Computing Homework 2

Code

Due: Wednesday 12/6 at 11:59pm, Canvas submission

Honor Pledge

On my honor, I have neither received nor given any unauthorized assistance on this Homework.

- SIGNED: 218007361

Hide

```
# Load packages
library(ggplot2)
library(rstanarm)
library(bayesplot)
library(bayesrules)
library(tidyverse)
library(tidybayes)
library(broom.mixed)
```

We will use `penguins_data` (generated for you below) to build various models of penguin `body_mass_g`. Throughout, we'll utilize weakly informative priors and a basic understanding that the average penguin weighs somewhere between 3,500 and 4,500 grams. One predictor of interest is penguin species: `Adelie` or `Gentoo`.

Hide

```
penguin_data <- penguins_bayes %>%
  filter(species %in% c("Adelie", "Gentoo")) %>%
  select(flipper_length_mm, body_mass_g, species) %>%
  na.omit()
```

1. Modeling Main Effects (3pt)

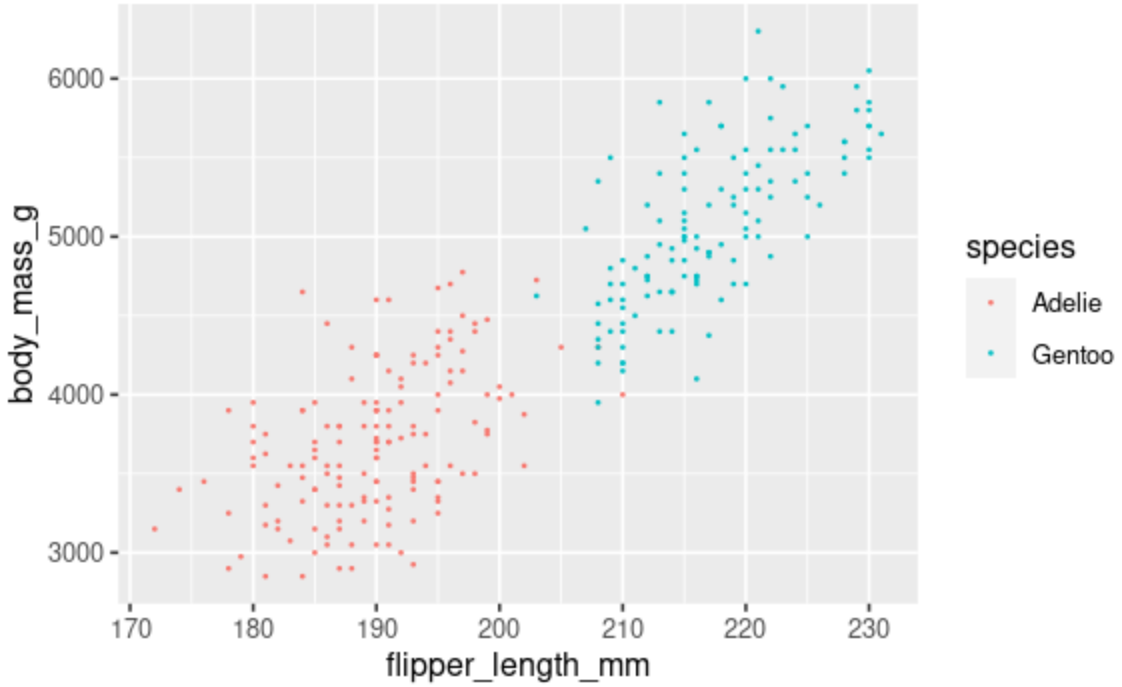
Let's begin our analysis of penguin `body_mass_g` by exploring its relationship with `flipper_length_mm` and `species`.

Q: Plot and summarize the observed relationships among these three variables. (0.5pt)

Hint: use `ggplot()` and set `color = species` to color the two types of penguins differently.

Hide

```
ggplot(penguin_data, aes(x=flipper_length_mm, y=body_mass_g, col=species)) + geom_point(size=0.
2)
```



Q: Use `stan_glm()` to simulate a posterior Normal regression model of `body_mass_g` by `flipper_length_mm` and `species`, without an interaction term. (0.5pt)

