

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
- `mracehis` 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.

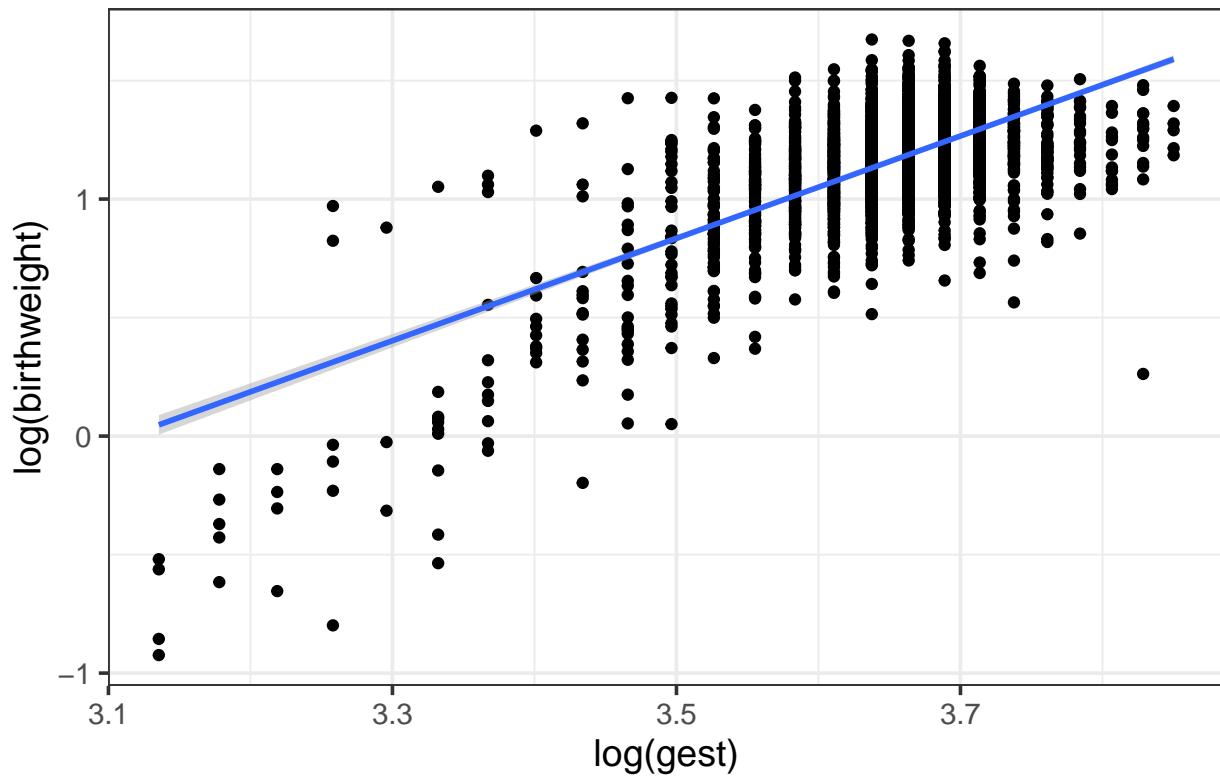
```
ds <- ds %>%
  rename(birthweight = dbwt, gest = combgest) %>%
  mutate(preterm = ifelse(gest<32, "Y", "N")) %>%
  filter(ilive=="Y", gest< 99, birthweight<9.999)
```

