Assignment 2

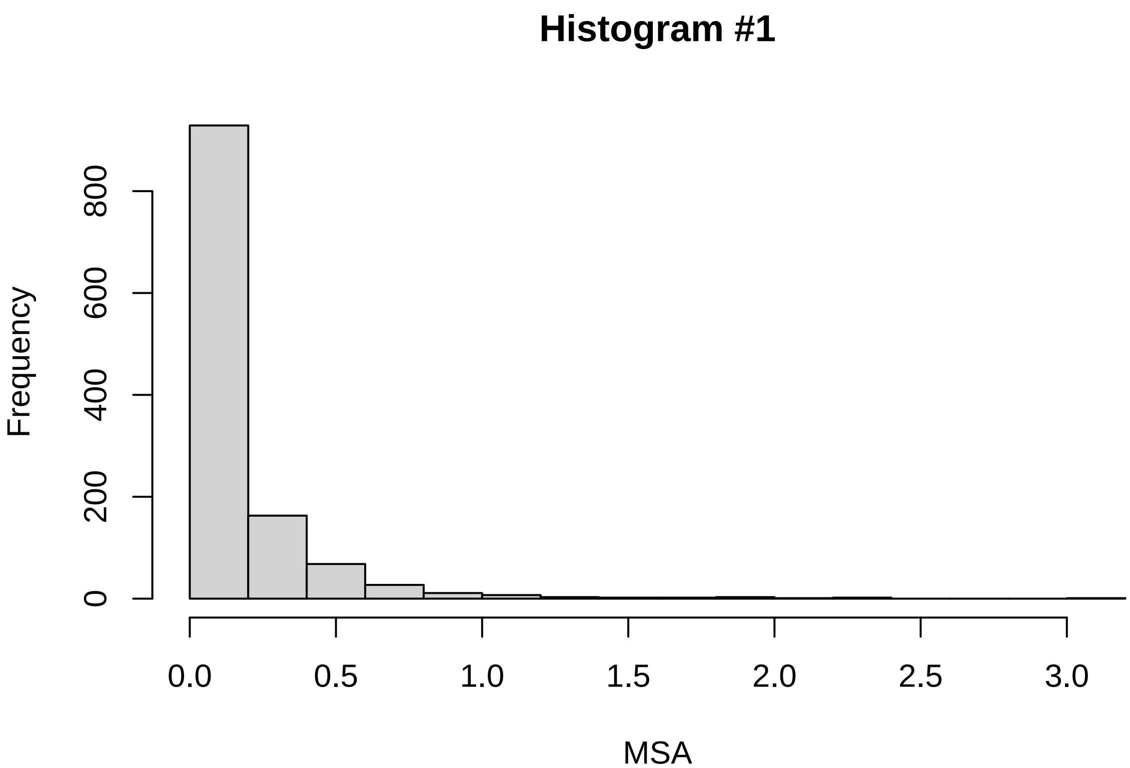
Ilke Sun

16/04/2021

Task 1

```r
dat <- dat %>%
  mutate(log_msa = log(msa))

hist(dat$msa, xlab = "MSA", main = "Histogram #1")
```




```r
mod0 <- cmdstan_model("mod0.stan")
```

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

```r
print(mod0)
```

```
## data {
##   int<lower=0> K;
##   int<lower=0> N;
##   real y[N];
## }
```

```
##
## parameters {
##   simplex[K] p;
##   ordered[K] mu;
##   vector<lower=0>[K] sigma;
## }
##
## model {
##   vector[K] log_p = log(p);
##   sigma ~ normal(0, 3);
##   mu ~ normal(0, 8);
##
##   for (n in 1:N) {
##     vector[K] lps = log_p;
##     for (k in 1:K)
##       lps[k] += normal_lpdf(y[n] | mu[k], sigma[k]);
##     target += log_sum_exp(lps);
##   }
## }
```

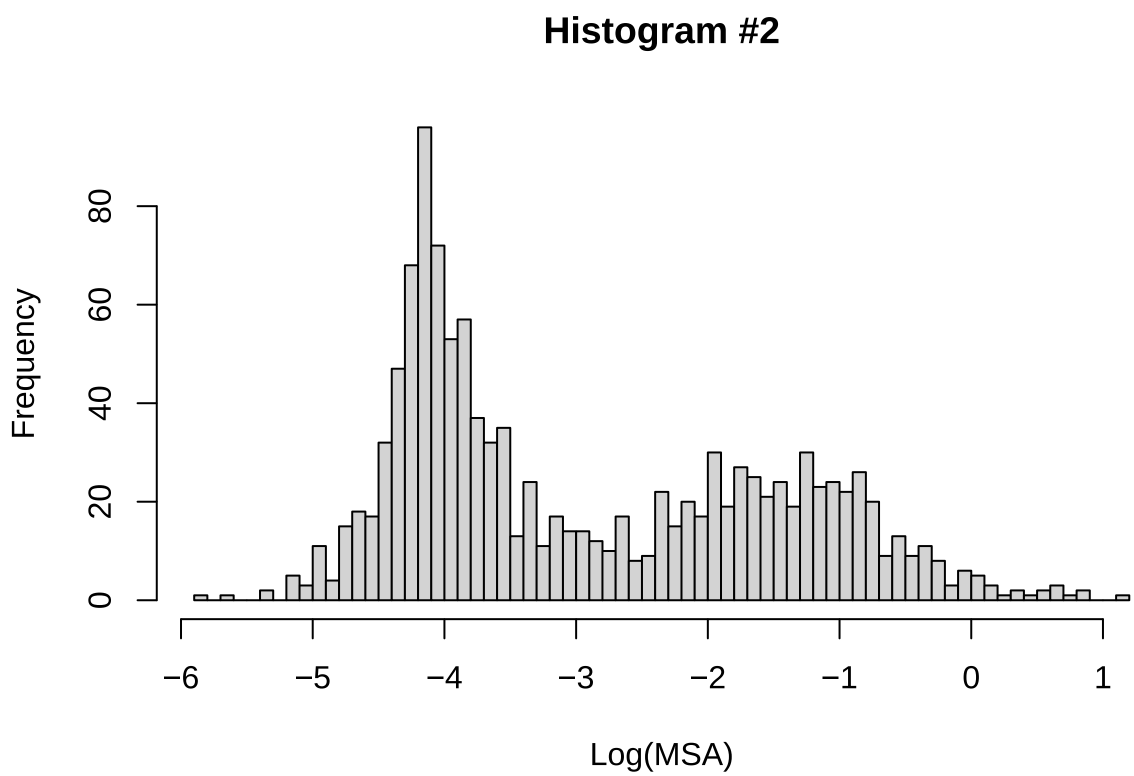
```r
stan_dat <- list(K = 2, N = 1219, y = dat$log_msa)
fit0 <- mod0$sample(stan_dat, parallel_chains = 4, refresh = 0, seed = 666,
                    show_messages = F)
```

```
## Running MCMC with 4 parallel chains...
##
## Chain 3 finished in 8.0 seconds.
## Chain 4 finished in 8.1 seconds.
## Chain 1 finished in 8.5 seconds.
## Chain 2 finished in 9.2 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 8.4 seconds.
## Total execution time: 9.4 seconds.
```

```r
print(fit0)
```

