STA365 HW1

Ilke Sun

06/02/2021

I Data and Variables Setup

Firstly, we will read our data into R.

```
data <- readRDS("hw1_data.RDS")
t <- data$t
y <- data$y
```

II Visualization

The graphs and summaries below helped me decide on my priors.

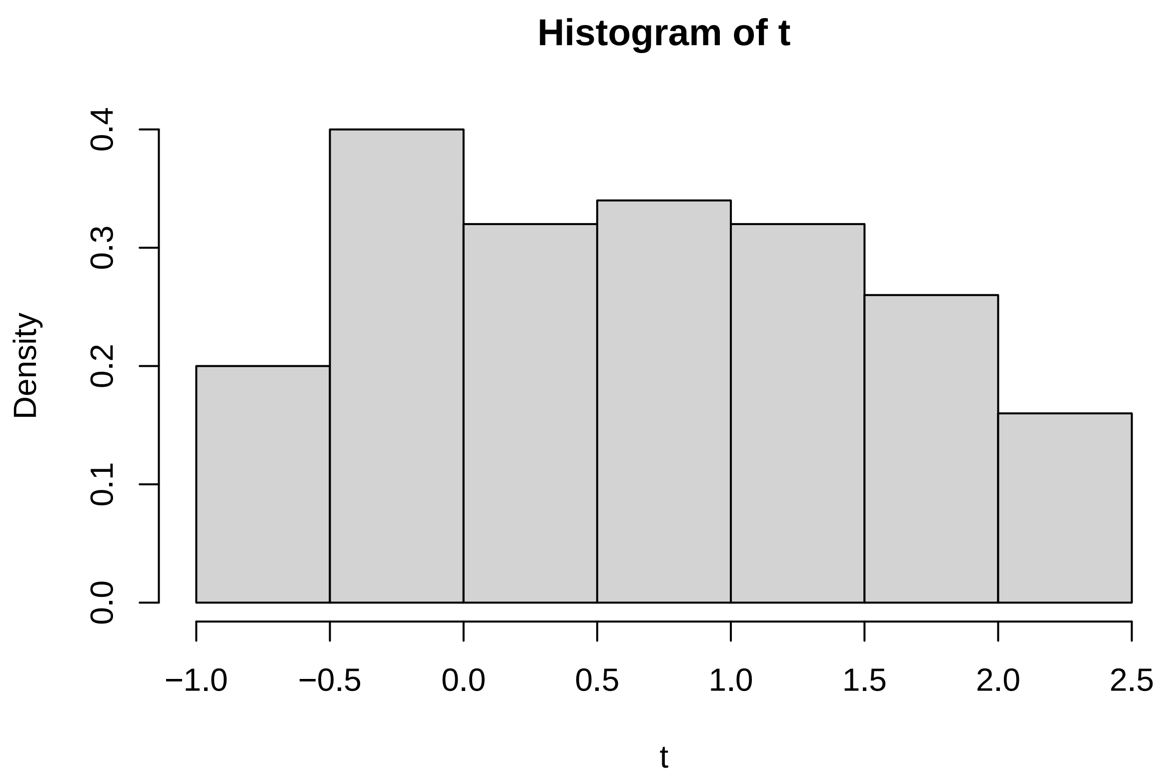
```
summary(t)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
## -0.9903 -0.1267  0.6052  0.6415  1.2940  2.4965
```

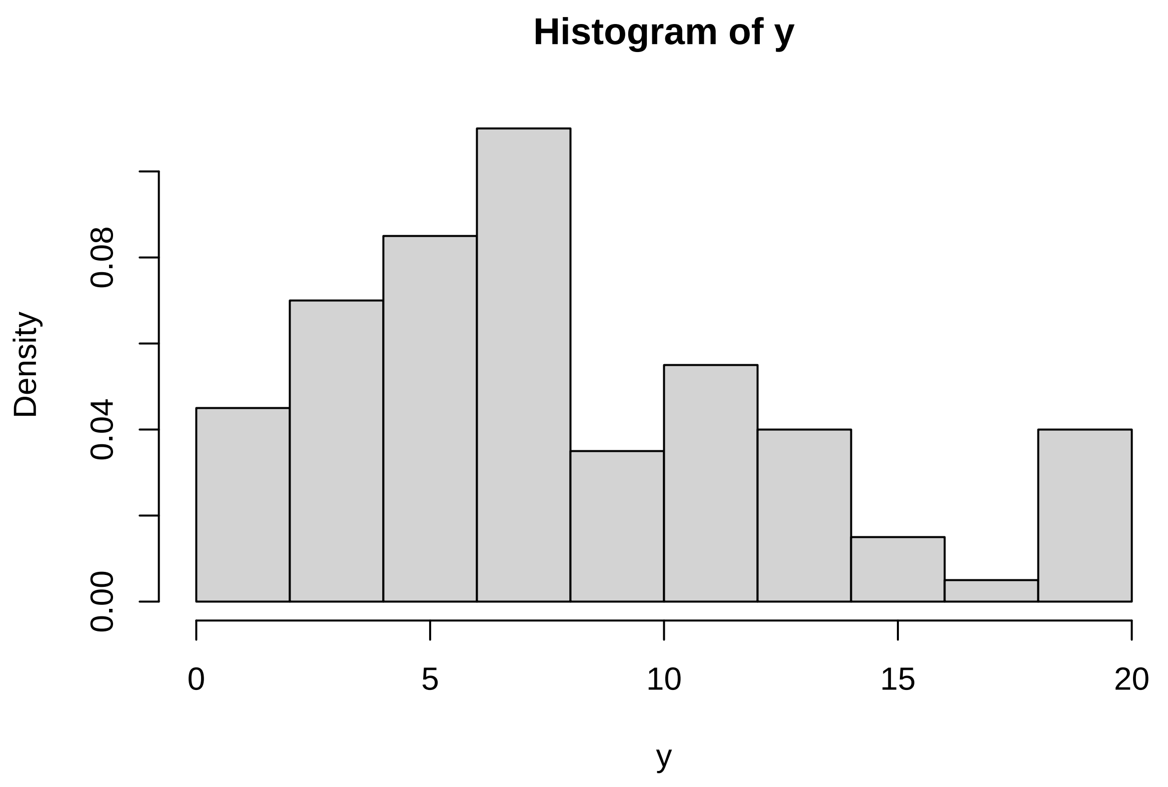
```
summary(y)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  0.4893  4.1870  6.9887  7.8625 10.8495 19.6650
```

