

# Week 6: Visualizing the Bayesian Workflow

14/02/24

## Introduction

This lab will be looking at trying to replicate some of the visualizations in the lecture notes, involving prior and posterior predictive checks, and LOO model comparisons.

The dataset is a 0.1% of all births in the US in 2017. I've pulled out a few different variables, but as in the lecture, we'll just focus on birth weight and gestational age.

## The data

Read it in, along with all our packages.

```
library(tidyverse)
library(here)
# for bayes stuff
library(rstan)
library(bayesplot)
library(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

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

I'm going 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.

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

## Question 1

Use plots or tables to show three interesting observations about the data. Remember:

- Explain what your graph/ tables show
- Choose a graph type that's appropriate to the data type
- If you use `geom_smooth`, please also plot the underlying data

Feel free to replicate one of the scatter plots in the lectures as one of the interesting observations, as those form the basis of our models.

## The model

As in lecture, we will look at two candidate models

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

$$\log(y_i) \sim N(\beta_1 + \beta_2 \log(x_i), \sigma^2)$$

Model 2 has an interaction term between gestation and prematurity

$$\log(y_i) \sim N(\beta_1 + \beta_2 \log(x_i) + \beta_3 z_i + \beta_4 \log(x_i) z_i, \sigma^2)$$

- $y_i$  is weight in kg
- $x_i$  is gestational age in weeks, CENTERED AND STANDARDIZED
- $z_i$  is preterm (0 or 1, if gestational age is less than 32 weeks)

## Prior predictive checks

Let's put some weakly informative priors on all parameters i.e. for the  $\beta$ s

$$\beta \sim N(0, 1)$$

and for  $\sigma$

$$\sigma \sim N^+(0, 1)$$

where the plus means positive values only i.e. Half Normal.

Let's check to see what the resulting distribution of birth weights look like given Model 1 and the priors specified above, assuming we had no data on birth weight (but observations of gestational age).

## Question 2

For Model 1, simulate values of  $\beta$ s and  $\sigma$  based on the priors above. Do 1000 simulations. Use these values to simulate (log) birth weights from the likelihood specified in Model 1, based on the set of observed gestational weights. **Remember the gestational weights should be centered and standardized.**

- Plot the resulting distribution of simulated (log) birth weights.
- Plot ten simulations of (log) birthweights against gestational age.

## Run the model

Now we're going to run Model 1 in Stan. The stan code is in the `code/models` folder.

First, get our data into right form for input into stan.

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

# put into a list
stan_data <- list(N = nrow(ds),
                  log_weight = ds$log_weight,
                  log_gest = ds$log_gest_c)
```

Now fit the model

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

SAMPLING FOR MODEL 'simple\_weight' NOW (CHAIN 1).

Chain 1:

Chain 1: Gradient evaluation took 0.000228 seconds

Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 2.28 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: 0.406295 seconds (Warm-up)  
Chain 1: 0.341219 seconds (Sampling)  
Chain 1: 0.747514 seconds (Total)  
Chain 1:

SAMPLING FOR MODEL 'simple\_weight' NOW (CHAIN 2).

Chain 2:  
Chain 2: Gradient evaluation took 0.000158 seconds  
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 1.58 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: 0.371259 seconds (Warm-up)  
Chain 2: 0.358343 seconds (Sampling)  
Chain 2: 0.729602 seconds (Total)  
Chain 2:

SAMPLING FOR MODEL 'simple\_weight' NOW (CHAIN 3).

Chain 3:  
Chain 3: Gradient evaluation took 0.00016 seconds  
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 1.6 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: 0.376515 seconds (Warm-up)
Chain 3:           0.333709 seconds (Sampling)
Chain 3:           0.710224 seconds (Total)
Chain 3:

```

SAMPLING FOR MODEL 'simple\_weight' NOW (CHAIN 4).

```

Chain 4:
Chain 4: Gradient evaluation took 0.000139 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 1.39 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: 0.366856 seconds (Warm-up)
Chain 4:           0.349966 seconds (Sampling)
Chain 4:           0.716822 seconds (Total)
Chain 4:

```

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

	mean	se_mean	sd	2.5%	25%	50%
beta[1]	1.1626250	7.634607e-05	0.002583881	1.1575321	1.1609497	1.1626383

