Applied Bayesian modeling - HW5

Score: Each question is worth 10 points. The maximum number of points in this HW is 40 points, with 10 points extra credit. For calculating a final HW grade, the points will be rescaled to a maximum score of (50)/40*100% = 125%.

In this homework, we will consider elements of a Bayesian workflow. The basis of this HW is the excellent blog post by Monica Alexander on this topic:
https://www.monicaalexander.com/posts/2020-28-02-bayes_viz/

We use the same dataset, repeat some of steps, and add some additional ones.

You can choose whether you answer the questions using functionality from the brms or rstan package. You may find the code from Monica's blog post, example code below, and/or code from module 11 helpful. In addition, added below is some example code using brm.

```
library(tidyverse)
library(bayesplot)
library(loo)
library(tidybayes)
library(here)

# consider using
library(brms)
# or
#library(rstan)
#options(mc.cores = parallel::detectCores())
#rstan_options(auto_write = TRUE)
```

Data

The dataset is a 0.1% sample of all births in the US in 2017 from NBER, details on how this was constructed is here https://github.com/MJAlexander/states-mortality/blob/master/birthweight_bayes_viz/births_2017_prep.R For UMass students, this data set is uploaded in https://drive.google.com/drive/folders/1XX_wfHa89E-0quq3LTmxQ03idkpHTRwP.

```
ds <- read_rds(here("../data","births_2017_sample.RDS"))
head(ds)
```

```
## # A tibble: 6 x 8
##   mager mracehisp meduc   bmi sex   combgest  dbwt ilive
##   <dbl>     <dbl> <dbl> <dbl> <chr>    <dbl> <dbl> <chr>
## 1    21         7     3  21   F           40  3.53 Y
## 2    19         7     3  46.3 M           39  3.64 Y
## 3    35         4     6  19.6 M           39  2.86 Y
## 4    27         1     4  26.6 M           40  4.37 Y
## 5    27         2     2  22.2 M           37  2.61 Y
## 6    19         3     3  18.8 M           40  2.86 Y
```

Brief overview of variables:

- `mager` mum's age
- `mracehisp` mum's race/ethnicity see here for codes: https://data.nber.org/natality/2017/natl2017.pdf page 15
- `meduc` mum's education see here for codes: https://data.nber.org/natality/2017/natl2017.pdf page 16
- `bmi` mum's bmi
- `sex` baby's sex
- `combgest` gestational age in weeks
- `dbwt` birth weight in kg
- `ilive` alive at time of report y/n/ unsure

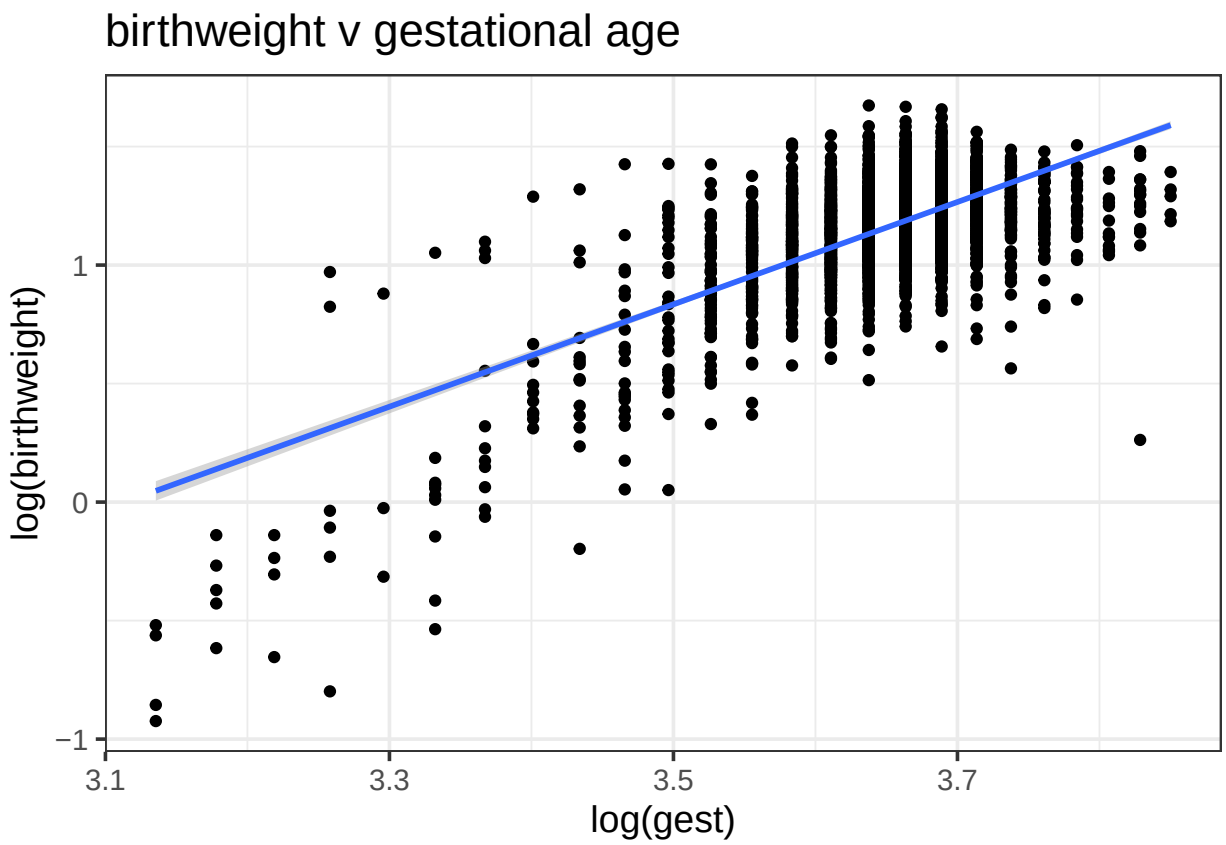
Code below is to rename some variables, remove any observations with missing gestational age or birth weight, restrict just to babies that were alive, and make a preterm variable based on gestational age being less than 32 weeks.

```
ds <- ds %>%
  rename(birthweight = dbwt, gest = combgest) %>%
  mutate(preterm = ifelse(gest<32, "Y", "N")) %>%
  filter(ilive=="Y",gest< 99, birthweight<9.999)
```

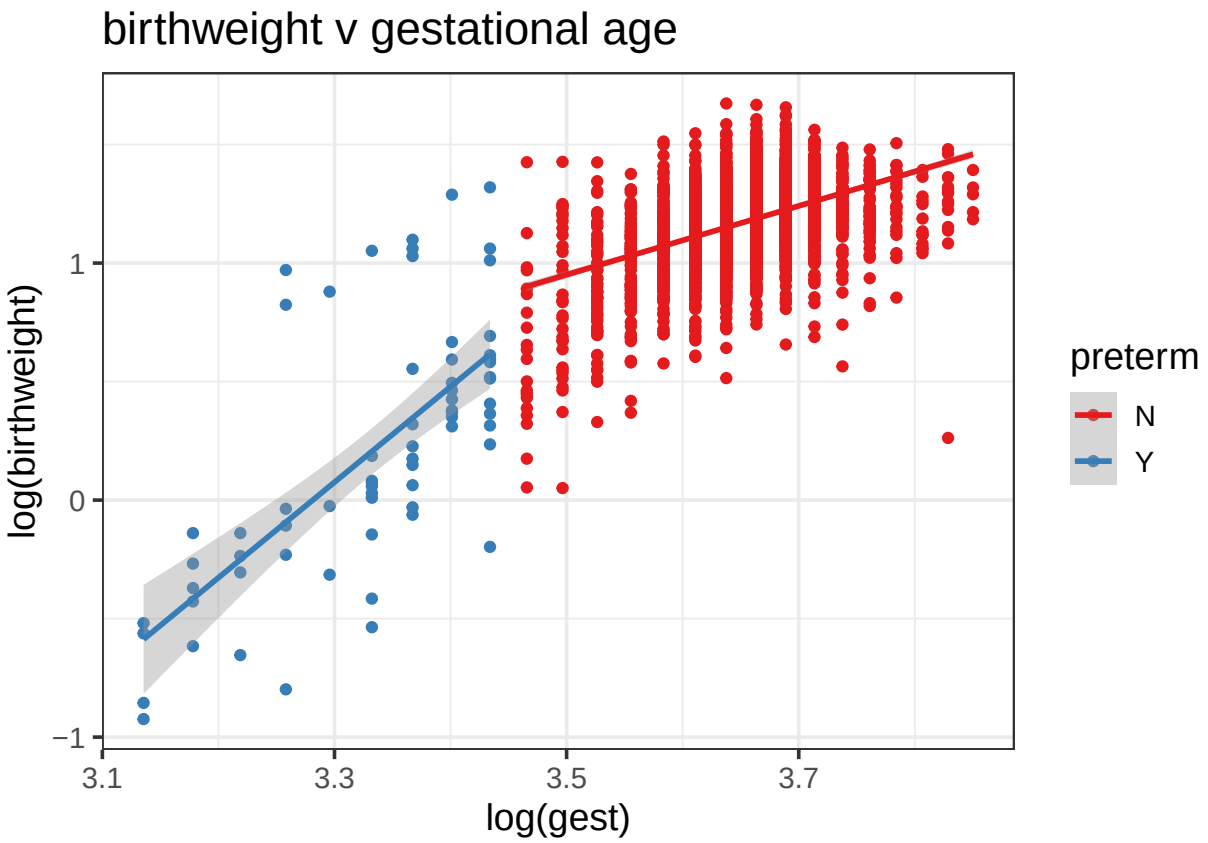
Some EDA

Some plots, taken from Monica's blog

```
ds %>%
  ggplot(aes(log(gest), log(birthweight))) +
  geom_point() + geom_smooth(method = "lm") +
  scale_color_brewer(palette = "Set1") +
  theme_bw(base_size = 14) +
  ggtitle("birthweight v gestational age")
```




```
ds %>%
  ggplot(aes(log(gest), log(birthweight), color = preterm)) +
  geom_point() + geom_smooth(method = "lm") +
  scale_color_brewer(palette = "Set1") +
  theme_bw(base_size = 14) +
  ggtitle("birthweight v gestational age")
```



#

Models We consider the following 2 models:

Model 1 has log birth weight as a function of log gestational age

$$y_i|\beta_0, \beta_1, \sigma \sim N(\beta_0 + \beta_1 x_i, \sigma^2)$$

Model 2 has an interaction term between gestation and prematurity

$$y_i|\beta_0, \beta_1, \beta_2, \beta_3, \sigma \sim N(\beta_0 + \beta_1 x_i + \beta_2 z_i + \beta_3 x_i z_i, \sigma^2)$$

- y_i is log-transformed weight in kg
- x_i is log-gestational age in weeks, CENTERED AND STANDARDIZED, i.e. x_i equal log(gestational age), minus its mean and divided by its standard deviation
- z_i is preterm (0 or 1, if gestational age is less than 32 weeks)

For model fitting, we introduce the variables on the log-scale, and center and standardize the covariate:

```
ds$log_weight <- log(ds$birthweight)
ds$log_gest <- (log(ds$gest)/sd(log(ds$gest)))
ds$log_gest_c <- (log(ds$gest) - mean(log(ds$gest)))/sd(log(ds$gest))
```

Prior predictive checks using brm

Monica's blog includes code in R to carry out a prior predictive check. The code here outlines how to do this using brm.

Setting priors in brm, and a note of caution regarding the intercept

Firstly, to set our own priors, we need to specify priors for usage in the brm-fit. The argument "prior" can be used for that when calling the brm function. Let's start by checking the default priors in brm. For model 1, we have equation

$$E(y_i|\beta_0, \beta_1) = \beta_0 + \beta_1 \cdot x_i,$$

where x_i refers to log-transformed gestational age, which is centered (we subtracted the mean) and standardized (we divided by its standard deviation). This shows the default priors for model 1 used in brm:

```
get_prior(log_weight~log_gest_c, data = ds)
```

```
##                   prior     class       coef group resp dpar nlpar lb ub
##                  (flat)         b
##                  (flat)         b log_gest_c
##  student_t(3, 1.2, 2.5) Intercept
##    student_t(3, 0, 2.5)     sigma                                     0
##        source
##       default
##  (vectorized)
##       default
##       default
```

The overview states that for regression coefficients (class = b), flat priors are used. In our model, we have one coefficient given by log_gest_c. A student_t(3, 0, 2.5) prior is used for sigma with a lower bound (lb) of 0.

The prior used for the intercept needs some more explanation, to avoid confusion in more complicated models. The overview states that a student_t(3, 1.2, 2.5) is used for "Intercept", which I'll refer to as the brm-Intercept. Using the default formula as we did above, the brm-Intercept parameter refers to the intercept of a model with centered covariates, i.e. for a general formula y ~ x, the brm-Intercept refers to α in $E(y_i|\alpha, \beta_1) = \alpha + \beta_1(x_i - \bar{x})$. This is also clear when checking the stan-model as we did in module 10, see below as well:

```
mod_tmp <- brm(log_weight~ log_gest_c, data = ds,
   chains = 2,
    iter = 20, # silly short run, just to get the stan model
    cores = getOption("mc.cores", 2),
     file = "output/hw5_fit_tmp",
   file_refit = "on_change")
```

The stan model explain that b_Intercept is β_0, while Intercept refers to a temporary intercept for centered predictors.

```
stancode(mod_tmp)
```

```
## // generated with brms 2.17.0
## functions {
## }
## data {
##   int<lower=1> N;  // total number of observations
##   vector[N] Y;  // response variable
##   int<lower=1> K;  // number of population-level effects
```

```
##   matrix[N, K] X;  // population-level design matrix
##   int prior_only;  // should the likelihood be ignored?
## }
## transformed data {
##   int Kc = K - 1;
##   matrix[N, Kc] Xc;  // centered version of X without an intercept
##   vector[Kc] means_X;  // column means of X before centering
##   for (i in 2:K) {
##     means_X[i - 1] = mean(X[, i]);
##     Xc[, i - 1] = X[, i] - means_X[i - 1];
##   }
## }
## parameters {
##   vector[Kc] b;  // population-level effects
##   real Intercept;  // temporary intercept for centered predictors
##   real<lower=0> sigma;  // dispersion parameter
## }
## transformed parameters {
##   real lprior = 0;  // prior contributions to the log posterior
##   lprior += student_t_lpdf(Intercept | 3, 1.2, 2.5);
##   lprior += student_t_lpdf(sigma | 3, 0, 2.5)
##     - 1 * student_t_lccdf(0 | 3, 0, 2.5);
## }
## model {
##   // likelihood including constants
##   if (!prior_only) {
##     target += normal_id_glm_lpdf(Y | Xc, Intercept, b, sigma);
##   }
##   // priors including constants
##   target += lprior;
## }
## generated quantities {
##   // actual population-level intercept
##   real b_Intercept = Intercept - dot_product(means_X, b);
## }
```

Note that the reason for brm to use this parametrization is computational efficiency, it still fits the same model and produces an estimate of the original intercept β_0 (b_Intercept) in the summary output. Also note that in our model, because our covariate is centered, we DO get that $\alpha = \beta_0$. However, if x is NOT centered (or if you add additional covariates that are not centered like in model 2), then brm-Intercept α is not equal to the intercept β_0 from a model with uncentered covariates.