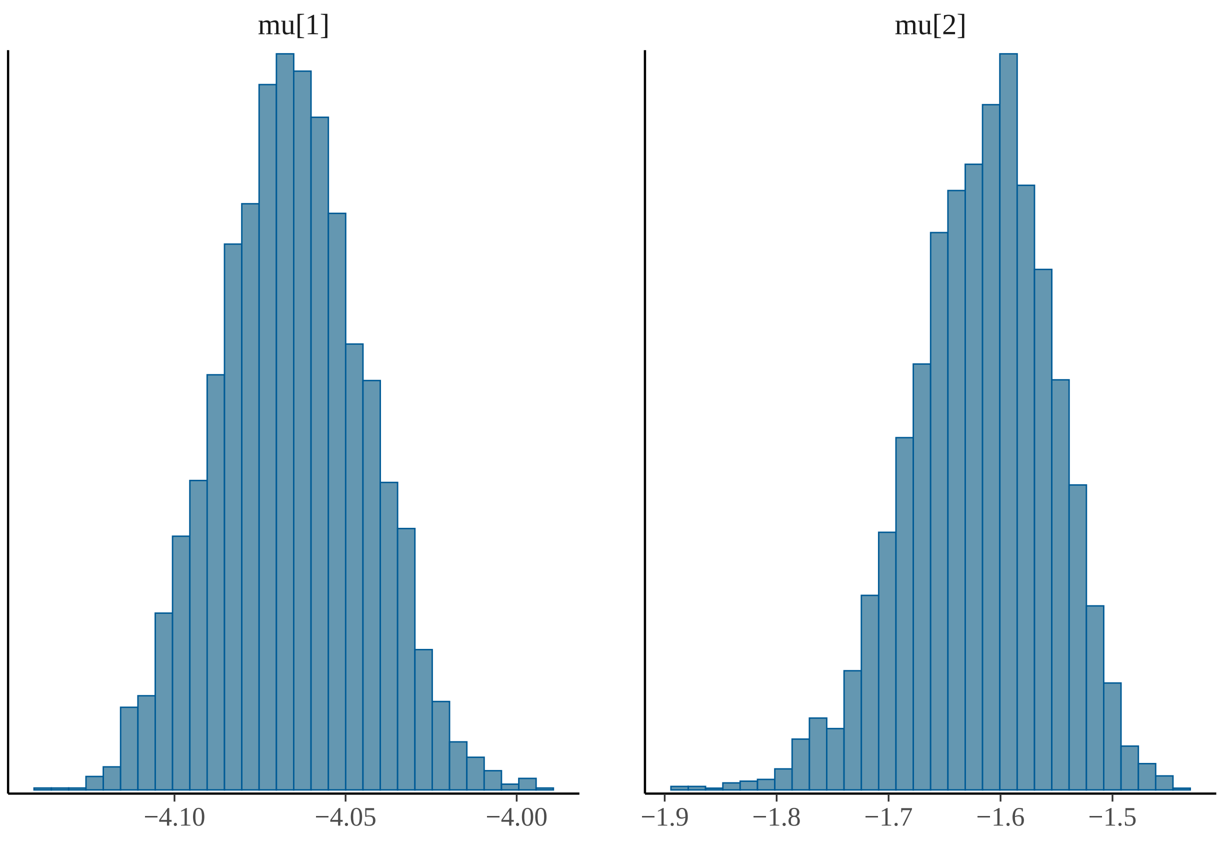
```
##  variable     mean   median   sd  mad       q5      q95 rhat ess_bulk ess_tail
##  lp__     -1849.19 -1848.90 1.55 1.41 -1852.18 -1847.26 1.00     1906     2249
##  p[1]         0.53     0.53 0.02 0.02     0.50     0.56 1.00     2132     2496
##  p[2]         0.47     0.47 0.02 0.02     0.44     0.50 1.00     2132     2496
##  mu[1]       -4.07    -4.07 0.02 0.02    -4.10    -4.03 1.00     2679     2652
##  mu[2]       -1.62    -1.61 0.06 0.06    -1.73    -1.52 1.00     2183     1790
##  sigma[1]     0.41     0.41 0.02 0.02     0.38     0.44 1.00     2278     1970
##  sigma[2]     0.93     0.93 0.05 0.05     0.86     1.03 1.00     2202     1735
```

```r
hist(dat$log_msa, xlab = "Log(MSA)", main = "Histogram #2", breaks = 90)
```



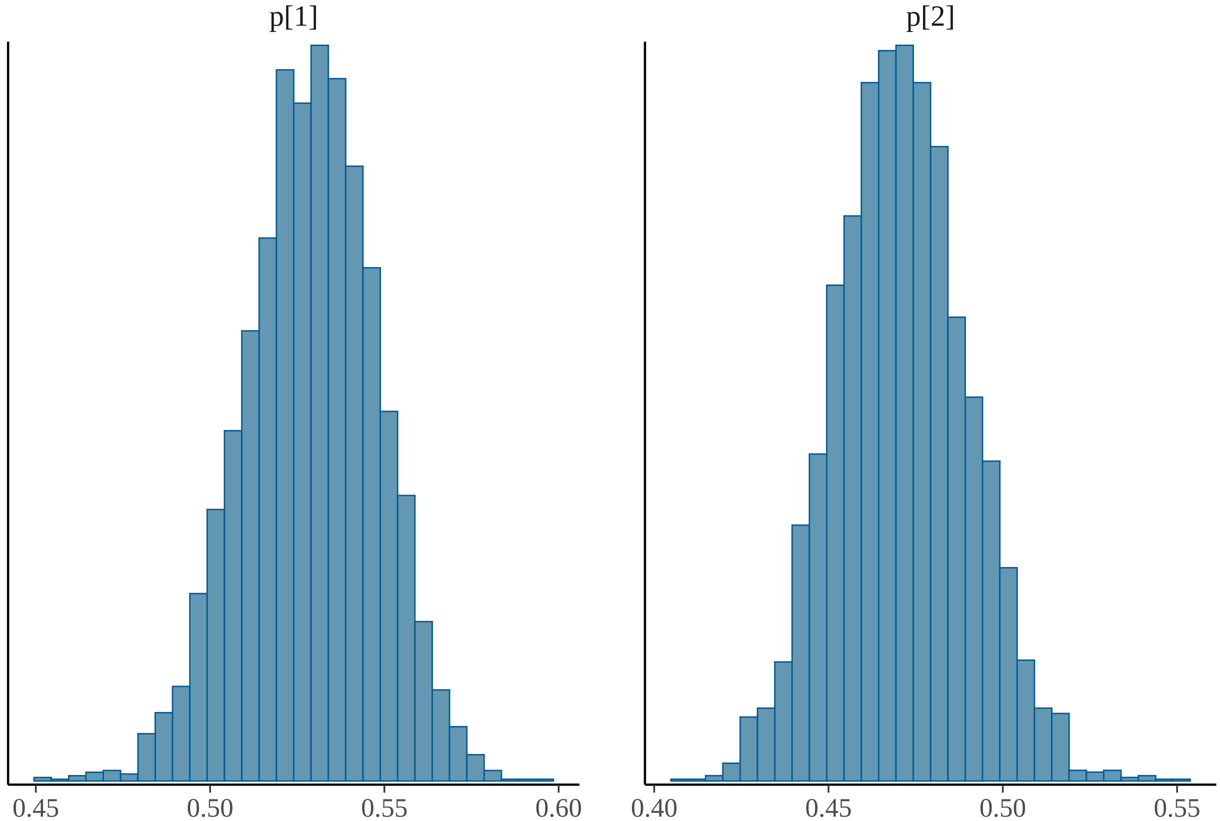

```r
mcmc_hist(fit0$draws(c("mu[1]", "mu[2]")))
```

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



```r
mcmc_hist(fit0$draws(c("p[1]", "p[2]")))
```

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



The first model, `mod0`, explores the data to determine the value for p which is the probability that the animal is in the high energy state which indicates that Z = 2. In this model, since there are two compartments, (1-p) corresponds to Z = 1 which states that the animal is in low energy state. In order to calculate p, I have sampled all of the data which includes 1219 observations. The priors on μ_1 and μ_2 are identical since there is no reason to assume that they would be different. In addition, σ_1 and σ_2 have identical priors. Both μ and σ are assumed to have normal distribution and with the identified priors the results for the μ_1 and μ_2, which are given above, seem close to the real data, hence, believe that the priors on these variables are sound and work for this model. According to this model p is 0.53. Thus, the animal is in the high energy state with 53% probability and in low energy state with 47%.

