BR6

Steph Jordan

```r
library(bayesrules)
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()
```

```r
library(janitor)
```

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

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

```r
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)

##
## Attaching package: 'rstan'

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

```r
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
```

Exercise 6.1

a. Steps to calculate grid model

1. Define a discrete set of possible pi values
2. Evaluate the prior f(pi) for all values of pi, and the likelihood values for all values of pi.
3. Solve for the posterior model f(pi|y)
4. Randomly sample N pi values with respect to their calculated posterior probabilities

b. I would increase the length of the pi values vector. This would enable us to have a more "granular" or finely tuned understanding of the posterior (we will have thinner grid "slices").

Exercise 6.2

Exercise 6.3

a. The approximation might not reach exploration of lower pi values. Therefore, it could overestimate the plausibility of pi values in the range it reached (high pi values), and underestimate the plausibility of pi values outside the range it reached (low pi values), leading to a density plot that is more right skewed than the posterior.

b. High correlation means that the effective sample size ratio is small, which is a warning sign that our approximation could be unreliable. Some correlation between values is to be expected given each value's dependence on its immediate neighbors. But this correlation should fade with distance (i.e., the chain value of x_i should be close to x_(i-1) but far from x_(i-100)). If there is high correlation, it's possible that our model is moving slowly, or our effective sample size ratio is too small.

c. If the approximation has a tendency to get "stuck," say, at low values of pi, then it will produce a density plot with "blips" that is not as smooth as the posterior curve. The approximation will overestimate the plausibility of certain low values of pi, producing erroneous peaks in the density curve.

Exercise 6.4

a. It is important to look at MCMC diagnostics because they allow us to assess how close the approximation is to the posterior. Diagnostics can also tell us how big our Markov chain sample size needs to be in order to produce a reliable approximation.

b. MCMC simulations are helpful because they allow for approximations of more complicated Bayesian models (in contrast to grid approximations, which sample only from a discretized approximation of the posterior pdf). MCMC simulations allow us to approximate the posterior of more complicated and scaled up Bayesian models. When we know the posterior, they serve as a means to verify our work; when we're not able to develop a posterior, MCMC models provide a crucial approximation.

c. The benefit of using RSTAN is that it combines the technical simplicity of R with the computing power of the Stan engine. This allows us to run MCMC models with long chains, without overwhelming our computational power.

d. I still don't understand what would cause an MCMC simulation to mix slowly or quickly. I think it might have to do with the effective sample size ratio–if this is too small, the model mixes slowly?

Exercise 6.5

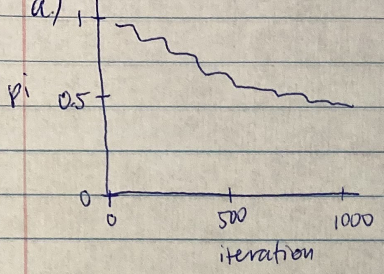
a. We'll follow the steps outlined in Chapt 6.

```r
grid_data <- data.frame(pi_grid = seq(from = 0, to = 1, length = 5))
grid_data
```

```
##   pi_grid
## 1    0.00
```

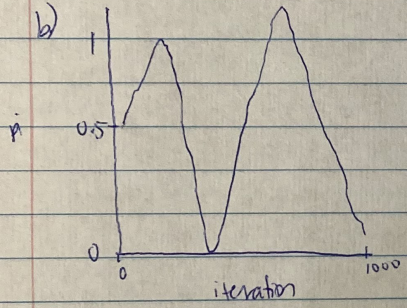
6.2

a)



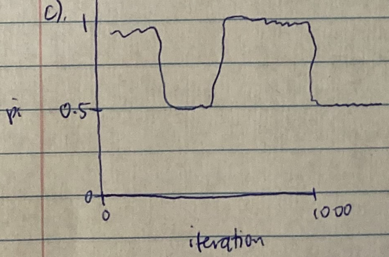
since chain is mixing slowly, fails to reach low values of pi

b)



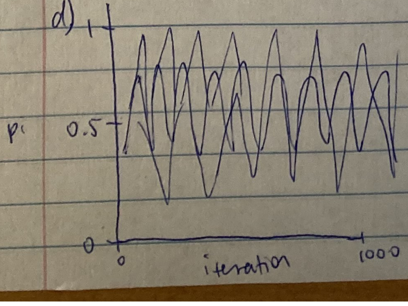
chain values that are many steps apart still have dependence; this yields a high autocorrelation

c)



since the chain has "flat" stretches at low values of pi, it gets "stuck" at certain levels

d)



the chain visits low & high values of pi and demonstrates little dependence between one value & the next

Figure 1: caption

