Applied_Stat_Lab_6

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.2     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.4.4     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.0
v purrr     1.0.1
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

```
library(here)
```

```
here() starts at /Users/larskutschinski/Desktop/AppliedStats/AppliedStats22
```

```
# for bayes stuff
library(rstan)
```

```
Loading required package: StanHeaders

rstan version 2.32.5 (Stan version 2.32.2)

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)
```

```
For within-chain threading using `reduce_sum()` or `map_rect()` Stan functions,
change `threads_per_chain` option:
rstan_options(threads_per_chain = 1)


Attaching package: 'rstan'

The following object is masked from 'package:tidyr':

    extract
```

```
library(bayesplot)
```

```
This is bayesplot version 1.10.0
- 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
```

```
library(loo)
```

```
This is loo version 2.6.0
- 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 'co

Attaching package: 'loo'

The following object is masked from 'package:rstan':

    loo
```

```
library(tidybayes)

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    16         2     2  23   M           39  3.18 Y
2    25         7     2  43.6 M           40  4.14 Y
3    27         2     3  19.5 F           41  3.18 Y
4    26         1     3  21.5 F           36  3.40 Y
5    28         7     2  40.6 F           34  2.71 Y
6    31         7     3  29.3 M           35  3.52 Y
```

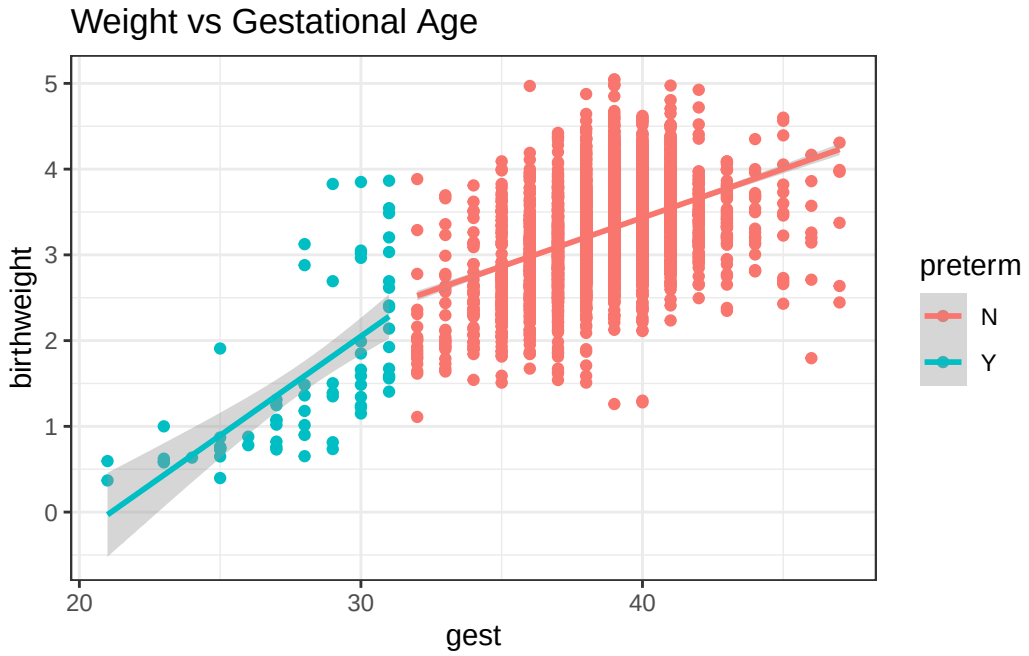
```
ds <- ds %>%
  rename(birthweight = dbwt, gest = combgest) %>%
  mutate(preterm = ifelse(gest<32, "Y", "N")) %>%
  filter(ilive=="Y",gest< 99, birthweight<9.999)
```

Question 1

(a)

```
ds %>%
  ggplot() +
  aes(x = gest, y = birthweight, color = preterm, group = preterm) +
  geom_point() +
  theme_bw() +
  geom_smooth(method = "lm") +
  labs(title = "Weight vs Gestational Age")
```

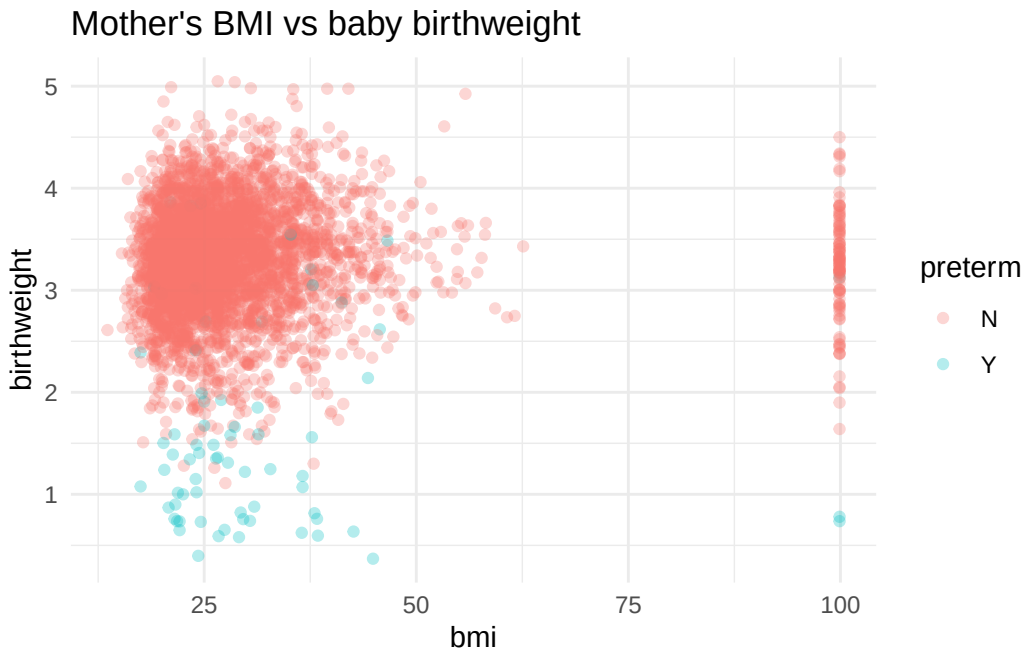
```
`geom_smooth()` using formula = 'y ~ x'
```



The gestation vs birthweight plot shows that there seems to be a difference in slopes for preterm vs full term babies for birthweight as a function of gestation. However since the cutoff for preterm is not data dependent but rather given it seems a bit arbitrary and moving the cutoff a bit to the right (>32) would result in slopes getting closer. Plus there is a big uncertainty in the line since there are significantly fewer datapoints for preterm babies.