Some EDA

Some plots, taken from Monica's blog

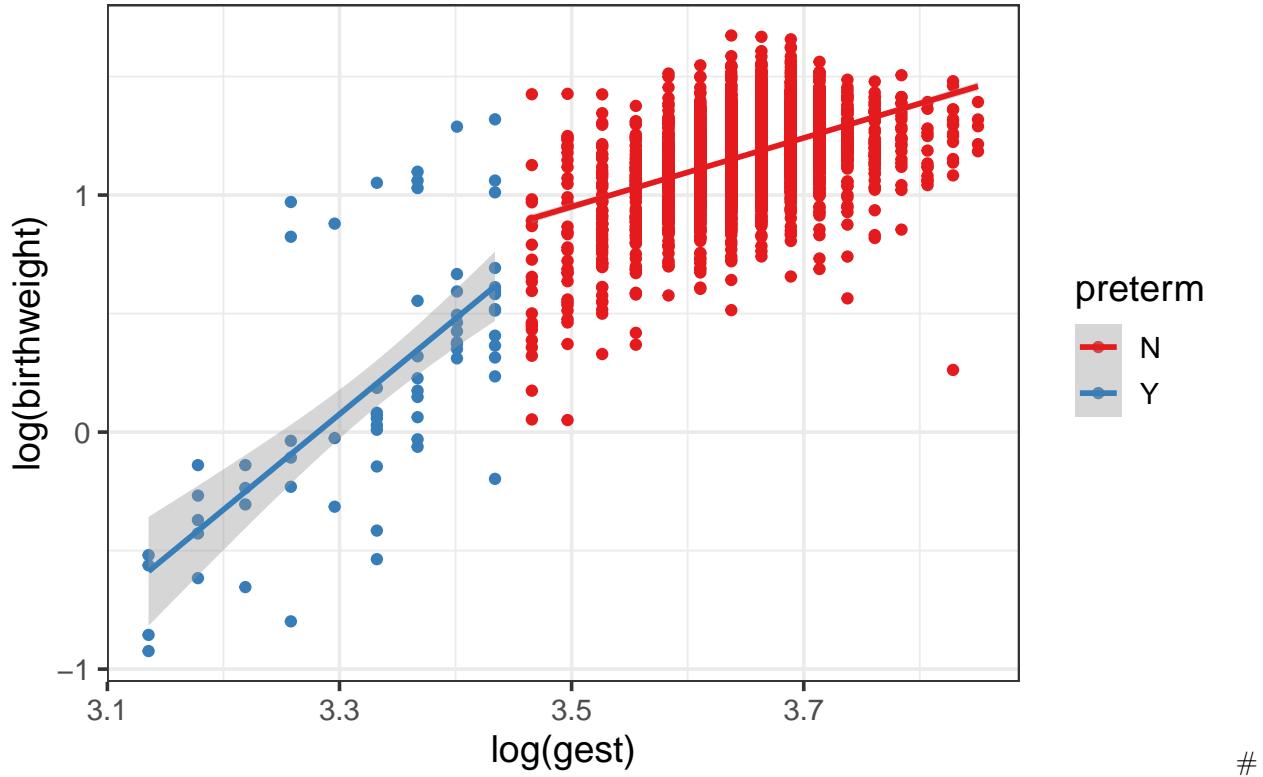
```
ds %>%
  ggplot(aes(log(gest), log(birthweight))) +
  geom_point() + geom_smooth(method = "lm") +
  scale_color_brewer(palette = "Set1") +
  theme_bw(base_size = 14) +
  ggtitle("birthweight v gestational age")
```

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")
```

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 $\log(\text{gestational age})$, 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 nelpar 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 $y \sim x$, 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 $y \sim 0 + \text{Intercept} + \text{covariates}$, 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 $y \sim 0 + \text{Intercept} + \text{covariates}$, 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_fiti_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
```