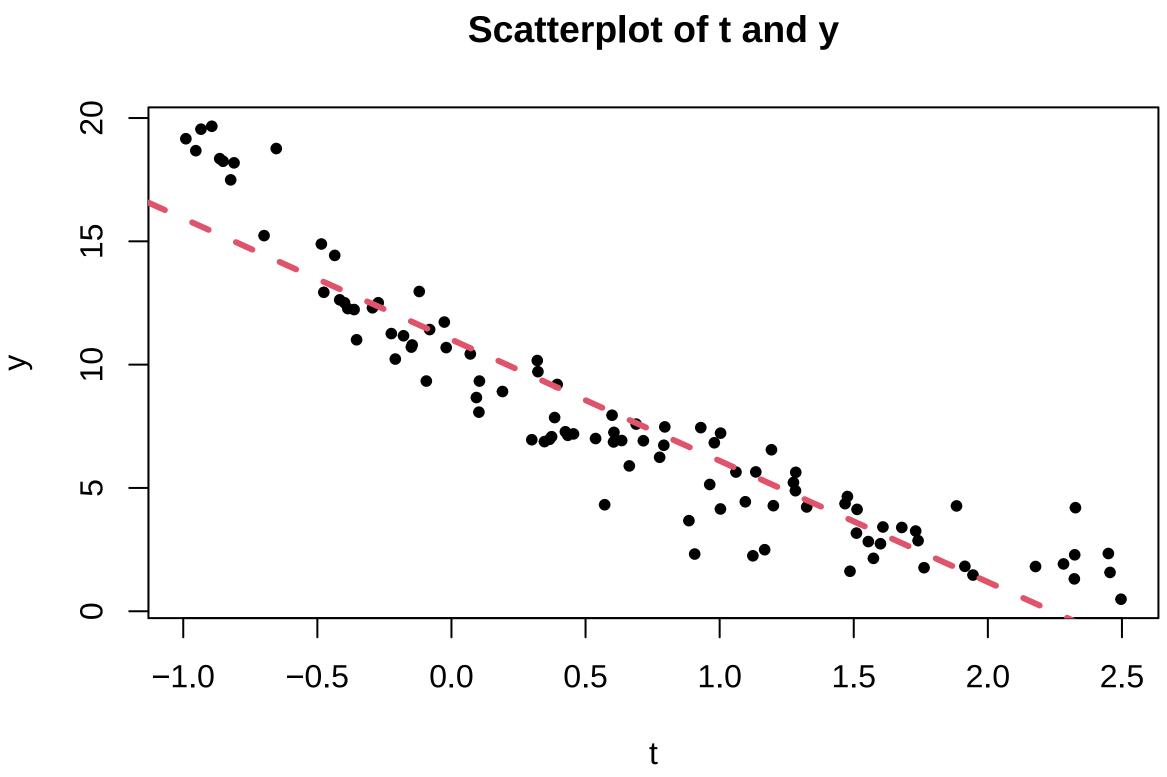
```
lm0 <- lm(y ~ t)

hist(t,
     freq = F,
     xlab = "t",
     main = "Histogram of t")
```



```
hist(y,
     freq = F,
     xlab = "y",
     main = "Histogram of y")
```



```r
plot(y ~ t,
     pch = 20,
     main = "Scatterplot of t and y",
     xlab = "t",
     ylab = "y")
abline(lm0, col = 2, lty = 2, lw = 3)
```



III Setting Priors

For the first model, I have chosen a normal prior distribution. The reason behind this intuition was the normality behind normality that is when normal likelihood function is a continuous distribution then normal distribution with a know variance σ has a normal conjugate prior. Also, as we can see from the plot above, where the red-dashed-line is the line fitted with OLS method that the linear regression fits not too badly. After looking at the regression results summary we can see that the both intercept and slope coefficient is statistically significant. From the plot above, it is also observable that towards the extreme values the error tends to be higher. In order to account for this I have enlarged τ_σ where $\sigma \sim (0, \tau_\sigma)$, I believed that higher variance for the normally distributed sigma may help account for the greater error towards the extreme values. In addition, to these we also know that t_i will always be between [-1, 2.5], hence in my Stan code I have limited the value to be in that range. I have also bounded β in my Stan code because we expect the relationship between the covariate and the observation to be negative. Like σ, α and β are also normally distributed with mean 0 and variances τ_α and τ_β respectively. I have decided on these τ values by using the regression summary and some other observations. Firstly, I thought that τ_α and τ_β will be close to standard errors that correspond to the coefficients in the linear regression. I have finalized setting these parameters by doing MCMC checks for μ_i^* and y_i^* and comparing them to what we know such as values above the value 50 is very unlikely. For the second model, I have chosen a lognormal prior distribution. As we observe from the likelihood in the instructions now $\mu = \alpha + \beta t_i$ and the likelihood is $\log(y_i) \sim N(\alpha + \beta t_i, \sigma^2)$. I have, again, used the information that was given in the instructions (i.e., t always between [-1,2.5] and we expect a negative relationship). Also, in order to keep the model in natural scale so that we can compare it with Model 1, I have modified μ in my Model 1 Stan code to $\log(\mu)$. Even though now our prior is lognormal, the conjugate prior for the lognormal is the same for the normal distribution. Hence, I have α, β and σ normally distributed as well. Like the previous model I have decided on τ values by looking at the data and the plots on Part 1. For the precise values, I have looked at the MCMC checks for $\log(\mu_i^*)$ and y_i^*.

IV Modelling

Model 1

```
writeLines(readLines("mod1.stan"))
```

```
## data {
##   int<lower=0> N;
##   vector[N] y;
##   vector<lower=-1, upper=2.5>[N] t;
##
##   //priors
##   real<lower=0> tau_a;
##   real<lower=0> tau_b;
##   real<lower=0> tau_s;
##
##   int<lower=0, upper=1> only_prior;
## }
##
## parameters {
##   real a;
##   real<upper=0> b;
##   real<lower=0> sigma;
## }
##
## transformed parameters {
##   vector[N] mu = exp(a + b*t);
## }
##
## model {
##   //priors
##   a ~ normal(0, tau_a);
##   b ~ normal(0, tau_b);
##   sigma ~ normal(0, tau_s);
##
##   if(only_prior == 0) {
##     y ~ normal(mu, sigma);
##   }
## }
##
## generated quantities {
##   vector[N] log_lik;
##   vector[N] y_pred;
##   for (i in 1:N) {
##     log_lik[i] = normal_lpdf(y[i]| mu[i], sigma);
##     y_pred[i] = normal_rng(mu[i], sigma);
##   }
## }
```

```
mod1 <- cmdstan_model("mod1.stan", compile = T)
```

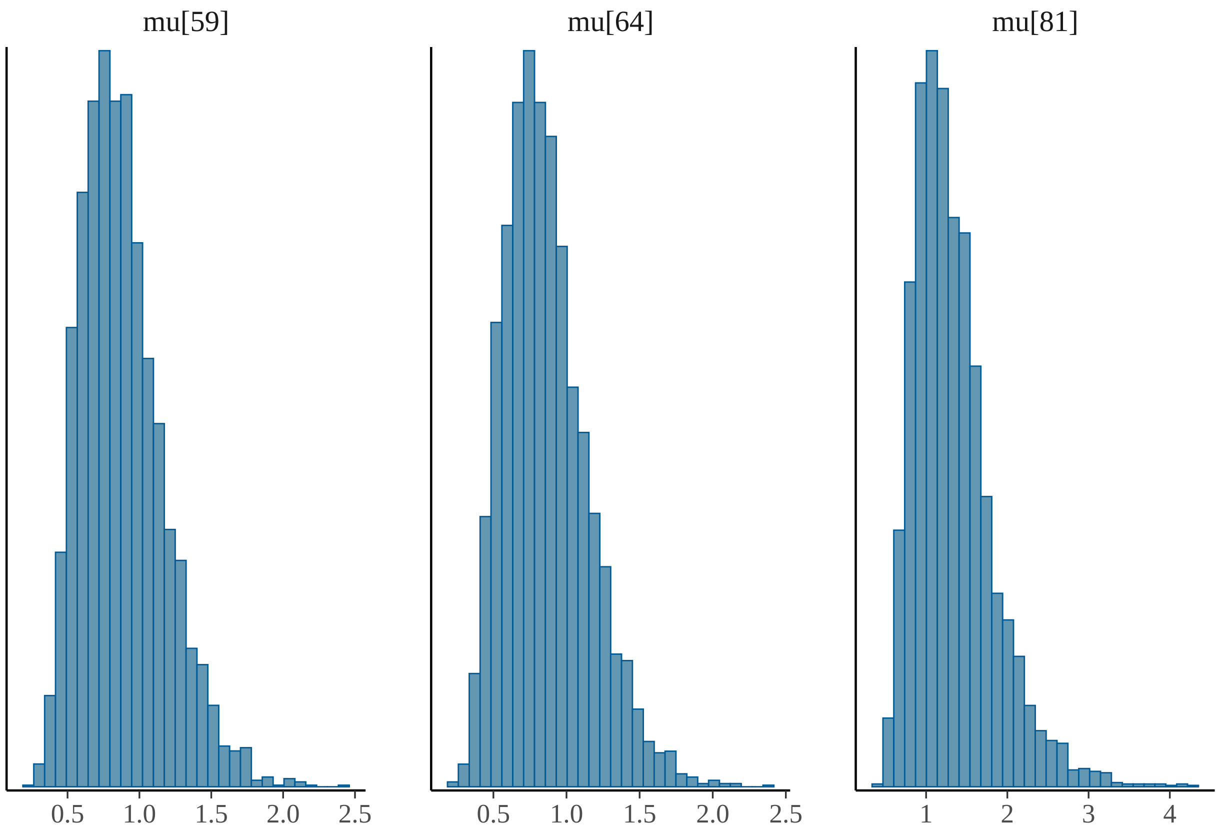
```
## Model executable is up to date!
```

```
data_list <- list(N = 100, y = y, t = t, tau_a = 0.3,
                  tau_b = 0.6, tau_s = 2, only_prior = 1)

fit1 <- mod1$sample(data_list, set.seed(333), refresh = 0, show_messages = T)
```

```
## Running MCMC with 4 sequential chains...
##
## Chain 1 finished in 0.2 seconds.
## Chain 2 finished in 0.3 seconds.
## Chain 3 finished in 0.3 seconds.
## Chain 4 finished in 0.3 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.3 seconds.
## Total execution time: 1.4 seconds.
```