Hide

```
penguin_main <- stan_glm(
  body_mass_g ~ flipper_length_mm + species,
  data = penguin_data, family = gaussian,
  prior_intercept = normal(4500, 500),
  prior = normal(0, 5, autoscale = TRUE),
  prior_aux = exponential(1, autoscale = TRUE),
  chains = 4, iter = 5000*2, seed = 84735)
```

```
SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 7.8e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.78 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 1: Iteration: 1000 / 10000 [ 10%]  (Warmup)
Chain 1: Iteration: 2000 / 10000 [ 20%]  (Warmup)
Chain 1: Iteration: 3000 / 10000 [ 30%]  (Warmup)
Chain 1: Iteration: 4000 / 10000 [ 40%]  (Warmup)
Chain 1: Iteration: 5000 / 10000 [ 50%]  (Warmup)
Chain 1: Iteration: 5001 / 10000 [ 50%]  (Sampling)
Chain 1: Iteration: 6000 / 10000 [ 60%]  (Sampling)
Chain 1: Iteration: 7000 / 10000 [ 70%]  (Sampling)
Chain 1: Iteration: 8000 / 10000 [ 80%]  (Sampling)
Chain 1: Iteration: 9000 / 10000 [ 90%]  (Sampling)
Chain 1: Iteration: 10000 / 10000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.213649 seconds (Warm-up)
Chain 1:                0.268808 seconds (Sampling)
Chain 1:                0.482457 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 8e-06 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 2: Iteration: 1000 / 10000 [ 10%]  (Warmup)
Chain 2: Iteration: 2000 / 10000 [ 20%]  (Warmup)
Chain 2: Iteration: 3000 / 10000 [ 30%]  (Warmup)
Chain 2: Iteration: 4000 / 10000 [ 40%]  (Warmup)
Chain 2: Iteration: 5000 / 10000 [ 50%]  (Warmup)
Chain 2: Iteration: 5001 / 10000 [ 50%]  (Sampling)
Chain 2: Iteration: 6000 / 10000 [ 60%]  (Sampling)
Chain 2: Iteration: 7000 / 10000 [ 70%]  (Sampling)
Chain 2: Iteration: 8000 / 10000 [ 80%]  (Sampling)
Chain 2: Iteration: 9000 / 10000 [ 90%]  (Sampling)
Chain 2: Iteration: 10000 / 10000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.239047 seconds (Warm-up)
Chain 2:                0.274972 seconds (Sampling)
Chain 2:                0.514019 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
```

```
Chain 3:
Chain 3: Gradient evaluation took 9e-06 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.09 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 3: Iteration: 1000 / 10000 [ 10%]  (Warmup)
Chain 3: Iteration: 2000 / 10000 [ 20%]  (Warmup)
Chain 3: Iteration: 3000 / 10000 [ 30%]  (Warmup)
Chain 3: Iteration: 4000 / 10000 [ 40%]  (Warmup)
Chain 3: Iteration: 5000 / 10000 [ 50%]  (Warmup)
Chain 3: Iteration: 5001 / 10000 [ 50%]  (Sampling)
Chain 3: Iteration: 6000 / 10000 [ 60%]  (Sampling)
Chain 3: Iteration: 7000 / 10000 [ 70%]  (Sampling)
Chain 3: Iteration: 8000 / 10000 [ 80%]  (Sampling)
Chain 3: Iteration: 9000 / 10000 [ 90%]  (Sampling)
Chain 3: Iteration: 10000 / 10000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.215789 seconds (Warm-up)
Chain 3:                0.261308 seconds (Sampling)
Chain 3:                0.477097 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 9e-06 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.09 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 4: Iteration: 1000 / 10000 [ 10%]  (Warmup)
Chain 4: Iteration: 2000 / 10000 [ 20%]  (Warmup)
Chain 4: Iteration: 3000 / 10000 [ 30%]  (Warmup)
Chain 4: Iteration: 4000 / 10000 [ 40%]  (Warmup)
Chain 4: Iteration: 5000 / 10000 [ 50%]  (Warmup)
Chain 4: Iteration: 5001 / 10000 [ 50%]  (Sampling)
Chain 4: Iteration: 6000 / 10000 [ 60%]  (Sampling)
Chain 4: Iteration: 7000 / 10000 [ 70%]  (Sampling)
Chain 4: Iteration: 8000 / 10000 [ 80%]  (Sampling)
Chain 4: Iteration: 9000 / 10000 [ 90%]  (Sampling)
Chain 4: Iteration: 10000 / 10000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0.209214 seconds (Warm-up)
Chain 4:                0.271446 seconds (Sampling)
Chain 4:                0.48066 seconds (Total)
Chain 4:
```

Q: Create and interpret both visual and numerical diagnostics of your MCMC simulation. (0.5pt)

