# BR11

Steph Jordan

```
library(bayesrules)
library(rstanarm)
```

```
## Loading required package: Rcpp
```

```
## This is rstanarm version 2.21.1
```

```
## - See https://mc-stan.org/rstanarm/articles/priors for changes to default priors!
```

```
## - Default priors may change, so it's safest to specify priors, even if equivalent to the defaults.
```

```
## - For execution on a local, multicore CPU with excess RAM we recommend calling
```

```
##   options(mc.cores = parallel::detectCores())
```

```
library(bayesplot)
```

```
## This is bayesplot version 1.8.1
```

```
## - 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(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(broom.mixed)
library(tidybayes)
```

## Exercise 11.1

If we find a relationship between two independent variables and the dependent variable (and those two independent variables are independent of eachother), then we might want to build a model that incorporates both of their influences' on the outcome variable.

## Exercise 11.2

a. Ford is the reference category

b. The difference in a Ford car's miles per gallon and a Subaru car's miles per gallon.

c. The typical miles per gallon of a Ford car.

## Exercise 11.3

a. B0 represents the typical size of a Mr Stripey tomato at 0 days of growth; B1 represents the amount a tomato increases in weight for 1 more day of growth; B2 represents the typical difference in weights between a Mr Stripey tomato and a Roma tomato at any day of age.

b. If B2 were 0 there would be no difference in weight between Roma and Mr Stripey tomatoes.

## Exercise 11.4

a. The relationship between tomato size and age varies depending on the type of tomato.

b. B3 represents the differences in the relationship between age and weight for the 2 different types of tomatoes (the different slopes that each of these relationships possess).

## Exercise 11.5

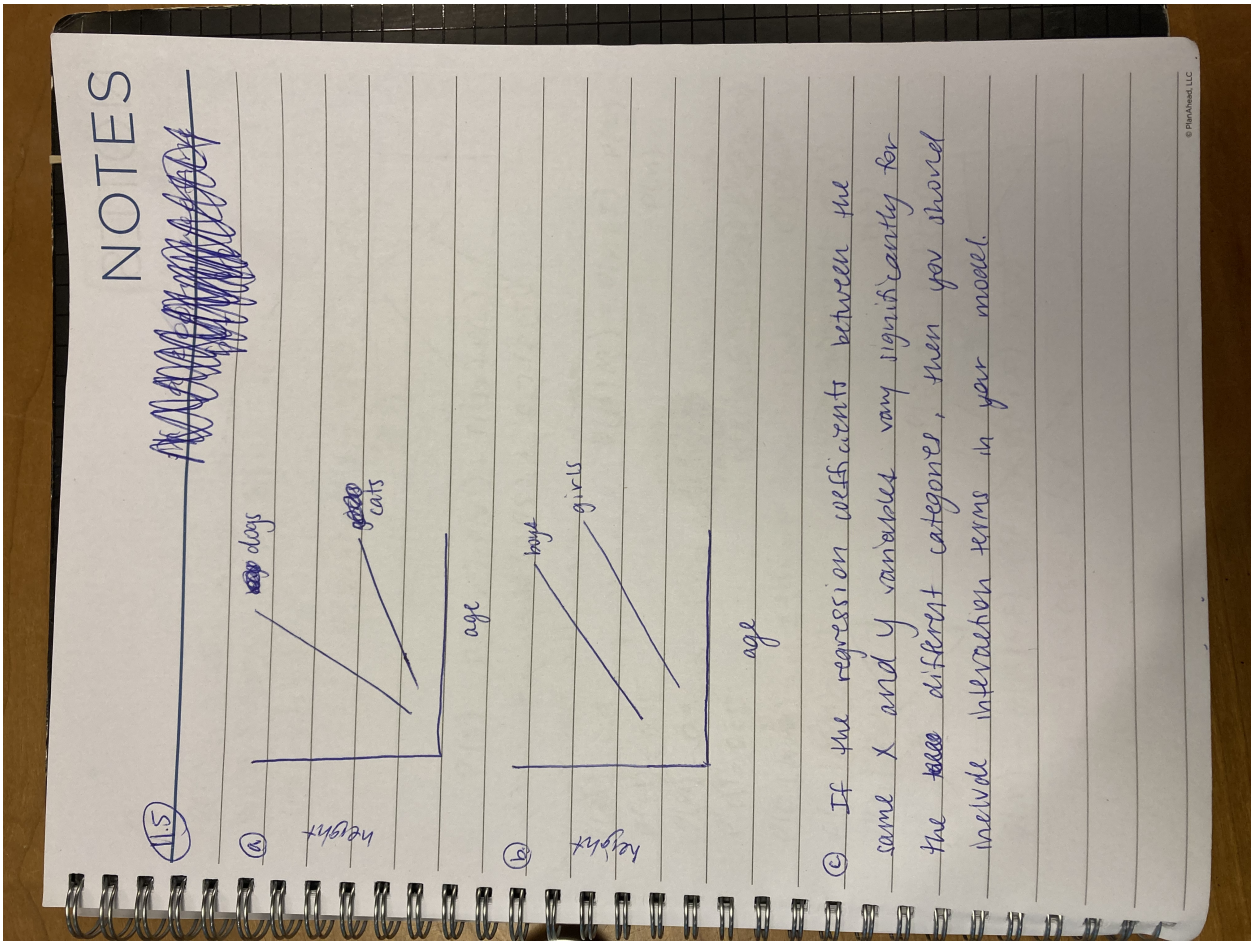



Figure 1: 11.5

## Exercise 11.6

a. By adding more predictors, we can improve the predictive accuracy of our posterior model.

b. By removing predictors, we can better isolate the relationship between two variables.

c. Height. Height has been found to correlate with foot size.

d. Whether the child knows how to swim. I think the relationship between swimming ability and shoe size would be spurious, and would uncessarily complicate the model.

## Exercise 11.7

a. A good model produces a posterior distribution that closely matches the observed distribution; has a low MAE scaled; has a high percentage of values that fall within the within_50 interval.

b. A bad model has a posterior distribution that deviates strongly from the shape of the observed distribution (perhaps the wrong type of model was chosen); has a high MAE scaled; has a low percentage of values that fall within the within_50 and within_95 interval.

## Exercise 11.8

1) visualization: use pp_check() to compare shape of observed and predicted distributions; use pp_intervals() to visually assess how the observed values compare to the posterior predicted intervals
2) cross-validation: break the data set into k>=10 different folds; see how the model performs on each of these different subsets of the main dataset through observing the MAE, MAE scaled values, within_50, and within_95 for each fold (the methodology of the folds is that the first iteration trains on the first 9 subsets and tests on the 10th subset; the second iteration trains on the 2-10 and tests on the 1st; the third iteration trains on 1 and 3-10 and tests on the 2nd, etc. . . ).
3) ELPD: the larger the expected logged posterior predictive pdf across a new set of data points, the more accurate the posterior predictions of y. Calculate the ELPD for each model, and then compare them by looking at the absolute differences and the standard error differences.

## Exercise 11.9

We want to include enough predictors that our model is accurate (i.e. closely fit to our data), but not so many predictors that our model is "overfit" (i.e. biased). Therefore, when considering how many predictor variables to include, we entertain the bias-variance tradeoff, considering the fact that we don't want our model to be biased (overfit) nor do we want it to have too much variance (want it to be an accurate predictor).

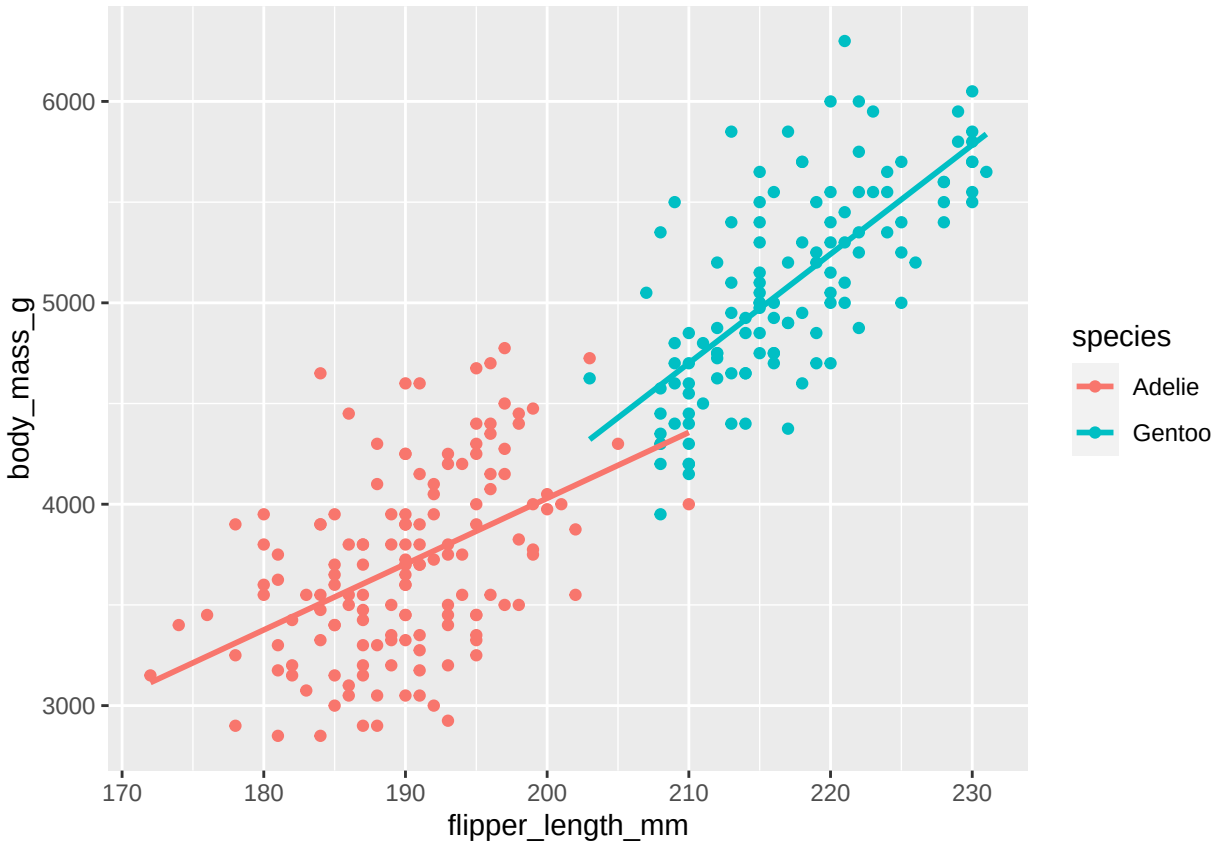
## Exercise 11.10

a. Plotting penguin data

```
data("penguins_bayes")
# Alternative penguin data
penguin_data <- penguins_bayes |>
  filter(species %in% c("Adelie", "Gentoo")) |> drop_na()
```

```
ggplot(penguin_data, aes(x = flipper_length_mm, y = body_mass_g, color=species)) +
 geom_point() +
  geom_smooth(method = "lm", se = FALSE)
```

