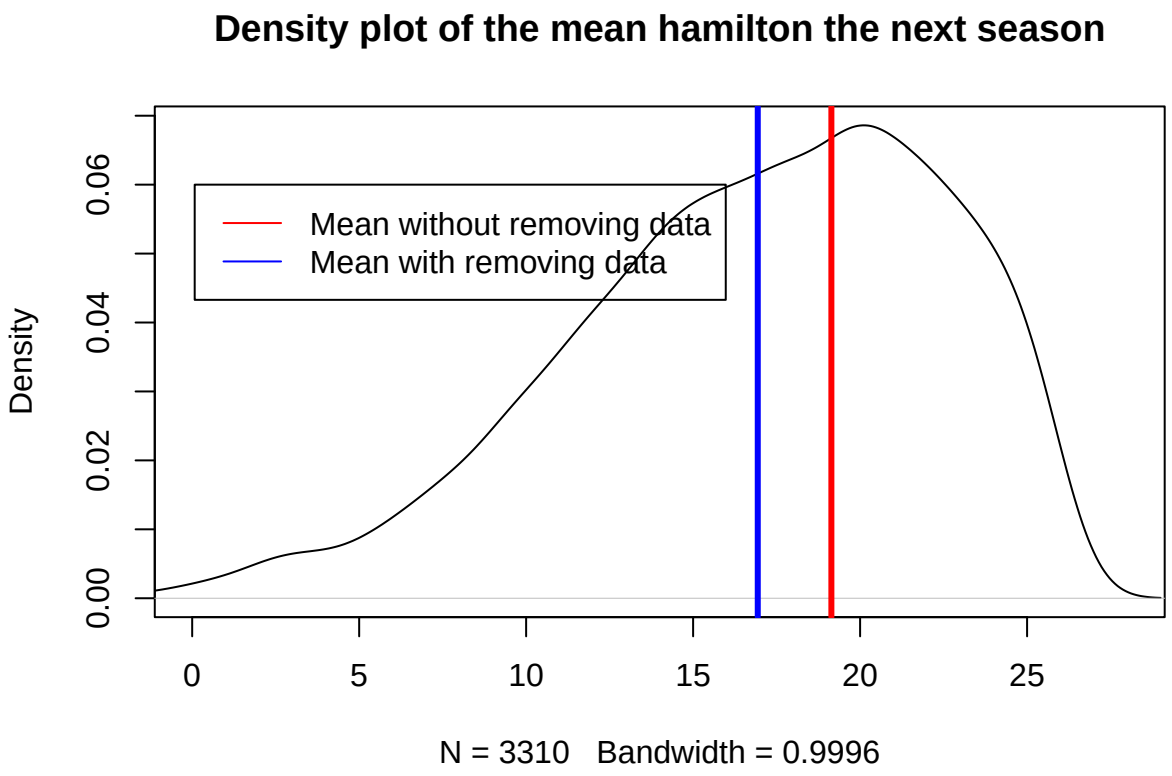


Nonhierarcial model (Pooled model)

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

```
  J = ncol(ham_data),
  mean_mu = 18,
  mean_sigma = 6
)

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