```
## 2    0.25
## 3    0.50
## 4    0.75
## 5    1.00
```

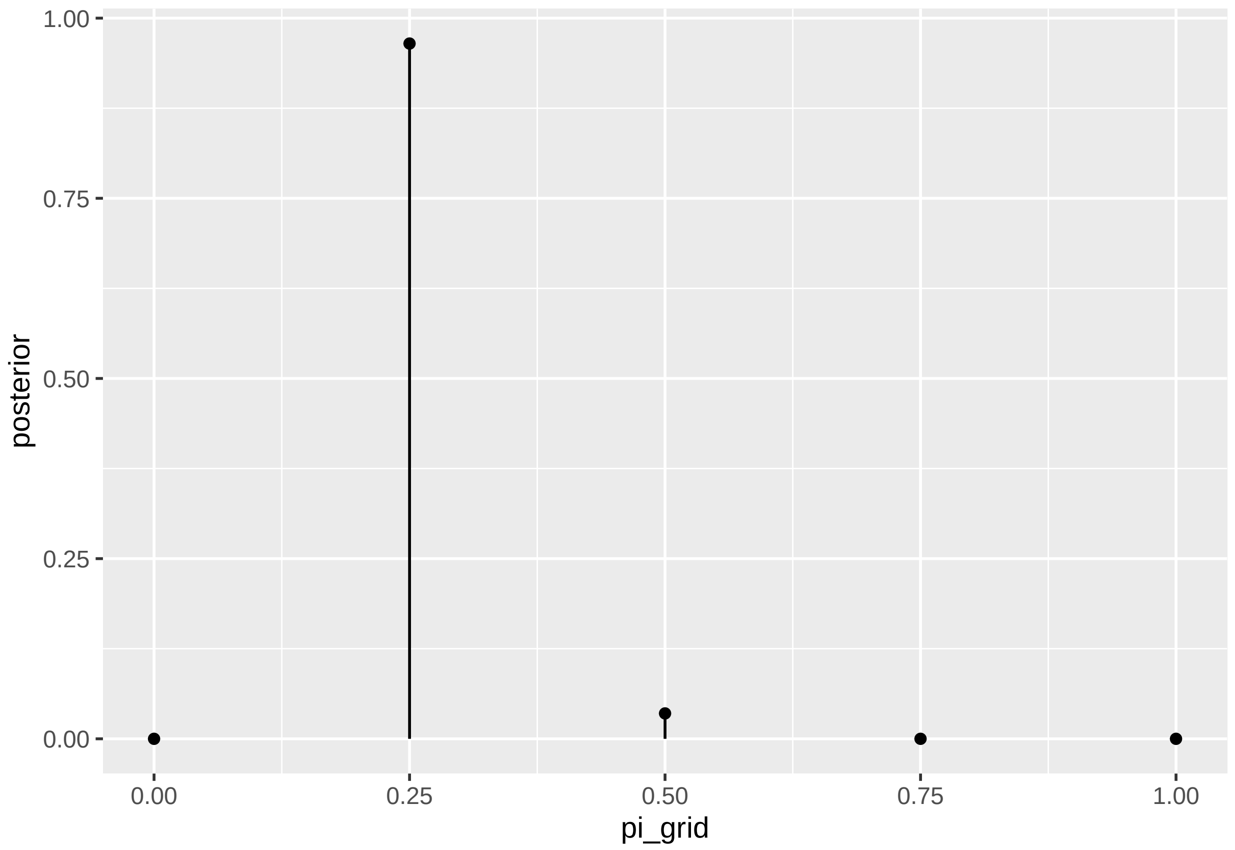
```
# Step 2: Evaluate the prior & likelihood at each pi
grid_data <- grid_data |>
  mutate(prior = dbeta(pi_grid, 3, 8),
         likelihood = dbinom(2, 10, pi_grid))

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

# Confirm that the posterior approximation sums to 1
grid_data |>
  summarize(sum(unnormalized), sum(posterior))
```

```
##   sum(unnormalized) sum(posterior)
## 1         0.8765603              1
```

```
# Examine the grid approximated posterior
round(grid_data, 2)
```

```
##   pi_grid prior likelihood unnormalized posterior
## 1    0.00  0.00       0.00         0.00      0.00
## 2    0.25  3.00       0.28         0.85      0.96
## 3    0.50  0.70       0.04         0.03      0.04
## 4    0.75  0.01       0.00         0.00      0.00
## 5    1.00  0.00       0.00         0.00      0.00
```

```
# Plot the grid approximated posterior
ggplot(grid_data, aes(x = pi_grid, y = posterior)) +
  geom_point() +
  geom_segment(aes(x = pi_grid, xend = pi_grid, y = 0, yend = posterior))
```



b.

Repeating step a with 201 grid slices:

```
grid_data <- data.frame(pi_grid = seq(from = 0, to = 1, length = 201))
grid_data
```