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

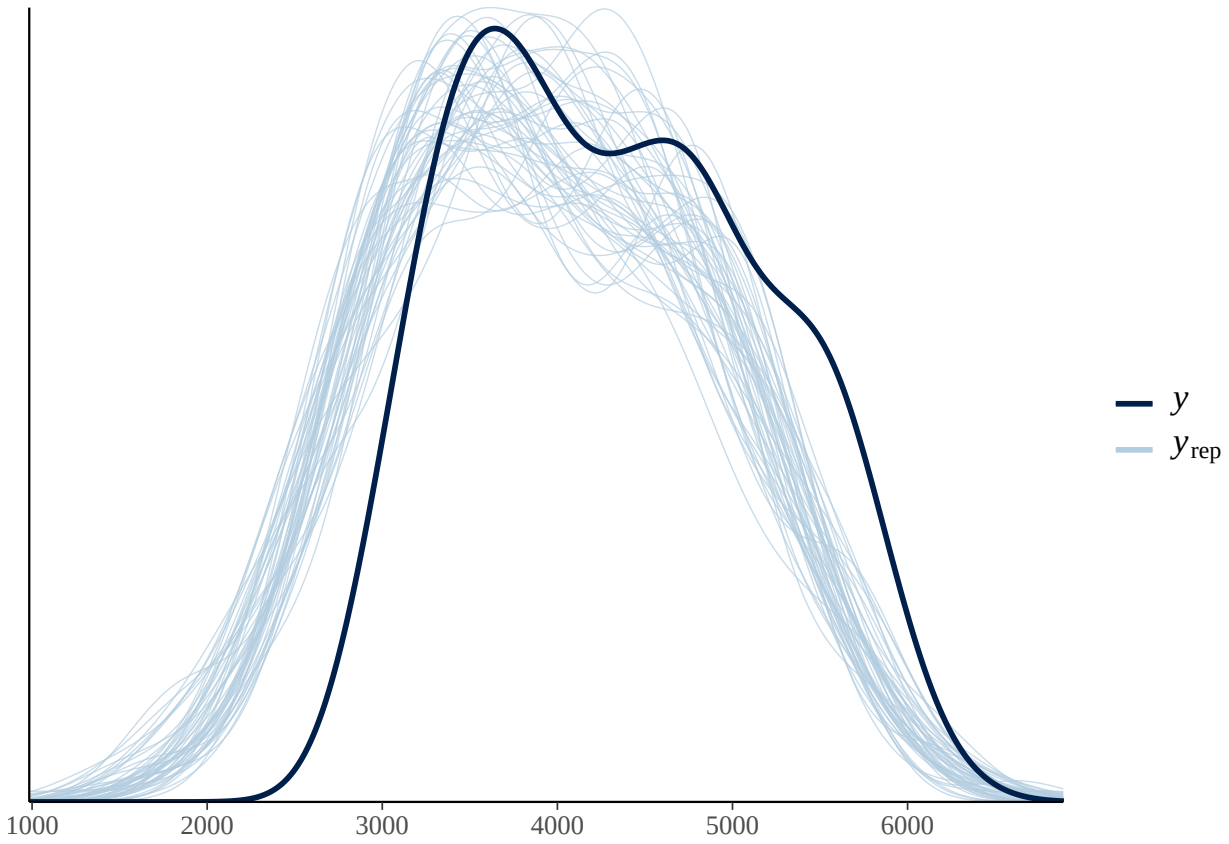



b. Building the model

```
penguin_model_1 <- stan_glm(
  body_mass_g ~ flipper_length_mm + species,
  data = penguin_data, refresh=0, family = gaussian,
  prior_intercept = normal(3100, 50),
  prior = normal(33, 2.5, autoscale = TRUE),
  prior_aux = exponential(0.001, autoscale = TRUE),
  chains = 4, iter = 4000*2, seed = 84735)
```

c. Checking out some visual and numerical diagnostics of the model

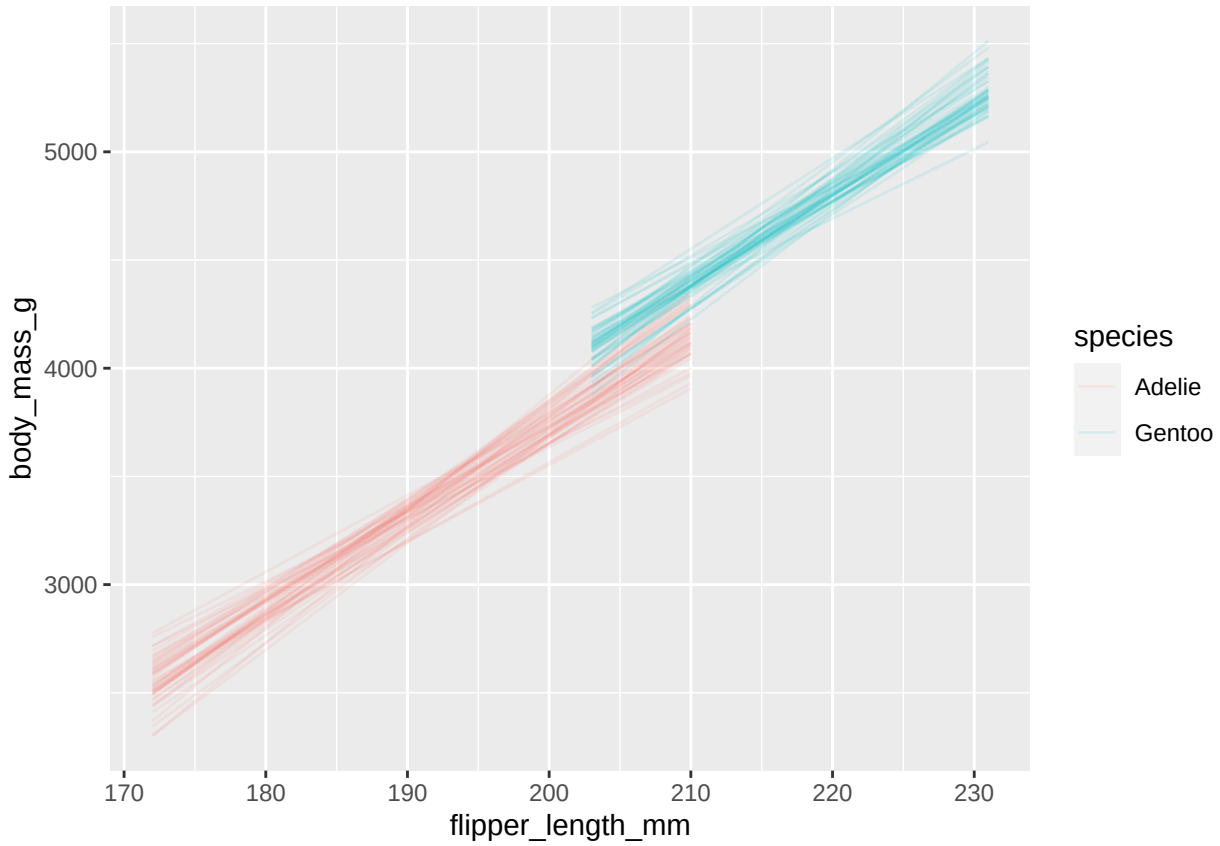
```
pp_check(penguin_model_1)
```



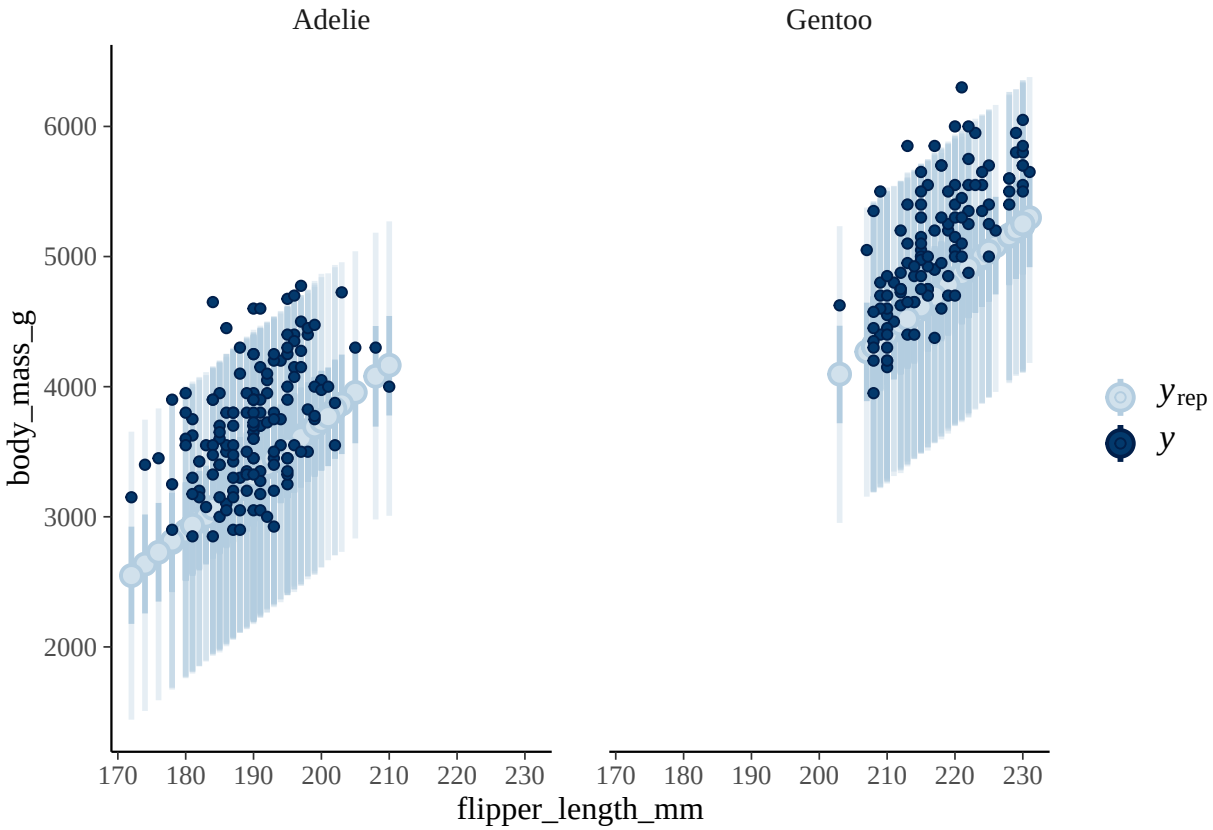

Check out some draws from our model

```
penguin_data %>%
  add_fitted_draws(penguin_model_1, 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(...).
```




```
set.seed(84735)
predictions_1 <- posterior_predict(penguin_model_1, newdata = penguin_data)


ppc_intervals_grouped(penguin_data$body_mass_g, yrep = predictions_1,
                      x = penguin_data$flipper_length_mm, group = penguin_data$species,
                      prob = 0.5, prob_outer = 0.95,
                      facet_args = list(scales = "fixed")) +
  labs(x = "flipper_length_mm", y = "body_mass_g")
```




From both of these visual diagnostics, we can see that our model follows the shape of the distribution pretty well, and that the majority of our predicted values fall within at least the 95% posterior prediction interval.