The good news: If you would like to specify specific priors on β_0 (as opposed to α), and work with samples from the prior and posterior on β_0, for a general model with $E(y_i|\boldsymbol{\beta}) = \beta_0 + \sum_{k=1}^{K} \beta_k x_{i,k}$, you can still use brm, you just need to change the formula that you use. To be able to specify a prior on the intercept β_0, we can use the formula y ~ 0 + Intercept + covariates, e.g. for our model 1, we can use:

```
get_prior(log_weight ~ 0 + Intercept + log_gest_c, data = ds)
```

```
##                 prior class       coef group resp dpar nlpar lb ub       source
##                (flat)     b                                            default
##                (flat)     b  Intercept                              (vectorized)
##                (flat)     b log_gest_c                              (vectorized)
##  student_t(3, 0, 2.5) sigma                                    0         default
```

When using this formula, the parameter Intercept now does refer to β_0, regardless of whether or not the covariates were centered. What happens internally is that we tell brm to NOT use an intercept term but instead, to use a covariate with 1s, with its coefficient labeled "Intercept". Hence "Intercept" is considered a regression coefficient. To show this in the stan model as well:

```
mod_tmp2 <- brm(log_weight~ 0 + Intercept + log_gest_c, data = ds,
   chains = 2,
    iter = 20, # silly short run, just to get the stan model
    cores = getOption("mc.cores", 2),
     file = "output/hw5_fit_tmp2",
   file_refit = "on_change")
```

The stan model shows that we only have regression coefficients:

```
stancode(mod_tmp2)
```

```
## // generated with brms 2.17.0
## functions {
## }
## data {
##   int<lower=1> N;  // total number of observations
##   vector[N] Y;  // response variable
##   int<lower=1> K;  // number of population-level effects
##   matrix[N, K] X;  // population-level design matrix
##   int prior_only;  // should the likelihood be ignored?
## }
## transformed data {
## }
## parameters {
##   vector[K] b;  // population-level effects
##   real<lower=0> sigma;  // dispersion parameter
## }
## transformed parameters {
##   real lprior = 0;  // prior contributions to the log posterior
##   lprior += student_t_lpdf(sigma | 3, 0, 2.5)
##     - 1 * student_t_lccdf(0 | 3, 0, 2.5);
## }
## model {
##   // likelihood including constants
##   if (!prior_only) {
##     target += normal_id_glm_lpdf(Y | X, 0, b, sigma);
##   }
##   // priors including constants
##   target += lprior;
## }
## generated quantities {
## }
```

And the vector "Intercept" that went into the design matrix X has just 1s:

```
mod_tmp2$data$Intercept[1:5] # showing first 5 entries
```

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

With that explanation out of the way, how do we set prior for brm-fits? We use the function set_prior for that, see ?set_prior for further information. In summary, the arguments of set_prior are the name of the parameter and the densities that you'd like to use. The exact specification follows the information printed by get_prior. Here is an example, for the model with formula y ~ 0 + Intercept + covariates, that includes a prior for Intercept β_0:

```
prior1 <- c(set_prior("normal(0,100)", class = "b", coef = "Intercept"),
            # for formula where the Intercept is considered a regression coefficient
            set_prior("normal(0,100)", class = "b"), # for any other regression coefficients
            set_prior("student_t(3,0, 2.5)", lb = 0, class = "sigma")) # SD, note the lower bound at 0
prior1
```

```
##                 prior class      coef group resp dpar nlpar   lb   ub source
##         normal(0,100)     b Intercept                       <NA> <NA>   user
##         normal(0,100)     b                                 <NA> <NA>   user
##  student_t(3,0, 2.5) sigma                                    0 <NA>   user
```

Here is a fit using these priors:

```
mod1_prior1 <- brm(log_weight~ 0 + Intercept + log_gest_c, data = ds,
   seed = 1236,
   prior = prior1,
   chains = 4,
    iter = 2000, thin = 1,
    cores = getOption("mc.cores", 4),
     file = "output/hw5_fit1_prior1",
   file_refit = "on_change")
```

```
summary(mod1_prior1)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: log_weight ~ 0 + Intercept + log_gest_c
##    Data: ds (Number of observations: 3840)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Population-Level Effects:
##            Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept      1.16      0.00     1.16     1.17 1.00     5311     3093
## log_gest_c     0.15      0.00     0.14     0.15 1.00     4924     2930
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.17      0.00     0.16     0.17 1.00     2052     2532
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

See stan model

```
stancode(mod1_prior1)
```

```
## // generated with brms 2.17.0
## functions {
## }
## data {
##   int<lower=1> N;  // total number of observations
##   vector[N] Y;  // response variable
##   int<lower=1> K;  // number of population-level effects
##   matrix[N, K] X;  // population-level design matrix
##   int prior_only;  // should the likelihood be ignored?
## }
## transformed data {
## }
## parameters {
##   vector[K] b;  // population-level effects
##   real<lower=0> sigma;  // dispersion parameter
## }
## transformed parameters {
##   real lprior = 0;  // prior contributions to the log posterior
##   lprior += normal_lpdf(b[1] | 0,100);
##   lprior += normal_lpdf(b[2] | 0,100);
##   lprior += student_t_lpdf(sigma | 3,0, 2.5)
##     - 1 * student_t_lccdf(0 | 3,0, 2.5);
## }
## model {
##   // likelihood including constants
##   if (!prior_only) {
##     target += normal_id_glm_lpdf(Y | X, 0, b, sigma);
##   }
##   // priors including constants
##   target += lprior;
## }
## generated quantities {
## }
```

At last: Prior predictive checks using brm

We can generate data sets based on these priors using the sample_prior argument, as follows:

```
mod1_priorpredict1 <- brm(log_weight ~ 0 + Intercept + log_gest_c, data = ds,
   seed = 1236,
   prior = prior1,
   sample_prior = "only", # we are drawing from the prior!
   chains = 4,
    iter = 2000, thin = 1,
    cores = getOption("mc.cores", 4),
     file = "output/hw5_fit1_priorpredict1",
   file_refit = "on_change")
```

Note that if you look at the summary now, you won't get a model fit (given that data are not used), instead, it will show a summary of the prior distributions:

```
summary(mod1_priorpredict1)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: log_weight ~ 0 + Intercept + log_gest_c
##    Data: ds (Number of observations: 3840)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Population-Level Effects:
##            Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     -0.38    103.38  -204.61   202.47 1.00     3642     2808
## log_gest_c    -1.86     99.67  -192.13   187.91 1.00     3360     2897
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     2.81      3.12     0.10    10.23 1.00     2897     1984
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

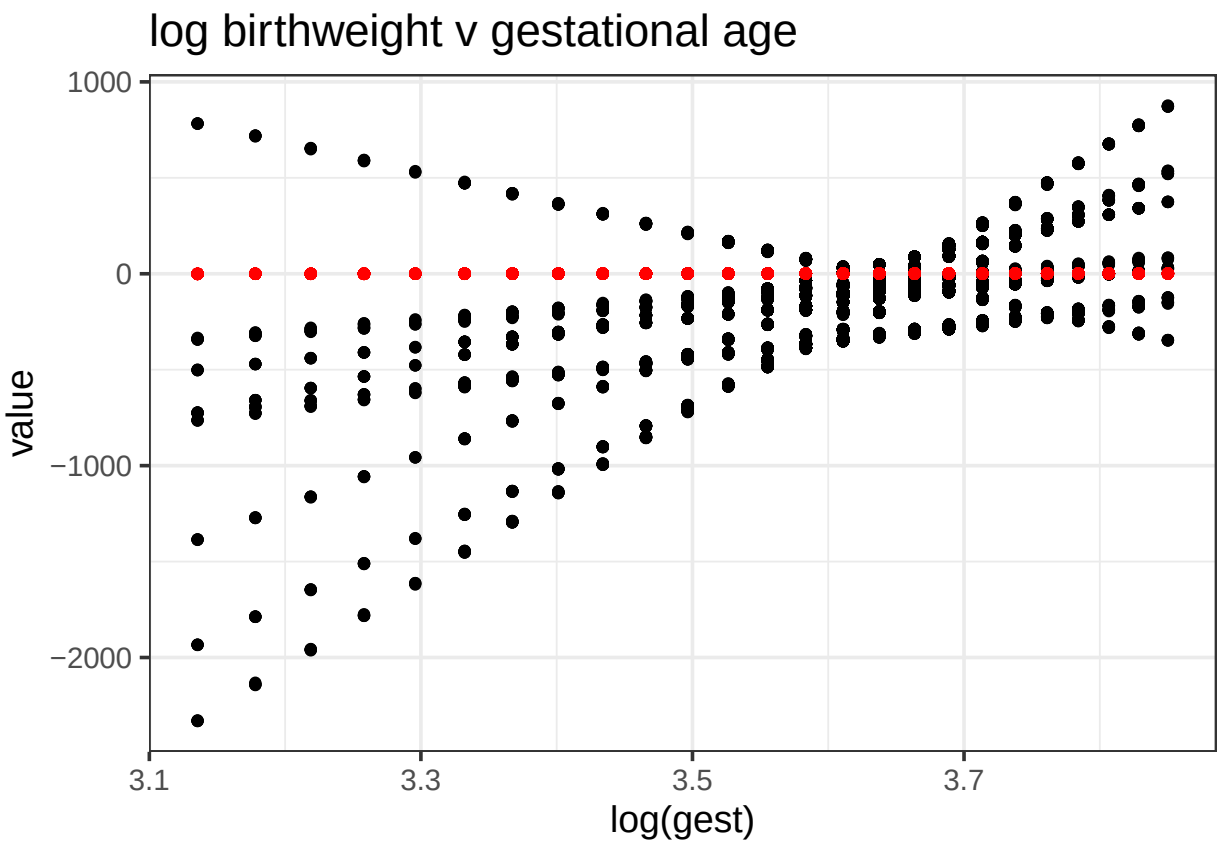
As before, we can generate data sets using the posterior_predict function

```
ynew_si <- posterior_predict(mod1_priorpredict1)
dim(ynew_si)
```

```
## [1] 4000 3840
```

We have generated 4000 data sets for log-transformed birth weights. Here I am just plotting the first 10 against gestational age:

```
bw_si <- as_tibble((t(ynew_si[1:10,])),.name_repair = "unique")
bw_si %>% mutate(gest = ds$gest, observed_data = log(ds$birthweight))%>%
  pivot_longer(-c(gest, observed_data)) %>%
  ggplot(aes(x = log(gest),y = value)) +
  geom_point() +
  geom_point(aes(y = observed_data), color = "red") +
  scale_color_brewer(palette = "Set1") +
  theme_bw(base_size = 14) +
  ggtitle("log birthweight v gestational age")
```



Same thing for weakly informative priors

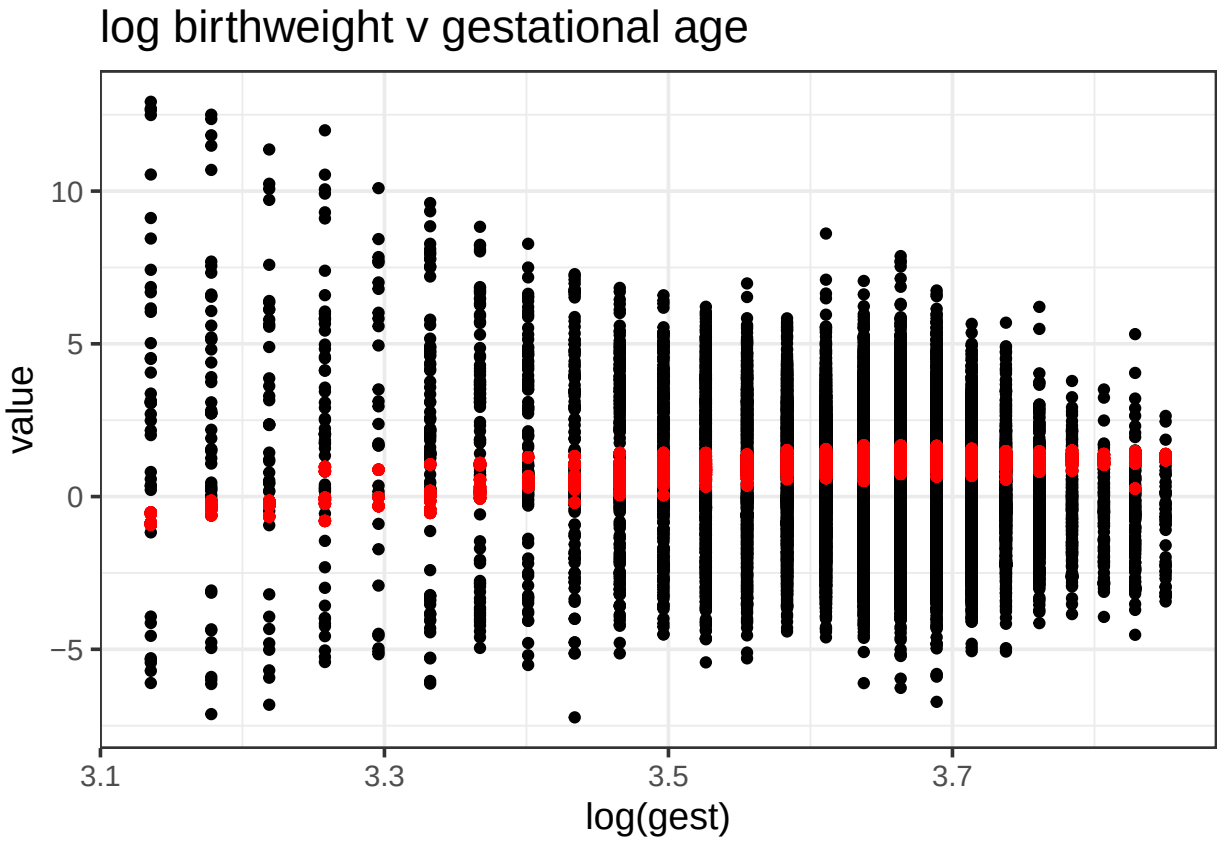
```
prior2 <- c(set_prior("normal(0,1)", class = "b", coef = "Intercept"),
            set_prior("normal(0,1)", class = "b"),
            set_prior("student_t(3,0, 1)", lb = 0, class = "sigma"))
prior2
```

```
##              prior class      coef group resp dpar nlpar   lb   ub source
##        normal(0,1)     b Intercept                       <NA> <NA>   user
##        normal(0,1)     b                                 <NA> <NA>   user
##  student_t(3,0, 1) sigma                                    0 <NA>   user
```

```
mod1_priorpredict2 <- brm(log_weight~ 0 + Intercept + log_gest_c, data = ds,
   seed = 1236,
   prior = prior2,
   sample_prior = "only", # we are drawing from the prior!
   chains = 4,
    iter = 2000, thin = 1,
    cores = getOption("mc.cores", 4),
     file = "output/hw5_fit1_priorpredict2",
   file_refit = "on_change")
```

```
ynew2_si <- posterior_predict(mod1_priorpredict2)
bw2_si <- as_tibble((t(ynew2_si[1:10,])),.name_repair = "unique")
bw2_si %>% mutate(gest = ds$gest, observed_data = log(ds$birthweight))%>%
  pivot_longer(-c(gest, observed_data)) %>%
  ggplot(aes(x = log(gest),y = value)) +
```

```
  geom_point() +
  geom_point(aes(y = observed_data), color = "red") +
  scale_color_brewer(palette = "Set1") +
  theme_bw(base_size = 14) +
  ggtitle("log birthweight v gestational age")
```

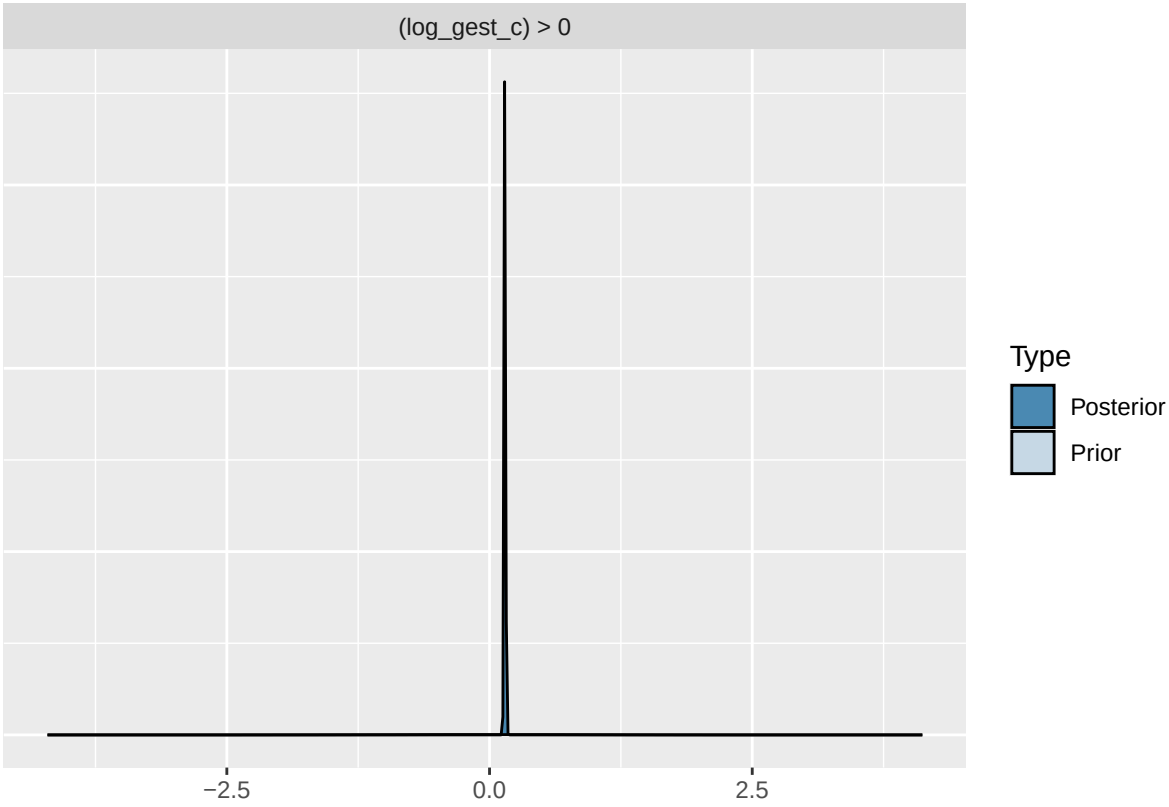


If we go ahead with prior set 2 and fit the model, we can make plots to compare prior and posterior densities for the model parameters for that fit. Brm has the option to also save samples from the priors, that can be used for those plots:

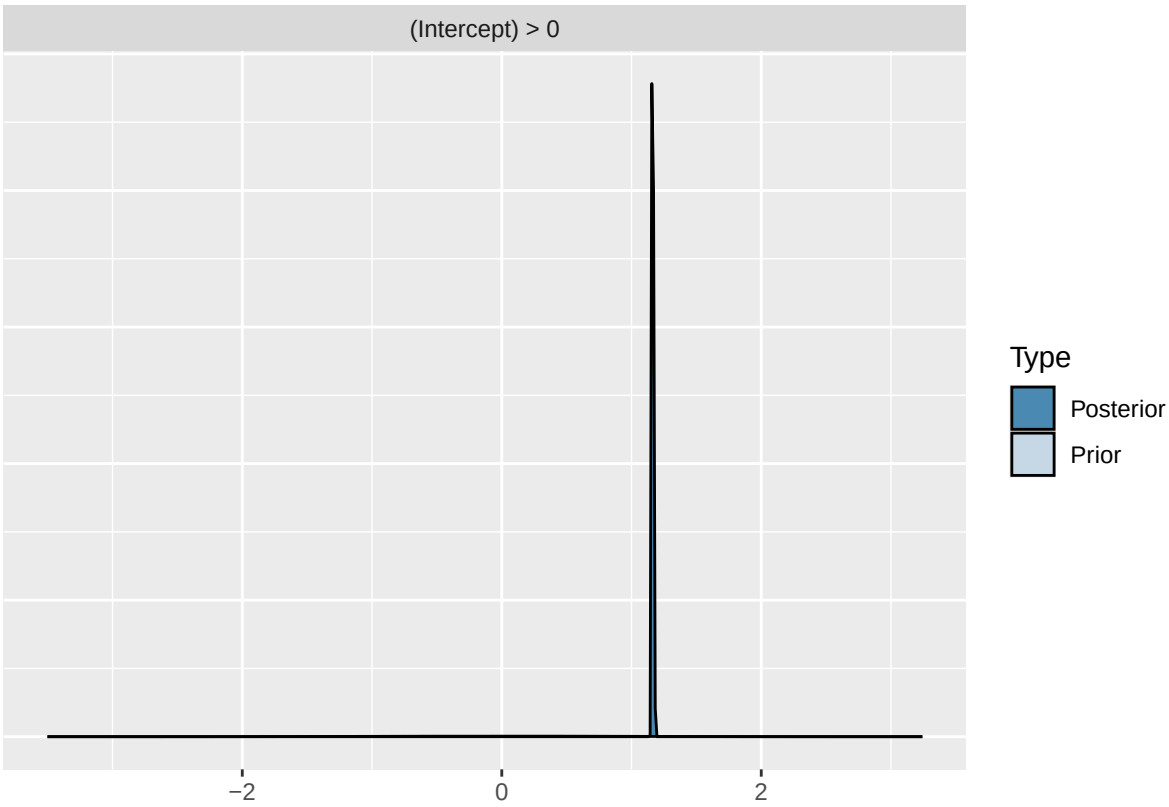
```
mod1_prior2 <- brm(log_weight ~ 0 + Intercept + log_gest_c, data = ds,
   seed = 1236,
   prior = prior2,
   sample_prior = "yes", # also sample from the priors
   chains = 4,
    iter = 2000, thin = 1,
    cores = getOption("mc.cores", 4),
     file = "output/hw5_fit1_prior2",
   file_refit = "on_change")
```

Compare priors and posteriors, note that this function works to get the plots without having to extract the samples:

```
plot(hypothesis(mod1_prior2, "log_gest_c > 0"))
```




```
plot(hypothesis(mod1_prior2, "Intercept > 0"))
```



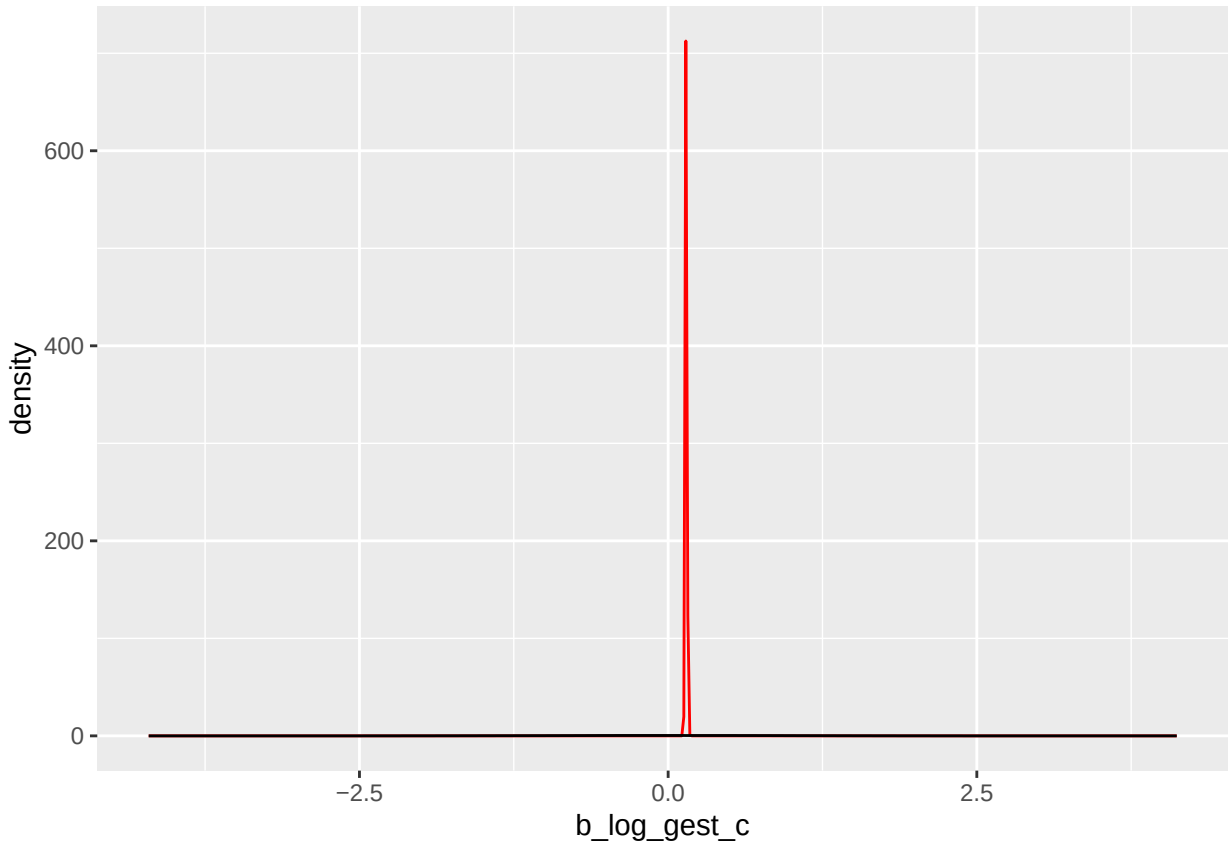
To work with the samples yourself, you can pull out the samples using these functions:

```
samps <- as_draws_matrix(mod1_prior2)
samps
```

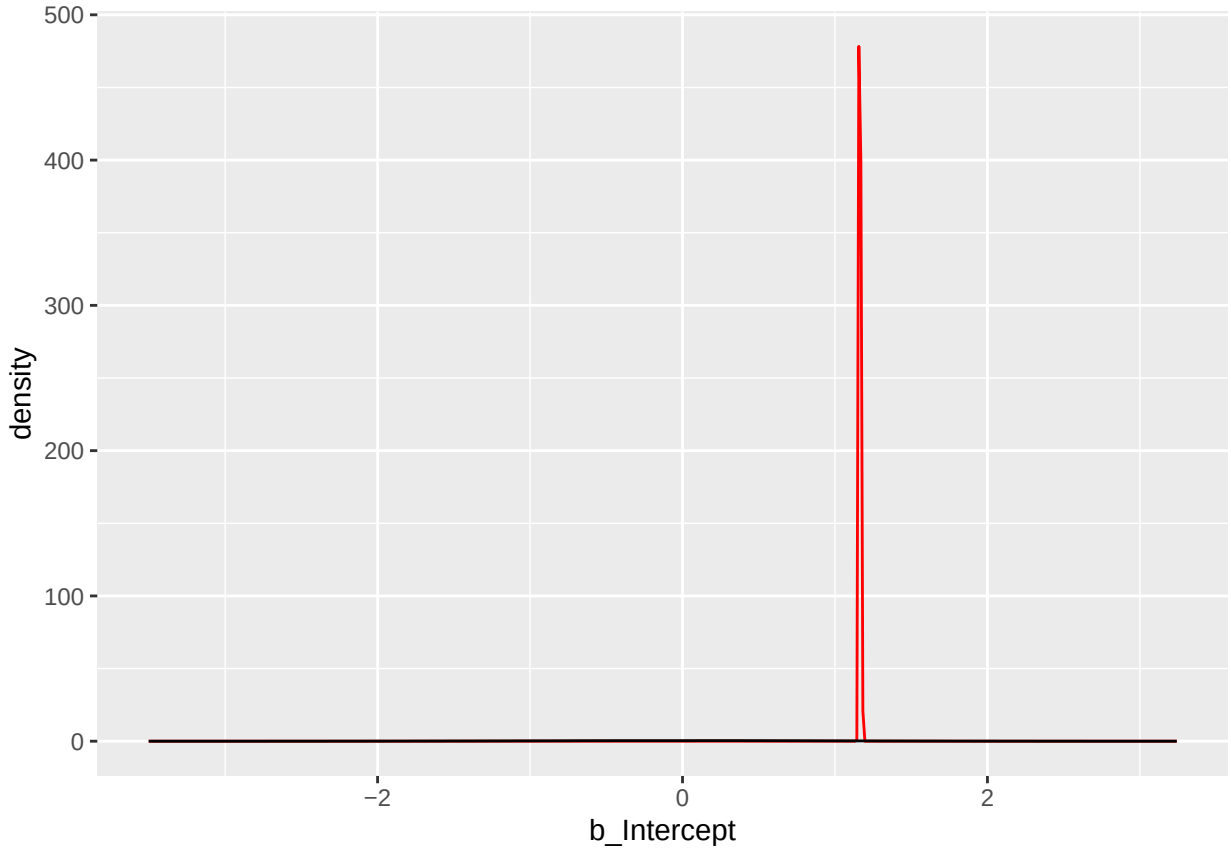
```
## # A draws_matrix: 1000 iterations, 4 chains, and 8 variables
##     variable
## draw b_Intercept b_log_gest_c sigma prior_b_Intercept prior_b_log_gest_c
##   1          1.2         0.15  0.17             1.630              -0.27
##   2          1.2         0.15  0.17             0.365               0.45
##   3          1.2         0.15  0.17             0.655              -0.51
##   4          1.2         0.15  0.17             0.701              -0.21
##   5          1.2         0.15  0.17             1.188               0.93
##   6          1.2         0.14  0.17            -0.705              -0.52
##   7          1.2         0.15  0.17            -1.076               0.43
##   8          1.2         0.14  0.17            -1.303               2.03
##   9          1.2         0.14  0.17            -0.022               1.18
##   10         1.2         0.15  0.17            -0.821               0.25
##     variable
## draw prior_sigma lprior lp__
##   1         0.46   -2.8 1405
##   2         1.90   -2.9 1405
##   3         0.48   -2.9 1406
##   4         0.25   -2.9 1406
##   5         1.02   -2.9 1406
##   6         0.82   -2.9 1405
##   7         1.06   -2.9 1405
##   8         0.18   -2.8 1405
##   9         0.82   -2.9 1406
##   10        0.50   -2.8 1406
## # ... with 3990 more draws
```

Now plot

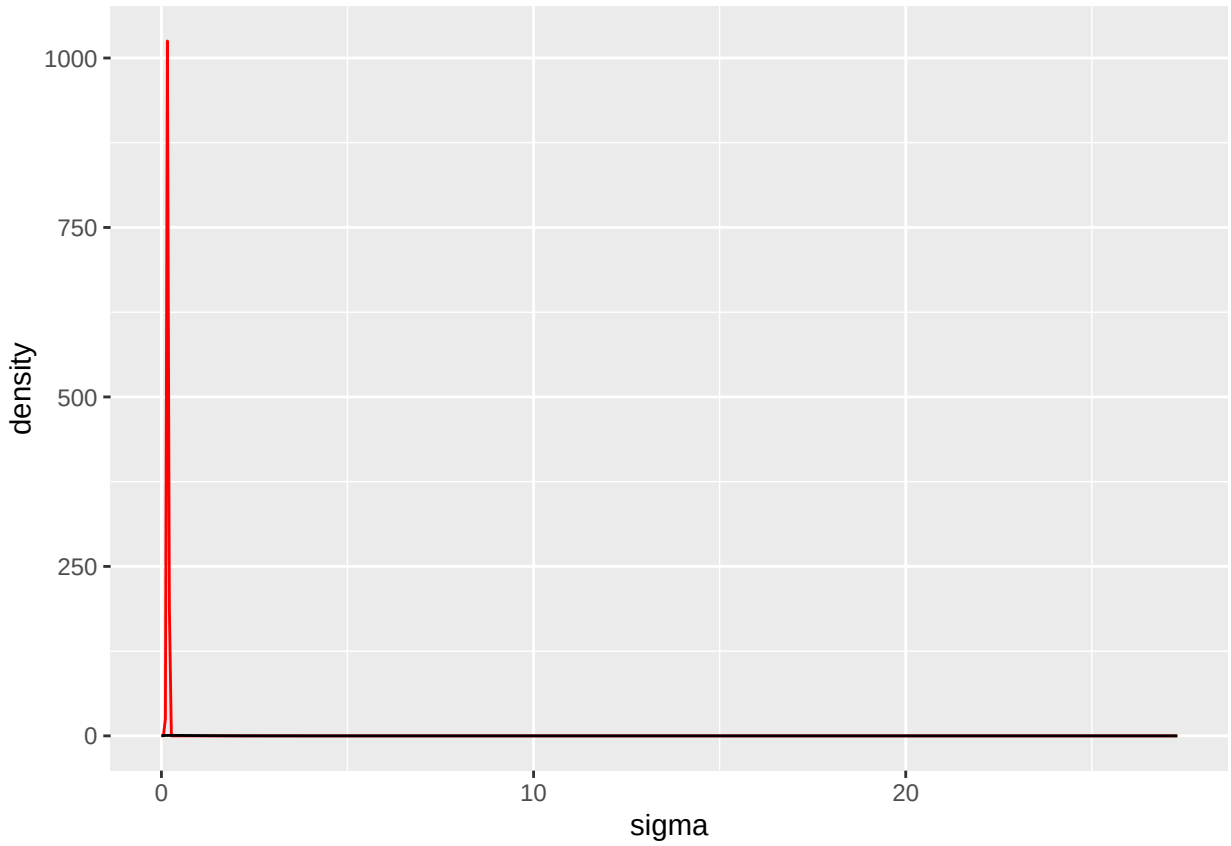
```
as_tibble(samps) %>%
  ggplot() +
  geom_density(aes(x = b_log_gest_c), color = "red") +
  geom_density(aes(x = prior_b_log_gest_c))
```




```
as_tibble(samps) %>%
  ggplot() +
  geom_density(aes(x = b_Intercept), color = "red") +
  geom_density(aes(x = prior_b_Intercept))
```




```
as_tibble(samps) %>%
  ggplot() +
  geom_density(aes(x = sigma), color = "red") +
  geom_density(aes(x = prior_sigma))
```



Question 1: prior predictive check for model 2

Consider model 2 as introduced above. Simulate data from the model using two sets of prior distributions, using prior set 1 with $\beta_k \sim N(0, 100)$, $\sigma \sim t_3^+(2.5)$ (student-t with 3 degrees of freedom and scale parameter 2.5, truncated to positive outcomes); and prior set 2 with $\beta_k \sim N(0, 1)$, $\sigma \sim t_3^+(1)$.

Produce outputs to illustrate how the prior predictive distributions of the data compare to the observed data and/or to assess whether the priors are reasonable in sets 1 and 2.