(b)

```
ds %>%
  ggplot() +
  aes(x = bmi, y = birthweight, color = preterm) +
  geom_point(alpha = 0.3) +
  theme_minimal() +
  labs(title = "Mother's BMI vs baby birthweight")
```



With this plot we tried to see whether there was any effect of mother's bmi on birthweight. However the result seems a bit unreliable, since people with a bmi of 100 do not exist.

(c)

```
ds %>%
  mutate(race = as_factor(case_when(
    mracehisp == 1 ~ "NHW",
    mracehisp == 2 ~ "NHB",
    mracehisp == 3 ~ "NHAIAN",
    mracehisp == 4 ~ "NHA",
    mracehisp == 5 ~ "NHOPI",
    mracehisp == 6 ~ "Hisp >1 race",
    mracehisp == 7 ~ "Hisp",
    mracehisp == 8 ~ "Unknown"
  ))) %>%
  group_by(race, sex) %>%
  summarize(
    n = n(),
    mean_gest = mean(gest),
    med_gest = median(gest),
    var_gest = var(gest),
```

```
    mean_bw = mean(birthweight),
    med_bw = median(birthweight),
    var_bw = var(birthweight),
    num_preterm = sum(preterm == "Y"),
    prop_preterm = mean(preterm == "Y")
  ) %>%
  arrange(desc(n))
```

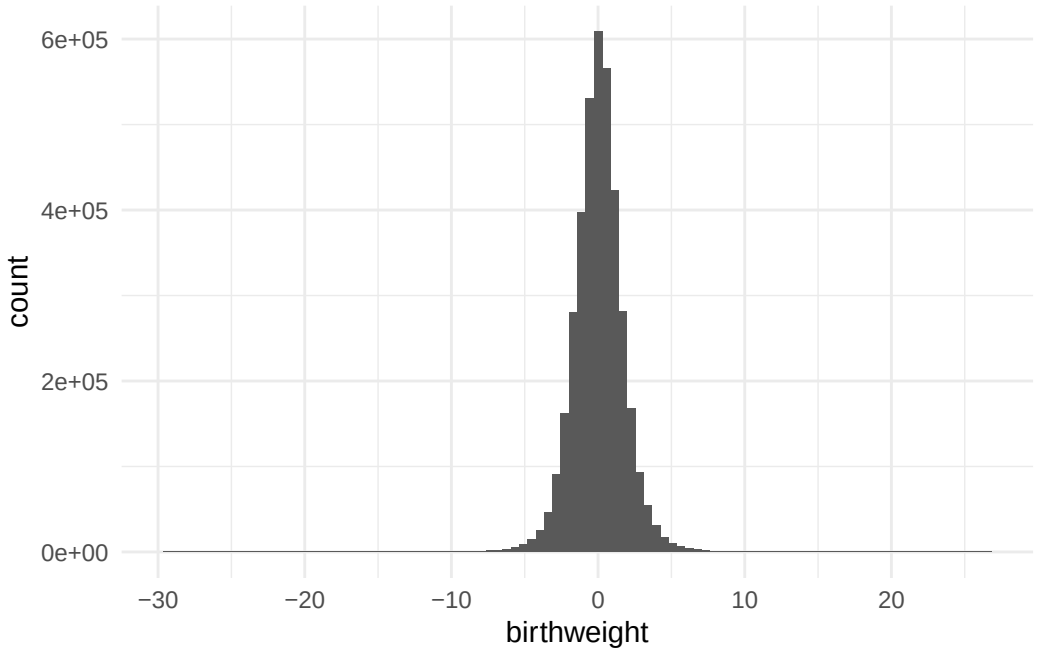
```
`summarise()` has grouped output by 'race'. You can override using the
`.groups` argument.
```

```
# A tibble: 16 x 11
# Groups:   race [8]
   race         sex       n mean_gest med_gest var_gest mean_bw med_bw var_bw
   <fct>        <chr> <int>     <dbl>    <dbl>    <dbl>   <dbl>  <dbl>  <dbl>
 1 NHW          M      1020      38.8     39       5.17    3.41   3.46  0.322
 2 NHW          F       995      38.8     39       5.20    3.27   3.31  0.316
 3 Hisp         F       447      38.6     39       6.03    3.19   3.23  0.305
 4 Hisp         M       426      38.4     39       5.54    3.30   3.33  0.313
 5 NHB          F       293      38.0     38       7.30    2.99   3.06  0.382
 6 NHB          M       281      38.2     39       7.39    3.17   3.18  0.425
 7 NHA          M       123      38.4     39       3.30    3.17   3.20  0.259
 8 NHA          F       118      38.6     39       5.65    3.16   3.14  0.245
 9 Hisp >1 race M        38      38.2     38       6.77    3.18   3.37  0.447
10 Hisp >1 race F        36      38.4     39       4.69    3.21   3.28  0.424
11 Unknown      F        17      38.8     39       4.28    3.25   3.3   0.297
12 Unknown      M        15      37.7     37       2.07    3.20   3.32  0.463
13 NHAIAN       F        14      38.4     39      27.3     3.10   3.18  0.946
14 NHAIAN       M        12      39.2     39       6.20    3.71   3.72  0.120
15 NHOPI        M         4      40.2     39.5    18.9     3.21   3.11  0.210
16 NHOPI        F         3      35       36       7       2.78   2.98  0.130
# i 2 more variables: num_preterm <int>, prop_preterm <dbl>
```

Given that we will be modeling data for babies and that we have access to the race of mothers and the sex of the kids, it is wise to investigate whether we might be favoring one group over another by not modeling them separately or by not including race and sex as a component to explain the disparities. In order to determine whether there are any obvious variations between the gestation period, birthweight, and number of observations in each group, we produced a numerical summary. There does not seem to be too much variation in things like proportion of babies born pre-term but there are some things we should try to look more in depth at like why is the variance of gestation times so much higher for NHB than others.