Some numerical summaries

```
prediction_summary_cv(model = penguin_model_1, data = penguin_data, k = 10)
```

```
## $folds
##    fold      mae mae_scaled within_50 within_95
## 1     1 427.1594  0.6729862 0.4814815 1.0000000
## 2     2 537.3611  0.8637755 0.4230769 0.9230769
## 3     3 536.7048  0.8241943 0.4074074 0.8888889
## 4     4 619.8548  0.9002595 0.1923077 0.9615385
## 5     5 356.5858  0.5338050 0.5555556 0.9259259
## 6     6 405.2069  0.6354953 0.5000000 1.0000000
## 7     7 405.6854  0.6270410 0.5384615 0.9615385
## 8     8 504.9042  0.7843185 0.4074074 1.0000000
## 9     9 357.7319  0.6067848 0.5769231 0.9230769
## 10   10 579.3295  0.8830854 0.3333333 1.0000000
##
## $cv
##        mae mae_scaled within_50 within_95
## 1 473.0524  0.7331745 0.4415954 0.9584046
```

The numerical statistics confirm what the visualizations demonstrated–most (95%) of our data are within the 95% prediction interval, and close to half are within the 50% prediction interval.

d. Creating a tidy() summary of the model

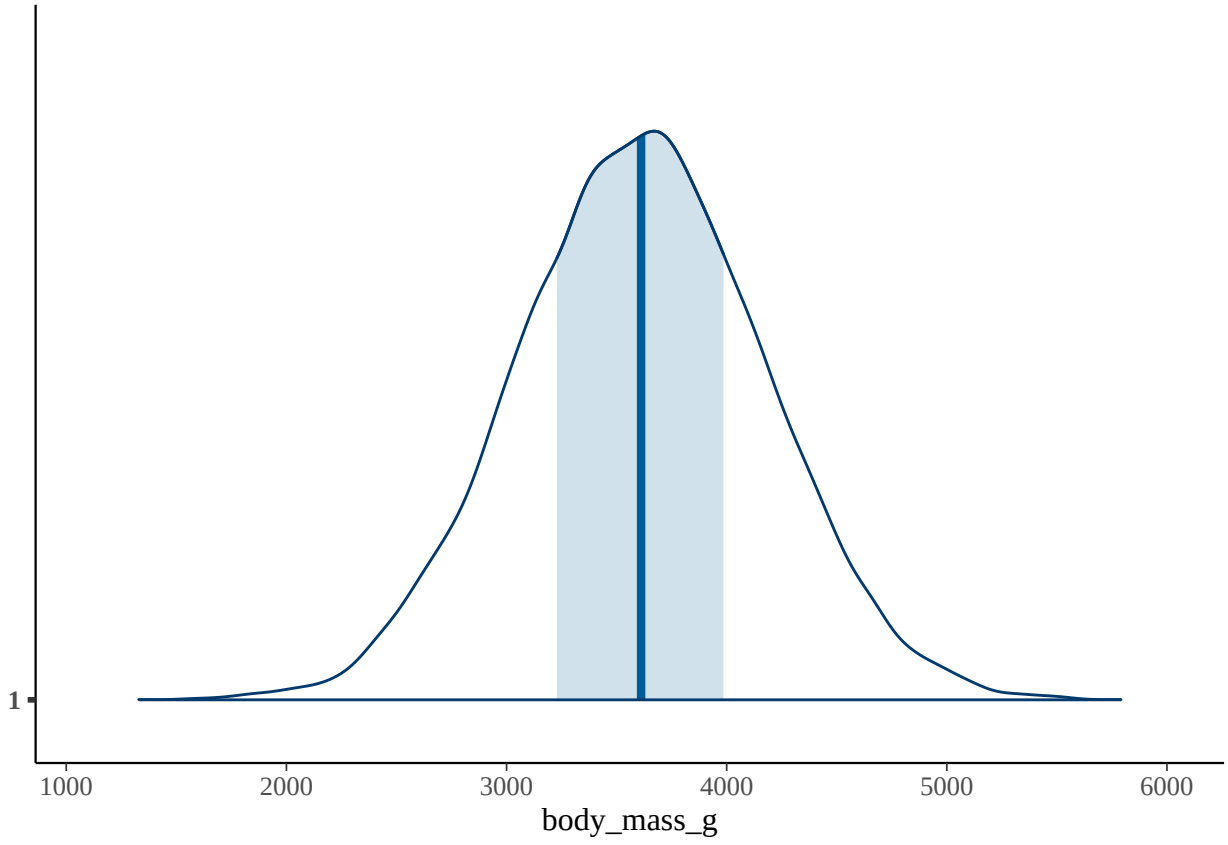
```
tidy(penguin_model_1, effects = c("fixed", "aux"),
     conf.int = TRUE, conf.level = 0.80)
```

```
## # A tibble: 5 x 5
##   term              estimate std.error conf.low conf.high
##   <chr>                <dbl>     <dbl>    <dbl>     <dbl>
## 1 (Intercept)        -4744.     966.    -6021.    -3475.
## 2 flipper_length_mm     42.4      5.10     35.8      49.1
## 3 speciesGentoo        234.     154.       33.8     436.
## 4 sigma                553.      44.6     503.      616.
## 5 mean_PPD            3940.      62.0    3856.     4014.
```

The flipper_length_mm coefficient refers to the amount of increase in body_mass caused by a one unit change in flipper_length for Adelie penguins. The speciesGentoo coefficient refers to the difference in body_mass_g for a given flipper_length between the Adelie and Gentoo species.

e. Simulating and plotting posterior model for an Adelie penguin with flipper length 197.

```
# Simulate a set of predictions
set.seed(84735)
prediction <- posterior_predict(
  penguin_model_1,
  newdata = data.frame(flipper_length_mm= 197,
                       species = "Adelie"))

# Plot the posterior predictive models
mcmc_areas(prediction) +
  xlab("body_mass_g")
```



The body_mass of an Adelie penguin with flipper length 197 is around 3700, with likely values ranging from 3250 to 4000.

## Exercise 11.11

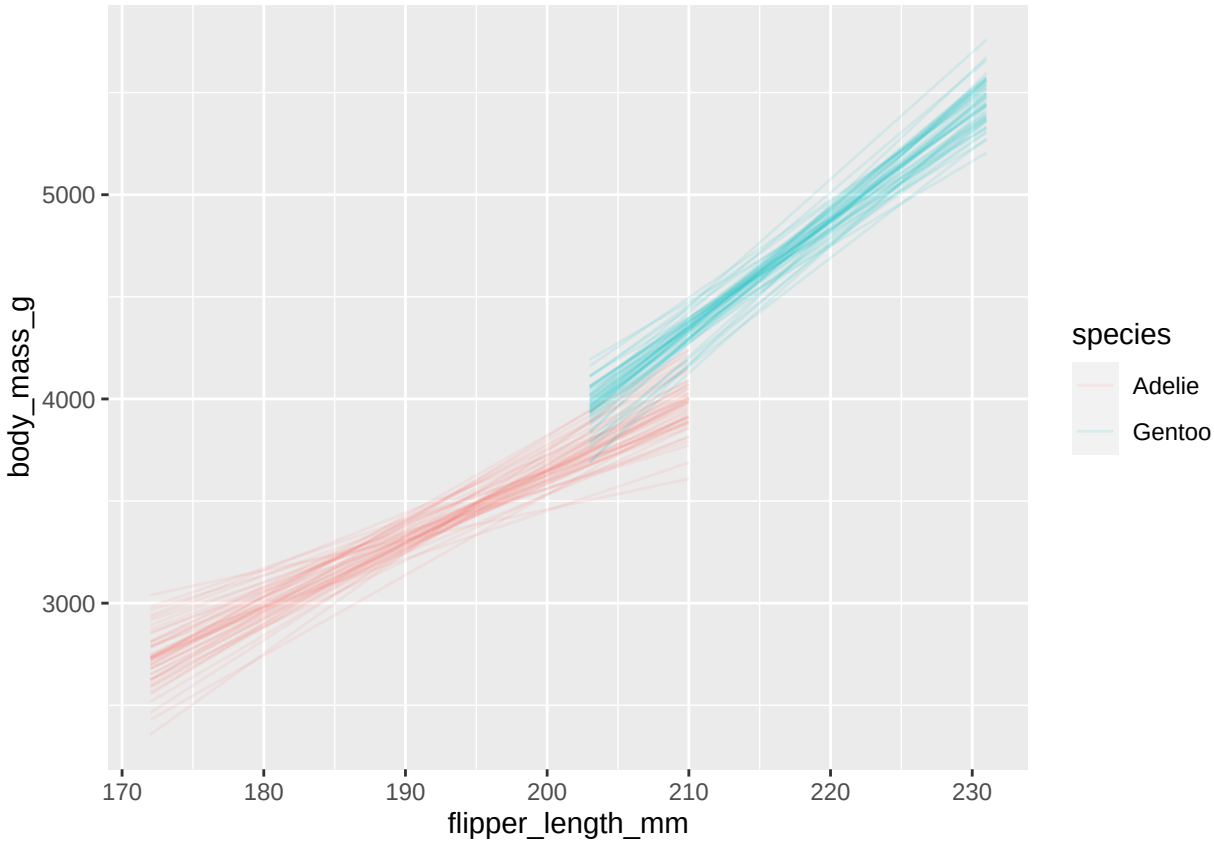
a. Modeling body mass by flipper length and species including an interaction term.

```
penguin_model_2 <- stan_glm(
  body_mass_g ~ flipper_length_mm + species + flipper_length_mm:species,
  data = penguin_data, family = gaussian, refresh=0,
  prior_intercept = normal(3100, 50),
  prior = normal(33, 2.5, autoscale = TRUE),
  prior_aux = exponential(0.001, autoscale = TRUE),
  chains = 4, iter = 4000*2, seed = 84735, verbose=FALSE)
```

b. Simulating and plotting 50 posterior lines

```
penguin_data %>%
  add_fitted_draws(penguin_model_2, 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(...).
```




The slope of the line is slightly less steep than the model without the interaction term for both Adelie and

Gentoo penguins. This implies that the interaction "dampens" the strength of the relationship between flipper_length and body_mass for both species. We can estimate the overall slope (including the relationship between flipper_length_mm and body_mass_g and the relationship between the interaction between flipper_length_mm/species and body_mass_g) as around 33 for Adelie penguins, and 50 for Gentoo penguins.

c. Using tidy() summary

```
tidy(penguin_model_2, effects = c("fixed", "aux"),
     conf.int = TRUE, conf.level = 0.80)
```

```
## # A tibble: 6 x 5
##   term                              estimate std.error conf.low conf.high
##   <chr>                                <dbl>     <dbl>    <dbl>     <dbl>
## 1 (Intercept)                        -2967.    1192.   -4535.     -1435.
## 2 flipper_length_mm                     33.1      6.29    25.1       41.3
## 3 speciesGentoo                      -3900.    1703.   -6086.     -1676.
## 4 flipper_length_mm:speciesGentoo       20.2      8.24     9.34      30.8
## 5 sigma                                542.      43.0    492.       601.
## 6 mean_PPD                            3952.      59.8   3871.      4024.
```

Bring in some loo comparisons to evaluate models relative to eachother:

```
set.seed(84735)
loo_1 <- loo(penguin_model_1)
loo_2 <- loo(penguin_model_2)

loo_compare(loo_1, loo_2)
```

```
##                 elpd_diff se_diff
## penguin_model_2  0.0       0.0
## penguin_model_1 -5.7       1.9
```

Theoretically, it would make sense that this interaction term would be essential–it seems likely that the relationship between flipper_length_mm and body_mass would vary based on species. Empirically, this is confirmed. The loo for the second model is greater than the loo for the first model; therefore, the second model (with the interaction term) performs better than the first–however, this also possibly due to overfitting.