Solution

We can repeat the steps as per the intro above, to create prior predictive draws. Note that the priors are the same (because the assignment for the regression coefficient is the same) but repeating them here for completeness:

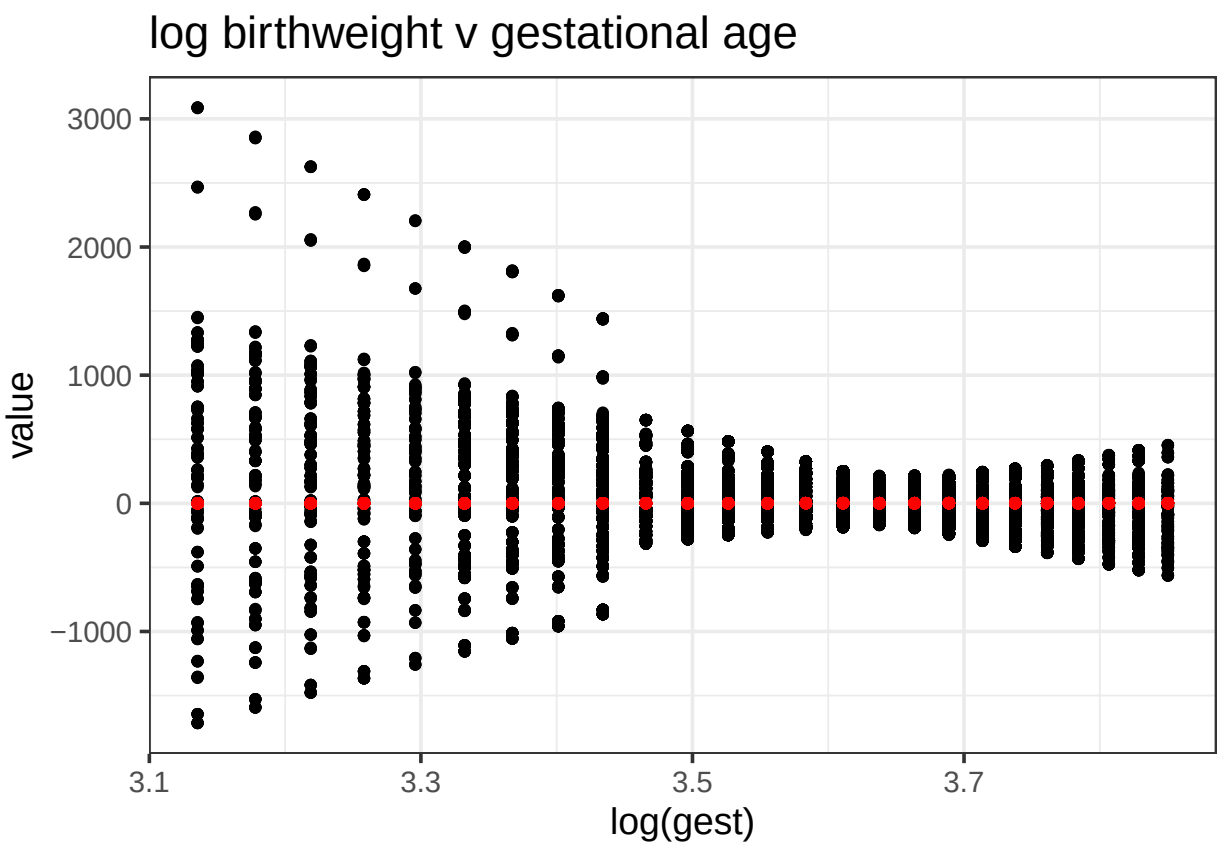
```
prior1 <- c(set_prior("normal(0,100)", class = "b", coef = "Intercept"),
            # for formula where the Intercept is considered a regression coefficient
            set_prior("normal(0,100)", class = "b"), # for any other regression coefficients
            set_prior("student_t(3,0, 2.5)", lb = 0, class = "sigma")) # SD, note the lower bound at 0


prior2 <- c(set_prior("normal(0,1)", class = "b", coef = "Intercept"),
            set_prior("normal(0,1)", class = "b"),
            set_prior("student_t(3,0, 1)", lb = 0, class = "sigma"))
```

```
mod2_priorpredict1 <- brm(log_weight ~ 0 + Intercept + log_gest_c*preterm, data = ds,
   seed = 1236,
   prior = prior1,
   sample_prior = "only",
   chains = 4,
   iter = 2000, thin = 1,
   cores = getOption("mc.cores", 4),
   file = "output/hw5_fit2_priorpredict1",
   file_refit = "on_change")
```

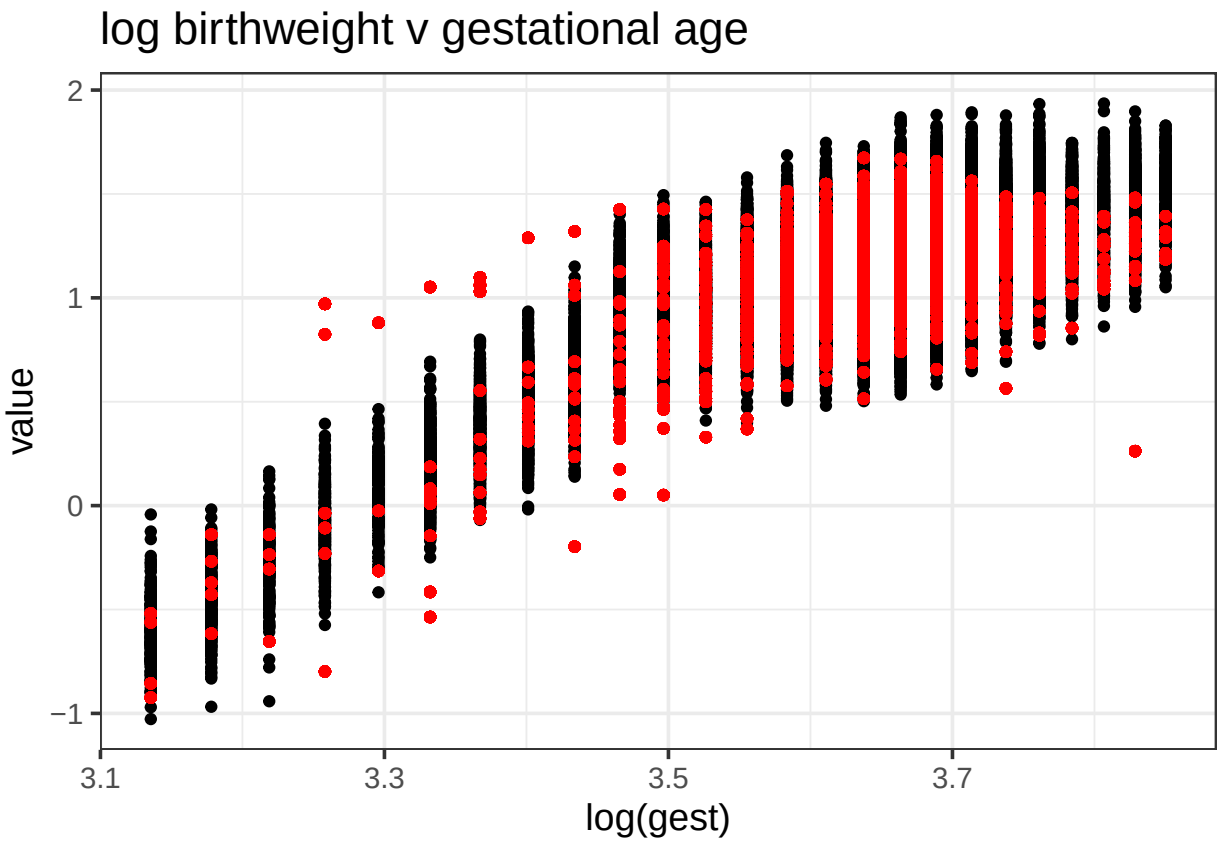
```
mod2_priorpredict2 <- brm(log_weight ~ 0 + Intercept + log_gest_c*preterm, data = ds,
   seed = 1236,
   prior = prior2,
   sample_prior = "yes",
   chains = 4,
   iter = 2000, thin = 1,
   cores = getOption("mc.cores", 4),
   file = "output/hw5_fit2_priorpredict2",
   file_refit = "on_change")
```

In terms of summarizing the prior predictive draws, I started with the same simple approach as above and saw that set1 resulted in implausible outcomes but then noted that for the 2nd set of priors, just a subset of draws did not necessarily seem to not cover the entire range of plausible outcomes. To investigate that further, I then generated a different summary, which is a density of birthweight at specific values of gestational age, to see if those looked reasonable in terms of covering the plausible range of outcomes.

```
ynew_si <- posterior_predict(mod2_priorpredict1)
bw_si <- as_tibble((t(ynew_si[1:50,])),.name_repair = "unique")
bw_si %>% mutate(gest = ds$gest, observed_data = log(ds$birthweight))%>%
  pivot_longer(-c(gest, observed_data)) %>%
  ggplot(aes(x = log(gest),y = value)) +
  geom_point() +
  geom_point(aes(y = observed_data), color = "red") +
  scale_color_brewer(palette = "Set1") +
  theme_bw(base_size = 14) +
  ggtitle("log birthweight v gestational age")
```

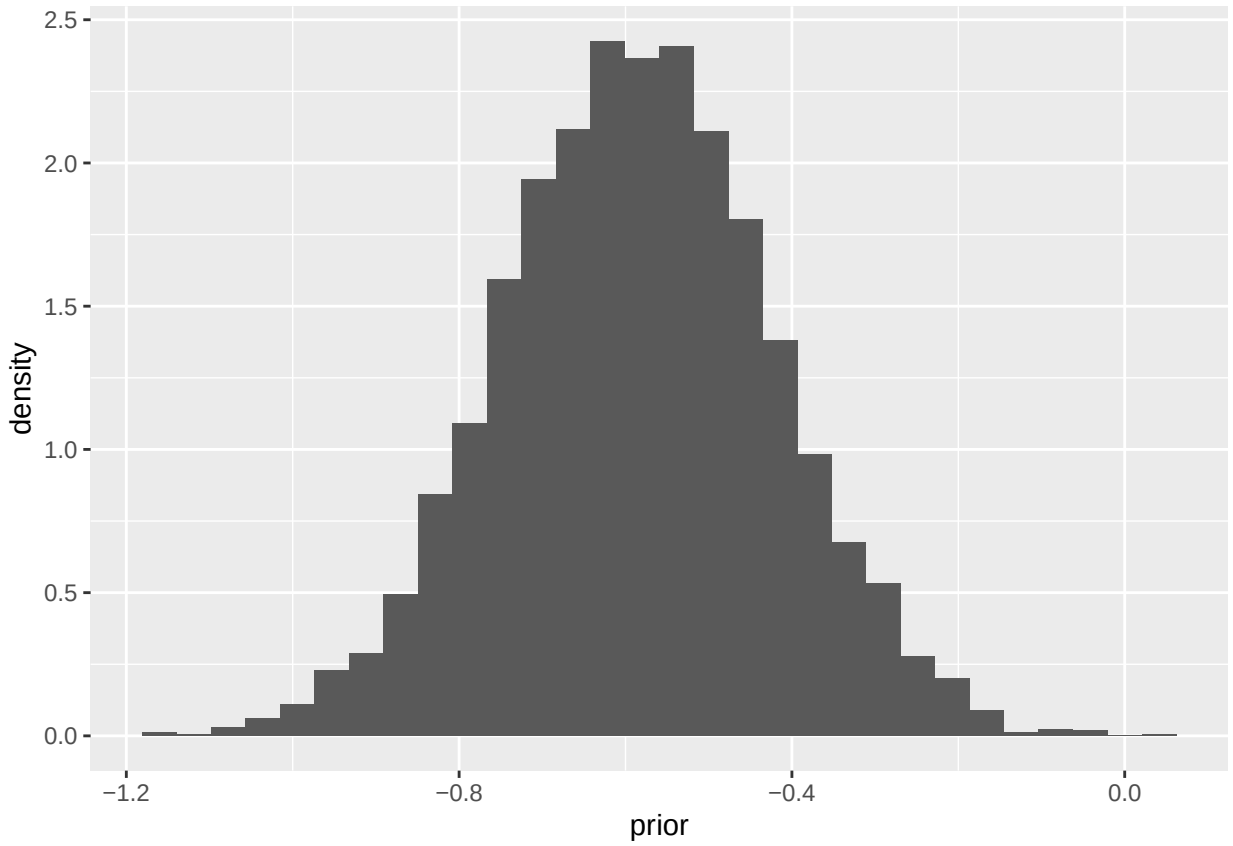



```
ynew_si <- posterior_predict(mod2_priorpredict2)
bw_si <- as_tibble((t(ynew_si[1:50,])),.name_repair = "unique")
bw_si %>% mutate(gest = ds$gest, observed_data = log(ds$birthweight))%>%
  pivot_longer(-c(gest, observed_data)) %>%
  ggplot(aes(x = log(gest),y = value)) +
  geom_point() +
  geom_point(aes(y = observed_data), color = "red") +
  scale_color_brewer(palette = "Set1") +
  theme_bw(base_size = 14) +
  ggtitle("log birthweight v gestational age")
```

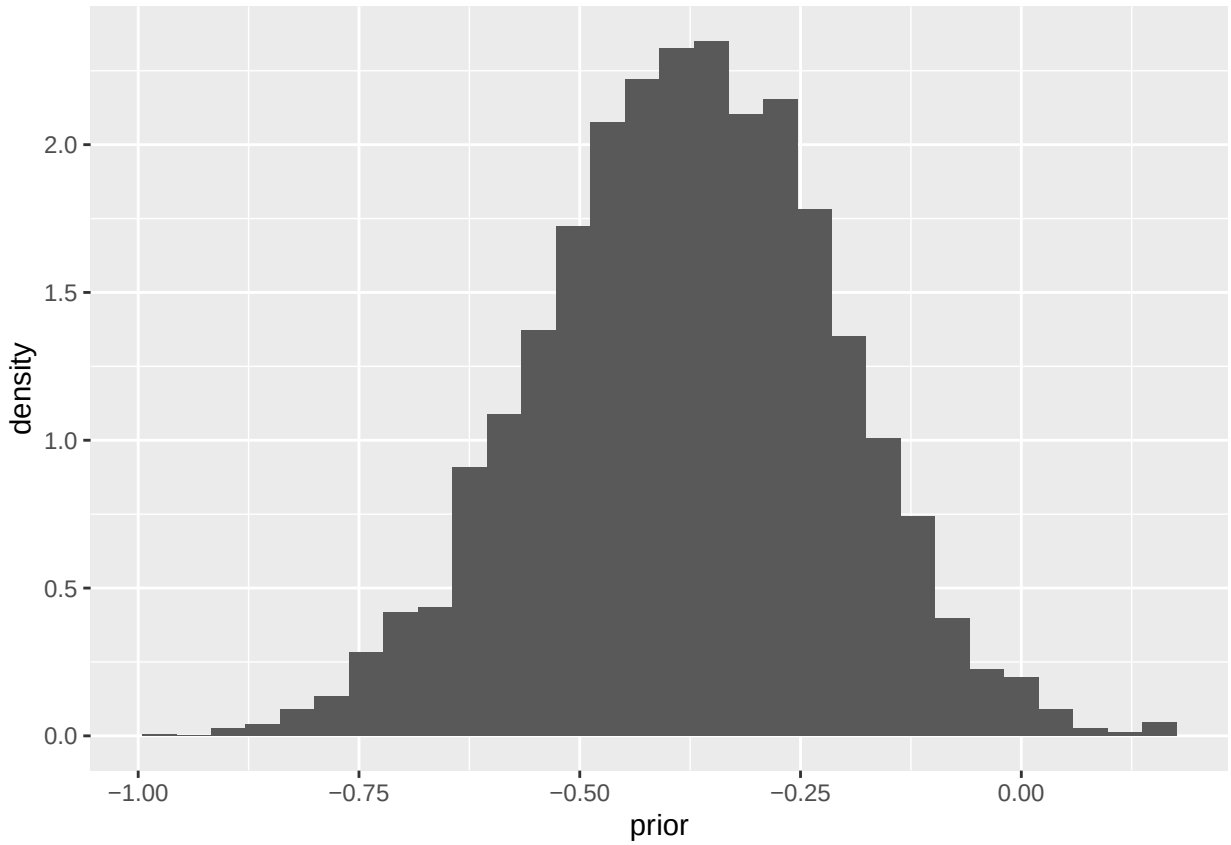


The plot above, using just a subset of prior predictive draws, does not show that the simulated range captures the data. To check this further, we can obtain the density at different ranges of gestational age, as follows:

```
newdata1 <- data.frame( log_gest_c = min(ds$log_gest_c), preterm = "Y")
ynew_s <- posterior_predict(mod2_priorpredict2, newdata1)
tibble(prior = ynew_s) %>%
  ggplot(aes(x = prior)) +
  geom_histogram(aes(y = ..density..))
```




```
newdata1 <- data.frame( log_gest_c = 0.9*min(ds$log_gest_c), preterm = "Y")
ynew_s <- posterior_predict(mod2_priorpredict2, newdata1)
tibble(prior = (ynew_s)) %>%
  ggplot(aes(x = prior)) +
  geom_histogram(aes(y = ..density..))
```



Without being an expert on birth weights, just based on comparing these densities to the data at low gestational age, the densities of the prior predictive samples seems reasonable in terms of their range but it is not clear to me if a priori, it would allow for sufficient prior mass for more extreme outcomes. If this were a real data analysis, using more complicated models, and substantive knowledge would indicate indicate that these priors ranges are a little too narrow, I would go back and reconsider the priors again to produce wider ones. For this homework exercise, we do continue with the second set of priors and check in the next question whether the priors are vague, relative to the posteriors. They are, so I do not expect sensitivity to prior choice here.

Question 2: prior-post comparison

Fit model 2 with prior set 2. Produce plots that compare the priors and posteriors. Are the priors vague relative to the posteriors?

