Exercises from Chpater 6 - Bayes Rules

Braulio Güémez

12/10/2021

```
# Load packages
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(janitor)
```

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

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

```
library(rstan)
```

```
## Loading required package: StanHeaders

## 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)

## Do not specify '-march=native' in 'LOCAL_CPPFLAGS' or a Makevars file

##
## Attaching package: 'rstan'
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
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(bayesrules)
```

6.1

a.

Step 1: Define a grid with numbers that correspond to the desired parameter to estimate . The more numbers (pi, lambda, mu), the more defined our grid will be.

Step 2: Estimate the prior and the likelihood values at each of the values.

Step 3: Approximate the posterior by multiplying the likelihood values and the prior values at each of values from the grid.

Step 4: Sample values from the approximated normal posterior probabilities

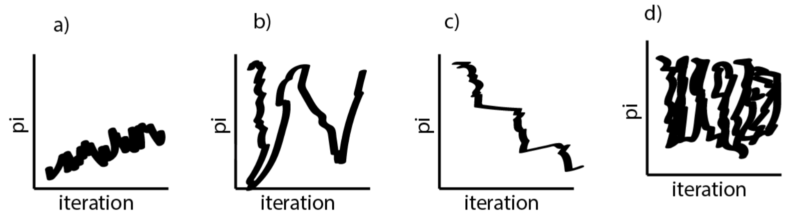
b. In step 1, I would include a larger vector of numbers to have as many bins as possible to have a closer approximation to the posterior.

In step 4, I could also take a larger sample to have a robust estimation of the posterior probabilities.

In sum, the more information we plug in the model, the most robust posterior approximation

6.2

```
knitr::include_graphics("C:/Users/bguemez/OneDrive - Duke University/1. First Year/3. Stats/R_exercises
```



6.3

a. The chain is mixing too slowly.

The posterior approximation may not capture the total plausible values of the real posterior.

b. The chain has high correlation.

This implies that the sample is not behaving as constituted by independent observations which would undermine the validity of the estimated posterior.

c. The chain has a tendency to get "stuck."

The approximated posterior will over sample the values where the chain got stuck.

6.4

a. MCMC diagnostics are important because we don't have the real posterior model to compare it with the simulation! The only thing we have to estimate a decent approximate posterior model are those techniques. There is not another way.
b. Because we can't calculate a posterior model by hand when we have multiple parameters that come from different data sources.
c. Rstan permits us to work in R but using Stan notation. Stan is particularly useful because it is flexible in the many assumptions of the models. In the chapter, we learn that we can plug in the data, the parameter(s) and the type of model involved in the simulation of the posterior. All in one step! That is very convenient.
d. I still don't understand the general logic of how multiple parameters can be plugged in the posterior simulation.

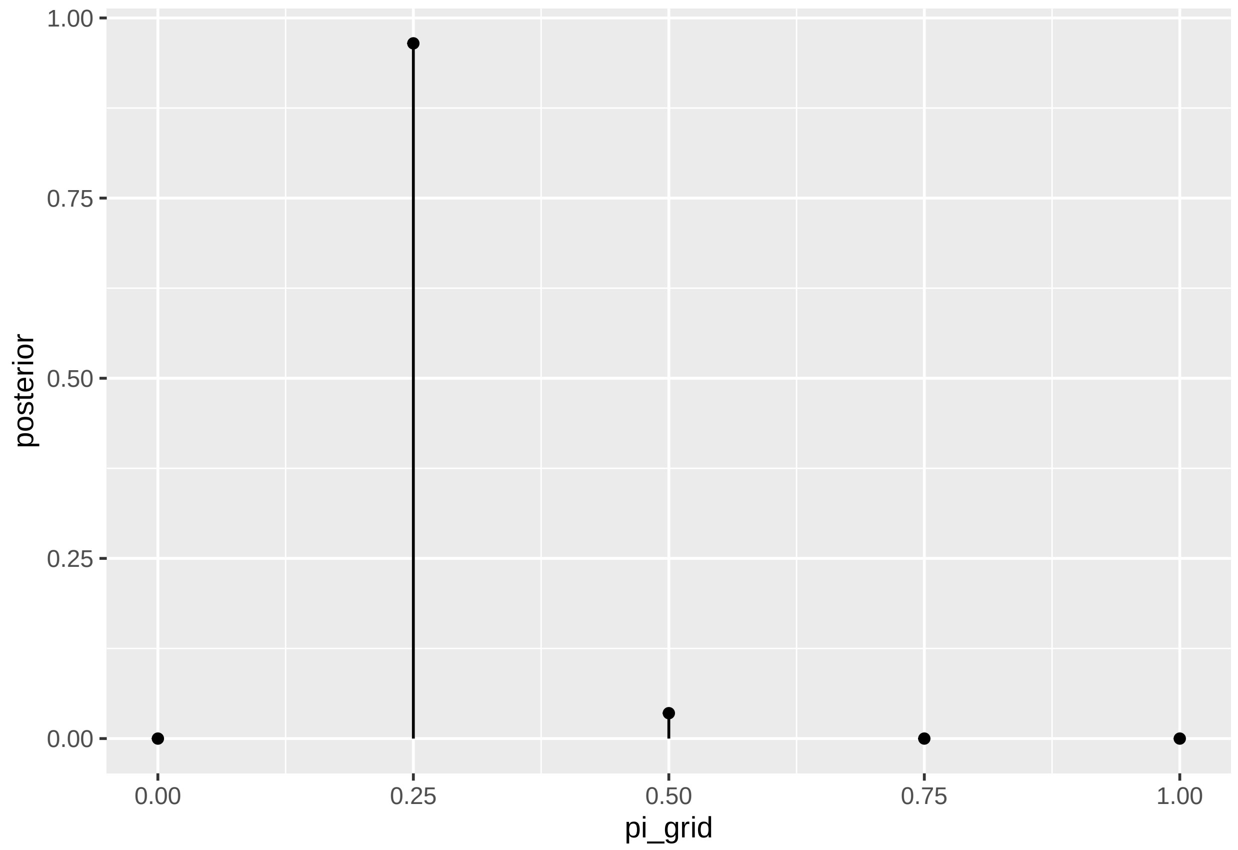
6.5

```r
# Load packages
library(tidyverse)
library(janitor)
library(rstan)
library(bayesplot)
```

a.

```r
# Step 1: Define a grid
grid_data  <- data.frame(pi_grid = c(0,.25,.5,.75,1))

# Step 2: Evaluate the prior & likelihood at each pi
grid_data <- grid_data %>%
  mutate(prior = dbeta(pi_grid, 3, 8),
         likelihood = dbinom(2, 10, pi_grid))
```

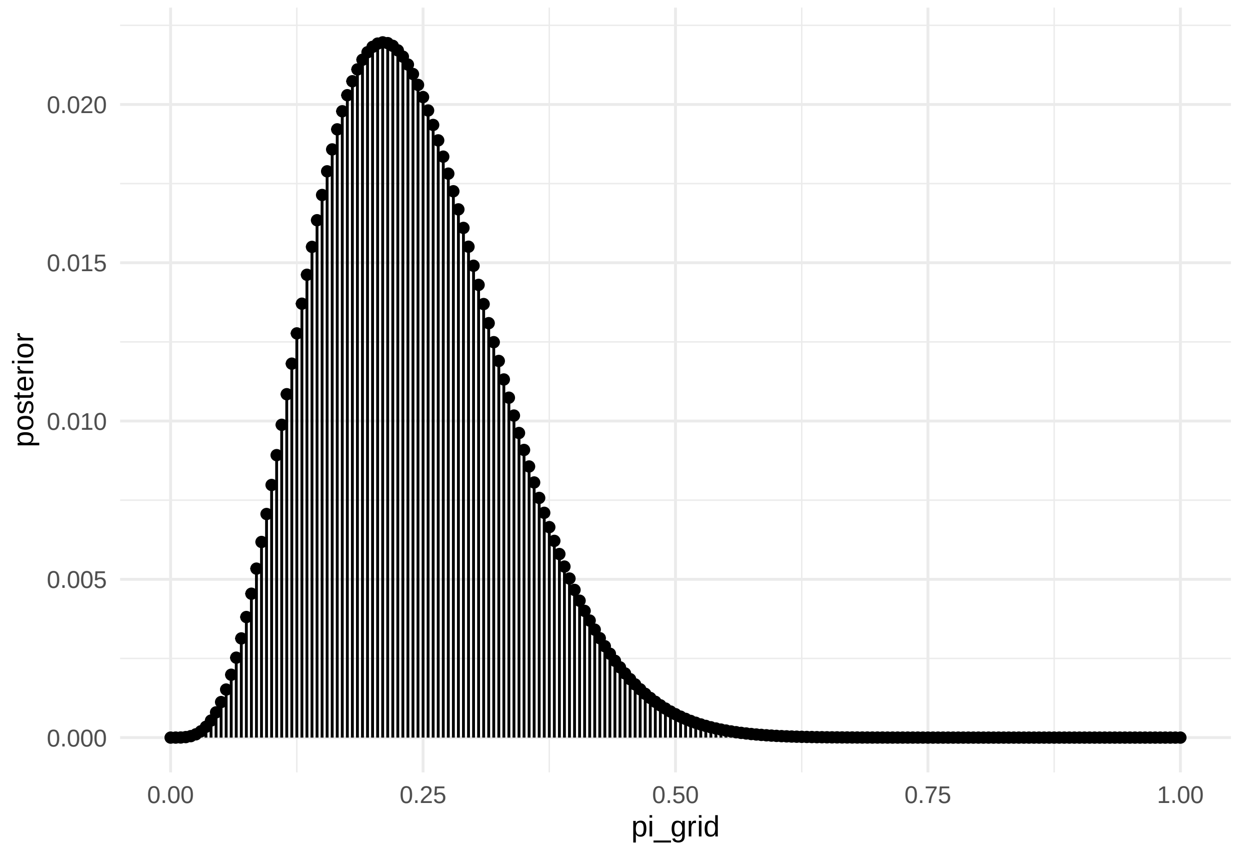
```
# Step 3: Approximate the posterior
grid_data <- grid_data %>%
  mutate(unnormalized = likelihood * prior,
         posterior = unnormalized / sum(unnormalized))

grid_data |> ggplot(aes(pi_grid,posterior)) + geom_point() + geom_segment(aes(x = pi_grid, xend = pi_gri
```



The values are *very* concentrated in pi=.25, we need a larger grid!

b.

```
# Step 1: Define a grid
grid_data  <- data.frame(pi_grid = seq(0,1,length=201))

# Step 2: Evaluate the prior & likelihood at each pi
grid_data <- grid_data %>%
  mutate(prior = dbeta(pi_grid, 3, 8),
         likelihood = dbinom(2, 10, pi_grid))

# Step 3: Approximate the posterior
grid_data <- grid_data %>%
  mutate(unnormalized = likelihood * prior,
         posterior = unnormalized / sum(unnormalized))

grid_data |> ggplot(aes(pi_grid,posterior)) + geom_point() + geom_segment(aes(x = pi_grid, xend = pi_gri
```