## Exercise 11.12

a. Simulating model with 3 predictors

```
penguin_model_3 <- stan_glm(
  body_mass_g ~ flipper_length_mm + bill_length_mm + bill_depth_mm,
  data = penguin_data, family = gaussian, refresh=0,
  prior_intercept = normal(3100, 50),
  prior = normal(33, 2.5, autoscale = TRUE),
  prior_aux = exponential(0.001, autoscale = TRUE),
  chains = 4, iter = 4000*2, seed = 84735, verbose=FALSE)
```

b. Using posterior_interval() to produce 95% credible intervals for the model parameters

```
posterior_interval(penguin_model_3, prob=0.8)
```

```
##                            10%         90%
## (Intercept)       -7378.32923 -5480.31256
## flipper_length_mm    27.45303    37.79006
## bill_length_mm       56.63118    83.84360
## bill_depth_mm        29.00648    74.40261
```

```
## sigma                406.01564   490.09006
```

c. All variables have a positive association with body mass.

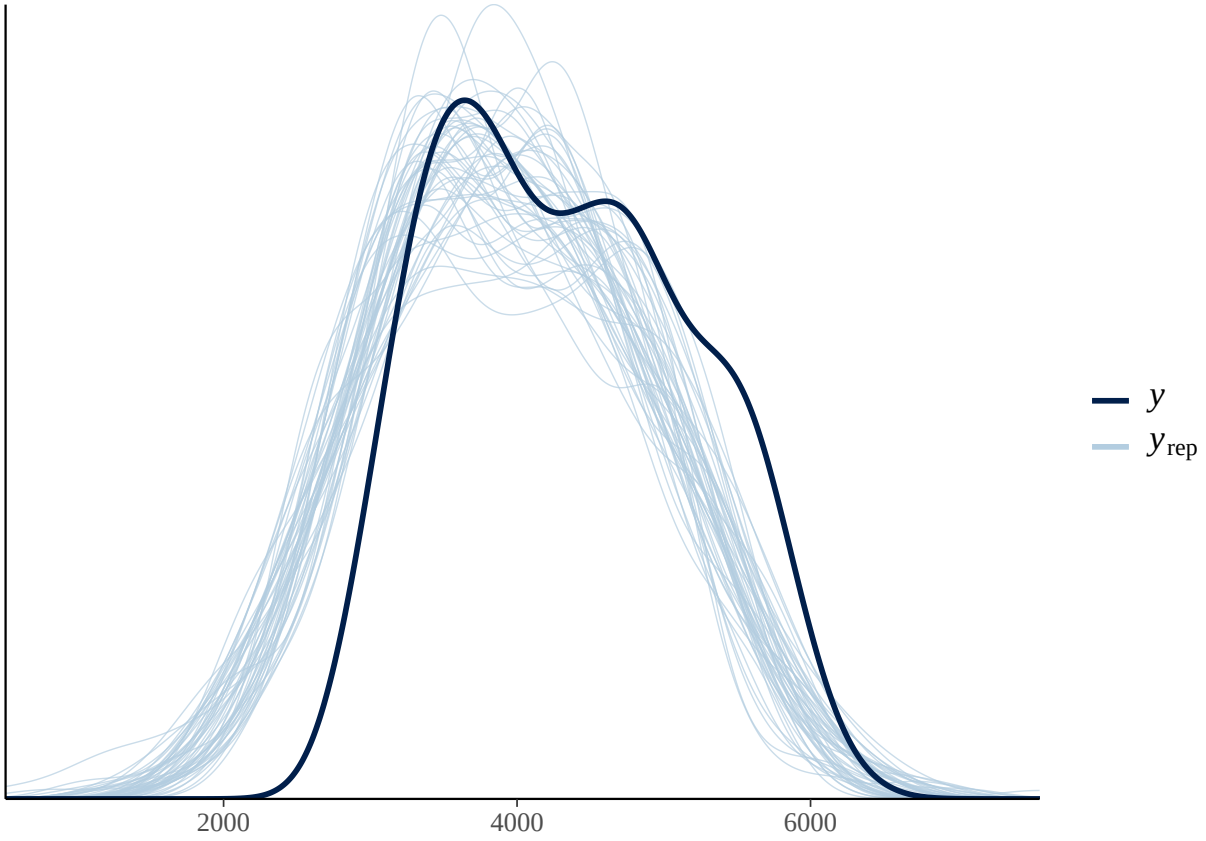
## Exercise 11.13

a. Simulating the 4 models:

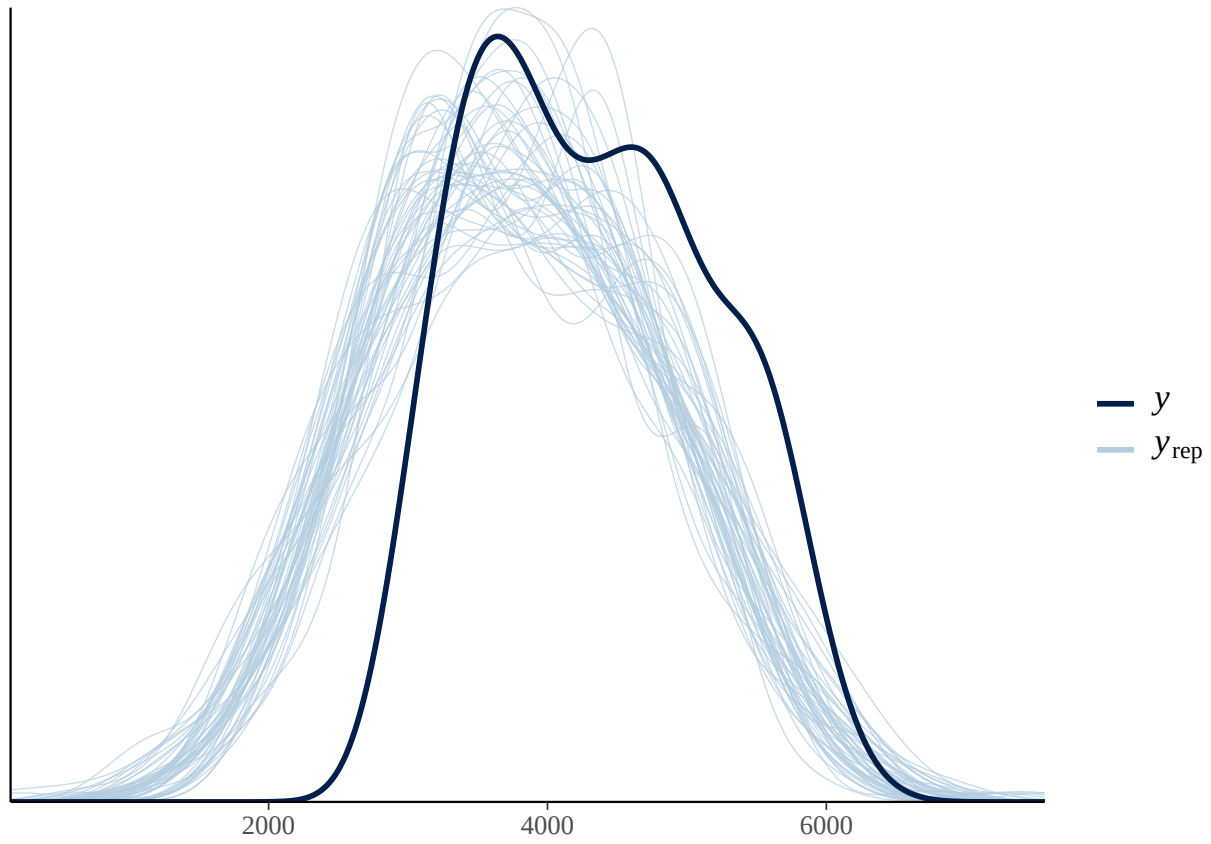
```
penguin_model_4 <- stan_glm(
  body_mass_g ~ flipper_length_mm,
  data = penguin_data, family = gaussian, refresh=0,
  prior_intercept = normal(3100, 50),
  prior = normal(30, 2.5, autoscale = TRUE),
  prior_aux = exponential(0.001, autoscale = TRUE),
  chains = 4, iter = 4000*2, seed = 84735, verbose=FALSE)

penguin_model_5 <- stan_glm(
  body_mass_g ~ species,
  data = penguin_data, family = gaussian, refresh=0,
  prior_intercept = normal(3100, 50),
  prior = normal(30, 2.5, autoscale = TRUE),
  prior_aux = exponential(0.001, autoscale = TRUE),
  chains = 4, iter = 4000*2, seed = 84735, verbose=FALSE)

penguin_model_6 <- stan_glm(
  body_mass_g ~ flipper_length_mm +species,
  data = penguin_data, family = gaussian, refresh=0,
  prior_intercept = normal(3100, 50),
  prior = normal(30, 2.5, autoscale = TRUE),
  prior_aux = exponential(0.001, autoscale = TRUE),
  chains = 4, iter = 4000*2, seed = 84735, verbose=FALSE)

penguin_model_7 <- stan_glm(
  body_mass_g ~ flipper_length_mm +bill_length_mm + bill_depth_mm,
  data = penguin_data, family = gaussian, refresh=0,
  prior_intercept = normal(3100, 50),
  prior = normal(30, 2.5, autoscale = TRUE),
  prior_aux = exponential(0.001, autoscale = TRUE),
  chains = 4, iter = 4000*2, seed = 84735, verbose=FALSE)
```

b. Checking the models using pp_check()