```
##     pi_grid
## 1     0.000
## 2     0.005
## 3     0.010
## 4     0.015
## 5     0.020
## 6     0.025
## 7     0.030
## 8     0.035
## 9     0.040
## 10    0.045
## 11    0.050
## 12    0.055
## 13    0.060
## 14    0.065
## 15    0.070
## 16    0.075
## 17    0.080
## 18    0.085
## 19    0.090
## 20    0.095
## 21    0.100
## 22    0.105
## 23    0.110
## 24    0.115
## 25    0.120
## 26    0.125
## 27    0.130
## 28    0.135
## 29    0.140
## 30    0.145
## 31    0.150
## 32    0.155
## 33    0.160
## 34    0.165
## 35    0.170
## 36    0.175
## 37    0.180
## 38    0.185
## 39    0.190
## 40    0.195
## 41    0.200
## 42    0.205
## 43    0.210
## 44    0.215
## 45    0.220
## 46    0.225
## 47    0.230
## 48    0.235
## 49    0.240
## 50    0.245
## 51    0.250
## 52    0.255
## 53    0.260
## 54    0.265
## 55    0.270
## 56    0.275
## 57    0.280
## 58    0.285
## 59    0.290
## 60    0.295
## 61    0.300
## 62    0.305
## 63    0.310
## 64    0.315
## 65    0.320
## 66    0.325
## 67    0.330
## 68    0.335
## 69    0.340
## 70    0.345
## 71    0.350
## 72    0.355
## 73    0.360
## 74    0.365
## 75    0.370
## 76    0.375
## 77    0.380
## 78    0.385
## 79    0.390
## 80    0.395
## 81    0.400
## 82    0.405
## 83    0.410
## 84    0.415
## 85    0.420
## 86    0.425
## 87    0.430
## 88    0.435
## 89    0.440
## 90    0.445
## 91    0.450
## 92    0.455
## 93    0.460
## 94    0.465
## 95    0.470
## 96    0.475
## 97    0.480
## 98    0.485
## 99    0.490
## 100   0.495
## 101   0.500
## 102   0.505
## 103   0.510
## 104   0.515
## 105   0.520
## 106   0.525
## 107   0.530
## 108   0.535
## 109   0.540
## 110   0.545
## 111   0.550
## 112   0.555
## 113   0.560
## 114   0.565
## 115   0.570
## 116   0.575
## 117   0.580
## 118   0.585
## 119   0.590
## 120   0.595
## 121   0.600
## 122   0.605
## 123   0.610
## 124   0.615
## 125   0.620
## 126   0.625
## 127   0.630
## 128   0.635
## 129   0.640
## 130   0.645
## 131   0.650
## 132   0.655
## 133   0.660
## 134   0.665
## 135   0.670
## 136   0.675
## 137   0.680
## 138   0.685
## 139   0.690
## 140   0.695
## 141   0.700
## 142   0.705
## 143   0.710
## 144   0.715
## 145   0.720
## 146   0.725
## 147   0.730
## 148   0.735
## 149   0.740
## 150   0.745
## 151   0.750
## 152   0.755
## 153   0.760
## 154   0.765
## 155   0.770
## 156   0.775
## 157   0.780
## 158   0.785
## 159   0.790
## 160   0.795
## 161   0.800
## 162   0.805
## 163   0.810
## 164   0.815
## 165   0.820
## 166   0.825
## 167   0.830
## 168   0.835
## 169   0.840
## 170   0.845
## 171   0.850
## 172   0.855
## 173   0.860
## 174   0.865
## 175   0.870
## 176   0.875
## 177   0.880
## 178   0.885
## 179   0.890
## 180   0.895
## 181   0.900
## 182   0.905
## 183   0.910
## 184   0.915
## 185   0.920
## 186   0.925
## 187   0.930
## 188   0.935
## 189   0.940
## 190   0.945
## 191   0.950
## 192   0.955
## 193   0.960
## 194   0.965
## 195   0.970
## 196   0.975
## 197   0.980
## 198   0.985
## 199   0.990
## 200   0.995
## 201   1.000
```

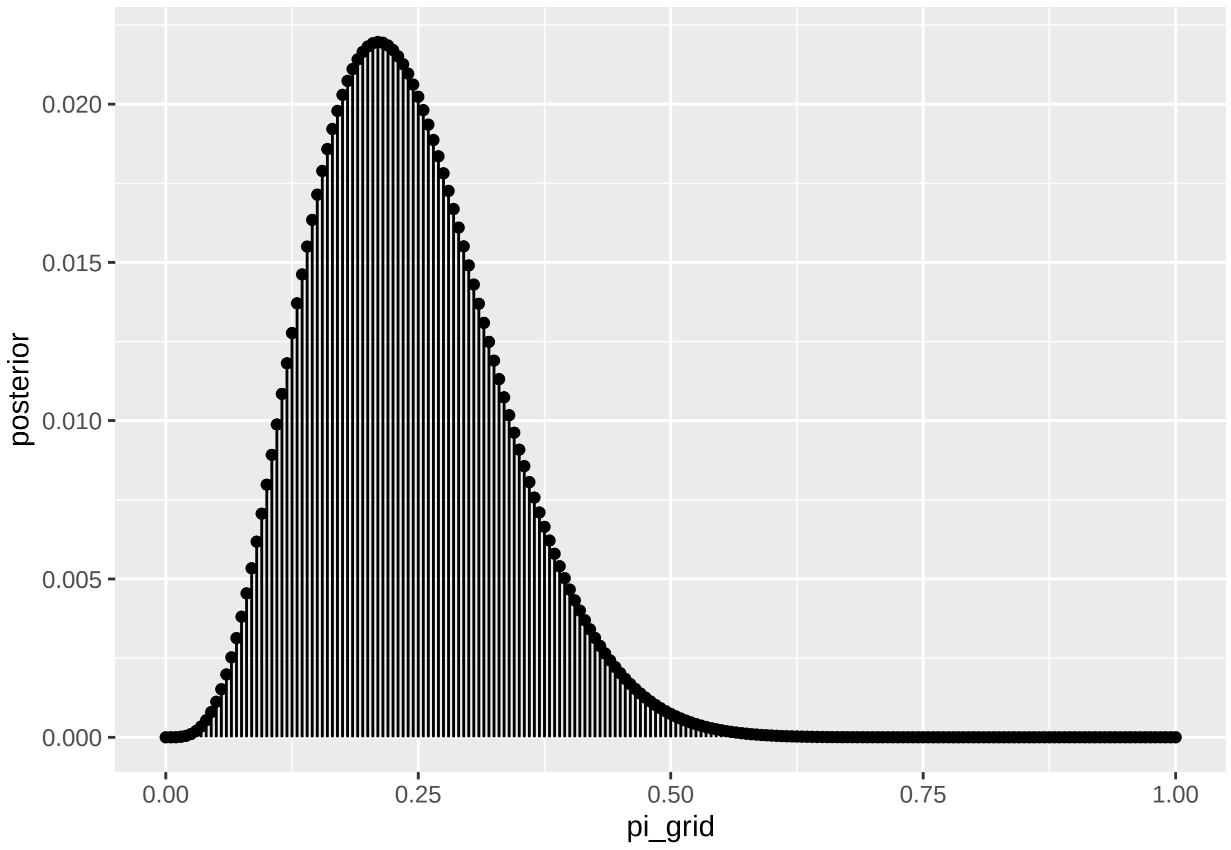
```
# Step 2: Evaluate the prior & likelihood at each pi
grid_data <- grid_data |>
  mutate(prior = dbeta(pi_grid, 3, 8),
         likelihood = dbinom(2, 10, pi_grid))

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

# Confirm that the posterior approximation sums to 1
grid_data |>
  summarize(sum(unnormalized), sum(posterior))
```

```
##   sum(unnormalized) sum(posterior)
## 1          41.79567              1
```

