How Environmental Conditions Affect Raptor Movement

VS1003280193

20/04/2021

Abstract: The goal of this project is to understand how environmental conditions affect animal behavior. In our case, we are focused on analyzing the movement states of raptors (large birds). Raptors have two types of movement states: high energy state in which they move a lot, and a low energy state in which the bird is resting or not moving much. Given an indicatior of state Z, where Z= 1 corresponds to the low energy state, and Z= 2 corresponds to the high energy state, I model the log-MSA as yi|Z,μ,σ~N(μZ,σ2Z). I further assume that for each observation, the corresponding state Zi is an iid draw from a binomial distribution with Pr(Z= 2) =pi, which might vary by individual. I used STAN to fit two different 2-component mixture models for the given data. The first model has afixed p, while the second model has the probability of being in state 2 as logit(pi) =β0+β1wind+β2temp. Then,I conducted both prior and posterior predictive checks to identify how well each model fit. Next, I used leave-one-out cross validation to select the best model.

```
#preview of data set
dat <-read.csv("Verreauxs.accel.txt", sep = "\t")
head(dat)
```

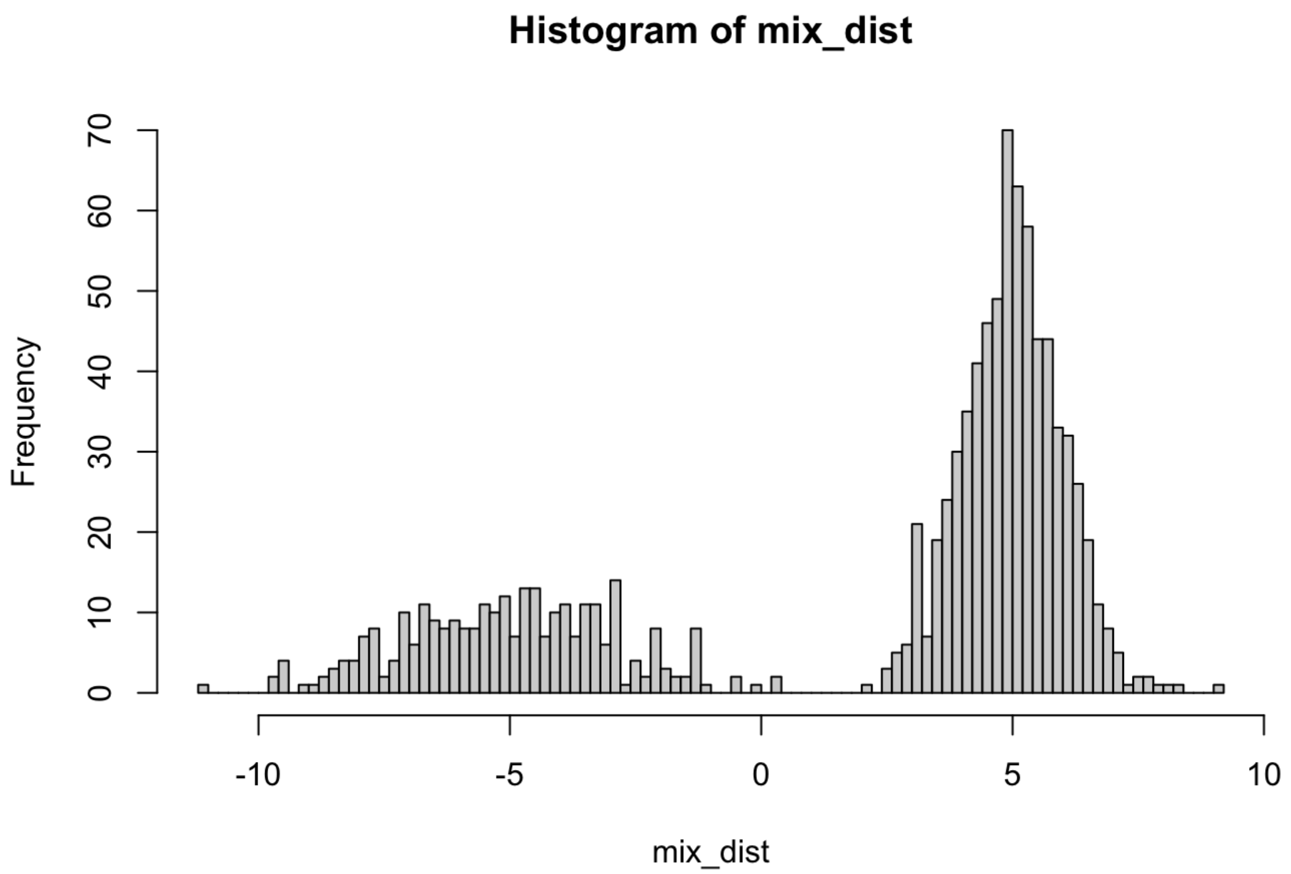
```
##             date_time        msa wind_speed saws_temp hrseg
## 1 2013-04-16 13:03:44 0.23787533        3.6      16.5     1
## 2 2013-04-16 13:05:38 0.09222497        3.6      16.5     1
## 3 2013-04-16 13:07:27 0.27394695        3.6      16.5     1
## 4 2013-04-16 13:37:51 0.28776082        3.6      16.5     5
## 5 2013-04-16 13:39:44 0.59848213        3.6      16.5     5
## 6 2013-04-16 13:41:36 0.12466199        3.6      16.5     5
```

• MSA: our observation • wind_speed: The wind speed (m/s) • saws_temp: Temperature (celsius)

Task 1:

```
#simulate data
length_dataset = 1000
m1 = -5
m2 = 5
sd1 = 2
sd2 = 1
mix_dist <- c()
p = 0.7
for (i in 1:length_dataset){
  Z <- rbinom(1, 1, 0.7) + 1 #is individual i in state 1 or 2
  if (Z == 1){
    mix_dist[i] <- rnorm(1, m1, sd1)
  } else {
    mix_dist[i] <- rnorm(1, m2, sd2)
  }
}

#check mixture distribution
hist(mix_dist, breaks = 100)
```



I decided to use a binomial model to simulate Z[i] because each individual i in the model is either in state 1 or state 2. The data with m1 = -5 and sd1 = 2 has a low frequency with respect to the mix distribution. Whereas the simulated data with m1 = 5 and sd2 = 1 resembles a normal distribution more closely and has a higher frequency. In this model, we happen to know that the two mixture components should correspond to the two energy states.