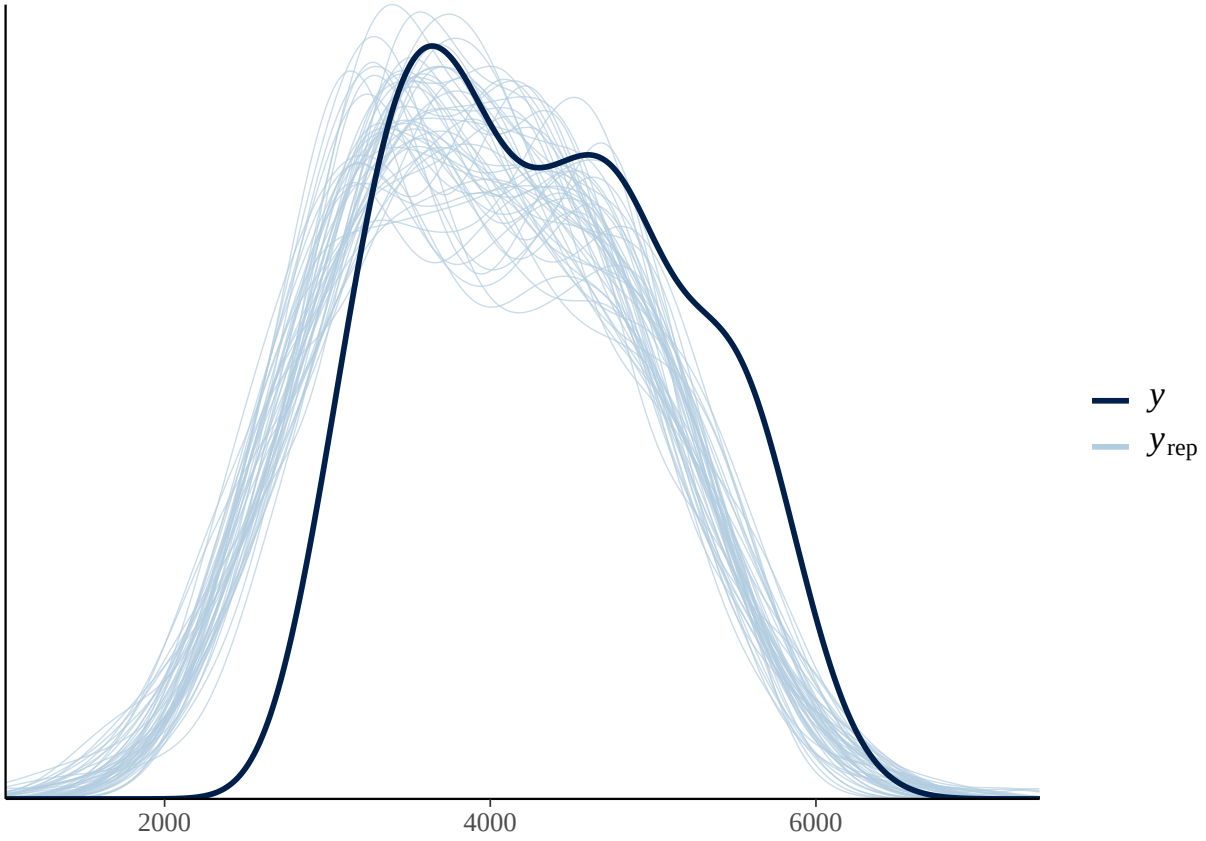
```
pp_check(penguin_model_4)
```



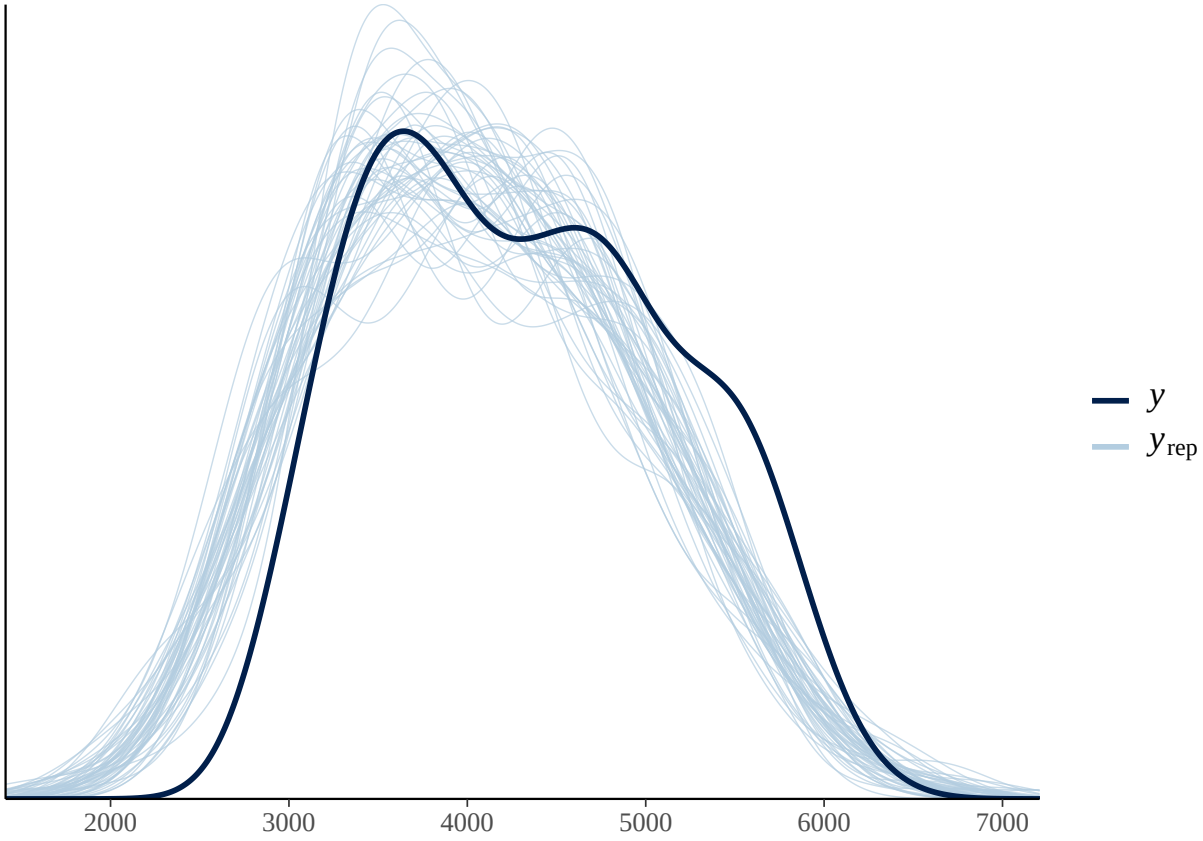

```
pp_check(penguin_model_5)
```




```
pp_check(penguin_model_6)
```




```
pp_check(penguin_model_7)
```




c. Running prediction_summary_cv() for all 4 models

```
prediction_summary_cv(model = penguin_model_4, data = penguin_data, k = 10)
```

```
## $folds
##    fold      mae mae_scaled within_50 within_95
## 1     1 277.5584  0.4312143 0.6296296 0.9629630
## 2     2 443.0565  0.7244831 0.5000000 0.9230769
## 3     3 448.3643  0.7065236 0.4074074 0.9629630
## 4     4 448.6988  0.6934619 0.5000000 0.9615385
## 5     5 483.4807  0.7232793 0.4074074 0.9629630
## 6     6 435.8434  0.6466922 0.5384615 0.9615385
## 7     7 557.5388  0.8735257 0.3846154 0.9615385
## 8     8 439.7493  0.6859029 0.4814815 1.0000000
## 9     9 638.5287  1.0093497 0.3461538 0.9615385
## 10   10 558.9304  0.8497559 0.3333333 1.0000000
##
## $cv
##        mae mae_scaled within_50 within_95
## 1 473.1749  0.7344189  0.452849  0.965812
```

```
prediction_summary_cv(model = penguin_model_5, data = penguin_data, k = 10)
```

```
## $folds
##    fold      mae mae_scaled within_50 within_95
## 1     1 478.8397  0.5888939 0.6296296 0.9629630
## 2     2 484.0497  0.5830579 0.6153846 1.0000000
## 3     3 719.8279  0.9222648 0.3333333 0.9629630
## 4     4 652.7829  0.8162966 0.3461538 0.9615385
```

```
## 5     5 711.4065  0.8911732 0.4444444 1.0000000
## 6     6 638.8281  0.7932029 0.3846154 1.0000000
## 7     7 593.7773  0.7081885 0.4615385 1.0000000
## 8     8 651.0882  0.8241095 0.4444444 0.9259259
## 9     9 598.5754  0.7384212 0.4615385 1.0000000
## 10   10 555.6032  0.7228286 0.4814815 0.8888889
##
## $cv
##        mae mae_scaled within_50 within_95
## 1 608.4779  0.7588437 0.4602564 0.9702279
```

```
prediction_summary_cv(model = penguin_model_6, data = penguin_data, k = 10)
```

```
## $folds
##    fold      mae mae_scaled within_50 within_95
## 1     1 455.0142  0.7071949 0.4074074 0.9629630
## 2     2 459.8898  0.7322272 0.4615385 0.9615385
## 3     3 421.8791  0.7022220 0.4444444 0.9629630
## 4     4 482.2685  0.6987949 0.5000000 1.0000000
## 5     5 524.1386  0.7890686 0.3703704 0.9629630
## 6     6 476.2849  0.7004069 0.5000000 1.0000000
## 7     7 473.0871  0.6867554 0.5000000 1.0000000
## 8     8 574.8380  0.9211613 0.4074074 0.9629630
## 9     9 456.3656  0.6939703 0.4615385 0.9615385
## 10   10 481.5396  0.8292572 0.4814815 0.8888889
##
## $cv
##        mae mae_scaled within_50 within_95
## 1 480.5305  0.7461059 0.4534188 0.9663818
```

```
prediction_summary_cv(model = penguin_model_7, data = penguin_data, k = 10)
```

```
## $folds
##    fold      mae mae_scaled within_50 within_95
## 1     1 405.6118  0.7672877 0.4814815 0.9259259
## 2     2 287.2404  0.5441486 0.5769231 1.0000000
## 3     3 498.0721  0.9831604 0.4444444 0.9629630
## 4     4 380.9953  0.7070793 0.4615385 0.9615385
## 5     5 408.0243  0.7555006 0.3333333 1.0000000
## 6     6 416.3024  0.8696447 0.3846154 0.9615385
## 7     7 433.7261  0.8595316 0.3846154 0.9615385
## 8     8 208.5511  0.3782775 0.7777778 0.9629630
## 9     9 347.1518  0.7428398 0.4615385 0.8846154
## 10   10 402.6337  0.7612632 0.4074074 0.9259259
##
## $cv
##        mae mae_scaled within_50 within_95
## 1 378.8309  0.7368734 0.4713675 0.9547009
```

d. Comparing ELPD for all 4 models

```
# Calculate ELPD for the 4 models
set.seed(84735)
loo_1 <- loo(penguin_model_4)
loo_2 <- loo(penguin_model_5)
loo_3 <- loo(penguin_model_6)
```

```
loo_4 <- loo(penguin_model_7)

# Results
c(loo_1$estimates[1], loo_2$estimates[1],
  loo_3$estimates[1], loo_4$estimates[1])
```

```
## [1] -2053.593 -2124.003 -2052.450 -1994.006
```

```
loo_compare(loo_1, loo_2, loo_3, loo_4)
```

```
##                 elpd_diff se_diff
## penguin_model_7    0.0       0.0
## penguin_model_6  -58.4       5.8
## penguin_model_4  -59.6       5.6
## penguin_model_5 -130.0       9.6
```