Question 2

```
set.seed(123)
nsims <- 1000
sigma <- abs(rnorm(nsims))
beta0 <- rnorm(nsims)
beta1 <- rnorm(nsims)

lgc <- ds %>% mutate(log_gests_centered = scale(log(gest))) %>% pull(log_gests_centered)
dsims <- tibble(log_gest_c = lgc)

for (i in 1:nsims) {
  this_mu <- beta0[i] + beta1[i] * dsims$log_gest_c
  dsims[paste0(i)] <- rnorm(3842, mean = this_mu, sd = sigma[i])
}


data_for_plot <- dsims %>% select(-log_gest_c) %>% pivot_longer(cols = everything())
data_for_plot %>%
  ggplot(aes(x = value)) +
  geom_histogram(bins = 100) +
  theme_minimal() +
  scale_x_continuous(name = "birthweight")
```



Question 3

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

stan_data <- list(N = nrow(ds),
                  log_weight = ds$log_weight,
                  log_gest = ds$log_gest_c)

mod1 <- stan(data = stan_data,
             file = here("stan", "simple_weight.stan"),
             iter = 500,
             seed = 243)
```

```
Trying to compile a simple C file


Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
using C compiler: ‘Apple clang version 14.0.3 (clang-1403.0.22.14.1)’
using SDK: ‘MacOSX13.3.sdk’
clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Libra
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/St
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/Rc
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/Rc
/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/
namespace Eigen {
              ^
              ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/St
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/Rc
/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/
#include <complex>
         ^~~~~~~~~
3 errors generated.
make: *** [foo.o] Error 1

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
```

```
Chain 1:
Chain 1: Gradient evaluation took 0.000117 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.17 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:   1 / 500 [  0%]  (Warmup)
Chain 1: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 1: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 1: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 1: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 1: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 1: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 1: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 1: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 1: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 1: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 1: Iteration: 500 / 500 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.204 seconds (Warm-up)
Chain 1:                0.16 seconds (Sampling)
Chain 1:                0.364 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 6.9e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.69 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:   1 / 500 [  0%]  (Warmup)
Chain 2: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 2: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 2: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 2: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 2: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 2: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 2: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 2: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 2: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 2: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 2: Iteration: 500 / 500 [100%]  (Sampling)
```

```
Chain 2:
Chain 2:  Elapsed Time: 0.198 seconds (Warm-up)
Chain 2:                0.184 seconds (Sampling)
Chain 2:                0.382 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 6.6e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.66 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:   1 / 500 [  0%]  (Warmup)
Chain 3: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 3: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 3: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 3: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 3: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 3: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 3: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 3: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 3: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 3: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 3: Iteration: 500 / 500 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.205 seconds (Warm-up)
Chain 3:                0.176 seconds (Sampling)
Chain 3:                0.381 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 7.2e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.72 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:   1 / 500 [  0%]  (Warmup)
Chain 4: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 4: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 4: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 4: Iteration: 200 / 500 [ 40%]  (Warmup)
```

```
Chain 4: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 4: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 4: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 4: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 4: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 4: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 4: Iteration: 500 / 500 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0.193 seconds (Warm-up)
Chain 4:                0.169 seconds (Sampling)
Chain 4:                0.362 seconds (Total)
Chain 4:

Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians
Running the chains for more iterations may help. See
https://mc-stan.org/misc/warnings.html#bulk-ess

Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tai
Running the chains for more iterations may help. See
https://mc-stan.org/misc/warnings.html#tail-ess
```

```r
summary(mod1)$summary[c("beta[1]", "beta[2]", "sigma"),]
```

```
            mean      se_mean          sd      2.5%       25%       50%
beta[1] 1.1626293 8.109795e-05 0.002794886 1.1571530 1.1607401 1.1626100
beta[2] 0.1437074 7.050797e-05 0.002760482 0.1381359 0.1418908 0.1436313
sigma   0.1688448 1.025235e-04 0.001845326 0.1652251 0.1675565 0.1689364
             75%     97.5%    n_eff      Rhat
beta[1] 1.1646213 1.1679552 1187.705 0.9970491
beta[2] 0.1454944 0.1491751 1532.828 0.9972723
sigma   0.1700804 0.1722141  323.966 1.0095667
```

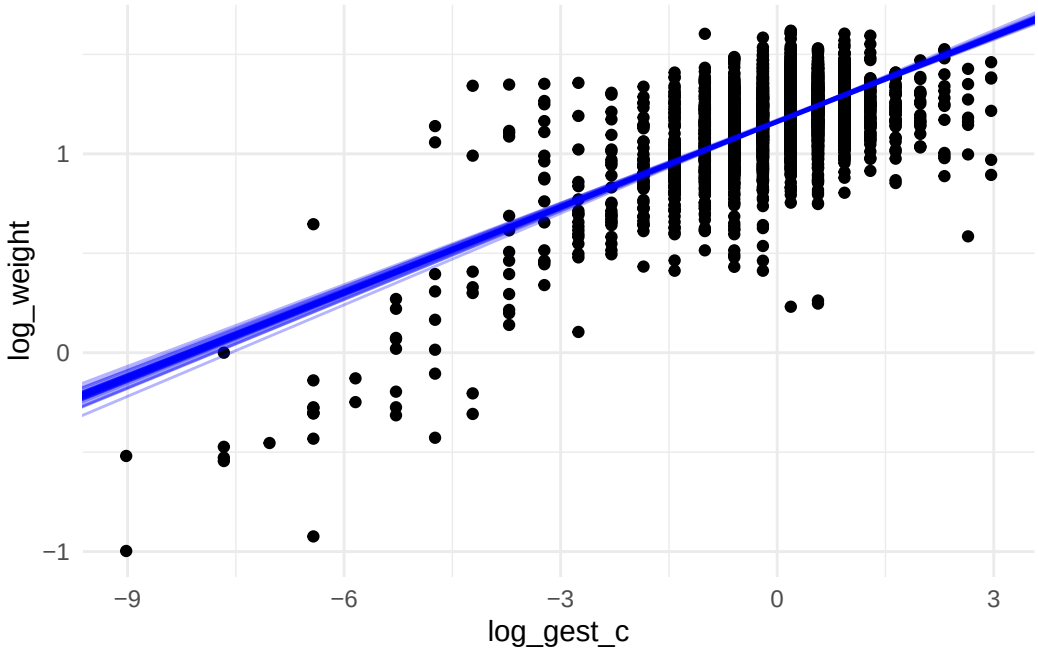
```r
mean_gest_37 <- (log(37) - mean(log(ds$gest)))/sd(log(ds$gest))
pred_37 <- 1.1626293 + 0.1437074 * mean_gest_37
pred_37 <- exp(pred_37)
pred_37
```

