7393 Exam 2 R Code Andrei Matveev

```r
#library(data.table)
library(truncnorm)
library(coda)
#devtools::install_github("stan-dev/loo")
library(loo)
```

```
## This is loo version 2.4.0

## - Online documentation and vignettes at mc-stan.org/loo

## - As of v2.0.0 loo defaults to 1 core but we recommend using as many as possible. Use the 'cores' ar
```

```r
library(rstan)
```

```
## Loading required package: StanHeaders

## Loading required package: ggplot2

## rstan (Version 2.21.2, GitRev: 2e1f913d3ca3)

## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)

##
## Attaching package: 'rstan'

## The following object is masked from 'package:coda':
##
##     traceplot
```

```r
library(shinystan)
```

```
## Loading required package: shiny

##
## This is shinystan version 2.5.0
```

```r
library(invgamma)
library(bayesplot)
```

```
## This is bayesplot version 1.7.2

## - Online documentation and vignettes at mc-stan.org/bayesplot

## - bayesplot theme set to bayesplot::theme_default()

##    * Does _not_ affect other ggplot2 plots

##    * See ?bayesplot_theme_set for details on theme setting
```

```r
#library(posterior)
library(rstanarm)
```

```
## Loading required package: Rcpp
## This is rstanarm version 2.21.1
## - See https://mc-stan.org/rstanarm/articles/priors for changes to default priors!
## - Default priors may change, so it's safest to specify priors, even if equivalent to the defaults.
## - For execution on a local, multicore CPU with excess RAM we recommend calling
##   options(mc.cores = parallel::detectCores())
##
## Attaching package: 'rstanarm'
## The following object is masked from 'package:rstan':
##
##     loo
```

```r
devtools::install_github("jfrench/bayesutils")
```

```
## Skipping install of 'bayesutils' from a github remote, the SHA1 (17473e8d) has not changed since las
##   Use `force = TRUE` to force installation
```

```r
setwd('/Users/AM/Documents/_CU Masters/2020 fall Bayesian_7393/code/Bayesian_Statistics_Class_Code/Exam
#df1 = load("diamonds_simple.rda")
#rm(df1, diamonds_simple)
df = bayesutils::diamonds_simple

df$lprice = log(df$price)
df$lcarat = log(df$carat)

df_covid = bayesutils::covid_dec4
```

```r
lq_theta_y = function(sigmaSQ, beta0, beta1, lpr = df$lprice, lcar = df$lcarat) {
  ld = dnorm(beta0, 0, 100, log = TRUE) +
    dnorm(beta1, 0, 100, log = TRUE) +
    dinvgamma(sigmaSQ, shape = 0.01, rate = 0.01, log = TRUE) +
    sum(dnorm(lpr, mean = (beta0 + beta1 * lcar), sd = sqrt(sigmaSQ), log = TRUE))
  return(ld)
}
```

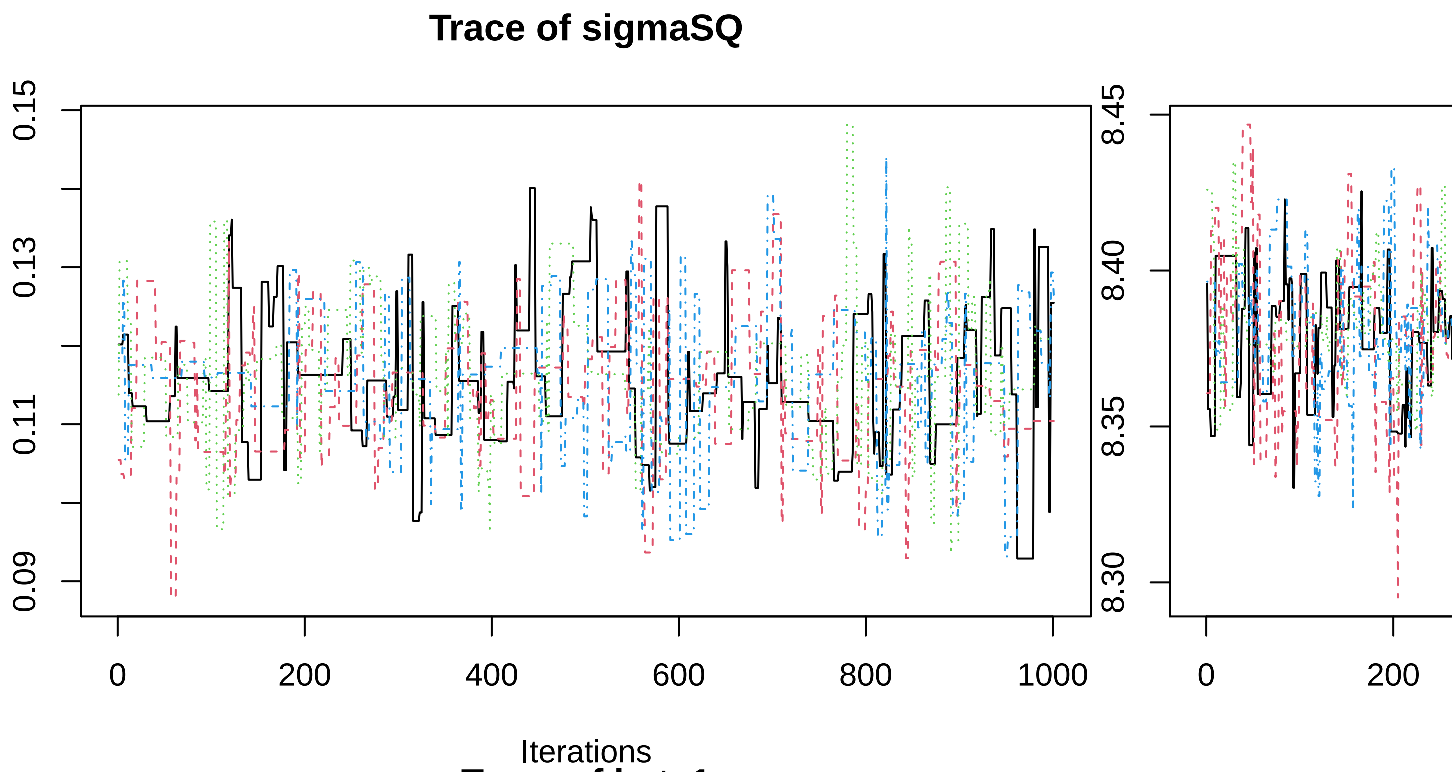
```r
mh = function(B, theta_start) {
  theta = array(0, c((B+1), 3), dimnames = list(c(), c("sigmaSQ", "beta0", "beta1")))

  theta[1,1] = theta_start[1]
  theta[1,2] = theta_start[2]
  theta[1,3] = theta_start[3]

  for (i in 2:dim(theta)[1]) {

    ###  step for sigmaSQ
    beta0_star = theta[(i-1),2]
    beta1_star = theta[(i-1),3]

    sigmaSQ_star = rtruncnorm(n = 1, a=0, b=Inf, mean = theta[(i-1),1], sd = 0.1)

    num_logr = lq_theta_y(sigmaSQ = sigmaSQ_star, beta0 = beta0_star, beta1 = beta1_star) -
      log(dtruncnorm(x = sigmaSQ_star, a=0, b=Inf, mean = theta[(i-1),1], sd = 0.1))
    den_logr = lq_theta_y(sigmaSQ = theta[i-1, 1], beta0 = beta0_star, beta1 = beta1_star) -
      log(dtruncnorm(x = theta[i-1, 1], a=0, b=Inf, mean = sigmaSQ_star, sd = 0.1))
    logr = num_logr - den_logr
    if (log(runif(1)) <= min(logr, 0)) {
      theta[i,1] = sigmaSQ_star
    } else {
      theta[i,1] = theta[(i - 1), 1]
    }

    ###  step for beta0
    beta1_star = theta[(i-1),3] # it is the repeated code, but I need it to keep the interpretability
    sigmaSQ_star = theta[i,1] # update sigmaSQ_star after the Gibbs step for sigmaSQ

    beta0_star = rnorm(1, theta[(i-1),2], 0.1) # !!! check the parametrization (0.1 or 0.1^2)

    num_logr = lq_theta_y(sigmaSQ = sigmaSQ_star, beta0 = beta0_star, beta1 = beta1_star) -
      dnorm(x = beta0_star, mean = theta[(i-1),2], sd = 0.1, log = TRUE)
    den_logr = lq_theta_y(sigmaSQ = sigmaSQ_star, beta0 = theta[i-1, 2], beta1 = beta1_star) -
      dnorm(x = theta[(i-1),2], mean = beta0_star, sd = 0.1, log = TRUE)


    logr = num_logr - den_logr
    if (log(runif(1)) <= min(logr, 0)) {
      theta[i,2] = beta0_star
    } else {
      theta[i,2] = theta[(i - 1), 2]
    }

    ###  step for beta1
    sigmaSQ_star = theta[i,1] # it is the repeated code, but I need it to keep the interpretability
    beta0_star = theta[i,2] # update beta0_star after the Gibbs step for beta0

    beta1_star = rnorm(1, theta[(i-1),3], 0.1) #

    num_logr = lq_theta_y(sigmaSQ = sigmaSQ_star, beta0 = beta0_star, beta1 = beta1_star) -
      dnorm(x = beta1_star, mean = theta[(i-1),3], sd = 0.1, log = TRUE)
    den_logr = lq_theta_y(sigmaSQ = sigmaSQ_star, beta0 = beta0_star, beta1 = theta[i-1, 3]) -
      dnorm(x = theta[(i-1),3], mean = beta1_star, sd = 0.1, log = TRUE)


    logr = num_logr - den_logr
    if (log(runif(1)) <= min(logr, 0)) {
      theta[i,3] = beta1_star
    } else {
      theta[i,3] = theta[(i - 1), 3]
    }

  }
  return(theta)
}
```

