BR8

Steph Jordan

```
library(rstan)
```

```
## Loading required package: StanHeaders

## Loading required package: ggplot2

## rstan (Version 2.21.2, GitRev: 2e1f913d3ca3)

## 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)
```

```
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(janitor)
```

```
##
## Attaching package: 'janitor'

## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
library(bayesrules)
library(tidyverse)
```

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

## 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
## v purrr   0.3.4

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

Exercise 8.1

1. develop the prior model

2. collect data
3. develop the posterior model by combining the data and the prior model

Exercise 8.2

a. The main drawback is that this is only one estimate of the posterior lambda; different posterior models could have yielded a range of different lambdas. What is more useful is to present the lambda value and the confidence interval associated with it.

b. There's a 95% chance that lambda will fall between 1 and 3.4.

Exercise 8.3

a. Yes: >40% of dogs have a license (hypothesis), <40% of dogs have a license (null hypothesis)
b. No–no specific proposal to test
c. Yes: >60% of voters support a new regulation, <60% of voters support the regulation
d. No–"about" 3 is not precise enough (we need greater than, less than, or equal to in order to formulate a hypothesis and null hypothesis)

Exercise 8.4

a. Posterior odds is the "odds" form of posterior probability. It is the probability that a hypothesis is true given observation of a certain outcome divided by the probability that a hypothesis is false given observation of a certain outcome. In other words, its the likelihood, given the fact that a certain condition is true, of the hypothesis being true.

b. Prior odds is simply the probability that a certain condition is true divided by the probability that it is false. There is no prior, previous condition here.

c. Bayes factor is the ratio of posterior odds to prior odds. It demonstrates how much our understanding of the hypothesis changed given our observed data.

Exercise 8.5

a. Sampling variability in the data (the mean observed depends on the exact random sample we get) and posterior variability in pi (our confidence interval illustrates our confidence in the mean posterior probability). The former refers to deviation in all the samples we could draw from our population; the latter refers to the fact that our posterior estimate for the average is still just an estimate–it is not *the* average. Therefore, we must consider all possible values of pi, even though some are more likely than others.

b. How many of the incoming students will major in computer science.

c. It is conditional on both the data and the parameter.

Exercise 8.6

a. We'll use qbeta()

```
qbeta(c(0.025, 0.975), 4, 5)
```

```
## [1] 0.1570128 0.7551368
```

b. We'll use qbeta()

```
qbeta(c(0.2, 0.8), 4, 5)
```

```
## [1] 0.3032258 0.5836554
```

c. We'll use qgamma()

```
qgamma(c(0.025, 0.975), 1, 8)
```

```
## [1] 0.003164726 0.461109932
```

Exercise 8.7

a. We'll use qgamma()

```
qgamma(c(0.005, 0.995), 1, 5)
```

```
## [1] 0.001002508 1.059663473
```

b. We'll use qnorm()

```
qnorm(c(0.025, 0.975), 10, 2)
```

```
## [1]  6.080072 13.919928
```

c. We'll use qnorm()

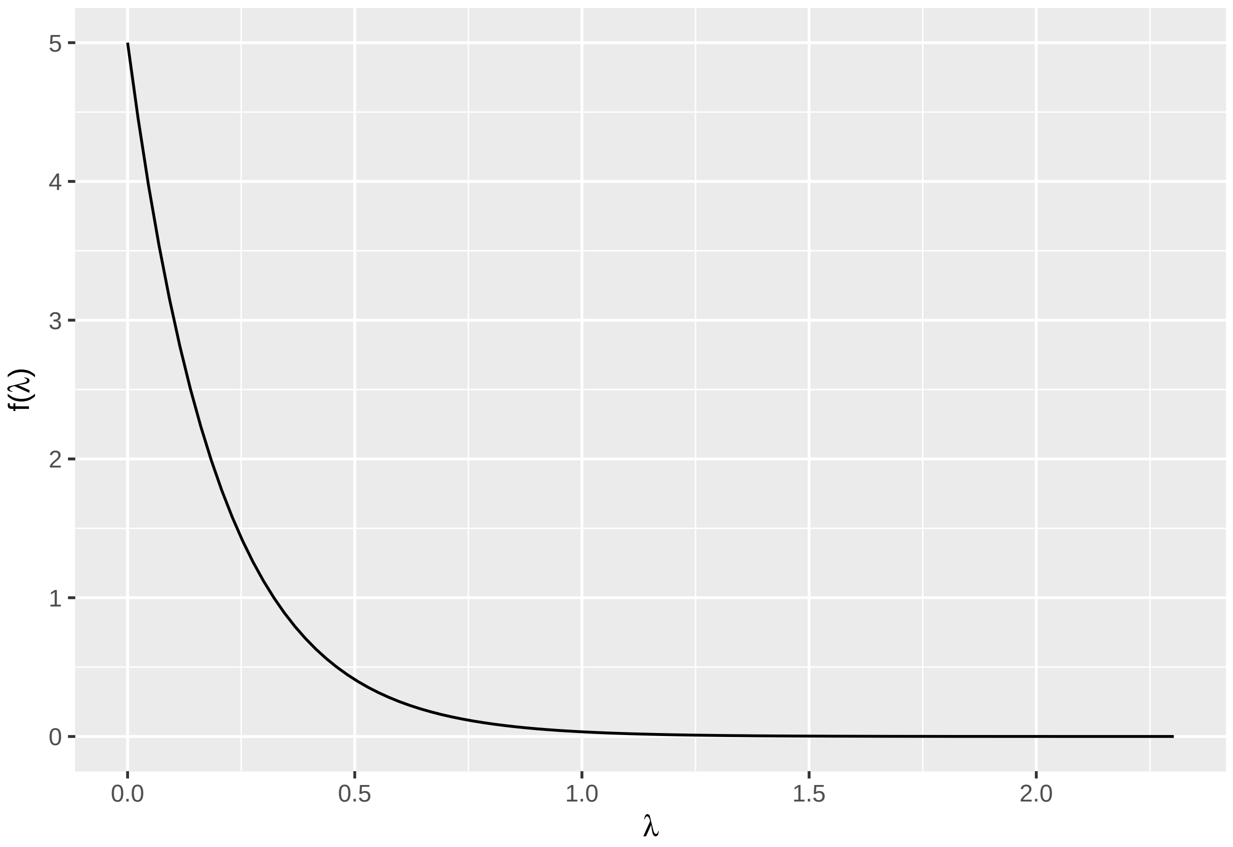
```
qnorm(c(0.005, 0.995), -3, 1)
```

```
## [1] -5.5758293 -0.4241707
```

Exercise 8.8

a. We will first plot the distribution to see visually where the top 95% might be.

```
plot_gamma(1, 5)
```



We know from this plot that the bottom of the confidence interval will be percentile 0 (lambda==0), and we can use our quantile function to figure out where 0+95=95th percentile is.

```r
qgamma(c(0, 0.95), 1, 5)
```

```
## [1] 0.0000000 0.5991465
```

The confidence interval is (0, 0.599)

b. Using the middle 95 approach, we get:

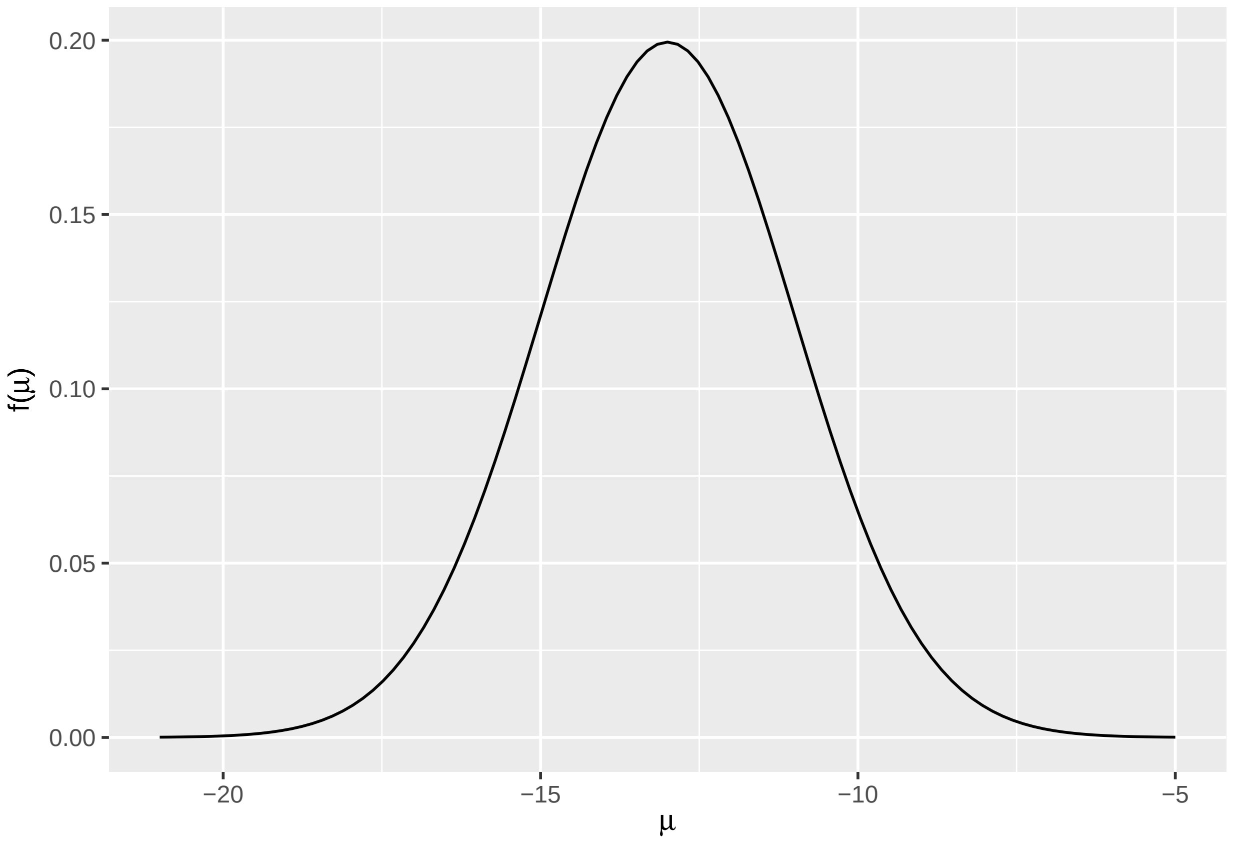
```r
qgamma(c(0.025, 0.975), 1, 5)
```

```
## [1] 0.005063562 0.737775891
```

c. They are not the same. The middle 95% reaches a significantly higher upper bound than the highest posterior density credible interval. The highest posterior density credible interval is more applicable here, because the graph is significiantly skewed, so it produces a more accurate estimation of the range of values of pi than the middle 95% estimation.

d. We'll start by plotting the distribution

```r
plot_normal(-13, 2)
```



Since the majority of values are distributed symmetrically around the mean/mode (-13), we know the highest 95% interval will be equivalent to the middle 95%. Therefore, we can use our known quantile calculations to determine the highest 95% interval.

```r
qnorm(c(0.025, 0.0975), -13, 2)
```

```
## [1] -16.91993 -15.59186
```

e. Using the middle 95% approach yields the same result:

```
qnorm(c(0.025, 0.0975), -13, 2)
```

```
## [1] -16.91993 -15.59186
```

f. They are the same, since the values are distributed symmetrically around the mean. Therefore, the middle 95% will be equivalent to the highest 95%.

Exercise 8.9

a. We'll use the method demonstrated in the textbook, and subtract 1 since we are testing for pi>0.4

```
post_prob <- 1- pbeta(0.40, 4, 3)
post_prob
```

```
## [1] 0.8208
```

b. We'll again use the formula for posterior odds given in the book

```
post_odds <- post_prob/(1-post_prob)
post_odds
```

```
## [1] 4.580357
```

c. Same formula as posterior odds but with prior probabilities

```
prior_prob <- 1- pbeta(0.4, 1, 0.8)
prior_odds <- prior_prob/(1-prior_prob)
prior_odds
```

```
## [1] 1.98098
```

d. Bayes factor is the result of dividing the prior and posterior:

```
post_odds/prior_odds
```

```
## [1] 2.312168
```

Since BF>1, the plausibility of the alternative hypothesis being true increased given the data.

e. The alternative hypothesis (pi>0.4) is more likely than the null hypothesis (pi<=0.4), and the greater probability of the alternative strengthens in light of the observed data.

Exercise 8.10

a. We'll use the method demonstrated in the textbook

```
post_prob <- pnorm(0.52, 5, 3)
post_prob
```

```
## [1] 0.06767498
```

b. We'll again use the formula for posterior odds given in the book

```
post_odds <- post_prob/(1-post_prob)
post_odds
```

```
## [1] 0.07258732
```

c. Same formula as posterior odds but with prior probabilities

```
prior_prob <- pnorm(0.52, 10, 10)
prior_odds <- prior_prob/(1-prior_prob)
prior_odds
```

```
## [1] 0.2070949
```

d. Bayes factor is the result of dividing the prior and posterior:

```
post_odds/prior_odds
```

```
## [1] 0.3505027
```

Since BF<1, the plausibility of the alternative hypothesis being true decreased given the data.

e. The alternative hypothesis (pi<0.52) is less likely than the null hypothesis (pi>=0.52), and the data strengthened the fact that the alternative hypothesis is less likely than the null hypothesis.