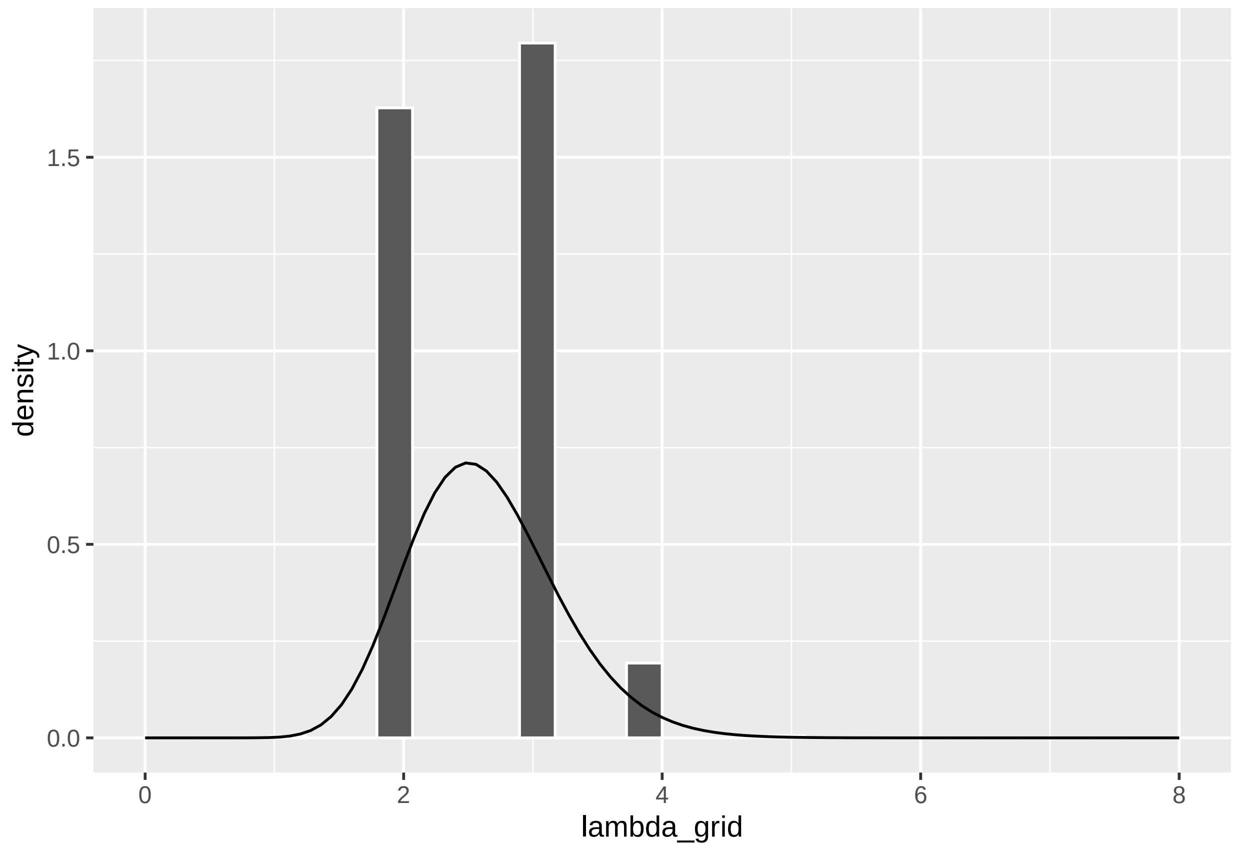
6.6

```
# Step 1: Define a grid of lambda values
grid_data   <- data.frame(
  lambda_grid = seq(from = 0, to = 8))

# Step 2: Evaluate the prior & likelihood at each lambda
grid_data <- grid_data %>%
  mutate(prior = dgamma(lambda_grid, 20,5),
      likelihood = dpois(0, lambda_grid) * dpois(1, lambda_grid)*dpois(0,lambda_grid))

# Step 3: Approximate the posterior
grid_data <- grid_data %>%
  mutate(unnormalized = likelihood*prior,
      posterior = unnormalized/sum(unnormalized))

# Set the seed
set.seed(84735)

# Step 4: sample from the discretized posterior
post_sample <- sample_n(grid_data, size = 10000,
                        weight = posterior, replace = TRUE)

# Histogram of the grid simulation with posterior pdf
```

```
ggplot(post_sample, aes(x = lambda_grid)) +
  geom_histogram(aes(y = ..density..), color = "white") +
  stat_function(fun = dgamma, args = list(20+1, 5+3)) +
  lims(x = c(0, 8))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```

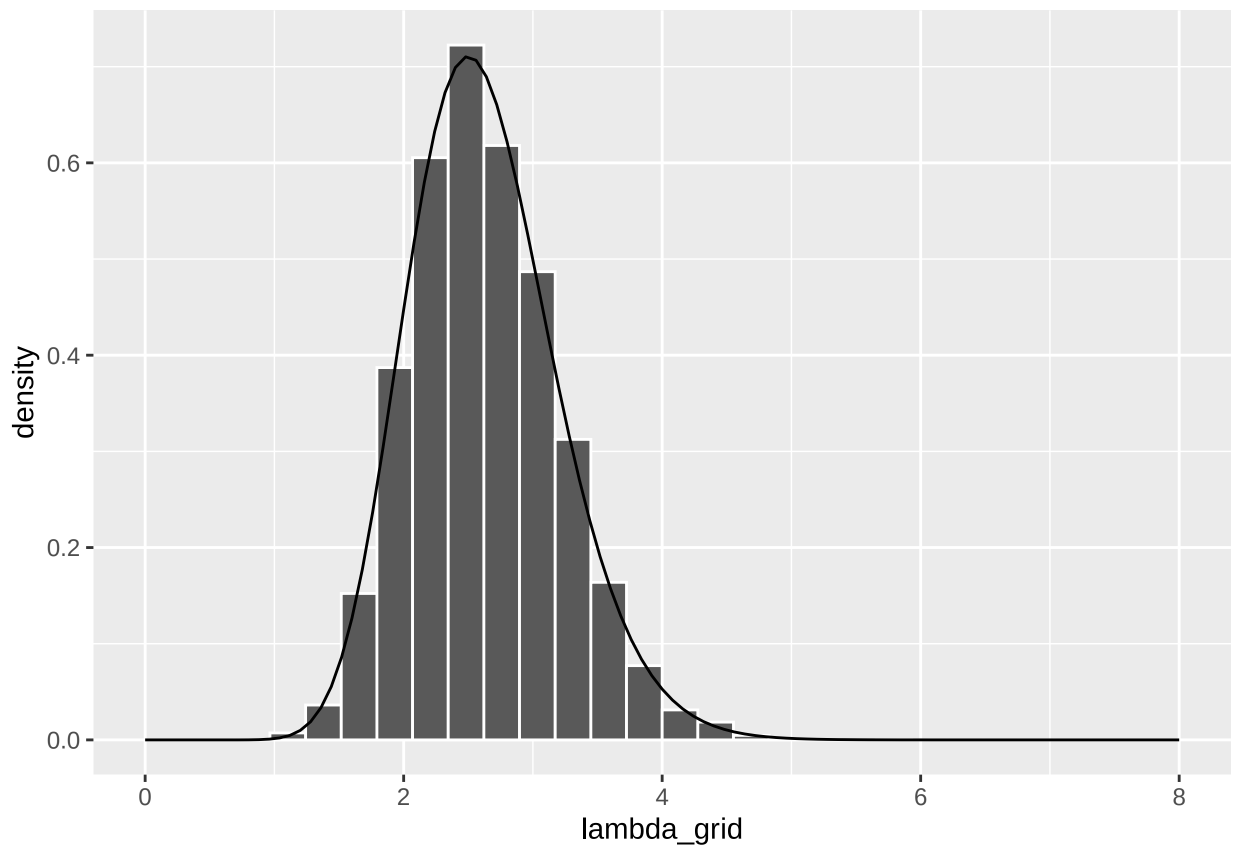


```
# Step 1: Define a grid of lambda values
grid_data   <- data.frame(
  lambda_grid = seq(from = 0, to = 8, length=201))

# Step 2: Evaluate the prior & likelihood at each lambda
grid_data <- grid_data %>%
  mutate(prior = dgamma(lambda_grid, 20,5),
      likelihood = dpois(0, lambda_grid) * dpois(1, lambda_grid)*dpois(0,lambda_grid))

# Step 3: Approximate the posterior
grid_data <- grid_data %>%
  mutate(unnormalized = likelihood*prior,
      posterior = unnormalized/sum(unnormalized))

# Set the seed
```

```r
set.seed(84735)

# Step 4: sample from the discretized posterior
post_sample <- sample_n(grid_data, size = 10000,
                        weight = posterior, replace = TRUE)

# Histogram of the grid simulation with posterior pdf
ggplot(post_sample, aes(x = lambda_grid)) +
  geom_histogram(aes(y = ..density..), color = "white") +
  stat_function(fun = dgamma, args = list(20+1, 5+3)) +
  lims(x = c(0, 8))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```

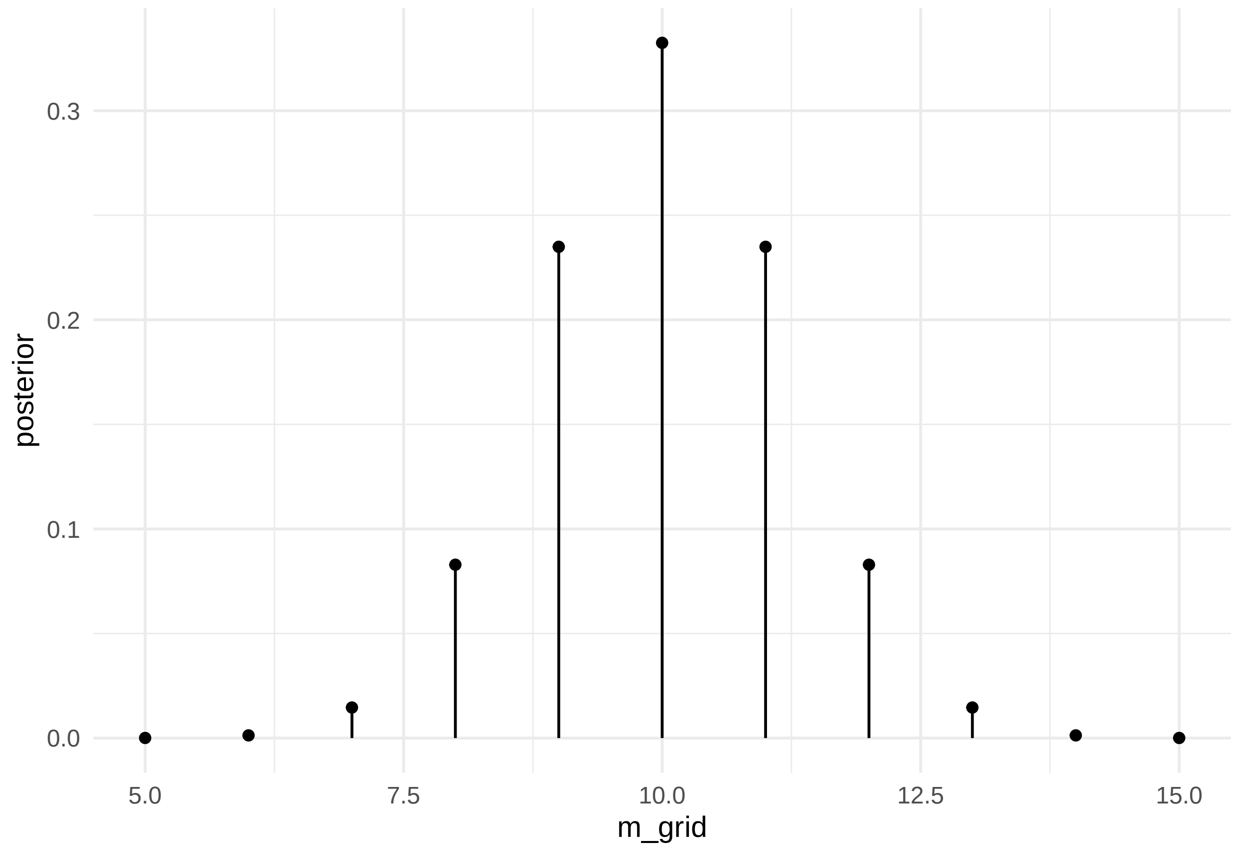


6.7

a.