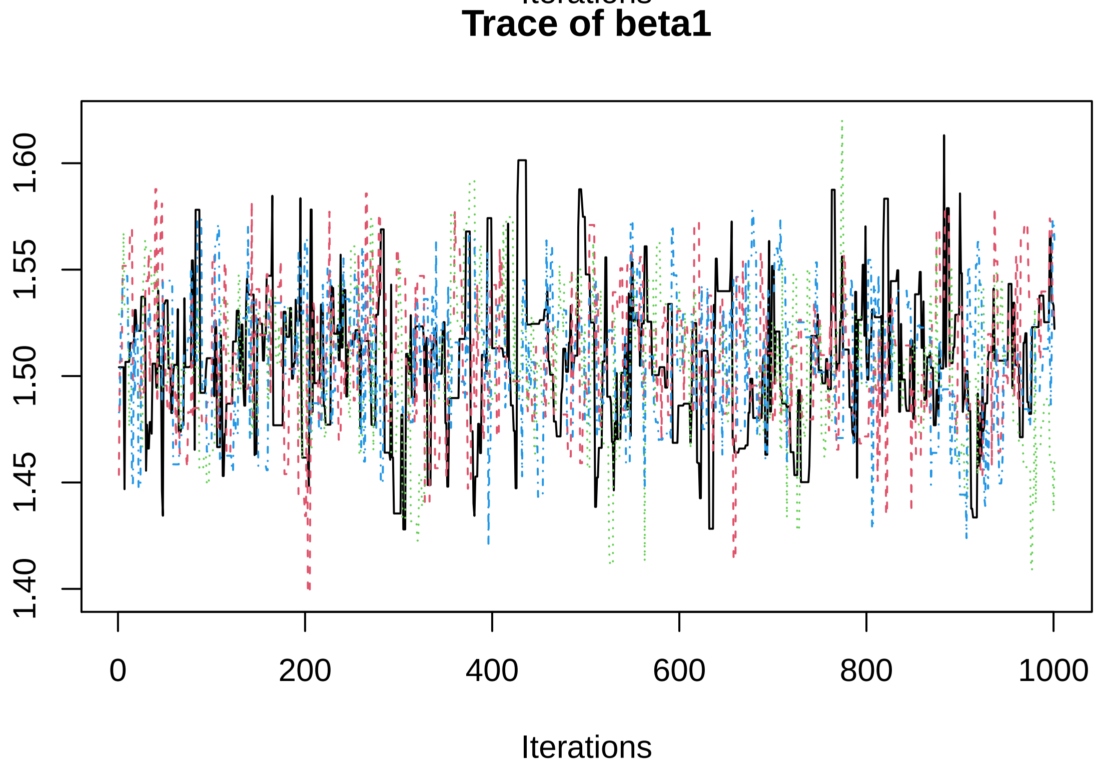
```r
B = 10^5
keep = (B/2 + 1):(B + 1)
chain1 = mh(B, theta_start = c(0.1, -1, -1))
```

```r
chain2 = mh(B, theta_start = c(0.3, 0, 0))
chain3 = mh(B, theta_start = c(0.5, -1, 1))
chain4 = mh(B, theta_start = c(0.2, 1, 1))

mc = mcmc.list(mcmc(chain1[keep,]), mcmc(chain2[keep,]),
                 mcmc(chain3[keep,]), mcmc(chain4[keep,]))
summary(mc)
```

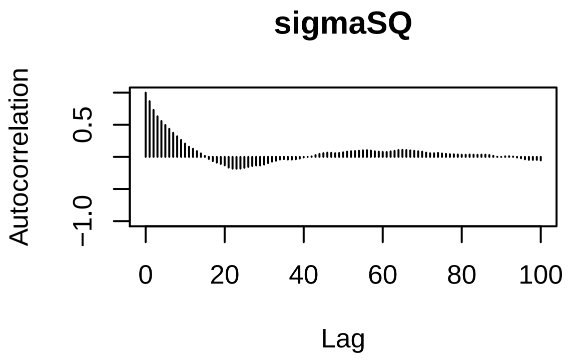
```
##
## Iterations = 1:50001
## Thinning interval = 1
## Number of chains = 4
## Sample size per chain = 50001
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean       SD  Naive SE Time-series SE
## sigmaSQ 0.1151 0.009198 2.057e-05      7.053e-05
## beta0   8.3795 0.022435 5.017e-05      1.643e-04
## beta1   1.5061 0.031364 7.013e-05      2.142e-04
##
## 2. Quantiles for each variable:
##
##            2.5%    25%    50%    75%  97.5%
## sigmaSQ 0.09875 0.1086 0.1146 0.1209 0.1346
## beta0   8.33571 8.3643 8.3796 8.3947 8.4234
## beta1   1.44474 1.4851 1.5060 1.5274 1.5677
```

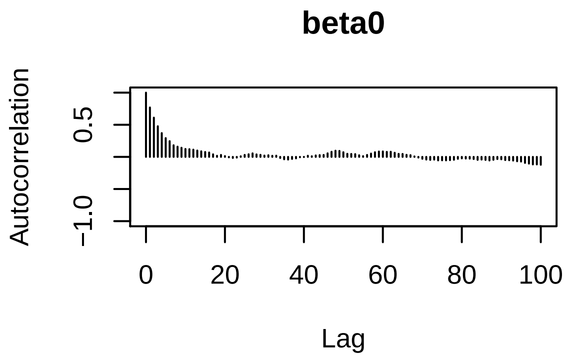
```r
keep = (B/2 + 17001):(B/2 + 18001)

mc = mcmc.list(mcmc(chain1[keep,]), mcmc(chain2[keep,]),
                 mcmc(chain3[keep,]), mcmc(chain4[keep,]))
coda::traceplot(mc)
```

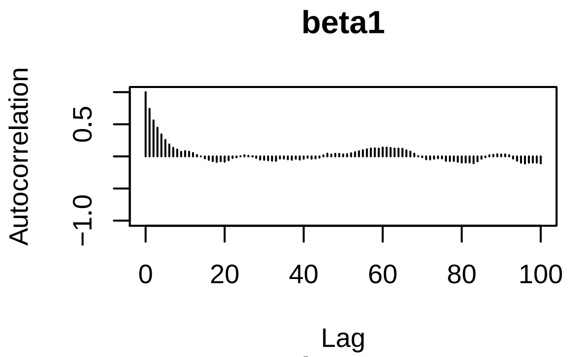


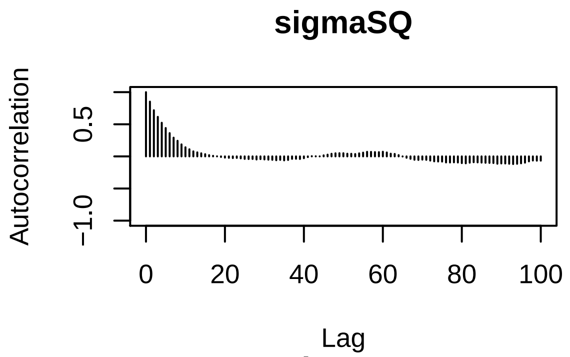


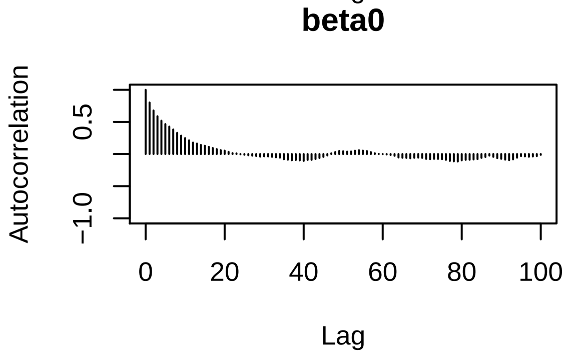
```
coda::autocorr.plot(mc, lag.max = 100, auto.layout = TRUE)
```