Exercise 8.14

a. Beta-Binomial, because pi will represent a fraction between 0 and 1 of the percentage of adults who don't believe in climate change.

b. Beta(1, 4): I think that most adults in the U.S. now believe in climate change. I would guess 1/4 (25%) of adults do not.

c. The authors are much more pessimistic; they believe a greater percentage of US adults do not believe in climate change.

d. Loading data

```
data(pulse_of_the_nation)
```

```
glimpse(pulse_of_the_nation)
```

```
## Rows: 1,000
## Columns: 15
## $ income             <dbl> 8, 68, 46, 51, 100, 54, 83, 114, 90, 5, 57, 47, 39, ~
## $ age                <dbl> 64, 56, 63, 48, 32, 64, 61, 64, 64, 68, 63, 44, 57, ~
## $ party              <fct> Democrat, Democrat, Independent, Republican, Democra~
## $ trump_approval     <fct> Strongly disapprove, Strongly disapprove, Somewhat A~
## $ education          <fct> College degree, High school, Some college, High scho~
## $ robots             <fct> Unlikely, Unlikely, Unlikely, Unlikely, Unlikely, Un~
## $ climate_change     <fct> Real and Caused by People, Real and Caused by People~
## $ transformers       <dbl> 1, 0, 0, 0, 1, 0, 0, 2, 0, 0, 0, 1, 0, 0, 1, 1, 5, 1~
## $ science_is_honest <fct> Strongly Agree, Somewhat Agree, Somewhat Agree, Some~
## $ vaccines_are_safe <fct> Somewhat Disagree, Somewhat Disagree, Strongly Agree~
## $ books              <dbl> 20, 6, 0, 1, 30, 15, 0, 40, 0, 1, 0, 2, 12, 6, 6, 24~
## $ ghosts             <fct> Yes, No, No, No, Yes, No, Yes, No, No, No, No, Yes, ~
## $ fed_sci_budget     <fct> Too Low, Too High, About Right, About Right, Too Low~
## $ earth_sun          <lgl> FALSE, FALSE, TRUE, FALSE, FALSE, TRUE, TRUE, TRUE, ~
## $ wise_unwise        <fct> Wise but Unhappy, Wise but Unhappy, Happy but Unwise~
```

```
climate <- pulse_of_the_nation |> select(climate_change)
climate
```

```
## # A tibble: 1,000 x 1
##    climate_change
##    <fct>
##  1 Real and Caused by People
##  2 Real and Caused by People
##  3 Real but not Caused by People
##  4 Not Real At All
##  5 Real and Caused by People
##  6 Real and Caused by People
##  7 Real but not Caused by People
##  8 Real and Caused by People
##  9 Real and Caused by People
## 10 Real and Caused by People
## # ... with 990 more rows
```

```
clim_count <- climate |> summarise(x=length(climate_change))
clim_count
```

```
## # A tibble: 1 x 1
##       x
##   <int>
## 1  1000
```

```
not_real <- climate |> filter(climate_change=="Not Real At All") |> summarise(x=length(climate_change))
not_real
```

```
## # A tibble: 1 x 1
##       x
##   <int>
## 1   150
```

Calculate the sample proportion

```
not_real/clim_count
```

```
##      x
## 1 0.15
```

e. Our posterior model of pi is as follows:

$$Beta(151, 852)$$

   A middle 95% CI is as follows:

```
qbeta(c(0.025, 0.975), 151, 852)
```

```
## [1] 0.1291000 0.1733164
```

Exercise 8.15

a. It is more likely that pi is greater than 0.1 than pi is less than 0.1, since the entire 95% middle confidence interval is above 0.1.
b. Calculating posterior probability of the alternative hypothesis:

```
post_prob <- 1- pbeta(0.1, 151, 852)
post_prob
```

```
## [1] 0.9999997
```

It is highly likely (99%) that pi is greater than 0.1.

c. Calculating the Bayes factor

```
post_odds <- post_prob/(1-post_prob)

prior_prob <- 1- pbeta(0.1, 1, 2)
prior_odds <- prior_prob/(1-prior_prob)

post_odds/prior_odds
```

```
## [1] 750017.6
```

Since the Bayes factor is VERY much greater than 1, the plausibility of the alternative hypothesis being true greatly increased with the data.

d. Pi is likely to be greater than 0.1.

Exercise 8.16

a. Building the MCMC model

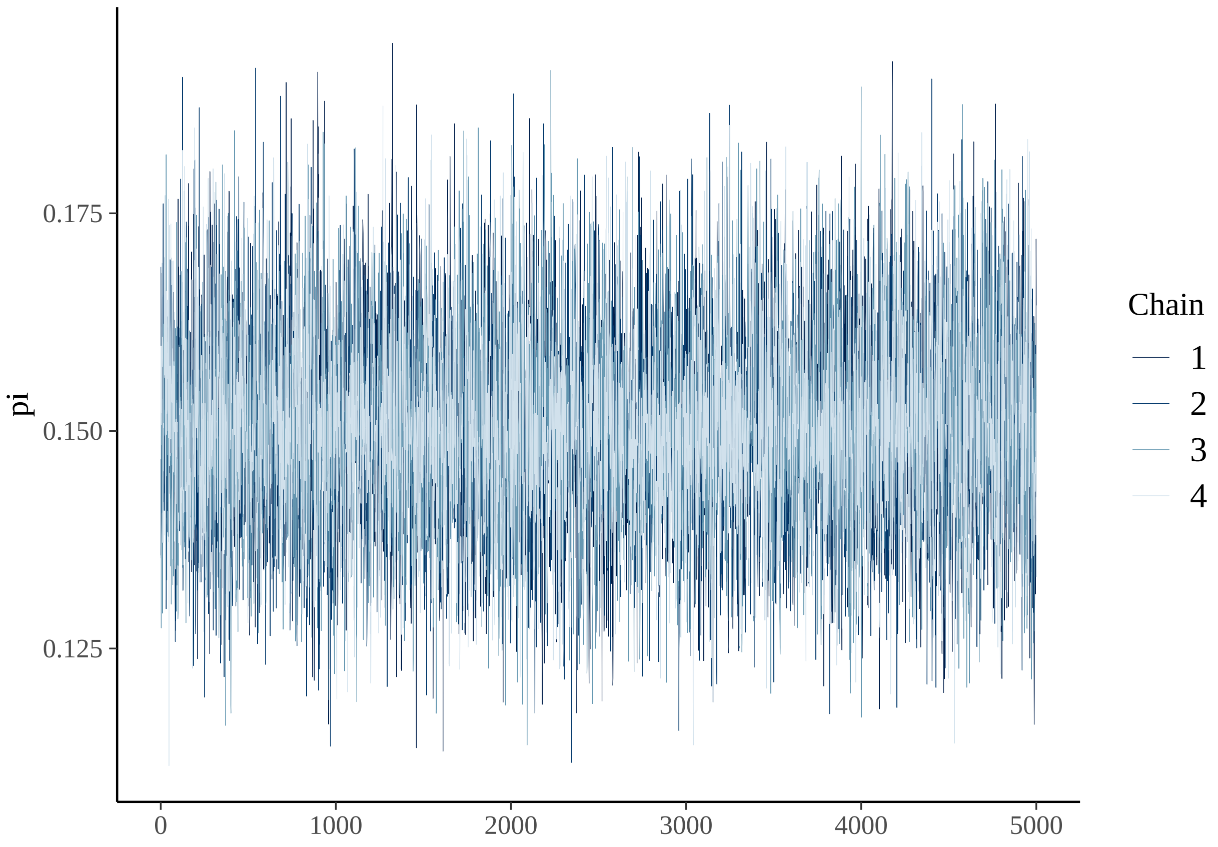
```
# STEP 1: DEFINE the model
bb_model <- '
  data {
    int<lower = 0, upper = 1000> Y;
  }
  parameters {
    real<lower = 0, upper = 1> pi;
  }
  model {
    Y ~ binomial(1000, pi);
    pi ~ beta(1, 2);
  }
'
```

```
# STEP 2: SIMULATE the posterior
bb_sim <- stan(model_code = bb_model, data = list(Y = 150),
               chains = 4, iter = 5000*2, seed = 84735)
```

```
## Trying to compile a simple C file
```