```

beta[2] 0.1436183 8.105504e-05 0.002791943 0.1380281 0.1417563 0.1436199
sigma   0.1689127 1.051837e-04 0.001979909 0.1650908 0.1676042 0.1688619
              75%      97.5%      n_eff      Rhat
beta[1] 1.1643919 1.1677313 1145.4383 0.9970543
beta[2] 0.1455075 0.1489575 1186.4598 0.9984953
sigma   0.1701148 0.1728405 354.3181 1.0046933

```

### Question 3

Based on Model 1, give an estimate of the expected birthweight of a baby who was born at a gestational age of 37 weeks.

### Question 4

Based on Model 1, create a scatter plot showing the underlying data (on the appropriate scale) and 50 posterior draws of the linear predictor.

### Question 5

Write a Stan model to run Model 2, and run it. Report a summary of the results, and interpret the coefficient estimate on the interaction term.

## PPCs

Now we've run two candidate models let's do some posterior predictive checks. The `bayesplot` package has a lot of inbuilt graphing functions to do this. For example, let's plot the distribution of our data (`y`) against 100 different datasets drawn from the posterior predictive distribution:

```

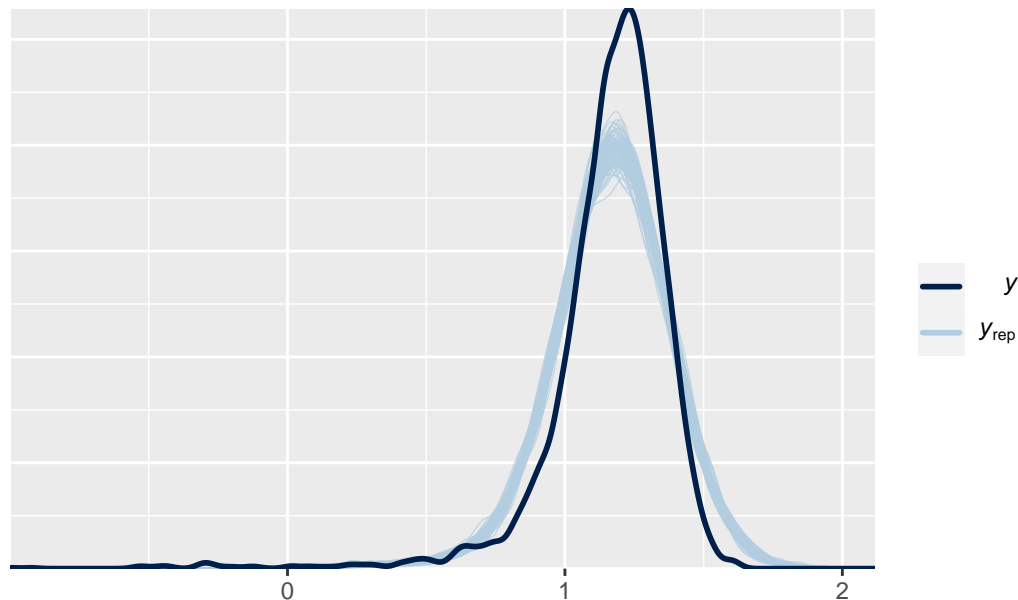
set.seed(1856)
y <- ds$log_weight
yrep1 <- extract(mod1)[["log_weight_rep"]]
dim(yrep1)

```

```
[1] 1000 3842
```

```
samp100 <- sample(nrow(yrep1), 100)
ppc_dens_overlay(y, yrep1[samp100, ]) + ggtitle("distribution of observed versus predicted")
```

distribution of observed versus predicted birthweights



## Question 6

Make a similar plot to the one above but for Model 2, and **not** using the bayes plot in built function (i.e. do it yourself just with `geom_density`)

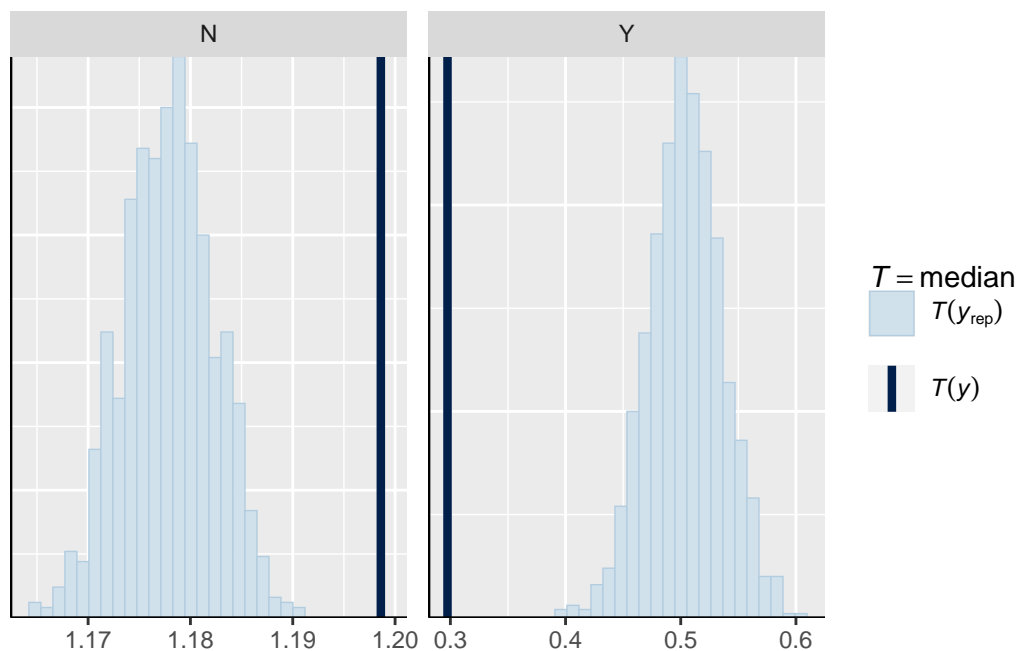
## Test statistics

We can also look at some summary statistics in the PPD versus the data, again either using `bayesplot` – the function of interest is `ppc_stat` or `ppc_stat_grouped` – or just doing it ourselves using `ggplot`.

E.g. medians by prematurity for Model 1