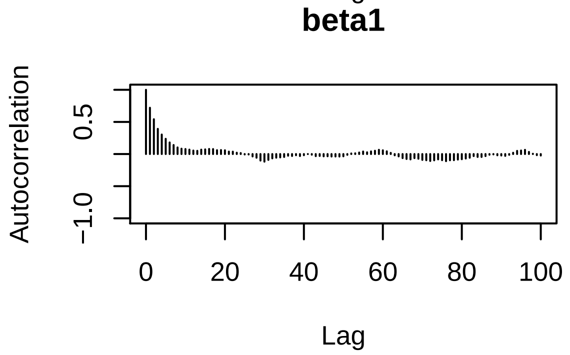

sigmaSQ
Autocorrelation
−1.0
0.5
0
20
40
60
80
100
Lag

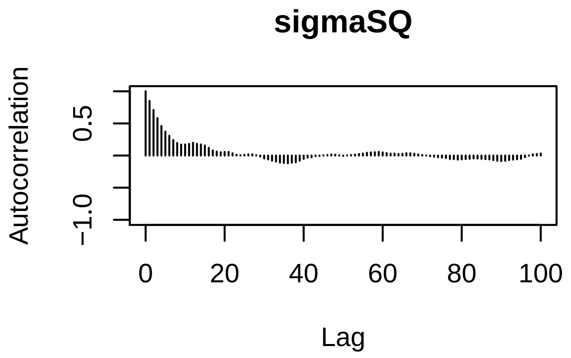

beta0
Autocorrelation
−1.0
0.5
0
20
40
60
80
100
Lag

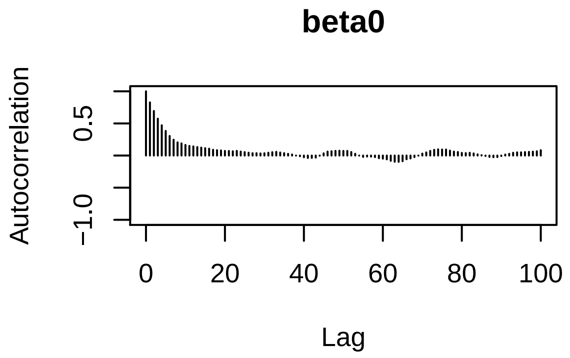

beta1
Autocorrelation
−1.0
0.5
0
20
40
60
80
100
Lag

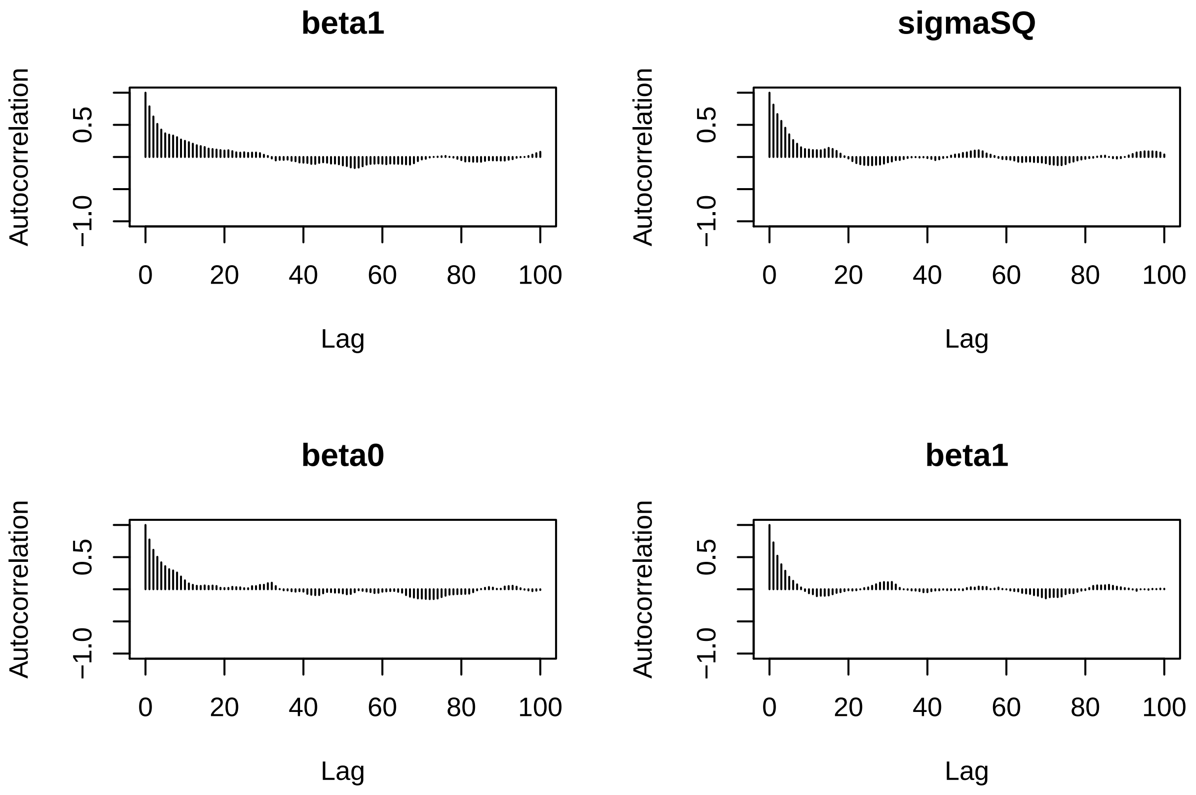

sigmaSQ
Autocorrelation
−1.0
0.5
0
20
40
60
80
100
Lag


beta0
Autocorrelation
−1.0
0.5
0
20
40
60
80
100
Lag


beta1
Autocorrelation
−1.0
0.5
0
20
40
60
80
100
Lag


sigmaSQ
Autocorrelation
−1.0
0.5
0
20
40
60
80
100
Lag


beta0
Autocorrelation
−1.0
0.5
0
20
40
60
80
100
Lag



```r
gelman.diag(mc, autoburnin = FALSE)
```

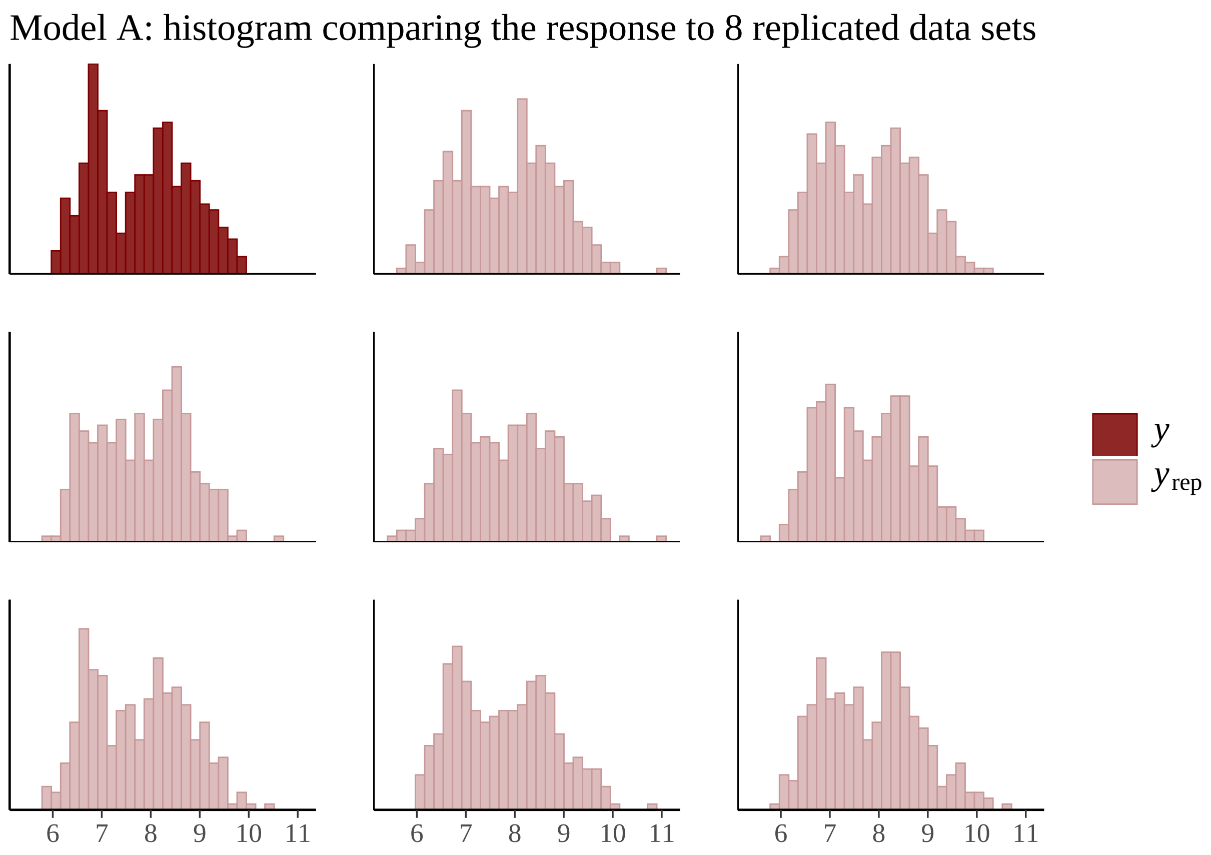
```
## Potential scale reduction factors:
##
##         Point est. Upper C.I.
## sigmaSQ       1.00       1.01
## beta0         1.01       1.01
## beta1         1.00       1.00
##
## Multivariate psrf
##
## 1
```

```r
geweke.diag(mc)
```