At this stage, the model does not know anything about the two states. For this model, the latent variable is the energy state and the observable variable is the minimum specific acceleration (MSA). First I'll sample Z, and then sample the observables x from a distribution which depends on z.

```
#prior predictive check

stan.data <- list(y = mix_dist,
                  N = length_dataset,
                  K = 2,
                  run_estimation = 0) #we give the data as imput but the model is specif
ied
                                      #to not use it when run_estimation = 0
#define model
writeLines(readLines("simple_mixture_model.stan"))
```

```
##
## data {
##
##   int<lower=0> N;  //the length of the data
##   int<lower=1> K; // the number of distributions in the mixture
##   vector[N] y; // msa
##   int run_estimation;
## }
##
##
## parameters {
##   ordered[K] mu;
##   vector<lower=0.01, upper=100>[K] sigma;
##   real<lower=0.01, upper=0.99> p;
## }
##
##
## model {
##   //priors
##   mu ~ normal(0, 1000); //a very vague prior
##   sigma ~ normal(0, 1);
##   p ~ beta(2,2); //somewhat vague
##
##   if (run_estimation==1){
##     for (i in 1:N){
##       target += log_mix(p, normal_lpdf(y[i] | mu[2], sigma[2]),
##                            normal_lpdf(y[i] | mu[1], sigma[1]));
##     }
##   }
## }
##
## generated quantities{
##
##   vector[N] log_lik;
##   vector[N] y_pred;
##
##   for (i in 1:N){
##     log_lik[i] = bernoulli_rng(p)+1;
##     if (log_lik[i]==1){
##       y_pred[i] = normal_rng(mu[1],sigma[1]);
##     }
##     if (log_lik[i]==2){
##       y_pred[i] = normal_rng(mu[2],sigma[2]);
##     }
##   }
## }
```

```
file <- file.path("simple_mixture_model.stan")
mod <- cmdstan_model(file)
```

```
## Model executable is up to date!
```

```
mod$print()
```

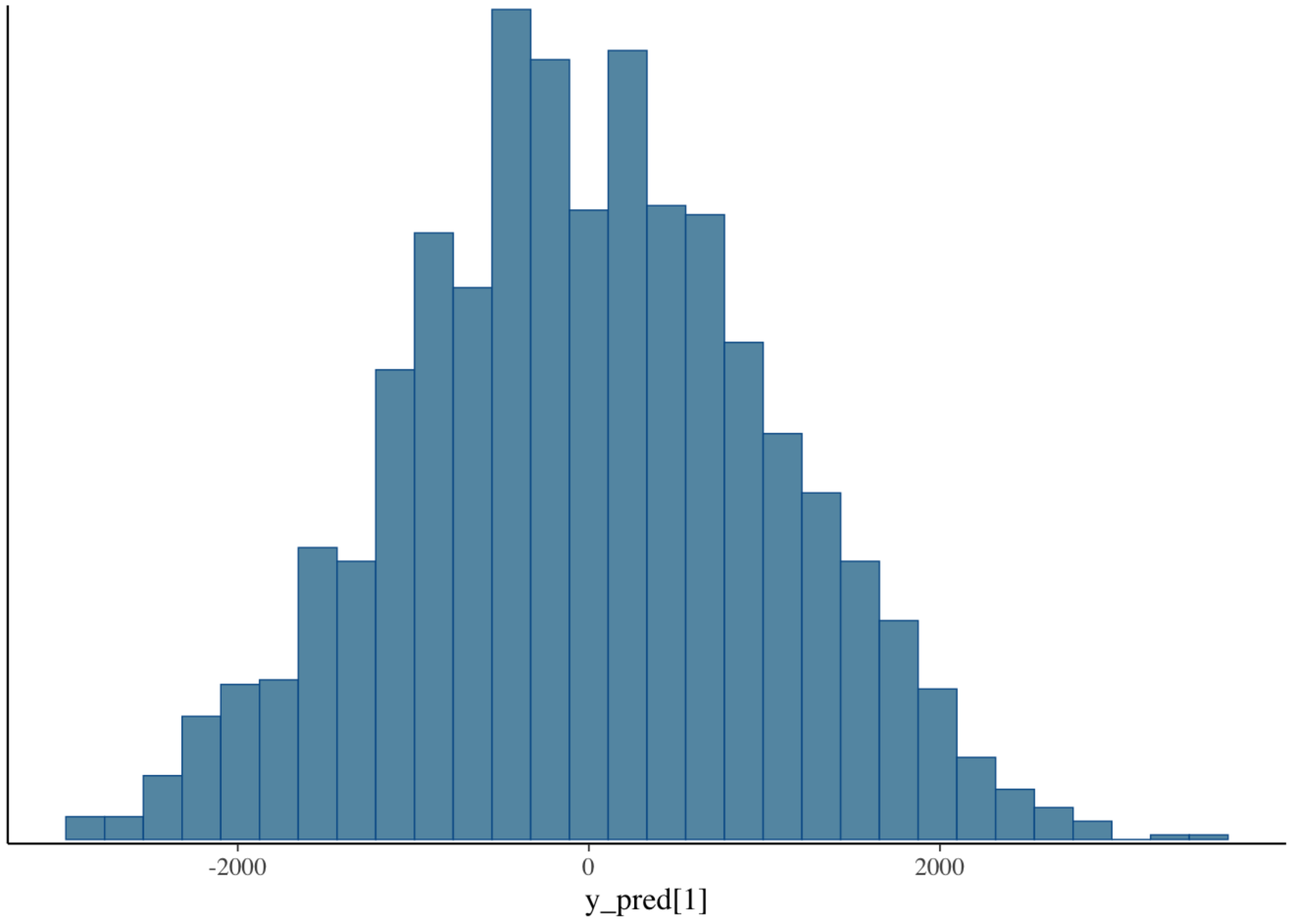
```
##
## data {
##
##   int<lower=0> N;  //the length of the data
##   int<lower=1> K; // the number of distributions in the mixture
##   vector[N] y; // msa
##   int run_estimation;
## }
##
##
## parameters {
##   ordered[K] mu;
##   vector<lower=0.01, upper=100>[K] sigma;
##   real<lower=0.01, upper=0.99> p;
## }
##
##
## model {
##   //priors
##   mu ~ normal(0, 1000); //a very vague prior
##   sigma ~ normal(0, 1);
##   p ~ beta(2,2); //somewhat vague
##
##   if (run_estimation==1){
##     for (i in 1:N){
##       target += log_mix(p, normal_lpdf(y[i] | mu[2], sigma[2]),
##                            normal_lpdf(y[i] | mu[1], sigma[1]));
##     }
##   }
## }
##
## generated quantities{
##
##   vector[N] log_lik;
##   vector[N] y_pred;
##
##   for (i in 1:N){
##     log_lik[i] = bernoulli_rng(p)+1;
##     if (log_lik[i]==1){
##       y_pred[i] = normal_rng(mu[1],sigma[1]);
##     }
##     if (log_lik[i]==2){
##       y_pred[i] = normal_rng(mu[2],sigma[2]);
##     }
##   }
## }
```

```
#run the sampler
prior_predictive_check <- mod$sample(
  data = stan.data,
  seed = 123,
  chains = 2,
  parallel_chains = 2,
  refresh = 500,
  iter_warmup = 1000,
  iter_sampling = 1000)
```

```
## Running MCMC with 2 parallel chains...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 1.5 seconds.
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 finished in 1.5 seconds.
##
## Both chains finished successfully.
## Mean chain execution time: 1.5 seconds.
## Total execution time: 1.8 seconds.
```

```
#prior predictive distribution and summary
param <- "y_pred[1]"
mcmc_hist(prior_predictive_check$draws(param)) #because we use vague prior information,
 our prior predictive distributions are also very vague
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
prior_predictive_check$summary(param)
```

```
## # A tibble: 1 x 10
##   variable   mean median    sd   mad     q5   q95  rhat ess_bulk ess_tail
##   <chr>     <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>    <dbl>    <dbl>
## 1 y_pred[1] -12.3  -44.8 1053. 1064. -1760. 1789.  1.00    1597.    1249.
```

```
#fit on simulated data

stan.data <- list(y = mix_dist,
                  N = length_dataset,
                  K = 2,
                  run_estimation = 1)

#define model
file <- file.path("simple_mixture_model.stan")
mod <- cmdstan_model(file)
```

```
## Model executable is up to date!
```

```
mod$print()
```