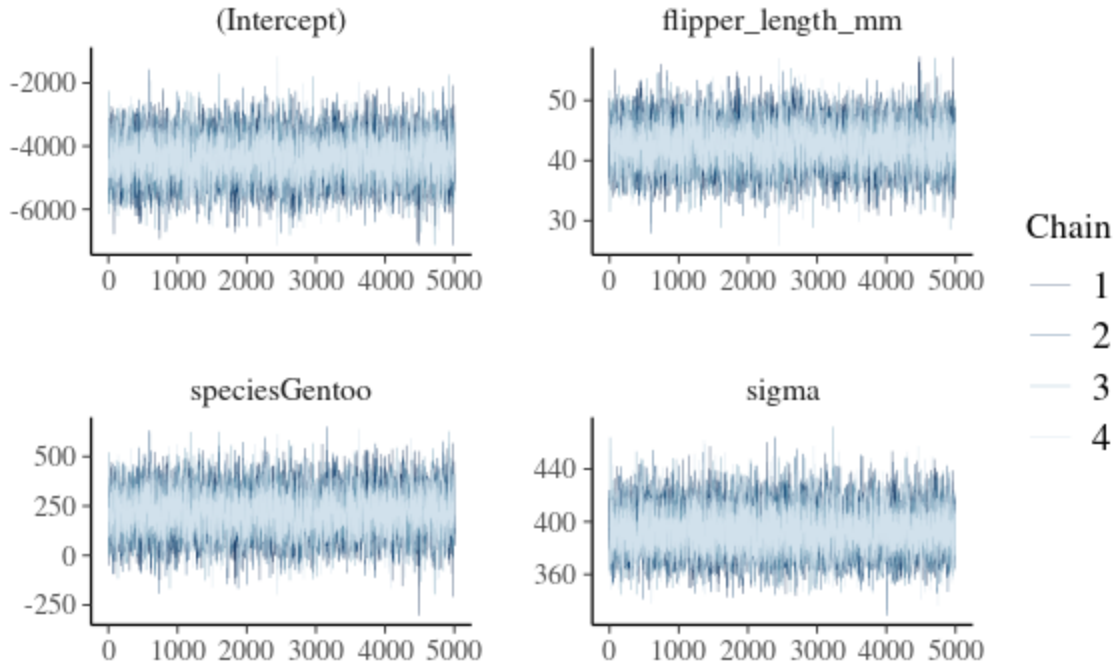
Hide

```
prior_summary(penguin_main)
```

```
Priors for model 'penguin_main'
------
Intercept (after predictors centered)
 ~ normal(location = 4500, scale = 500)

Coefficients
  Specified prior:
    ~ normal(location = [0,0], scale = [5,5])
  Adjusted prior:
    ~ normal(location = [0,0], scale = [ 277.76,8387.97])

Auxiliary (sigma)
  Specified prior:
    ~ exponential(rate = 1)
  Adjusted prior:
    ~ exponential(rate = 0.0012)
------
See help('prior_summary.stanreg') for more details
```

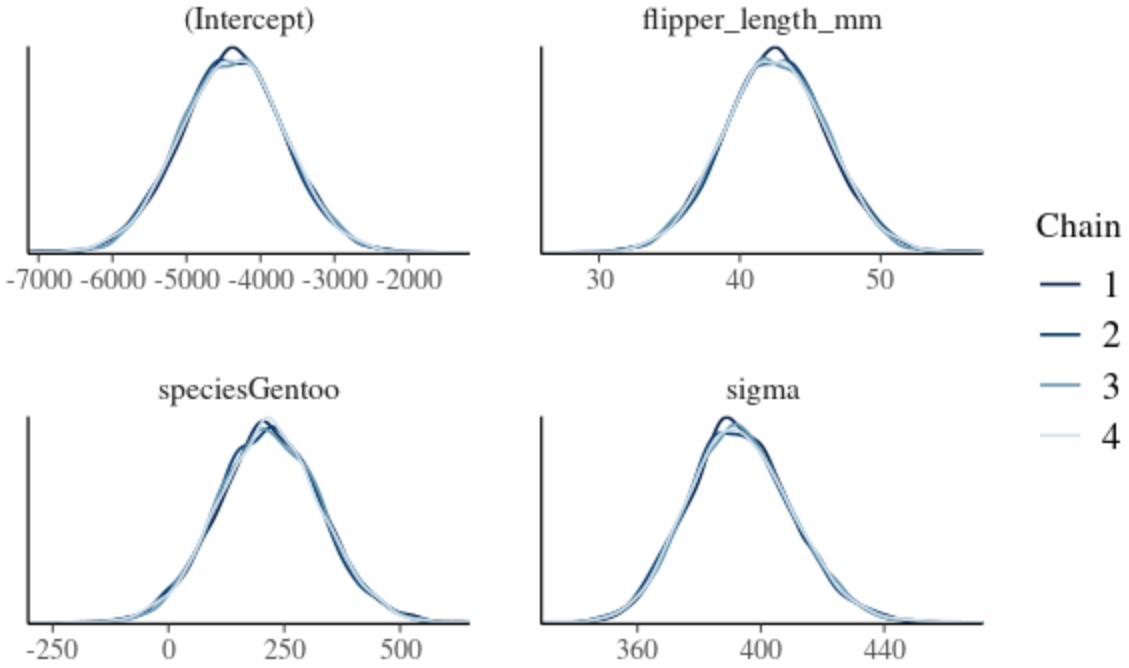
Hide

```
mcmc_trace(penguin_main, size=0.1)
```