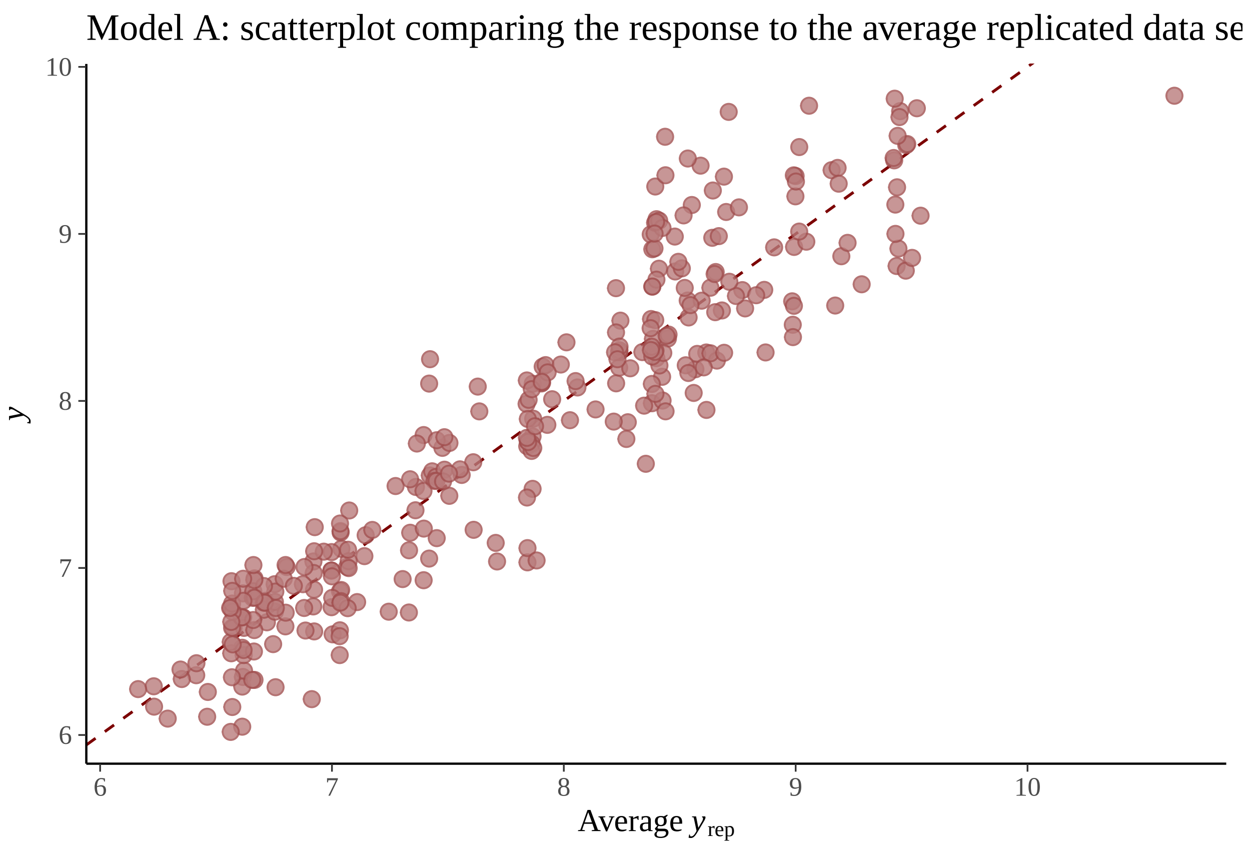
```
## [[1]]
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##  sigmaSQ    beta0    beta1
## -0.40871  0.06296  0.40829
##
##
## [[2]]
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
## sigmaSQ   beta0   beta1
## -0.2999  0.2575 -0.3738
##
##
## [[3]]
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
## sigmaSQ   beta0   beta1
##  0.2173  0.8936  0.2712
##
##
## [[4]]
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
## sigmaSQ   beta0   beta1
##  0.9673  0.7504  0.4157
```

```r
# extracting the MCMC samples from the soda_fit object
samples = extract(r_fit_A)
ncycles = length(samples[[1]])
T <- length(df$price)

# each row of yrep is a sample from the pp distribution
yrep = matrix(0, ncol = T, nrow = ncycles)
for (i in seq_len(T)) {
  mui = samples$beta0 + samples$beta1 * df$lcarat[i]
  yrep[, i] = rnorm(ncycles, mean = mui, sd = sqrt(samples$sigmaSQ))
}

# approximate posterior predictive density for an observation
# with the same covariates as observation 1
#plot(density(yrep[,1]))
color_scheme_set("red")

# posterior predictive check
y = df$lprice
ppc_hist(y, yrep[sample(1:ncycles, 8),]) + ggtitle("Model A: histogram comparing the response to 8 repli
```

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

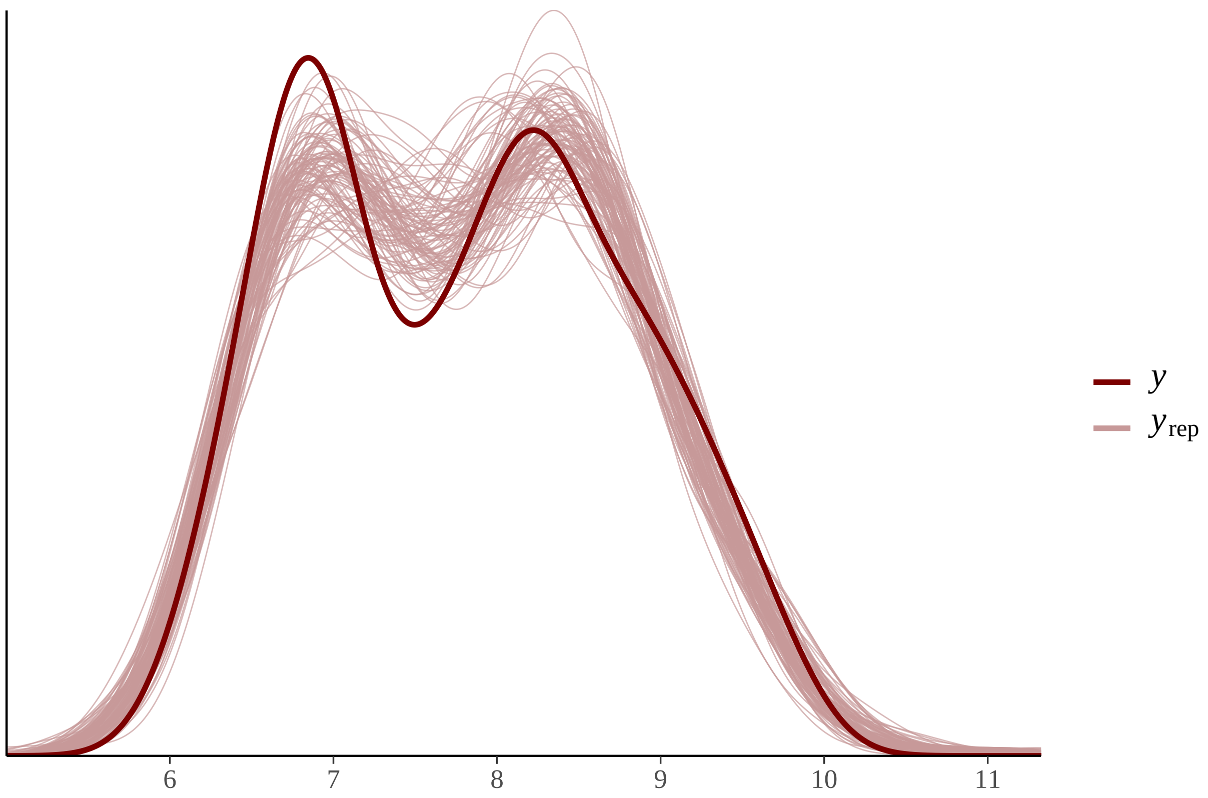


```
# scatterplot of y vs average yrep
ppc_scatter_avg(y, yrep) + ggtitle("Model A: scatterplot comparing the response to the average replicat
```

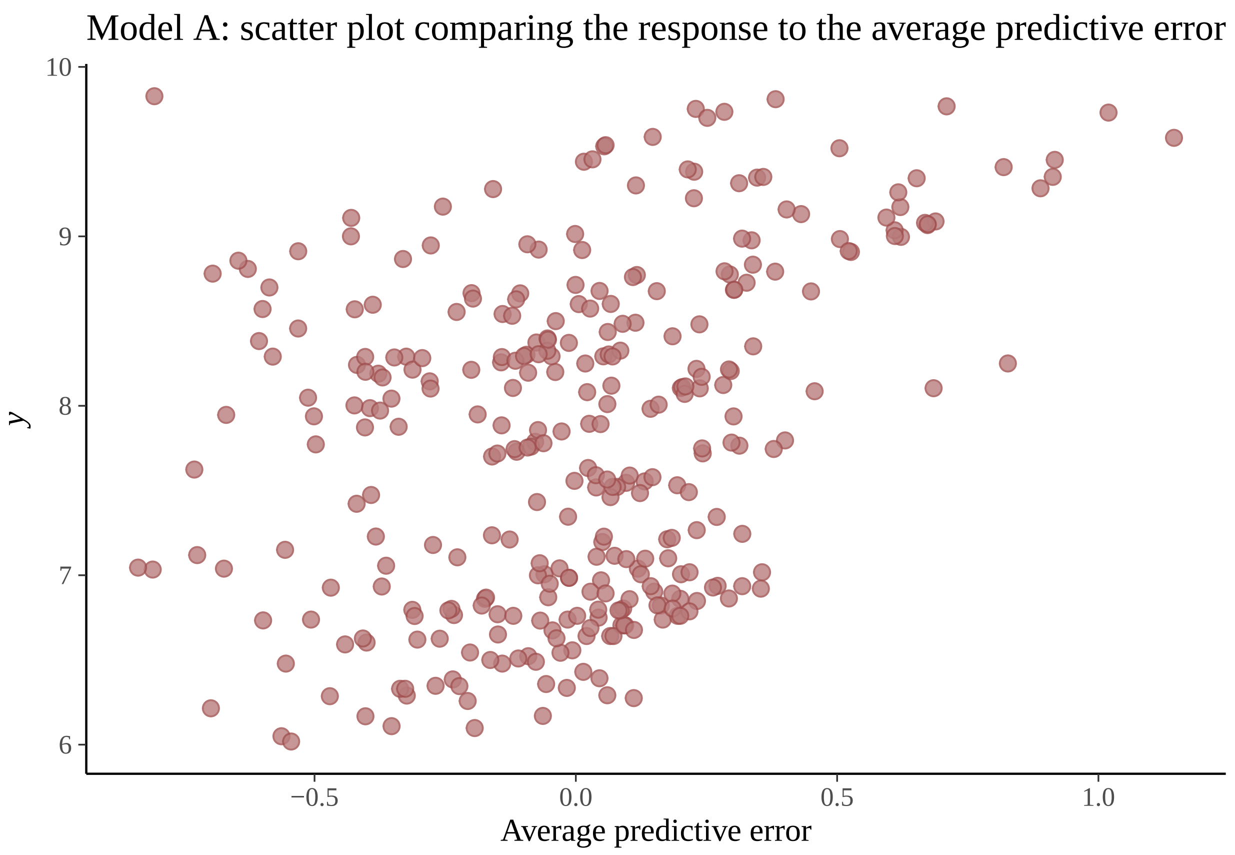


```
ppc_dens_overlay(y, yrep = yrep[sample(1:ncycles, 100),]) + ggtitle("Model A: density plot comparing the
```