```
[1] 2.936397
```

A baby born at a gestational age of 37 weeks has an estimated birthweight of 2.936397

Question 4

```
posterior_draws <- extract(mod1, pars = c("beta[1]", "beta[2]"))
sample_indices <- sample(1:length(posterior_draws$`beta[1]`), 50)
beta_1_draws <- posterior_draws$`beta[1]`[sample_indices]
beta_2_draws <- posterior_draws$`beta[2]`[sample_indices]

p <- ggplot(ds, aes(x = log_gest_c, y = log_weight)) +
  geom_point() +
  theme_minimal()

for(i in 1:50){
  beta_1 <- beta_1_draws[i]
  beta_2 <- beta_2_draws[i]


  p <- p + geom_abline(intercept = beta_1, slope = beta_2, colour = "blue", alpha = 0.3)
}
print(p)
```



Question 5

```r
preterm <- ifelse(ds$preterm == "Y", 1, 0)

stan_data[["preterm"]] <- preterm

mod2 <- stan(data = stan_data,
             file = here("stan", "simple_weight_interaction.stan"),
             iter = 250,
             seed = 243)
```

```
Trying to compile a simple C file


Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
using C compiler: ‘Apple clang version 14.0.3 (clang-1403.0.22.14.1)’
using SDK: ‘MacOSX13.3.sdk’
clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Libra
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/St
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/Rc
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/Rc
/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/
namespace Eigen {
               ^
               ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/St
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/Rc
/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/
#include <complex>
         ^~~~~~~~~
3 errors generated.
make: *** [foo.o] Error 1

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.00012 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.2 seconds.
```

```
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: WARNING: There aren't enough warmup iterations to fit the
Chain 1:          three stages of adaptation as currently configured.
Chain 1:          Reducing each adaptation stage to 15%/75%/10% of
Chain 1:          the given number of warmup iterations:
Chain 1:            init_buffer = 18
Chain 1:            adapt_window = 95
Chain 1:            term_buffer = 12
Chain 1:
Chain 1: Iteration:   1 / 250 [  0%]  (Warmup)
Chain 1: Iteration:  25 / 250 [ 10%]  (Warmup)
Chain 1: Iteration:  50 / 250 [ 20%]  (Warmup)
Chain 1: Iteration:  75 / 250 [ 30%]  (Warmup)
Chain 1: Iteration: 100 / 250 [ 40%]  (Warmup)
Chain 1: Iteration: 125 / 250 [ 50%]  (Warmup)
Chain 1: Iteration: 126 / 250 [ 50%]  (Sampling)
Chain 1: Iteration: 150 / 250 [ 60%]  (Sampling)
Chain 1: Iteration: 175 / 250 [ 70%]  (Sampling)
Chain 1: Iteration: 200 / 250 [ 80%]  (Sampling)
Chain 1: Iteration: 225 / 250 [ 90%]  (Sampling)
Chain 1: Iteration: 250 / 250 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 1.94 seconds (Warm-up)
Chain 1:                3.133 seconds (Sampling)
Chain 1:                5.073 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 7.1e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.71 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: WARNING: There aren't enough warmup iterations to fit the
Chain 2:          three stages of adaptation as currently configured.
Chain 2:          Reducing each adaptation stage to 15%/75%/10% of
Chain 2:          the given number of warmup iterations:
Chain 2:            init_buffer = 18
Chain 2:            adapt_window = 95
Chain 2:            term_buffer = 12
```

```
Chain 2:
Chain 2: Iteration:   1 / 250 [  0%]  (Warmup)
Chain 2: Iteration:  25 / 250 [ 10%]  (Warmup)
Chain 2: Iteration:  50 / 250 [ 20%]  (Warmup)
Chain 2: Iteration:  75 / 250 [ 30%]  (Warmup)
Chain 2: Iteration: 100 / 250 [ 40%]  (Warmup)
Chain 2: Iteration: 125 / 250 [ 50%]  (Warmup)
Chain 2: Iteration: 126 / 250 [ 50%]  (Sampling)
Chain 2: Iteration: 150 / 250 [ 60%]  (Sampling)
Chain 2: Iteration: 175 / 250 [ 70%]  (Sampling)
Chain 2: Iteration: 200 / 250 [ 80%]  (Sampling)
Chain 2: Iteration: 225 / 250 [ 90%]  (Sampling)
Chain 2: Iteration: 250 / 250 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 2.411 seconds (Warm-up)
Chain 2:                3.042 seconds (Sampling)
Chain 2:                5.453 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 8.1e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.81 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: WARNING: There aren't enough warmup iterations to fit the
Chain 3:          three stages of adaptation as currently configured.
Chain 3:          Reducing each adaptation stage to 15%/75%/10% of
Chain 3:          the given number of warmup iterations:
Chain 3:            init_buffer = 18
Chain 3:            adapt_window = 95
Chain 3:            term_buffer = 12
Chain 3:
Chain 3: Iteration:   1 / 250 [  0%]  (Warmup)
Chain 3: Iteration:  25 / 250 [ 10%]  (Warmup)
Chain 3: Iteration:  50 / 250 [ 20%]  (Warmup)
Chain 3: Iteration:  75 / 250 [ 30%]  (Warmup)
Chain 3: Iteration: 100 / 250 [ 40%]  (Warmup)
Chain 3: Iteration: 125 / 250 [ 50%]  (Warmup)
Chain 3: Iteration: 126 / 250 [ 50%]  (Sampling)
Chain 3: Iteration: 150 / 250 [ 60%]  (Sampling)
Chain 3: Iteration: 175 / 250 [ 70%]  (Sampling)
```