```
## Inference for the input samples (4 chains: each with iter = 2000; warmup = 0):
##
##                   Q5    Q50    Q95   Mean  SD  Rhat Bulk_ESS Tail_ESS
## mu              17.9   19.0   20.2   19.0 0.7  1.01      460      651
## sigma            6.3    7.0    7.9    7.0 0.5  1.01      588      187
## tau              0.1    0.2    0.4    0.2 0.1  1.01      370      165
## mus[1]          17.9   19.0   20.2   19.0 0.7  1.01      484      674
## mus[2]          17.8   19.0   20.2   19.0 0.7  1.01      477      630
## mus[3]          17.9   19.0   20.2   19.0 0.7  1.01      499      750
## mus[4]          17.9   19.0   20.2   19.0 0.7  1.01      474      632
## mus[5]          17.9   19.0   20.2   19.0 0.7  1.01      488      636
## log_lik[1,1]    -3.2   -3.0   -2.9   -3.0 0.1  1.01      636      778
## log_lik[2,1]    -3.4   -3.2   -3.1   -3.2 0.1  1.01      494      791
## log_lik[3,1]    -3.4   -3.2   -3.1   -3.2 0.1  1.01      494      791
## log_lik[4,1]    -3.0   -2.9   -2.8   -2.9 0.1  1.01      605      181
## log_lik[5,1]    -3.4   -3.2   -3.1   -3.2 0.1  1.01      494      791
## log_lik[6,1]    -3.4   -3.2   -3.1   -3.2 0.1  1.01      494      791
## log_lik[7,1]    -3.0   -2.9   -2.8   -2.9 0.1  1.01      605      181
## log_lik[8,1]    -3.0   -2.9   -2.8   -2.9 0.1  1.01      605      181
## log_lik[9,1]    -3.4   -3.2   -3.1   -3.2 0.1  1.01      494      791
## log_lik[10,1]   -4.4   -4.1   -3.8   -4.1 0.2  1.02      445      875
## log_lik[11,1]   -3.4   -3.2   -3.1   -3.2 0.1  1.01      494      791
## log_lik[12,1]   -3.4   -3.2   -3.1   -3.2 0.1  1.01      494      791
```

```
## log_lik[13,1]   -7.5   -6.6   -5.8   -6.6 0.5  1.01      427      237
## log_lik[14,1]   -3.4   -3.2   -3.1   -3.2 0.1  1.01      494      791
## log_lik[15,1]   -3.4   -3.2   -3.1   -3.2 0.1  1.01      494      791
## log_lik[16,1]   -3.4   -3.2   -3.1   -3.2 0.1  1.01      494      791
## log_lik[17,1]   -3.0   -2.9   -2.8   -2.9 0.1  1.01      605      181
## log_lik[18,1]   -3.0   -2.9   -2.8   -2.9 0.1  1.01      605      181
## log_lik[19,1]   -3.0   -2.9   -2.8   -2.9 0.1  1.01      605      181
## log_lik[20,1]   -3.0   -2.9   -2.8   -2.9 0.1  1.01      454      181
## log_lik[21,1]   -3.0   -2.9   -2.8   -2.9 0.1  1.01      454      181
## log_lik[1,2]    -3.0   -2.9   -2.8   -2.9 0.1  1.01      625      208
## log_lik[2,2]    -3.2   -3.0   -2.9   -3.0 0.1  1.01      640      717
## log_lik[3,2]    -5.0   -4.6   -4.2   -4.6 0.2  1.02      403     1006
## log_lik[4,2]    -3.0   -2.9   -2.8   -2.9 0.1  1.01      625      208
## log_lik[5,2]    -7.5   -6.5   -5.8   -6.6 0.5  1.01      407      191
## log_lik[6,2]    -3.4   -3.2   -3.1   -3.2 0.1  1.01      483      819
## log_lik[7,2]    -3.4   -3.2   -3.1   -3.2 0.1  1.01      