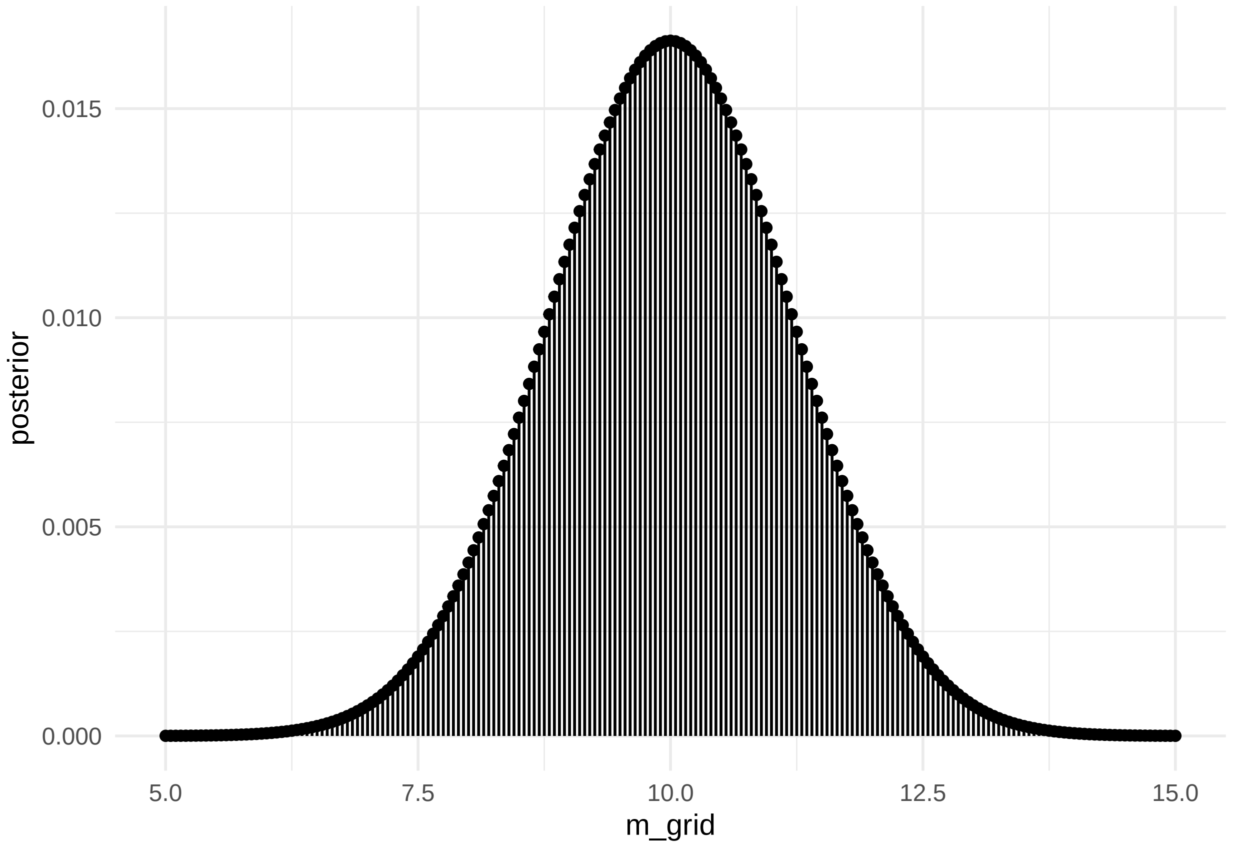
```r
dta <- c(7.1,8.9,8.4,8.6)

# Step 1: Define a grid
```

```
grid_data  <- data.frame(m_grid = c(5:15))

# Step 2: Evaluate the prior & likelihood at each pi
grid_data <- grid_data %>%
  mutate(prior = dnorm(m_grid, 10, 1.2),
         likelihood = exp(-((mean(dta)-10)^2/(2*(1.3^2/4)))))

# Step 3: Approximate the posterior
grid_data <- grid_data %>%
  mutate(unnormalized = likelihood * prior,
         posterior = unnormalized / sum(unnormalized))


grid_data |> ggplot(aes(m_grid,posterior)) + geom_point() + geom_segment(aes(x = m_grid, xend = m_grid,
```



b.

```
dta <- c(7.1,8.9,8.4,8.6)

# Step 1: Define a grid
grid_data  <- data.frame(m_grid=seq(5,15,length=201))

# Step 2: Evaluate the prior & likelihood at each pi
grid_data <- grid_data %>%
  mutate(prior = dnorm(m_grid, 10, 1.2),
        likelihood = exp(-((mean(dta)-10)^2/(2*(1.3^2/4)))))

# Step 3: Approximate the posterior
grid_data <- grid_data %>%
  mutate(unnormalized = likelihood * prior,
        posterior = unnormalized / sum(unnormalized))

grid_data |> ggplot(aes(m_grid,posterior)) + geom_point() + geom_segment(aes(x = m_grid, xend = m_grid,
```



6.8

a.You might want to estimate consumer preference for certain product in different places where data about it has been collected. In this case, there are multiple parameters theta that can't be incorporated in a single grid approximation.

b. The curse of dimensionality means that the grid approximation with multiple parameters becomes computationally expensive. That is why MCMC approaches are a flexible alternative.

6.9.

a.

Both samples are not taken directly from the posterior pdf. They are approximations.

b

Both are useful to approximate posteriors using one parameter.

c.

The obtained samples are independent and they are easier to understand than MCMC.

d.

You can calculate posteriors that involve multiple parameters and it is not that computationally expensive.

6.10

a. It is a Markov Chain because the decision of whether you go to a Thai restaurant depends on what restaurant you went to the day before.
b. The events are independent. This is not a Markov Chain.
c. This may be a Markov Chain because, depending on their level of expertise, you can "learn" how your roommate plays and increase the probability of winning.

6.11

a.

```
# STEP 1: DEFINE the model
bb_model <- "
  data {
    int<lower = 0, upper = 20> Y;
  }
  parameters {
    real<lower = 0, upper = 1> pi;
  }
  model {
    Y ~ binomial(20, pi);
    pi ~ beta(1, 1);
  }
"
```

b. I'll assume the vector of counts is 1 because it was not provided in the exercise

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

c. I'll assume the vector of y is = 1 because it was not provided in the exercise

```
n_model <- '
data {
    vector[1] Y;
}
parameters {
    real mu;
}
model {
   Y ~ normal(mu, 1);
   mu ~ normal(0, 10);
}
'
```

6.12

a.

```
# STEP 1: DEFINE the model
bb_model <- "
  data {
    int<lower = 0, upper = 20> Y;
  }
  parameters {
    real<lower = 0, upper = 1> pi;
  }
  model {
    Y ~ binomial(20, pi);
    pi ~ beta(1, 1);
  }
"
# STEP 2: SIMULATE the posterior
bb_sim <- stan(model_code = bb_model, data = list(Y = 12),
               chains = 4, iter = 5000*2, seed = 84735)
```

```
##
## SAMPLING FOR MODEL '045e2984ac830d5f013700d9d865d64d' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.066 seconds (Warm-up)
## Chain 1:                0.075 seconds (Sampling)
## Chain 1:                0.141 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '045e2984ac830d5f013700d9d865d64d' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.066 seconds (Warm-up)
## Chain 2:                0.068 seconds (Sampling)
## Chain 2:                0.134 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '045e2984ac830d5f013700d9d865d64d' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.073 seconds (Warm-up)
## Chain 3:                0.07 seconds (Sampling)
## Chain 3:                0.143 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '045e2984ac830d5f013700d9d865d64d' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.071 seconds (Warm-up)
## Chain 4:                0.085 seconds (Sampling)
## Chain 4:                0.156 seconds (Total)
## Chain 4:
```

b.

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

```
gp_sim <- stan(model_code = gp_model, data = list(Y = 3),
               chains = 4, iter = 5000*2, seed = 84735)
```

```
## Error in mod$fit_ptr() :
##   Exception: mismatch in number dimensions declared and found in context; processing stage=data init

## failed to create the sampler; sampling not done
```

c.

```
n_model <- '
data {
    vector[1] Y;
}
parameters {
    real mu;
}
model {
   Y ~ normal(mu, 1);
   mu ~ normal(0, 10);
}
'


gn_sim <- stan(model_code = n_model, data = list(Y=12.2),
               chains = 4, iter = 5000*2, seed = 84735)
```

```
## Error in mod$fit_ptr() :
##   Exception: mismatch in number dimensions declared and found in context; processing stage=data init

## failed to create the sampler; sampling not done
```

6.13

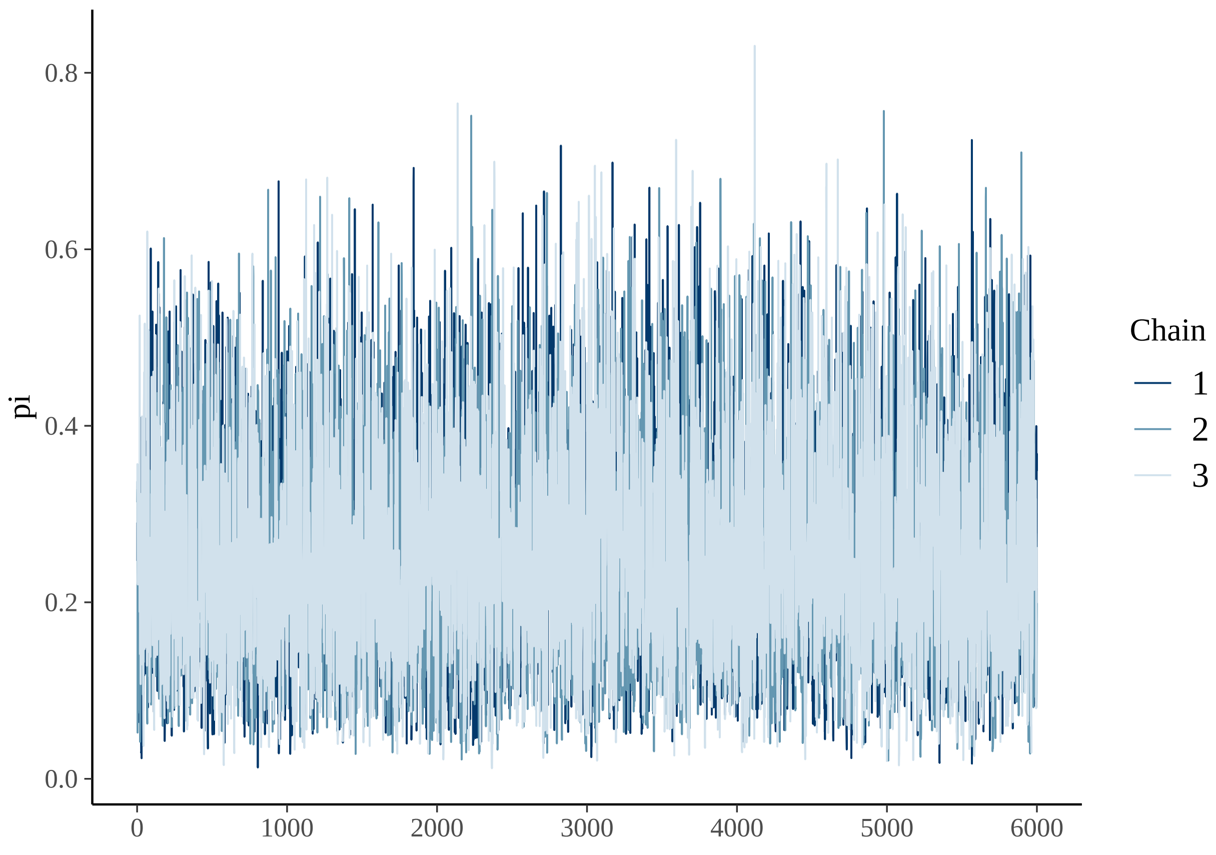
a.

```
# STEP 1: DEFINE the model
bb_model <- "
  data {
    int<lower = 0, upper = 10> Y;
  }
  parameters {
    real<lower = 0, upper = 1> pi;
  }
  model {
    Y ~ binomial(10, pi);
    pi ~ beta(1, 1);
  }
"
# STEP 2: SIMULATE the posterior
bb_sim <- stan(model_code = bb_model, data = list(Y = 2),
               chains = 3, iter = 12000, seed = 84735)
```

```
##
## SAMPLING FOR MODEL '0c314148b2a0b37d14d5efbcbcef4b33' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:     1 / 12000 [  0%]  (Warmup)
## Chain 1: Iteration:  1200 / 12000 [ 10%]  (Warmup)
## Chain 1: Iteration:  2400 / 12000 [ 20%]  (Warmup)
## Chain 1: Iteration:  3600 / 12000 [ 30%]  (Warmup)
## Chain 1: Iteration:  4800 / 12000 [ 40%]  (Warmup)
## Chain 1: Iteration:  6000 / 12000 [ 50%]  (Warmup)
## Chain 1: Iteration:  6001 / 12000 [ 50%]  (Sampling)
## Chain 1: Iteration:  7200 / 12000 [ 60%]  (Sampling)
## Chain 1: Iteration:  8400 / 12000 [ 70%]  (Sampling)
## Chain 1: Iteration:  9600 / 12000 [ 80%]  (Sampling)
## Chain 1: Iteration: 10800 / 12000 [ 90%]  (Sampling)
## Chain 1: Iteration: 12000 / 12000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.091 seconds (Warm-up)
## Chain 1:                0.085 seconds (Sampling)
## Chain 1:                0.176 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '0c314148b2a0b37d14d5efbcbcef4b33' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:     1 / 12000 [  0%]  (Warmup)
## Chain 2: Iteration:  1200 / 12000 [ 10%]  (Warmup)
## Chain 2: Iteration:  2400 / 12000 [ 20%]  (Warmup)
## Chain 2: Iteration:  3600 / 12000 [ 30%]  (Warmup)
## Chain 2: Iteration:  4800 / 12000 [ 40%]  (Warmup)
## Chain 2: Iteration:  6000 / 12000 [ 50%]  (Warmup)
## Chain 2: Iteration:  6001 / 12000 [ 50%]  (Sampling)
## Chain 2: Iteration:  7200 / 12000 [ 60%]  (Sampling)
## Chain 2: Iteration:  8400 / 12000 [ 70%]  (Sampling)
## Chain 2: Iteration:  9600 / 12000 [ 80%]  (Sampling)
## Chain 2: Iteration: 10800 / 12000 [ 90%]  (Sampling)
## Chain 2: Iteration: 12000 / 12000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.083 seconds (Warm-up)
## Chain 2:                0.1 seconds (Sampling)
## Chain 2:                0.183 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '0c314148b2a0b37d14d5efbcbcef4b33' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:     1 / 12000 [  0%]  (Warmup)
## Chain 3: Iteration:  1200 / 12000 [ 10%]  (Warmup)
## Chain 3: Iteration:  2400 / 12000 [ 20%]  (Warmup)
## Chain 3: Iteration:  3600 / 12000 [ 30%]  (Warmup)
## Chain 3: Iteration:  4800 / 12000 [ 40%]  (Warmup)
## Chain 3: Iteration:  6000 / 12000 [ 50%]  (Warmup)
## Chain 3: Iteration:  6001 / 12000 [ 50%]  (Sampling)
## Chain 3: Iteration:  7200 / 12000 [ 60%]  (Sampling)
## Chain 3: Iteration:  8400 / 12000 [ 70%]  (Sampling)
## Chain 3: Iteration:  9600 / 12000 [ 80%]  (Sampling)
## Chain 3: Iteration: 10800 / 12000 [ 90%]  (Sampling)
## Chain 3: Iteration: 12000 / 12000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.088 seconds (Warm-up)
## Chain 3:                0.082 seconds (Sampling)
## Chain 3:                0.17 seconds (Total)
## Chain 3:
```