```
# Examine the grid approximated posterior
round(grid_data, 2)
```

```
##     pi_grid prior likelihood unnormalized posterior
## 1      0.00  0.00       0.00         0.00      0.00
## 2      0.00  0.01       0.00         0.00      0.00
## 3      0.01  0.03       0.00         0.00      0.00
## 4      0.01  0.07       0.01         0.00      0.00
## 5      0.02  0.13       0.02         0.00      0.00
## 6      0.03  0.19       0.02         0.00      0.00
## 7      0.03  0.26       0.03         0.01      0.00
## 8      0.04  0.34       0.04         0.01      0.00
## 9      0.04  0.43       0.05         0.02      0.00
## 10     0.04  0.53       0.06         0.03      0.00
## 11     0.05  0.63       0.07         0.05      0.00
## 12     0.06  0.73       0.09         0.06      0.00
## 13     0.06  0.84       0.10         0.08      0.00
## 14     0.06  0.95       0.11         0.11      0.00
## 15     0.07  1.06       0.12         0.13      0.00
## 16     0.07  1.17       0.14         0.16      0.00
## 17     0.08  1.29       0.15         0.19      0.00
## 18     0.09  1.40        0.16          0.22       0.01
## 19     0.09  1.51        0.17          0.26       0.01
## 20     0.10  1.62        0.18          0.30       0.01
## 21     0.10  1.72        0.19          0.33       0.01
## 22     0.10  1.83        0.20          0.37       0.01
## 23     0.11  1.93        0.21          0.41       0.01
## 24     0.12  2.02        0.22          0.45       0.01
## 25     0.12  2.12        0.23          0.49       0.01
## 26     0.12  2.21        0.24          0.53       0.01
## 27     0.13  2.30        0.25          0.57       0.01
## 28     0.14  2.38        0.26          0.61       0.01
## 29     0.14  2.45        0.26          0.65       0.02
## 30     0.14  2.53        0.27          0.68       0.02
## 31     0.15  2.60        0.28          0.72       0.02
## 32     0.16  2.66        0.28          0.75       0.02
## 33     0.16  2.72        0.29          0.78       0.02
## 34     0.16  2.77        0.29          0.80       0.02
## 35     0.17  2.82        0.29          0.83       0.02
## 36     0.18  2.87        0.30          0.85       0.02
## 37     0.18  2.91        0.30          0.87       0.02
## 38     0.18  2.94        0.30          0.88       0.02
## 39     0.19  2.97        0.30          0.89       0.02
## 40     0.20  3.00        0.30          0.90       0.02
## 41     0.20  3.02        0.30          0.91       0.02
## 42     0.21  3.04        0.30          0.92       0.02
## 43     0.21  3.05        0.30          0.92       0.02
## 44     0.22  3.06        0.30          0.92       0.02
## 45     0.22  3.06        0.30          0.91       0.02
## 46     0.22  3.06        0.30          0.91       0.02
## 47     0.23  3.06        0.29          0.90       0.02
## 48     0.24  3.05        0.29          0.89       0.02
## 49     0.24  3.04        0.29          0.88       0.02
## 50     0.24  3.02        0.29          0.86       0.02
## 51     0.25  3.00        0.28          0.85       0.02
## 52     0.26  2.98        0.28          0.83       0.02
## 53     0.26  2.96        0.27          0.81       0.02
## 54     0.26  2.93        0.27          0.79       0.02
## 55     0.27  2.90        0.26          0.77       0.02
## 56     0.28  2.87        0.26          0.74       0.02
## 57     0.28  2.83        0.25          0.72       0.02
## 58     0.29  2.79        0.25          0.70       0.02
## 59     0.29  2.75        0.24          0.67       0.02
## 60     0.30  2.71        0.24          0.65       0.02
## 61     0.30  2.67        0.23          0.62       0.01
## 62     0.30  2.62        0.23          0.60       0.01
## 63     0.31  2.58        0.22          0.57       0.01
## 64     0.32  2.53        0.22          0.55       0.01
## 65     0.32  2.48        0.21          0.52       0.01
## 66     0.32  2.43        0.20          0.50       0.01
## 67     0.33  2.38        0.20          0.47       0.01
## 68     0.34  2.32        0.19          0.45       0.01
## 69     0.34  2.27        0.19          0.43       0.01
## 70     0.35  2.22        0.18          0.40       0.01
## 71     0.35  2.16        0.18          0.38       0.01
## 72     0.36  2.11        0.17         0.36      0.01
## 73     0.36  2.05        0.16         0.34      0.01
## 74     0.36  2.00        0.16         0.32      0.01
## 75     0.37  1.94        0.15         0.30      0.01
## 76     0.38  1.89        0.15         0.28      0.01
## 77     0.38  1.83        0.14         0.26      0.01
## 78     0.38  1.78        0.14         0.24      0.01
## 79     0.39  1.72        0.13         0.23      0.01
## 80     0.40  1.67        0.13         0.21      0.01
## 81     0.40  1.61        0.12         0.19      0.00
## 82     0.41  1.56        0.12         0.18      0.00
## 83     0.41  1.51        0.11         0.17      0.00
## 84     0.42  1.45        0.11         0.15      0.00
## 85     0.42  1.40        0.10         0.14      0.00
## 86     0.42  1.35        0.10         0.13      0.00
## 87     0.43  1.30        0.09         0.12      0.00
## 88     0.44  1.25        0.09         0.11      0.00
## 89     0.44  1.20        0.08         0.10      0.00
## 90     0.44  1.16        0.08         0.09      0.00
## 91     0.45  1.11        0.08         0.08      0.00
## 92     0.46  1.06        0.07         0.08      0.00
## 93     0.46  1.02        0.07         0.07      0.00
## 94     0.47  0.98        0.07         0.06      0.00
## 95     0.47  0.93        0.06         0.06      0.00
## 96     0.48  0.89        0.06         0.05      0.00
## 97     0.48  0.85        0.06         0.05      0.00
## 98     0.48  0.81        0.05         0.04      0.00
## 99     0.49  0.78        0.05         0.04      0.00
## 100    0.50  0.74        0.05         0.03      0.00
## 101    0.50  0.70        0.04         0.03      0.00
## 102    0.50  0.67        0.04         0.03      0.00
## 103    0.51  0.64        0.04         0.02      0.00
## 104    0.52  0.60        0.04         0.02      0.00
## 105    0.52  0.57        0.03         0.02      0.00
## 106    0.52  0.54        0.03         0.02      0.00
## 107    0.53  0.51        0.03         0.02      0.00
## 108    0.54  0.48        0.03         0.01      0.00
## 109    0.54  0.46        0.03         0.01      0.00
## 110    0.54  0.43        0.02         0.01      0.00
## 111    0.55  0.41        0.02         0.01      0.00
## 112    0.56  0.38        0.02         0.01      0.00
## 113    0.56  0.36        0.02         0.01      0.00
## 114    0.57  0.34        0.02         0.01      0.00
## 115    0.57  0.32        0.02         0.01      0.00
## 116    0.58  0.30        0.02         0.00      0.00
## 117    0.58  0.28        0.01         0.00      0.00
## 118    0.58  0.26        0.01         0.00      0.00
## 119    0.59  0.24        0.01         0.00      0.00
## 120    0.60  0.23        0.01         0.00      0.00
## 121    0.60  0.21        0.01         0.00      0.00
## 122    0.60  0.20        0.01         0.00      0.00
## 123    0.61  0.18        0.01         0.00      0.00
## 124    0.62  0.17        0.01         0.00      0.00
## 125    0.62  0.16        0.01         0.00      0.00
## 126    0.62  0.15       0.01          0.00       0.00
## 127    0.63  0.14       0.01          0.00       0.00
## 128    0.64  0.13       0.01          0.00       0.00
## 129    0.64  0.12       0.01          0.00       0.00
## 130    0.64  0.11       0.00          0.00       0.00
## 131    0.65  0.10       0.00          0.00       0.00
## 132    0.66  0.09       0.00          0.00       0.00
## 133    0.66  0.08       0.00          0.00       0.00
## 134    0.66  0.08       0.00          0.00       0.00
## 135    0.67  0.07       0.00          0.00       0.00
## 136    0.68  0.06       0.00          0.00       0.00
## 137    0.68  0.06       0.00          0.00       0.00
## 138    0.69  0.05       0.00          0.00       0.00
## 139    0.69  0.05       0.00          0.00       0.00
## 140    0.70  0.04       0.00          0.00       0.00
## 141    0.70  0.04       0.00          0.00       0.00
## 142    0.70  0.03       0.00          0.00       0.00
## 143    0.71  0.03       0.00          0.00       0.00
## 144    0.72  0.03       0.00          0.00       0.00
## 145    0.72  0.03       0.00          0.00       0.00
## 146    0.72  0.02       0.00          0.00       0.00
## 147    0.73  0.02       0.00          0.00       0.00
## 148    0.74  0.02       0.00          0.00       0.00
## 149    0.74  0.02       0.00          0.00       0.00
## 150    0.74  0.01       0.00          0.00       0.00
## 151    0.75  0.01       0.00          0.00       0.00
## 152    0.76  0.01       0.00          0.00       0.00
## 153    0.76  0.01       0.00          0.00       0.00
## 154    0.76  0.01       0.00          0.00       0.00
## 155    0.77  0.01       0.00          0.00       0.00
## 156    0.78  0.01       0.00          0.00       0.00
## 157    0.78  0.01       0.00          0.00       0.00
## 158    0.78  0.00       0.00          0.00       0.00
## 159    0.79  0.00       0.00          0.00       0.00
## 160    0.80  0.00       0.00          0.00       0.00
## 161    0.80  0.00       0.00          0.00       0.00
## 162    0.80  0.00       0.00          0.00       0.00
## 163    0.81  0.00       0.00          0.00       0.00
## 164    0.82  0.00       0.00          0.00       0.00
## 165    0.82  0.00       0.00          0.00       0.00
## 166    0.83  0.00       0.00          0.00       0.00
## 167    0.83  0.00       0.00          0.00       0.00
## 168    0.84  0.00       0.00          0.00       0.00
## 169    0.84  0.00       0.00          0.00       0.00
## 170    0.84  0.00       0.00          0.00       0.00
## 171    0.85  0.00       0.00          0.00       0.00
## 172    0.86  0.00       0.00          0.00       0.00
## 173    0.86  0.00       0.00          0.00       0.00
## 174    0.86  0.00       0.00          0.00       0.00
## 175    0.87  0.00       0.00          0.00       0.00
## 176    0.88  0.00       0.00          0.00       0.00
## 177    0.88  0.00       0.00          0.00       0.00
## 178    0.88  0.00       0.00          0.00       0.00
## 179    0.89  0.00       0.00          0.00       0.00
## 180     0.90  0.00        0.00          0.00       0.00
## 181     0.90  0.00        0.00          0.00       0.00
## 182     0.90  0.00        0.00          0.00       0.00
## 183     0.91  0.00        0.00          0.00       0.00
## 184     0.92  0.00        0.00          0.00       0.00
## 185     0.92  0.00        0.00          0.00       0.00
## 186     0.92  0.00        0.00          0.00       0.00
## 187     0.93  0.00        0.00          0.00       0.00
## 188     0.94  0.00        0.00          0.00       0.00
## 189     0.94  0.00        0.00          0.00       0.00
## 190     0.95  0.00        0.00          0.00       0.00
## 191     0.95  0.00        0.00          0.00       0.00
## 192     0.96  0.00        0.00          0.00       0.00
## 193     0.96  0.00        0.00          0.00       0.00
## 194     0.96  0.00        0.00          0.00       0.00
## 195     0.97  0.00        0.00          0.00       0.00
## 196     0.98  0.00        0.00          0.00       0.00
## 197     0.98  0.00        0.00          0.00       0.00
## 198     0.98  0.00        0.00          0.00       0.00
## 199     0.99  0.00        0.00          0.00       0.00
## 200     1.00  0.00        0.00          0.00       0.00
## 201     1.00  0.00        0.00          0.00       0.00
```