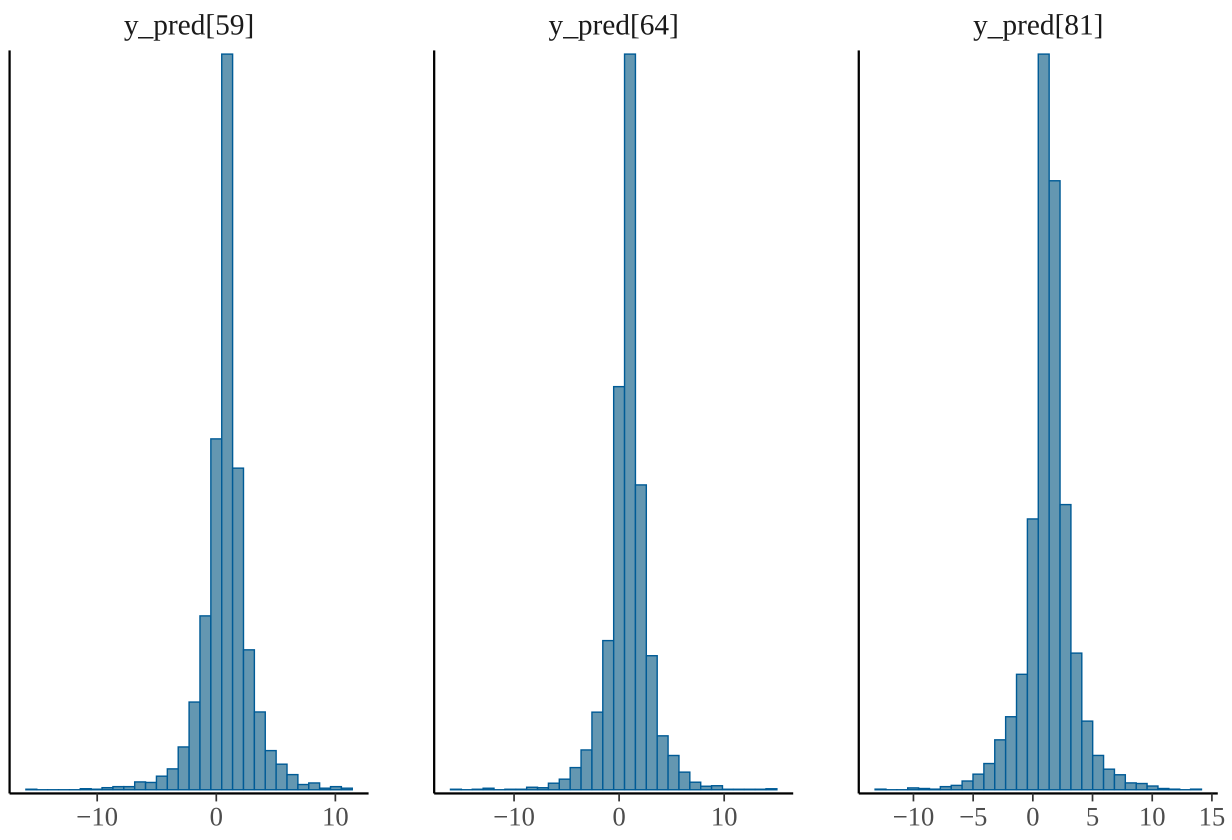
```
mcmc_hist(fit1$draws(c("mu[59]","mu[64]","mu[81]")))
```

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



```
mcmc_hist(fit1$draws(c("y_pred[59]","y_pred[64]","y_pred[81]")))
```

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



```
fit1$print(max_rows = 20)
```

```
##  variable  mean median   sd  mad    q5   q95 rhat ess_bulk ess_tail
##    lp__   -2.63  -2.28 1.38 1.19 -5.37 -1.06 1.00     1326     1614
##    a       0.00  -0.01 0.30 0.30 -0.49  0.49 1.00     2649     2582
##    b      -0.47  -0.39 0.37 0.35 -1.16 -0.03 1.00     1658     1179
##    sigma   1.60   1.35 1.22 1.18  0.11  3.94 1.00     1582     1239
##    mu[1]   0.85   0.81 0.29 0.27  0.46  1.38 1.00     2848     2892
##    mu[2]   1.27   1.19 0.44 0.40  0.70  2.11 1.00     2384     2265
##    mu[3]   1.75   1.50 0.95 0.65  0.79  3.58 1.00     2197     2354
##    mu[4]   0.58   0.54 0.31 0.32  0.16  1.14 1.00     2489     2683
##    mu[5]   1.66   1.45 0.83 0.60  0.78  3.28 1.00     2227     2376
##    mu[6]   0.58   0.55 0.31 0.32  0.16  1.15 1.00     2506     2683
##    mu[7]   0.80   0.76 0.29 0.27  0.40  1.33 1.00     2858     2881
##    mu[8]   0.76   0.73 0.29 0.28  0.36  1.30 1.00     2840     2818
##    mu[9]   0.58   0.55 0.31 0.32  0.16  1.15 1.00     2502     2683
##    mu[10]  0.89   0.85 0.29 0.28  0.50  1.43 1.00     2823     2576
##    mu[11]  1.26   1.19 0.44 0.40  0.70  2.09 1.00     2389     2294
##    mu[12]  1.62   1.43 0.79 0.58  0.77  3.16 1.00     2241     2479
##    mu[13]  1.12   1.07 0.35 0.32  0.65  1.77 1.00     2530     2488
##    mu[14]  1.14   1.08 0.36 0.33  0.66  1.80 1.00     2507     2416
##    mu[15]  0.57   0.53 0.31 0.32  0.15  1.13 1.00     2464     2723
##    mu[16]  0.71   0.68 0.30 0.29  0.30  1.26 1.00     2780     2763
## 
##  # showing 20 of 304 rows (change via 'max_rows' argument)
```

```
data_list$only_prior = 0
```

```r
fit2 <- mod1$sample(data_list, set.seed(333), refresh = 0, show_messages = F)
```