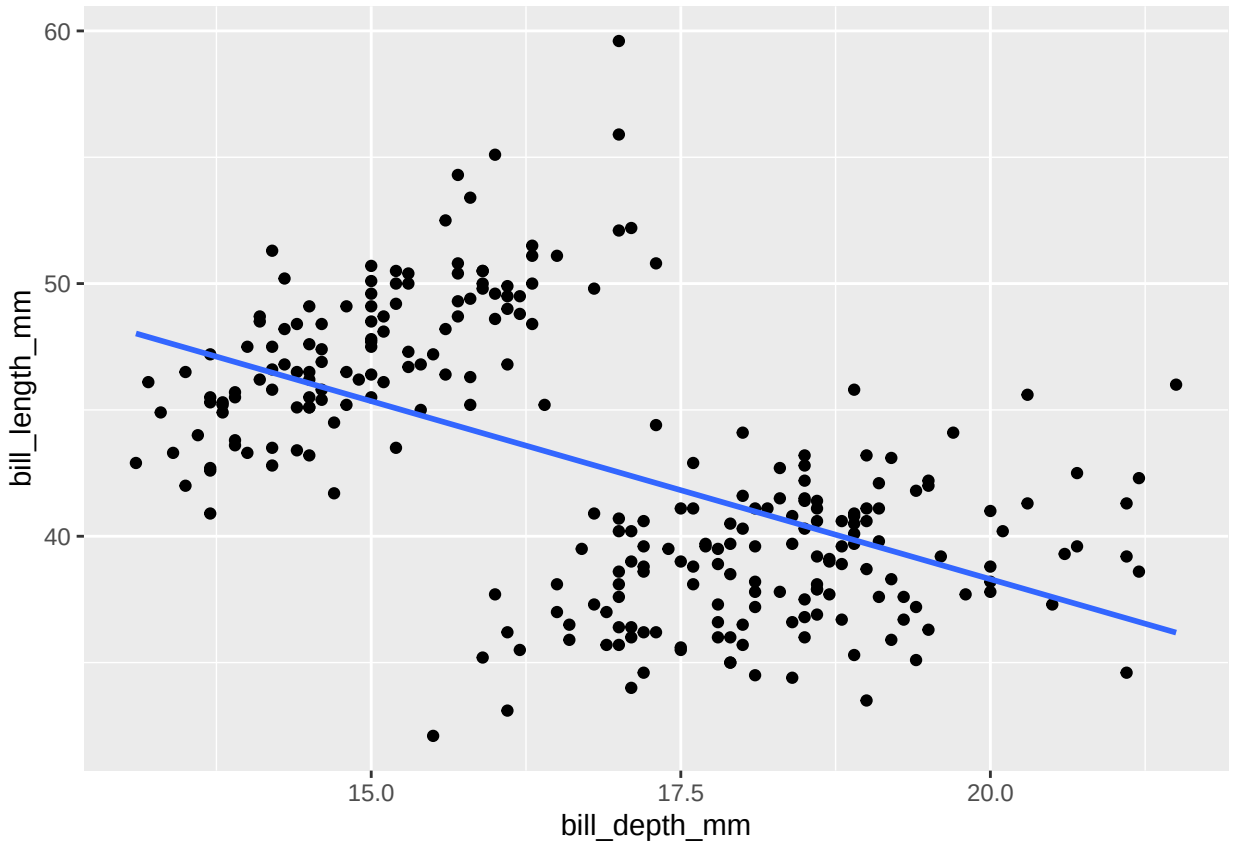
e. Penguin_model_7 has the lowest MAE_scaled (0.722), and the highest percentage of values within the 50% posterior predicted interval. Penguin_model_7 also has the best ELPD (the larger the ELPD, the more accurate the predictions)–we know this because loo_compare outputs the best model, and then lists all other models in comparison to the best model. Therefore, I would say the model 7 is the best.

## Exercise 11.14

First, let's look at the relationship between bill_length and bill_depth

```
penguin_data <- penguin_data |> drop_na()
ggplot(penguin_data, aes(x = bill_depth_mm, y = bill_length_mm)) +
 geom_point() +
  geom_smooth(method = "lm", se = FALSE)
```

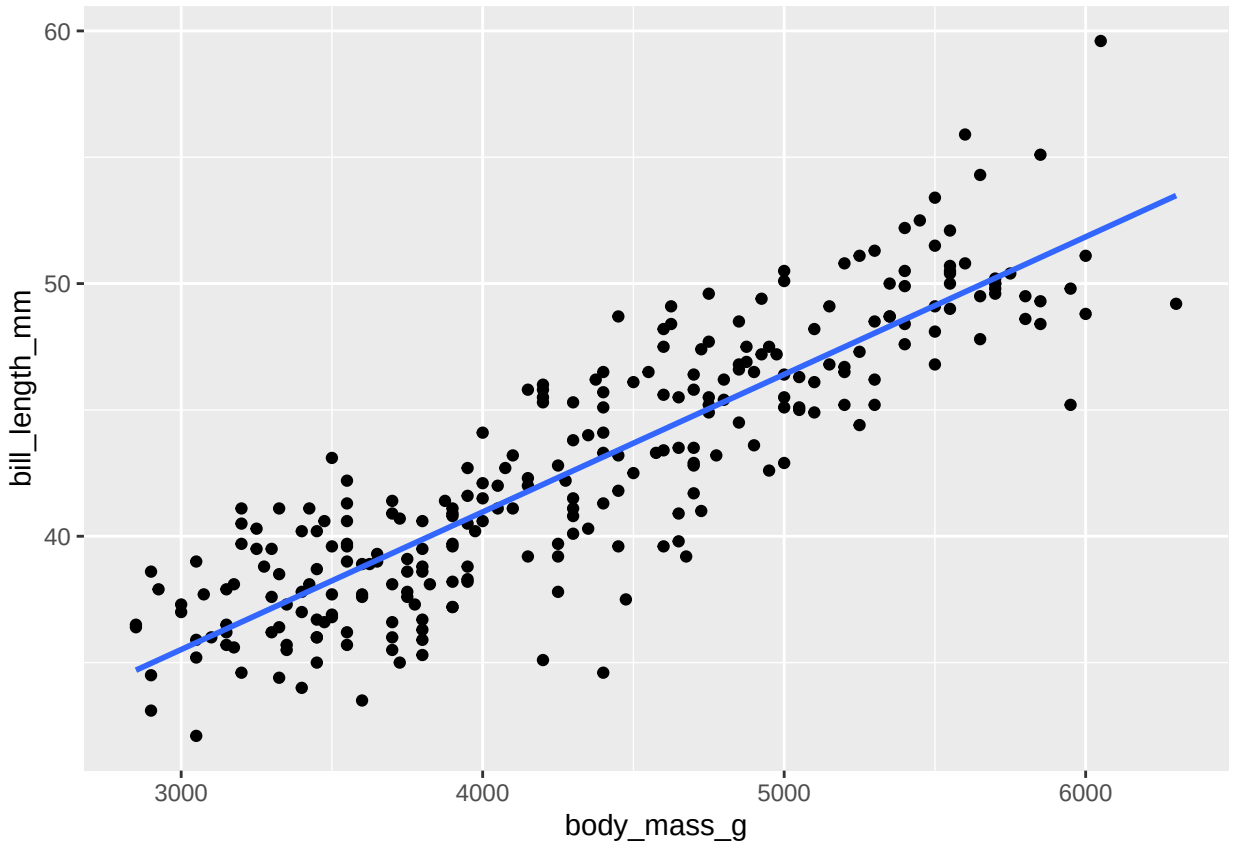
```
## `geom_smooth()` using formula 'y ~ x'
```




Now, let's check out the relationship with body_mass

```
penguin_data <- penguin_data |> drop_na()
ggplot(penguin_data, aes(x = body_mass_g, y = bill_length_mm)) +
 geom_point() +
  geom_smooth(method = "lm", se = FALSE)
```