Task 2

```r
mod1 <- cmdstan_model("mod1.stan")
```

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

```r
print(mod1)
```

```
## data {
##   int<lower=0> K;
##   int<lower=0> N;
##   vector [N]y;
##   real<lower=0, upper=1>p;
## }
```

```
##
## parameters {
##   ordered[K] mu;
##   vector<lower=0>[K] sigma;
## }
##
## model {
##   mu ~ normal(0, 8);
##   sigma ~ normal(0, 3);
##
##   for (n in 1:N) {
##     target += log_mix(p,
##                       normal_lpdf(y[n]| mu[1], sigma[1]),
##                       normal_lpdf(y[n]| mu[2], sigma[2]));
##   }
## }
##
## generated quantities {
##   vector[N] log_lik;
##   for (n in 1:N) {
##     for (k in 1:K)
##       log_lik[n] = normal_lpdf(y[n]| mu[k], sigma[k]);
##   }
## }
```

```r
stan_dat <- list(K = 2, N = 1219, y = dat$log_msa, p = 0.53)
# with p = 0.53 incredibly low r-hats and ess's
fit1 <- mod1$sample(stan_dat, parallel_chains = 4, refresh = 0, seed = 666,
                    show_messages = F)
```

```
## Running MCMC with 4 parallel chains...
##
## Chain 3 finished in 5.3 seconds.
## Chain 1 finished in 5.5 seconds.
## Chain 4 finished in 5.5 seconds.
## Chain 2 finished in 6.5 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 5.7 seconds.
## Total execution time: 6.6 seconds.
```

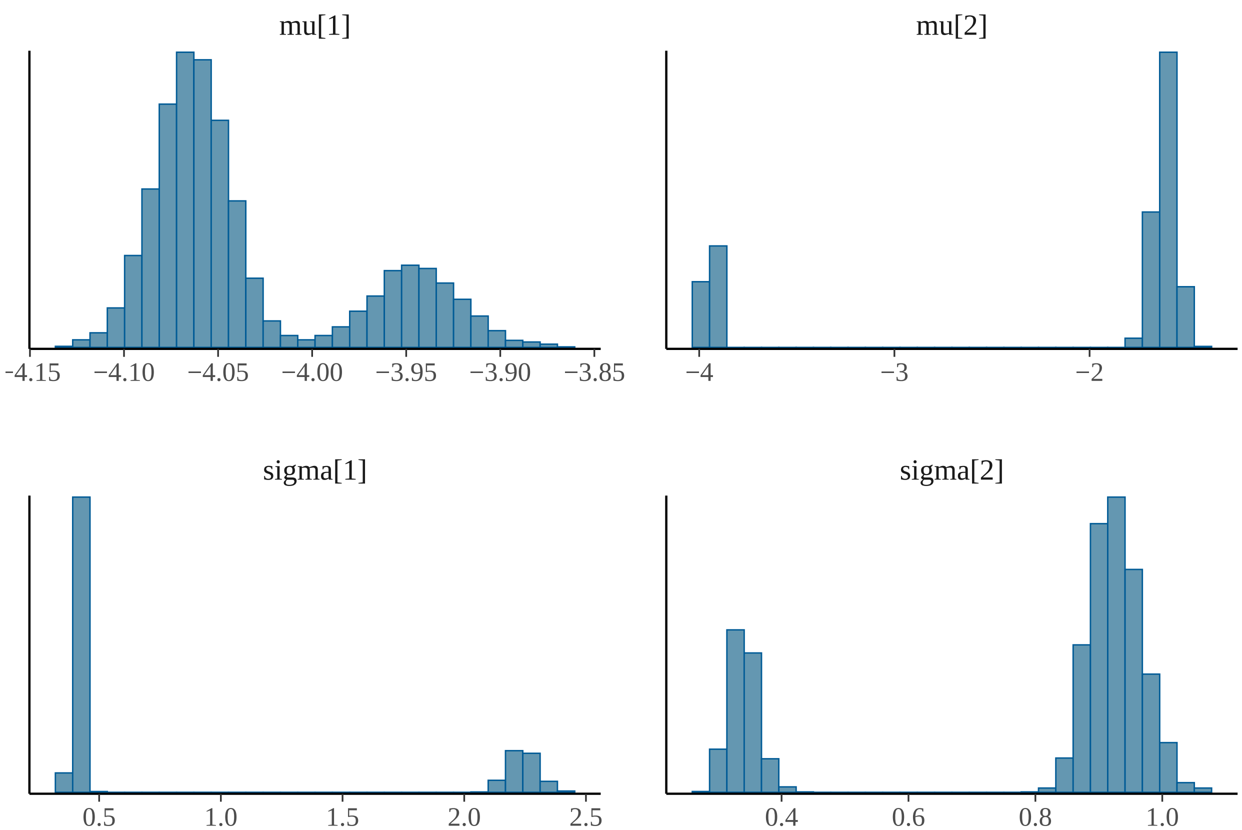
```r
print(fit1) # real bad
```

```
##    variable     mean   median     sd  mad       q5      q95 rhat ess_bulk
##  lp__       -1956.46 -1847.52 189.19 2.02 -2285.09 -1845.70 1.53        7
##  mu[1]         -4.03    -4.06   0.06 0.03    -4.09    -3.92 1.53        7
##  mu[2]         -2.19    -1.63   1.01 0.08    -3.96    -1.54 1.53        7
##  sigma[1]       0.87     0.42   0.79 0.02     0.39     2.29 1.53        7
##  sigma[2]       0.78     0.91   0.26 0.06     0.32     0.99 1.53        7
##  log_lik[1]    -7.48    -0.88  11.62 0.08   -30.47    -0.79 1.53        7
##  log_lik[2]    -3.51    -1.21   4.09 0.07   -11.71    -1.13 1.53        7
##  log_lik[3]    -8.31    -0.92  13.00 0.08   -34.01    -0.83 1.53        7
##  log_lik[4]    -8.61    -0.94  13.50 0.08   -35.29    -0.85 1.53        7
##  log_lik[5]   -13.99    -1.57  21.83 0.10   -57.19    -1.46 1.53        7
##  ess_tail
```

```
##        34
##        29
##        30
##        28
##        29
##        29
##        29
##        29
##        29
##        29
##
## # showing 10 of 1224 rows (change via 'max_rows' argument)
```

```r
mcmc_hist(fit1$draws(c("mu[1]", "mu[2]", "sigma[1]", "sigma[2]")))
```

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




```r
mod2 <- cmdstan_model("mod2.stan")
```