Illustrate the traces

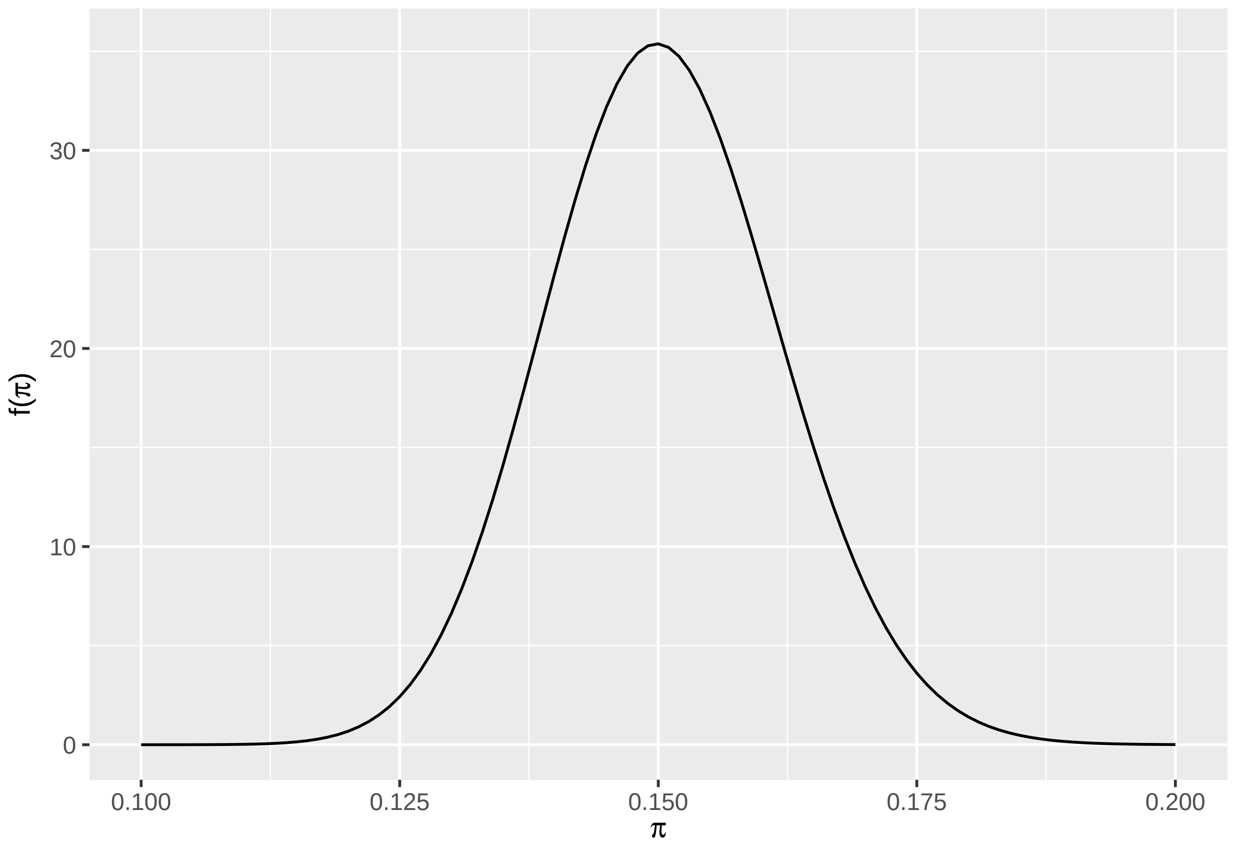
```
mcmc_trace(bb_sim, pars = "pi", size = 0.1)
```



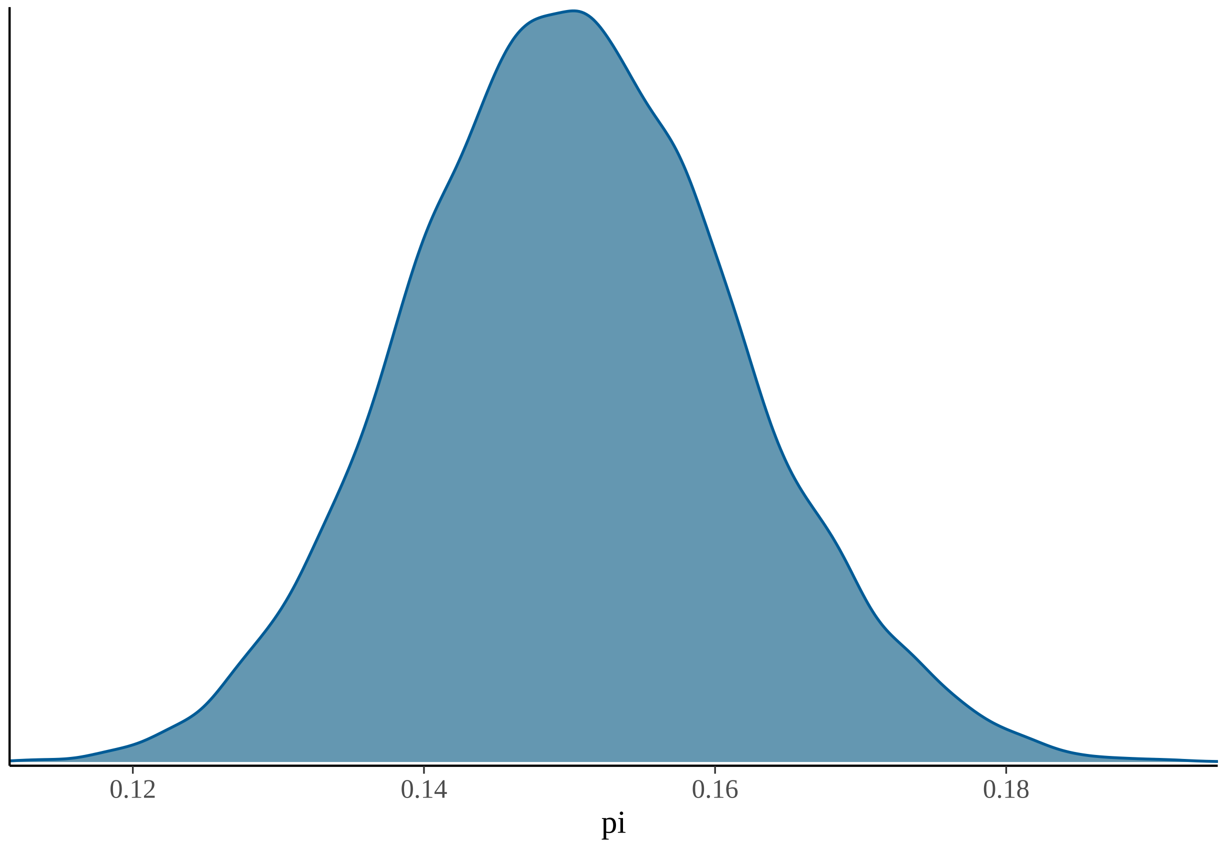
Exercise 8.17

a. Approximating 95% confidence interval. First, we'll plot the data

```
plot_beta(alpha = 151, beta = 852) + lims(x = c(0.1,0.2))
```



```r
# MCMC posterior approximation
mcmc_dens(bb_sim, pars = "pi")
```