Hide

```
mcmc_dens_overlay(penguin_main)
```

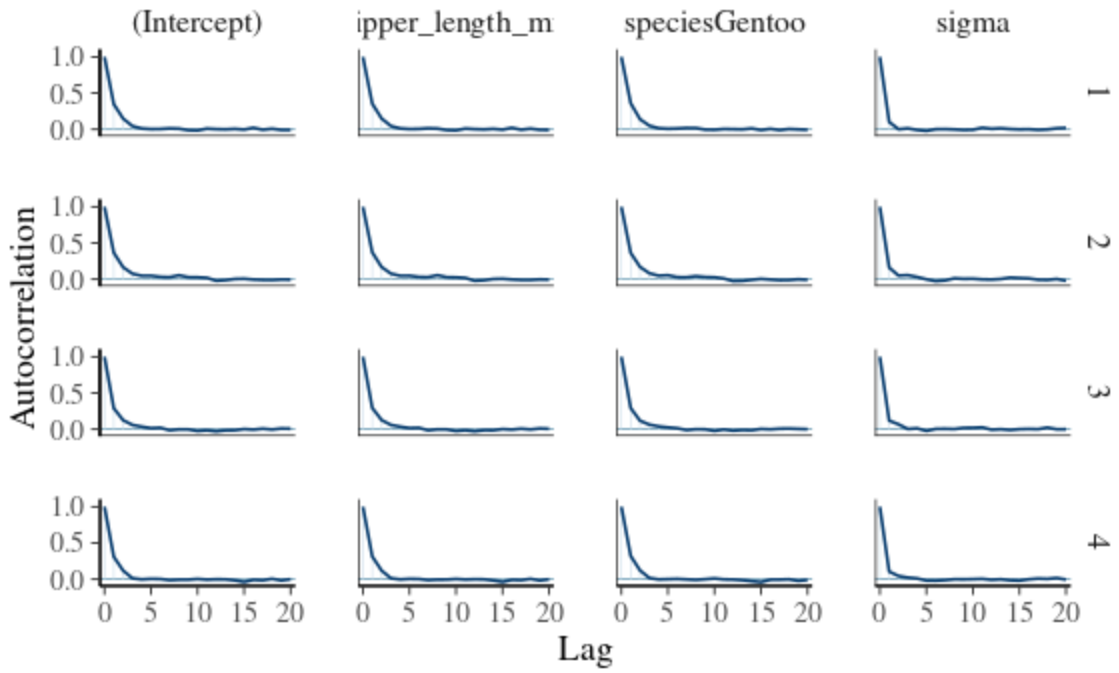


The autoscaled features of our model were found to be: prior = normal(0, 277.76), normal(0, 8387.97) sigma = exp(0.0012)

The trace plot of our variables seem to show our model mixing well. The density overlays suggest the same.

Hide

```
mcmc_acf(penguin_main)
```



Hide

```
neff_ratio(penguin_main)
```

```
      (Intercept) flipper_length_mm     speciesGentoo
          0.45895           0.45455           0.44825
            sigma
          0.72730
```

Hide

```
rhat(penguin_main)
```

```
      (Intercept) flipper_length_mm     speciesGentoo
        1.0000971         1.0000955         1.0002064
            sigma
        0.9999555
```

The autocorrelation drops off quick for each chain, suggesting each mcmc draw mimics independence. neff_ratio > 0.1 and rhat < 1.05, which suggests it mixes fast and has similar variance across each parallel chain.

Q: Produce a `tidy()` summary of this model. Interpret the non-intercept coefficients' posterior median values in context. (0.5pt)

Hide

```
tidy(penguin_main, effects = c("fixed", "aux"))
```

term <chr>	**estimate** <dbl>	**std.error** <dbl>
(Intercept)	-4373.78136	704.375794
flipper_length_mm	42.51283	3.703816
speciesGentoo	216.92383	112.076597
sigma	392.67753	17.038478
mean_PPD	4318.53295	33.816816