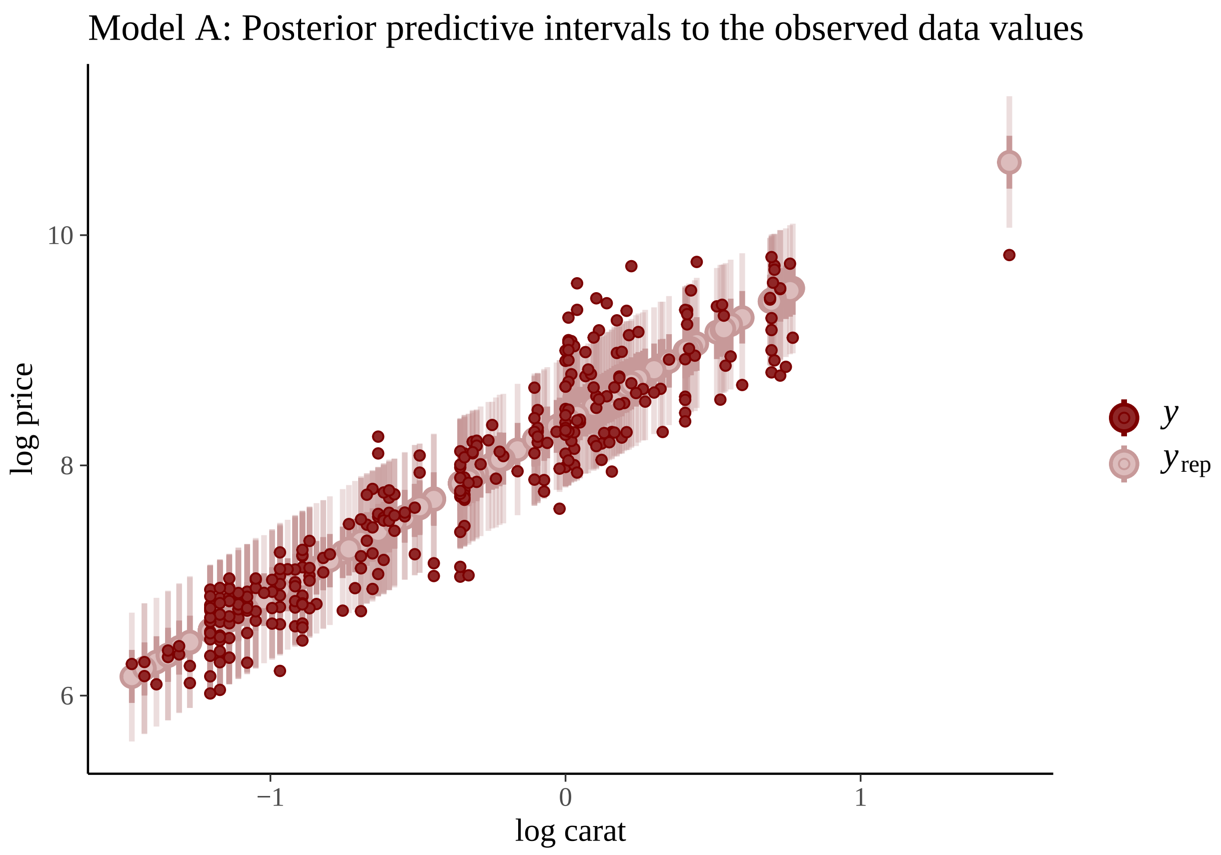
Model A: density plot comparing the density of the response to the densities of 10



```
ppc_error_scatter_avg(y, yrep = yrep) + ggtitle("Model A: scatter plot comparing the response to the ave
```



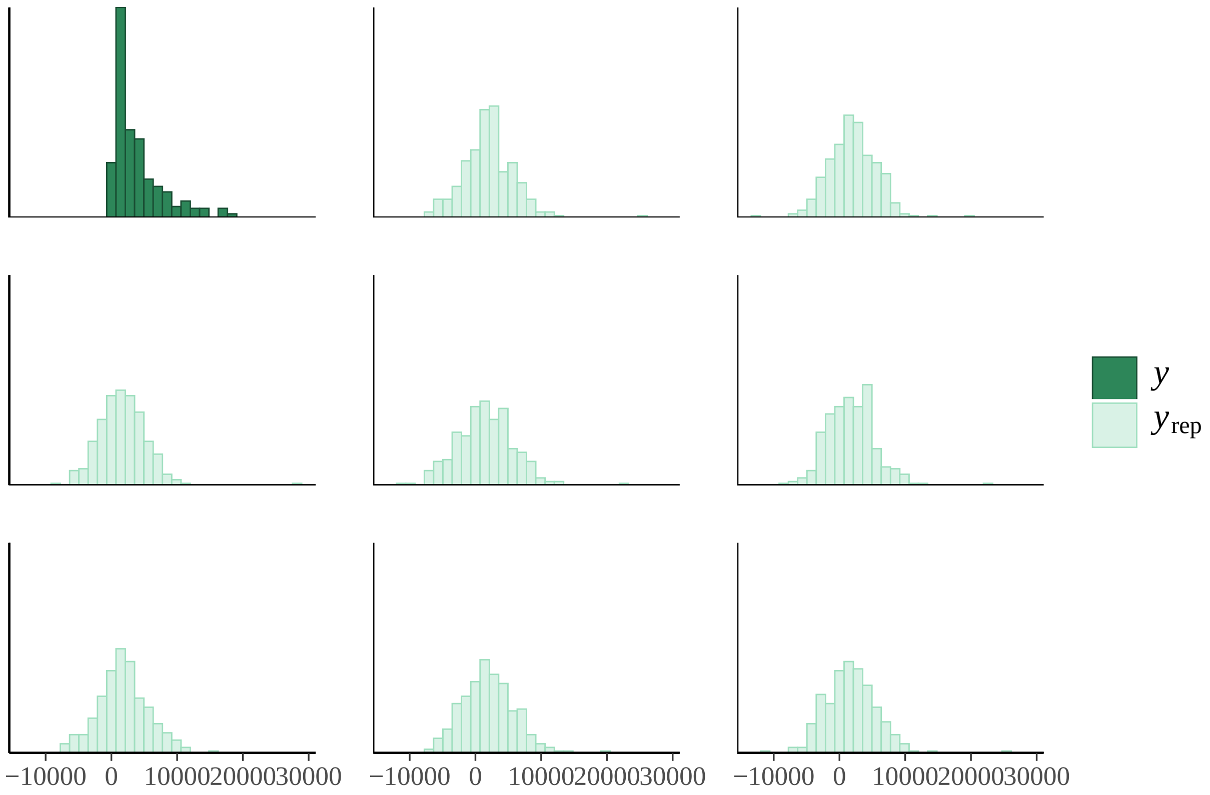
```
ppc_intervals(y = y, yrep = yrep, x = df$lcarat) + ggtitle("Model A: Posterior predictive intervals to t
```



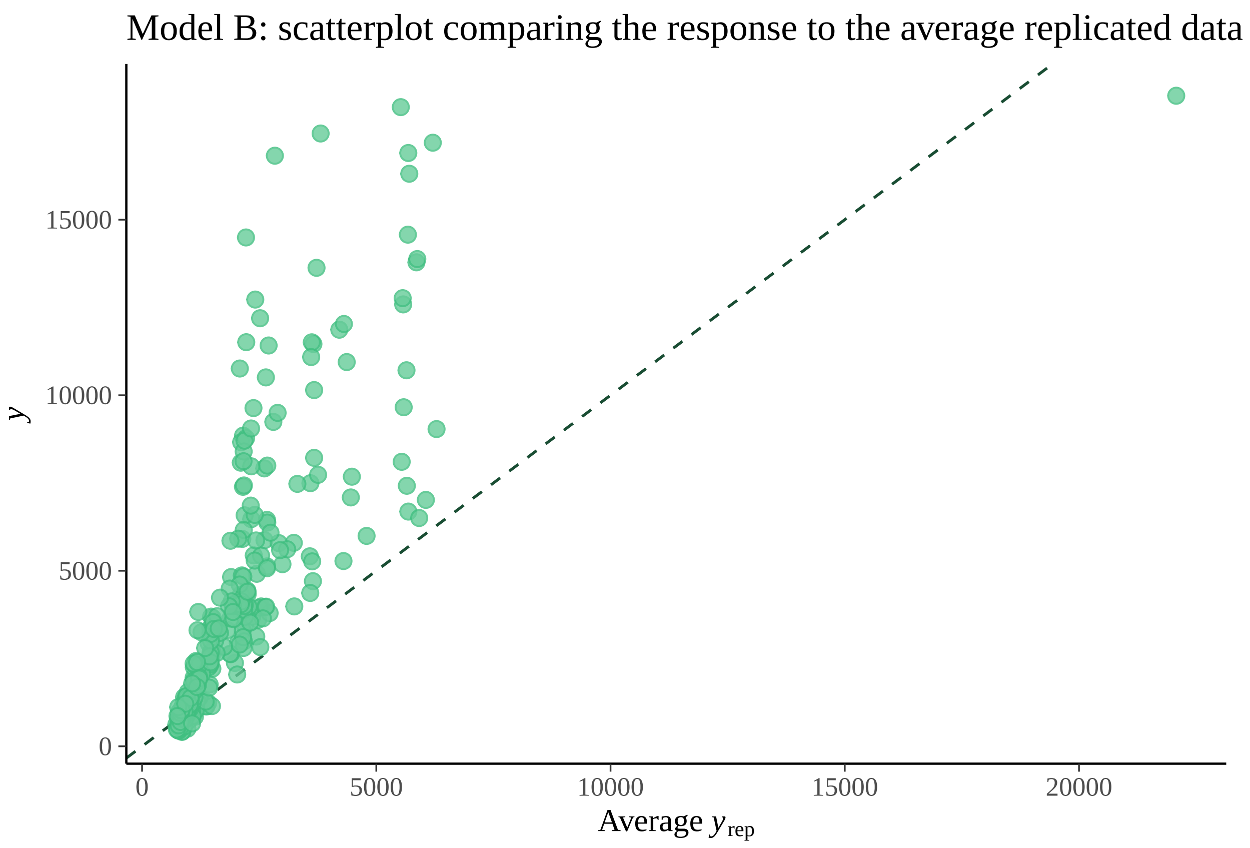
```r
# extracting the MCMC samples from the soda_fit object
samples = extract(r_fit_B)
ncycles = length(samples[[1]])
T <- length(df$price)

# each row of yrep is a sample from the pp distribution
yrep = matrix(0, ncol = T, nrow = ncycles)
for (i in seq_len(T)) {
  mui = samples$beta0 + samples$beta1 * df$carat[i] + samples$beta2 * df$carat[i] * df$carat[i]
  yrep[, i] = rnorm(ncycles, mean = mui, sd = sqrt(samples$sigmaSQ))
}

# approximate posterior predictive density for an observation
# with the same covariates as observation 1
#plot(density(yrep[,1]))
color_scheme_set("green")

# posterior predictive check
y = df$price
ppc_hist(y, yrep[sample(1:ncycles, 8),]) + ggtitle("Model B: histogram comparing the response to 8 repli
```