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

```r
print(mod2)
```

```
## data {
##   int<lower=1> K;
##   int<lower=1> N;
##   real wind[N];
##   real temp[N];
##   real y[N];
## }
```

```
##
## parameters {
##   ordered[K] mu;
##   vector<lower=0>[K] sigma;
##   real b0;
##   real b1;
##   real b2;
## }
##
## transformed parameters {
##   real p;
##   for (n in 1:N)
##     p = inv_logit(b0 + b1 * wind[n] + b2 * temp[n]);
## }
##
## model {
##   mu ~ normal(0, 8);
##   sigma ~ normal(0, 3);
##   b0 ~ normal(0,5);
##   b1 ~ normal(0,5);
##   b2 ~ normal(0,5);
##
##   for (n in 1:N) {
##     target += log_sum_exp(log(p) + normal_lpdf(y[n]| mu[1], sigma[1]),
##               log1m(p) + normal_lpdf(y[n]| mu[2], sigma[2]));
##   }
## }
```

```r
stan_dat <- list(K = 2, N = 1219, y = dat$log_msa, wind = dat$wind_speed,
                 temp = dat$saws_temp)
fit2 <- mod2$sample(stan_dat, parallel_chains = 4, refresh = 0, seed = 666,
                    show_messages = F, adapt_delta = 0.99)
```

```
## Running MCMC with 4 parallel chains...

## Chain 2 Rejecting initial value:

## Chain 2   Gradient evaluated at the initial value is not finite.

## Chain 2   Stan can't start sampling from this initial value.

## Chain 3 Rejecting initial value:

## Chain 3   Gradient evaluated at the initial value is not finite.

## Chain 3   Stan can't start sampling from this initial value.

## Chain 3 Rejecting initial value:

## Chain 3   Gradient evaluated at the initial value is not finite.

## Chain 3   Stan can't start sampling from this initial value.

## Chain 2 finished in 516.6 seconds.
## Chain 4 finished in 733.4 seconds.
## Chain 1 finished in 881.3 seconds.
## Chain 3 finished in 946.8 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 769.5 seconds.
```

```
## Total execution time: 7004.1 seconds.
##
## Warning: 214 of 4000 (5.0%) transitions ended with a divergence.
## This may indicate insufficient exploration of the posterior distribution.
## Possible remedies include:
##   * Increasing adapt_delta closer to 1 (default is 0.8)
##   * Reparameterizing the model (e.g. using a non-centered parameterization)
##   * Using informative or weakly informative prior distributions
## 2180 of 4000 (55.0%) transitions hit the maximum treedepth limit of 10 or 2^10-1 leapfrog steps.
## Trajectories that are prematurely terminated due to this limit will result in slow exploration.
## Increasing the max_treedepth limit can avoid this at the expense of more computation.
## If increasing max_treedepth does not remove warnings, try to reparameterize the model.
```

```r
print(fit2)
```

```
##  variable     mean   median    sd   mad       q5      q95 rhat ess_bulk
##  lp__     -1928.22 -1885.63 99.32 57.46 -2151.42 -1846.84 2.29        5
##  mu[1]       -4.52    -4.05  1.94  0.05    -7.99    -3.92 1.69       11
##  mu[2]       -1.99    -1.94  0.44  0.46    -2.93    -1.53 2.15        5
##  sigma[1]     0.72     0.45  0.86  0.10     0.39     2.35 1.96        6
##  sigma[2]     1.16     1.24  0.23  0.30     0.87     1.42 1.98        5
##  b0          -0.24     2.14  6.17  5.32   -12.19     5.84 1.51        7
##  b1          -3.04    -3.52  3.93  4.26    -7.90     4.12 1.34       13
##  b2           0.11     0.54  2.10  0.96    -4.50     1.84 1.47        8
##  p            0.43     0.49  0.17  0.07     0.00     0.55 2.01        5
##  ess_tail
##        15
##        17
##        13
##        15
##        16
##        19
##       104
##        15
##        15
```

```r
stan_dat <- list(K = 2, N = 1219, y = dat$log_msa, p = 0.38)
fit3 <- mod1$sample(stan_dat, parallel_chains = 4, refresh = 0, seed = 666,
                    show_messages = F)
```

```
## Running MCMC with 4 parallel chains...
##
## Chain 4 finished in 4.8 seconds.
## Chain 2 finished in 5.0 seconds.
## Chain 3 finished in 5.0 seconds.
## Chain 1 finished in 5.2 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 5.0 seconds.
## Total execution time: 5.4 seconds.
```

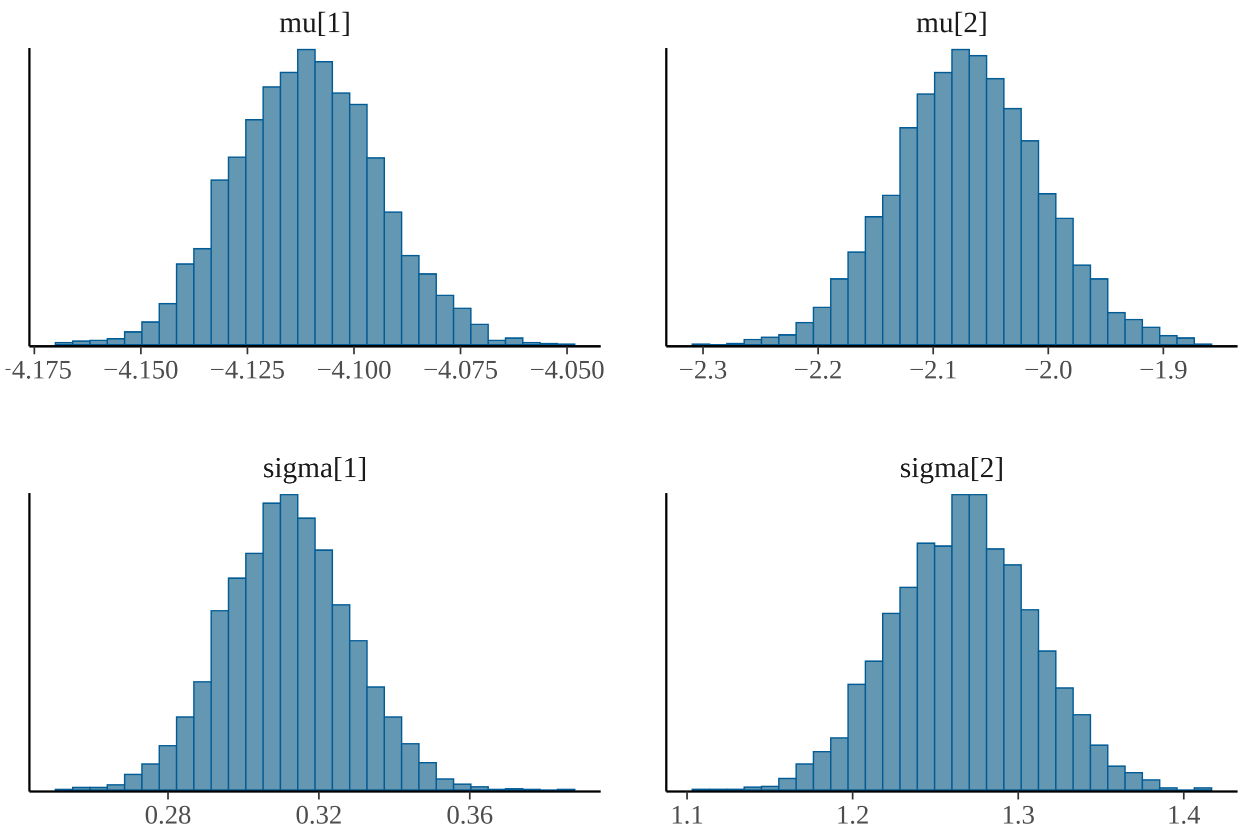
```r
print(fit3)
```

```
##    variable     mean   median   sd  mad       q5      q95 rhat ess_bulk
##  lp__       -1863.85 -1863.53 1.37 1.20 -1866.47 -1862.24 1.00     1985
```

```
##   mu[1]          -4.11     -4.11 0.02 0.02     -4.14     -4.08 1.00      3155
##   mu[2]          -2.07     -2.07 0.06 0.06     -2.17     -1.96 1.00      3344
##   sigma[1]        0.31      0.31 0.02 0.02      0.28      0.34 1.00      2949
##   sigma[2]        1.27      1.27 0.04 0.04      1.20      1.34 1.00      3019
##   log_lik[1]     -1.28     -1.28 0.04 0.04     -1.35     -1.21 1.00      2765
##   log_lik[2]     -1.19     -1.19 0.03 0.03     -1.23     -1.14 1.00      3742
##   log_lik[3]     -1.34     -1.34 0.04 0.04     -1.41     -1.27 1.00      2803
##   log_lik[4]     -1.37     -1.37 0.04 0.04     -1.44     -1.29 1.00      2825
##   log_lik[5]     -1.91     -1.91 0.06 0.06     -2.01     -1.82 1.00      3906
##   ess_tail
##       3038
##       2855
##       3285
##       2550
##       2916
##       2567
##       3047
##       2564
##       2462
##       3456
##
##   # showing 10 of 1224 rows (change via 'max_rows' argument)
```

```r
mcmc_hist((fit3$draws(c("mu[1]", "mu[2]", "sigma[1]", "sigma[2]"))))
```