483      819
## log_lik[8,2]    -3.9   -3.7   -3.5   -3.7 0.1  1.02      455      856
## log_lik[9,2]    -3.4   -3.2   -3.1   -3.2 0.1  1.01      483      819
## log_lik[10,2]   -3.4   -3.2   -3.1   -3.2 0.1  1.01      483      819
## log_lik[11,2]   -3.4   -3.2   -3.1   -3.2 0.1  1.01      483      819
## log_lik[12,2]   -3.4   -3.2   -3.1   -3.2 0.1  1.01      483      819
## log_lik[13,2]   -3.2   -3.0   -2.9   -3.0 0.1  1.01      640      717
## log_lik[14,2]   -3.0   -2.9   -2.8   -2.9 0.1  1.01      625      208
## log_lik[15,2]   -3.2   -3.0   -2.9   -3.0 0.1  1.01      640      717
## log_lik[16,2]   -7.5   -6.5   -5.8   -6.6 0.5  1.01      407      191
## log_lik[17,2]   -3.2   -3.0   -2.9   -3.0 0.1  1.01      640      717
## log_lik[18,2]   -3.4   -3.2   -3.1   -3.2 0.1  1.01      483      819
## log_lik[19,2]   -3.4   -3.2   -3.1   -3.2 0.1  1.01      483      819
## log_lik[20,2]   -3.4   -3.2   -3.1   -3.2 0.1  1.01      483      819
## log_lik[21,2]   -3.4   -3.2   -3.1   -3.2 0.1  1.01      483      819
## log_lik[1,3]    -3.0   -2.9   -2.8   -2.9 0.1  1.01      652      187
## log_lik[2,3]    -3.4   -3.2   -3.1   -3.2 0.1  1.01      503      793
## log_lik[3,3]    -3.0   -2.9   -2.8   -2.9 0.1  1.01      652      187
## log_lik[4,3]    -3.6   -3.4   -3.2   -3.4 0.1  1.01      512      825
## log_lik[5,3]    -3.4   -3.2   -3.1   -3.2 0.1  1.01      503      793
## log_lik[6,3]    -5.0   -4.6   -4.2   -4.6 0.2  1.02      440     1032
## log_lik[7,3]    -3.4   -3.2   -3.1   -3.2 0.1  1.01      503      793
## log_lik[8,3]    -3.9   -3.7   -3.5   -3.7 0.1  1.02      467      835
## log_lik[9,3]    -3.6   -3.4   -3.2   -3.4 0.1  1.01      512      825
## log_lik[10,3]   -3.4   -3.2   -3.1   -3.2 0.1  1.01      503      793
## log_lik[11,3]   -3.6   -3.4   -3.2   -3.4 0.1  1.01      512      825
## log_lik[12,3]   -3.4   -3.2   -3.1   -3.2 0.1  1.01      503      793
## log_lik[13,3]   -3.4   -3.2   -3.1   -3.2 0.1  1.01      503      793
## log_lik[14,3]   -3.4   -3.2   -3.1   -3.2 0.1  1.01      503      793
## log_lik[15,3]   -3.0   -2.9   -2.8   -2.9 0.1  1.01      652      187
## log_lik[16,3]   -3.4   -3.2   -3.1   -3.2 0.1  1.01      503      793
## log_lik[17,3]   -3.4   -3.2   -3.1   -3.2 0.1  1.01      503      793
## log_lik[18,3]   -6.5   -5.8   -5.2   -5.8 0.4  1.01      403      318
## log_lik[19,3]   -3.6   -3.4   -3.2   -3.4 0.1  1.01      512      825
## log_lik[20,3]   -3.0   -2.9   -2.8   -2.9 0.1  1.01      652      187
## log_lik[21,3]   -3.0   -2.9   -2.8   -2.9 0.1  1.01      652      187
## log_lik[1,4]    -3.0   -2.9   -2.8   -2.9 0.1  1.01      627      186
## log_lik[2,4]    -3.2   -3.0   -2.9   -3.0 0.1  1.01      639      863
## log_lik[3,4]    -3.6   -3.4   -3.2   -3.4 0.1  1.01      485      695
```