```
ppc_stat_grouped(ds$log_weight, yrep1, group = ds$preterm, stat = 'median')
```





## Question 7

Use a test statistic of the proportion of births under 2.5kg. Calculate the test statistic for the data, and the posterior predictive samples for both models, and plot the comparison (one plot per model).

## LOO

Finally let's calculate the LOO elpd for each model and compare. The first step of this is to get the point-wise log likelihood estimates from each model:

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

And then we can use these in the `loo` function to get estimates for the elpd. Note the `save_psis = TRUE` argument saves the calculation for each simulated draw, which is needed for the LOO-PIT calculation below.

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

Look at the output:

```
loo1
```

Computed from 1000 by 3842 log-likelihood matrix

	Estimate	SE
elpd_loo	1377.2	72.6
p_loo	9.6	1.5
looic	-2754.5	145.2

-----

Monte Carlo SE of elpd\_loo is 0.1.

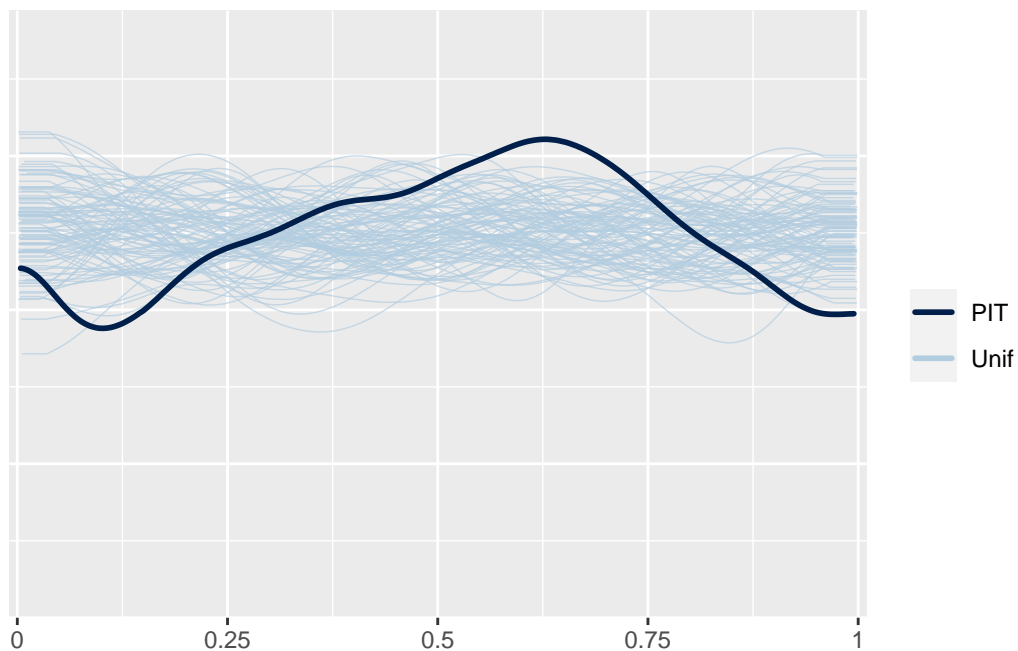
All Pareto k estimates are good ( $k < 0.5$ ).  
See `help('pareto-k-diagnostic')` for details.

## Question 8

Get the LOO estimate of elpd for Model 2 and compare the two models with the `loo_compare` function. Interpret the results.

We can also compare the LOO-PIT of each of the models to standard uniforms. For example for Model 1:

```
ppc_loo_pit_overlay(yrep = yrep1, y = y, lw = weights(loo1$psis_object))
```



### Bonus question (not required)

Create your own PIT histogram “from scratch” for Model 2.

### Question 9

Based on the original dataset, choose one (or more) additional covariates to add to the linear regression model. Run the model in Stan, and compare with Model 2 above on at least 2 posterior predictive checks.