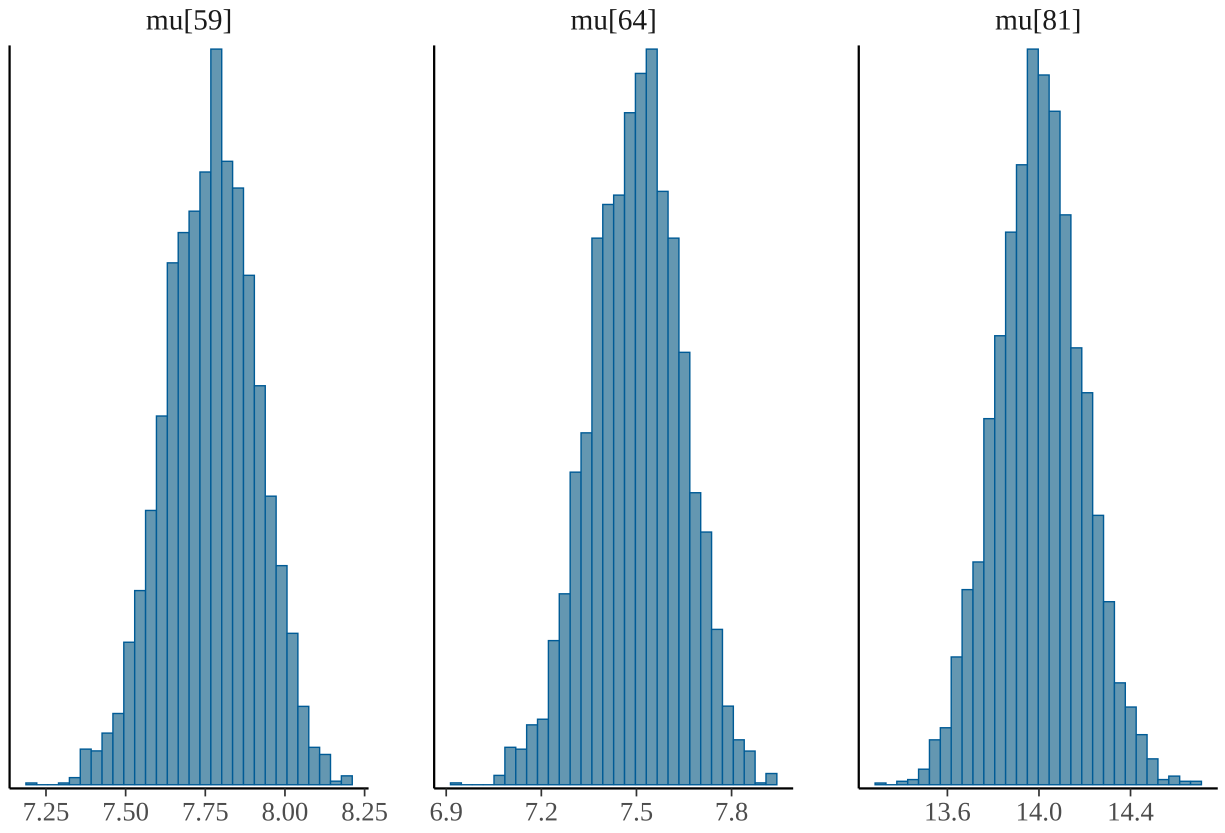
```
## Running MCMC with 4 sequential chains...
##
## Chain 1 finished in 0.8 seconds.
## Chain 2 finished in 0.5 seconds.
## Chain 3 finished in 0.6 seconds.
## Chain 4 finished in 0.4 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.6 seconds.
## Total execution time: 2.5 seconds.
```

```r
fit2$summary()
```

```
## # A tibble: 304 x 10
##    variable    mean  median     sd    mad      q5     q95  rhat ess_bulk
##    <chr>      <dbl>   <dbl>  <dbl>  <dbl>   <dbl>   <dbl> <dbl>    <dbl>
##  1 lp__     -95.9   -95.6   1.22   1.01   -98.3   -94.5    1.00    1893.
##  2 a          2.31    2.31  0.0140 0.0143   2.28    2.33   1.00    2878.
##  3 b         -0.686  -0.686 0.0200 0.0202  -0.719  -0.654  1.00    2768.
##  4 sigma      1.17    1.17  0.0853 0.0849   1.04    1.32   1.00    2812.
##  5 mu[1]      7.34    7.35  0.143  0.147    7.11    7.58   1.00    2398.
##  6 mu[2]     13.2    13.2   0.171  0.167   12.9    13.5    1.00    4586.
##  7 mu[3]     19.3    19.3   0.358  0.359   18.7    19.9    1.00    4949.
##  8 mu[4]      3.56    3.56  0.135  0.134    3.34    3.78   1.00    2411.
##  9 mu[5]     18.2    18.2   0.314  0.310   17.6    18.7    1.00    5119.
## 10 mu[6]      3.67    3.67  0.136  0.136    3.44    3.89   1.00    2408.
## # ... with 294 more rows, and 1 more variable: ess_tail <dbl>
```

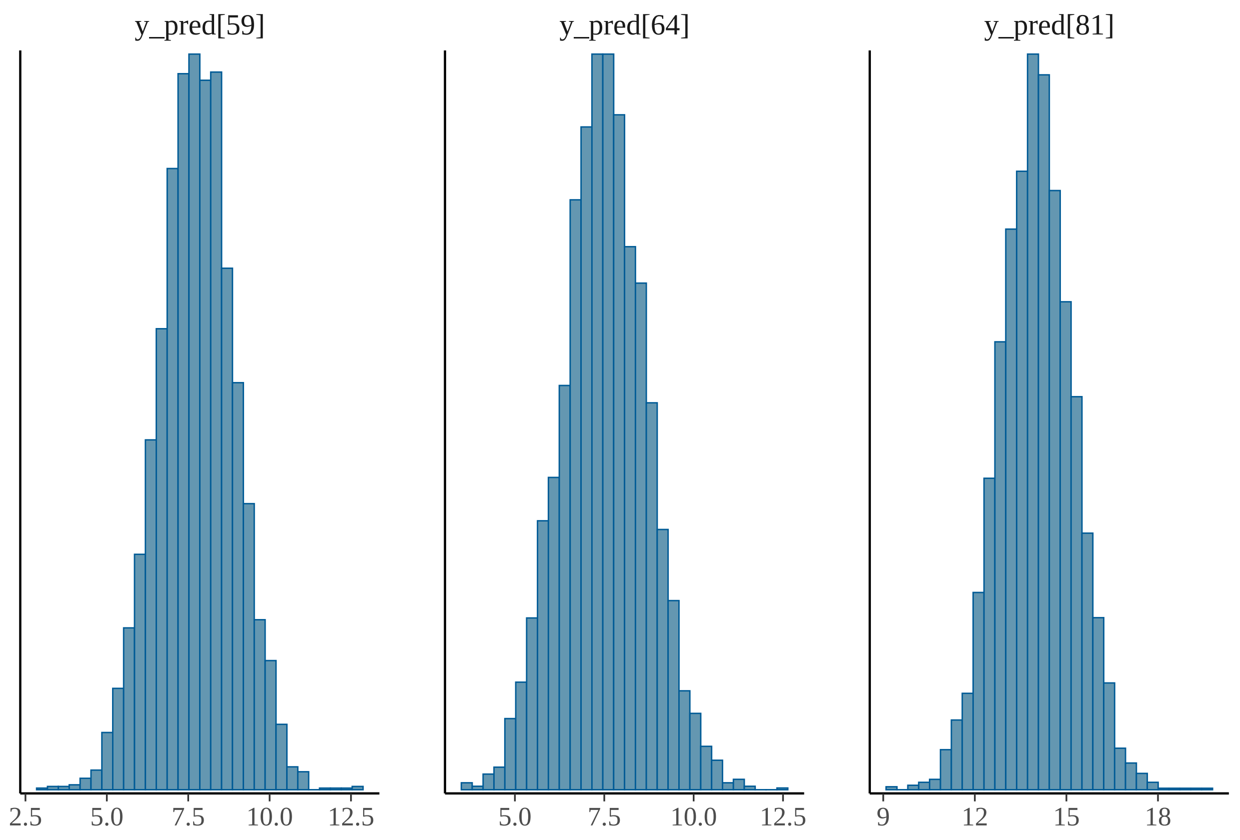
```r
mcmc_hist(fit2$draws(c("mu[59]","mu[64]","mu[81]")))
```