b.

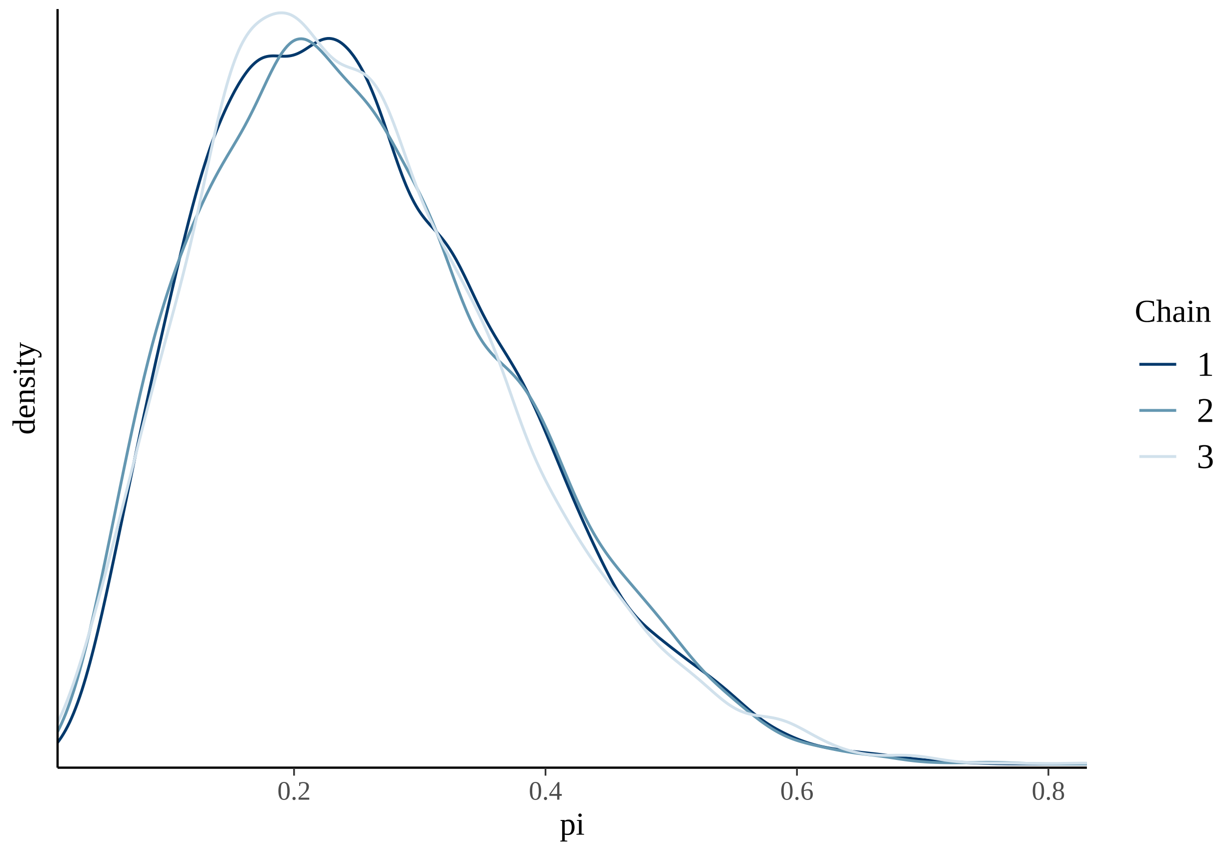
```
mcmc_trace(bb_sim, pars = "pi")
```



c. The range of values is from 0 to 6000 iterations. I set it up for 12,000 but because they remove the first half that count as "burn-in" or "warm-up" samples, only 6000 are shown in the plot.

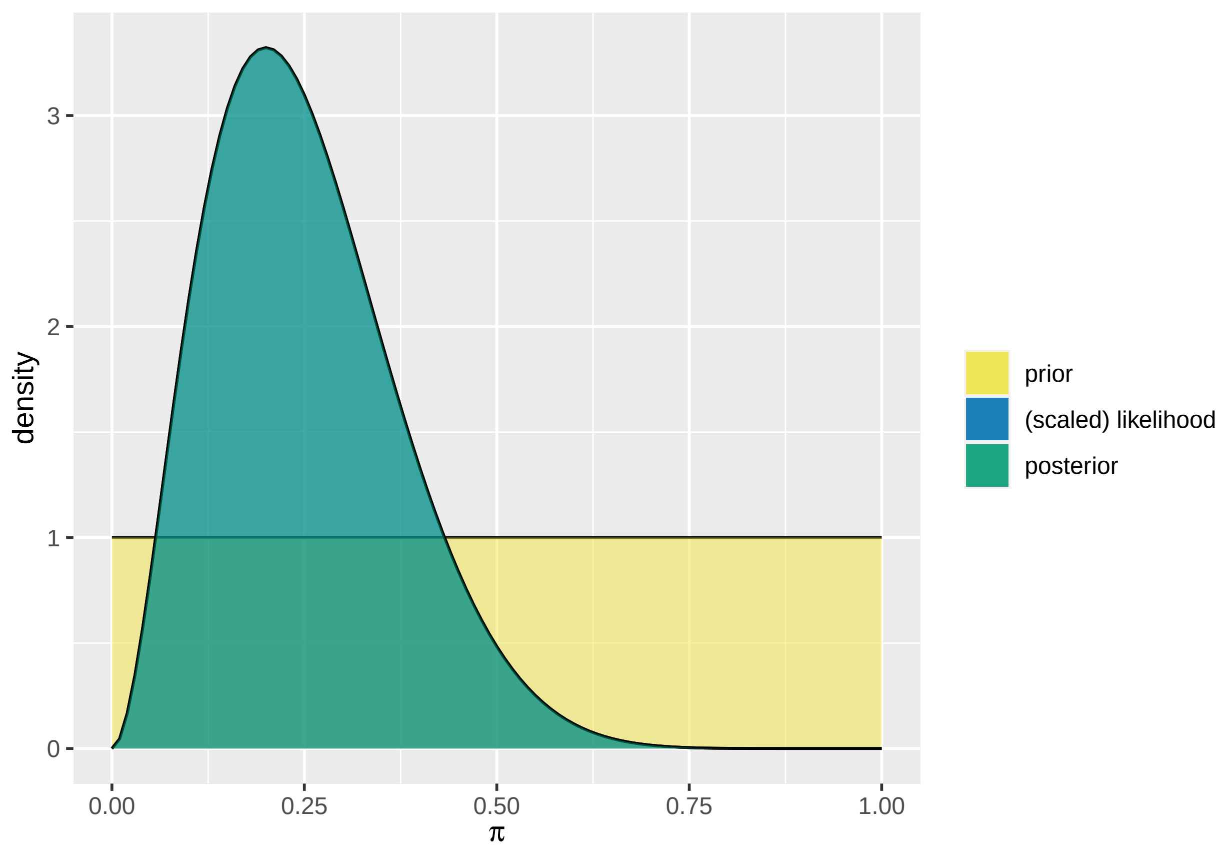
d.

```
mcmc_dens_overlay(bb_sim, pars = "pi") +
  ylab("density")
```