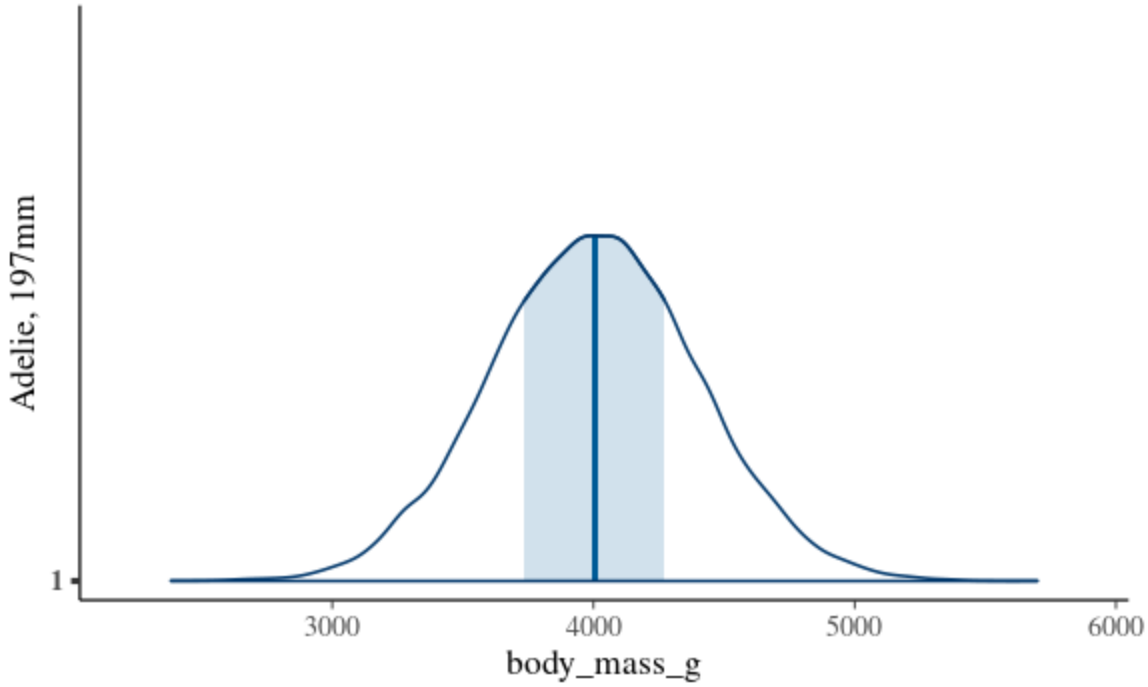
5 rows

For every unit of flipper_length_mm, body_mass_g increases 42.513 units. Between the species types, there is a difference of 216.92 units of body_mass_g. The average standard deviation of this model is 392.68 units. The mean body mass is 4318.53 units with a standard deviation of 33.82.

Q: Simulate, plot, and describe the posterior predictive model for the body mass of an `Adelie` penguin that has a flipper length of 197. (1pt)

Hide

```
Adelie197_prediction <- posterior_predict(penguin_main,
  newdata = data.frame(flipper_length_mm <- c(197),
                       species <- c('Adelie'))
)
mcmc_areas(Adelie197_prediction) + xlab("body_mass_g") + ylab("Adelie, 197mm")
```



For an Adelie penguin with flippers the length of 197mm, the average body mass is around 4000g.

2. Modeling Interaction (3pt)

Building from the previous exercise, our next goal is to model `body_mass_g` by `flipper_length_mm` and `species` with an **interaction** term between these two predictors.

Q: Use `stan_glm()` to simulate the posterior for this model, with four chains at 10,000 iterations each. (1pt)

Hide

```
penguin_interact <- stan_glm(
  body_mass_g ~ flipper_length_mm + species + flipper_length_mm:species,
  data = penguin_data, family = gaussian,
  prior_intercept = normal(4500, 500),
  prior = normal(0, 5, autoscale = TRUE),
  prior_aux = exponential(1, autoscale = TRUE),
  chains = 4, iter = 5000*2, seed = 84735)
```

```
SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 1.7e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.17 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 1: Iteration: 1000 / 10000 [ 10%]  (Warmup)
Chain 1: Iteration: 2000 / 10000 [ 20%]  (Warmup)
Chain 1: Iteration: 3000 / 10000 [ 30%]  (Warmup)
Chain 1: Iteration: 4000 / 10000 [ 40%]  (Warmup)
Chain 1: Iteration: 5000 / 10000 [ 50%]  (Warmup)
Chain 1: Iteration: 5001 / 10000 [ 50%]  (Sampling)
Chain 1: Iteration: 6000 / 10000 [ 60%]  (Sampling)
Chain 1: Iteration: 7000 / 10000 [ 70%]  (Sampling)
Chain 1: Iteration: 8000 / 10000 [ 80%]  (Sampling)
Chain 1: Iteration: 9000 / 10000 [ 90%]  (Sampling)
Chain 1: Iteration: 10000 / 10000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 1.93673 seconds (Warm-up)
Chain 1:                2.24396 seconds (Sampling)
Chain 1:                4.18069 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 1.2e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.12 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 2: Iteration: 1000 / 10000 [ 10%]  (Warmup)
Chain 2: Iteration: 2000 / 10000 [ 20%]  (Warmup)
Chain 2: Iteration: 3000 / 10000 [ 30%]  (Warmup)
Chain 2: Iteration: 4000 / 10000 [ 40%]  (Warmup)
Chain 2: Iteration: 5000 / 10000 [ 50%]  (Warmup)
Chain 2: Iteration: 5001 / 10000 [ 50%]  (Sampling)
Chain 2: Iteration: 6000 / 10000 [ 60%]  (Sampling)
Chain 2: Iteration: 7000 / 10000 [ 70%]  (Sampling)
Chain 2: Iteration: 8000 / 10000 [ 80%]  (Sampling)
Chain 2: Iteration: 9000 / 10000 [ 90%]  (Sampling)
Chain 2: Iteration: 10000 / 10000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 1.99958 seconds (Warm-up)
Chain 2:                2.23935 seconds (Sampling)
Chain 2:                4.23892 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
```