```
Chain 3: Iteration: 200 / 250 [ 80%]  (Sampling)
Chain 3: Iteration: 225 / 250 [ 90%]  (Sampling)
Chain 3: Iteration: 250 / 250 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 2.228 seconds (Warm-up)
Chain 3:                3.105 seconds (Sampling)
Chain 3:                5.333 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 7.7e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.77 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: WARNING: There aren't enough warmup iterations to fit the
Chain 4:          three stages of adaptation as currently configured.
Chain 4:          Reducing each adaptation stage to 15%/75%/10% of
Chain 4:          the given number of warmup iterations:
Chain 4:            init_buffer = 18
Chain 4:            adapt_window = 95
Chain 4:            term_buffer = 12
Chain 4:
Chain 4: Iteration:   1 / 250 [  0%]  (Warmup)
Chain 4: Iteration:  25 / 250 [ 10%]  (Warmup)
Chain 4: Iteration:  50 / 250 [ 20%]  (Warmup)
Chain 4: Iteration:  75 / 250 [ 30%]  (Warmup)
Chain 4: Iteration: 100 / 250 [ 40%]  (Warmup)
Chain 4: Iteration: 125 / 250 [ 50%]  (Warmup)
Chain 4: Iteration: 126 / 250 [ 50%]  (Sampling)
Chain 4: Iteration: 150 / 250 [ 60%]  (Sampling)
Chain 4: Iteration: 175 / 250 [ 70%]  (Sampling)
Chain 4: Iteration: 200 / 250 [ 80%]  (Sampling)
Chain 4: Iteration: 225 / 250 [ 90%]  (Sampling)
Chain 4: Iteration: 250 / 250 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 2.339 seconds (Warm-up)
Chain 4:                3.022 seconds (Sampling)
Chain 4:                5.361 seconds (Total)
Chain 4:

Warning: There were 90 transitions after warmup that exceeded the maximum treedepth. Increase
```

```
https://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded

Warning: Examine the pairs() plot to diagnose sampling problems

Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians
Running the chains for more iterations may help. See
https://mc-stan.org/misc/warnings.html#bulk-ess

Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tai
Running the chains for more iterations may help. See
https://mc-stan.org/misc/warnings.html#tail-ess
```

```r
summary(mod2)$summary[c(paste0("beta[", 1:4, "]"), "sigma"),]
```

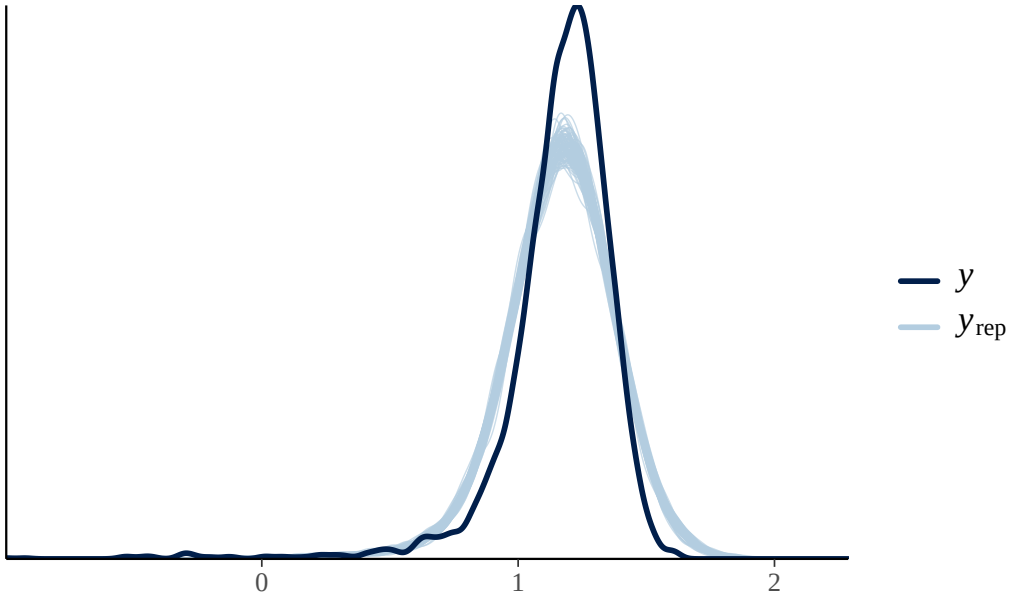
```
              mean      se_mean          sd       2.5%        25%         50%
beta[1]  1.16262773 0.0001173868 0.002616419  1.1574284  1.1607501  1.16264393
beta[2]  0.14396044 0.0001059324 0.002484259  0.1391578  0.1424342  0.14390417
beta[3] -0.11831873 0.0923725461 1.016421193 -2.2539745 -0.8357009 -0.08310194
beta[4]  0.08188719 0.1002080332 0.977482549 -2.0153036 -0.5034171  0.08991635
sigma    0.16899002 0.0001067829 0.002134460  0.1650126  0.1675804  0.16893586
              75%     97.5%     n_eff      Rhat
beta[1] 1.1645209 1.1675101 496.79352 1.0004793
beta[2] 0.1456240 0.1486601 549.96642 0.9970058
beta[3] 0.6111934 1.4864710 121.07698 1.0250350
beta[4] 0.6007094 2.2113658  95.15091 1.0318872
sigma   0.1704295 0.1729181 399.55167 1.0067583
```

The value of the estimated interaction term coefficient is quite low at 0.08188719, indicating that it does not significantl contribute to birthweight.

Question 6