Now, we'll get the interval from the MCMC approximation

```
bb_chains_df <- as.data.frame(bb_sim, pars = "lp__", include = FALSE)
# Calculate posterior summaries of pi
bb_chains_df %>%
  summarize(post_mean = mean(pi),
            post_median = median(pi),
            post_mode = sample_mode(pi),
            lower_95 = quantile(pi, 0.025),
            upper_95 = quantile(pi, 0.975))
```

```
##   post_mean post_median post_mode  lower_95  upper_95
## 1 0.1503917   0.1501722 0.1502598 0.1286285 0.1734771
```

Now we'll find the same information using the tidy shortcut

```
library(broom.mixed)
tidy(bb_sim, conf.int = TRUE, conf.level = 0.95)
```

```
## # A tibble: 1 x 5
##   term  estimate std.error conf.low conf.high
##   <chr>    <dbl>     <dbl>    <dbl>     <dbl>
## 1 pi       0.150    0.0113    0.129     0.173
```

b. We'll use the method outlined in the book:

```
bb_chains_df %>%
  mutate(exceeds = pi > 0.10) %>%
  tabyl(exceeds)
```

```
##   exceeds     n percent
##      TRUE 20000       1
```

According to the MCMC model, all approximated values are greater than 0.1 (so there is a 100% chance that pi>0.1).

c. The lower bound of the 95% confidence interval is very close to our estimates from 8.14 (0.12 to 0.17).

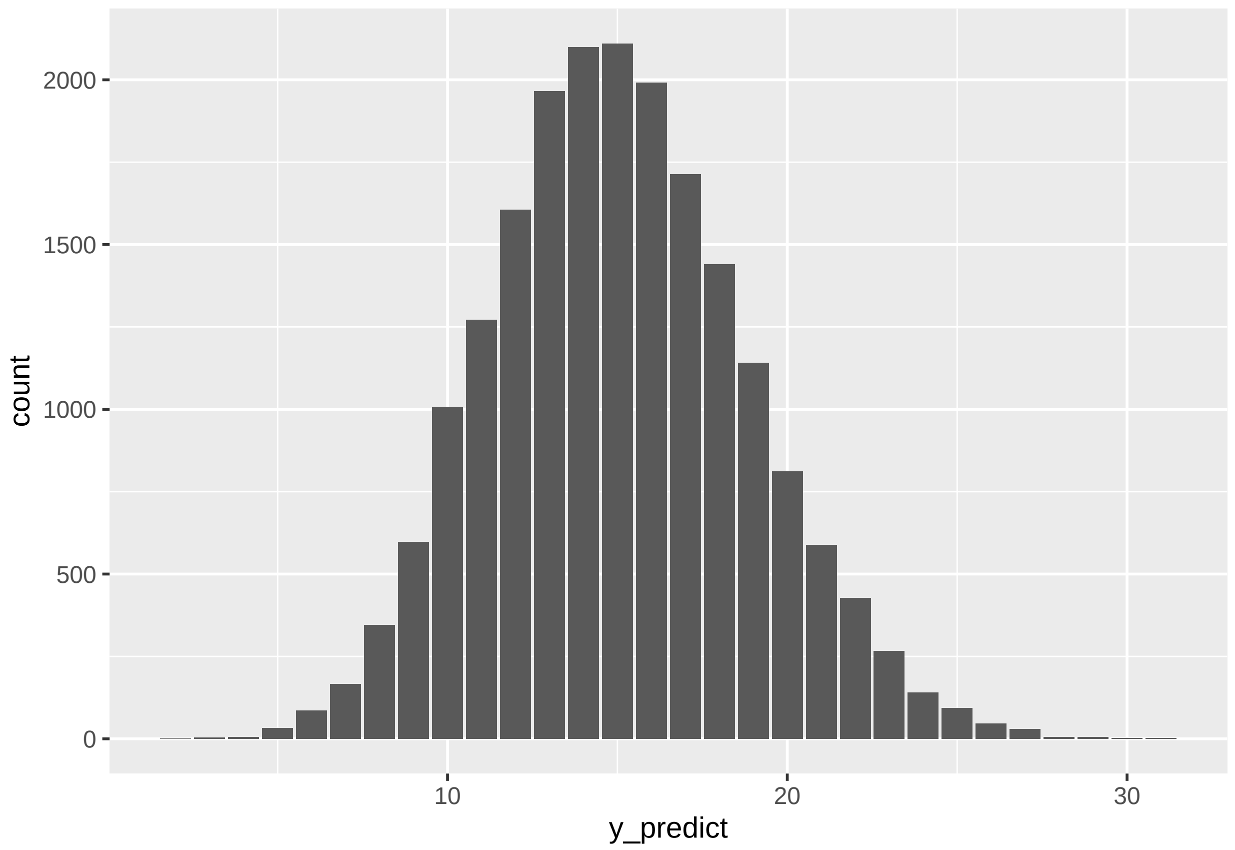
Exercise 8.18

a. We'll again follow the book's suggestions:

```
set.seed(1)

# Predict a value of Y' for each pi value in the chain
bb_chains_df <- bb_chains_df %>%
  mutate(y_predict = rbinom(length(pi), size = 100, prob = pi))

# Check it out
bb_chains_df %>%
  head(3)
```

```
##          pi y_predict
## 1 0.1576199        13
## 2 0.1357043        12
## 3 0.1372085        14
```

```
# Plot the 20,000 predictions
ggplot(bb_chains_df, aes(x = y_predict)) +
  stat_count()
```



b.

The number of adults who don't believe in climate change, after adding 100 more into our sample pool, is most likely to range between 12 and 18.

c. Finding probability that at least 20 don't believe in climate change NOTE: not sure methodology here

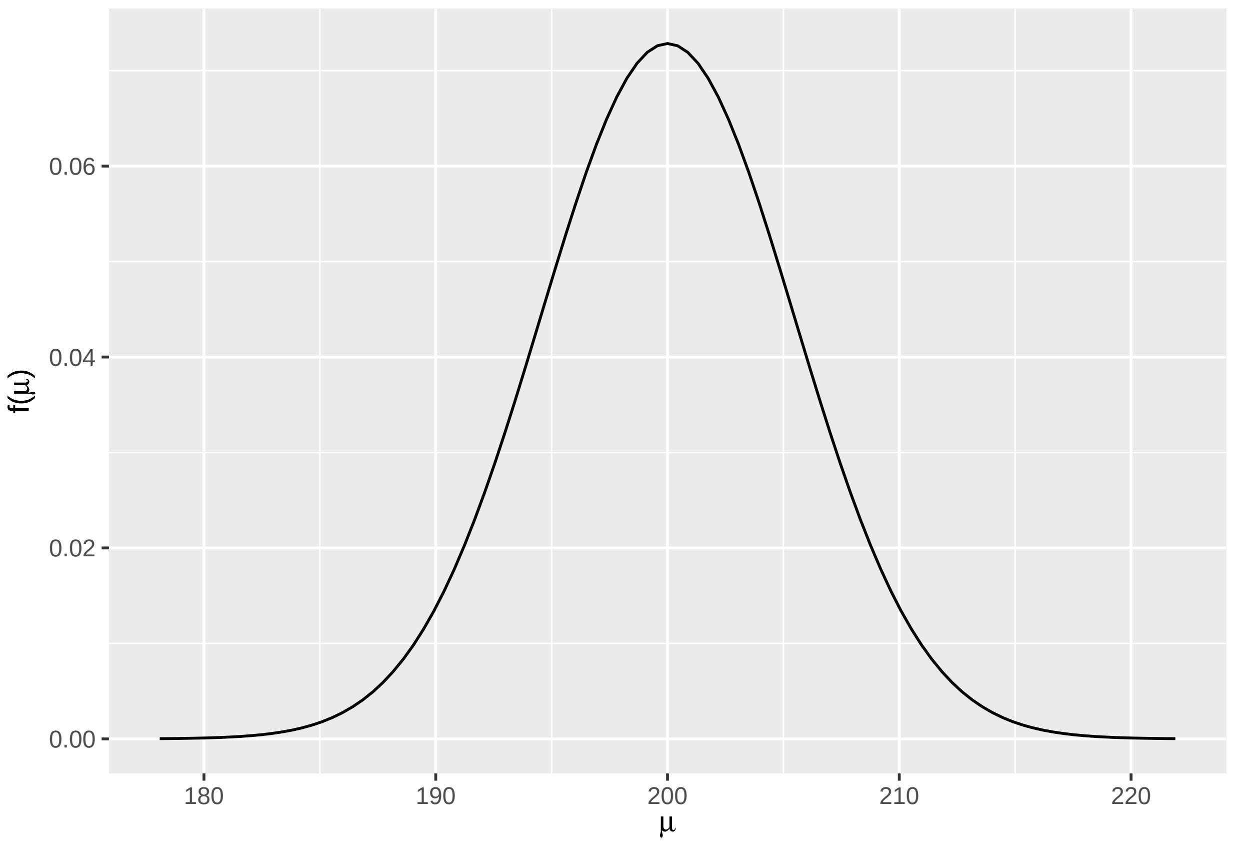
```
denom <- bb_chains_df  %>%
  summarize(total=length(pi))

nom <- bb_chains_df %>%
  filter(y_predict>=20) %>%
  summarize(total=length(pi))

nom/denom
```

```
##    total
## 1 0.1209
```