```r
# Plot the grid approximated posterior
ggplot(grid_data, aes(x = pi_grid, y = posterior)) +
  geom_point() +
  geom_segment(aes(x = pi_grid, xend = pi_grid, y = 0, yend = posterior))
```



Exercise 6.6

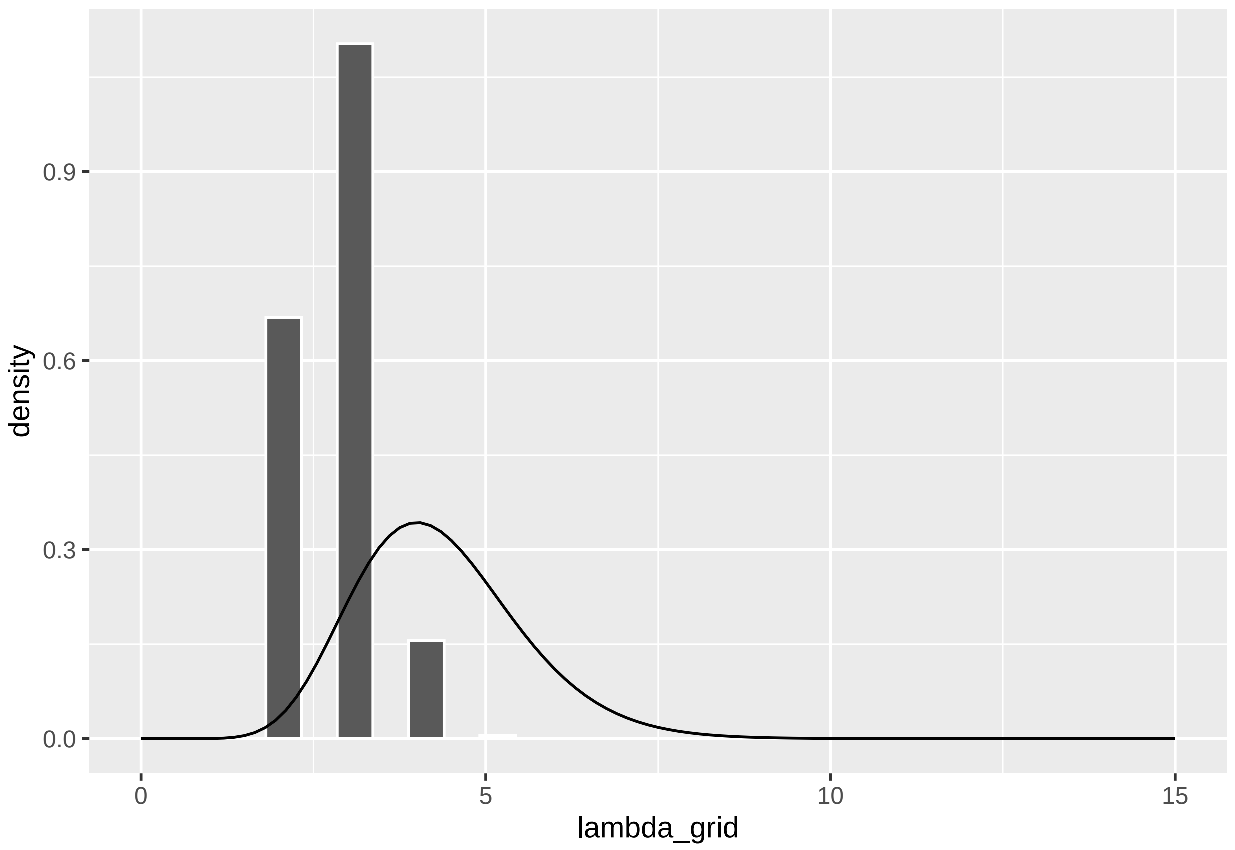
a. Approximating the posterior model of lambda for lambda in {0, 1, 2, . . . 8}