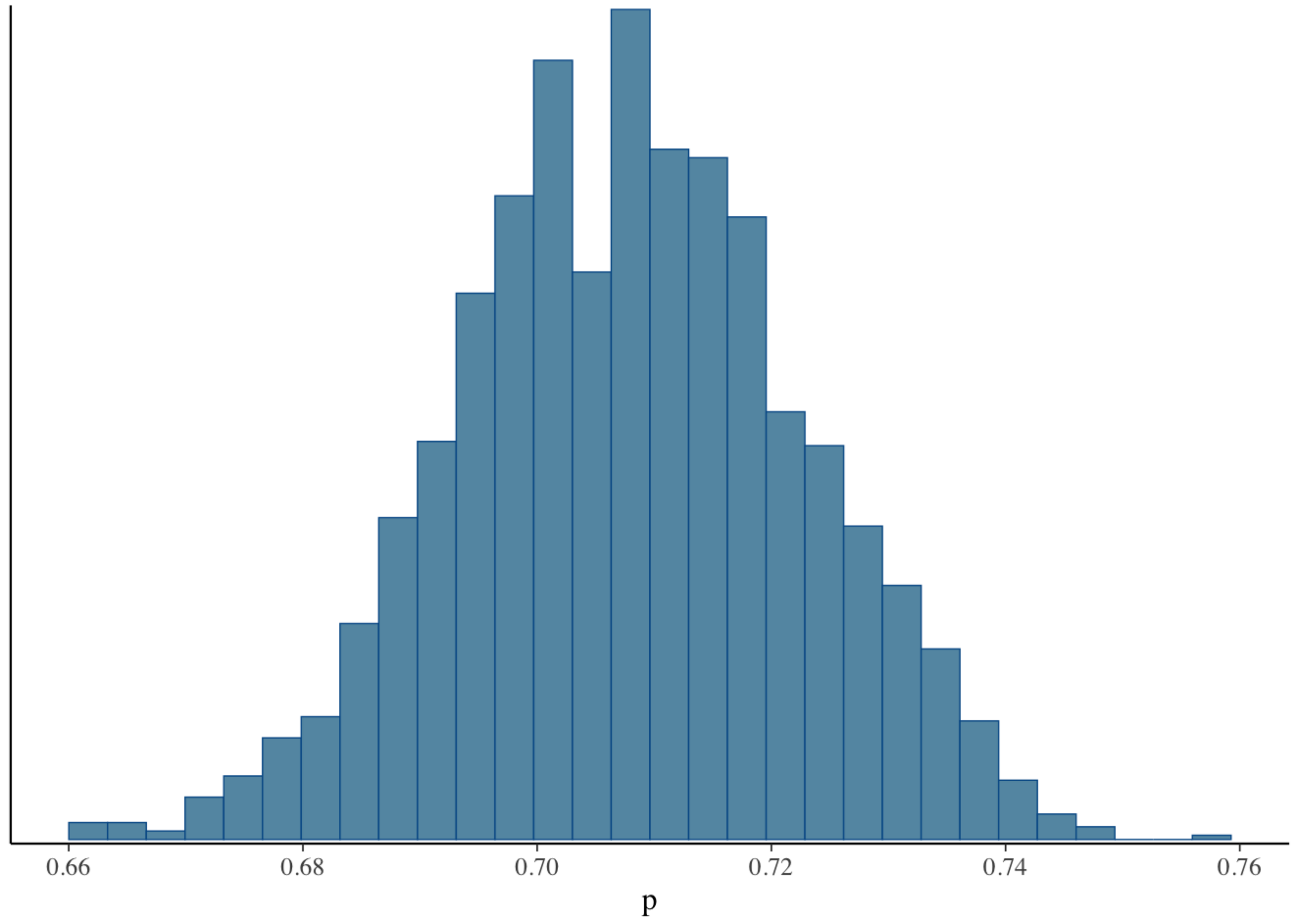
```
##
## data {
##
##   int<lower=0> N;  //the length of the data
##   int<lower=1> K; // the number of distributions in the mixture
##   vector[N] y; // msa
##   int run_estimation;
## }
##
##
## parameters {
##   ordered[K] mu;
##   vector<lower=0.01, upper=100>[K] sigma;
##   real<lower=0.01, upper=0.99> p;
## }
##
##
## model {
##   //priors
##   mu ~ normal(0, 1000); //a very vague prior
##   sigma ~ normal(0, 1);
##   p ~ beta(2,2); //somewhat vague
##
##   if (run_estimation==1){
##     for (i in 1:N){
##       target += log_mix(p, normal_lpdf(y[i] | mu[2], sigma[2]),
##                            normal_lpdf(y[i] | mu[1], sigma[1]));
##     }
##   }
## }
##
## generated quantities{
##
##   vector[N] log_lik;
##   vector[N] y_pred;
##
##   for (i in 1:N){
##     log_lik[i] = bernoulli_rng(p)+1;
##     if (log_lik[i]==1){
##       y_pred[i] = normal_rng(mu[1],sigma[1]);
##     }
##     if (log_lik[i]==2){
##       y_pred[i] = normal_rng(mu[2],sigma[2]);
##     }
##   }
## }
```

```
#run model
simple_mixture_fit <- mod$sample(
  data = stan.data,
  seed = 123,
  chains = 2,
  parallel_chains = 2,
  refresh = 500,
  iter_warmup = 1000,
  iter_sampling = 1000)
```

```
## Running MCMC with 2 parallel chains...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 12.5 seconds.
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 finished in 13.1 seconds.
##
## Both chains finished successfully.
## Mean chain execution time: 12.8 seconds.
## Total execution time: 13.2 seconds.
```

```
#posterior distribution and summary
param <- "p"
mcmc_hist(simple_mixture_fit$draws(param))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

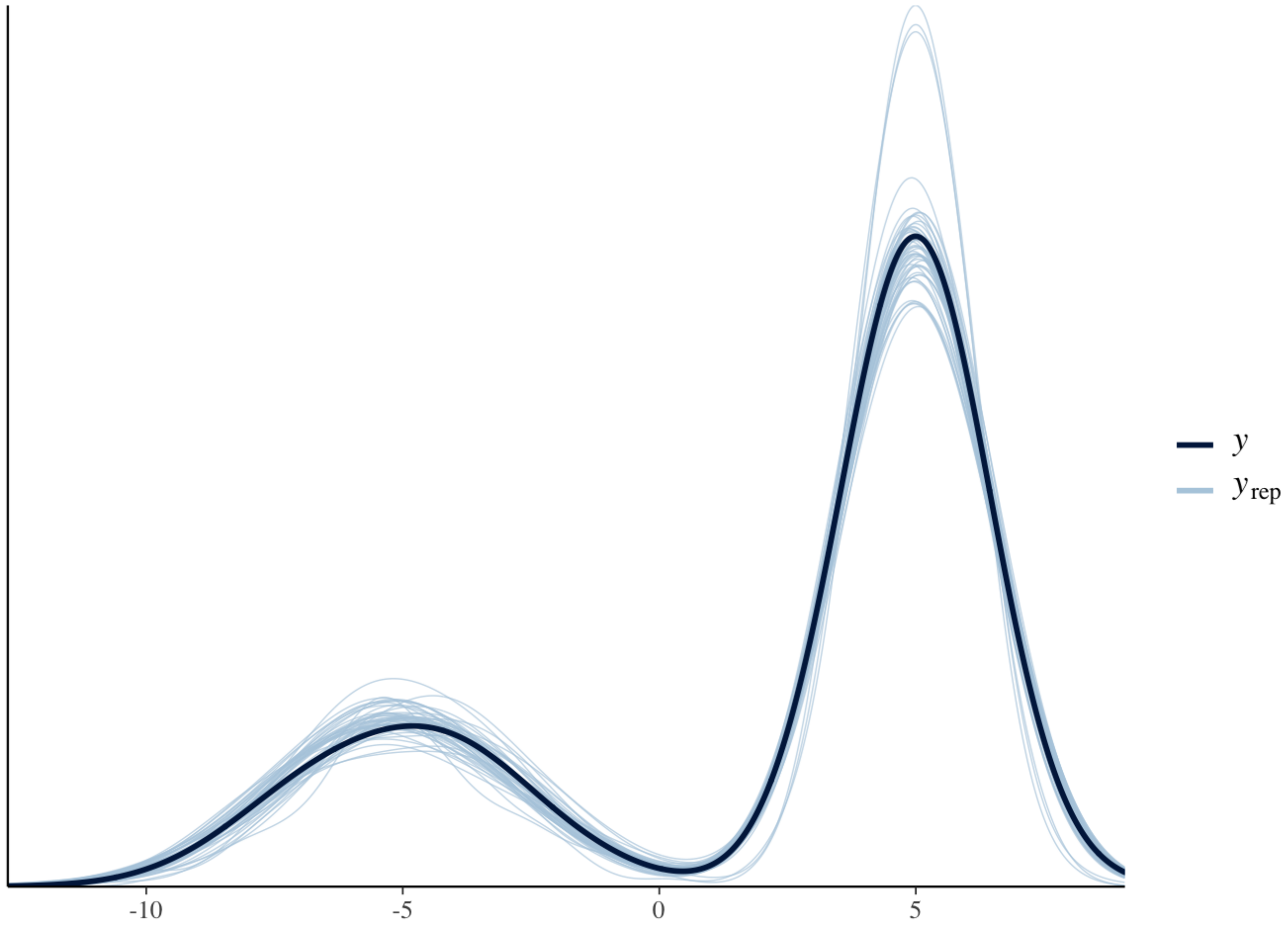