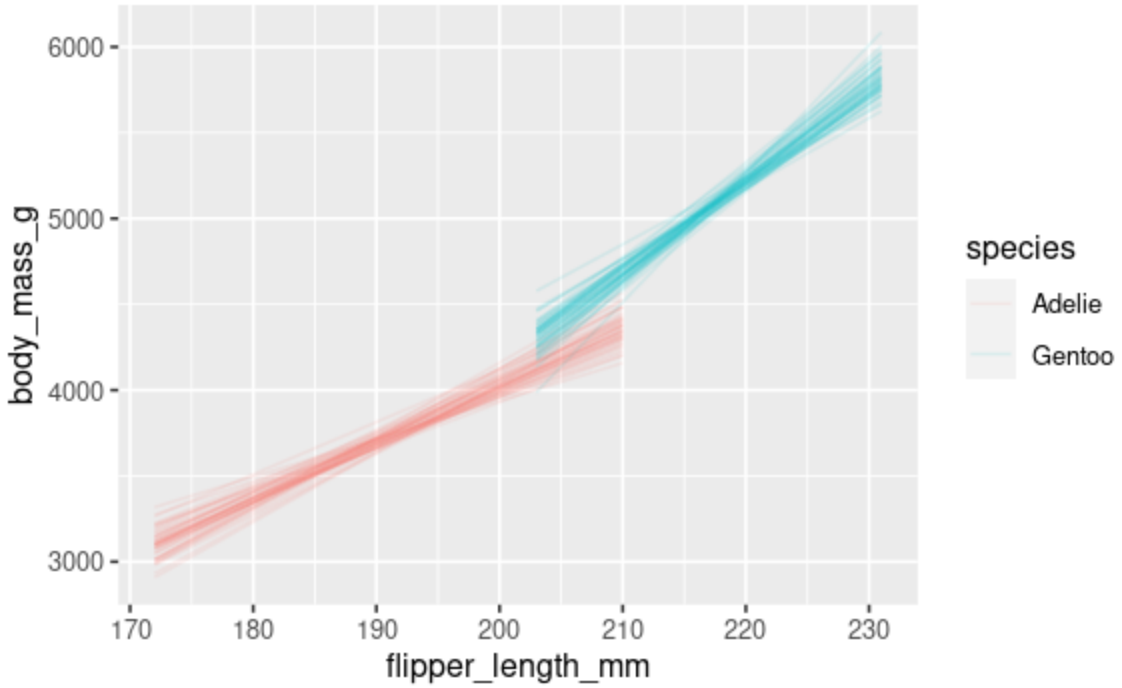
```
Chain 3:
Chain 3: Gradient evaluation took 1.1e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.11 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 3: Iteration: 1000 / 10000 [ 10%]  (Warmup)
Chain 3: Iteration: 2000 / 10000 [ 20%]  (Warmup)
Chain 3: Iteration: 3000 / 10000 [ 30%]  (Warmup)
Chain 3: Iteration: 4000 / 10000 [ 40%]  (Warmup)
Chain 3: Iteration: 5000 / 10000 [ 50%]  (Warmup)
Chain 3: Iteration: 5001 / 10000 [ 50%]  (Sampling)
Chain 3: Iteration: 6000 / 10000 [ 60%]  (Sampling)
Chain 3: Iteration: 7000 / 10000 [ 70%]  (Sampling)
Chain 3: Iteration: 8000 / 10000 [ 80%]  (Sampling)
Chain 3: Iteration: 9000 / 10000 [ 90%]  (Sampling)
Chain 3: Iteration: 10000 / 10000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 1.99098 seconds (Warm-up)
Chain 3:                2.22584 seconds (Sampling)
Chain 3:                4.21682 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 1e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.1 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 4: Iteration: 1000 / 10000 [ 10%]  (Warmup)
Chain 4: Iteration: 2000 / 10000 [ 20%]  (Warmup)
Chain 4: Iteration: 3000 / 10000 [ 30%]  (Warmup)
Chain 4: Iteration: 4000 / 10000 [ 40%]  (Warmup)
Chain 4: Iteration: 5000 / 10000 [ 50%]  (Warmup)
Chain 4: Iteration: 5001 / 10000 [ 50%]  (Sampling)
Chain 4: Iteration: 6000 / 10000 [ 60%]  (Sampling)
Chain 4: Iteration: 7000 / 10000 [ 70%]  (Sampling)
Chain 4: Iteration: 8000 / 10000 [ 80%]  (Sampling)
Chain 4: Iteration: 9000 / 10000 [ 90%]  (Sampling)
Chain 4: Iteration: 10000 / 10000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 2.00203 seconds (Warm-up)
Chain 4:                2.1747 seconds (Sampling)
Chain 4:                4.17674 seconds (Total)
Chain 4:
```

Q: Simulate and plot 50 posterior model lines. Briefly describe what you learn from this plot. (1pt)

Hide

```
penguin_data %>%
  add_fitted_draws(penguin_interact, n = 50) %>%
  ggplot(aes(x=flipper_length_mm, y=body_mass_g, color=species)) +
    geom_line(aes(y=.value, group=paste(species, .draw)), alpha=0.1)
```

```
Warning: `fitted_draws` and `add_fitted_draws` are deprecated as their names were confusing.
Use [add_]epred_draws() to get the expectation of the posterior predictive.
Use [add_]linpred_draws() to get the distribution of the linear predictor.
For example, you used [add_]fitted_draws(..., scale = "response"), which
means you most likely want [add_]epred_draws(...).
```