```
s <- 20
r <- 5


# Step 1: Define a grid of 501 lambda values
grid_data   <- data.frame(
  lambda_grid = seq(from = 0, to = 8, length = 9))


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

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

```
# 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(13, 3)) +
  lims(x = c(0, 15))
```

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

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



b. Approximating the posterior model of lambda for 201 equally spaced lambda values between 0 and 1.

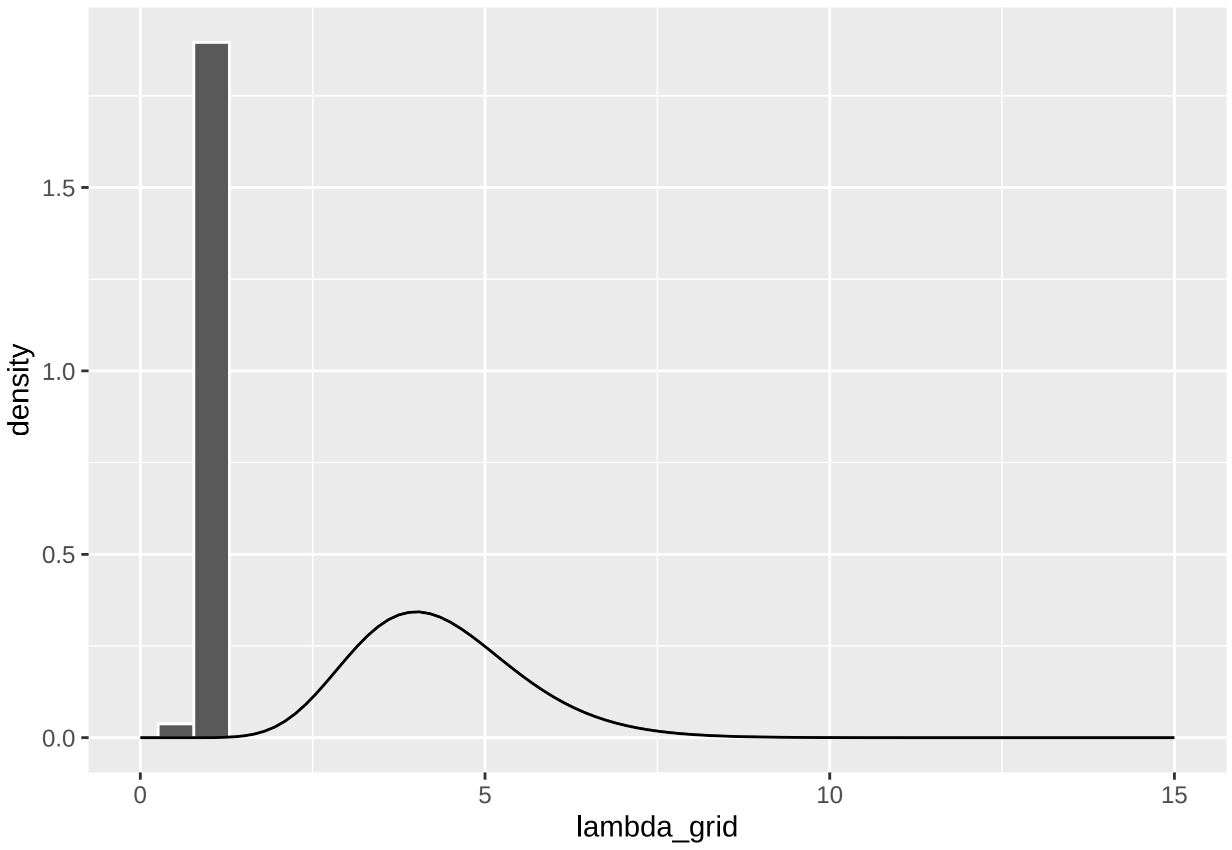
```r
s <- 20
r <- 5

# Step 1: Define a grid of 501 lambda values
grid_data   <- data.frame(
  lambda_grid = seq(from = 0, to = 1, length = 201))


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

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

```
# 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(13, 3)) +
  lims(x = c(0, 15))
```

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

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



Exercise 6.7

a. Approximating the posterior model of lambda for lambda in $\{5, 6, 7, \ldots 15\}$

```
# Step 1: Define a grid of 6 pi values
grid_data <- data.frame(mu_grid = seq(from = 5, to = 15, length = 11))
```

```
obsv_mean <- mean(c(7.1, 8.9, 8.4, 8.6))
obsv_sigma <- 1.69
n <- 4
##could use below code in for loop to calculate likelihoods for all mu's
#likelihood <- e((obsv_mean - mu)^(2)/(2*obsv_sigma^(2)/n))