Solution

See plots below. Yes the priors are vague relative to the posteriors, as we see that prior densities are relatively flat in areas with high posterior density.

We first need to fit the model:

```
mod2_prior2 <- brm(log_weight ~ 0 + Intercept + log_gest_c*preterm, data = ds,
   seed = 1236,
   prior = prior2,
   sample_prior = "yes",
```

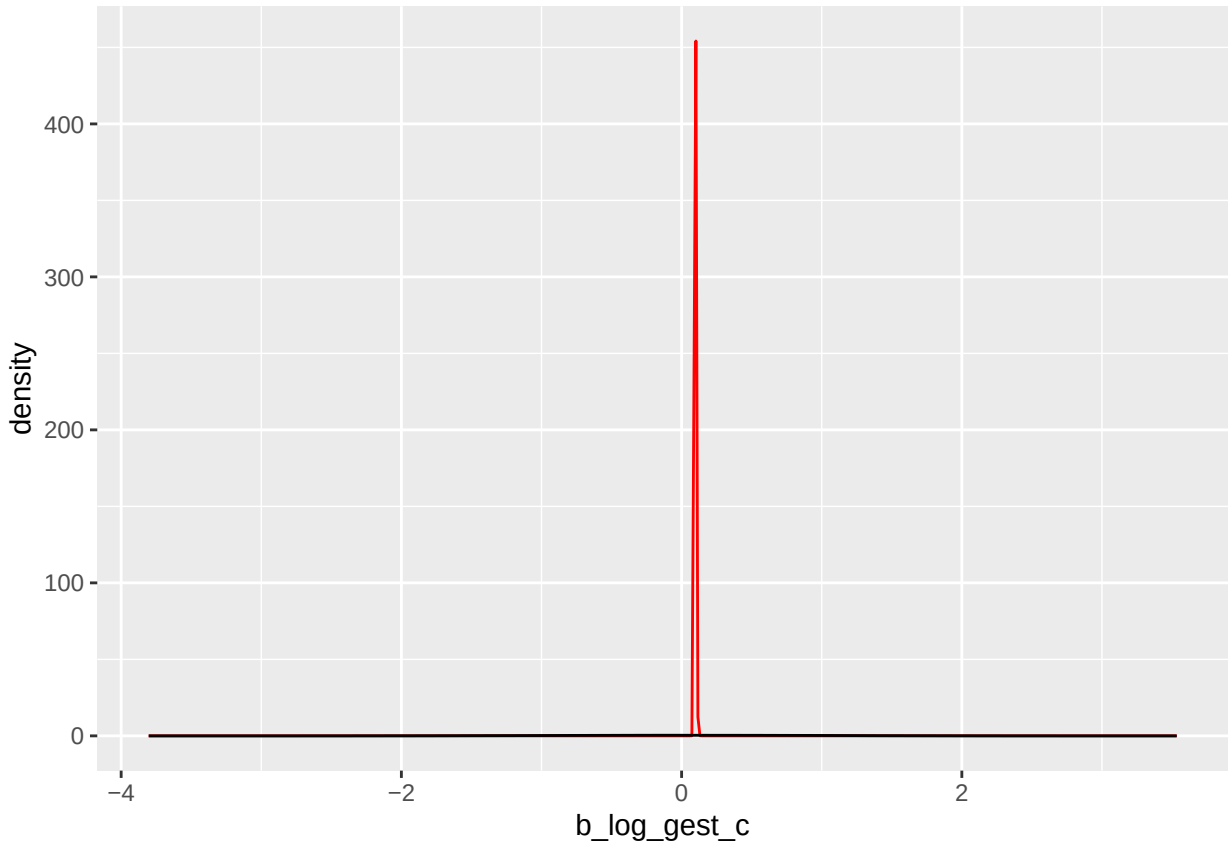
```
  chains = 4,
  iter = 2000, thin = 1,
  cores = getOption("mc.cores", 4),
  file = "output/hw5_fit2_prior2",
  file_refit = "on_change")
```

```
samps2 <- as_draws_matrix(mod2_prior2)
samps2
```

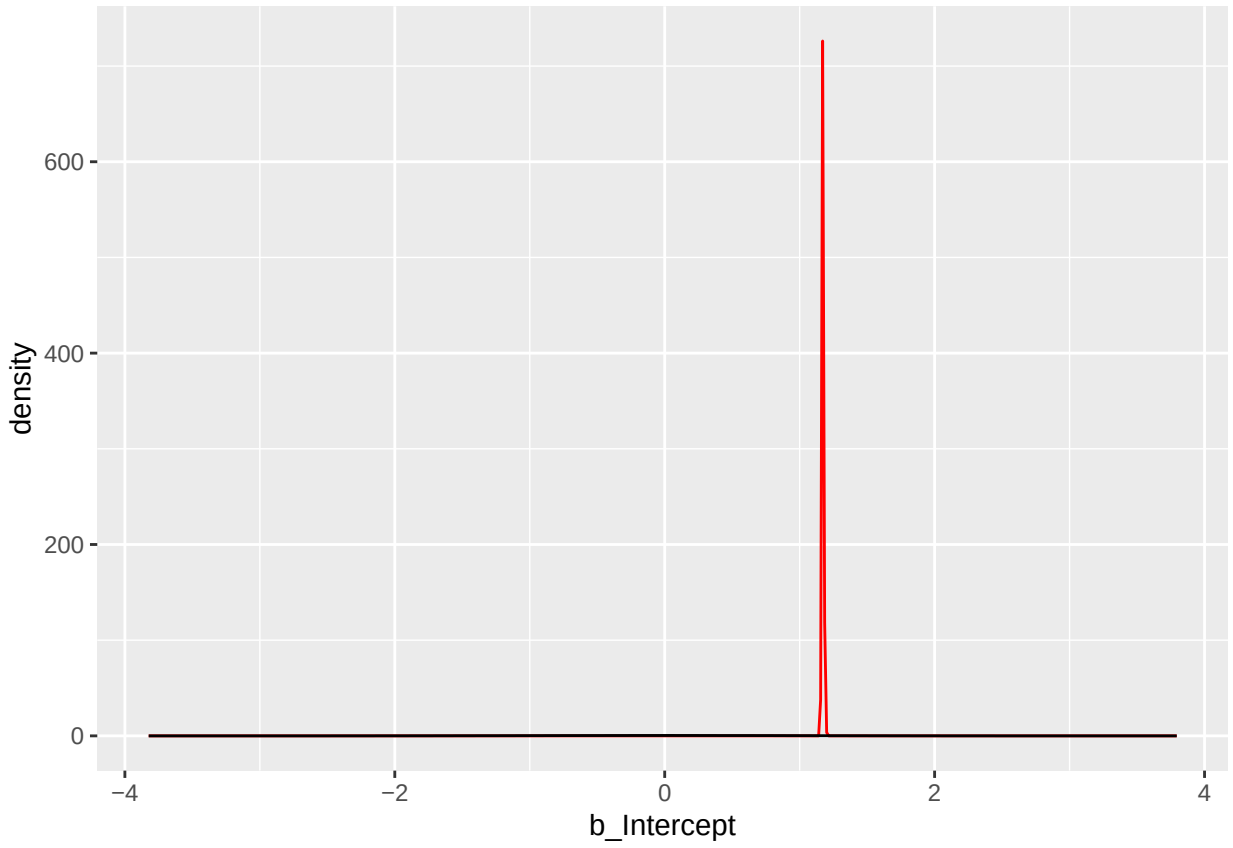
```
## # A draws_matrix: 1000 iterations, 4 chains, and 12 variables
##     variable
## draw b_Intercept b_log_gest_c b_pretermY b_log_gest_c:pretermY sigma
##   1          1.2        0.099       0.33                  0.18  0.16
##   2          1.2        0.097       0.34                  0.18  0.16
##   3          1.2        0.100       0.28                  0.17  0.16
##   4          1.2        0.095       0.37                  0.18  0.16
##   5          1.2        0.096       0.38                  0.19  0.16
##   6          1.2        0.096       0.35                  0.18  0.16
##   7          1.2        0.099       0.27                  0.17  0.16
##   8          1.2        0.098       0.31                  0.17  0.16
##   9          1.2        0.096       0.31                  0.17  0.16
##   10         1.2        0.101       0.37                  0.18  0.16
##     variable
## draw prior_b_Intercept prior_b_log_gest_c prior_b_pretermY
##   1              1.47            -0.344           0.399
##   2              0.47             2.201          -1.107
##   3             -0.63             0.024           0.069
##   4             -0.44             1.040           1.265
##   5             -0.15             0.555          -1.429
##   6             -0.60             1.046          -0.339
##   7              0.57            -0.256          -1.031
##   8              0.70             0.952          -0.651
##   9              0.99             1.681           0.943
##   10            -0.53            -0.443          -0.777
## # ... with 3990 more draws, and 4 more variables
```

Now plot

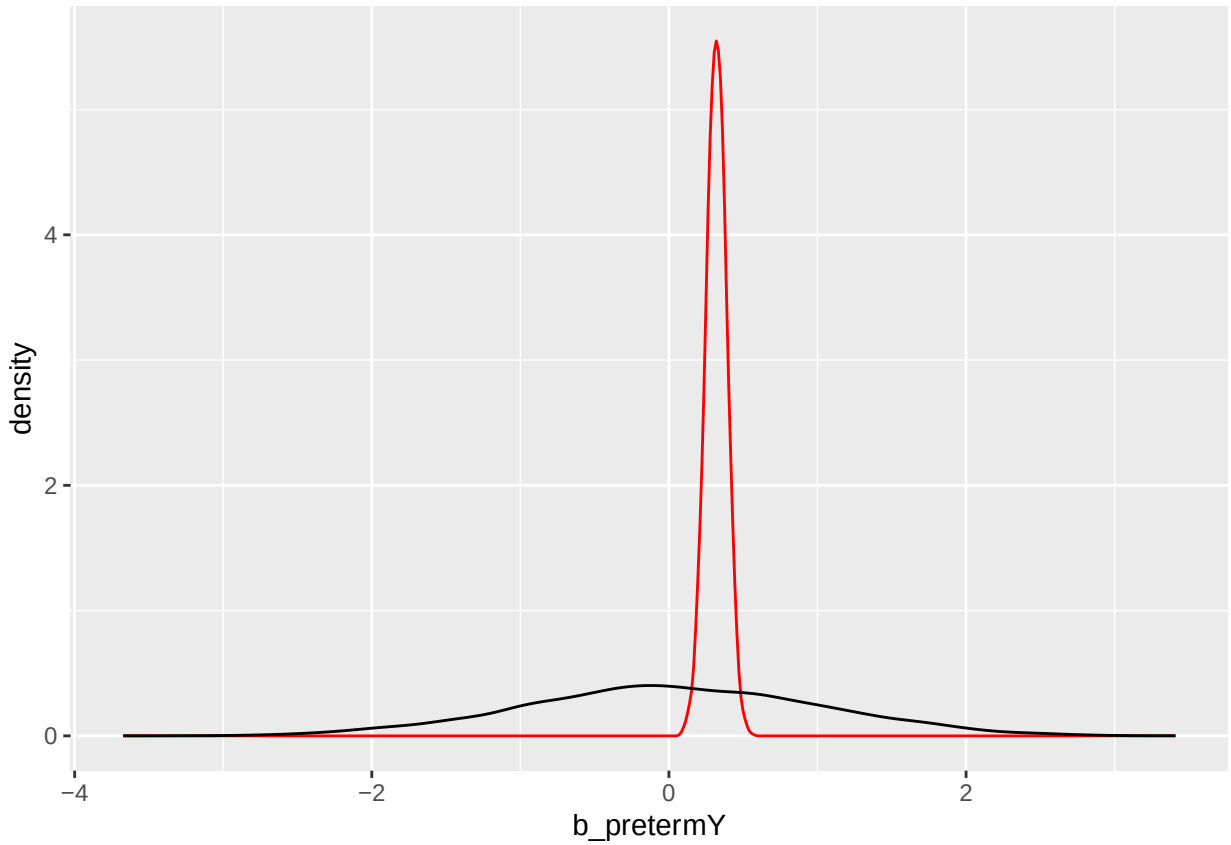
```
as_tibble(samps2) %>%
  ggplot() +
  geom_density(aes(x = b_log_gest_c), color = "red") +
  geom_density(aes(x = prior_b_log_gest_c))
```



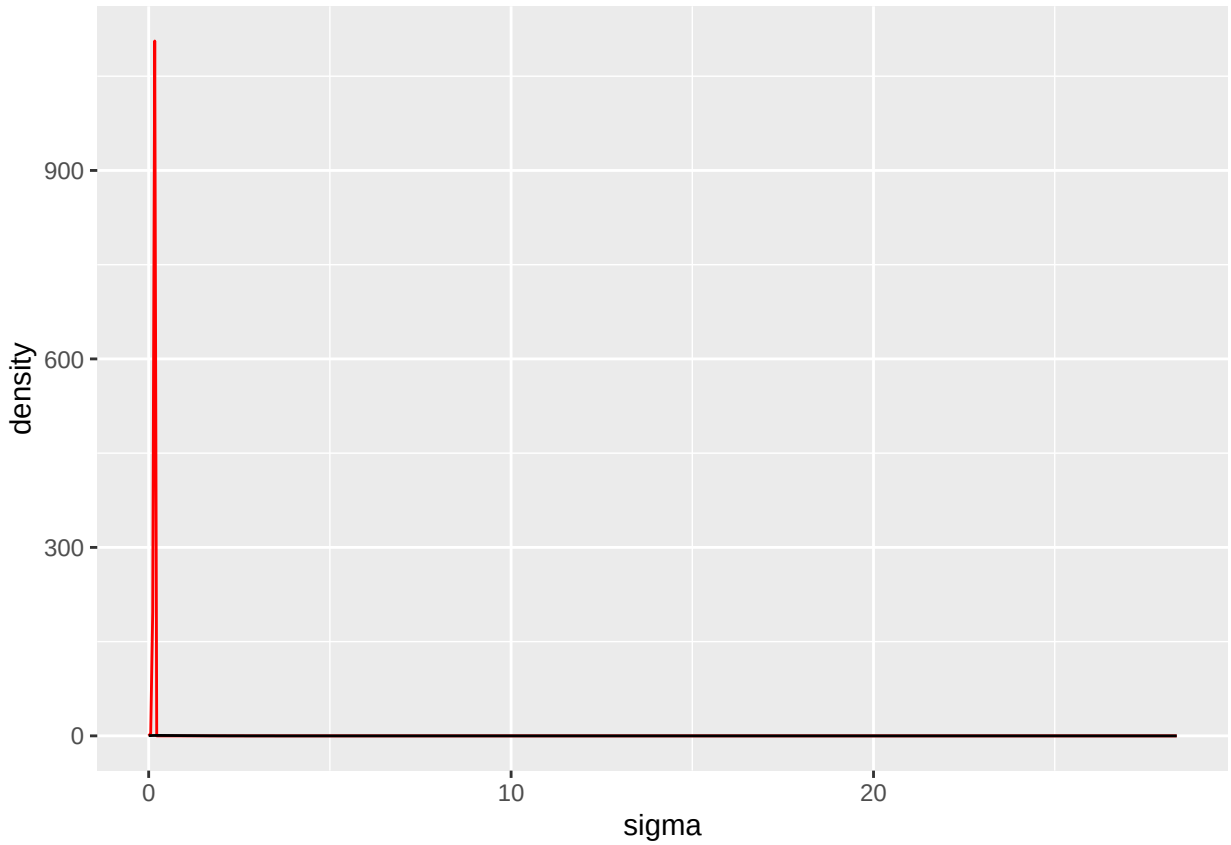

```
as_tibble(samps2) %>%
  ggplot() +
  geom_density(aes(x = b_Intercept), color = "red") +
  geom_density(aes(x = prior_b_Intercept))
```