Exercise 8.19

a. Normal-Normal, because measurements are likely to be symmetrical (like height)
b. As shown below, Normal(200, 30^(1/2)) yields a majority of values between 140 and 260.

```
plot_normal(200, 30^(1/2))
```



c. Calculating stats from penguins package

```
data("penguins_bayes")
glimpse(penguins_bayes)
```

```
## Rows: 344
## Columns: 9
## $ species              <fct> Adelie, Adelie, Adelie, Adelie, Adelie, Adelie, A~
## $ island               <fct> Torgersen, Torgersen, Torgersen, Torgersen, Torge~
## $ year                 <int> 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2~
## $ bill_length_mm       <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 38.9, 39.2, 34.~
## $ bill_depth_mm        <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 17.8, 19.6, 18.~
## $ flipper_length_mm    <int> 181, 186, 195, NA, 193, 190, 181, 195, 193, 190, ~
## $ body_mass_g          <int> 3750, 3800, 3250, NA, 3450, 3650, 3625, 4675, 347~
## $ above_average_weight <fct> 0, 0, 0, NA, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, ~
## $ sex                  <fct> male, female, female, NA, female, male, female, m~
```

```
pengs <- penguins_bayes |> filter(species=="Adelie")
pengs <- pengs |> drop_na()
pengs |>  summarize(mean=mean(flipper_length_mm), num=length(flipper_length_mm), sd=sd(flipper_length_mm
```

```
## # A tibble: 1 x 3
##    mean   num    sd
##   <dbl> <int> <dbl>
## 1  190.   146  6.52
```

d. Posterior model

```
summarize_normal_normal(mean=200, sd=5.48, sigma=6.52, y_bar=190.1, n=146)
```

```
##       model     mean     mode        var       sd
## 1     prior 200.0000 200.0000 30.0304000 5.480000
## 2 posterior 190.1951 190.1951  0.2883711 0.537002
```

Calculating 95% confidence interval:

```
qnorm(c(0.025, 0.975), 190.1951, 0.537)
```

```
## [1] 189.1426 191.2476
```

Exercise 8.20

a. Ha: mu in {200, 220}; Ho: mu is not in {200, 220}
b. This hypothesis should be about 1/4 of 95% (since 1/2 of this range–200 to 210– spans about 1/2 the confidence interval).
c. Calculating the posterior probability of our hypothesis using the posterior developed from our MCMC approximation:

```
post_prob_mu_greater_200 <- 1 - pnorm(200, 190.1951, 0.537)
post_prob_mu_less_220 <- pnorm(220, 190.1951, 0.537)
post_prob_mu_less_220*post_prob_mu_greater_200
```

```
## [1] 0
```

Given that there's a 95% chance that mu is between 189 and 191 according to our confidence interval from 8.19, and there is a mean of 191, it makes sense that there would be a 0% chance of mu being between 200 and 220.

d. Mu is likely to be around 191, and will most likely range between 189 and 191.

Exercise 8.21

a. Gamma-Poison, because this is a count per hour that will likely be unsymmetrical in distribution.
b. We'll use some equations from the textbook

$$sd = s^{1/2}/r$$

$$mean = s/r$$

Rearranging these formulas and subbing in 2 for the mean and 1 for the sd, we have: **r=2, s=4**.

Therefore, the prior can be modelled as: Gamma(4, 2)

c. Checking out loon data

```
data("loons")
glimpse(loons)
```

```
## Rows: 18
## Columns: 5
## $ year           <dbl> 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2~
## $ count          <dbl> 0, 7, 0, 1, 1, 4, 1, 4, 2, 2, 0, 9, 3, 0, 5, 7, 2, 5
## $ hours          <dbl> 143.50, 153.50, 228.30, 248.75, 203.75, 245.75, 223.00,~
## $ count_per_hour <dbl> 0.000000000, 0.045602606, 0.000000000, 0.004020101, 0.0~
## $ count_per_100  <dbl> 0, 5, 0, 0, 0, 2, 0, 2, 1, 1, 0, 4, 2, 0, 3, 3, 1, 3
```

```
loons <- loons |> drop_na()
loons |>  summarize(mean=mean(count_per_100), num=length(count_per_100), sd=sd(count_per_100), sum=sum(
## # A tibble: 1 x 4
##    mean   num    sd   sum
##   <dbl> <int> <dbl> <dbl>
## 1   1.5    18  1.58    27
```

d. First, specify the posterior model:

```
summarize_gamma_poisson(4, 2, 27, 18)
```

```
##       model shape rate mean mode    var        sd
## 1     prior     4    2 2.00  1.5 1.0000 1.0000000
## 2 posterior    31   20 1.55  1.5 0.0775 0.2783882
```

Calculating the 95% confidence interval:

```
qgamma(c(0.025, 0.975), 31, 20)
```

```
## [1] 1.053150 2.141343
```

The data, which has a mean (1.5) that is less than the prior mean (2), brings the posterior confidence interval down (ranging from 1.05 to 2.14).

Exercise 8.22

a. Ha: lambda is<1; Ho lambda is >=1
b. The alternative hypothesis is unlikely considering that the lower bound of the 95% confidence interval is 1.05.
c. Calculating probability of Ha:

```
post_prob <- pnorm(1, 31, 20)
post_prob
```

```
## [1] 0.0668072
```

d. Lambda is much more likely to be greater than 1 (that is, the null hypothesis is likely to be true).