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



```
mcmc_hist(fit2$draws(c("y_pred[59]","y_pred[64]","y_pred[81]")))
```

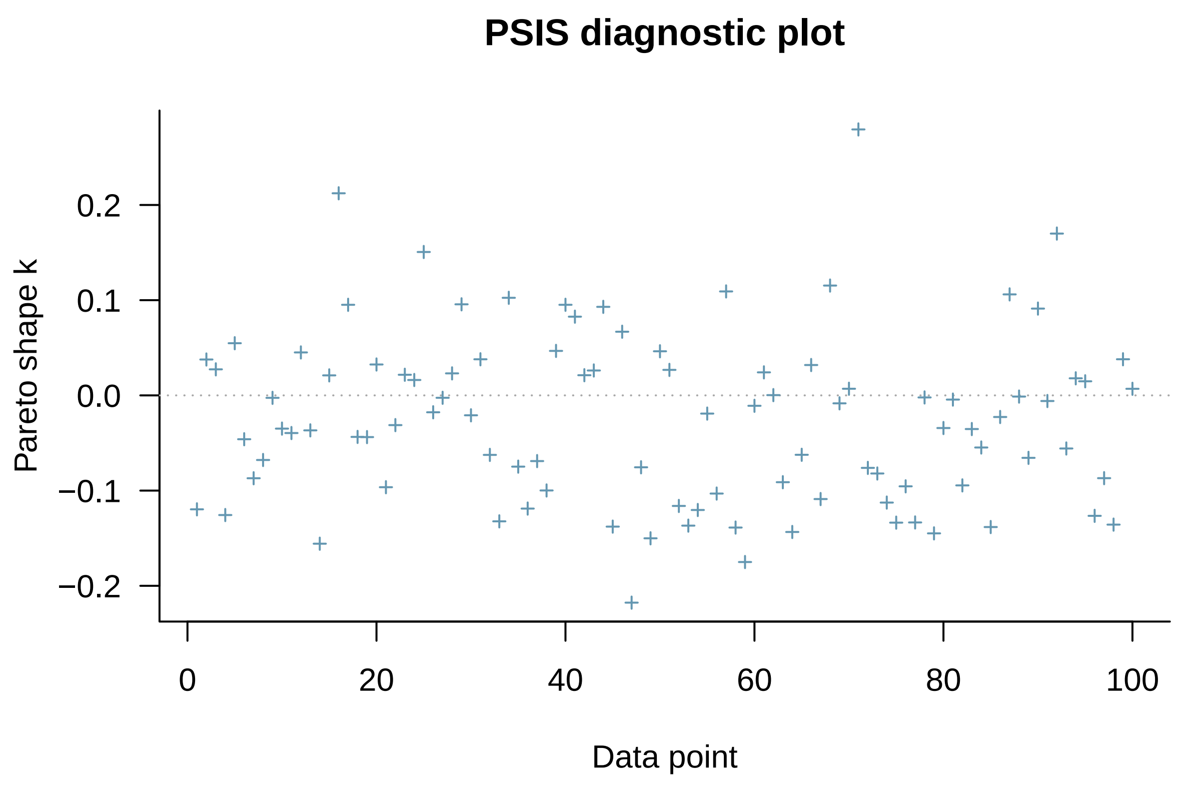
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
loo2 <- fit2$loo(save_psis = T)
print(loo2)
```

```
##
## Computed from 4000 by 100 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -158.6  7.1
## p_loo         2.8  0.6
## looic       317.2 14.1
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

```
plot(loo2)
```



```r
rm(data_list, fit1)
```

Model 2

```r
writeLines(readLines("mod2.stan"))
```

```
## data {
##   int<lower=0> N;
##   vector[N] y;
##   vector<lower=-1, upper=2.5>[N] t;
##
##   //priors
##   real<lower=0> tau_a;
##   real<lower=0> tau_b;
##   real<lower=0> tau_s;
##
##   int<lower=0, upper=1> only_prior;
## }
##
## parameters {
##   real a;
##   real<upper=0> b;
##   real<lower=0> sigma;
## }
##
## transformed parameters {
##   vector[N] log_mu = a + b * t;
## }
##
```

```
## model {
##   //priors
##   a ~ normal(0, tau_a);
##   b ~ normal(0, tau_b);
##   sigma ~ normal(0, tau_s);
##
##   if(only_prior == 0) {
##     y ~ normal(log_mu, sigma);
##   }
## }
##
## generated quantities {
##   vector[N] log_lik;
##   vector[N] y_pred;
##   for (i in 1:N) {
##     log_lik[i] = lognormal_lpdf(y[i] | log_mu[i], sigma);
##     y_pred[i] = normal_rng(log_mu[i], sigma);
##   }
## }
```

```
mod2 <- cmdstan_model("mod2.stan", compile = T)
```