```r
set.seed(1856)
y <- ds$log_weight
yrep1 <- extract(mod1)[["log_weight_rep"]]
yrep2 <- extract(mod2)[["log_weight_rep"]] # will need mod2 for later
samp100 <- sample(nrow(yrep1), 100)
ppc_dens_overlay(y, yrep1[samp100, ])  + ggtitle("distribution of observed versus predicte
```

distribution of observed versus predicted birthweights



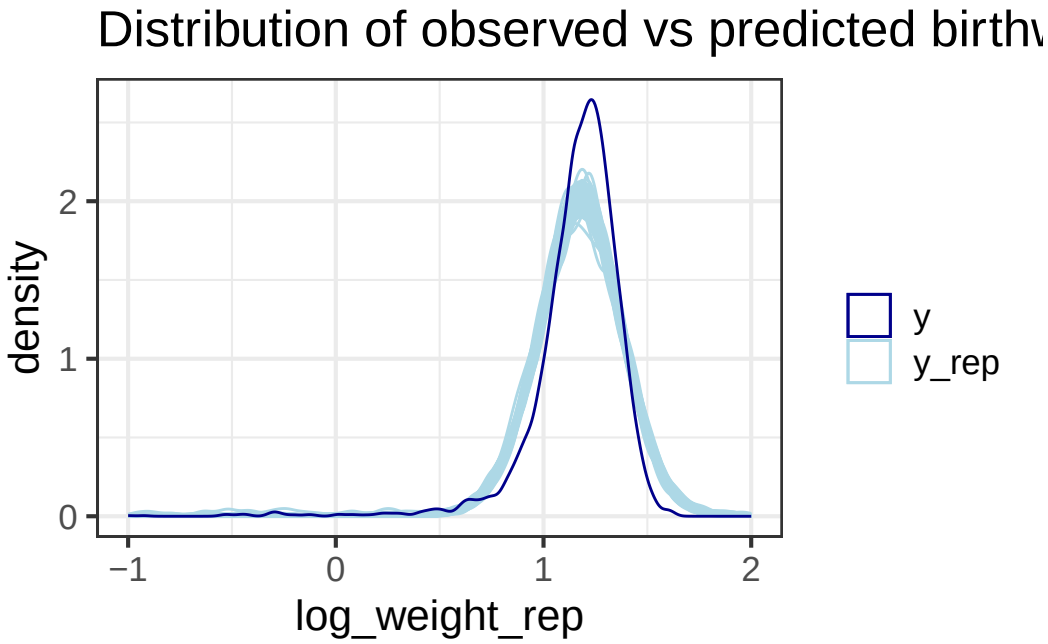

```
samp100_2 <- sample(nrow(yrep2), 100)
y2 <- as_tibble(t(yrep2[samp100_2, ]))
```

```
Warning: The `x` argument of `as_tibble.matrix()` must have unique column names if
`.name_repair` is omitted as of tibble 2.0.0.
i Using compatibility `.name_repair`.
```

```
colnames(y2) <- 1:100

dr <- as_tibble(y2)
dr <- dr %>% bind_cols(i = 1:3842, log_weight_obs = log(ds$birthweight))

dr <- dr %>%
  pivot_longer(`1`:`100`, names_to = "sim", values_to = "log_weight_rep")

dr %>%
  ggplot(aes(log_weight_rep, group = sim)) +
  geom_density(alpha = 0.2, aes(color = "y_rep")) +
  geom_density(data = ds %>% mutate(sim = 1),
               aes(x = log(birthweight), col = "y")) +
  scale_color_manual(name = "",
                     values = c("y" = "darkblue",
```

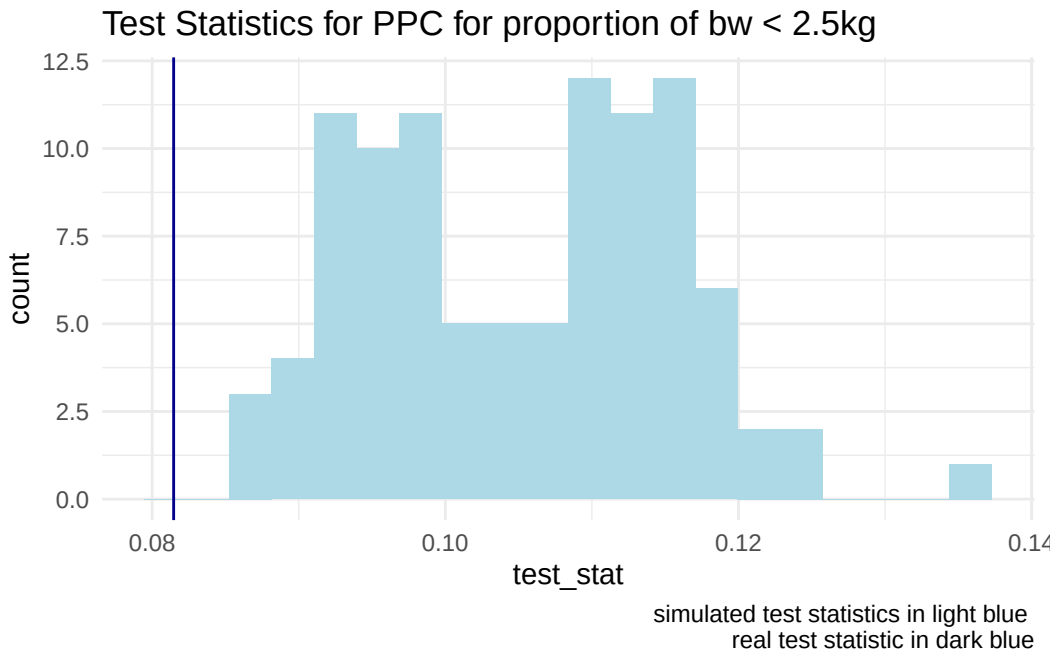
```
                                  "y_rep" = "lightblue")) +
  ggtitle("Distribution of observed vs predicted birthweights") +
  theme_bw(base_size = 16) +
  xlim(-1, 2)
```