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

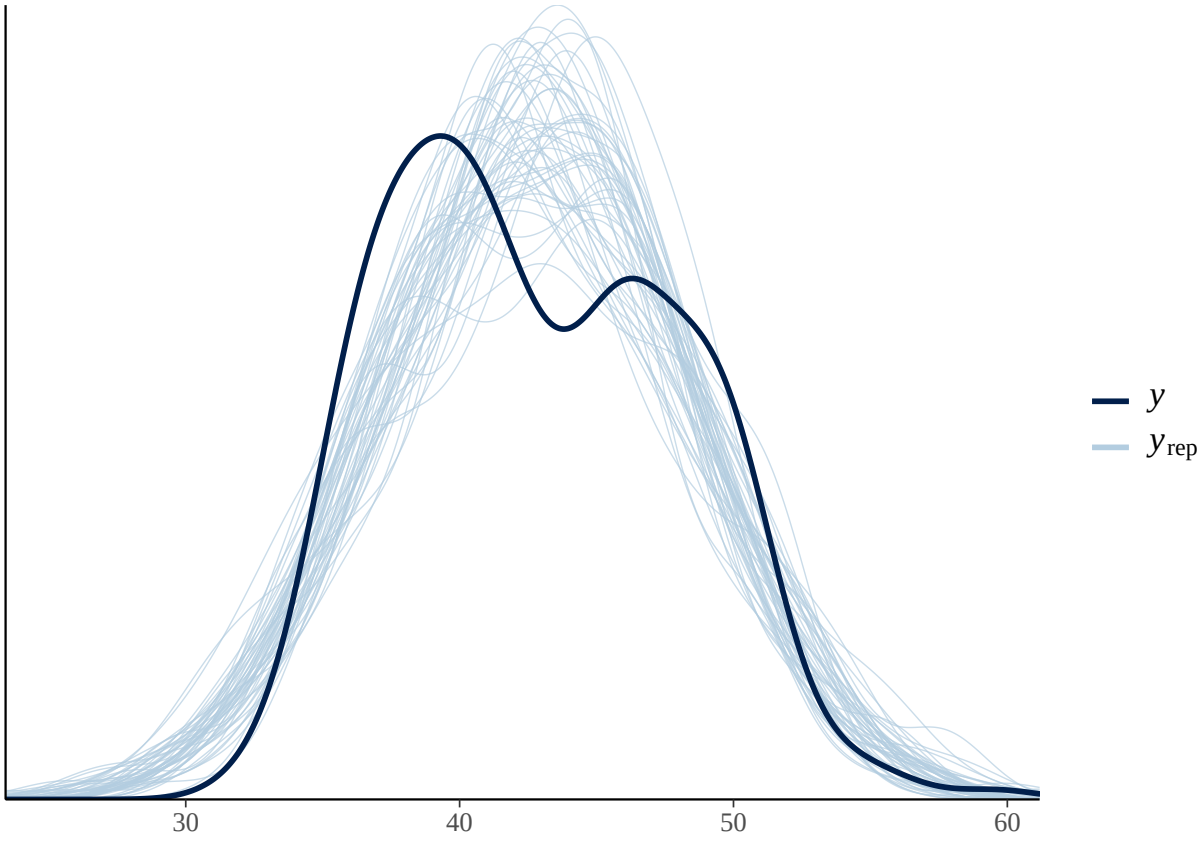



Building three penguin models

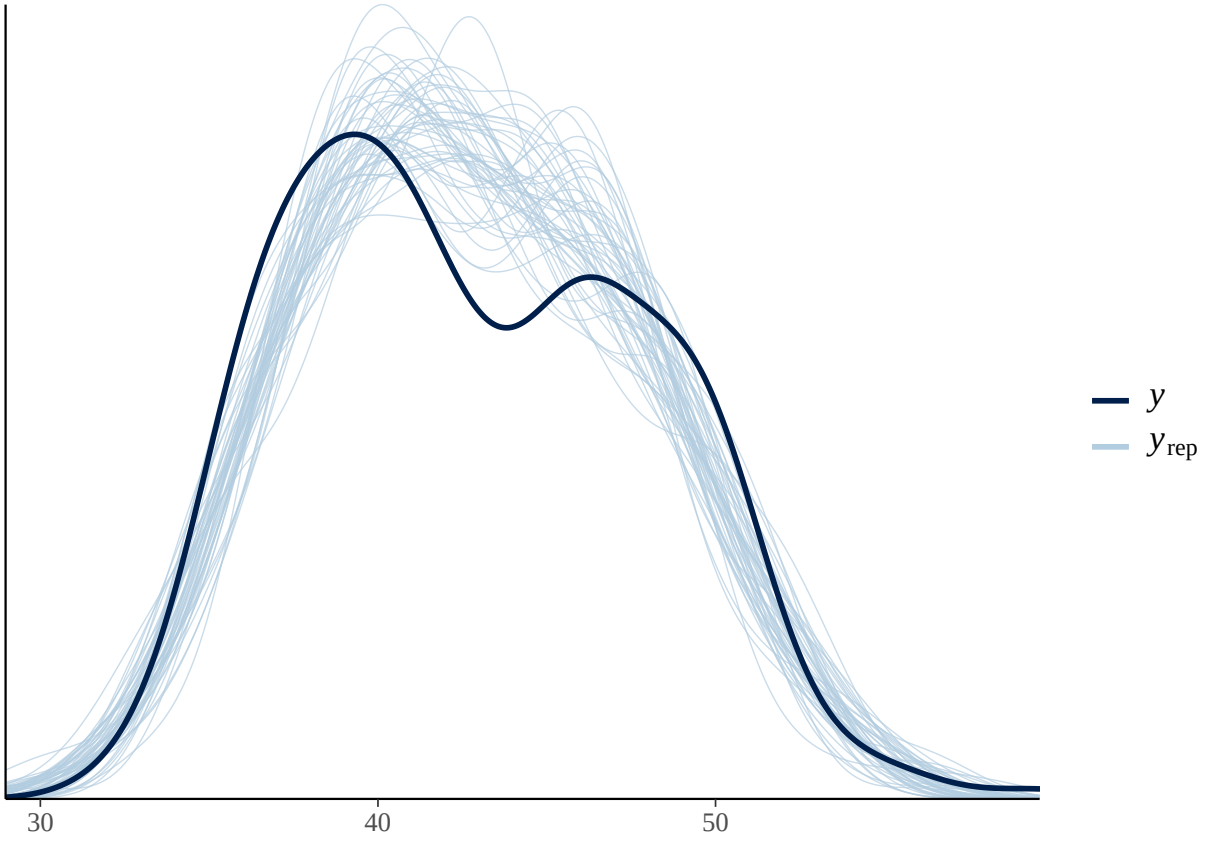
```
penguin_model_8 <- stan_glm(
  bill_length_mm ~ bill_depth_mm,
  data = penguin_data, refresh=0, family = gaussian,
  prior_intercept = normal(50, 5),
  prior = normal(-1.5, 0.005, autoscale = TRUE),
  prior_aux = exponential(0.001, autoscale = TRUE),
  chains = 4, iter = 4000*2, seed = 84735, verbose=FALSE)

penguin_model_9 <- stan_glm(
  bill_length_mm ~ body_mass_g,
  data = penguin_data, refresh=0, family = gaussian,
  prior_intercept = normal(35, 5),
  prior = normal(0.005, 0.005, autoscale = TRUE),
  prior_aux = exponential(0.001, autoscale = TRUE),
  chains = 4, iter = 4000*2, seed = 84735, verbose=FALSE)

penguin_model_10 <- stan_glm(
  bill_length_mm ~ bill_depth_mm+body_mass_g,
  data = penguin_data, refresh=0, family = gaussian,
  prior_intercept = normal(50, 5),
  prior = normal(0, 0.005, autoscale = TRUE),
  prior_aux = exponential(0.001, autoscale = TRUE),
  chains = 4, iter = 4000*2, seed = 84735, verbose=FALSE)
```

b. Evaluating models

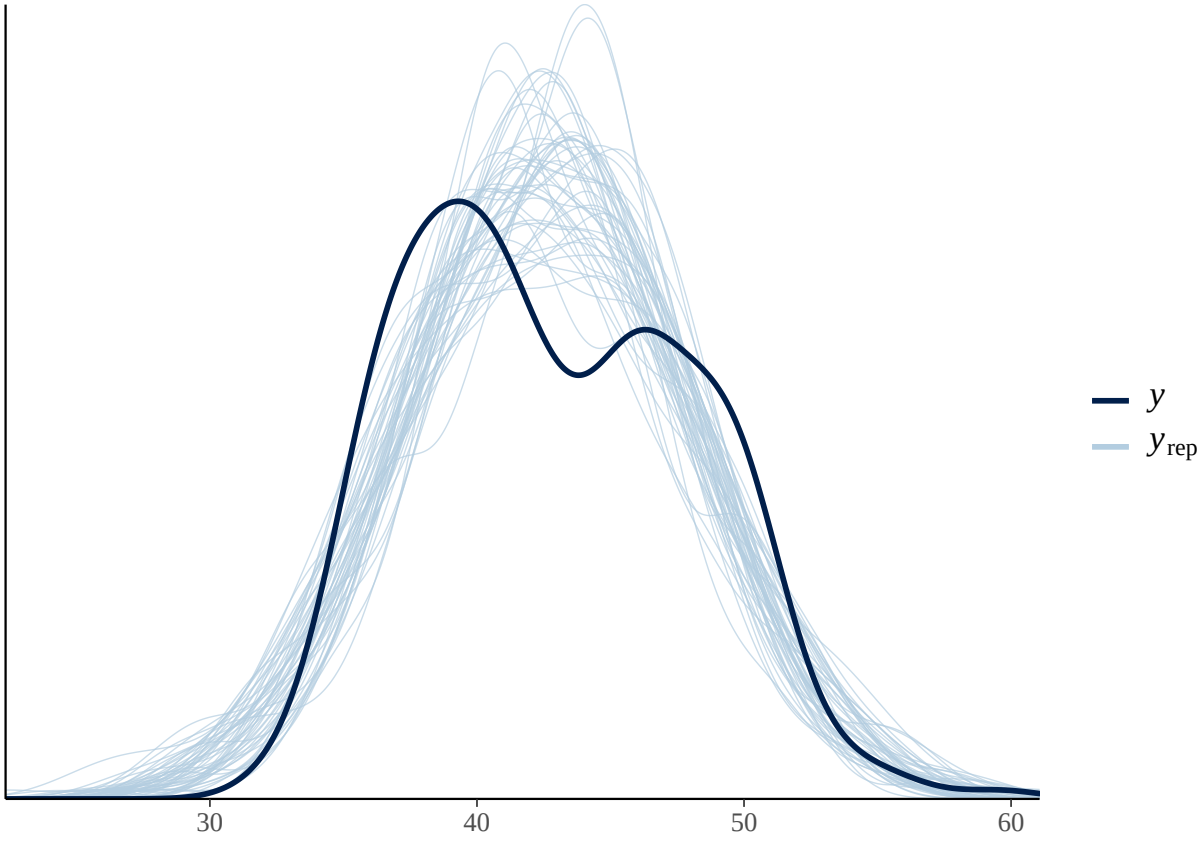
```
pp_check(penguin_model_8)
```




```
pp_check(penguin_model_9)
```




```
pp_check(penguin_model_10)
```




```
prediction_summary_cv(model = penguin_model_8, data = penguin_data, k = 10)
```

```
## $folds
##    fold      mae mae_scaled within_50 within_95
## 1     1 2.261209  0.5042860 0.7037037 0.9259259
## 2     2 3.315736  0.7566139 0.4615385 0.9230769
## 3     3 2.326412  0.5135774 0.6296296 0.9629630
## 4     4 2.130073  0.4711664 0.6923077 1.0000000
## 5     5 2.649268  0.5910488 0.5925926 0.9629630
## 6     6 4.275919  0.9758033 0.3076923 0.9615385
## 7     7 3.082429  0.6899118 0.5000000 0.9615385
## 8     8 3.273684  0.7224448 0.4444444 1.0000000
## 9     9 2.805641  0.6343910 0.5384615 0.9615385
## 10   10 4.601239  1.0966927 0.3703704 0.8888889
##
## $cv
##        mae mae_scaled within_50 within_95
## 1 3.072161  0.6955936 0.5240741 0.9548433
```

```
prediction_summary_cv(model = penguin_model_9, data = penguin_data, k = 10)
```

```
## $folds
##    fold      mae mae_scaled within_50 within_95
## 1     1 1.685671  0.6685488 0.5185185 0.9629630
## 2     2 1.234169  0.4742710 0.6923077 1.0000000
## 3     3 1.886409  0.7328654 0.4444444 0.9629630
## 4     4 1.444000  0.5696574 0.6153846 0.9615385
## 5     5 1.729281  0.6861726 0.4814815 0.8888889
```

```
## 6     6 1.554756  0.6106289 0.5384615 0.9615385
## 7     7 1.210328  0.4717706 0.6153846 0.9615385
## 8     8 2.184668  0.8537932 0.4444444 1.0000000
## 9     9 2.037541  0.7879526 0.3461538 1.0000000
## 10   10 1.723806  0.6865360 0.4444444 0.9629630
##
## $cv
##        mae mae_scaled within_50 within_95
## 1 1.669063  0.6542197 0.5141026 0.9662393
```

```
prediction_summary_cv(model = penguin_model_10, data = penguin_data, k = 10)
```

```
## $folds
##    fold      mae mae_scaled within_50 within_95
## 1     1 4.167887  0.7877940 0.4444444 1.0000000
## 2     2 3.566254  0.6726162 0.5000000 1.0000000
## 3     3 5.620458  1.0802408 0.3333333 0.9259259
## 4     4 4.081782  0.7712352 0.4615385 1.0000000
## 5     5 4.318533  0.8194117 0.3703704 1.0000000
## 6     6 3.639383  0.6973355 0.5000000 0.9615385
## 7     7 4.918917  0.9788260 0.2307692 0.9615385
## 8     8 3.594653  0.6806221 0.4814815 1.0000000
## 9     9 4.047367  0.7763710 0.3846154 1.0000000
## 10   10 3.939797  0.7596770 0.4444444 0.9259259
##
## $cv
##        mae mae_scaled within_50 within_95
## 1 4.189503   0.802413 0.4150997 0.9774929
```

As we can see from both the pp_check plots and the error statistics, the third model is significantly worse than the first two, likely due to the fact that the priors for the first two models were inferred from the data, while for the last model the prior was not able to be inferred from the data. Of the first two models, I prefer the second, for the posterior predictive distribution better matches the observed distribution, and the error statistics are smaller.