```
## log_lik[4,4]    -3.4   -3.2   -3.1   -3.2 0.1  1.02      479      822
## log_lik[5,4]    -3.4   -3.2   -3.1   -3.2 0.1  1.02      479      822
## log_lik[6,4]    -3.2   -3.0   -2.9   -3.0 0.1  1.01      639      863
## log_lik[7,4]    -3.9   -3.7   -3.5   -3.7 0.1  1.02      451      879
## log_lik[8,4]    -3.4   -3.2   -3.1   -3.2 0.1  1.02      479      822
## log_lik[9,4]    -7.5   -6.6   -5.8   -6.6 0.5  1.01      412      209
## log_lik[10,4]   -3.0   -2.9   -2.8   -2.9 0.1  1.01      627      186
## log_lik[11,4]   -3.4   -3.2   -3.1   -3.2 0.1  1.02      479      822
## log_lik[12,4]   -3.4   -3.2   -3.1   -3.2 0.1  1.02      479      822
## log_lik[13,4]   -3.0   -2.9   -2.8   -2.9 0.1  1.01      627      186
## log_lik[14,4]   -3.4   -3.2   -3.1   -3.2 0.1  1.02      479      822
## log_lik[15,4]   -3.4   -3.2   -3.1   -3.2 0.1  1.02      479      822
## log_lik[16,4]   -3.4   -3.2   -3.1   -3.2 0.1  1.02      479      822
## log_lik[17,4]   -3.4   -3.2   -3.1   -3.2 0.1  1.02      479      822
## log_lik[18,4]   -3.2   -3.0   -2.9   -3.0 0.1  1.01      639      863
## log_lik[19,4]   -3.6   -3.4   -3.2   -3.4 0.1  1.01      485      695
## log_lik[20,4]   -3.4   -3.2   -3.1   -3.2 0.1  1.02      479      822
## log_lik[21,4]   -3.4   -3.2   -3.1   -3.2 0.1  1.02      479      822
## log_lik[1,5]    -3.0   -2.9   -2.8   -2.9 0.1  1.01      607      185
## log_lik[2,5]    -3.4   -3.2   -3.1   -3.2 0.1  1.02      495      832
## log_lik[3,5]    -3.4   -3.2   -3.1   -3.2 0.1  1.02      495      832
## log_lik[4,5]    -3.0   -2.9   -2.8   -2.9 0.1  1.01      607      185
## log_lik[5,5]    -3.5   -3.4   -3.2   -3.4 0.1  1.01      497      789
## log_lik[6,5]    -3.4   -3.2   -3.1   -3.2 0.1  1.02      495      832
## log_lik[7,5]    -3.4   -3.2   -3.1   -3.2 0.1  1.02      495      832
## log_lik[8,5]    -3.4   -3.2   -3.1   -3.2 0.1  1.02      495      832
## log_lik[9,5]    -3.9   -3.7   -3.5   -3.7 0.1  1.02      473      779
## log_lik[10,5]   -3.5   -3.4   -3.2   -3.4 0.1  1.01      497      789
## log_lik[11,5]   -6.5   -5.8   -5.2   -5.8 0.4  1.01      434      323
## log_lik[12,5]   -3.4   -3.2   -3.1   -3.2 0.1  1.02      495      832
## log_lik[13,5]   -3.0   -2.9   -2.8   -2.9 0.1  1.01      607      185
## log_lik[14,5]   -3.1   -3.0   -2.9   -3.0 0.1  1.01      714      978
## log_lik[15,5]   -3.6   -3.4   -3.2   -3.4 0.1  1.01      499      664
## log_lik[16,5]   -3.5   -3.4   -3.2   -3.4 0.1  1.01      497      789
## log_lik[17,5]   -3.1   -3.0   -2.9   -3.0 0.1  1.01      714      978
## log_lik[18,5]   -3.4   -3.2   -3.1   -3.2 0.1  1.02      495      832
## log_lik[19,5]   -3.0   -2.9   -2.8   -2.9 0.1  1.01      607      185
## log_lik[20,5]   -5.0   -4.6   -4.2   -4.6 0.2  1.02      447      954
## log_lik[21,5]   -3.5   -3.4   -3.2   -3.4 0.1  1.01      497      789
## ypred            7.5   18.9   30.5   18.9 7.0  1.00     3921     3643
## ypred_6          7.2   19.0   30.9   19.0 7.1  1.00     3642     3752
## lp__          -259.2 -252.6 -248.7 -253.1 3.3  1.01      554      619
##
## For each parameter, Bulk_ESS and Tail_ESS are crude measures of
## effective sample size for bulk and tail quantities respectively (an ESS > 100
## per chain is considered good), and Rhat is the potential scale reduction
## factor on rank normalized split chains (at convergence, Rhat <= 1.05).