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

Model B: histogram comparing the response to 8 replicated data sets

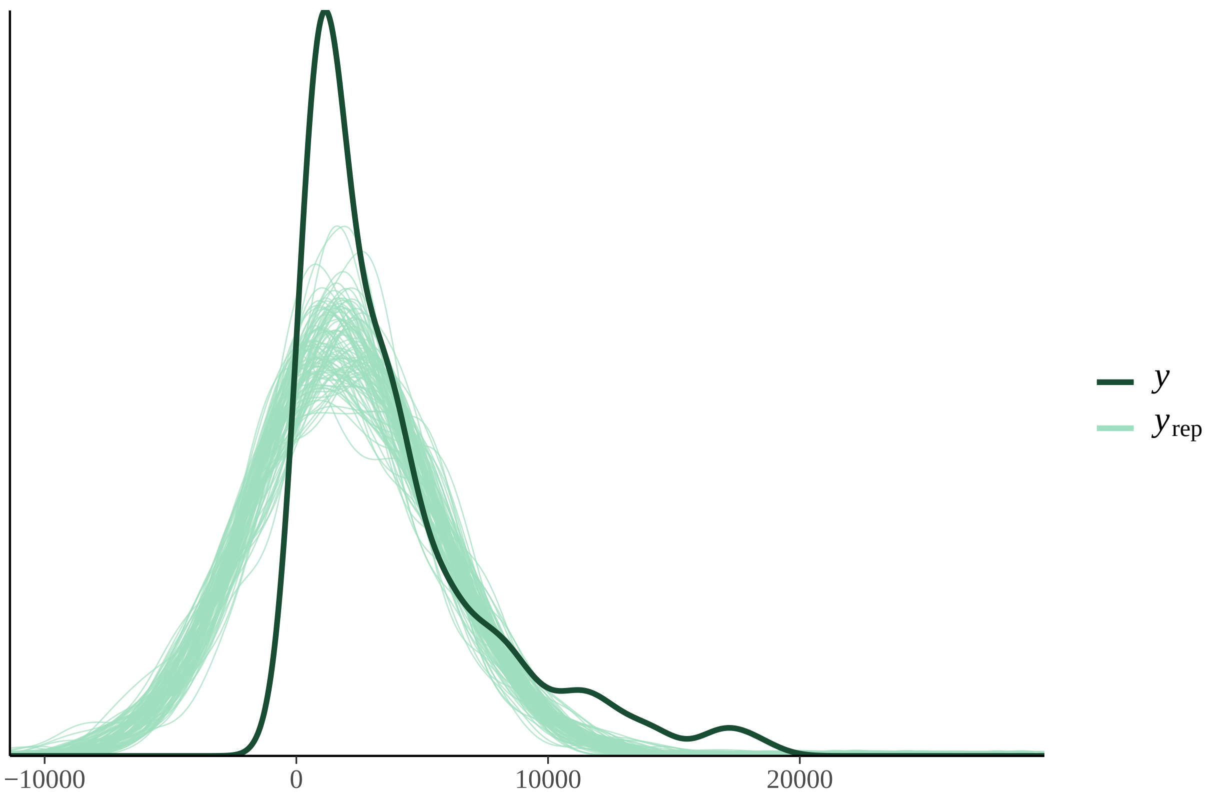


```
# scatterplot of y vs average yrep
ppc_scatter_avg(y, yrep) + ggtitle("Model B: scatterplot comparing the response to the average replicat
```



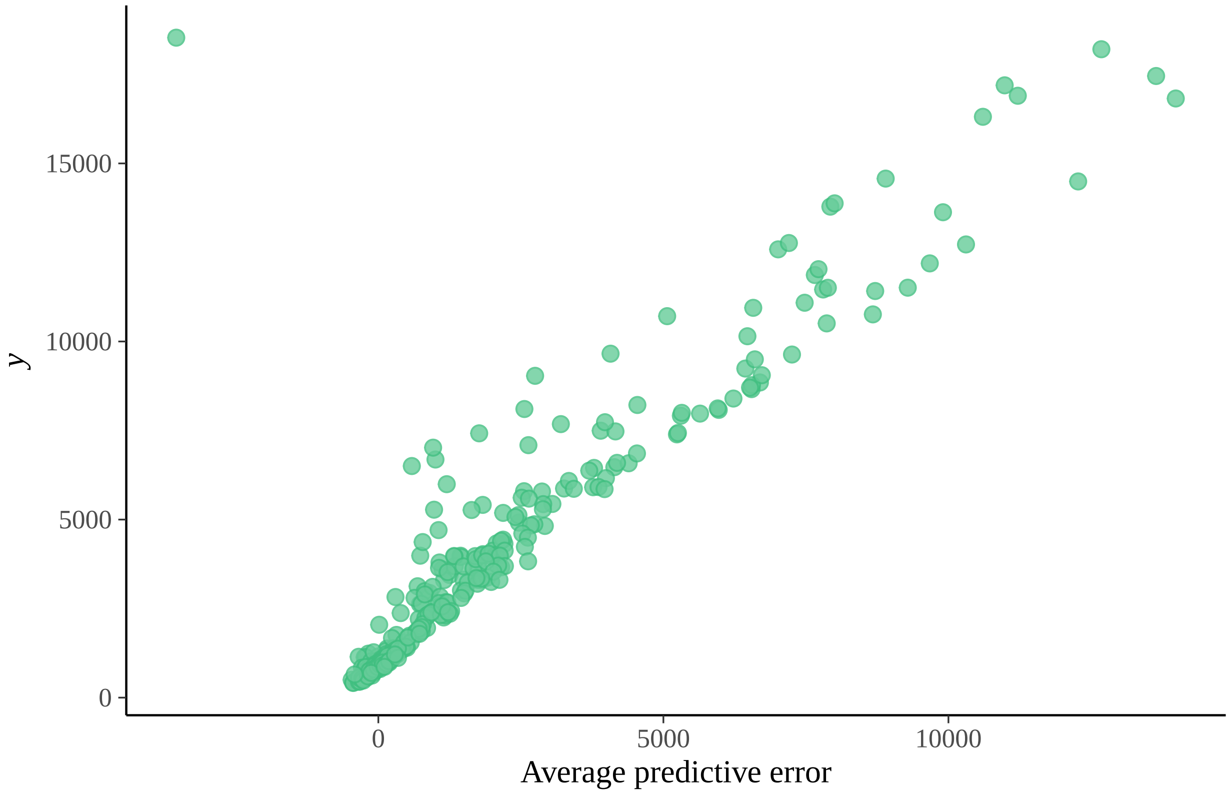
```
ppc_dens_overlay(y, yrep = yrep[sample(1:ncycles, 100),]) + ggtitle("Model B: density plot comparing the
```

Model B: density plot comparing the density of the response to the densities of 10