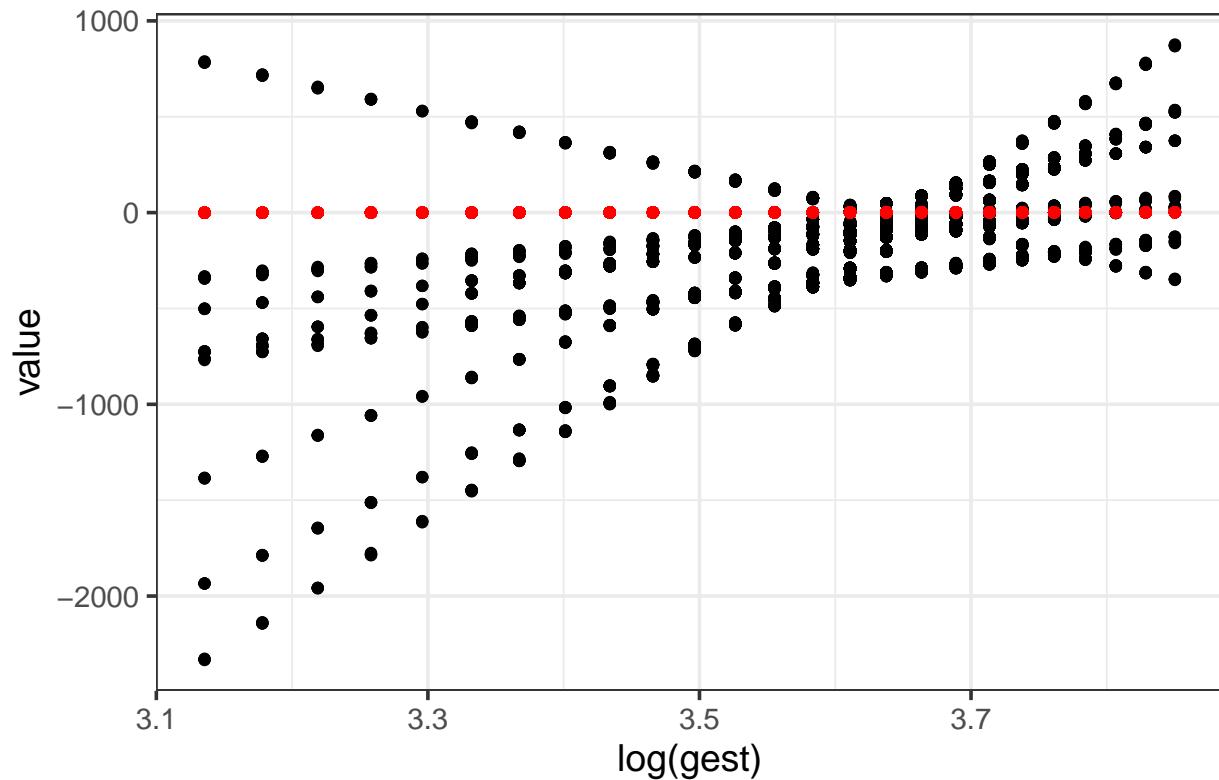
We have generated 4000 data sets for log-transformed birth weights. Here I am just plotting the first 10 against gestational age:

```

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")