## Inference for the input samples (4 chains: each with iter = 2000; warmup = 0):
##
##           Q5    Q50    Q95   Mean  SD  Rhat Bulk_ESS Tail_ESS
## mu      17.9   19.1   20.2   19.1 0.7     1     3558     2678
## sigma    6.4    7.1    8.0    7.1 0.5     1     3688     2657
## ypred    7.4   19.3   30.8   19.1 7.2     1     3937     3772
```

```
## lp__  -268.0 -265.8 -265.1 -266.1 1.0     1     1787     2472
##
## For each parameter, Bulk_ESS and Tail_ESS are crude measures of
## effective sample size for bulk and tail quantities respectively (an ESS > 100
## per chain is considered good), and Rhat is the potential scale reduction
## factor on rank normalized split chains (at convergence, Rhat <= 1.05).
```

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: 521.8584071
- Tail ESS of the hierarchical model: 638.2212389
- Bulk ESS of the pooled model: 3727.6666667
- Tail ESS of the pooled model: 3035.6666667

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

Posterior predictive checks

Predictive performance assessment

Sensitivity analysis

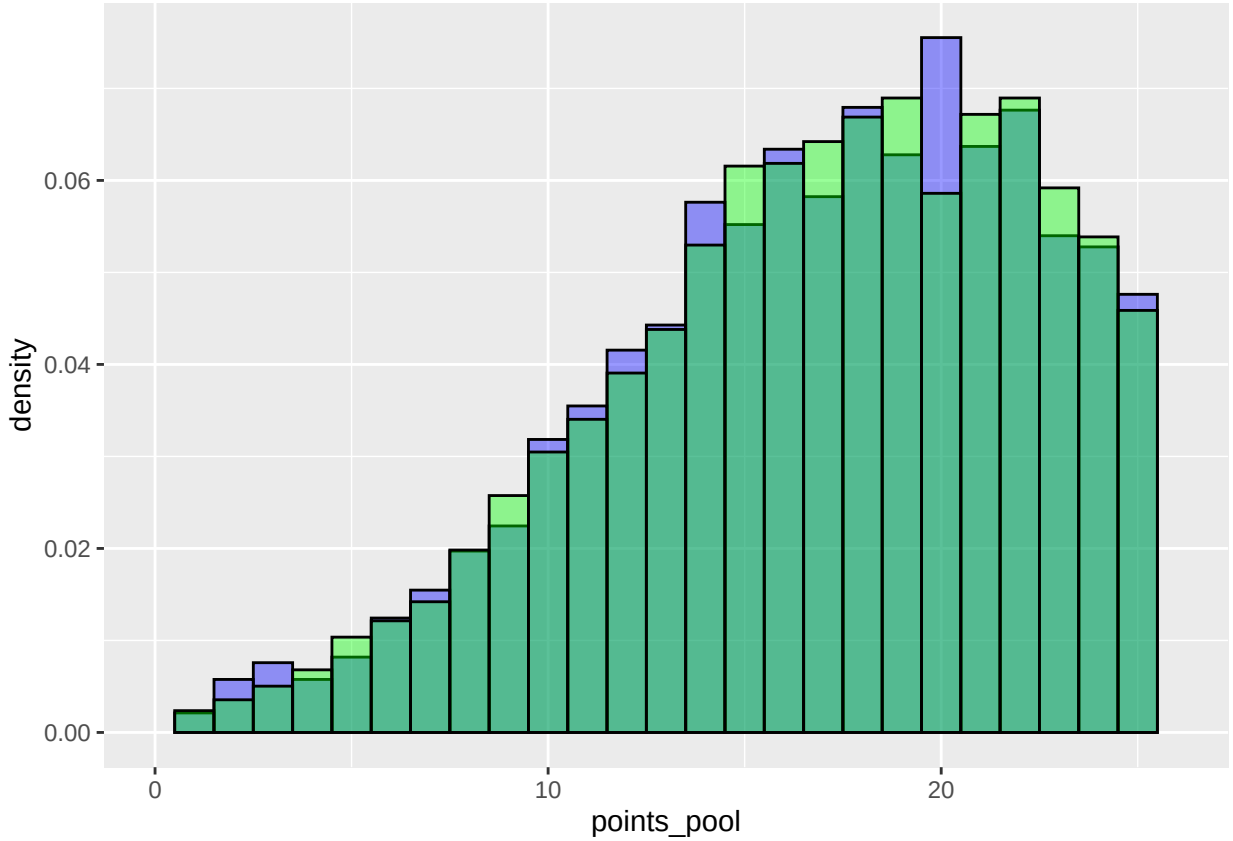
Model comparison