## Exercise 11.15

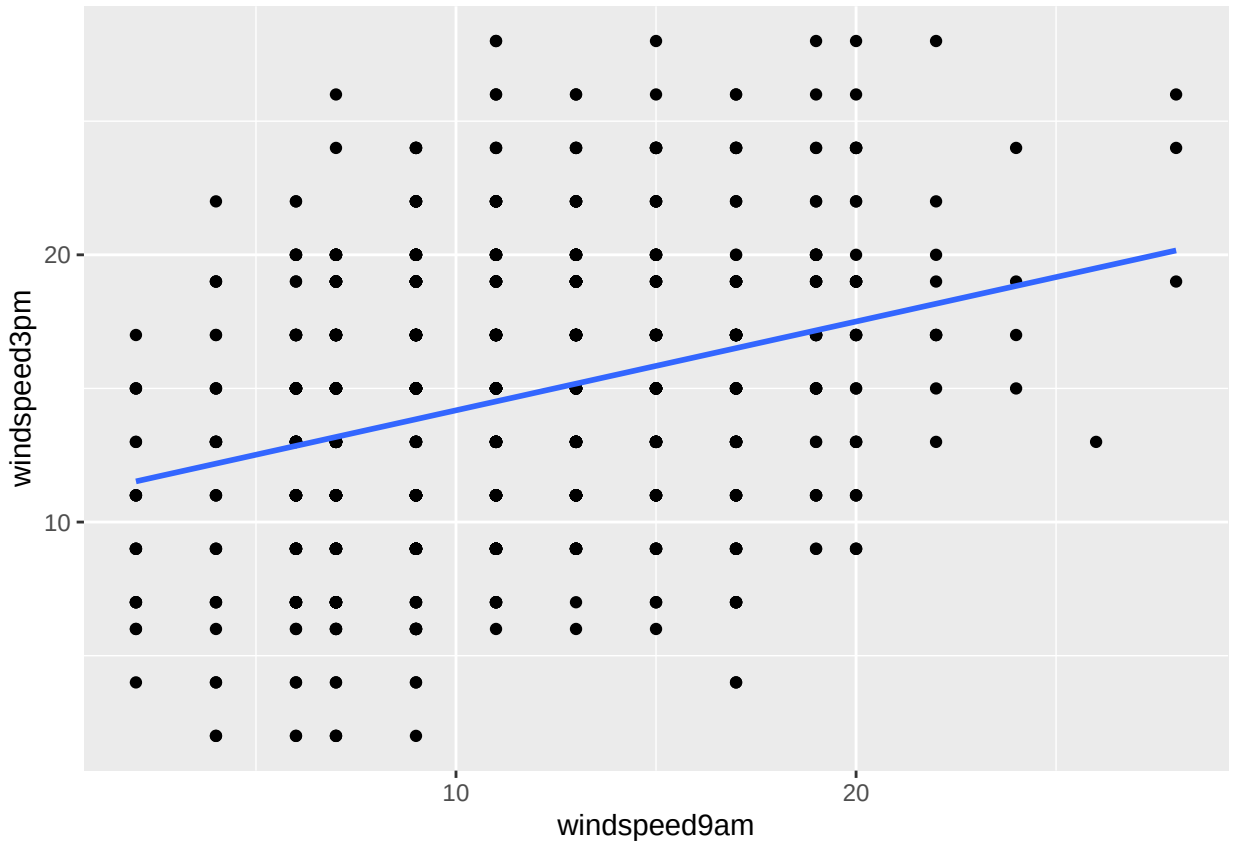
Downloading weather data

```
data("weather_perth")
# Alternative penguin data
weather_data <- weather_perth |> drop_na()
```

First, check out the data

```
ggplot(weather_data, aes(x = windspeed9am, y = windspeed3pm)) +
 geom_point() +
  geom_smooth(method = "lm", se = FALSE)
```

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

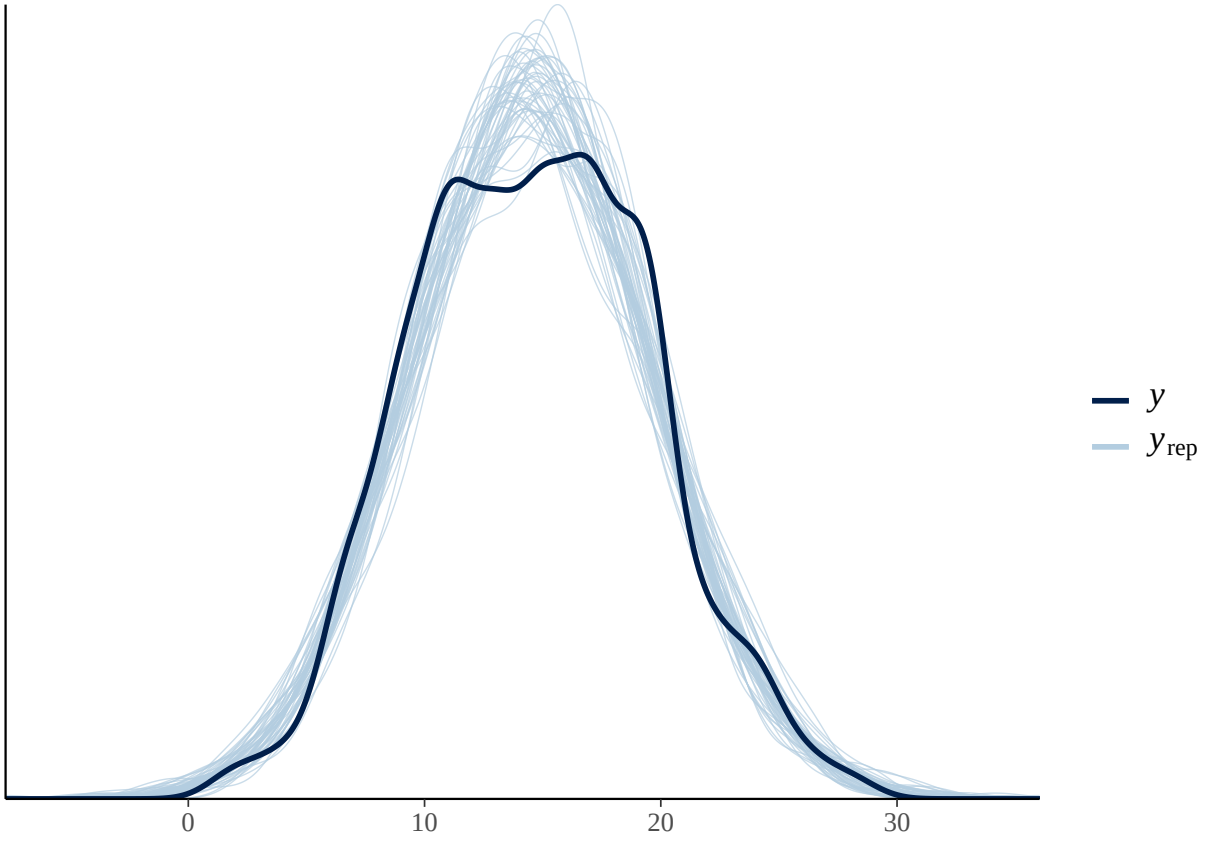



Building weather model

```
weather_model <- stan_glm(
  windspeed3pm ~ windspeed9am, refresh=0,
  data = weather_data, family = gaussian,
  prior_intercept = normal(10, 2),
  prior = normal(0.4, 0.005, autoscale = TRUE),
  prior_aux = exponential(0.001, autoscale = TRUE),
  chains = 4, iter = 4000*2, seed = 84735, verbose=FALSE)
```

b. Evaluating model

```
pp_check(weather_model)
```




```
prediction_summary_cv(model = weather_model, data = weather_data, k = 10)
```

```
## $folds
##    fold      mae mae_scaled within_50 within_95
## 1     1 3.238853  0.6902631      0.49      0.93
## 2     2 3.951084  0.8368496      0.42      0.98
## 3     3 3.277449  0.6917662      0.49      0.95
## 4     4 3.402771  0.7201394      0.48      0.95
## 5     5 3.484423  0.7303913      0.43      0.97
## 6     6 4.065505  0.8732405      0.36      0.95
## 7     7 3.849420  0.8268458      0.43      0.93
## 8     8 3.524971  0.7341515      0.46      0.98
## 9     9 3.373769  0.7124051      0.48      0.95
## 10   10 2.701749  0.5648839      0.53      0.97
##
## $cv
##        mae mae_scaled within_50 within_95
## 1 3.486999  0.7380936     0.457     0.956
```

The pp_check illustrates that the posterior predictive disribution pretty closely maps onto the observed distribution. The error statistics confirm this: the MAE scaled is 0.736 and around 46% of data values are within the 50% posterior predictive interval. Therefore, I'd say the model is pretty good, with the caveat that the prior values were inferred from the data, so the posterior model is at risk of overfitting–that is, not performing well on other datasets, since both the data and the prior hypotheses come from the same dataset.

## Final Project Notes

For my final project, I'm planning to look for a data set on social media usage and concern for digital privacy. I am curious to see how frequency of usage affects one's level of awareness/concern for privacy. I also want to see how other demographic variables relate to social media usage–including age, gender, geographic location and SES. I'm not sure I'll be able to find a dataset on this... if you have any recommendations for something similar, let me know!