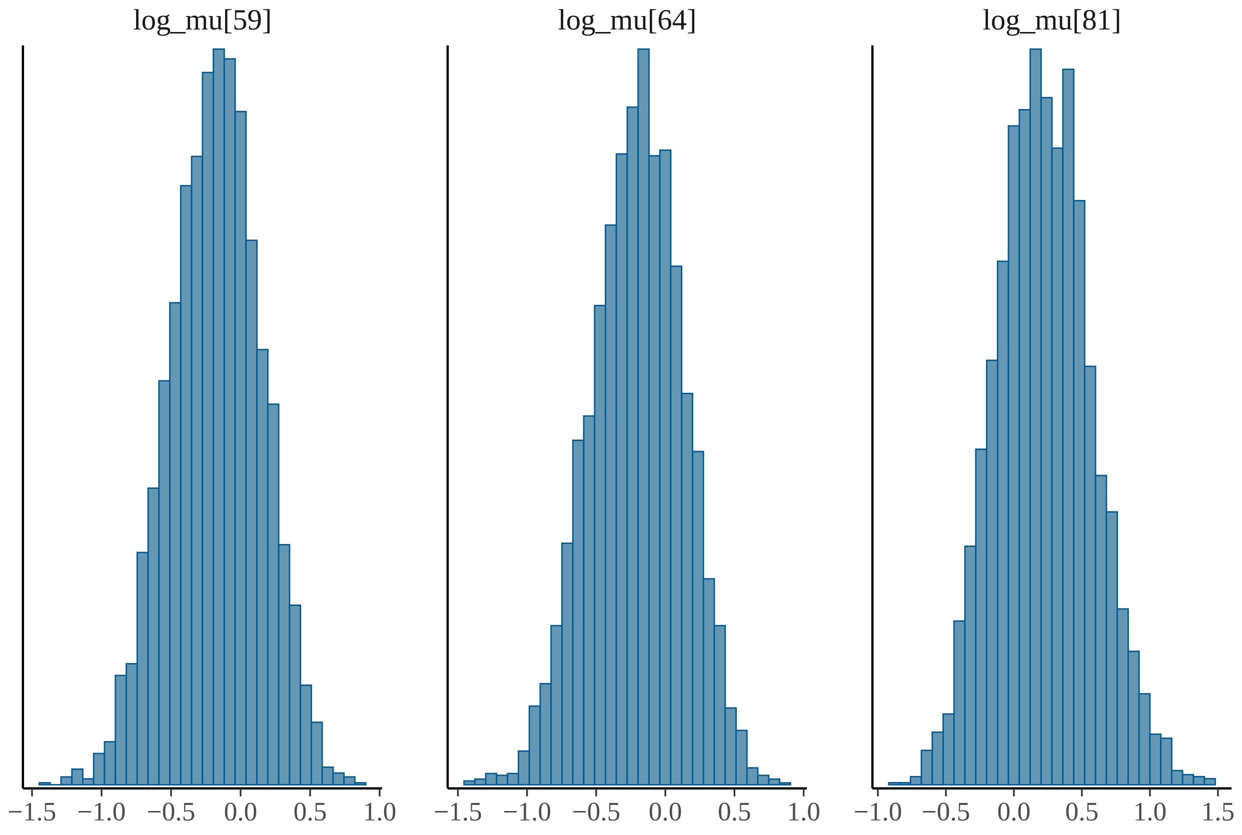
```
## Model executable is up to date!
```

```
data_list <- list(N = 100, y = y, t = t, tau_a = 0.3,
                  tau_b = 0.6, tau_s = 2, only_prior = 1)

fit3 <- mod2$sample(data_list, set.seed(333), refresh = 0, show_messages = T)
```

```
## Running MCMC with 4 sequential chains...
##
## Chain 1 finished in 0.2 seconds.
## Chain 2 finished in 0.2 seconds.
## Chain 3 finished in 0.2 seconds.
## Chain 4 finished in 0.2 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.2 seconds.
## Total execution time: 0.9 seconds.
```

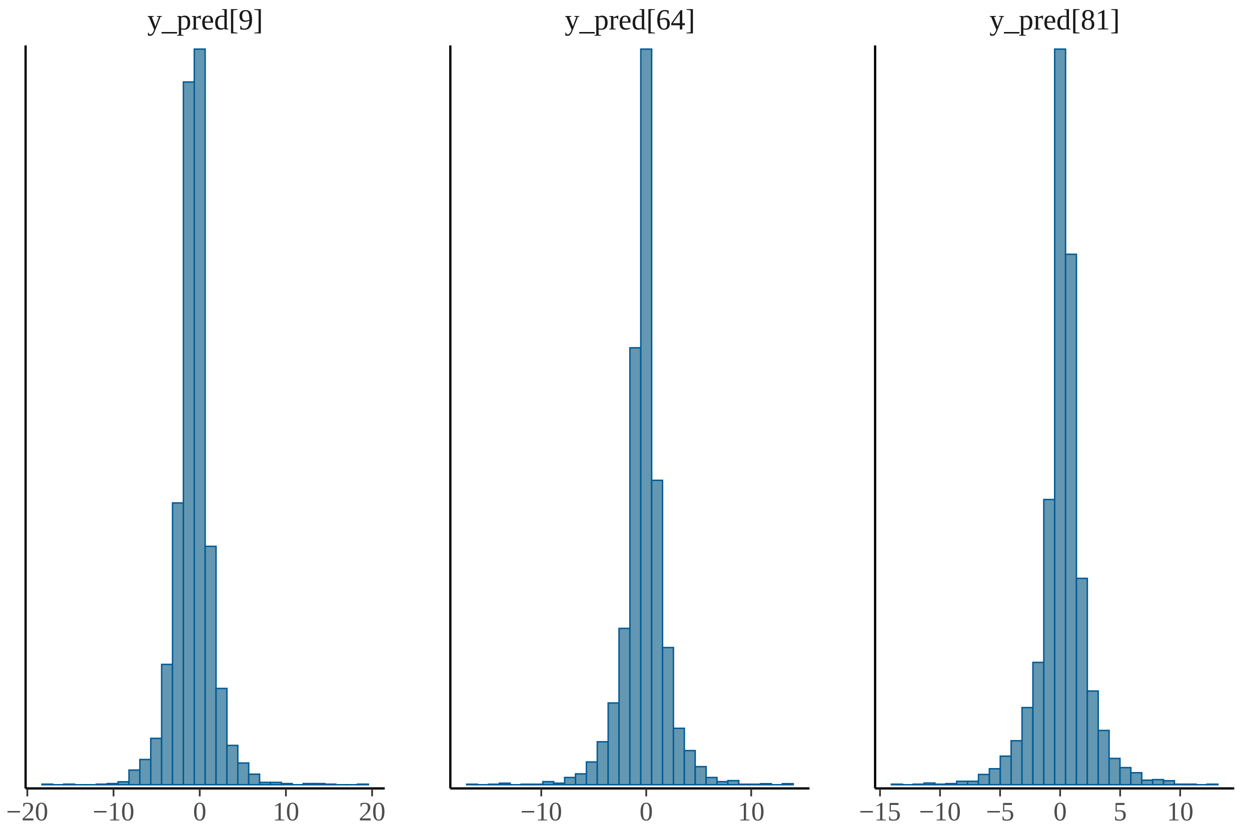
```
mcmc_hist(fit3$draws(c("log_mu[59]","log_mu[64]","log_mu[81]")))
```

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



```
mcmc_hist(fit3$draws(c("y_pred[9]","y_pred[64]","y_pred[81]")))
```

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



```r
fit3$print(max_rows = 20)
```

```
##    variable  mean median   sd  mad    q5   q95 rhat ess_bulk ess_tail
##  lp__       -2.63  -2.28 1.38 1.19 -5.37 -1.06 1.00     1326     1614
##  a           0.00  -0.01 0.30 0.30 -0.49  0.49 1.00     2649     2582
##  b          -0.47  -0.39 0.37 0.35 -1.16 -0.03 1.00     1658     1179
##  sigma       1.60   1.35 1.22 1.18  0.11  3.94 1.00     1582     1239
##  log_mu[1]  -0.22  -0.21 0.34 0.34 -0.79  0.32 1.00     2848     2892
##  log_mu[2]   0.18   0.18 0.33 0.34 -0.35  0.75 1.00     2384     2265
##  log_mu[3]   0.45   0.41 0.46 0.45 -0.24  1.27 1.00     2197     2354
##  log_mu[4]  -0.72  -0.62 0.62 0.59 -1.86  0.13 1.00     2489     2683
##  log_mu[5]   0.40   0.37 0.44 0.43 -0.25  1.19 1.00     2227     2376
##  log_mu[6]  -0.70  -0.60 0.61 0.58 -1.81  0.14 1.00     2506     2683
##  log_mu[7]  -0.29  -0.27 0.37 0.36 -0.93  0.28 1.00     2858     2881
##  log_mu[8]  -0.34  -0.31 0.39 0.39 -1.03  0.26 1.00     2840     2818
##  log_mu[9]  -0.70  -0.60 0.61 0.58 -1.82  0.14 1.00     2502     2683
##  log_mu[10] -0.17  -0.16 0.32 0.33 -0.70  0.36 1.00     2823     2576
##  log_mu[11]  0.18   0.17 0.33 0.34 -0.36  0.74 1.00     2389     2294
##  log_mu[12]  0.38   0.36 0.43 0.42 -0.26  1.15 1.00     2241     2479
##  log_mu[13]  0.07   0.06 0.30 0.31 -0.42  0.57 1.00     2530     2488
##  log_mu[14]  0.08   0.08 0.31 0.31 -0.41  0.59 1.00     2507     2416
##  log_mu[15] -0.75  -0.64 0.64 0.61 -1.93  0.13 1.00     2464     2723
##  log_mu[16] -0.43  -0.39 0.44 0.43 -1.21  0.23 1.00     2780     2763
## 
##  # showing 20 of 304 rows (change via 'max_rows' argument)
```

```r
data_list$only_prior = 0
```

```r
fit4 <- mod2$sample(data_list, set.seed(333), refresh = 0, show_messages = T)
```