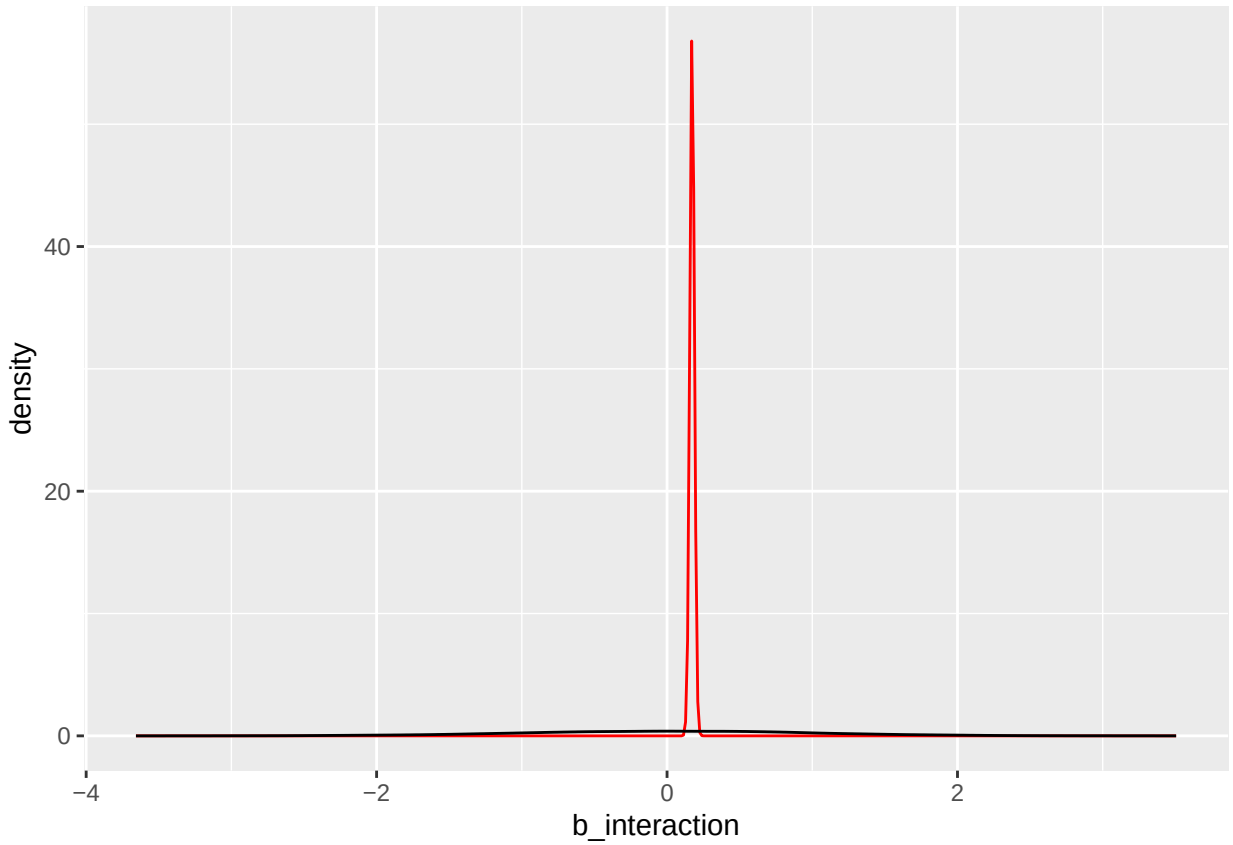

```
as_tibble(samps2) %>%
  ggplot() +
  geom_density(aes(x = b_pretermY), color = "red") +
  geom_density(aes(x = prior_b_pretermY))
```




```
as_tibble(samps2) %>%
  ggplot() +
  geom_density(aes(x = sigma), color = "red") +
  geom_density(aes(x = prior_sigma))
```

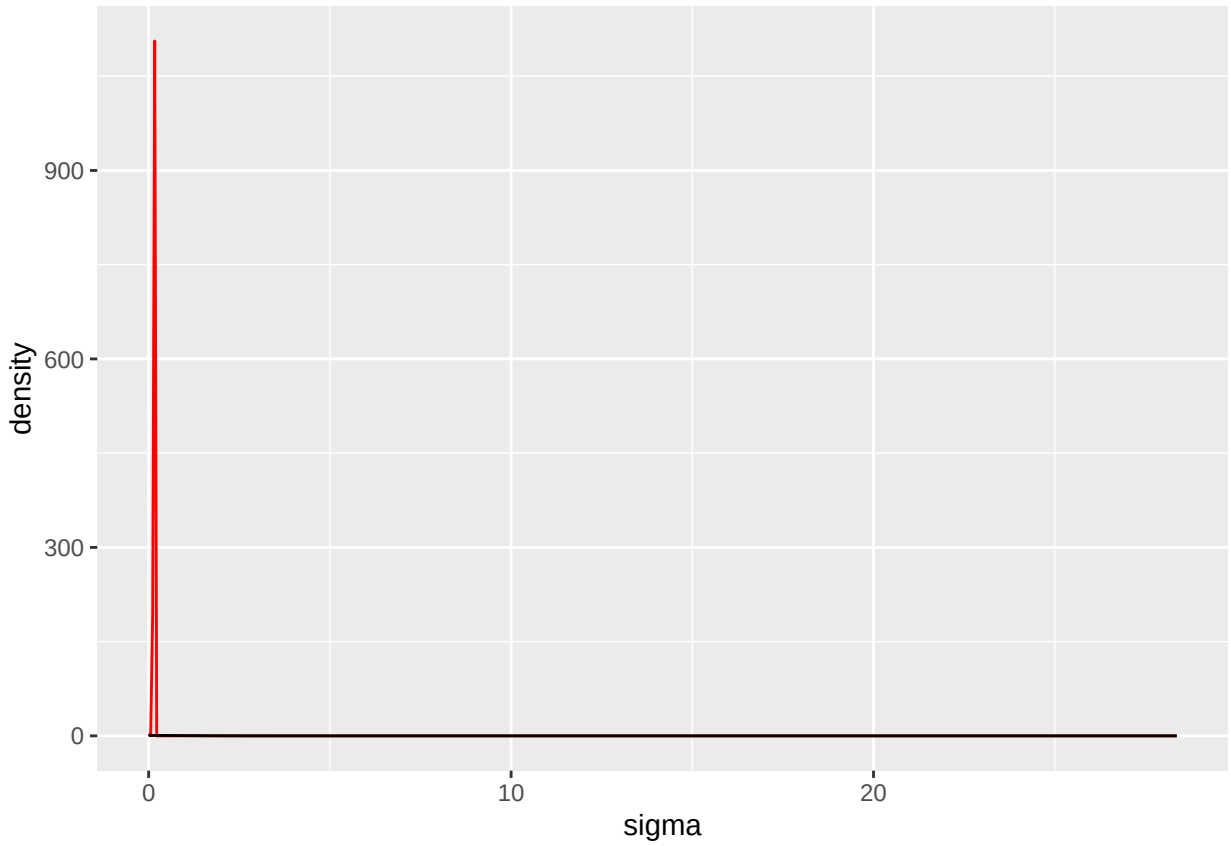



```
as_tibble(samps2) %>%
  rename("b_interaction" ="b_log_gest_c:pretermY", "prior_b_interaction" = "prior_b_log_gest_c:pretermY
  ggplot() +
  geom_density(aes(x = b_interaction), color = "red") +
  geom_density(aes(x = prior_b_interaction ))
```



I didn't ask for it but adding the comparison of prior and post for sigma for completeness

```
as_tibble(samps2) %>%
  ggplot() +
  geom_density(aes(x = sigma), color = "red") +
  geom_density(aes(x = prior_sigma ))
```



Question 3: Posterior predictive checks

We continue with models 1 and 2 with prior set 2. Consider three outcomes of interest: - test quantity 1 = median birth weight - test quantity 2 = proportion of births under 2.5kg - test quantity 3 = proportion of births under 2.5kg for premature births

Introduce notation for each test statistic, i.e. write each test statistic as $T(\boldsymbol{y})$.

For each combination of test statistic and model, plot the test statistics for replicated data sets together with the observed value and calculate the posterior predictive p-values.

Solution

Create the replicated data sets:

```
yrep1_si <- posterior_predict(mod1_prior2)
yrep2_si <- posterior_predict(mod2_prior2)
```

Test quantity 1 = median birth weight: $T(\boldsymbol{y}) = median(\boldsymbol{y})$. The posterior predictive p-values are 0 for both models, suggesting that the model is not able to replicate the median observed outcome.

```
mean(apply(yrep1_si, 1, median) > median(ds$log_weight))
```

```
## [1] 0
```

```
mean(apply(yrep2_si, 1, median) > median(ds$log_weight))
```

```
## [1] 0
```

Test quantity 2 = proportion of births under 2.5kg: $T(\boldsymbol{y}) = 1/nsum_n 1(\exp(y_i) < 2.5)$. The posterior predictive p-values are 0 for model 1 and 5.5% for model 2, suggesting slight better replication for model 2.

```
mean(apply(exp(yrep1_si) < 2.5, 1, mean) > mean(exp(ds$log_weight) < 2.5))
```

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

```
mean(apply(exp(yrep2_si) < 2.5, 1, mean) > mean(exp(ds$log_weight) < 2.5))
```

```
## [1] 0.944
```

Test quantity 3 = proportion of births under 2.5kg for preterm births: $T(\boldsymbol{y}) = 1/nsum_n 1(\exp(y_i) < 2.5, p_i = 1)$ where $p_i = 1$ refers to preterm. The posterior predictive p-values are 0 for both models, suggesting that the model is not able to replicate the observed outcome.

```
select_obs <- ds$preterm == "Y"
mean(apply(exp(yrep1_si[,select_obs]) < 2.5, 1, mean) > mean(exp(ds$log_weight[select_obs]) < 2.5))
```

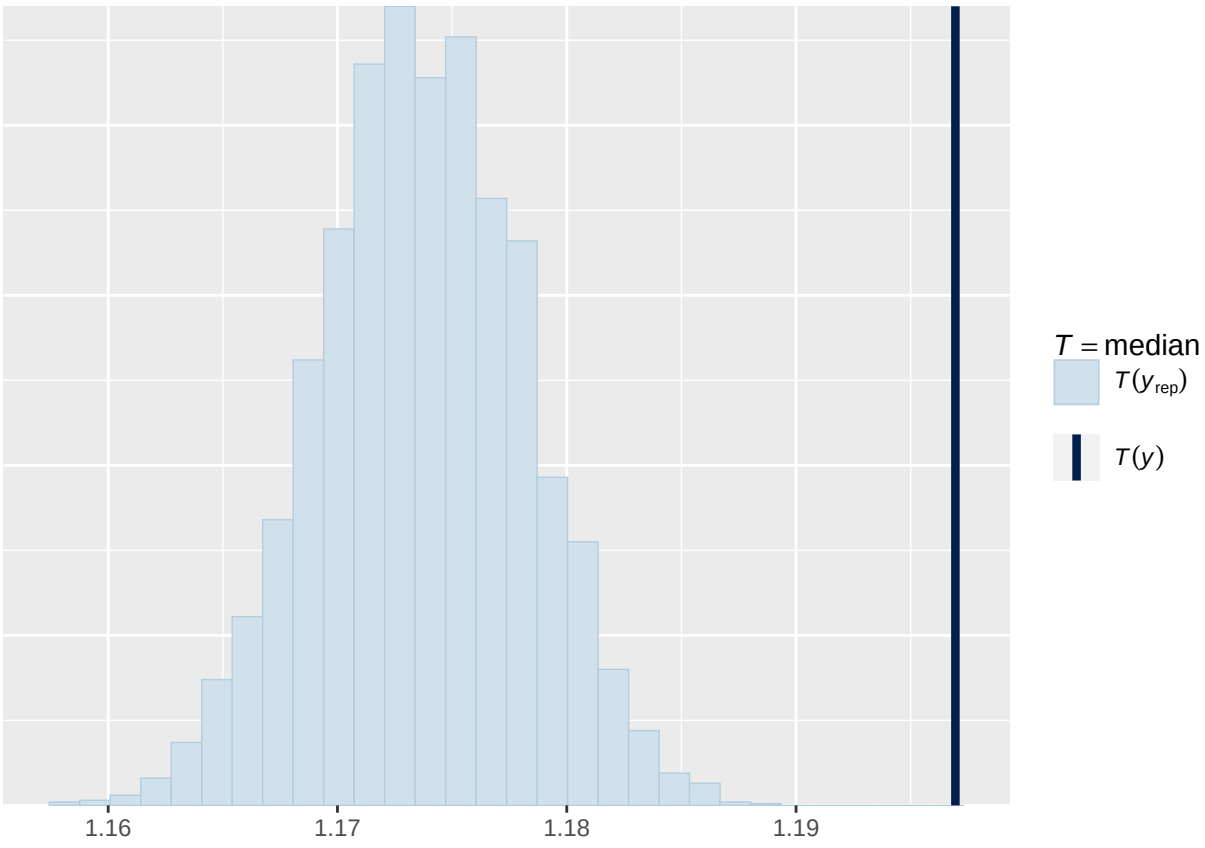
```
## [1] 1
```

```
mean(apply(exp(yrep2_si[,select_obs]) < 2.5, 1, mean) > mean(exp(ds$log_weight[select_obs]) < 2.5))
```

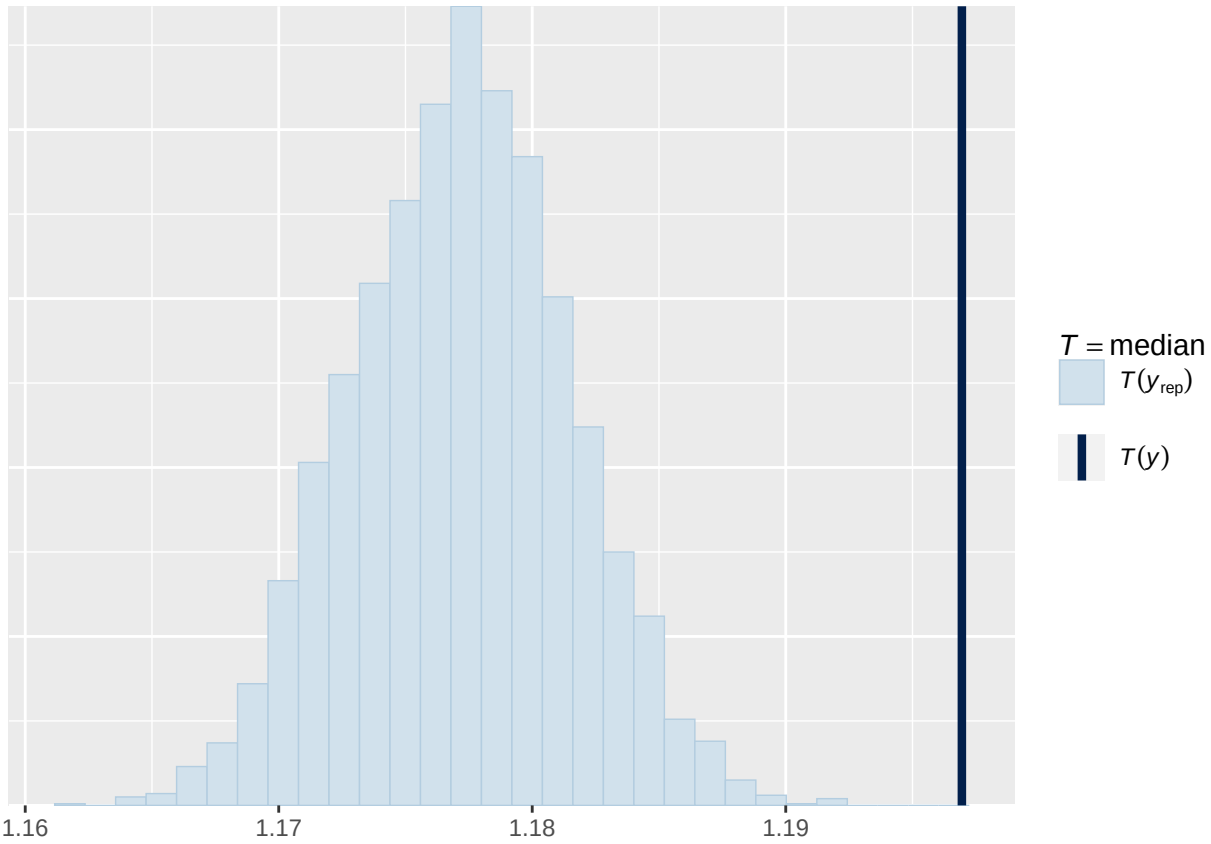
```
## [1] 1
```

Plots are as follows:

```
ppc_stat(ds$log_weight, yrep1_si, stat = 'median')
```