e. Both posteriors look identical!

```
plot_beta_binomial(1,1,2,10)
```



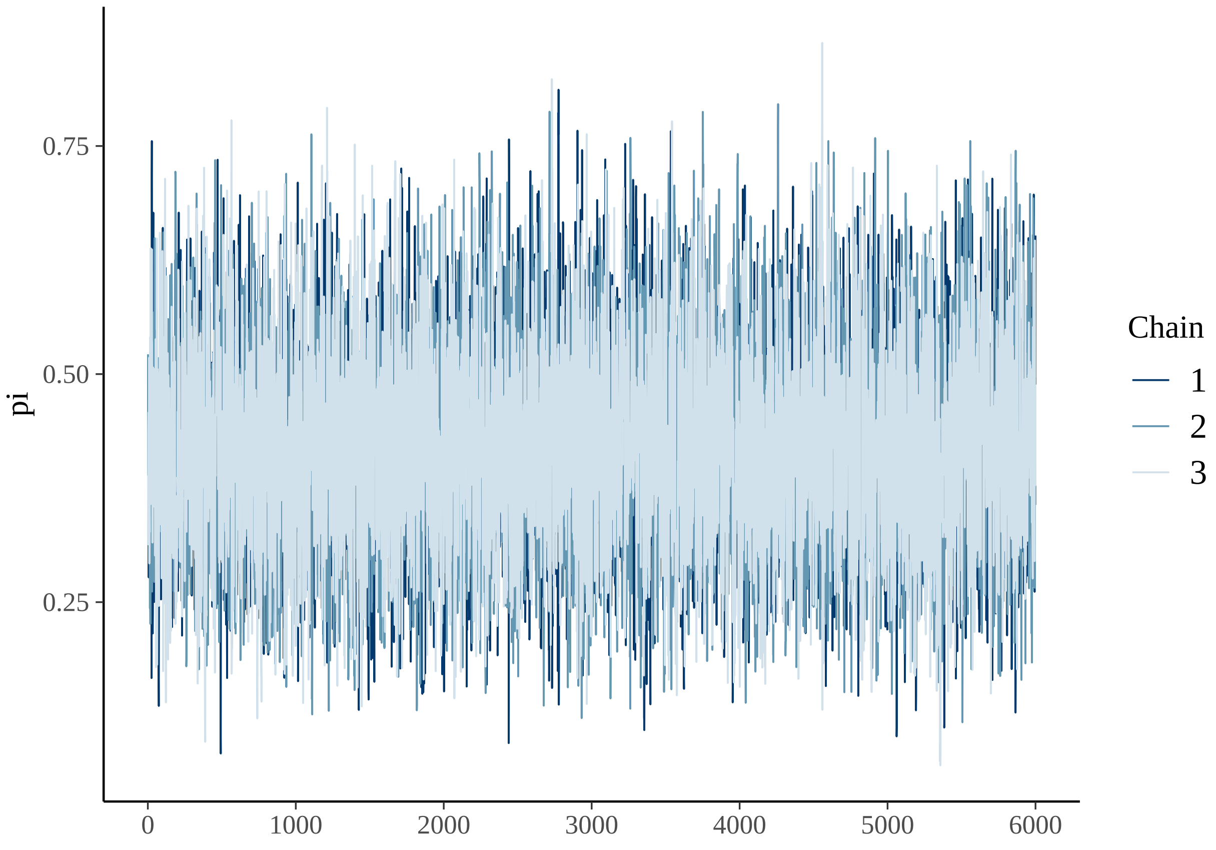
6.14

```
# STEP 1: DEFINE the model
bb_model <- "
  data {
    int<lower = 0, upper = 12> Y;
  }
  parameters {
    real<lower = 0, upper = 1> pi;
  }
  model {
    Y ~ binomial(12, pi);
    pi ~ beta(4, 3);
  }
"
# STEP 2: SIMULATE the posterior
bb_sim <- stan(model_code = bb_model, data = list(Y = 4),
               chains = 3, iter = 12000, seed = 84735)
```

```
##
## SAMPLING FOR MODEL 'b61b4553aa22e133ec03c9d7da030f7b' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:     1 / 12000 [  0%]  (Warmup)
## Chain 1: Iteration:  1200 / 12000 [ 10%]  (Warmup)
## Chain 1: Iteration:  2400 / 12000 [ 20%]  (Warmup)
## Chain 1: Iteration:  3600 / 12000 [ 30%]  (Warmup)
## Chain 1: Iteration:  4800 / 12000 [ 40%]  (Warmup)
## Chain 1: Iteration:  6000 / 12000 [ 50%]  (Warmup)
## Chain 1: Iteration:  6001 / 12000 [ 50%]  (Sampling)
## Chain 1: Iteration:  7200 / 12000 [ 60%]  (Sampling)
## Chain 1: Iteration:  8400 / 12000 [ 70%]  (Sampling)
## Chain 1: Iteration:  9600 / 12000 [ 80%]  (Sampling)
## Chain 1: Iteration: 10800 / 12000 [ 90%]  (Sampling)
## Chain 1: Iteration: 12000 / 12000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.079 seconds (Warm-up)
## Chain 1:                0.087 seconds (Sampling)
## Chain 1:                0.166 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'b61b4553aa22e133ec03c9d7da030f7b' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:     1 / 12000 [  0%]  (Warmup)
## Chain 2: Iteration:  1200 / 12000 [ 10%]  (Warmup)
## Chain 2: Iteration:  2400 / 12000 [ 20%]  (Warmup)
## Chain 2: Iteration:  3600 / 12000 [ 30%]  (Warmup)
## Chain 2: Iteration:  4800 / 12000 [ 40%]  (Warmup)
## Chain 2: Iteration:  6000 / 12000 [ 50%]  (Warmup)
## Chain 2: Iteration:  6001 / 12000 [ 50%]  (Sampling)
## Chain 2: Iteration:  7200 / 12000 [ 60%]  (Sampling)
## Chain 2: Iteration:  8400 / 12000 [ 70%]  (Sampling)
## Chain 2: Iteration:  9600 / 12000 [ 80%]  (Sampling)
## Chain 2: Iteration: 10800 / 12000 [ 90%]  (Sampling)
## Chain 2: Iteration: 12000 / 12000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.081 seconds (Warm-up)
## Chain 2:                0.089 seconds (Sampling)
## Chain 2:                0.17 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'b61b4553aa22e133ec03c9d7da030f7b' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:     1 / 12000 [  0%]  (Warmup)
## Chain 3: Iteration:  1200 / 12000 [ 10%]  (Warmup)
## Chain 3: Iteration:  2400 / 12000 [ 20%]  (Warmup)
## Chain 3: Iteration:  3600 / 12000 [ 30%]  (Warmup)
## Chain 3: Iteration:  4800 / 12000 [ 40%]  (Warmup)
## Chain 3: Iteration:  6000 / 12000 [ 50%]  (Warmup)
## Chain 3: Iteration:  6001 / 12000 [ 50%]  (Sampling)
## Chain 3: Iteration:  7200 / 12000 [ 60%]  (Sampling)
## Chain 3: Iteration:  8400 / 12000 [ 70%]  (Sampling)
## Chain 3: Iteration:  9600 / 12000 [ 80%]  (Sampling)
## Chain 3: Iteration: 10800 / 12000 [ 90%]  (Sampling)
## Chain 3: Iteration: 12000 / 12000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.081 seconds (Warm-up)
## Chain 3:                0.092 seconds (Sampling)
## Chain 3:                0.173 seconds (Total)
## Chain 3:
```

b.

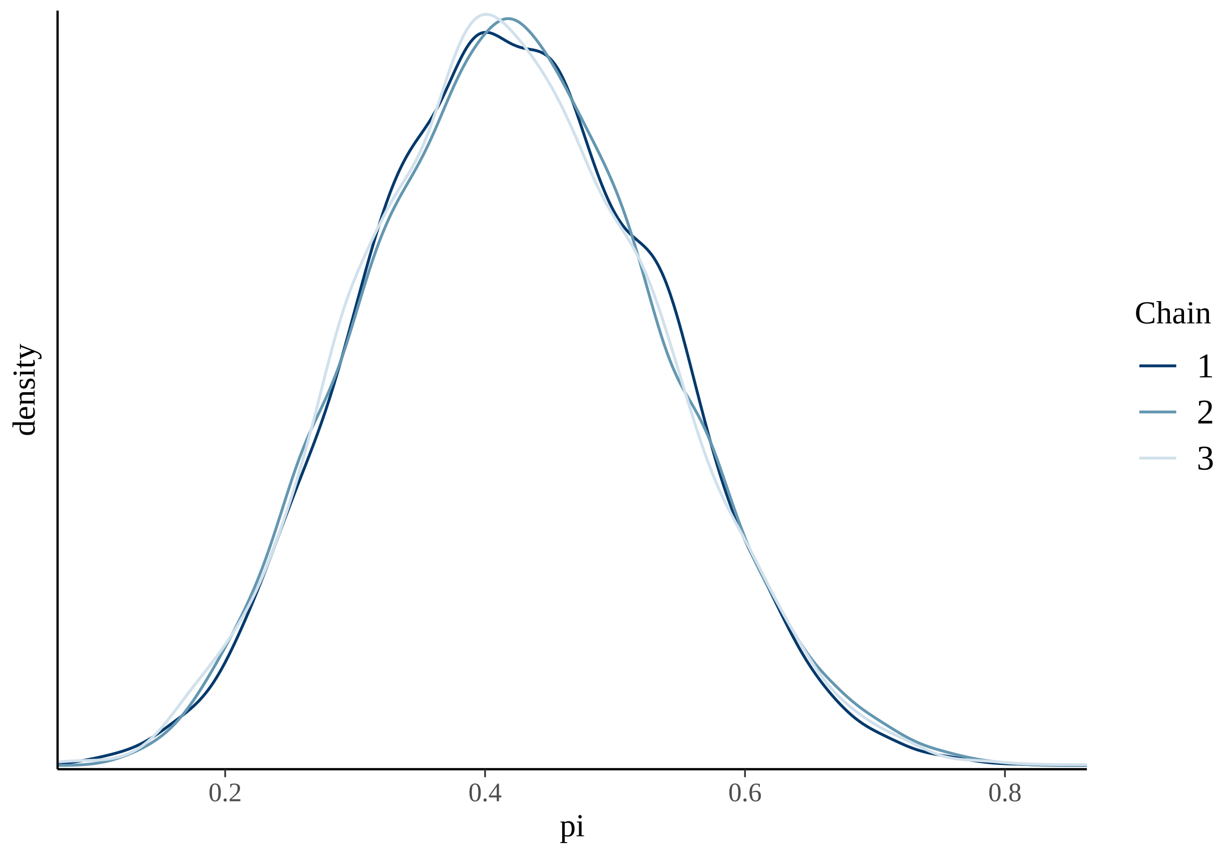
```
mcmc_trace(bb_sim, pars = "pi")
```



c. The range of values is from 0 to 6000 iterations. I set it up for 12,000 but because they remove the first half that count as “burn-in” or “warm-up” samples, only 6000 are shown in the plot.

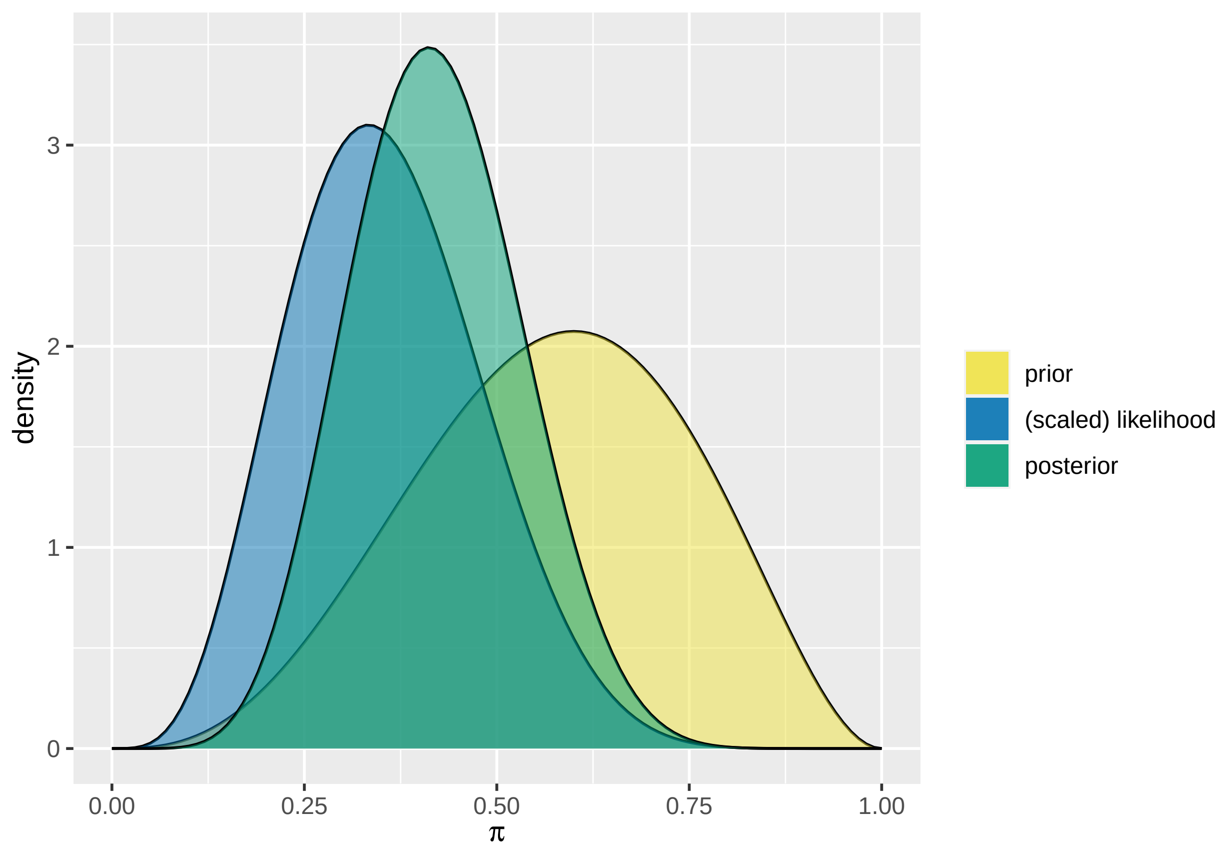
d.

```
mcmc_dens_overlay(bb_sim, pars = "pi") +
  ylab("density")
```



e. The real posterior and the approximate posterior look rather similar.

```
plot_beta_binomial(4,3,4,12)
```



Exercise 6.15

a.