```
simple_mixture_fit$summary(param)
```

```
## # A tibble: 1 x 10
##   variable  mean median     sd    mad    q5   q95  rhat ess_bulk ess_tail
##   <chr>    <dbl>  <dbl>  <dbl>  <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
## 1 p        0.708  0.708 0.0147 0.0148 0.684 0.733 0.999    1686.    1239.
```

The priors I chose for mu and sigma were both normal distributions as the Central Limit Theorem is applicable for this dataset. Mu is normally distributed with mean equal to zero and standard deviation equal to 1000, this is a vague prior which is chosen in an attempt to not favour one area of the parameter space over another. Sigma is normally distributed with mean equal to zero and standard deviation equal to 1. These priors are chosen with the intention to constrain the data within the given scale. Clearly, the histogram is approximately bell-shaped and symmetric about the mean at zero, which resembles a normal distribution as expected. Overall the convergence looks good for both parameters.

```
#posterior predictive check
y_pred <- as_draws_matrix(simple_mixture_fit$draws("y_pred"))
ppc_dens_overlay(mix_dist, y_pred[1:50,]) #bold navy blue line => data, light blue lines
=> samples from posterior
```



In the plot above, the dark line is the distribution of the observed outcomes and the lighter line is the density estimate from one of the replications of y from the posterior. This plot makes it easy to see this model is quite promising. The light blue lines of the predictive posterior are mostly covering the dark blue line of the original data. However, our generated model assumes every day the weather conditions arekept constant which is likely not the case and could affect the energy levels of the raptors. Overall, the posterior does seem capable of producing data similar to what is observed.

Task 2:

```
#Model 1 -- fixed p
stan.data <- list(y = dat$msa,
                  N = length(dat$msa),
                  K = 2,
                  run_estimation = 1)

#define model
writeLines(readLines("2component_mixture_model_1.stan"))
```

```
##
## data {
##   int<lower=0> N;
##   int<lower=1> K;
##   vector[N] y;
##   int run_estimation;
## }
##
##
## parameters {
##   ordered[K] mu;
##   vector<lower=0.01, upper=100>[K] sigma;
##   real<lower=0.01, upper=0.99> p;
## }
##
## transformed parameters {
##
## }
##
## model {
##   //priors
##   mu ~ normal(0, 100);
##   sigma ~ normal(0, 4);
##   p ~ beta(1,1);
##
##   //likelihood
##   if (run_estimation==1){
##     for (i in 1:N){
##       target += log_mix(p, normal_lpdf(log(y[i]) | mu[2], sigma[2]),
##                            normal_lpdf(log(y[i]) | mu[1], sigma[1]));
##     }
##   }
## }
##
## generated quantities{
##   vector[N] log_lik;
##   vector[N] y_pred;
##
##   for (i in 1:N){
##     log_lik[i] = bernoulli_rng(p)+1;
##     if (log_lik[i]==1){
##       y_pred[i] = exp(normal_rng(mu[1],sigma[1]));
##     }
##     if (log_lik[i]==2){
##       y_pred[i] = exp(normal_rng(mu[2],sigma[2]));
##     }
##   }
## }
```

```
file <- file.path("2component_mixture_model_1.stan")
mod <- cmdstan_model(file)
```