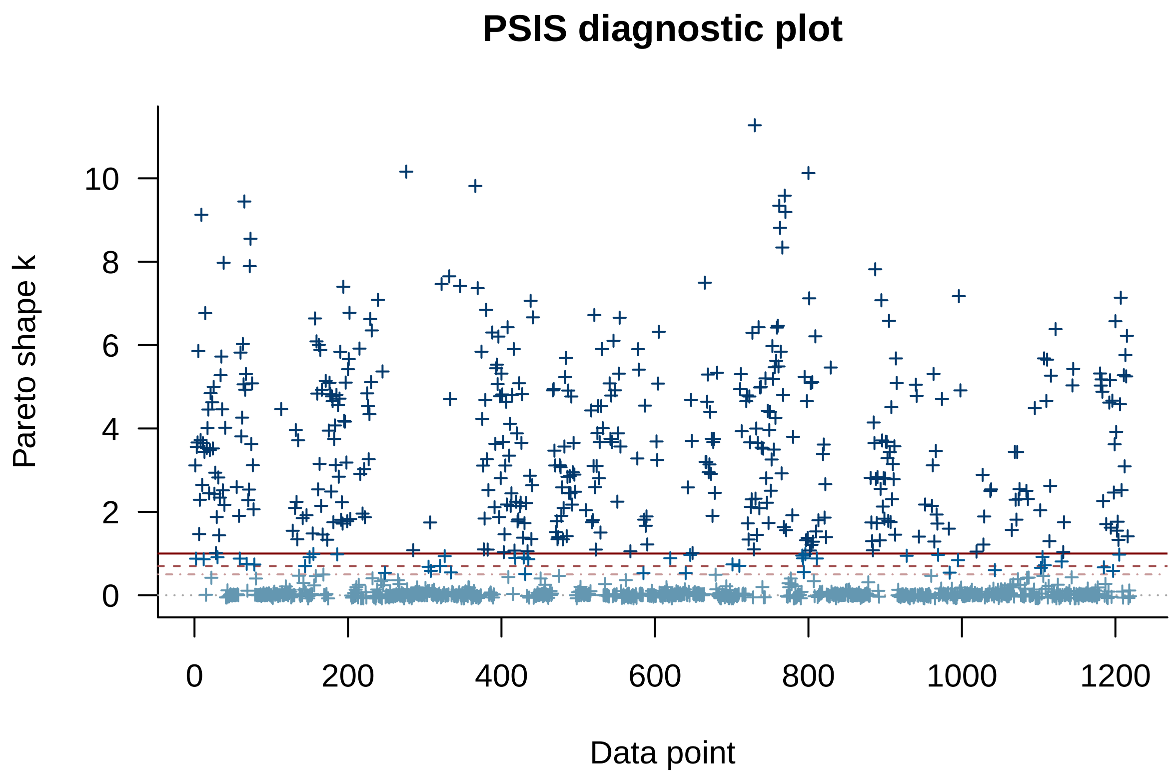
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```r
loo1 <- fit1$loo(save_psis = T)
print(loo1)
```

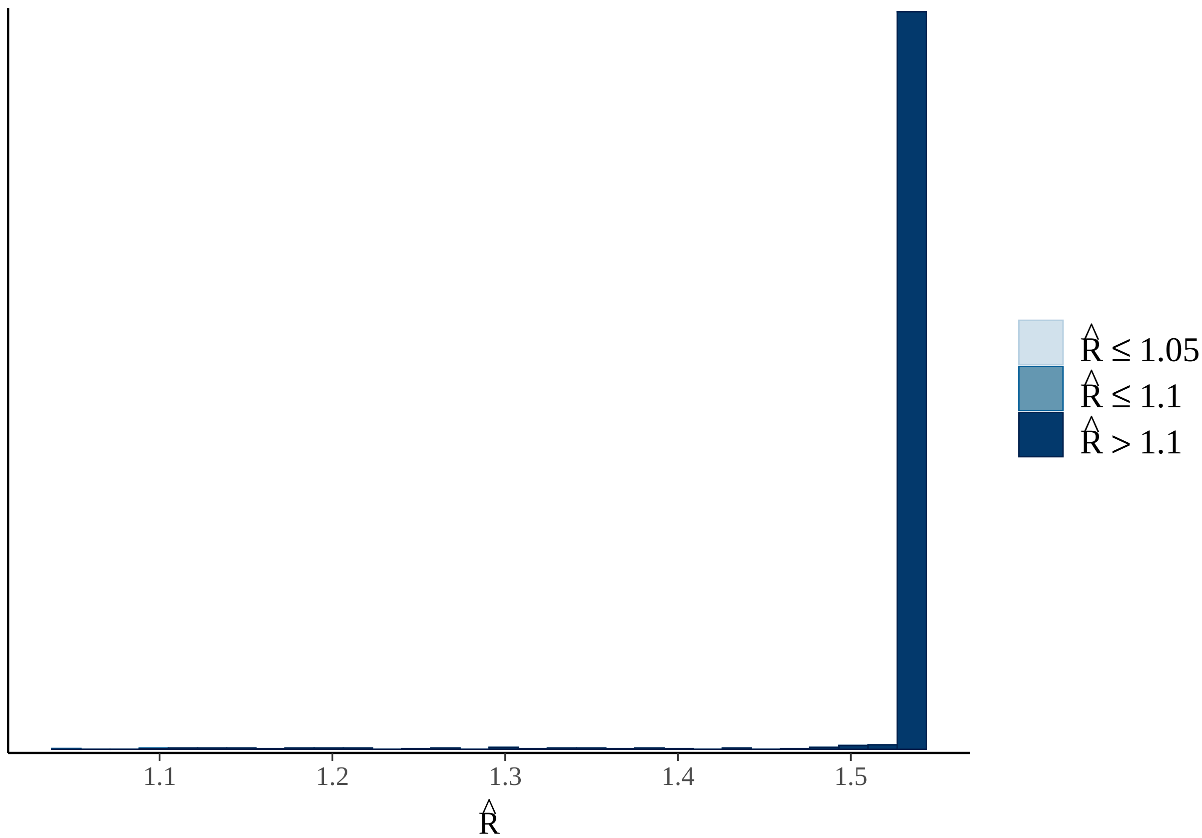
```
##
## Computed from 4000 by 1219 log-likelihood matrix
##
##          Estimate     SE
## elpd_loo -22469.8  823.3
## p_loo     20374.5  821.6
## looic     44939.6 1646.5
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     731   60.0%   0
##  (0.5, 0.7]   (ok)        13    1.1%   0
##    (0.7, 1]   (bad)       30    2.5%   0
##    (1, Inf)   (very bad) 445   36.5%   0
## See help('pareto-k-diagnostic') for details.
```