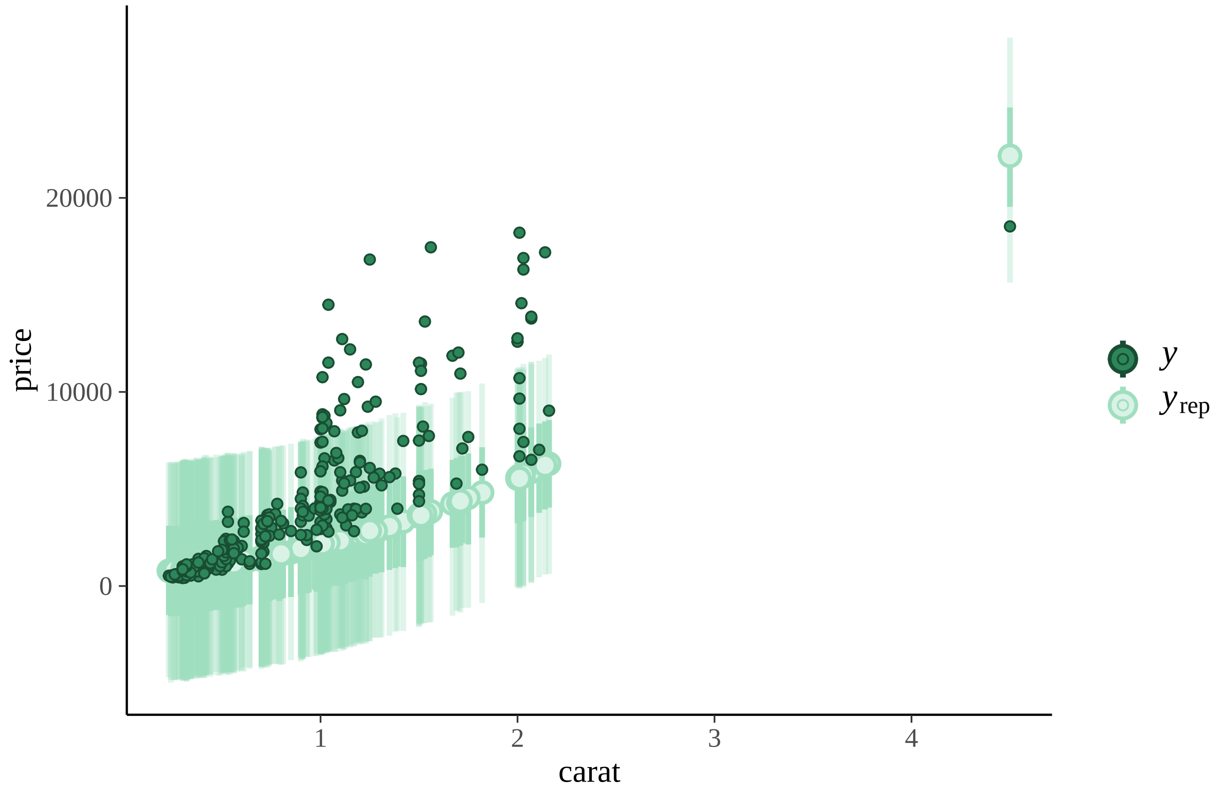
```
ppc_error_scatter_avg(y, yrep = yrep) + ggtitle("Model B: scatter plot comparing the response to the av
```

Model B: scatter plot comparing the response to the average predictive errc

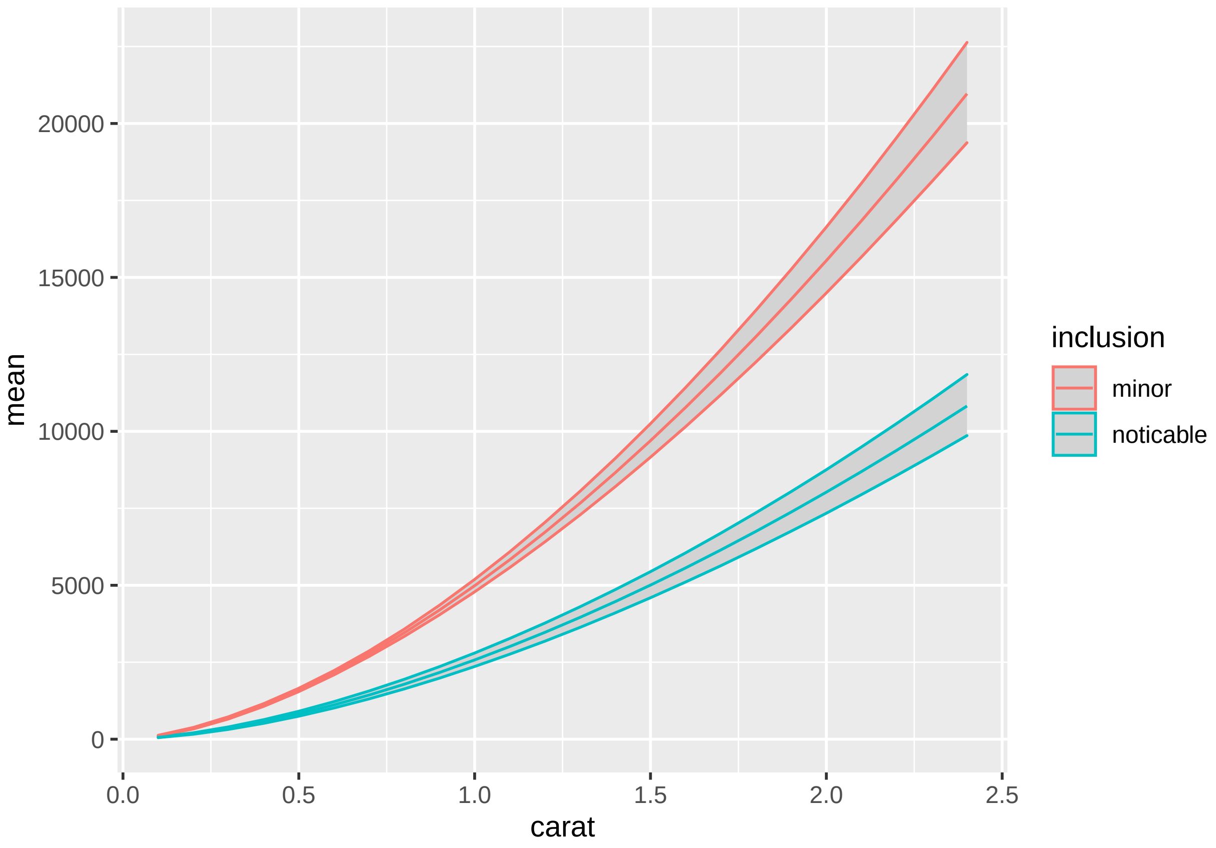


```
ppc_intervals(y = y, yrep = yrep, x = df$carat) + ggtitle("Model B: Posterior predictive intervals to tl
```

Model B: Posterior predictive intervals to the observed data values



```
sim_cond_mu_1 = rstan::extract(r_fit_D)$cond_mu_1
df_sim_cond_mu_1 = data.frame(
  carat = seq(0.1, 2.4, by=0.1),
  mean = apply(sim_cond_mu_1, 2, mean),
  lo0025 = apply(sim_cond_mu_1, 2, quantile, 0.025),
  hi0975 = apply(sim_cond_mu_1, 2, quantile, 0.975),
  inclusion = "noticable"
)

sim_cond_mu_0 = rstan::extract(r_fit_D)$cond_mu_0
df_sim_cond_mu_0 = data.frame(
  carat = seq(0.1, 2.4, by=0.1),
  mean = apply(sim_cond_mu_0, 2, mean),
  lo0025 = apply(sim_cond_mu_0, 2, quantile, 0.025),
  hi0975 = apply(sim_cond_mu_0, 2, quantile, 0.975),
  inclusion = "minor"
)

df_sim = rbind(df_sim_cond_mu_0, df_sim_cond_mu_1)
rm(df_sim_cond_mu_0, df_sim_cond_mu_1)

ggplot(df_sim,
       aes(x = carat,
           y = mean, colour = inclusion))+
  geom_ribbon(aes(ymin = lo0025,
                  ymax = hi0975),
              fill = "lightgrey") +
  geom_line()
```



```
# samples = extract(r_fit_D)
# ncycles = length(samples[[1]])
# T <- length(df$price)
# #each row of yrep is a sample from the pp distribution
# yrep = matrix(0, ncol = T, nrow = ncycles)
# for (i in seq_len(T)) {
#   mui = samples$beta0 + samples$beta1 * df$lcarat[i] +
#     samples$beta2 * as.integer(df$inclusions[i]=="noticeable")
#   yrep[, i] = rlnorm(ncycles, mean = mui, sd = sqrt(samples$sigmaSQ))
# }
# df$notic = as.integer(df$inclusions == "noticeable")
# color_scheme_set("blue")
# ppc_intervals_grouped(y = df$price, yrep = yrep, x = df$carat, prob = 0.5, prob_outer = .95, group =
# #
#
# find_ep = function(yrep, min = 0.1, max = 2.4, step=0.1) {
#   ep = array(0, dim=c((max-min+step)/step, 5), dimnames = list(c(), c("carat_min", "carat_max", "2.5%
#   n=0
#   for (i in seq(min+step, max, by=step)) {
#     n=n+1
#     subset = ((df$carat>(i-step)) & (df$carat<=i))
#     yrep_subset = yrep[,subset]
#     ep[n,1] = i-step
#     ep[n,2] = i
#     q = quantile(yrep_subset, probs = c(0.025, 0.975))
```

```r
#     ep[n,3] = q[[1]]
#     ep[n,4] = mean(yrep_subset)
#     ep[n,5] = q[[2]]
#   }
# return(ep)
# }
# find_ep(yrep)
# temp = ppc_intervals_data(y = df$price, yrep = yrep, x = df$lcarat, prob = 0.5, prob_outer = .95, gro
```

```r
cat("STAN Model E WAIC", waic_loo(r_fit_E)[1])
```