```
## Model executable is up to date!
```

```
mod$print()
```

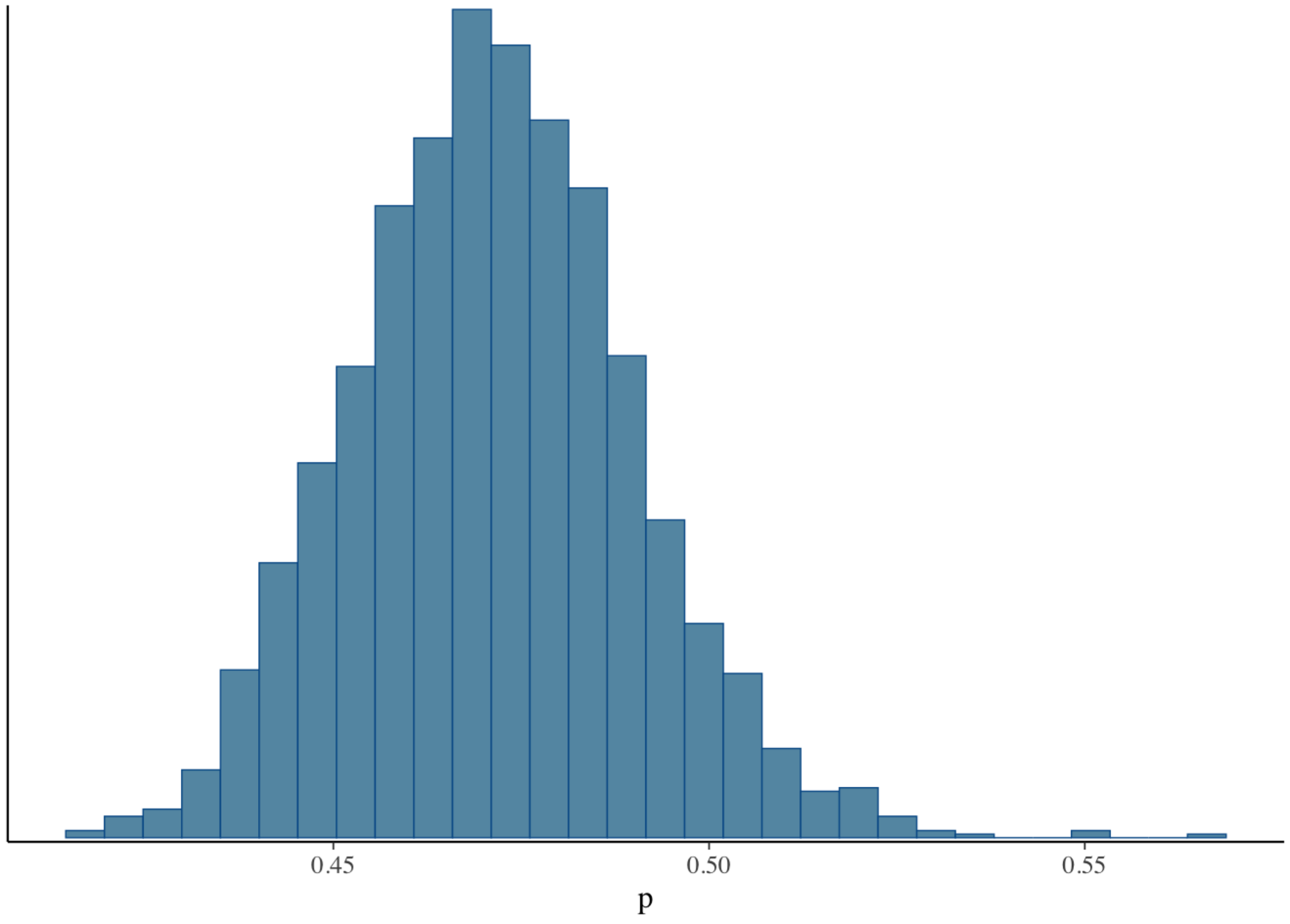
```
##
## data {
##   int<lower=0> N;
##   int<lower=1> K;
##   vector[N] y;
##   int run_estimation;
## }
##
##
## parameters {
##   ordered[K] mu;
##   vector<lower=0.01, upper=100>[K] sigma;
##   real<lower=0.01, upper=0.99> p;
## }
##
## transformed parameters {
##
## }
##
## model {
##   //priors
##   mu ~ normal(0, 100);
##   sigma ~ normal(0, 4);
##   p ~ beta(1,1);
##
##   //likelihood
##   if (run_estimation==1){
##     for (i in 1:N){
##       target += log_mix(p, normal_lpdf(log(y[i]) | mu[2], sigma[2]),
##                            normal_lpdf(log(y[i]) | mu[1], sigma[1]));
##     }
##   }
## }
##
## generated quantities{
##   vector[N] log_lik;
##   vector[N] y_pred;
##
##   for (i in 1:N){
##     log_lik[i] = bernoulli_rng(p)+1;
##     if (log_lik[i]==1){
##       y_pred[i] = exp(normal_rng(mu[1],sigma[1]));
##     }
##     if (log_lik[i]==2){
##       y_pred[i] = exp(normal_rng(mu[2],sigma[2]));
##     }
##   }
## }
```

```
#run model
model_1 <- mod$sample(
  data = stan.data,
  seed = 123,
  chains = 2,
  parallel_chains = 2,
  refresh = 500,
  iter_warmup = 1000,
  iter_sampling = 1000)
```

```
## Running MCMC with 2 parallel chains...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 finished in 7.3 seconds.
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 7.6 seconds.
##
## Both chains finished successfully.
## Mean chain execution time: 7.5 seconds.
## Total execution time: 7.8 seconds.
```