Exercise 8.23

a. Building the MCMC model

```
# STEP 1: DEFINE the model
bb_model <- "
  data {
    int<lower = 0, upper = 6> Y[18];
  }
  parameters {
    real<lower = 0, upper = 6> lambda;
  }
  model {
    Y ~ poisson(lambda);
    lambda ~ gamma(4, 2);
  }
"
```

```
loons$count_per_100
```

```
##  [1] 0 5 0 0 0 2 0 2 1 1 0 4 2 0 3 3 1 3
```

```r
# STEP 2: SIMULATE the posterior
bb_sim <- stan(model_code = bb_model, data = list(Y = loons$count_per_100), chains = 4, iter = 5000*2, s
```

```
## Trying to compile a simple C file

## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## clang -mmacosx-version-min=10.13 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/inclu
## In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/inclu
## /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/src/Core/util/
## namespace Eigen {
## ^
## /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/src/Core/util/
## namespace Eigen {
##                ^
##                ;
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/inclu
## /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/Core:96:10: fa
## #include <complex>
##          ^~~~~~~~~
## 3 errors generated.
## make: *** [foo.o] Error 1
##
## SAMPLING FOR MODEL '366c4a429c9c04fe3b25700dd5ce6a57' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 3.1e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.31 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: 0.074166 seconds (Warm-up)
## Chain 1:                0.08345 seconds (Sampling)
## Chain 1:                0.157616 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '366c4a429c9c04fe3b25700dd5ce6a57' NOW (CHAIN 2).
## Chain 2:
```

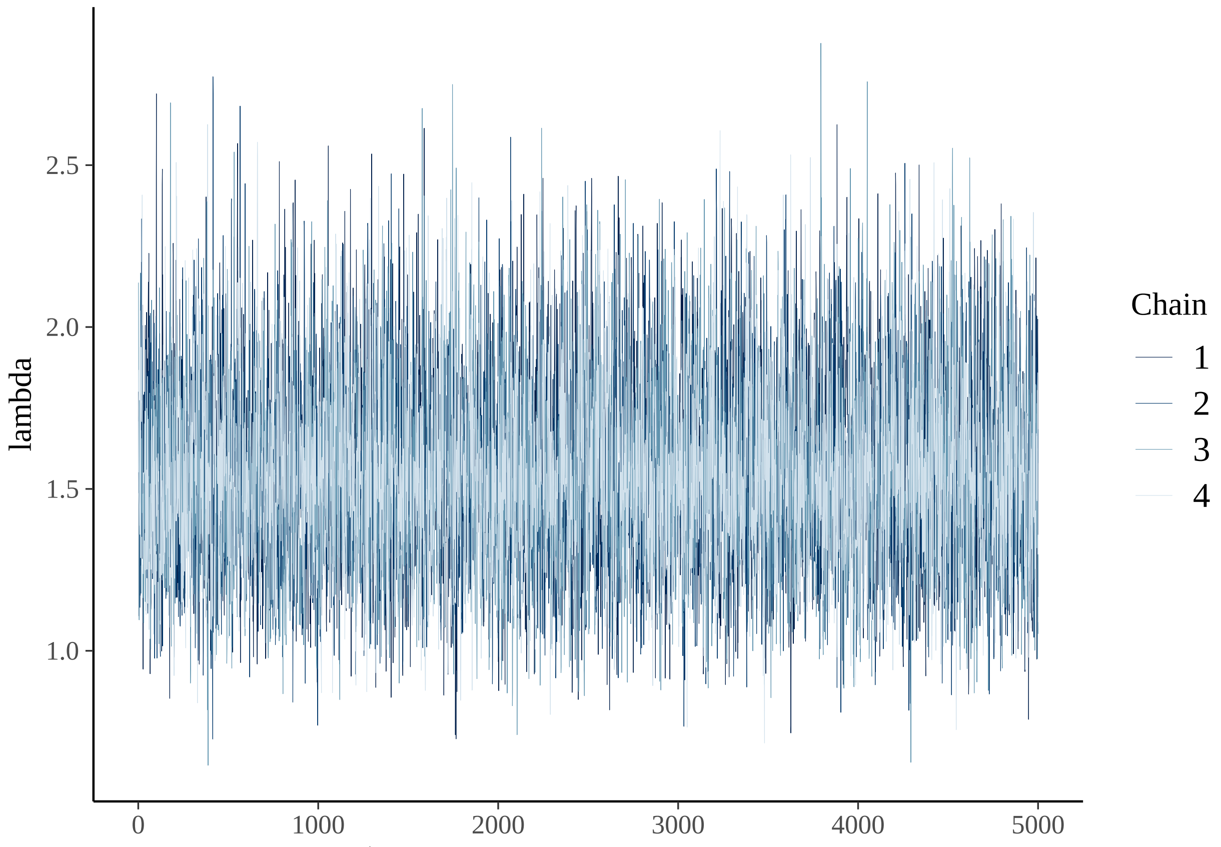
```
## Chain 2: Gradient evaluation took 8e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 10000 [  0%]  (Warmup)
## Chain 2: Iteration: 1000 / 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: 0.080447 seconds (Warm-up)
## Chain 2:                0.084087 seconds (Sampling)
## Chain 2:                0.164534 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '366c4a429c9c04fe3b25700dd5ce6a57' 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: 0.072176 seconds (Warm-up)
## Chain 3:                0.074506 seconds (Sampling)
## Chain 3:                0.146682 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '366c4a429c9c04fe3b25700dd5ce6a57' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 8e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.08 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: 0.075049 seconds (Warm-up)
## Chain 4:                0.07762 seconds (Sampling)
## Chain 4:                0.152669 seconds (Total)
## Chain 4:
```

b. Performing some MCMC diagnostics:

First, Illustrate the traces

```
mcmc_trace(bb_sim, pars = "lambda", size = 0.1)
```



That looks like it's mixing well!

Second, calculate the Neff ratio

```
neff_ratio(bb_sim, pars = c("lambda"))
```

```
## [1] 0.3577482
```

Since Neff>0.1, we consider this ratio adequate.

c. Calculating confidence interval: First, we'll get the interval from the MCMC approximation

```
bb_chains_df <- as.data.frame(bb_sim, pars = "lp__", include = FALSE)
# Calculate posterior summaries of pi
bb_chains_df %>%
  summarize(post_mean = mean(lambda),
            post_median = median(lambda),
            post_mode = sample_mode(lambda),
            lower_95 = quantile(lambda, 0.025),
            upper_95 = quantile(lambda, 0.975))
```

```
##   post_mean post_median post_mode lower_95 upper_95
## 1  1.552693    1.537711  1.491539 1.050863 2.158493
```

Now we'll find the same information using the tidy shortcut

```
library(broom.mixed)
tidy(bb_sim, conf.int = TRUE, conf.level = 0.95)
```

```
## # A tibble: 1 x 5
##   term   estimate std.error conf.low conf.high
##   <chr>     <dbl>     <dbl>    <dbl>     <dbl>
## 1 lambda     1.54     0.283     1.05      2.16
```

d. Approximating probability that lambda <1.

```
bb_chains_df %>%
  mutate(lesser = lambda < 1) %>%
  tabyl(lesser)
```

```
##  lesser     n percent
##   FALSE 19705 0.98525
##    TRUE   295 0.01475
```

e. The confidence interval is nearly exactly the same as that calculated in 8.21. The probability that lambda<1 is a bit lower in the approximation (0.014) than it was based on the posterior model estimate (0.06).