# Step 2: Evaluate the prior & likelihood at each lambda
grid_data <- grid_data %>%
  mutate(prior = dnorm(10, 1.44),
      likelihood = (dnorm(7.1, mean=mu_grid, sd=1.69)*dnorm(8.9, mean=mu_grid, sd=1.69)*dnorm(8.4, mean=
```

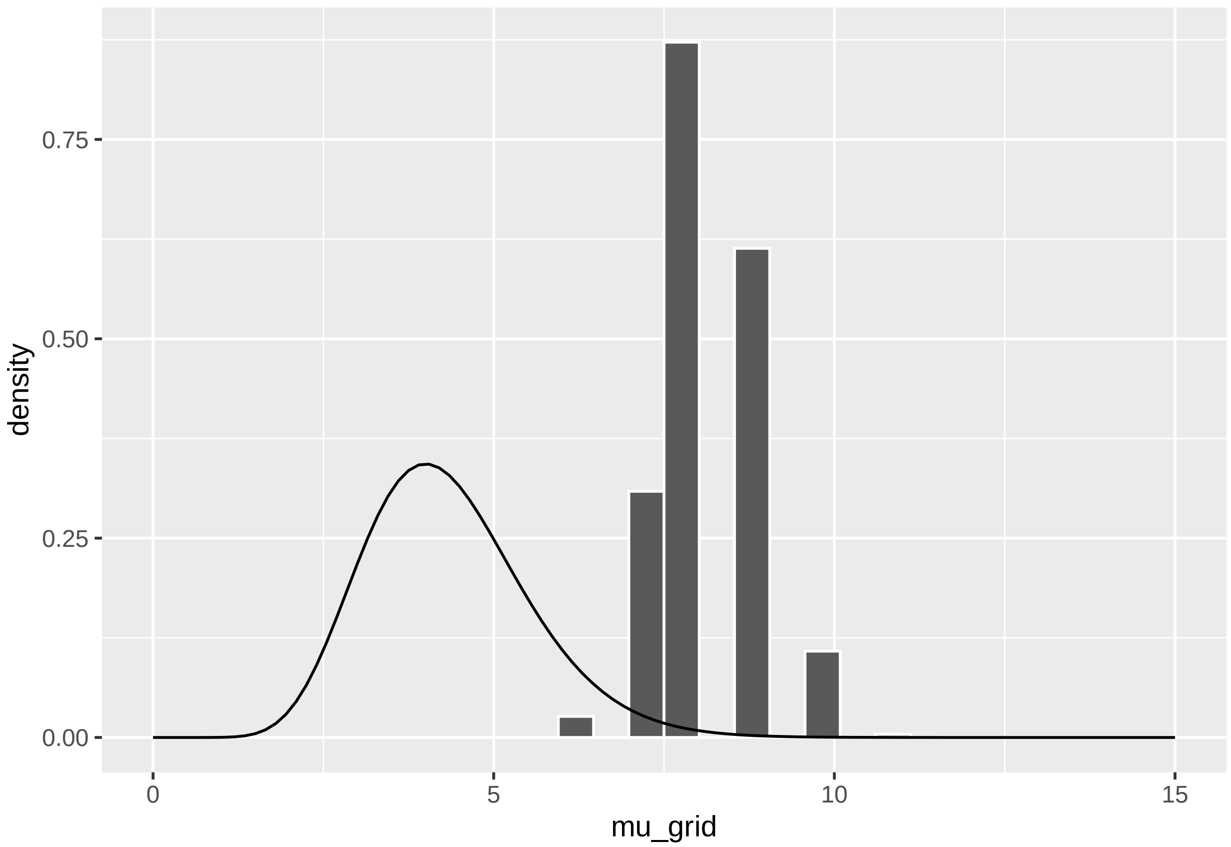
```r
# Step 3: Approximate the posterior
grid_data <- grid_data %>%
  mutate(unnormalized = likelihood*prior,
      posterior = (likelihood*prior)/sum(unnormalized))

# 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 posterior model below:

```r
# Histogram of the grid simulation with posterior pdf
ggplot(post_sample, aes(x = mu_grid)) +
  geom_histogram(aes(y = ..density..), color = "white") +
  stat_function(fun = dgamma, args = list(13, 3)) +
  lims(x = c(0, 15))
```

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

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



Exercise 6.8

a. Describe a situation in which we would want to have inference for multiple parameters (i.e., high-dimensional Bayesian models).

Elaborating on the example from the book, say we wanted to see how concussions varied in terms of the volume of the hippocampus *and* the length of headaches. This would mean we would have variation across the x and y axes of our model.

b. In your own words, explain how dimensionality can affect grid approximation and why this is a curse.

When we add mulitple parameters, we end up slicing the grid into not only vertical slices (when we discretize the x axis), but also horizontal slices (when we discretize the y axis). Therefore, our approximations become more and more fine, meaning there are bigger gaps between each approximated data point. This means that the overall approximation becomes increasingly discretized and less continuous. As we add more parameters, this trend grows.

Exercise 6.9

a. What drawback(s) do MCMC and grid approximation share? They both suffer from the limitations of sample size; the smaller the dataset of observed variables, the less reliably the approximation mimics the true posterior.
b. What advantage(s) do MCMC and grid approximation share? As our Bayesian models become increasingly complex, they will be impossible to specify precisely. These approximations will give us a sense of the posterior model even in cases when we cannot define it precisely.
c. What is an advantage of grid approximation over MCMC? Grid approximation allows us to estimate the posterior across a finite set of pi values, which allows us to limit computational complexity by only producing posterior values for a discrete set of inputs. This allows us to gain a sense of the overall trend the model demonstrates without requiring the computational overhead that is required to estimate the posterior for all values of pi on a continuous scale.
d. What is an advantage of MCMC over grid approximation? MCMC allows for approximation of more complex Bayesian models with more parameters. Grid approximation is effective with one parameter, but as we add more and more parameters to our models, grid approximation becomes increasingly computationally expensive.

Exercise 6.10

a. Yes, because the probability of eating at a restaurant on day i is dependent upon the probability that you ate out the day before, and plan to eat out the day after.
b. No, because the probability that you win one day is independent from the probability that you win on another.
c. Yes, because the probability that you win one day informs the probability that you could win again (e.g., if you are improving in your chess skill, or getting better at knowing your roommate's moves, then you might trend towards more wins, in which case a win on one day would imply a greater likelihood of a win the next day).