```
#posterior distribution and summary
param <- "p"
mcmc_hist(model_1$draws(param))
```

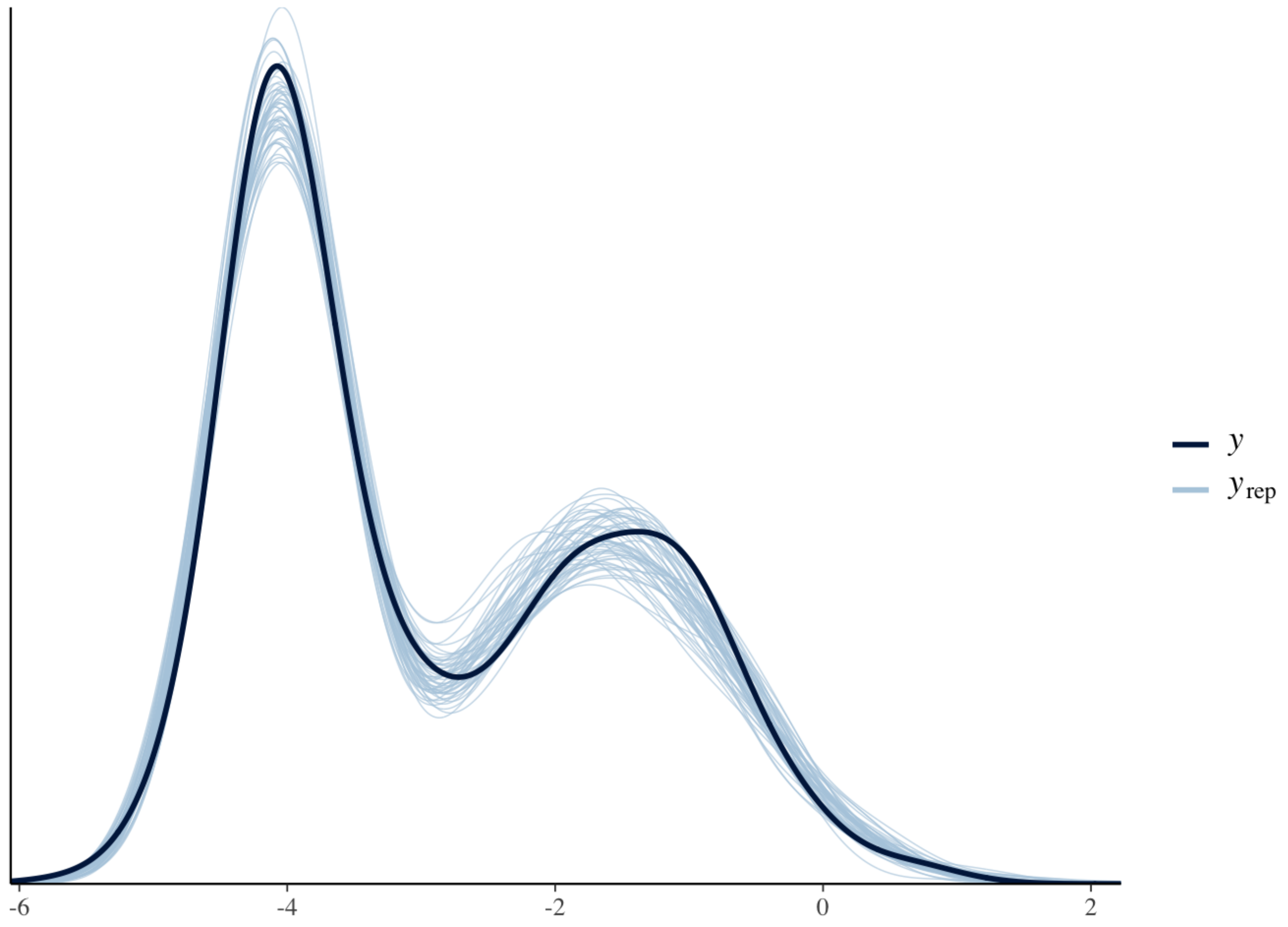
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
model_1$summary(param)
```

```
## # A tibble: 1 x 10
##   variable  mean median     sd    mad    q5   q95  rhat ess_bulk ess_tail
##   <chr>    <dbl>  <dbl>  <dbl>  <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
## 1 p        0.471  0.471 0.0187 0.0181 0.442 0.503  1.00    1005.      969.
```

```
#posterior predictive check
y_pred <- as_draws_matrix(model_1$draws("y_pred"))
ppc_dens_overlay(log(dat$msa), log(y_pred[1:50,]))
```



The posterior predictive check for Model 1 shows an almost perfect fit of the observed data and generated data. The light blue line which are from the predictive posterior overlays the dark blue which is the observed data almost exactly. In addition, the histogram which displays the posterior distribution resembles a bell shaped curve which reflects a more realistic normal distribution as desired. This suggests the model's predictions are consistent with the observed data.

```
#Model 2 -- state 2

stan.data <- list(y = dat$msa,
                  N = length(dat$msa),
                  K = 2,
                  wind = dat$wind_speed,
                  temp =dat$saws_temp,
                  run_estimation = 1)
inits_chain1 <- list(mu = c(-2,-1.5))
inits_chain2 <- list(mu = c(-3,-2.5))

#define model
writeLines(readLines("2component_mixture_model_2.stan"))
```

```
##
## data {
##   int<lower=0> N;
##   int<lower=1> K;
##   vector[N] y;
##   int run_estimation;
##   vector[N] wind;
##   vector[N] temp;
## }
##
## transformed data{
##   vector[N] log_y;
##
##   log_y = log(y);
## }
##
##
## parameters {
##   ordered[K] mu;
##   vector<lower=0.01>[K] sigma;
##   vector[3] beta;
## }
##
## transformed parameters {
##     vector[N] logit_p;
##     for (i in 1:N){
##           logit_p[i] = beta[1] + beta[2]*wind[i] + beta[3]*temp[i];
##     }
##
## }
##
## model {
##   //priors
##   mu[1] ~ normal(0, 5);
##   mu[2] ~ normal(0, 5);
##   sigma ~ normal(0, 2);
##   beta[1] ~ normal(0,5);
##   beta[2] ~ normal(0,5);
##   beta[3] ~ normal(0,5);
##
##   //likelihood
##   if (run_estimation==1){
##     for (i in 1:N){
##       target += log_mix(inv_logit(logit_p[i]), normal_lpdf(log_y[i] | mu[2], sigma
[2]),
##                            normal_lpdf(log_y[i] | mu[1], sigma[1]));
##     }
##   }
## }
##
## generated quantities{
##   vector[N] log_lik2;
##   vector[N] y_pred;
##
##   for (i in 1:N){
##     log_lik2[i] = bernoulli_logit_rng(logit_p[i])+1;
##     if (log_lik2[i]==1){
##       y_pred[i] = exp(normal_rng(mu[1],sigma[1]));
##     }
##     if (log_lik2[i]==2){
##       y_pred[i] = exp(normal_rng(mu[2],sigma[2]));
##     }
##   }
## }
```

```
file <- file.path("2component_mixture_model_2.stan")
mod <- cmdstan_model(file)
```

```
## Model executable is up to date!
```

```
mod$print()
```