```r
plot(loo1)
```




```r
rhats <- rhat(fit1)
mcmc_rhat_hist(rhats)
```

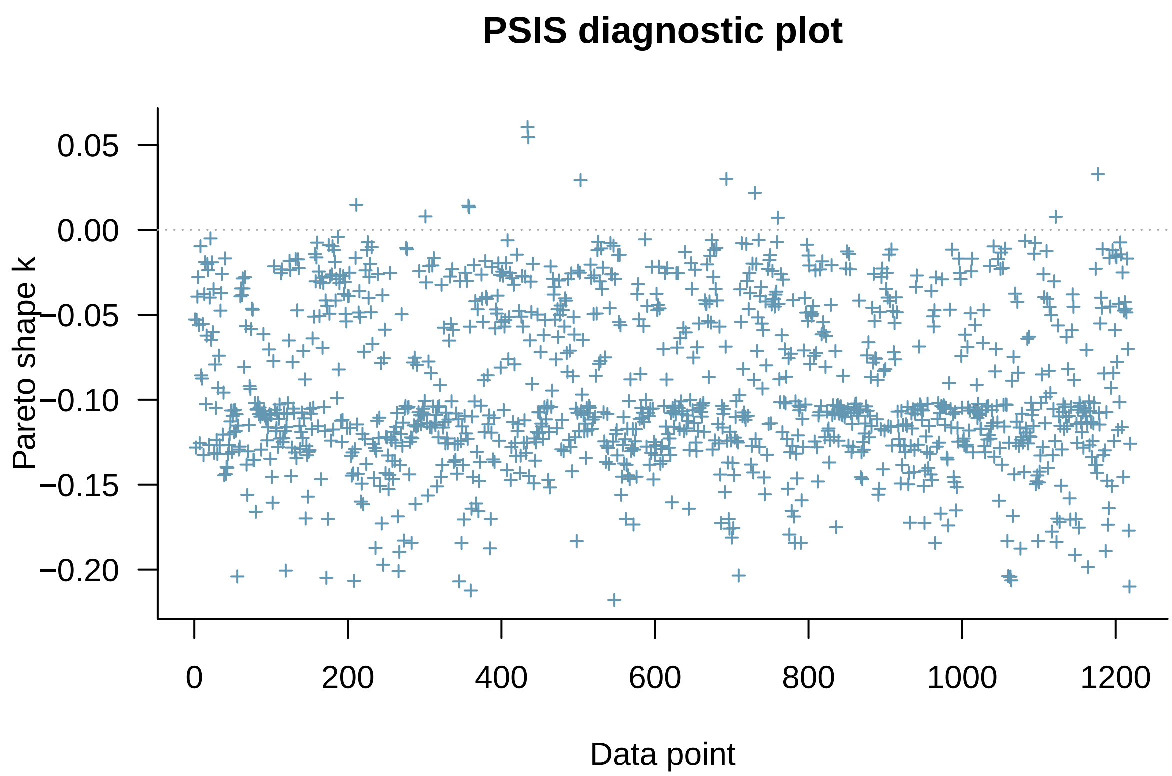
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```r
loo3 <- fit3$loo(save_psis = T)
print(loo3)
```

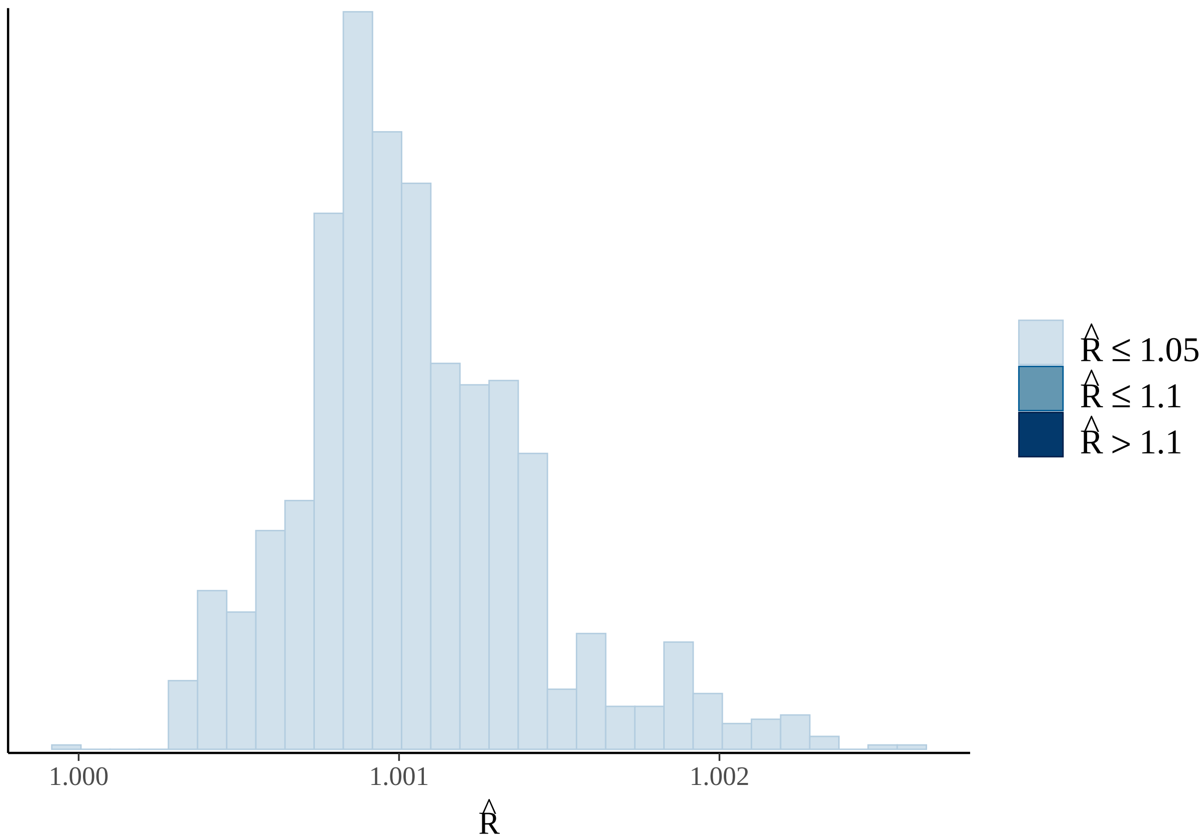
```
##
## Computed from 4000 by 1219 log-likelihood matrix
##
##           Estimate   SE
## elpd_loo  -2446.1 24.4
## p_loo        11.1  0.4
## looic      4892.3 48.7
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

```r
plot(loo3)
```




```r
rhats <- rhat(fit3)
mcmc_rhat_hist(rhats)
```

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




```r
loo_compare(loo1, loo3)
```

```
##        elpd_diff se_diff
## model2      0.0      0.0
## model1 -20023.6    827.5
```

For the second task I have created two models `mod1` and `mod2`. The first model for this task uses the fixed value for p which was calculated using the model in task 1, `mod0`. When we use this fixed value in the model we get a fit with really high r-hats and really low ESS values which is concerning. For this fit, `fit1`, I have also calculated the log likelihoods in order to be able to compare this model with the later one. The second model in this task estimates the probability of the animal being in high energy state by a logistic regression using the variables given in the data set. Variables that are used to estimate p are wind speed and temperature. Using this model, `mod2`, we have the second fit, also in this fit we observe even higher values for r-hat and lower values for ESS. However, estimating p using a logistic regression results in much different value for p than 0.53, which was the value calculated in the first task. Thus, I wanted to use the first model, `mod1`, which uses a fixed value for p and I have implemented the new estimate for p which was 0.38 in that model to create the third fit, `fit3`. Using p as a parameter in the Stan model have resulted in deficits in r-hats and ESS values for me so I wanted to use the fixed value model and create a new fit to compare the previously calculated value for p, 0.53, with the more recent estimation, 0.38. We observe that in `fit3` the r-hats are back on 1 and ESS values (for bot bulk and tail) are well above 500. I have calculated log likelihoods for this fit also so we can compare both fits using leave-one-out cross validation. Just by looking at the MCMC histograms and the summary for both fits, it is obvious that the `fit3` is much better. 36.5% of the Pareto k values in `fit1` are considered as very bad, whereas values corresponding to `fit3` are below 0.5. When we compare these two models we see that the later one is doing much better with a really big eldp difference between the two, -20023.6.