Q: Produce a `tidy()` summary for this model. Based on the summary, do you have evidence that the interaction terms are necessary for this model? Explain your reasoning. (1pt)

Hide

```
tidy(penguin_interact, effects = c("fixed", "aux"),
     conf.int = TRUE, conf.level = 0.8) %>%
  select(-std.error)
```

term <chr>	estimate <dbl>	conf.low <dbl>	conf.high <dbl>
(Intercept)	-2653.38852	-3827.27100	-1484.34869
flipper_length_mm	33.45768	27.28455	39.61458
speciesGentoo	-3971.23924	-5837.59765	-2094.30180
flipper_length_mm:speciesGentoo	20.44008	11.32477	29.49651

term <chr>	estimate <dbl>	conf.low <dbl>	conf.high <dbl>
sigma	387.53200	367.52505	409.64158
mean_PPD	4318.03061	4275.20807	4360.70981

6 rows

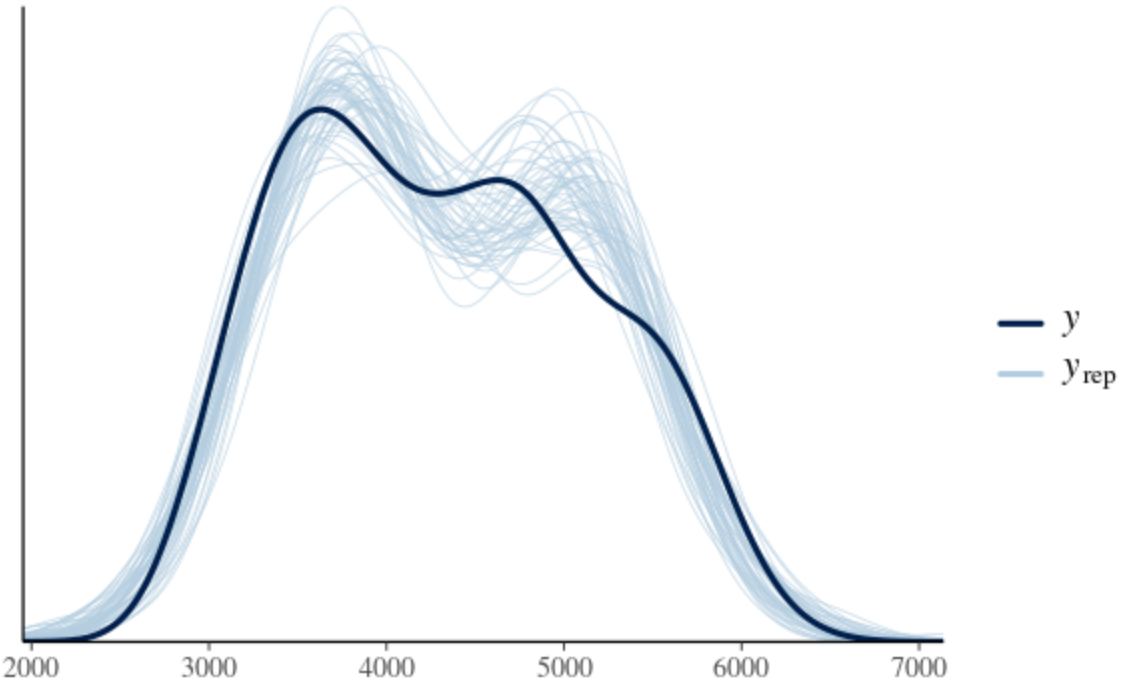
We do not have evidence that the interaction terms are necessary for this model. The 80% confidence levels between flipper_length_mm and flipper_length_mm:species overlap, meaning the difference in their means is not statistically significant, and we have no basis to utilize an interaction term.

3. Model Comparison (4pt)

Q: Produce and compare the `pp_check()` plots for both models above (`penguin_main` and `penguin_interact`). (1pt)