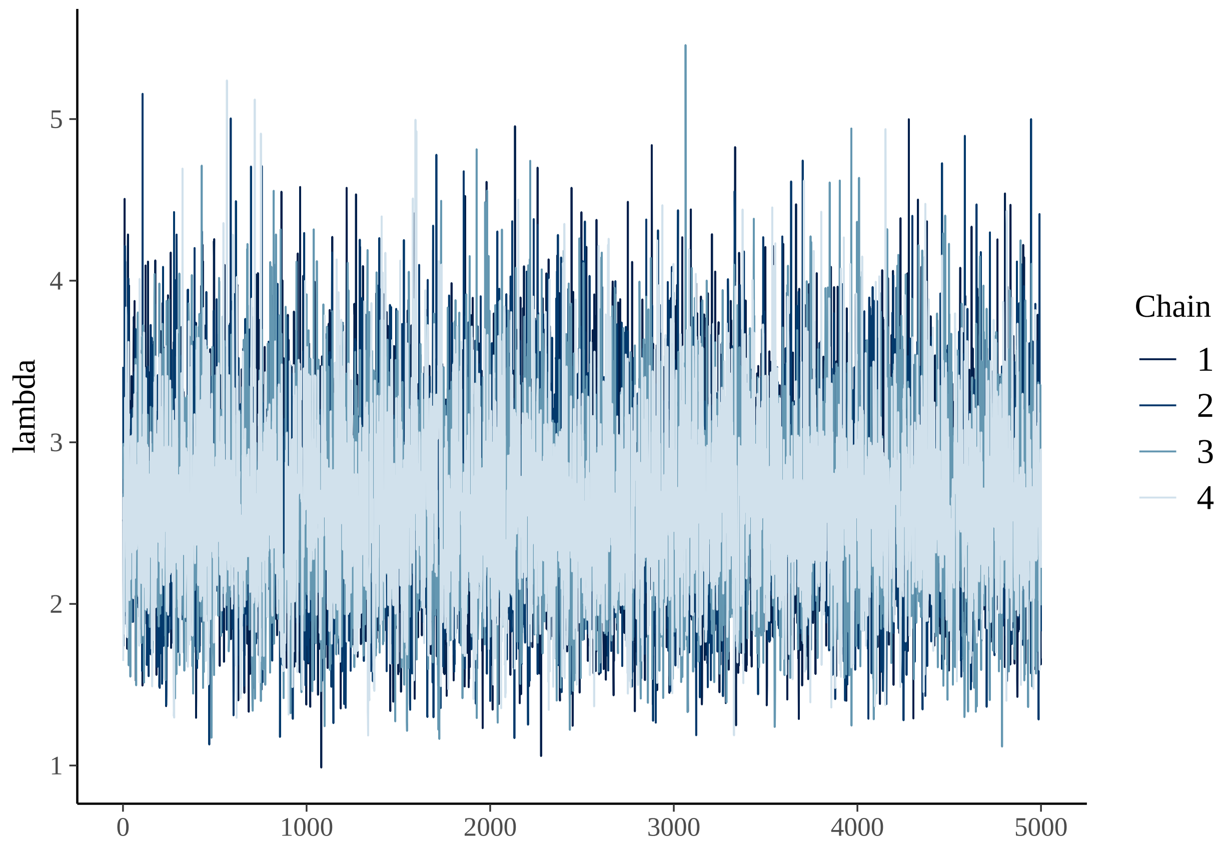
```
# STEP 1: DEFINE the model
gp_model <- "
  data {
    int<lower=0> Y[3];
  }
  parameters {
    real<lower=0> lambda;
  }
  model {
    Y ~ poisson(lambda);
    lambda ~ gamma(20,5);
  }
"

# Step 2: Approximate the posterior

gp_sim <- stan(model_code = gp_model, data = list(Y=c(0,1,0)),
               chains = 4, iter = 5000*2, seed = 84735)
```

```
##
```

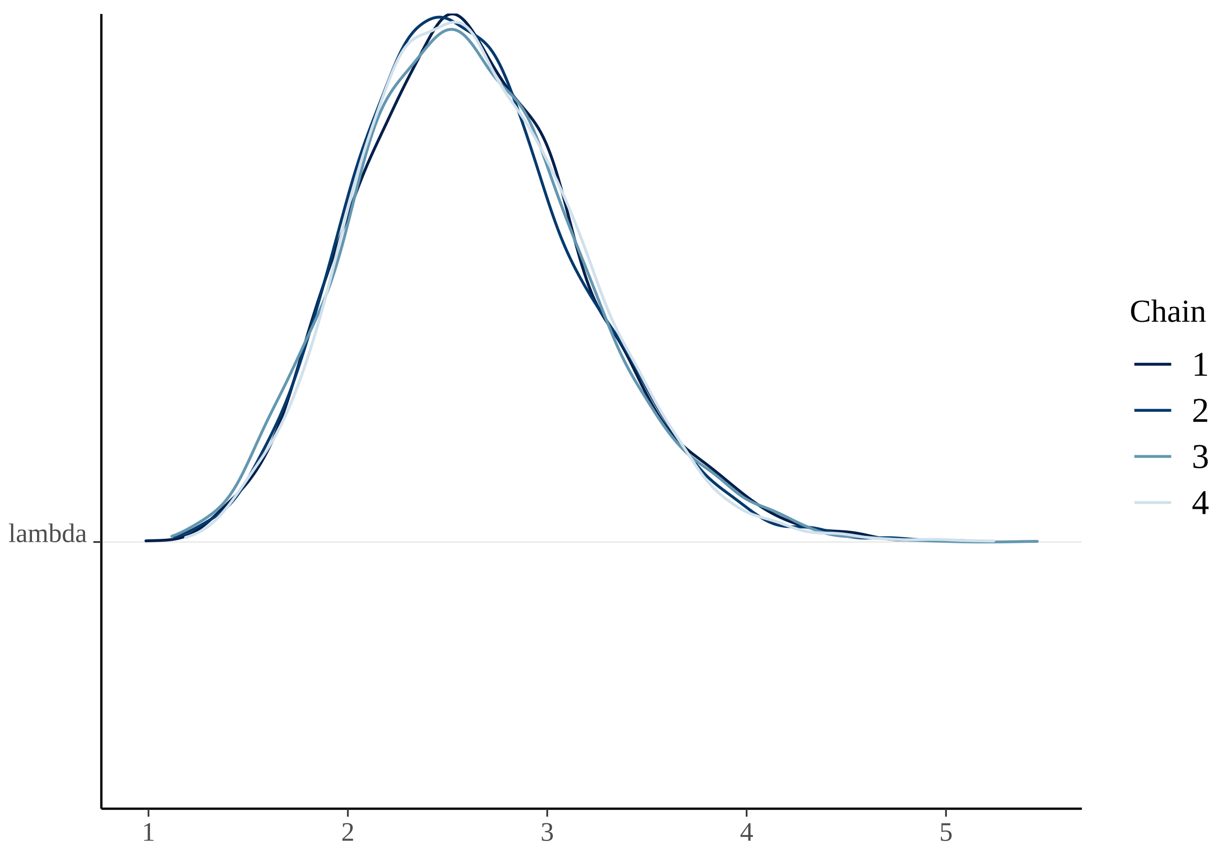
```
## SAMPLING FOR MODEL '2cdba1e7e78935579bf89a2786e25c8f' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## 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: 0.063 seconds (Warm-up)
## Chain 1:                0.06 seconds (Sampling)
## Chain 1:                0.123 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '2cdba1e7e78935579bf89a2786e25c8f' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.066 seconds (Warm-up)
## Chain 2:                0.062 seconds (Sampling)
## Chain 2:                0.128 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '2cdba1e7e78935579bf89a2786e25c8f' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.06 seconds (Warm-up)
## Chain 3:                0.065 seconds (Sampling)
## Chain 3:                0.125 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '2cdba1e7e78935579bf89a2786e25c8f' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.063 seconds (Warm-up)
## Chain 4:                0.081 seconds (Sampling)
## Chain 4:                0.144 seconds (Total)
## Chain 4:
```

b.

```
mcmc_trace(gp_sim,pars="lambda")
```



```
mcmc_dens_chains(gp_sim,pars="lambda")
```



c. 2.5 seems the most likely value

d.

The posterior mode is 2.6!

```
summarize_gamma_poisson(20,5,1,3)
```

```
##       model shape rate  mean mode      var        sd
## 1     prior    20    5 4.000  3.8 0.800000 0.8944272
## 2 posterior    21    8 2.625  2.5 0.328125 0.5728220
```

Exercise 6.16

a.