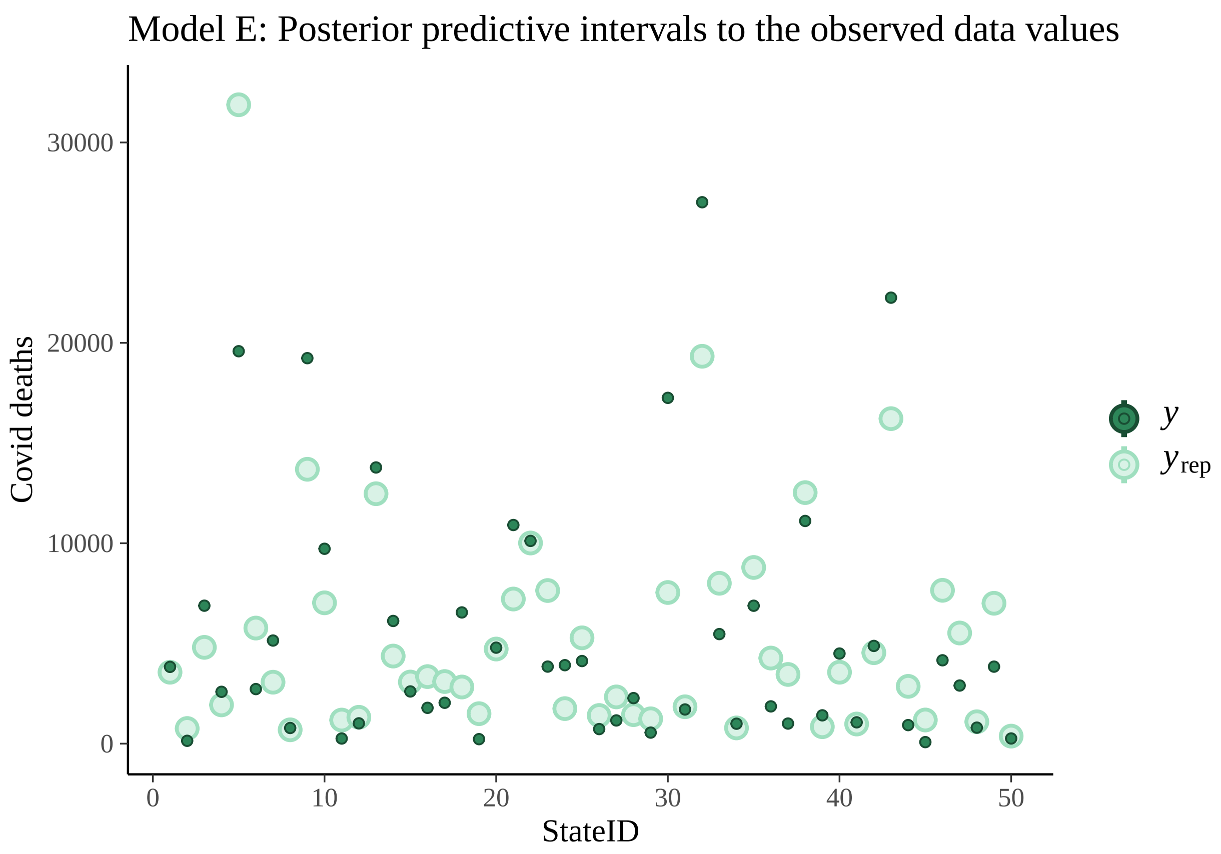
```
## Warning:
## 46 (92.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.

## Warning: Accessing waic using '$' is deprecated and will be removed in a
## future release. Please extract the waic estimate from the 'estimates' component
## instead.

## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.

## Warning: Accessing looic using '$' is deprecated and will be removed in a
## future release. Please extract the looic estimate from the 'estimates' component
## instead.

## STAN Model E WAIC 68766.75
```

```r
cat("STAN Model E LOOIC", waic_loo(r_fit_E)[2])
```

```
## Warning:
## 46 (92.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.

## Warning: Accessing waic using '$' is deprecated and will be removed in a
## future release. Please extract the waic estimate from the 'estimates' component
## instead.

## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.

## Warning: Accessing looic using '$' is deprecated and will be removed in a
## future release. Please extract the looic estimate from the 'estimates' component
## instead.

## STAN Model E LOOIC 61810.17
```

```r
setwd('/Users/AM/Documents/_CU Masters/2020 fall Bayesian_7393/code/Bayesian_Statistics_Class_Code/Exam
r_fit_E = readRDS(file = "Model_E_2020-12-07.rda")

samples = extract(r_fit_E)
ncycles = length(samples[[1]])
T <- length(df_covid$bs)

# each row of yrep is a sample from the pp distribution
yrep = matrix(0, ncol = T, nrow = ncycles)
for (i in seq_len(T)) {
  log_l = log(df_covid$population[i]) +
    samples$beta0 +
    samples$beta1 * df_covid$income[i] +
    samples$beta2 * df_covid$bs[i]
  yrep[, i] = rpois(ncycles, lambda = exp(log_l))
}
```

```
ppc_intervals(y = df_covid$deaths, yrep = yrep, prob = 0.5, prob_outer = .95) +
  ggtitle("Model E: Posterior predictive intervals to the observed data values") +
  xlab("StateID") + ylab("Covid deaths")
```



```
dif = ppc_intervals_data(y = df_covid$deaths, yrep = yrep, prob = 0.5, prob_outer = .95)
dif$ratio = abs(dif$y_obs - dif$m)/ dif$y_obs
cat("the worst prediction seems to be for StateID", which.max(dif$ratio),
    df_covid$state_name[which.max(dif$ratio)])
```

```
## the worst prediction seems to be for StateID 45 Vermont
```

```
dif$abs = abs(dif$y_obs - dif$m)
cat("the worst prediction seems to be for StateID", which.max(dif$abs),
    df_covid$state_name[which.max(dif$abs)])
```

```
## the worst prediction seems to be for StateID 5 California
```

```
-0.000054 * 30697
```

```
## [1] -1.657638
```

```
(24191 - 39682) * -0.000054
```

```
## [1] 0.836514
```

```
(17.50 - 39.00) * 0.060348
```

```
## [1] -1.297482
```

```
stan_mod_F = "
data {
  int T;
  int deaths[T]; // COVID-19 deaths
  vector[T] log_pop;    // log of U.S. states population,
  vector[T] income; // median income (USD)
  vector[T] bs;     // percentage of the population with bachelor's degrees.
}

parameters {
  real beta0;
  real beta1;
  real beta2;
  real phi;
}
transformed parameters {
  real eta[T];
  for (i in 1:T) {
   eta[i] = log_pop[i] + beta0 + beta1 * income[i] + beta2 * bs[i];
  }

}

model {
  beta0 ~ normal(0, 5);
  beta1 ~ normal(0, 5);
  beta2 ~ normal(0, 5);
  phi ~ gamma(0.01, 0.01);

  for (i in 1:T) {
    deaths[i] ~ neg_binomial_2_log(eta[i], phi);
  }
}
generated quantities {
  vector[T] log_lik;

  for (i in 1:T) {
    log_lik[i] = neg_binomial_2_log_lpmf(deaths[i] | eta[i],  phi);
  }

}

"

T <- length(df_covid$bs)
set.seed(95)

model_name = "Model_F_"
stan_dat_F = list(T = T, deaths = df_covid$deaths, log_pop = log(df_covid$population),
                  income = df_covid$income, bs = df_covid$bs)
r_fit_F = stan(model_code = stan_mod_F, data = stan_dat_F,
             iter = 5000, chains = 2,
             control = list(max_treedepth = 21))
```

```
## Trying to compile a simple C file
## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## clang -mmacosx-version-min=10.13 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/inclu
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/inclu
## /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/src/Core/util/
## namespace Eigen {
## ^
## /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/src/Core/util/
## namespace Eigen {
##                ^
##                ;
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/inclu
## /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/Core:96:10: f
## #include <complex>
##          ^~~~~~~~~
## 3 errors generated.
## make: *** [foo.o] Error 1
##
## SAMPLING FOR MODEL '95322de15106226da123bef5b1199d71' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.7e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.27 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 1: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 1: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 1: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 139.651 seconds (Warm-up)
## Chain 1:                39.5936 seconds (Sampling)
## Chain 1:                179.244 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '95322de15106226da123bef5b1199d71' NOW (CHAIN 2).
## Chain 2: Rejecting initial value:
## Chain 2:   Error evaluating the log probability at the initial value.
## Chain 2: Exception: neg_binomial_2_log_lpmf: Precision parameter is -1.43325, but must be > 0!  (in
##
## Chain 2:
## Chain 2: Gradient evaluation took 1.4e-05 seconds
```

```
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.14 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 2: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 2: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 2: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 2: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 2: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 2: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 2: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 2: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 2: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 2: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 2: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 120.099 seconds (Warm-up)
## Chain 2:                38.6467 seconds (Sampling)
## Chain 2:                158.746 seconds (Total)
## Chain 2:
```

```
summary(r_fit_F, pars = c("beta0", "beta1", "beta2", "phi"), prob = c(0.025, 0.975))$summary
```

```
##               mean      se_mean           sd         2.5%         97.5%
## beta0 -4.352149141 1.716854e-02 7.685934e-01 -5.789551446 -2.757540e+00
## beta1 -0.000044994 1.199203e-06 5.047454e-05 -0.000144384  5.430147e-05
## beta2  0.039237130 8.639763e-04 3.806285e-02 -0.035612120  1.139286e-01
## phi    2.529451696 1.211503e-02 4.869649e-01  1.685299747  3.591891e+00
##          n_eff     Rhat
## beta0 2004.133 1.002128
## beta1 1771.574 1.001698
## beta2 1940.881 1.000783
## phi   1615.646 1.001139
```

```
#sso <- launch_shinystan(r_fit_F)
file_name = paste(model_name, as.character(Sys.Date()), ".rda", sep="")
setwd('/Users/AM/Documents/_CU Masters/2020 fall Bayesian_7393/code/Bayesian_Statistics_Class_Code/Exam
```

```
cat("STAN Model F WAIC", waic_loo(r_fit_F)[1])
```

```
## Warning:
## 2 (4.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.

## Warning: Accessing waic using '$' is deprecated and will be removed in a
## future release. Please extract the waic estimate from the 'estimates' component
## instead.

## Warning: Accessing looic using '$' is deprecated and will be removed in a
## future release. Please extract the looic estimate from the 'estimates' component
## instead.

## STAN Model F WAIC 895.8925
```

```
cat("STAN Model F LOOIC", waic_loo(r_fit_F)[2])
```

```
## Warning:
## 2 (4.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.
## Warning: Accessing waic using '$' is deprecated and will be removed in a
## future release. Please extract the waic estimate from the 'estimates' component
## instead.
```

```
## Warning: Accessing looic using '$' is deprecated and will be removed in a
## future release. Please extract the looic estimate from the 'estimates' component
## instead.
```

```
## STAN Model F LOOIC 896.0408
```