```r
mod4 <- cmdstan_model("mod4.stan")
```

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

```r
stan_dat <- list(K = 3, N = 1219, y = dat$log_msa)
fit4 <- mod4$sample(stan_dat, parallel_chains = 4)
```

```
## Running MCMC with 4 parallel chains...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)

## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected because of the

## Chain 1 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/var/folders/0c/1f38

## Chain 1 If this warning occurs sporadically, such as for highly constrained variable types like covar

## Chain 1 but if this warning occurs often then your model may be either severely ill-conditioned or mi

## Chain 1

## Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)

## Chain 2 Informational Message: The current Metropolis proposal is about to be rejected because of the

## Chain 2 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/var/folders/0c/1f38

## Chain 2 If this warning occurs sporadically, such as for highly constrained variable types like covar

## Chain 2 but if this warning occurs often then your model may be either severely ill-conditioned or mi

## Chain 2

## Chain 3 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4 Iteration:    1 / 2000 [  0%]  (Warmup)

## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected because of the

## Chain 4 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/var/folders/0c/1f38

## Chain 4 If this warning occurs sporadically, such as for highly constrained variable types like covar

## Chain 4 but if this warning occurs often then your model may be either severely ill-conditioned or mi

## Chain 4

## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected because of the

## Chain 4 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/var/folders/0c/1f38

## Chain 4 If this warning occurs sporadically, such as for highly constrained variable types like covar

## Chain 4 but if this warning occurs often then your model may be either severely ill-conditioned or mi

## Chain 4

## Chain 2 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 3 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 1 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 4 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 3 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 1 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 2 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 4 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 1 Iteration:  400 / 2000 [ 20%]  (Warmup)
```

```
## Chain 3 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 3 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 3 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 4 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 2 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 4 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 1 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 3 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 2 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 4 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 3 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 4 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 3 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 4 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 3 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 4 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 3 Iteration: 1600 / 2000 [ 80%]  (Sampling)
```

```
## Chain 4 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 1 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 4 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 4 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1 finished in 23.4 seconds.
## Chain 4 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 4 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4 finished in 23.8 seconds.
## Chain 2 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 2 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 3 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3 finished in 24.9 seconds.
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 finished in 25.6 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 24.4 seconds.
## Total execution time: 25.8 seconds.
```

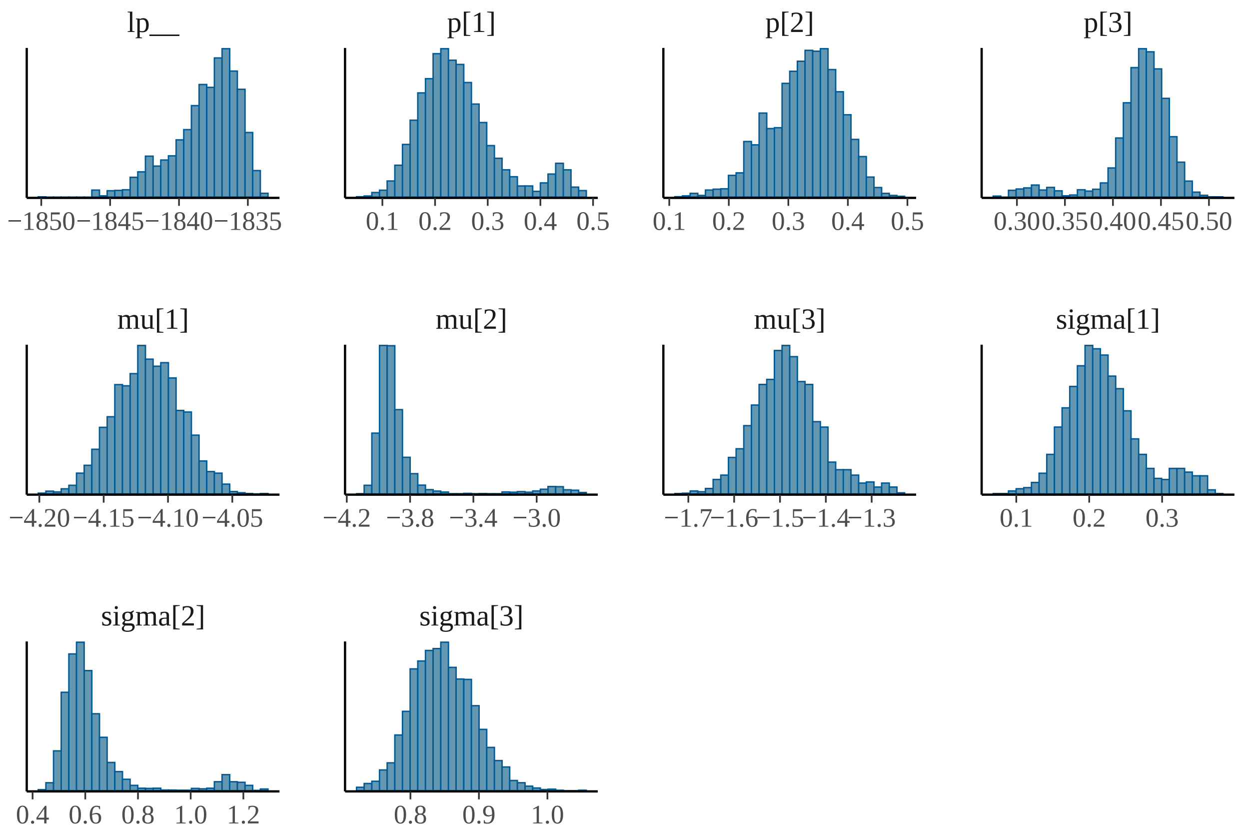
```
##
## Warning: 111 of 4000 (3.0%) transitions ended with a divergence.
## This may indicate insufficient exploration of the posterior distribution.
## Possible remedies include:
##   * Increasing adapt_delta closer to 1 (default is 0.8)
##   * Reparameterizing the model (e.g. using a non-centered parameterization)
##   * Using informative or weakly informative prior distributions
```

```r
print(fit4)
```

```
##  variable     mean   median   sd  mad       q5      q95 rhat ess_bulk ess_tail
##  lp__     -1838.03 -1837.53 2.38 2.16 -1842.69 -1834.96 1.08       37       42
##  p[1]         0.24     0.23 0.08 0.06     0.14     0.43 1.09       28       11
##  p[2]         0.33     0.33 0.06 0.06     0.23     0.42 1.06       50      266
##  p[3]         0.43     0.43 0.03 0.02     0.34     0.47 1.09       30       11
##  mu[1]       -4.12    -4.12 0.03 0.03    -4.16    -4.07 1.05       70      409
##  mu[2]       -3.85    -3.93 0.27 0.07    -4.02    -2.98 1.09       30       11
##  mu[3]       -1.48    -1.49 0.07 0.07    -1.60    -1.35 1.07       39       19
##  sigma[1]     0.22     0.21 0.05 0.04     0.15     0.33 1.10       27       12
##  sigma[2]     0.63     0.59 0.16 0.07     0.50     1.12 1.09       28       10
##  sigma[3]     0.85     0.85 0.05 0.05     0.78     0.93 1.01     1404     2280
```