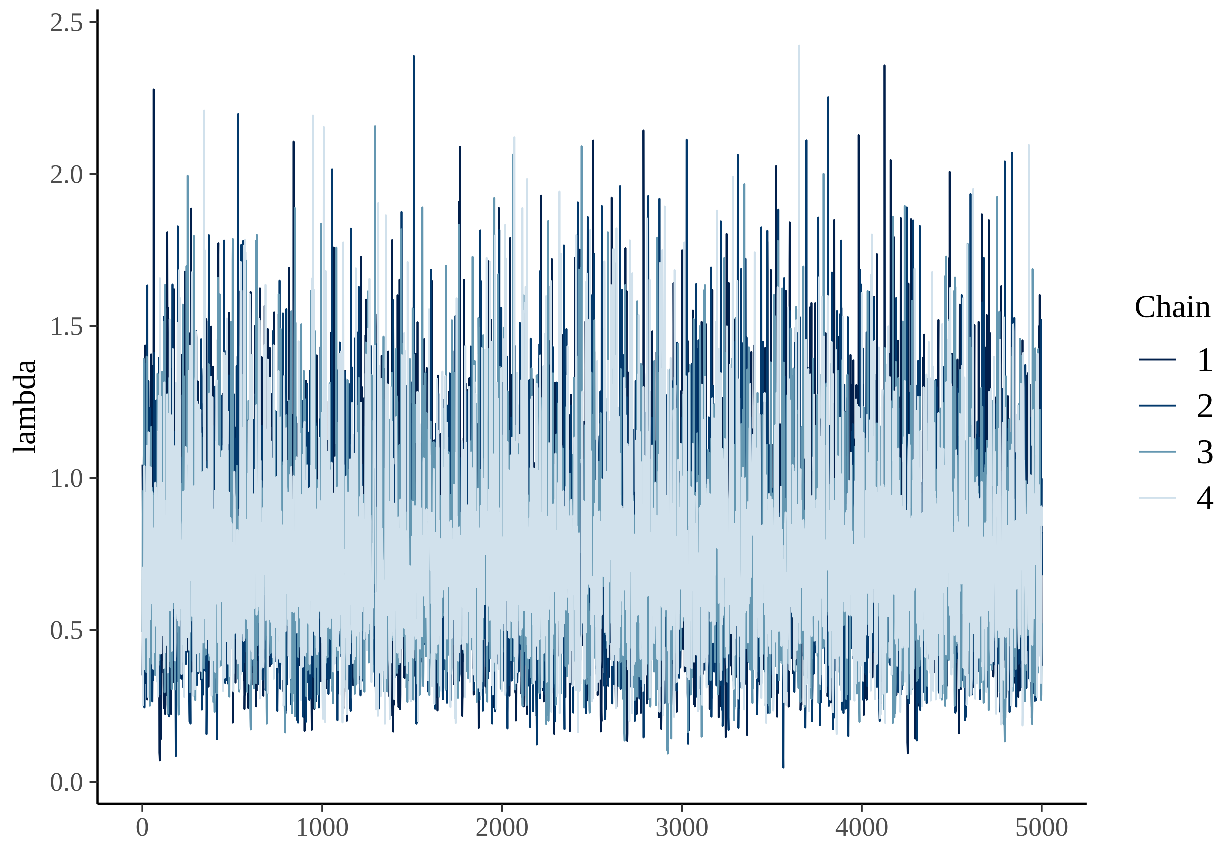
```
# STEP 1: DEFINE the model
gp_model <- "
  data {
    int<lower=0> Y[3];
  }
  parameters {
    real<lower=0> lambda;
  }
  model {
    Y ~ poisson(lambda);
    lambda ~ gamma(5,5);
  }
"

# Step 2: Approximate the posterior

gp_sim <- stan(model_code = gp_model, data = list(Y=c(0,1,0)),
               chains = 4, iter = 5000*2, seed = 84735)
```

```
##
## SAMPLING FOR MODEL 'c507babd202babfa2331bfaa1980cf34' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.079 seconds (Warm-up)
## Chain 1:                0.081 seconds (Sampling)
## Chain 1:                0.16 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'c507babd202babfa2331bfaa1980cf34' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.082 seconds (Warm-up)
## Chain 2:                0.098 seconds (Sampling)
## Chain 2:                0.18 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'c507babd202babfa2331bfaa1980cf34' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.092 seconds (Warm-up)
## Chain 3:                0.097 seconds (Sampling)
## Chain 3:                0.189 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'c507babd202babfa2331bfaa1980cf34' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0 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:
```

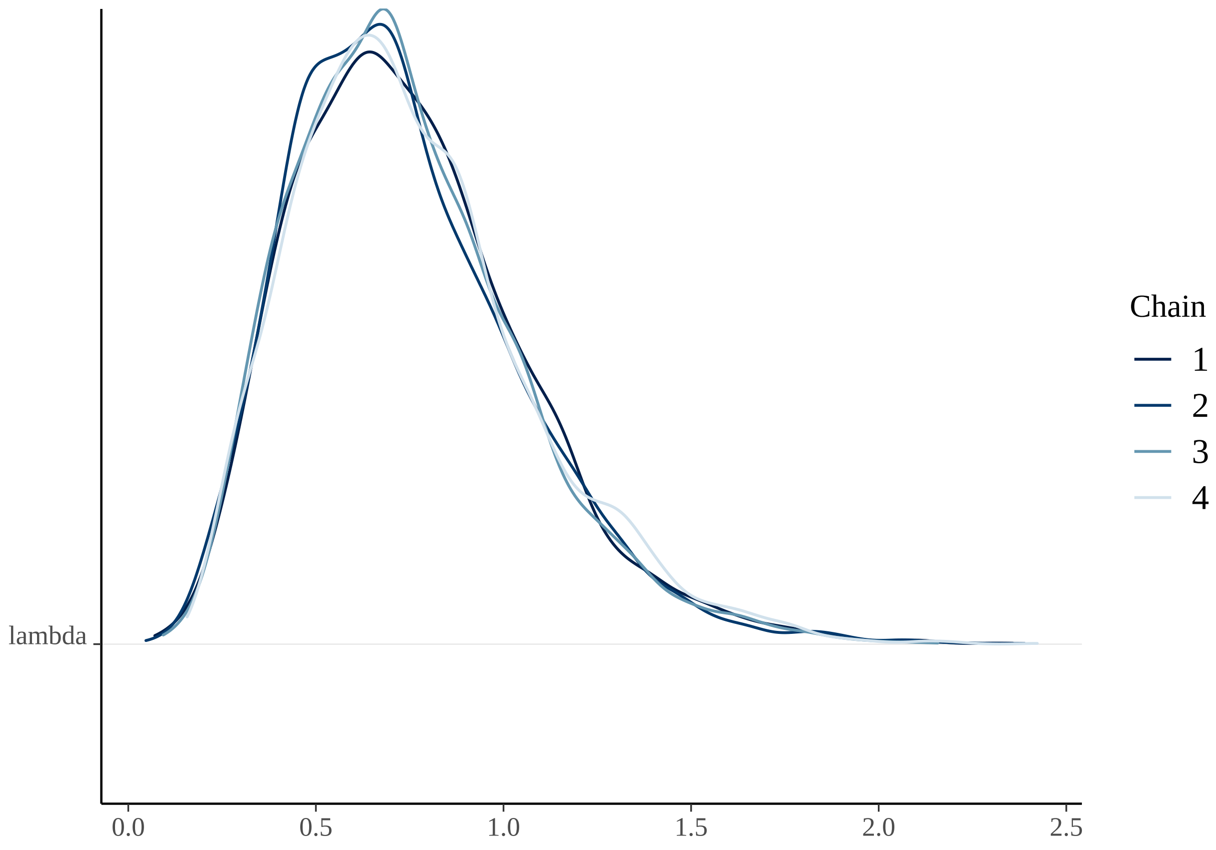
```
## Chain 4:  Elapsed Time: 0.079 seconds (Warm-up)
## Chain 4:                0.102 seconds (Sampling)
## Chain 4:                0.181 seconds (Total)
## Chain 4:
```

b.

```
mcmc_trace(gp_sim,pars="lambda")
```



```
mcmc_dens_chains(gp_sim,pars="lambda")
```



c. In this model, the most posterior plausible value seems to be .6

d.

The mode is exactly .6!

```
summarize_gamma_poisson(5,5,1,3)
```

```
##       model shape rate mean  mode     var        sd
## 1     prior     5    5 1.00 0.800 0.20000 0.4472136
## 2 posterior     6    8 0.75 0.625 0.09375 0.3061862
```

6.17

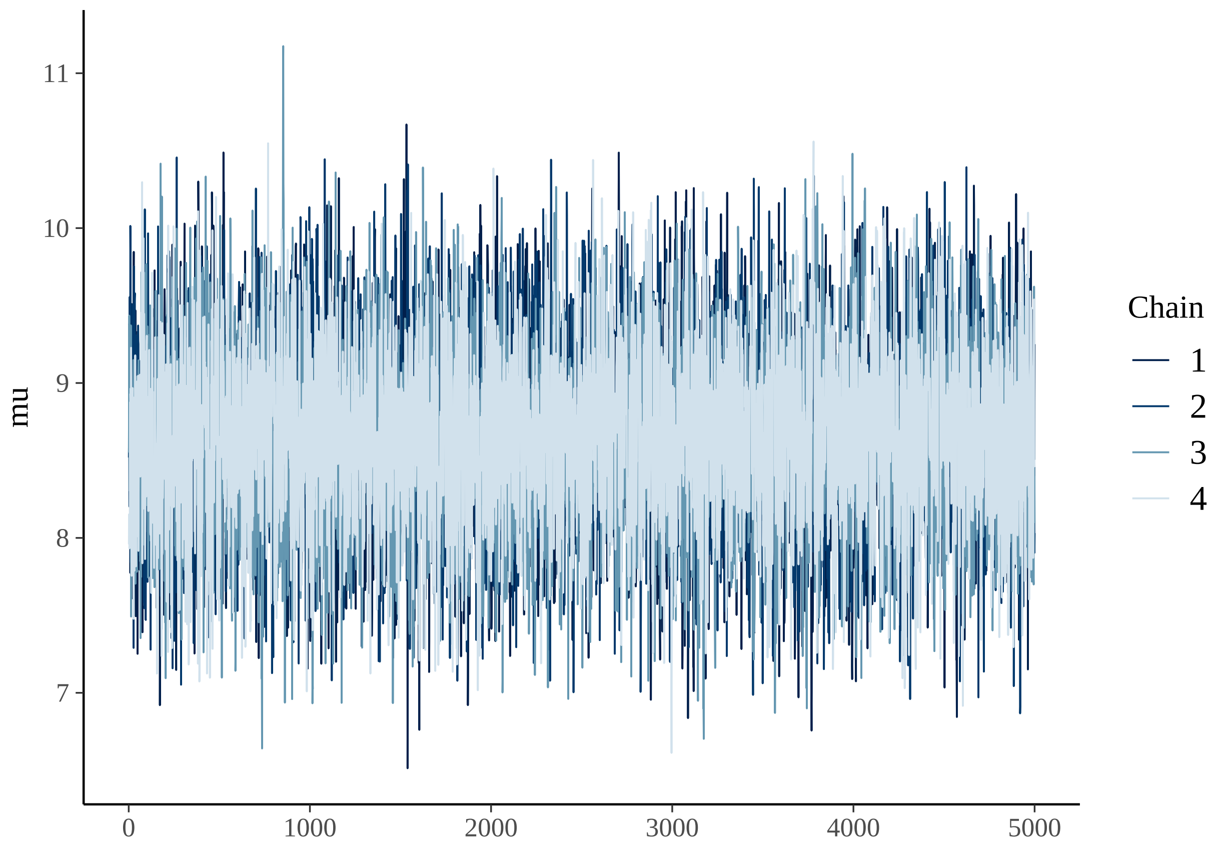
```
n_model <- '
data {
    vector[4] Y;
}
parameters {
    real mu;
}
model {
   Y ~ normal(mu, 1.3);
```

```
  mu ~ normal(10,1.2);
}
'


gn_sim <- stan(model_code = n_model, data = list(Y = c(7.1, 8.9, 8.4, 8.6)),
               chains = 4, iter = 5000*2, seed = 568956)
```