```

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

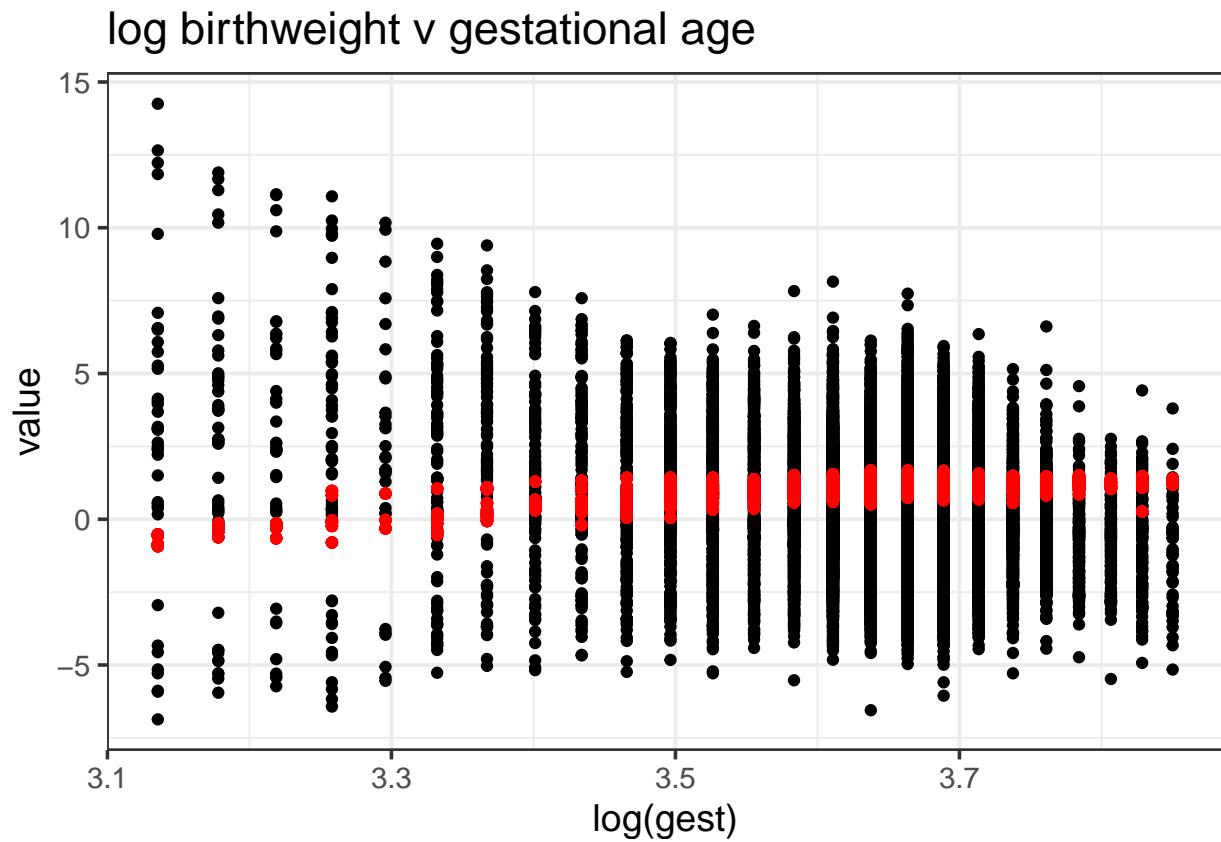
```
mod1_priorpredict2 <- brm(log_weight ~ 0 + Intercept + log_gest_c, data = ds,
                           seed = 1236,
                           prior = prior2,
                           sample_prior = "only", # we are drawing from the prior!
                           chains = 4,
                           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:

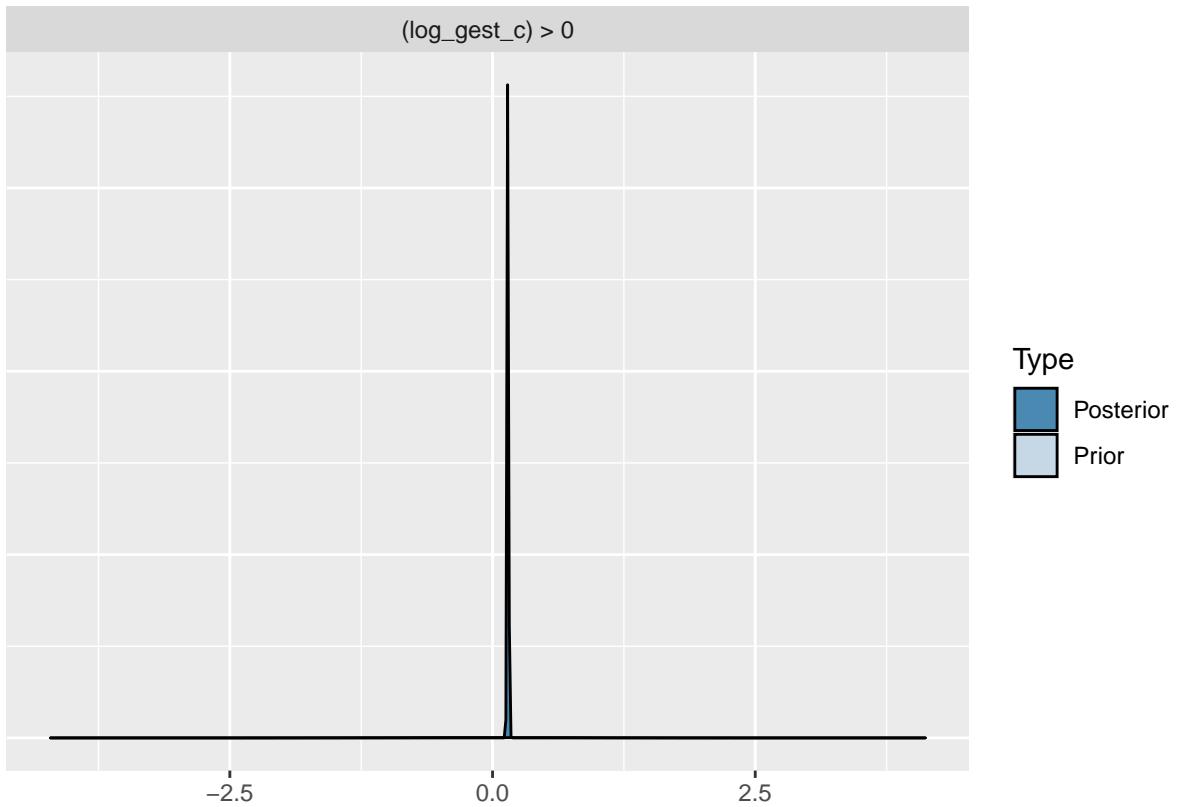
```