```r
mcmc_hist((fit4$draws()))
```

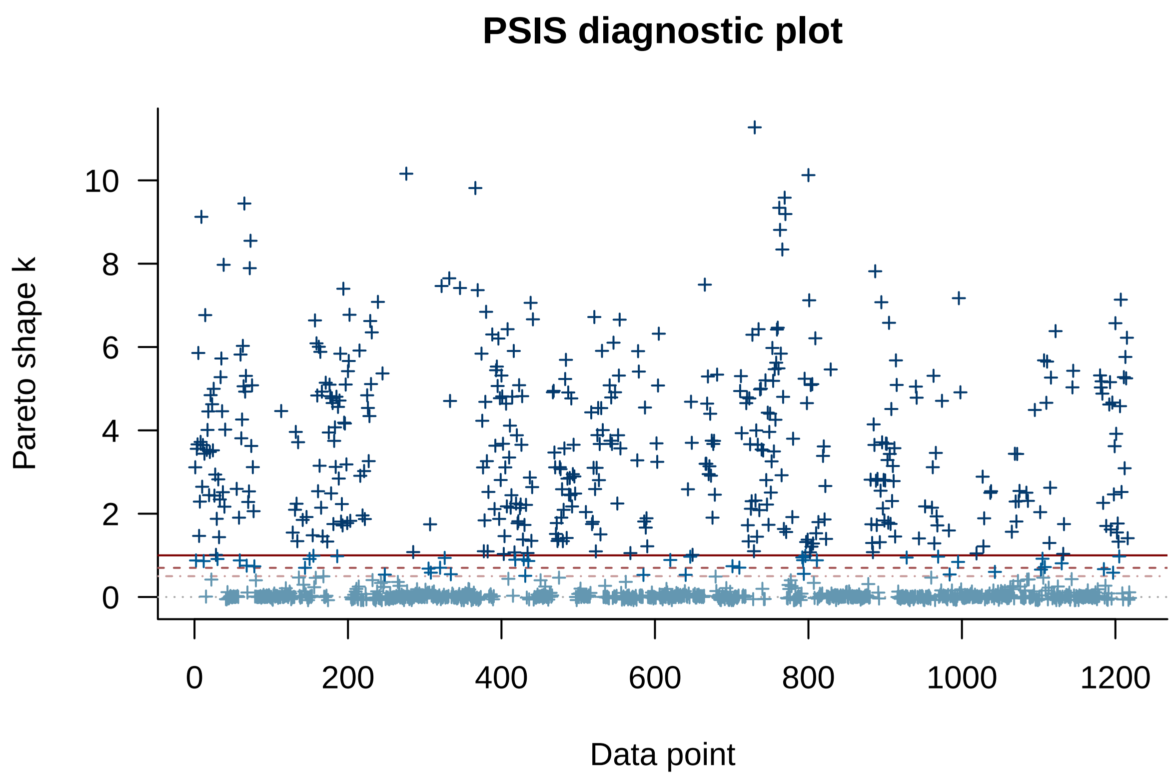
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```r
loo4 <- fit1$loo(save_psis = T)
print(loo1)
```

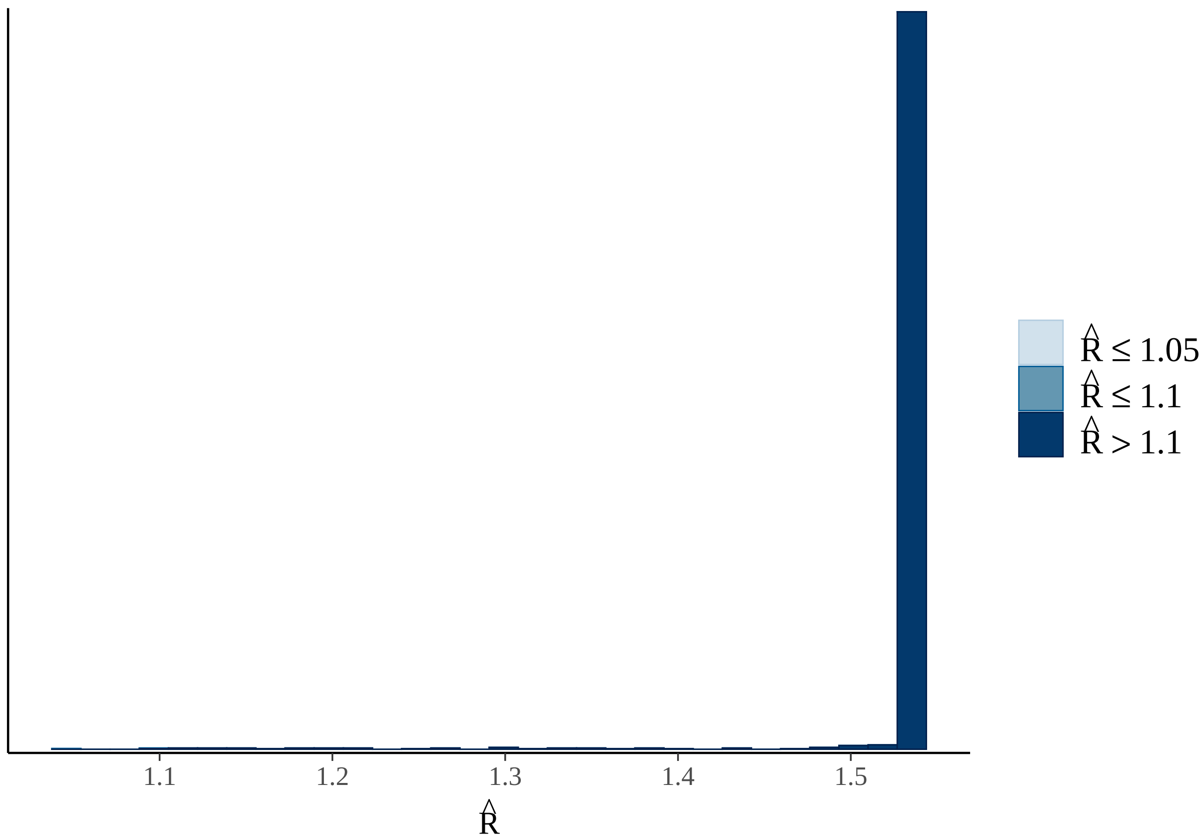
```
##
## Computed from 4000 by 1219 log-likelihood matrix
##
##          Estimate     SE
## elpd_loo -22469.8  823.3
## p_loo     20374.5  821.6
## looic     44939.6 1646.5
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     731   60.0%   0
##  (0.5, 0.7]   (ok)        13    1.1%   0
##    (0.7, 1]   (bad)       30    2.5%   0
##    (1, Inf)   (very bad) 445   36.5%   0
## See help('pareto-k-diagnostic') for details.
```

```r
plot(loo1)
```




```r
rhats <- rhat(fit1)
mcmc_rhat_hist(rhats)
```

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


1.1
1.2
1.3
1.4
1.5
$\hat{R}$
$\hat{R} \leq 1.05$
$\hat{R} \leq 1.1$
$\hat{R} > 1.1$