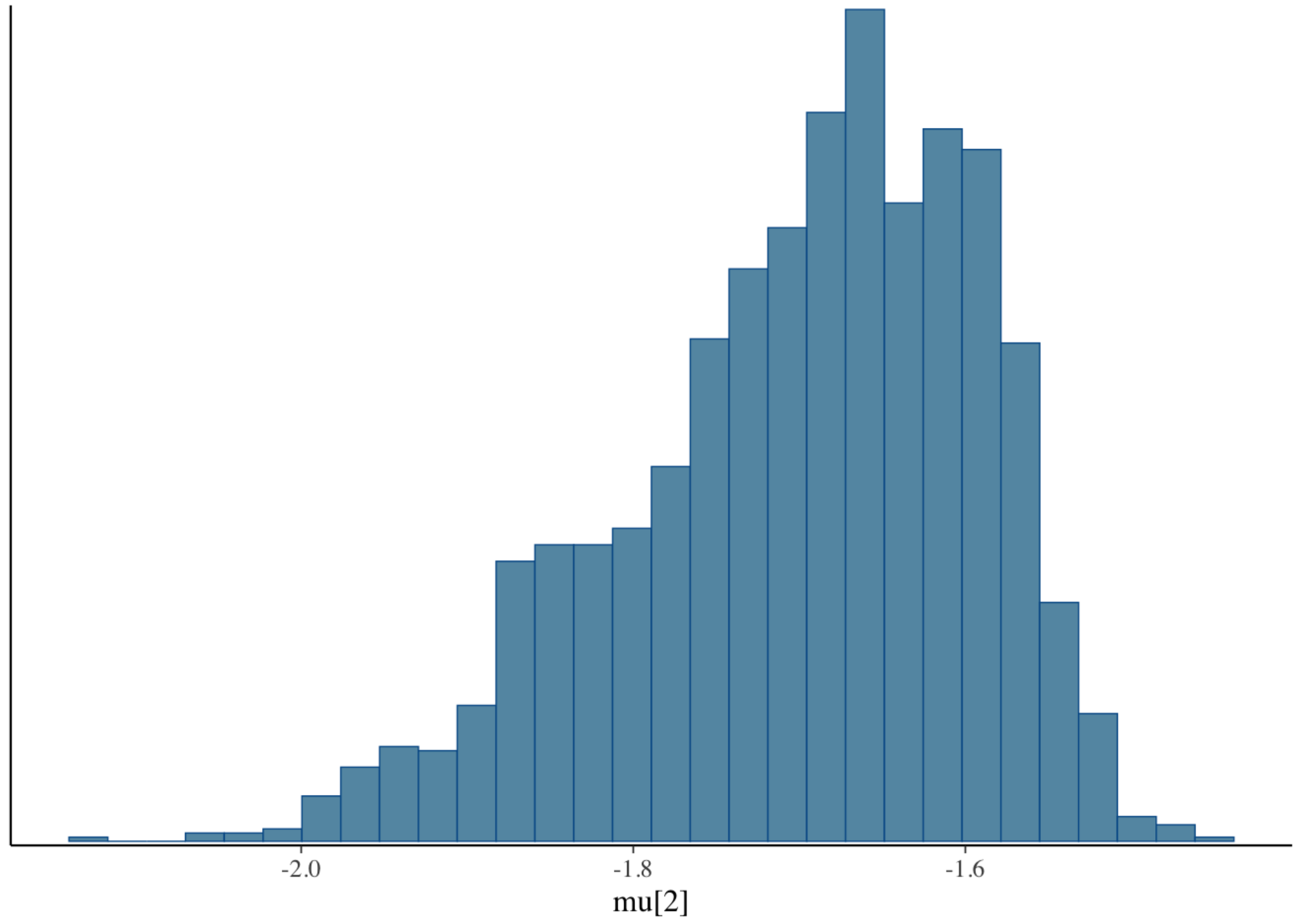
```
##
## data {
##   int<lower=0> N;
##   int<lower=1> K;
##   vector[N] y;
##   int run_estimation;
##   vector[N] wind;
##   vector[N] temp;
## }
##
## transformed data{
##   vector[N] log_y;
##
##   log_y = log(y);
## }
##
##
## parameters {
##   ordered[K] mu;
##   vector<lower=0.01>[K] sigma;
##   vector[3] beta;
## }
##
## transformed parameters {
##     vector[N] logit_p;
##     for (i in 1:N){
##           logit_p[i] = beta[1] + beta[2]*wind[i] + beta[3]*temp[i];
##     }
##
## }
##
## model {
##   //priors
##   mu[1] ~ normal(0, 5);
##   mu[2] ~ normal(0, 5);
##   sigma ~ normal(0, 2);
##   beta[1] ~ normal(0,5);
##   beta[2] ~ normal(0,5);
##   beta[3] ~ normal(0,5);
##
##   //likelihood
##   if (run_estimation==1){
##     for (i in 1:N){
##       target += log_mix(inv_logit(logit_p[i]), normal_lpdf(log_y[i] | mu[2], sigma
[2]),
##                             normal_lpdf(log_y[i] | mu[1], sigma[1]));
##     }
##   }
## }
##
## generated quantities{
##   vector[N] log_lik2;
##   vector[N] y_pred;
##
##   for (i in 1:N){
##     log_lik2[i] = bernoulli_logit_rng(logit_p[i])+1;
##     if (log_lik2[i]==1){
##       y_pred[i] = exp(normal_rng(mu[1],sigma[1]));
##     }
##     if (log_lik2[i]==2){
##       y_pred[i] = exp(normal_rng(mu[2],sigma[2]));
##     }
##   }
## }
```

```
#run model
model_2 <- mod$sample(
  data = stan.data,
  init = list(inits_chain1, inits_chain2),
  seed = 123,
  chains = 2,
  parallel_chains = 2,
  refresh = 500,
  iter_warmup = 1000,
  iter_sampling = 1000)
```

```
## Running MCMC with 2 parallel chains...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 finished in 34.4 seconds.
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 37.6 seconds.
##
## Both chains finished successfully.
## Mean chain execution time: 36.0 seconds.
## Total execution time: 37.7 seconds.
```

```
#posterior distribution and summary
param <- "mu[2]"
mcmc_hist(model_2$draws(param))
```