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")

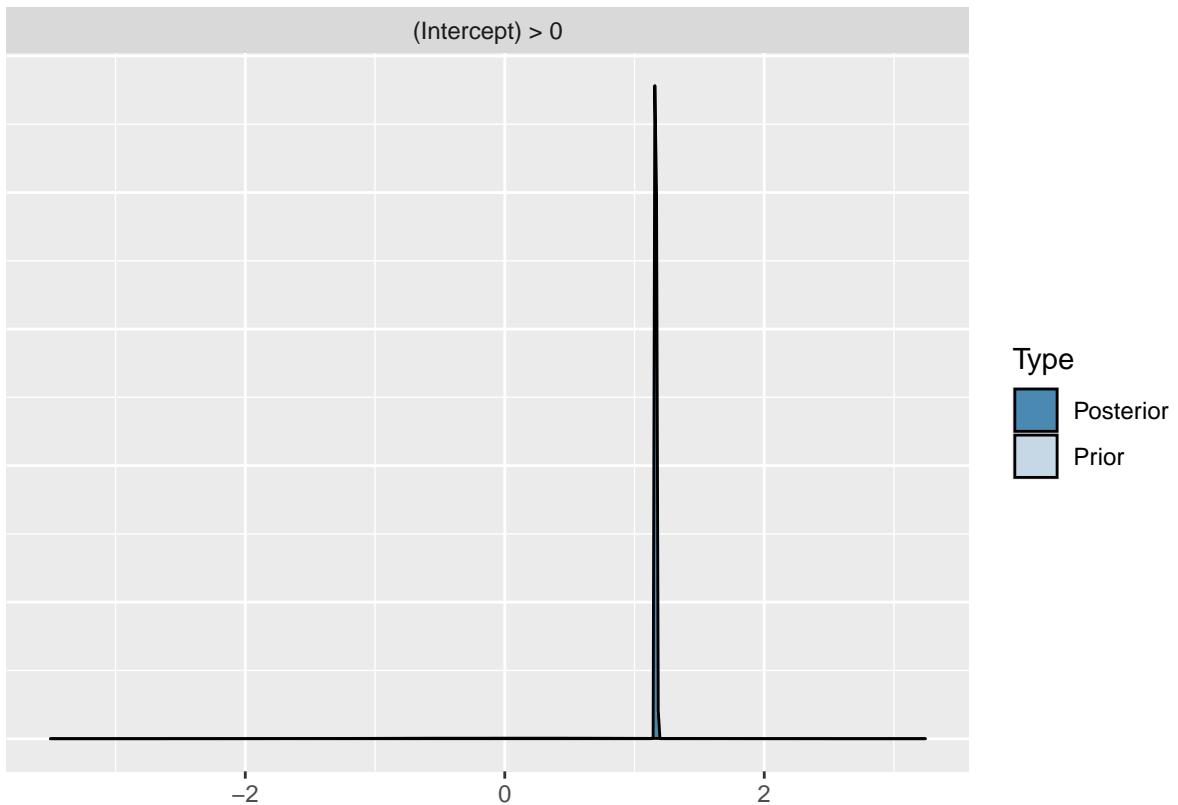
```

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"))
```