```
Warning: Removed 5258 rows containing non-finite values (`stat_density()`).
```



Question 7

```
test_stat_real <- mean(ds$birthweight < 2.5)
test_stat_rep <- dr %>% group_by(sim) %>%
  summarize(test_stat = mean(exp(log_weight_rep) < 2.5))

test_stat_rep %>%
  ggplot(aes(x = test_stat)) +
  geom_histogram(bins = 20, fill = "lightblue") +
  geom_vline(xintercept = test_stat_real, color = "darkblue") +
  theme_minimal() +
  labs(caption = "simulated test statistics in light blue \n real test statistic in dark b
       title = "Test Statistics for PPC for proportion of bw < 2.5kg")
```



Question 8

```
loglik1 <- extract(mod1)[["log_lik"]]
loglik2 <- extract(mod2)[["log_lik"]]

loo1 <- loo(loglik1, save_psis = TRUE)
```

```
Warning: Relative effective sample sizes ('r_eff' argument) not specified.
For models fit with MCMC, the reported PSIS effective sample sizes and
MCSE estimates will be over-optimistic.
```

```
loo2 <- loo(loglik2, save_psis = TRUE)
```

```
Warning: Relative effective sample sizes ('r_eff' argument) not specified.
For models fit with MCMC, the reported PSIS effective sample sizes and
MCSE estimates will be over-optimistic.
```

```
loo_compare(loo1, loo2)
```

```
       elpd_diff se_diff
model1       0.0       0.0
model2 -269843.3   38071.7
```

After comparing the two models using loo_compare we see that model 2 is better.

```
stan_data <- list(N = nrow(ds),
                  log_weight = ds$log_weight,
                  log_gest = ds$log_gest_c,
                  bmi = ds$bmi,
                  mager = ds$mager)

mod3 <- stan(data = stan_data,
             file = here("stan", "simple_weight_add_covariates.stan"),
             iter = 500,
             seed = 243)
```

```
Trying to compile a simple C file


Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
using C compiler: ‘Apple clang version 14.0.3 (clang-1403.0.22.14.1)’
using SDK: ‘MacOSX13.3.sdk’
clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Libra
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/St
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/Rc
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/Rc
/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/
namespace Eigen {
              ^
              ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/St
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/Rc
/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/
#include <complex>
         ^~~~~~~~~
3 errors generated.
```

```
make: *** [foo.o] Error 1

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000272 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 2.72 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:   1 / 500 [  0%]  (Warmup)
Chain 1: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 1: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 1: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 1: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 1: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 1: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 1: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 1: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 1: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 1: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 1: Iteration: 500 / 500 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.454 seconds (Warm-up)
Chain 1:                1.122 seconds (Sampling)
Chain 1:                1.576 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000144 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 1.44 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:   1 / 500 [  0%]  (Warmup)
Chain 2: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 2: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 2: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 2: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 2: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 2: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 2: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 2: Iteration: 350 / 500 [ 70%]  (Sampling)
```

```
Chain 2: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 2: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 2: Iteration: 500 / 500 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 2.098 seconds (Warm-up)
Chain 2:                1.312 seconds (Sampling)
Chain 2:                3.41 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000136 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 1.36 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:   1 / 500 [  0%]  (Warmup)
Chain 3: Iteration:  50 / 500 [ 10%]  (Warmup)
Chain 3: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 3: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 3: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 3: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 3: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 3: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 3: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 3: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 3: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 3: Iteration: 500 / 500 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 1.63 seconds (Warm-up)
Chain 3:                0.794 seconds (Sampling)
Chain 3:                2.424 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000147 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 1.47 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:   1 / 500 [  0%]  (Warmup)
Chain 4: Iteration:  50 / 500 [ 10%]  (Warmup)
```

```
Chain 4: Iteration: 100 / 500 [ 20%]  (Warmup)
Chain 4: Iteration: 150 / 500 [ 30%]  (Warmup)
Chain 4: Iteration: 200 / 500 [ 40%]  (Warmup)
Chain 4: Iteration: 250 / 500 [ 50%]  (Warmup)
Chain 4: Iteration: 251 / 500 [ 50%]  (Sampling)
Chain 4: Iteration: 300 / 500 [ 60%]  (Sampling)
Chain 4: Iteration: 350 / 500 [ 70%]  (Sampling)
Chain 4: Iteration: 400 / 500 [ 80%]  (Sampling)
Chain 4: Iteration: 450 / 500 [ 90%]  (Sampling)
Chain 4: Iteration: 500 / 500 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 1.892 seconds (Warm-up)
Chain 4:                1.797 seconds (Sampling)
Chain 4:                3.689 seconds (Total)
Chain 4:
```

```
Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians
Running the chains for more iterations may help. See
https://mc-stan.org/misc/warnings.html#bulk-ess
```

```
Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tai
Running the chains for more iterations may help. See
https://mc-stan.org/misc/warnings.html#tail-ess
```

```r
summary(mod3)$summary[c(paste0("beta[", 1:4, "]"), "sigma"),]
```

```
               mean      se_mean           sd         2.5%          25%
beta[1] 1.0755440140 7.557830e-04 0.0158173745 1.045193e+00 1.0650856723
beta[2] 0.1447572001 1.765947e-04 0.0026537554 1.396860e-01 0.1430014380
beta[3] 0.0004401072 6.210619e-06 0.0002007180 5.534001e-05 0.0002941533
beta[4] 0.0025668487 2.257211e-05 0.0004952075 1.571843e-03 0.0022232739
sigma   0.1681065956 7.439571e-05 0.0019588047 1.645045e-01 0.1669161791
                50%          75%        97.5%     n_eff      Rhat
beta[1] 1.0749223224 1.0855872031 1.1068234715  438.0005 0.9976062
beta[2] 0.1447623171 0.1466073705 0.1498534898  225.8222 1.0223186
beta[3] 0.0004396445 0.0005817142 0.0008148524 1044.4861 1.0001042
beta[4] 0.0025951160 0.0028860635 0.0034578611  481.3158 0.9982623
sigma   0.1681782766 0.1693405253 0.1719532792  693.2446 1.0030718
```