```
df <- data.frame(ham_pool_extracted, ham_extracted)
colnames(df) <- c("points_pool", "points_hier")
ggplot(
  data = df,
  mapping = aes(x=points_pool)
) + geom_histogram(
  aes(x=points_pool, y=..density..),
  binwidth = 1,
  colour="black",
  fill="blue",
  position = "identity",
  alpha = 0.4
) + geom_histogram(
  aes(x=points_hier, y=..density..),
  binwidth = 1,
  colour="black",
  fill="green",
  position = "identity",
  alpha = 0.4
) + xlim(
  0,
  26
)
```

```
## Warning: Removed 703 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 621 rows containing non-finite values (stat_bin).
```

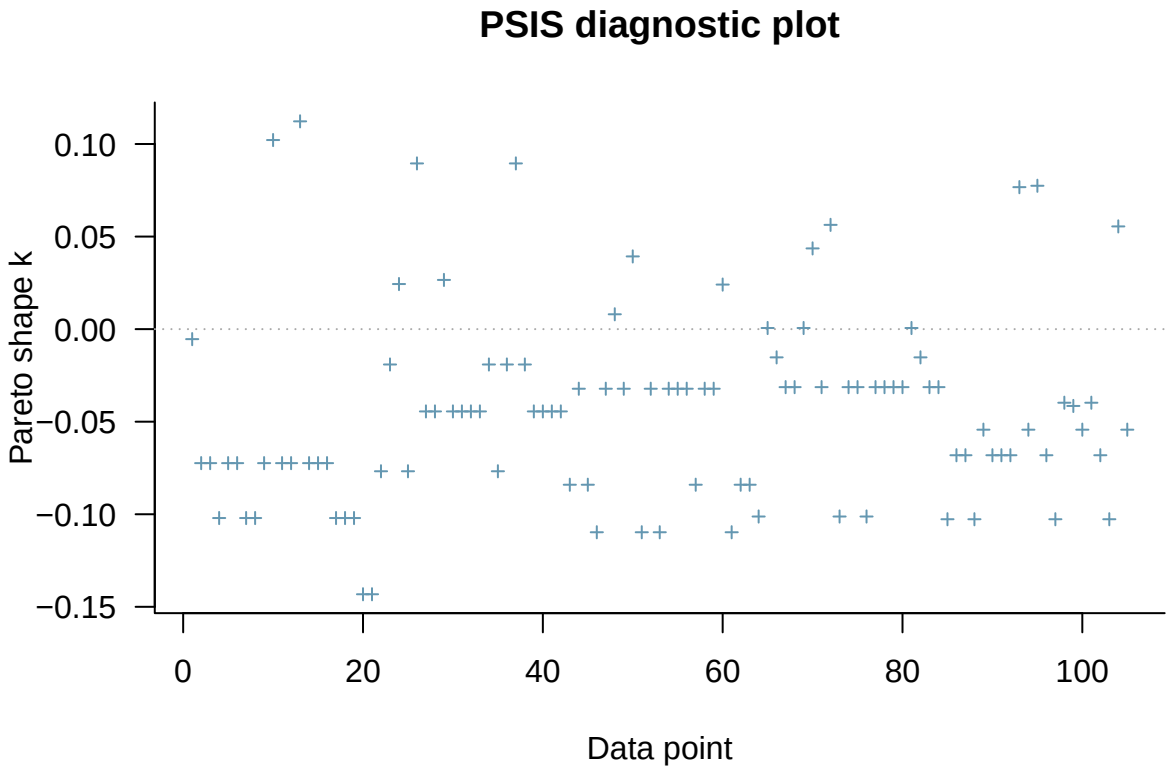
```
## Warning: Removed 2 rows containing missing values (geom_bar).
## Removed 2 rows containing missing values (geom_bar).
```




```
hier_extracted = extract_log_lik(hier_fit, merge_chains = FALSE)
r_eff = relative_eff(exp(hier_extracted))
hier_loo = loo(hier_extracted, r_eff = r_eff)
hier_elpd = hier_loo$estimates["elpd_loo",1]
paste("Hierarchical model PSIS-LOO elpd value: ", round(hier_elpd,1))
```

```
## [1] "Hierarchical model PSIS-LOO elpd value:  -357.8"
```

```
plot(hier_loo)
```



Discussion of issues and potential improvements

Conclusion what was learned from the data analysis

Self-reflection