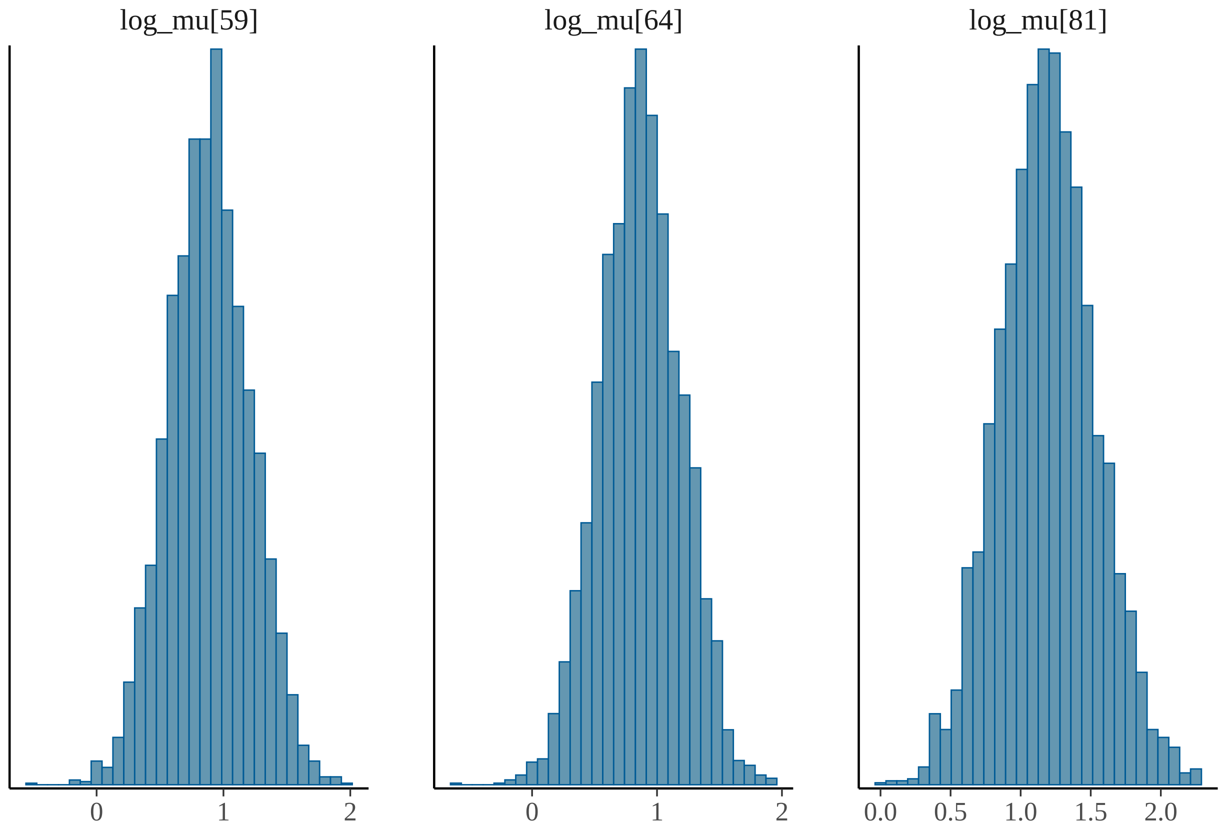
```
## Running MCMC with 4 sequential chains...
##
## Chain 1 finished in 0.2 seconds.
## Chain 2 finished in 0.2 seconds.
## Chain 3 finished in 0.2 seconds.
## Chain 4 finished in 0.2 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.2 seconds.
## Total execution time: 0.9 seconds.
```

```r
fit4$summary()
```

```
## # A tibble: 304 x 10
##    variable     mean   median    sd   mad       q5      q95  rhat ess_bulk
##    <chr>       <dbl>    <dbl> <dbl> <dbl>    <dbl>    <dbl> <dbl>    <dbl>
##  1 lp__     -278.    -278.    1.29  1.07  -280.    -2.77e+2  1.00    1520.
##  2 a           1.02     1.02  0.311 0.310    0.517  1.54e+0  1.00    2456.
##  3 b          -0.355   -0.308 0.256 0.259   -0.828 -3.34e-2  1.00    2096.
##  4 sigma       7.93     7.91  0.541 0.530    7.05   8.83e+0  1.00    2323.
##  5 log_mu[~    0.858    0.859 0.326 0.323    0.321  1.40e+0  1.00    2465.
##  6 log_mu[~    1.16     1.15  0.334 0.328    0.620  1.72e+0  1.00    2491.
##  7 log_mu[~    1.36     1.34  0.408 0.398    0.710  2.07e+0  1.00    2613.
##  8 log_mu[~    0.484    0.518 0.481 0.467   -0.360  1.21e+0  1.00    2602.
##  9 log_mu[~    1.33     1.31  0.393 0.388    0.692  2.01e+0  1.00    2595.
## 10 log_mu[~    0.499    0.530 0.473 0.459   -0.330  1.21e+0  1.00    2603.
## # ... with 294 more rows, and 1 more variable: ess_tail <dbl>
```