```r
loglik3 <- extract(mod3)[["log_lik"]]
loo3 <- loo(loglik3, save_psis = TRUE)
```

```
Warning: Relative effective sample sizes ('r_eff' argument) not specified.
For models fit with MCMC, the reported PSIS effective sample sizes and
MCSE estimates will be over-optimistic.
```

```
Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') i
```

```r
loo_compare(loo3, loo2)
```

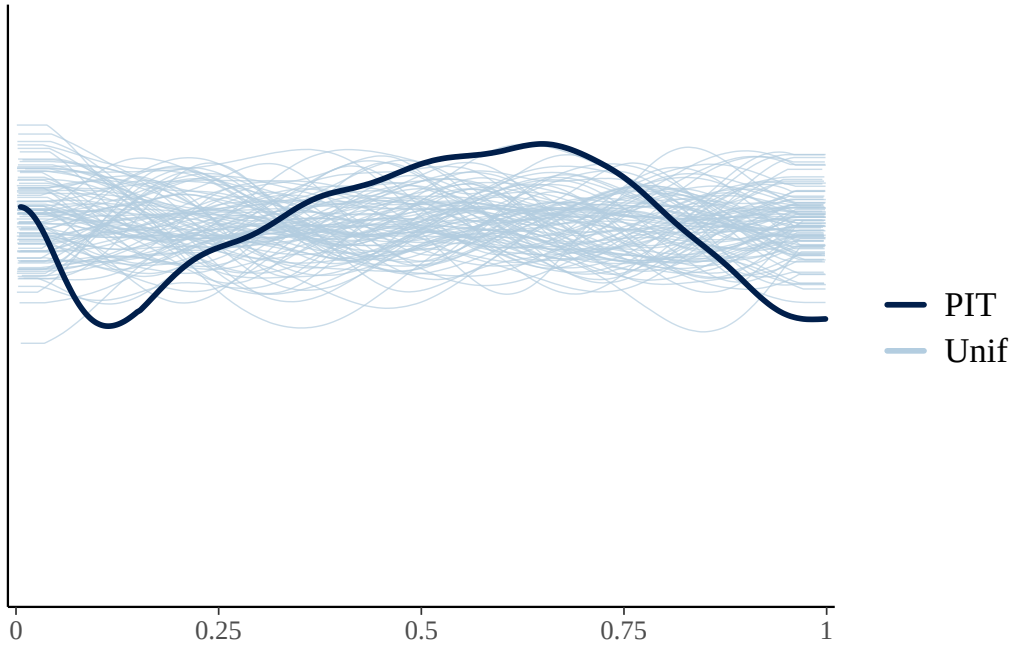
```
       elpd_diff se_diff
model1       0.0       0.0
model2 -269858.9   38071.5
```

We see that model 2 is better according to the loo_compare function. Let's compare the LOO-PIT for each model to compare the goodness of fit.

LOO-PIT for model 2:

```r
ppc_loo_pit_overlay(yrep = yrep2, y = y, lw = weights(loo2$psis_object))
```

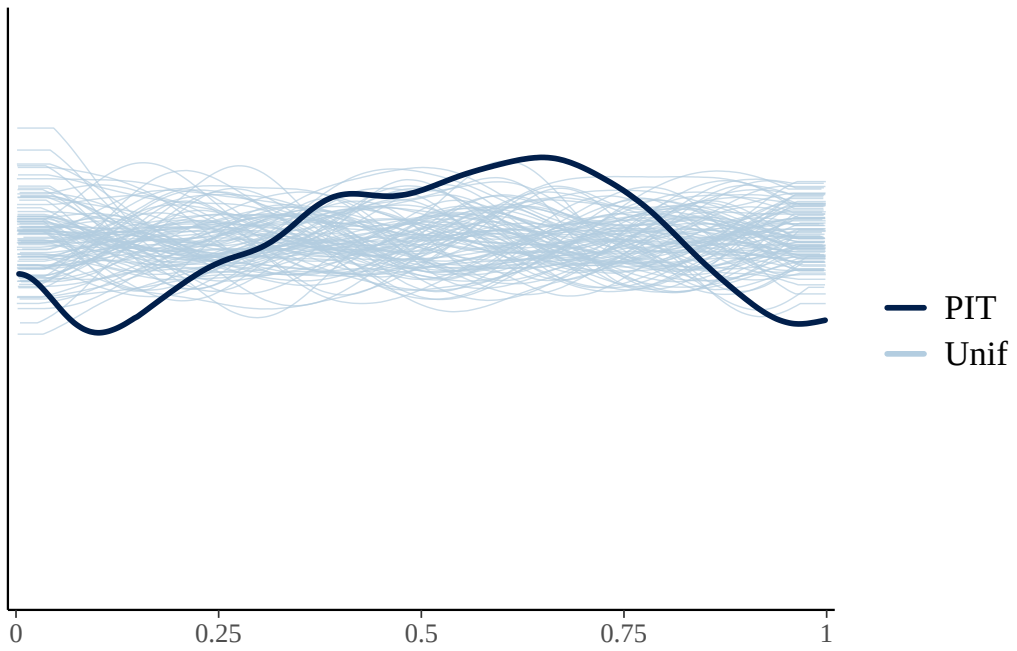
```
NOTE: The kernel density estimate assumes continuous observations and is not optimal for dis
```



LOO-PIT for model 3

```
yrep3 <- extract(mod3)[["log_weight_rep"]]
ppc_loo_pit_overlay(yrep = yrep3, y = y, lw = weights(loo3$psis_object))
```

```
NOTE: The kernel density estimate assumes continuous observations and is not optimal for dis
```



The LOO-PIT values for model 2 seem to look slightly more uniform, but it is hard to tell in this case.