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

To

```

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

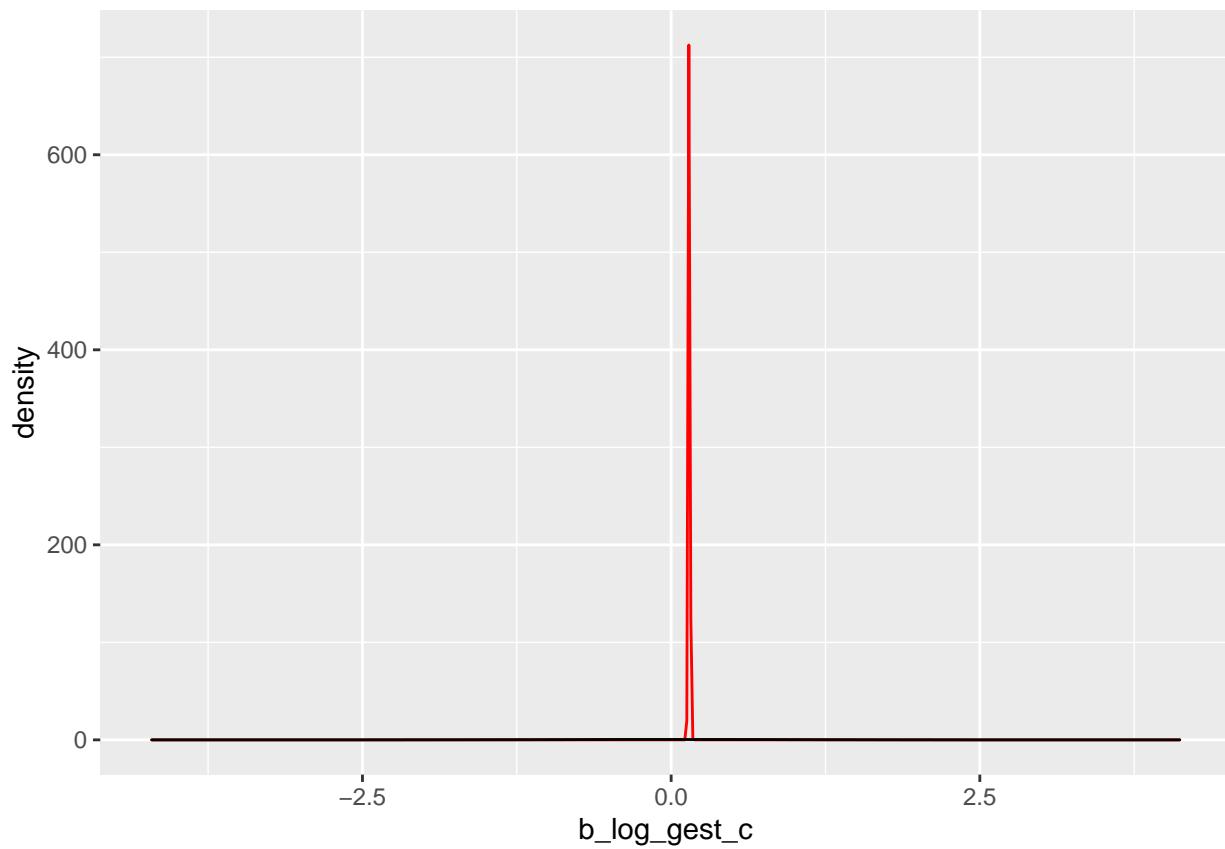
```