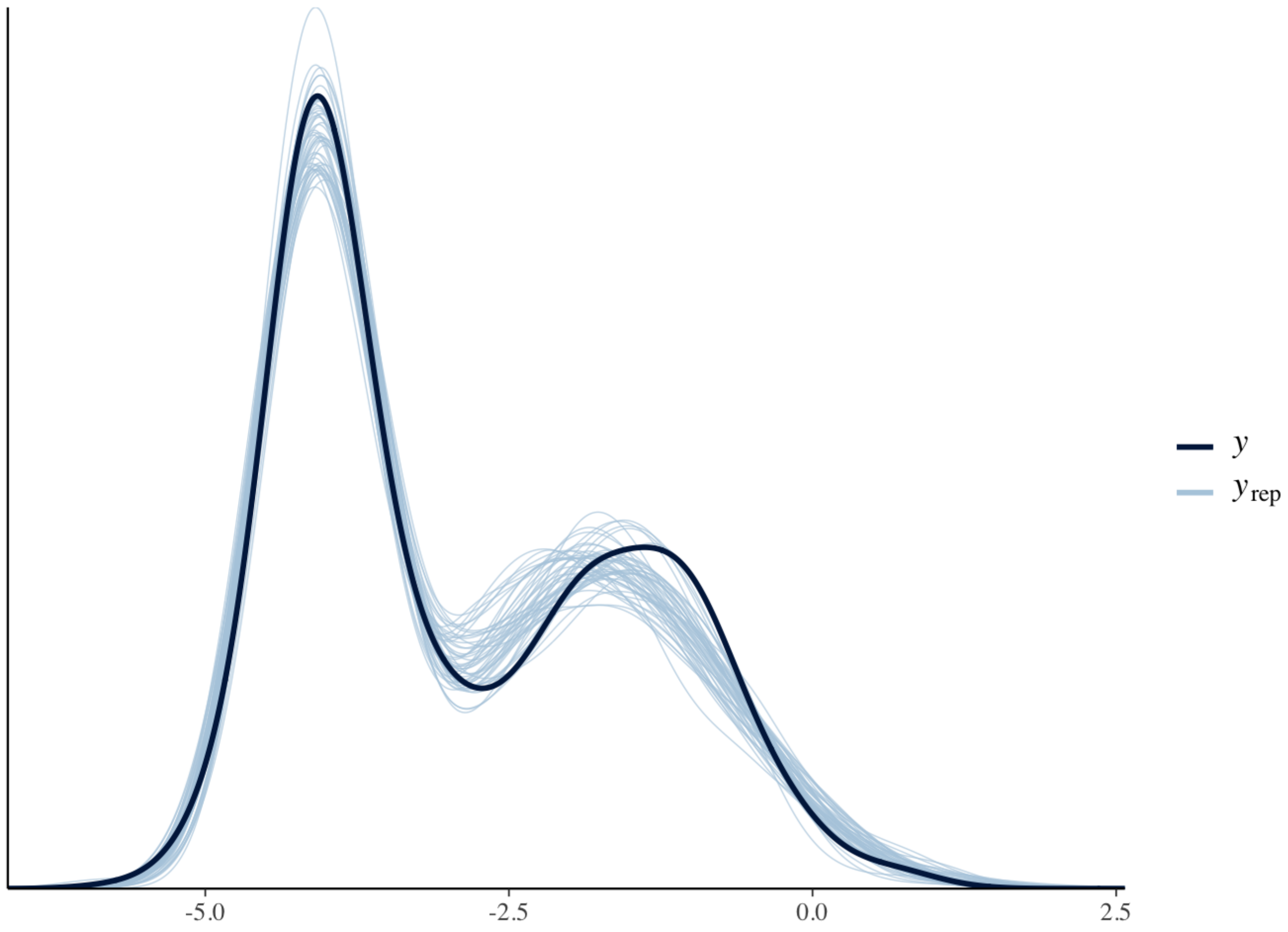
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
model_2$summary(param)
```

```
## # A tibble: 1 x 10
##   variable  mean median    sd   mad    q5   q95  rhat ess_bulk ess_tail
##   <chr>    <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
## 1 mu[2]    -1.70  -1.68 0.106 0.107 -1.89 -1.56  1.01     440.     844.
```

```
#posterior predictive check
y_pred <- as_draws_matrix(model_2$draws("y_pred"))
ppc_dens_overlay(log(dat$msa), log(y_pred[1:50,])) #the bold line refers to the data, th
e light lines are samples from the posterior predictive distribution
```



The posterior predictive check for Model 2 shows a decent fit of the observed data and generated data. The light blue line which are from the predictive posterior overlays the dark blue which is the observed data for the most part, with some deviations from the dark line in areas such as the peak and center of curve. In addition, the histogram which displays the posterior distribution does not resemble a bell shaped curve which fails to reflect a normal distribution. The histogram appears to be skewed left, meaning a large portion of the generated observations are larger than expected. The mean value of this model is also less than the median. This suggests the model’s predictions are not uniformly consistent with the observed data.

```
#leave-one-out Cross Validation

#LOO-CV model 1
loo1 <- model_1$loo(save_psis = TRUE)
```

```
##
## Computed from 2000 by 1219 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo   1681.7 4.7
## p_loo       270.6 0.8
## looic     -3363.4 9.4
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)        0    0.0%  <NA>
##  (0.5, 0.7]   (ok)          0    0.0%  <NA>
##    (0.7, 1]   (bad)         0    0.0%  <NA>
##    (1, Inf)   (very bad) 1219  100.0%  1224
## See help('pareto-k-diagnostic') for details.
```

```
loo_compare(loo1, loo2)
```

```
##        elpd_diff se_diff
## model2   0.0       0.0
## model1 -32.1       4.7
```

Comparing the elpd values of the two models, model1 has an elpd value of 1649.6 with an SE value of 0.3 and model2 has an elpd value of 1681.7 with an SE value of 4.7. Note, the two elpd values are close in value, therefore it is not obvious to choose a “better” model. The se_diff is not too large, so I can conclude it is not likely there is much difference in the predictive accuracy of the two models. However, the elpd_diff is -32.1 which implies the expected predictive accuracy for the first model (model1) is higher. This interpretation aligns with the predictive posterior checks done above for the two models. In my analysis, I concluded model1 followed a close normal distribution, and its generated data overlapped the observed data very closely. Therefore, choosing model1 makes the most sense in this case.

Task 3:

```
#stan.data <- list(y = dat$msa,  N = length(dat$msa), K = 3, run_estimation = 1)

#define model
writeLines(readLines("3component_mixture_model.stan"))
```

```
##
## data {
##   int<lower=0> N;
##   int<lower=1> K;
##   vector[N] y;
##   int run_estimation;
## }
##
##
## parameters {
##   ordered[K] mu;
##   vector<lower=0.01, upper=100>[K] sigma;
##   real<lower=0.01, upper=0.99> p;
## }
##
## transformed parameters {
##
## }
##
## model {
##   //priors
##   mu ~ normal(0, 100);
##   sigma ~ normal(0, 4);
##   p ~ beta(1,1);
##
##   //likelihood
##   if (run_estimation==1){
##     for (i in 1:N){
##       target += log_mix(p, normal_lpdf(log(y[i]) | mu[3], sigma[3]),
##                            normal_lpdf(log(y[i]) | mu[2], sigma[2]),
##                            normal_lpdf(log(y[i]) | mu[1], sigma[1]));
##     }
##   }
## }
##
## generated quantities{
##   vector[N] log_lik;
##   vector[N] y_pred;
##
##   for (i in 1:N){
##     log_lik[i] = bernoulli_rng(p)+1;
##     if (log_lik[i]==1){
##       y_pred[i] = exp(normal_rng(mu[1],sigma[1]));
##     }
##     if (log_lik[i]==2){
##       y_pred[i] = exp(normal_rng(mu[2],sigma[2]));
##     }
##     if (log_lik[i]==3){
##       y_pred[i] = exp(normal_rng(mu[3],sigma[3]));
##     }
##   }
## }
```

```
#file <- file.path("3component_mixture_model.stan")
#mod <- cmdstan_model(file)
#mod$print()

#run model
#model_1 <- mod$sample(
  #data = stan.data,
 # seed = 123,
 # chains = 2,
  #parallel_chains = 2,
  #refresh = 500,
  #iter_warmup = 1000,
  #iter_sampling = 1000)

#posterior distribution and summary
#param <- "p"
#mcmc_hist(model_1$draws(param))
#model_1$summary(param)

#posterior predictive check
#y_pred <- as_draws_matrix(model_1$draws("y_pred"))
#ppc_dens_overlay(log(dat$msa), log(y_pred[1:50,]))
```