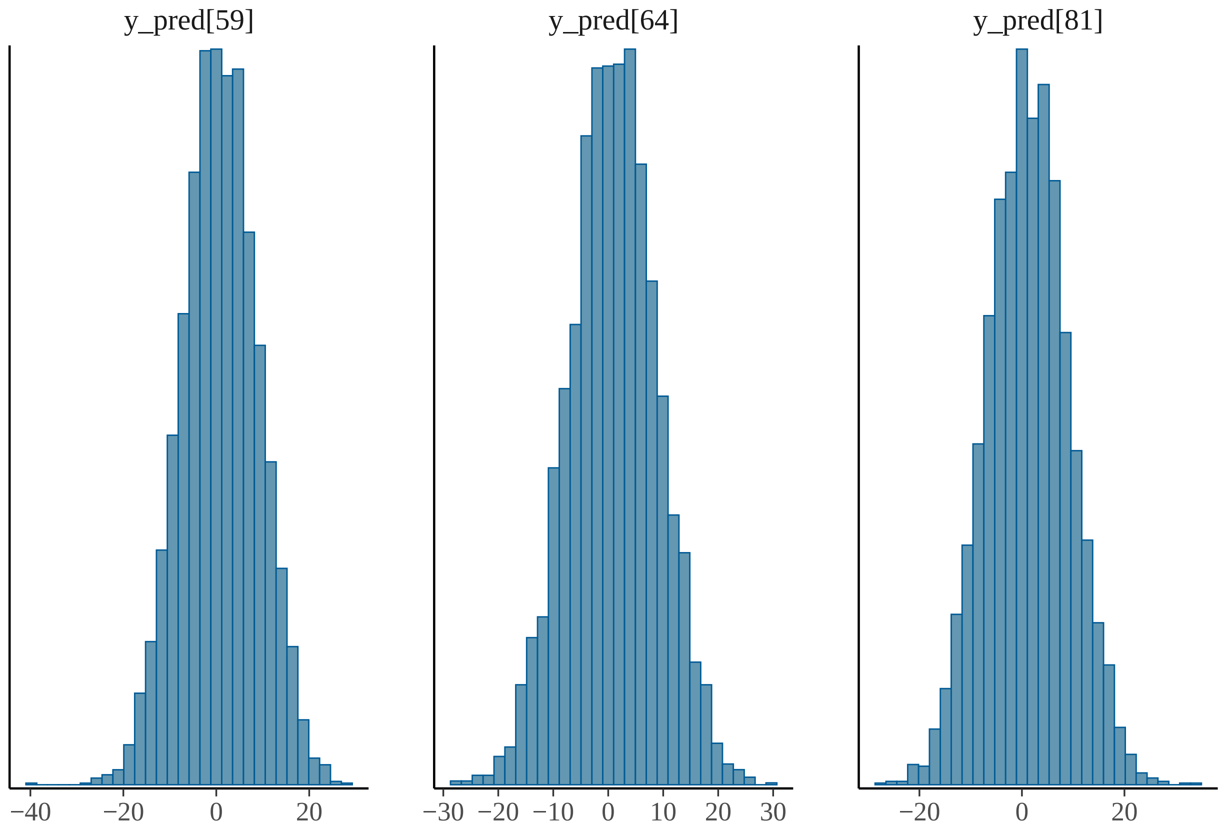
```r
mcmc_hist(fit4$draws(c("log_mu[59]","log_mu[64]","log_mu[81]")))
```

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



```r
mcmc_hist(fit4$draws(c("y_pred[59]","y_pred[64]","y_pred[81]")))
```

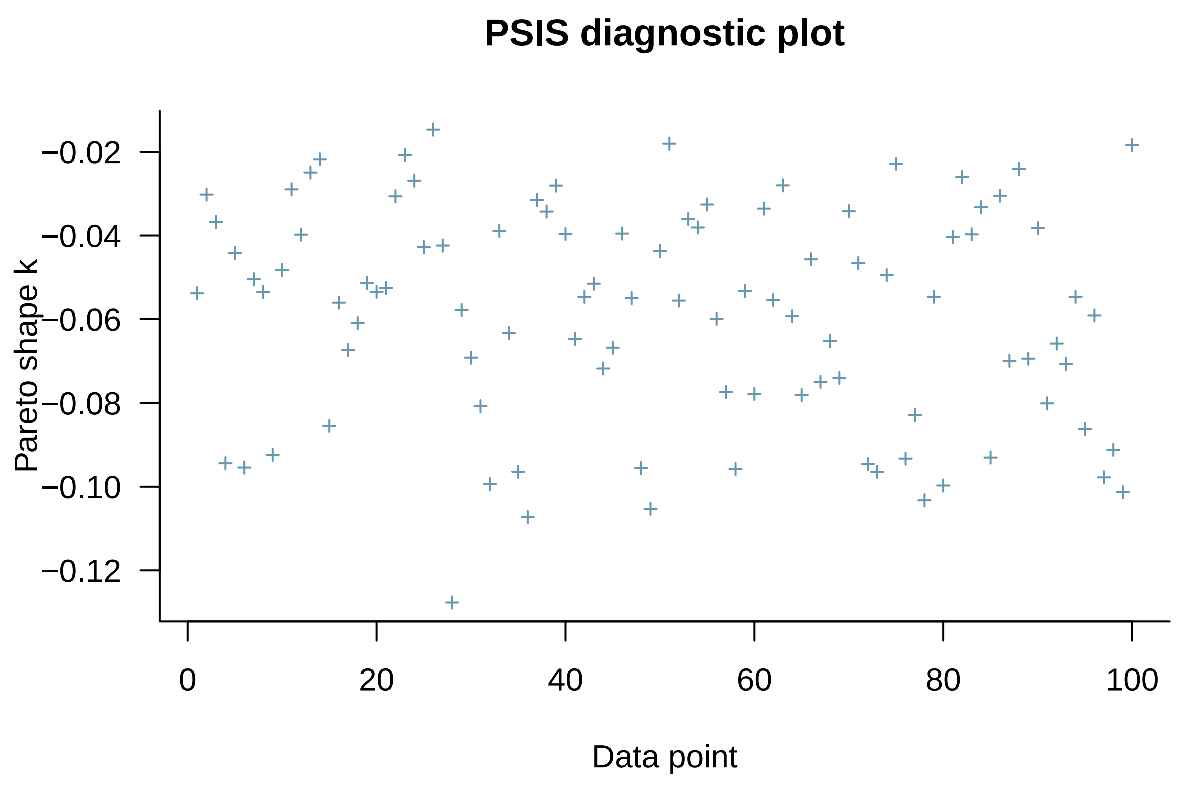
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```r
loo4 <- fit4$loo(save_psis = T)
print(loo4)
```

```
##
## Computed from 4000 by 100 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -483.4  7.4
## p_loo         0.5  0.0
## looic       966.9 14.8
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

```r
plot(loo4)
```



```r
rm(data_list, fit3)
```

Comparing Models

```r
loo_compare(loo2, loo4)
```

```
##        elpd_diff se_diff
## model1    0.0       0.0 
## model2 -324.8      10.6 
```

Both of these models have appropriate r-hats and ESS bulk. In all of our output r-hat is around 1 and ESS bulk is way above 500, most greater than 2000. We have considered these to be basic checks for an appropriate model which both of these models have passed. However, after looking at our posterior checks we have observed that Model 1 seems to fit to our data better. Model 1 has the appropriate centralization and the spread in the histogram outputs above whereas the Model 2 had a spread that was unnecessarily large. In the second model, the y_i^* has more negative values most probably because of this greater spread. Nevertheless, the first model did not have much negative values as we have expected and the y_i^* was in range of the data given to us, hence, the model probably had better and more accurate priors which have caused our Stan code to fit the data more accurately. Considering, that our mean of y is approximately 7.86 and natural log of that is approximately 2.06 we have expected μ and $\log(\mu)$ to be relatively close to these numbers. In our posterior checks, μ compared to $\log(\mu)$ had values much closer to the original mean and in the similar range. In order to be sure of my predictions, I have used leave one out cross validation and Pareto k diagnostic values. As can be observed from the PSIS plots above, all of my values are a good fit without exceptions, that is to say there are not any Pareto k values above 0.5 for either one of the models. Both models had similar standard errors in the leave one out cross validation but Model 1 had higher expected log pointwise predictive density (ELDP). This value is a sum of the N observations pointwise log predictive densities and it is either 0 or negative. When we compare the ELDPs corresponding to each of these models we observe a difference of 324.9. In addition, when we compare p_loos corresponding to each model, which is the effectiveness of the parameters in the model, we observe that Model 1 has 5 times greater value. Thus, we can conclude that one of these models is better than the other one, and that is Model 1.