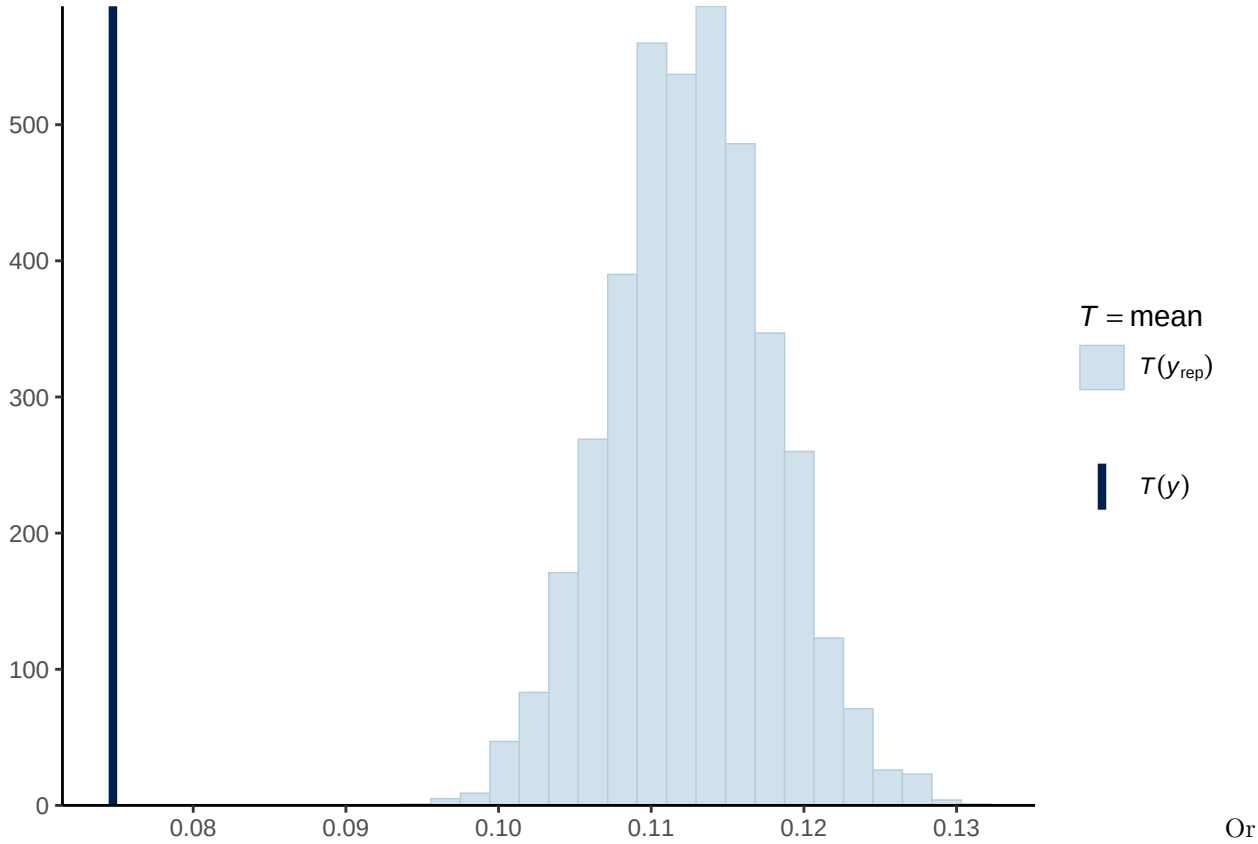



```
ppc_stat(ds$log_weight, yrep2_si, stat = 'median')
```



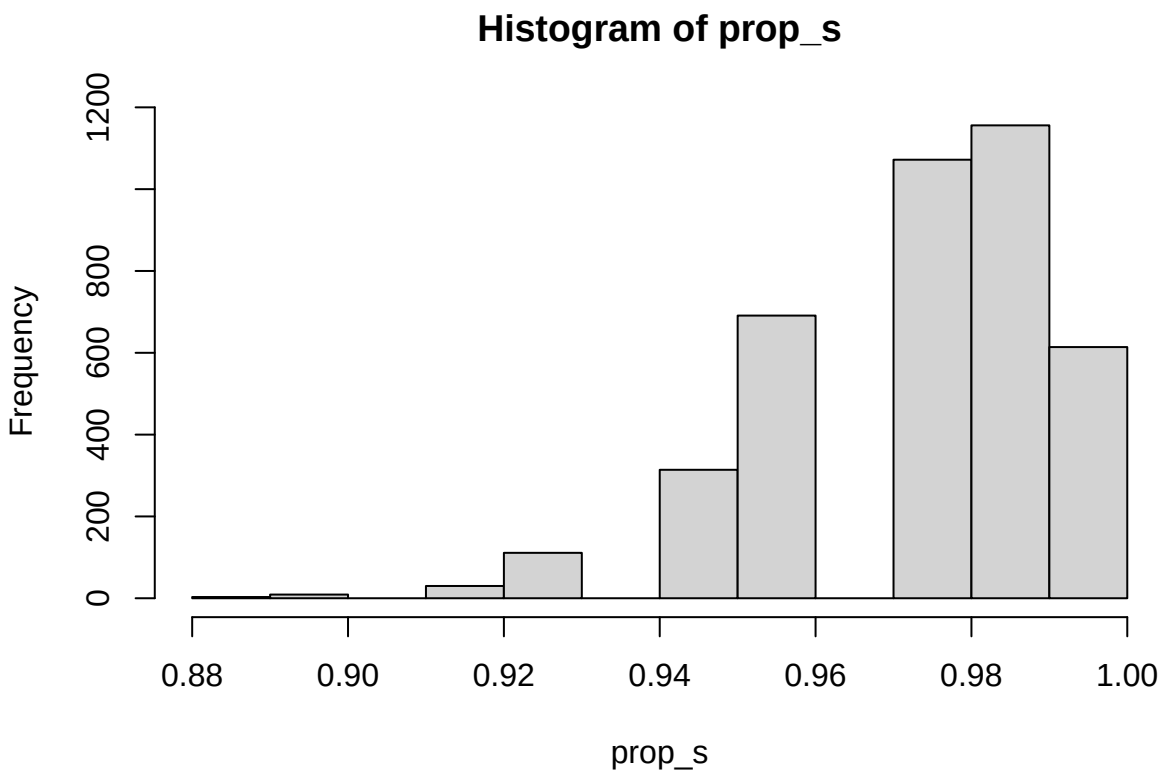
Using bayesplot

```
ppc_stat(as.numeric(exp(ds$log_weight) < 2.5), 1*(exp(yrep1_si) < 2.5), stat = 'mean')+
  theme_classic()
```

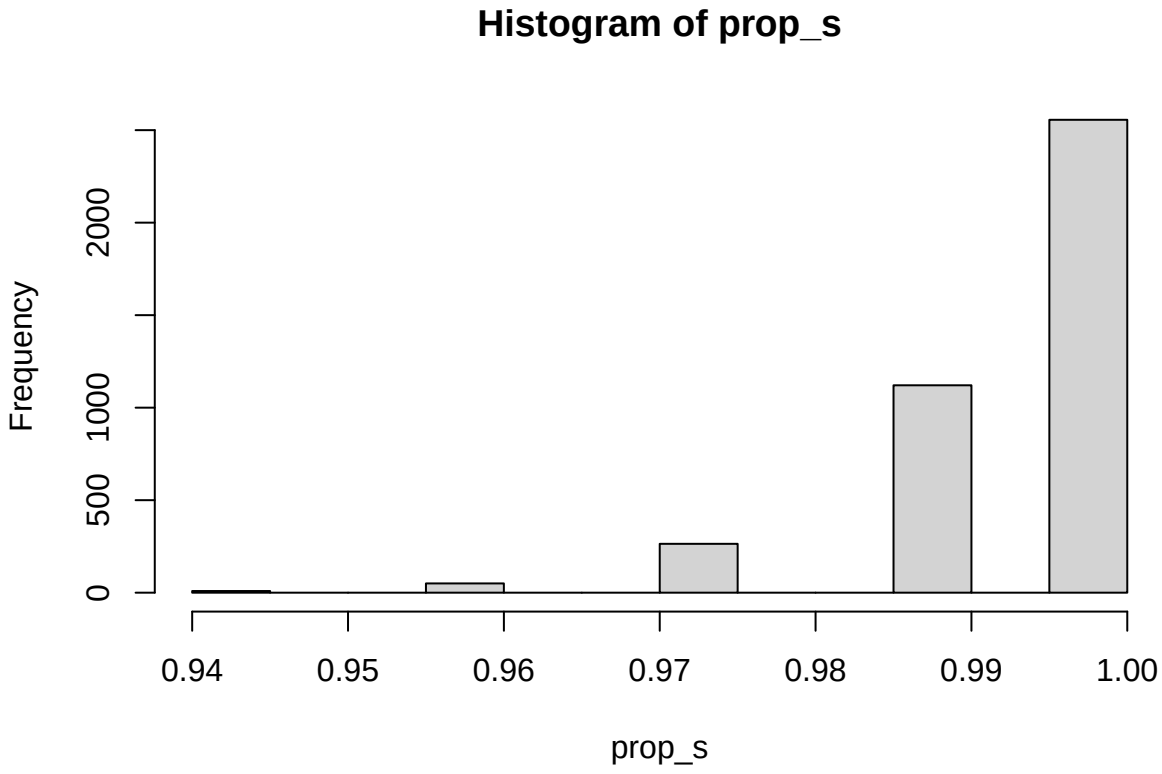


Or coding it up direclty:

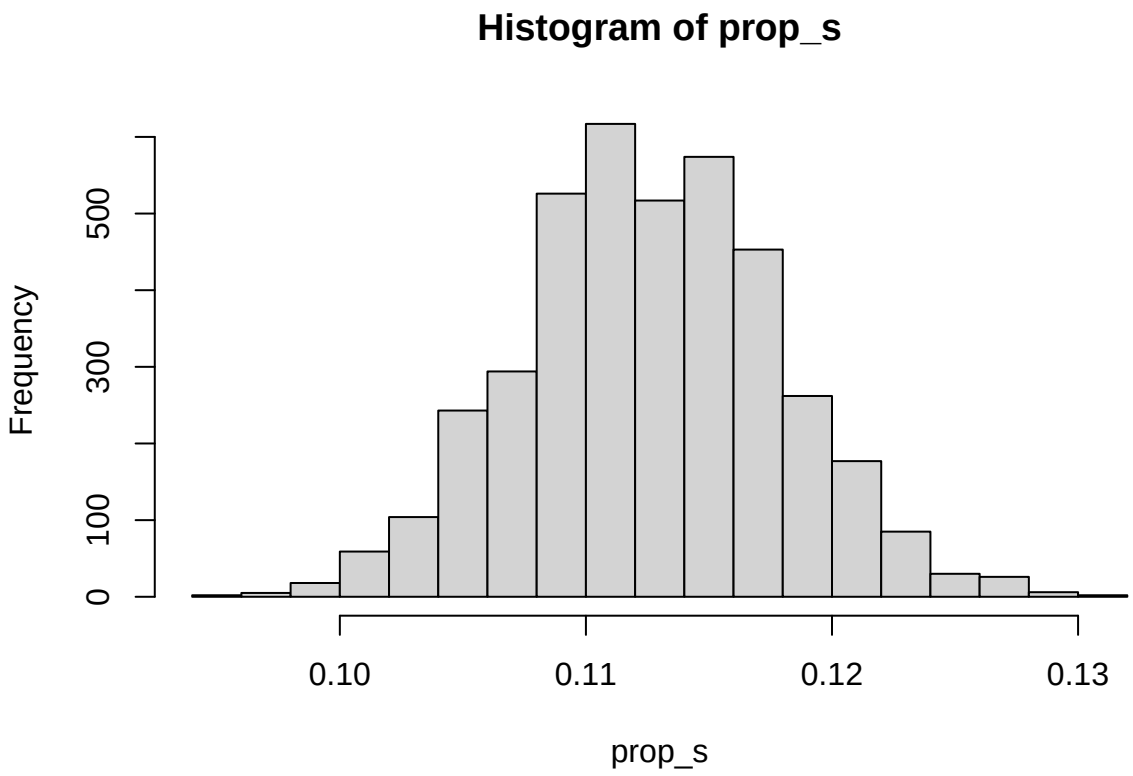
```
prop_s <- apply(exp(yrep1_si[,select_obs ]) < 2.5, 1, mean)
hist(prop_s, title = "Model 1")
abline(v = mean(exp(ds$log_weight[select_obs ]) < 2.5))
```



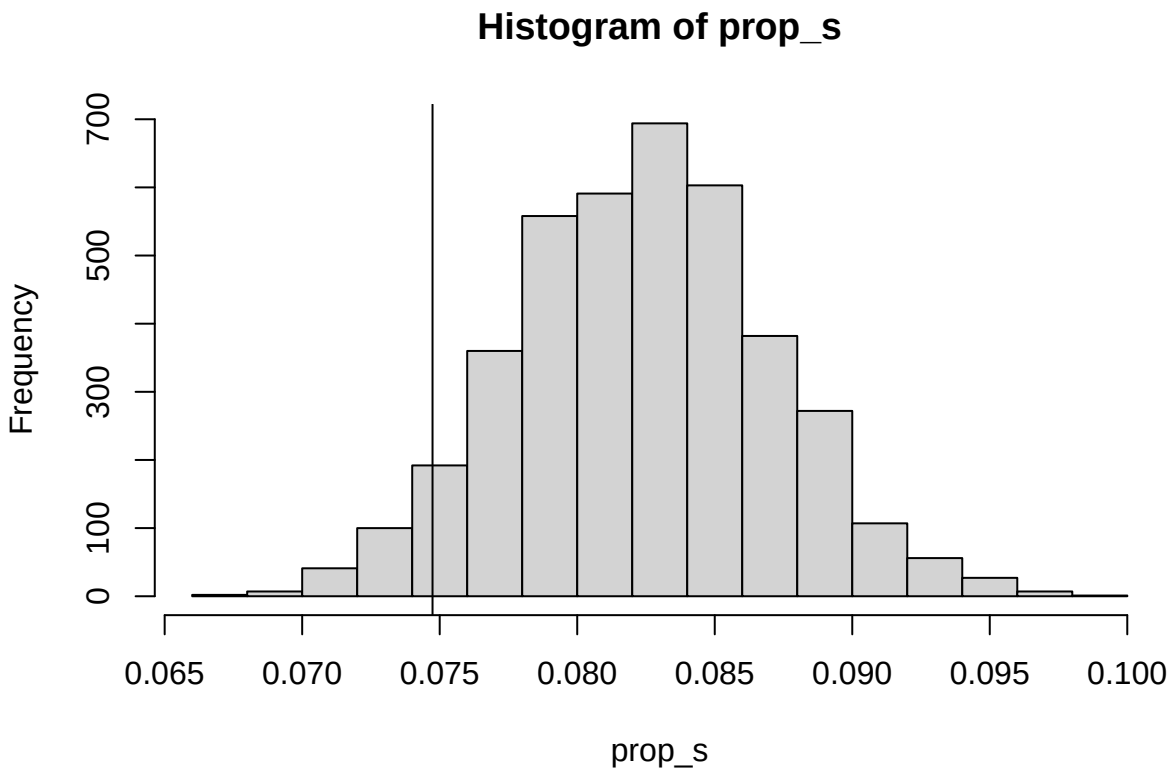

```
prop_s <- apply(exp(yrep2_si[,select_obs ]) < 2.5, 1, mean)
hist(prop_s, title = "Model 1")
abline(v = mean(exp(ds$log_weight[select_obs ]) < 2.5))
```



```r
prop_s <- apply(exp(yrep1_si) < 2.5, 1, mean)
hist(prop_s, title = "Model 1")
abline(v = mean(exp(ds$log_weight) < 2.5))
```




```r
prop_s <- apply(exp(yrep2_si) < 2.5, 1, mean)
hist(prop_s, title = "Model 2")
abline(v = mean(exp(ds$log_weight) < 2.5))
```



Question 4: leave-one-out cross-validation

Use the loo-package to carry out approximate leave-one-out cross-validation for the two models, both with prior set 2.

Specific questions:
- Obtain the Pareto k estimates and plot these against observation index. Explain if there are any instances of influential points suggested by the values.
- Obtain the loo-PIT plot for model 2, using ppc_loo_pit_overlay (see help file or code in module 11). Explain what the curves refer to and briefly, what the plot tells you about the fit.

- Obtain the pointwise ELPDs for models 1 and 2. Plot the pointwise differences in ELPDs (i.e. $ELPD_i$ from model 2 - $ELPD_i$ from model 1) against gestational age. What do you conclude?

Note that for a given model, ELPDs can be obtained as follows (example code):

```
loo_res2 <- loo(mod2_prior2, save_psis = TRUE)
loo_res2$pointwise[, "elpd_loo"]
```

the result is a vector with the ith entry referring to the ELPD for observation i.