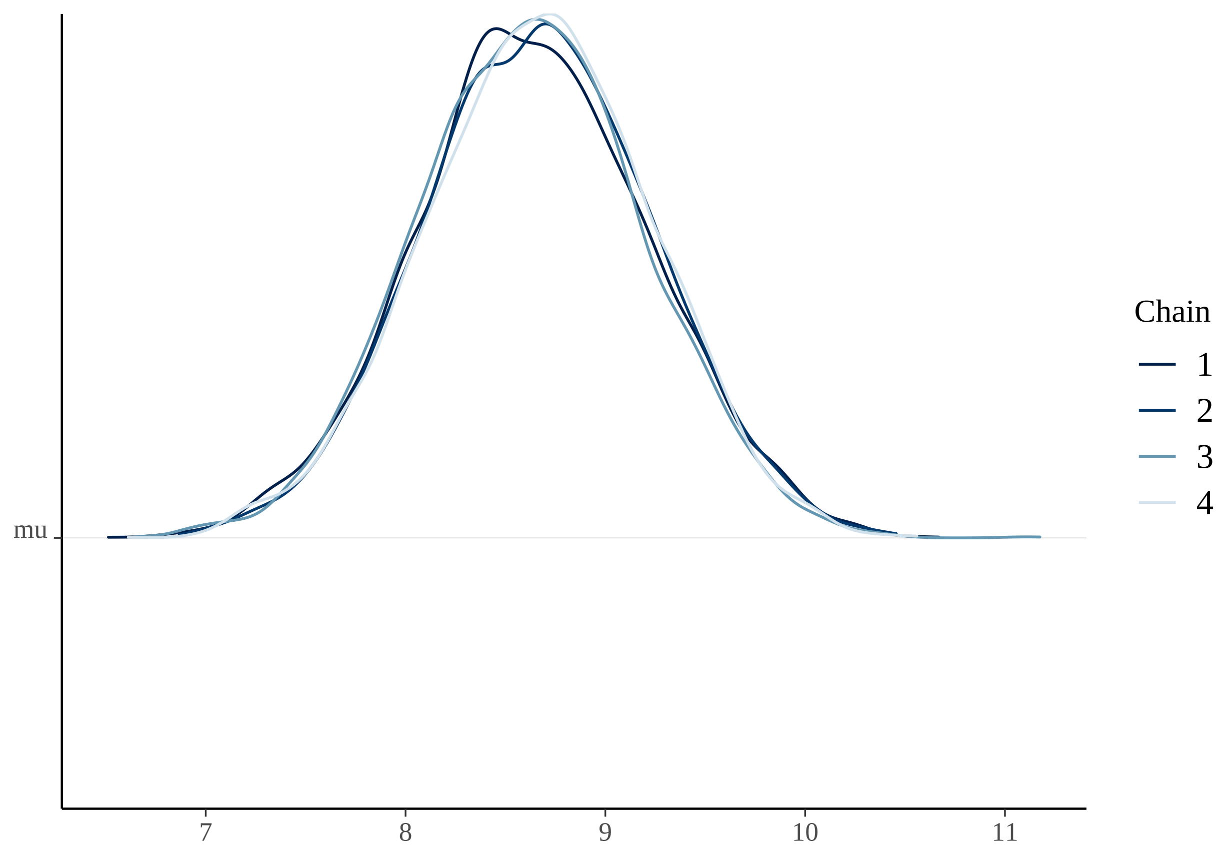
```
##
## SAMPLING FOR MODEL '86182bf1bd7fad6168ec1ee0c0de390e' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## 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: 0.056 seconds (Warm-up)
## Chain 1:                0.062 seconds (Sampling)
## Chain 1:                0.118 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '86182bf1bd7fad6168ec1ee0c0de390e' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.059 seconds (Warm-up)
## Chain 2:                0.058 seconds (Sampling)
## Chain 2:                0.117 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '86182bf1bd7fad6168ec1ee0c0de390e' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.057 seconds (Warm-up)
## Chain 3:                0.056 seconds (Sampling)
## Chain 3:                0.113 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '86182bf1bd7fad6168ec1ee0c0de390e' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.06 seconds (Warm-up)
## Chain 4:                0.066 seconds (Sampling)
## Chain 4:                0.126 seconds (Total)
## Chain 4:
```

b.

```
mcmc_trace(gn_sim,pars="mu")
```



```
mcmc_dens_chains(gn_sim,pars="mu")
```



c. In this model, the most posterior plausible value seems to be 8.5

d. The mode is 8.6 in the real posterior, which matches our finding above.

```
summarize_normal_normal(mean = 10,sd = 1.2,sigma = 1.3,y_bar = mean(c(7.1, 8.9, 8.4, 8.6)), n = 4)
```

```
##       model     mean     mode       var        sd
## 1     prior 10.00000 10.00000 1.4400000 1.2000000
## 2 posterior  8.64698  8.64698 0.3266577 0.5715398
```

6.18

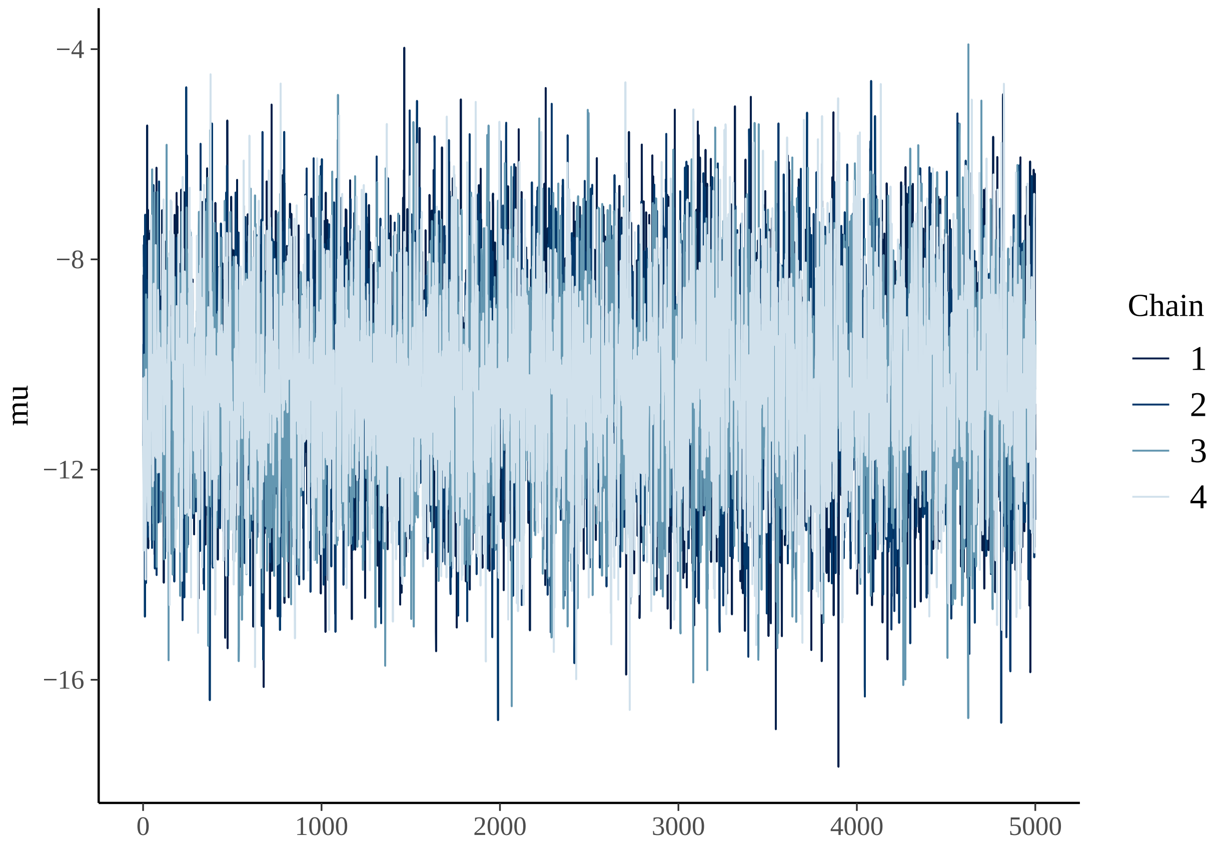
```
n_model <- '
data {
    vector[5] Y;
}
parameters {
    real mu;
}
model {
   Y ~ normal(mu, 8);
   mu ~ normal(-14,2);
}
```

```
'

gn_sim <- stan(model_code = n_model, data = list(Y = c(-10.1,5.5,.1,-1.4,11.5)),
               chains = 4, iter = 5000*2, seed = 568956)
```

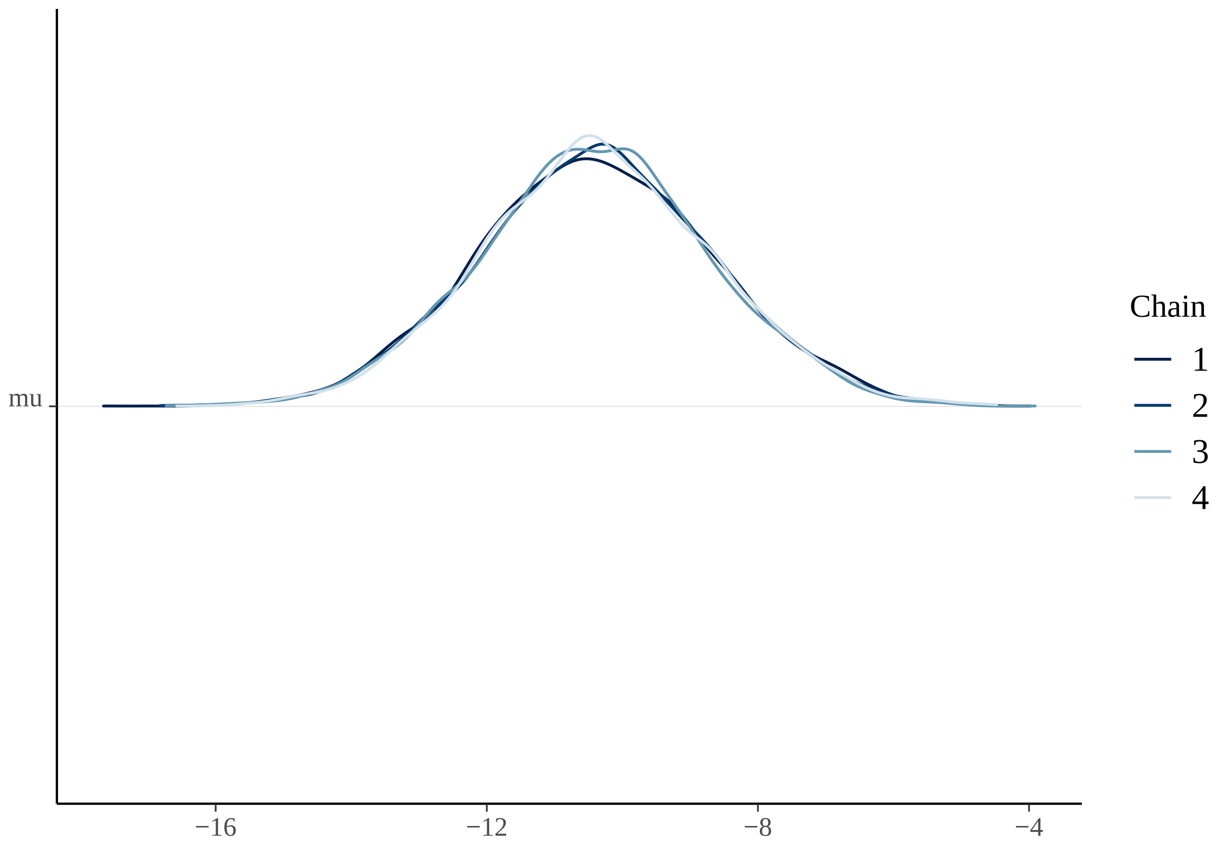
```
##
## SAMPLING FOR MODEL 'e4e3eb0ea1022bd18944ff5ef79abc3c' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## 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: 0.057 seconds (Warm-up)
## Chain 1:                0.056 seconds (Sampling)
## Chain 1:                0.113 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'e4e3eb0ea1022bd18944ff5ef79abc3c' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.056 seconds (Warm-up)
## Chain 2:                0.063 seconds (Sampling)
## Chain 2:                0.119 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'e4e3eb0ea1022bd18944ff5ef79abc3c' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.056 seconds (Warm-up)
## Chain 3:                0.067 seconds (Sampling)
## Chain 3:                0.123 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'e4e3eb0ea1022bd18944ff5ef79abc3c' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 0.055 seconds (Warm-up)
## Chain 4:                0.055 seconds (Sampling)
## Chain 4:                0.11 seconds (Total)
## Chain 4:
```

b.

```
mcmc_trace(gn_sim,pars="mu")
```



```
mcmc_dens_chains(gn_sim,pars="mu")
```



c. In this model, the most posterior plausible value seems to be -10

d. The mode is -10 in the real posterior, which matches our finding above!!

```
summarize_normal_normal(mean = -14,sd = 2,sigma = 8,y_bar = mean(c(-10.1,5.5,.1,-1.4,11.5)), n = 5)
```

```
##       model  mean  mode      var       sd
## 1     prior -14.0 -14.0 4.000000 2.000000
## 2 posterior -10.4 -10.4 3.047619 1.745743
```