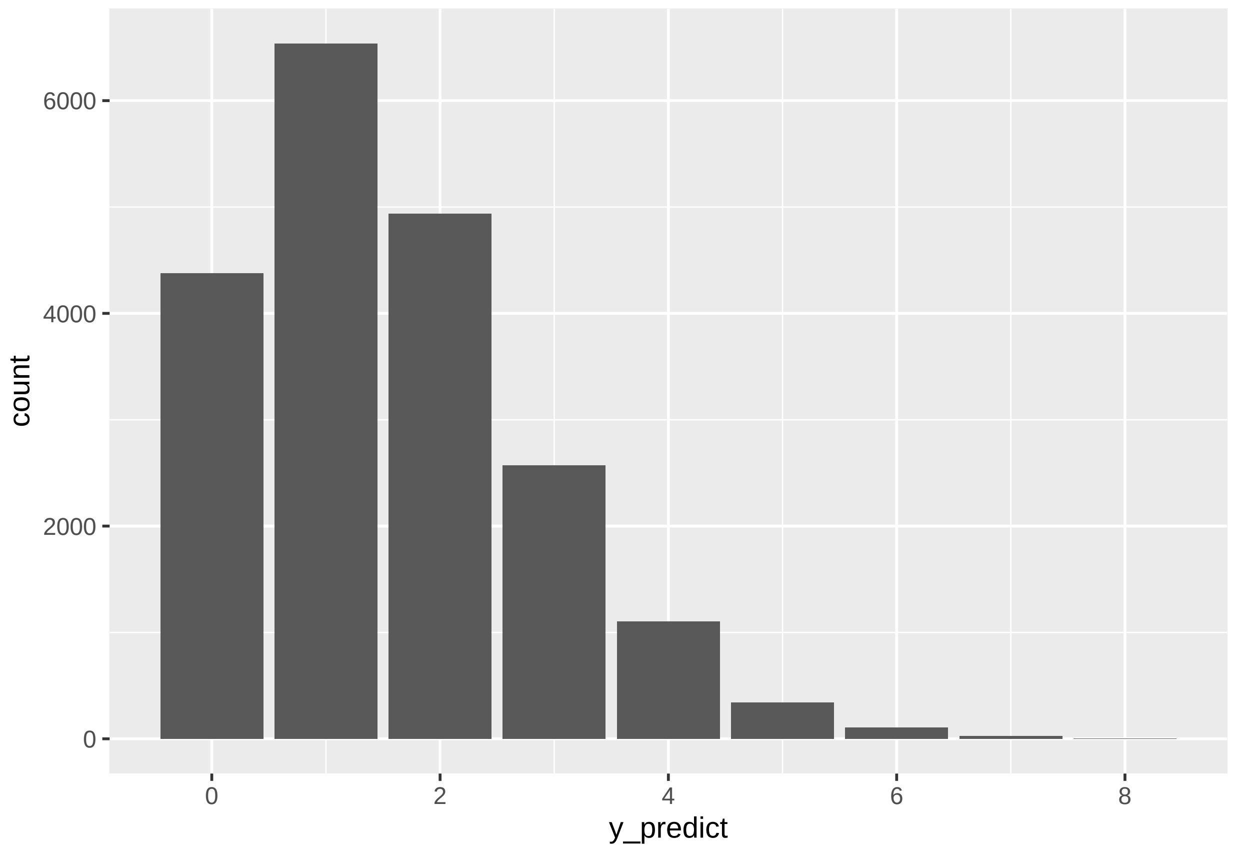
Exercise 8.24

a. Approximating the posterior predictive model:

```
set.seed(1)

# Predict a value of Y' for each lambda value in the chain
bb_chains_df <- bb_chains_df %>%
  mutate(y_predict = rpois(length(lambda), lambda))

# Check it out
bb_chains_df %>%
  head(100)
```

```
##        lambda y_predict
## 1   1.7085225         1
## 2   1.8000430         1
## 3   1.7557179         2
## 4   1.1076034         3
## 5   1.2293046         0
## 6   1.3511065         3
## 7   1.5693702         4
## 8   1.3368891         2
## 9   1.3350028         2
## 10  1.1711785         0
## 11  1.4800991         0
## 12  1.2770505         0
## 13  1.2681152         2
## 14  1.2637570         1
## 15  1.3926948         2
## 16  1.5906021         1
## 17  1.5720429         2
## 18  2.2005011         6
## 19  2.1036705         2
## 20  2.2727713         3
## 21  2.1834045         5
## 22  1.5811967         1
## 23  2.0686317         2
## 24  1.2846003         0
## 25  1.3712218         1
## 26  1.6210638         1
## 27  0.9431618         0
## 28  1.7417158         1
## 29  1.2093821         2
## 30  1.7907328         1
## 31  1.5384741         1
## 32  1.5755757         2
## 33  1.3835703         1
## 34  1.6119198         0
## 35  1.5378608         3
## 36  1.3643388         2
## 37  1.6348545         3
## 38  1.8254118         0
## 39  2.0051445         3
## 40  1.4765415         1
## 41  1.5407557         3
## 42  1.7726460         2
## 43  1.8236721         3
## 44  1.7667666         2
## 45  1.3942651         1
## 46  2.0455114         3
## 47  1.7363586         0
## 48  1.7363586         1
## 49  1.7176974         2
## 50  1.7091472         2
## 51  1.9023430         2
## 52  1.4995292         3
## 53  1.4995292         1
## 54  1.8329414         1
## 55  2.0496249         0
## 56  2.1388821         0
## 57  2.1280428         1
## 58  1.9681694         2
## 59  2.2276490         3
## 60  1.5832657         1
## 61  1.9152751         4
## 62  1.9353188         1
## 63  1.8516701         2
## 64  1.5829168         1
## 65  1.6608179         2
## 66  0.9290193         0
## 67  1.1601022         1
## 68  1.3266300         2
## 69  1.2801160         0
## 70  1.1280533         2
## 71  1.1494699         1
## 72  1.7288066         3
## 73  1.6348025         1
## 74  1.4707562         1
## 75  1.6575471         1
## 76  1.3368492         3
## 77  1.3761772         3
## 78  1.6399526         1
## 79  1.7212783         3
## 80  1.5706064         4
## 81  1.3997224         1
## 82  1.5709170         2
## 83  1.7085042         1
## 84  1.7714098         1
## 85  1.7944224         3
## 86  2.0528255         1
## 87  1.6528697         2
## 88  1.5585575         0
## 89  1.1309860         0
## 90  1.2642528         0
## 91  1.7440706         1
## 92  1.7437097         0
## 93  1.7437097         2
## 94  1.7891466         3
## 95  1.6214916         3
## 96  1.2783755         2
## 97  1.2783755         1
## 98  1.3442482         1
## 99  1.2987779         2
## 100 2.4666287         3
```

```
# Plot the 20,000 predictions
ggplot(bb_chains_df, aes(x = y_predict)) +
  stat_count()
```



b. The number of birds a birdwatcher will most likely observe in their next watching period is 1.

c. Finding probability that 0 birds are seen in next period: NOTE: not sure on methodology here

```
denom <- bb_chains_df  %>%
  summarize(total=length(lambda))

nom <- bb_chains_df %>%
  filter(y_predict==0) %>%
  summarize(total=length(lambda))

nom/denom
```

```
##     total
## 1 0.21895
```