Solution Q4

get psis-loo results for our models:

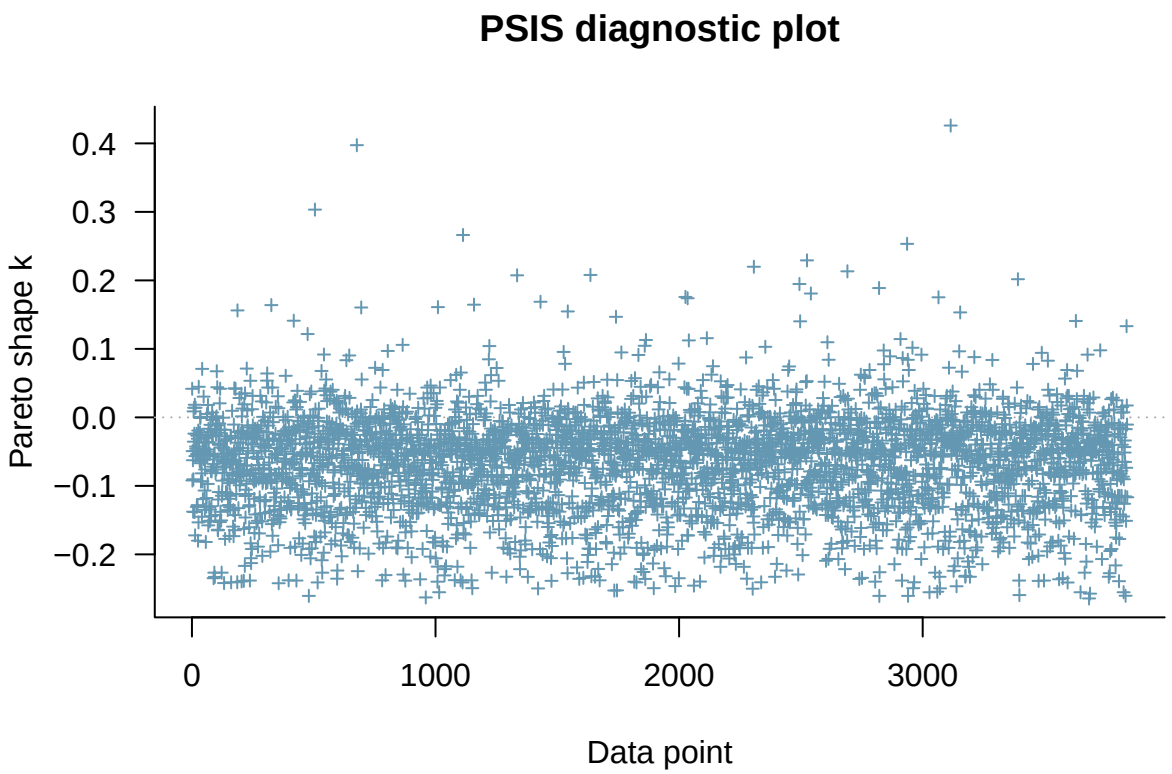
```
loo_res1 <- loo(mod1_prior2, save_psis = TRUE)
loo_res2 <- loo(mod2_prior2, save_psis = TRUE)
```

Check summaries, focus on note on Pareto k diagnostics:

```
loo_res2
```

```
##
## Computed from 4000 by 3840 log-likelihood matrix
##
##          Estimate    SE
## elpd_loo   1622.9  76.1
## p_loo        16.1   2.7
## looic     -3245.7 152.3
## ------
## 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.
```

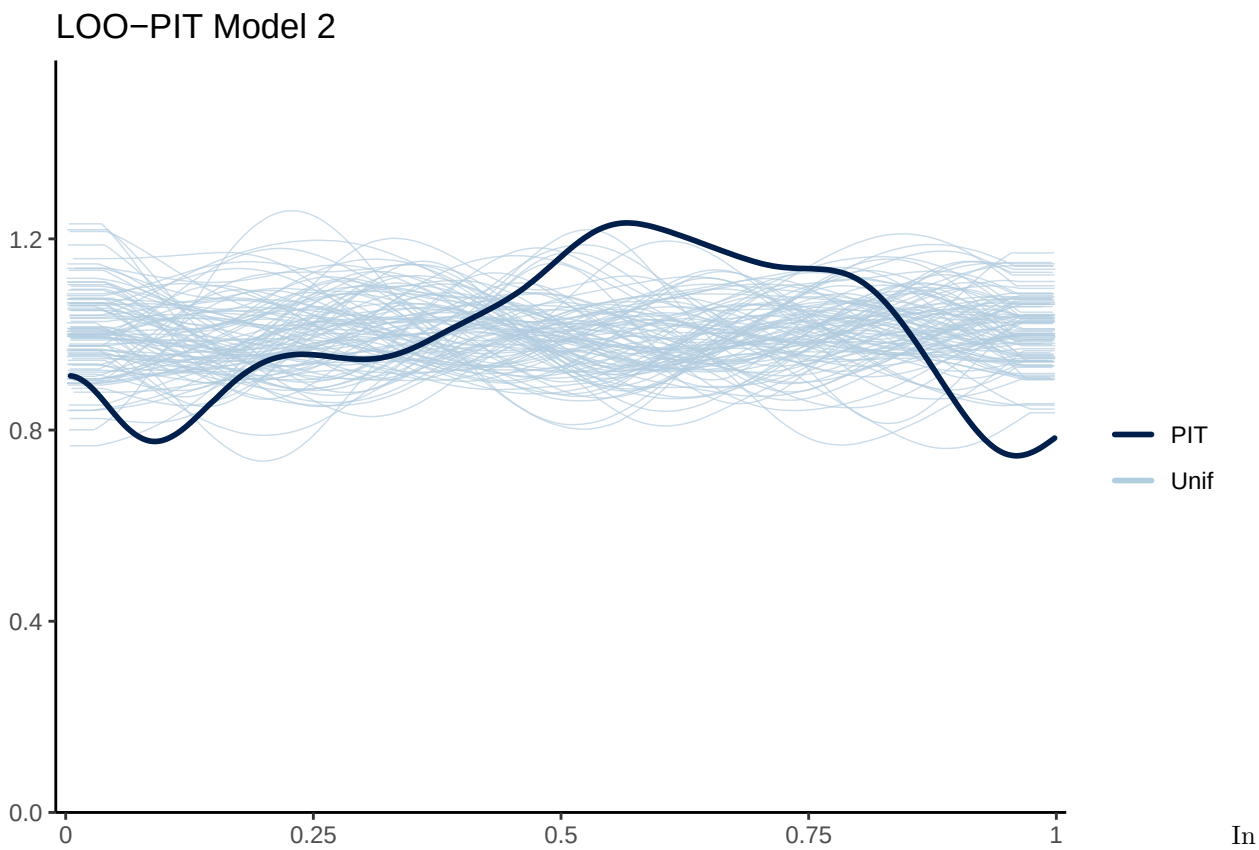
```
plot(loo_res2,
  diagnostic = c("k"),
  label_points = TRUE,
  main = "PSIS diagnostic plot"
)
```



Pareto k values are less than 0.5, indicating no concerns regarding influential points.

The loo-PIT for model 2 is as follows:

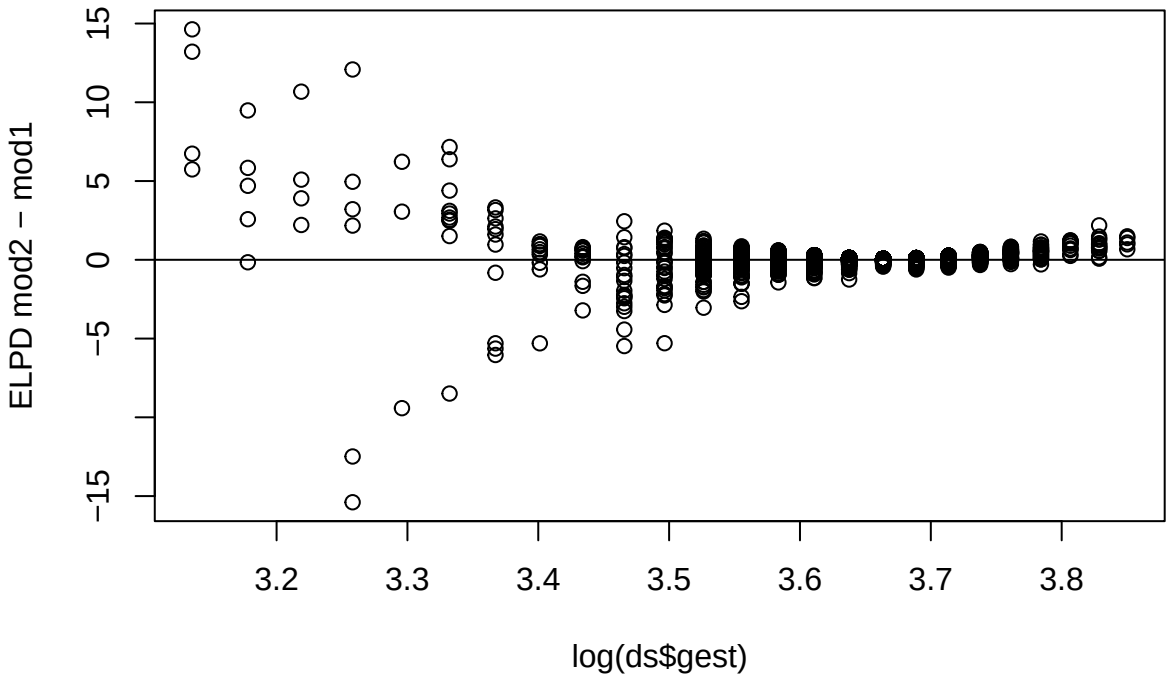
```
ppc_loo_pit_overlay(y = ds$log_weight, yrep2_si, lw = weights(loo_res2$psis_object))+
  ggtitle("LOO-PIT Model 2") +
  theme_classic()
```



In this plot, the black curve (labeled PIT) has the probability-integral-transform values for each of the y_i, using their respective approximated loo density, $PIT_i \approx P(\tilde{y}_i \leq y_i|\boldsymbol{y}_{-i})$ with random variable $\tilde{y}_i \sim p(\tilde{y}_i|\boldsymbol{y}_{-i})$. If $y_i \sim p(\tilde{y}_i|\boldsymbol{y}_{-i})$, then $PIT_i \sim U(0,1)$. The blue curves are densities based on n random draws from a U(0,1)$ density, illustrating what PIT density curves we expect to see when $PIT_i \sim U(0,1)$. In this result plot, we see some deviation away from a constant density at 1, the comparison with the sampled curves suggests that these deviations may be a little extreme as compared to those occurring at random. Deviations away from uniformity suggest that there are differences between what's predicted using the loo-predictive density and what's observed.

The plot with differences in ELPDs is obtained as follows:

```
#head(loo_res2$pointwise)
#loo_res2$pointwise[, "elpd_loo"]
plot(loo_res2$pointwise[, "elpd_loo"] - loo_res1$pointwise[, "elpd_loo"]  ~ log(ds$gest), ylab = "ELPD r
abline(h=0)
```



ELPDs for model 2 are greater than those for model 1 at lower gestational age, suggesting that model 2 fits these observations better, which is what we would expect based on the EDA. There are some observations where the order is switched, with ELPD from model 1 being larger than that of model 2. At higher gestational age, the differences in ELPDs are smaller in magnitude but the differences are again mostly positive.

We did not discuss this in class but we can also use the loo results to produce a comparison, the blog post includes this information as well.

```
compare(loo_res1, loo_res2)
```

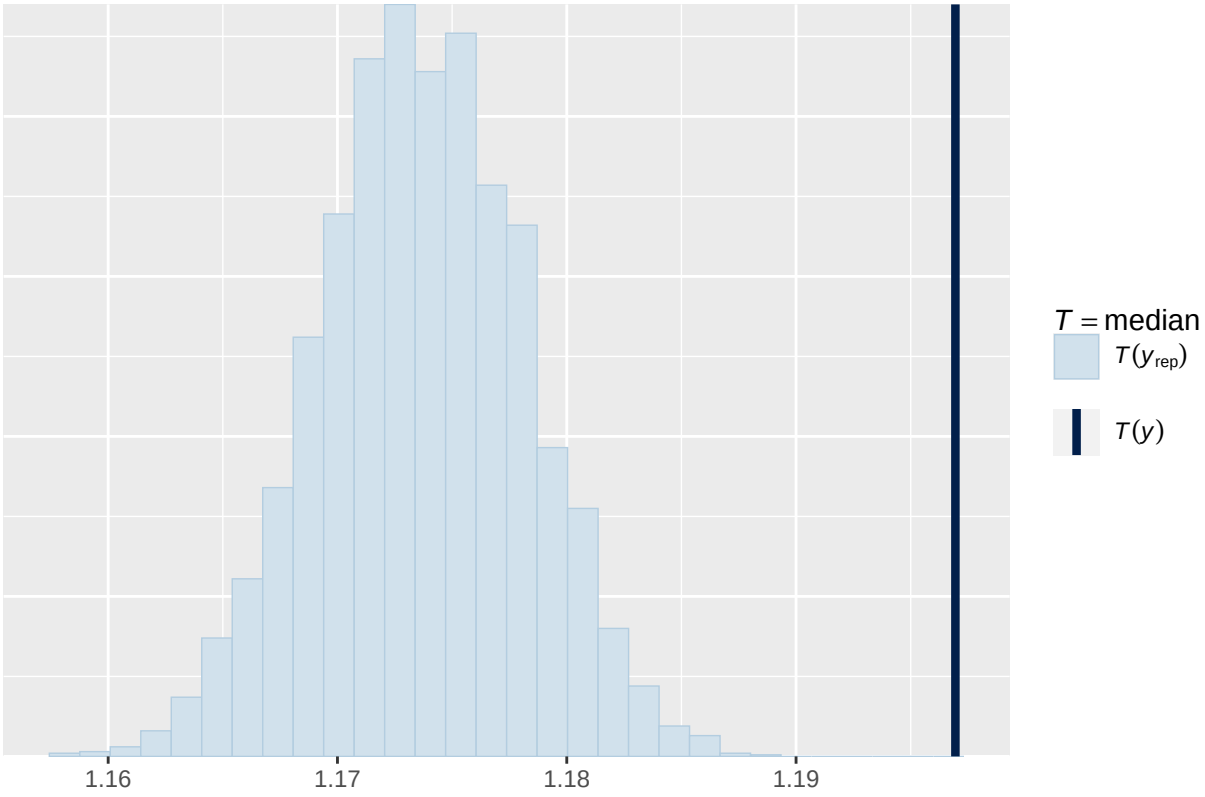
```
## elpd_diff        se
##     218.7      48.1
```

Here we find that model 2 is preferred.

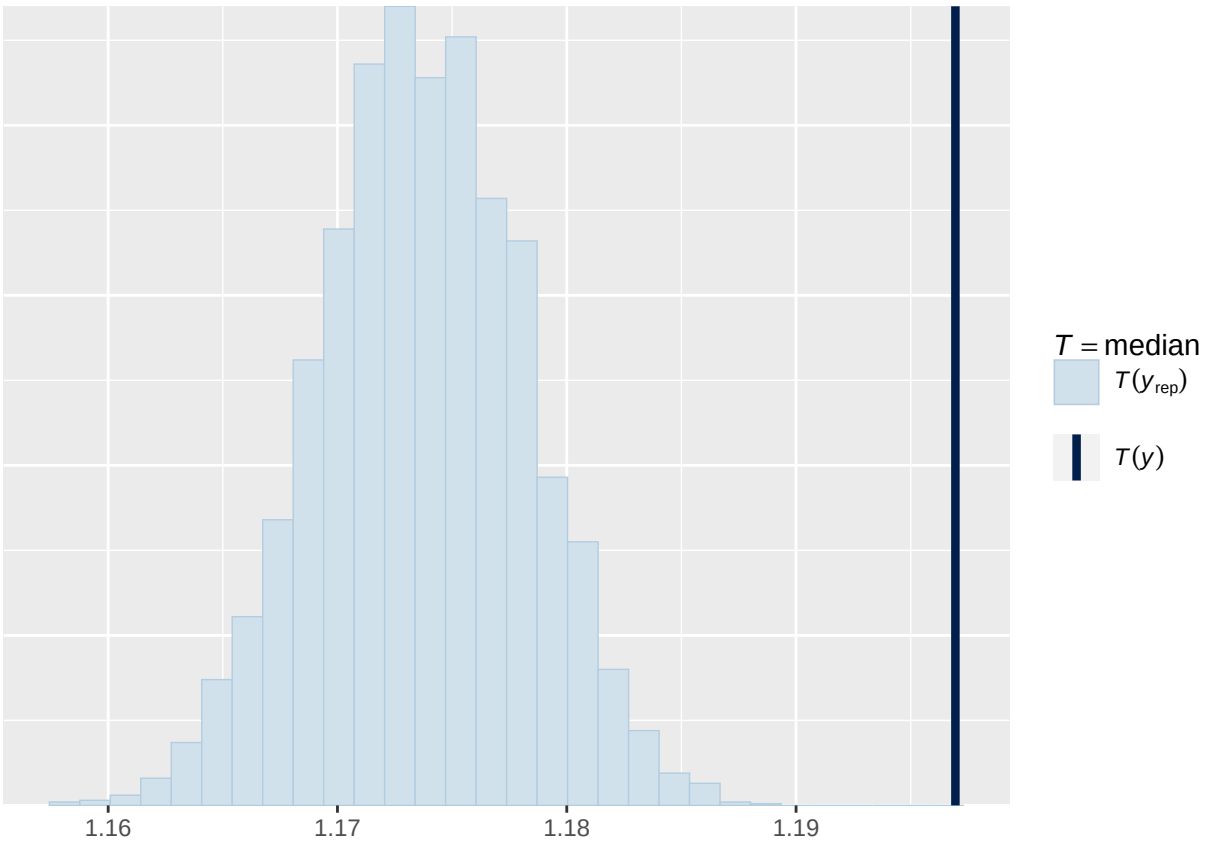
Note that with the loo-weights, we can also repeat the posterior predictive checks we did earlier: by using the weights, we are using approximate samples from the LOO densities:

```
ppc_stat(ds$log_weight, yrep1_si, lw = weights(loo_res1$psis_object), stat = 'median') +
  ggtitle("LOO-PIT Model 1, w weights")
```

LOO−PIT Model 1, w weights




```
ppc_stat(ds$log_weight, yrep1_si,  stat = 'median')
```


1.16
1.17
1.18
1.19
T = median
T(y_rep)
T(y)