Now plot

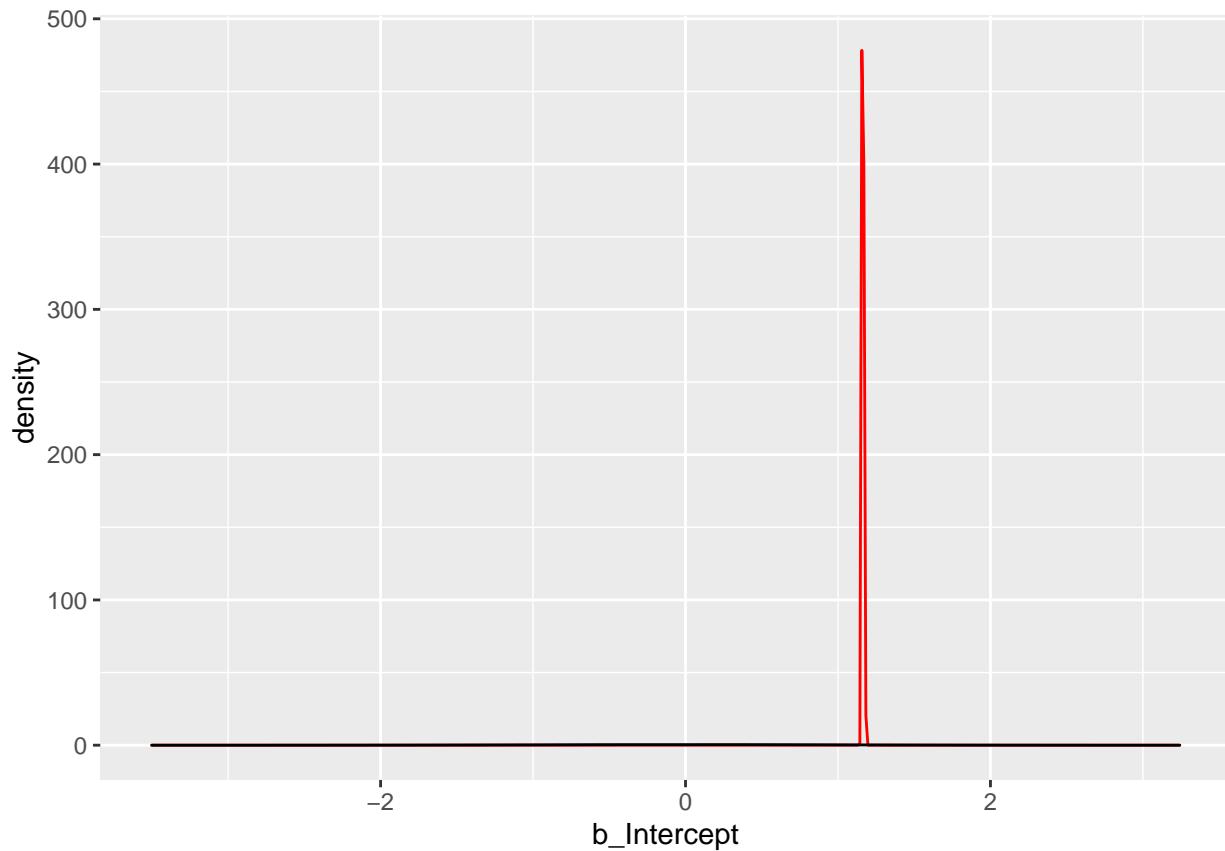
```

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

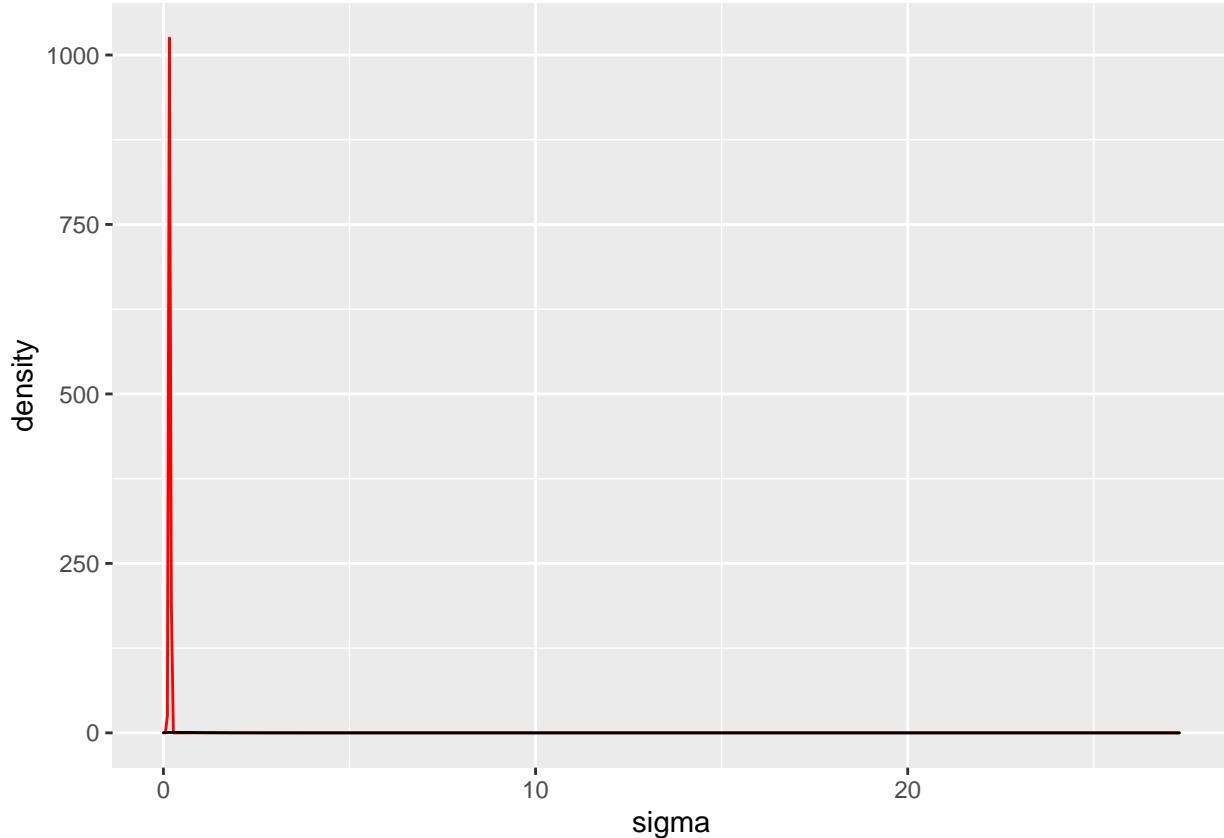
```



```
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.

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?

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 $T(\mathbf{y})$.

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.

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 i th entry referring to the ELPD for observation i .

Question 5: splines fit (extra credit)

Fit a splines model in brm and redo the posterior predictive checks to see if this model improves upon models 1 and/or 2.