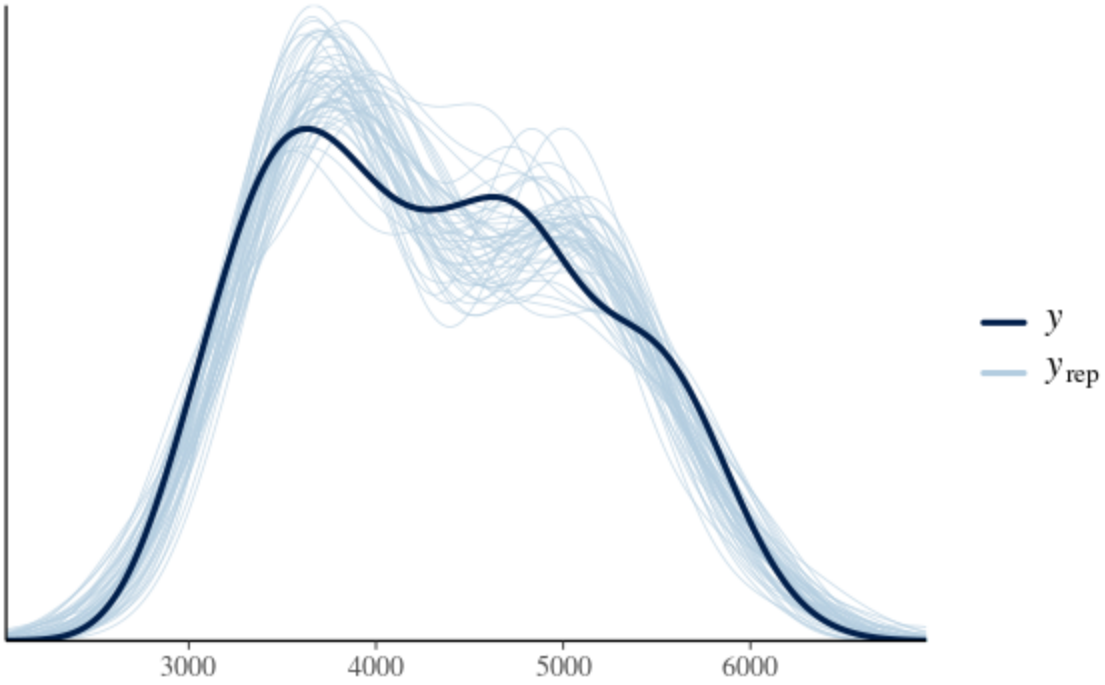
Hide

```
pp_check(penguin_main)
```



Hide

```
pp_check(penguin_interact)
```



Q: Use 10-fold cross-validation to assess and compare the posterior predictive quality of the two models using `prediction_summary_cv()`. (1pt)

Note: this step might take some time! In addition, you will want to `set.seed()` at some seed value before running `prediction_summary_cv()` in order to reproduce your results.

Hide

```
set.seed(84735)
prediction_summary_cv(model = penguin_main,
                      data = penguin_data, k = 10)
```

```
$folds
```

fold <int>	mae <dbl>	mae_scaled <dbl>	within_50 <dbl>	within_95 <dbl>
1	291.7689	0.7412312	0.4642857	0.9285714
2	359.1083	0.9084590	0.3703704	0.9629630
3	278.0164	0.7158447	0.4444444	0.9259259
4	236.3362	0.6032538	0.5357143	0.9285714
5	328.6551	0.8213442	0.4074074	0.9629630
6	300.3992	0.7639139	0.4444444	0.9629630
7	229.7054	0.5725455	0.5357143	0.9642857
8	323.0117	0.8249749	0.4074074	0.8888889
9	184.6524	0.4513300	0.6666667	1.0000000

fold <int>	mae <dbl>	mae_scaled <dbl>	within_50 <dbl>	within_95 <dbl>
10	271.5102	0.6773446	0.5000000	1.0000000

1-10 of 10 rows

$cv

mae <dbl>	mae_scaled <dbl>	within_50 <dbl>	within_95 <dbl>
280.3164	0.7080242	0.4776455	0.9525132

1 row

Hide

```
set.seed(84735)
prediction_summary_cv(model = penguin_interact,
                      data = penguin_data, k = 10)
```

$folds

fold <int>	mae <dbl>	mae_scaled <dbl>	within_50 <dbl>	within_95 <dbl>
1	235.1880	0.6013974	0.5357143	0.9285714
2	313.7863	0.8092109	0.4074074	0.9629630
3	269.6230	0.7080452	0.4074074	0.8888889
4	307.5274	0.8051625	0.4642857	0.8928571
5	411.7773	1.0508281	0.4074074	0.9629630
6	263.5868	0.6751313	0.5185185	0.9629630
7	204.4514	0.5156559	0.5714286	0.9642857
8	261.3623	0.6833286	0.4444444	0.8888889
9	192.9886	0.4837631	0.7037037	1.0000000
10	243.9403	0.6133414	0.6071429	1.0000000

1-10 of 10 rows

$cv

mae <dbl>	mae_scaled <dbl>	within_50 <dbl>	within_95 <dbl>
270.4231	0.6945864	0.506746	0.9452381

1 row

```
NA
```

Q: Evaluate and compare the ELPD posterior predictive accuracy of the two models. (1pt)

Hide

```
set.seed(84735)
loo_main <- loo(penguin_main)
loo_interact <- loo(penguin_interact)
loo_compare(loo_main, loo_interact)
```

```
                elpd_diff se_diff
penguin_interact  0.0       0.0
penguin_main     -3.7       2.4
```

Q: In summary, which one of the two models is ``better?'' Explain. (1pt)

Of the two models it seems that penguin_interact performs slightly better than penguin_main; However, their metrics are close, and not statistically significant enough. While penguin_interact performs better in mae and within_50, it is less accurate within_95. Also, when evaluating predictive accuracy using expected log-predictive densities, penguin_interact outperforms penguin_main, but only by one se_diff, not two; Meaning, there may be a difference, but it is not a significantly greater difference.