Exercise 6.11

a. The RSTAN syntax is as follows:

```
# 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. The RSTAN syntax is as follows:

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

c. The RSTAN syntax is as follows:

```
# STEP 1: DEFINE the model
gp_model <- "
  data {
    int<lower = 0> Y;
  }
  parameters {
    real<lower = 0> mu;
  }
  model {
    Y ~ normal(mu, 1);
    mu ~ normal(0, 100);
  }
"
```

Exercise 6.12

a. The RSTAN syntax is as follows:

```
# 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);
  }
"
```

```r
# STEP 2: SIMULATE the posterior
bb_sim <- stan(model_code = bb_model, data = list(Y = 12),
               chains = 4, iter = 5000*2, seed = 84735)
```

```
## 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 '79757aaadd38925ba0ee929aa4f540cd' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.7e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.27 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: 0.067299 seconds (Warm-up)
## Chain 1:                0.071514 seconds (Sampling)
## Chain 1:                0.138813 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '79757aaadd38925ba0ee929aa4f540cd' NOW (CHAIN 2).
```

```
## Chain 2:
## Chain 2: Gradient evaluation took 9e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.09 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: 0.067831 seconds (Warm-up)
## Chain 2:                0.070756 seconds (Sampling)
## Chain 2:                0.138587 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '79757aaadd38925ba0ee929aa4f540cd' 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: 0.066094 seconds (Warm-up)
## Chain 3:                0.064586 seconds (Sampling)
## Chain 3:                0.13068 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '79757aaadd38925ba0ee929aa4f540cd' 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: 0.067012 seconds (Warm-up)
## Chain 4:                0.06787 seconds (Sampling)
## Chain 4:                0.134882 seconds (Total)
## Chain 4:
```

b. The RSTAN syntax is as follows:

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

# STEP 2: SIMULATE the posterior
bb_sim <- stan(model_code = bb_model, data = list(Y = 3),
               chains = 4, iter = 5000*2, seed = 84735)
```

```
##
## SAMPLING FOR MODEL '79757aaadd38925ba0ee929aa4f540cd' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.1 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: 0.080757 seconds (Warm-up)
## Chain 1:                0.079759 seconds (Sampling)
## Chain 1:                0.160516 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '79757aaadd38925ba0ee929aa4f540cd' 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: 0.079209 seconds (Warm-up)
## Chain 2:                0.074004 seconds (Sampling)
## Chain 2:                0.153213 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '79757aaadd38925ba0ee929aa4f540cd' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.1 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: 0.07272 seconds (Warm-up)
## Chain 3:                0.077529 seconds (Sampling)
## Chain 3:                0.150249 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '79757aaadd38925ba0ee929aa4f540cd' 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: 0.068919 seconds (Warm-up)
## Chain 4:                0.08252 seconds (Sampling)
## Chain 4:                0.151439 seconds (Total)
## Chain 4:
```

c. The RSTAN syntax is as follows:

```
# STEP 1: DEFINE the model
gp_model <- "
  data {
    int<lower = 0> Y;
  }
  parameters {
    real<lower = 0> mu;
  }
  model {
    Y ~ normal(mu, 1);
    mu ~ normal(0, 100);
  }
"

# STEP 2: SIMULATE the posterior
bb_sim <- stan(model_code = bb_model, data = list(Y = 12.2),
               chains = 4, iter = 5000*2, seed = 84735)
```

```
## Error in mod$fit_ptr() :
##   Exception: int variable contained non-int values; processing stage=data initialization; variable na
```

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

Exercise 6.13

a. Building and simulating model

```
# 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(3, 8);
  }
"
# STEP 2: SIMULATE the posterior
bb_sim <- stan(model_code = bb_model, data = list(Y = 2),
               chains = 3, iter = 12000, seed = 84735)
```

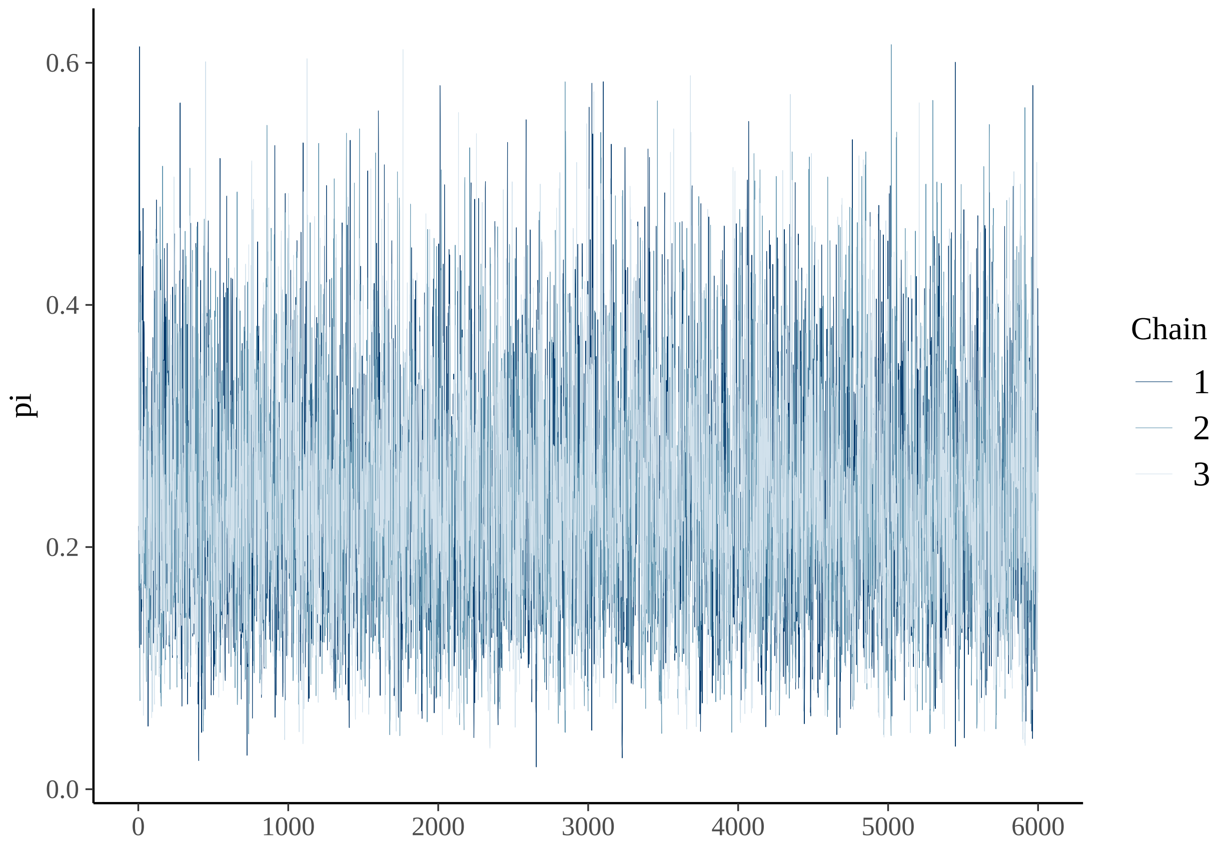
```
## Trying to compile a simple C file
```

```
## 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: f
## #include <complex>
##          ^~~~~~~~~
## 3 errors generated.
## make: *** [foo.o] Error 1
##
## SAMPLING FOR MODEL '6cc1e5f794ef111dd783f9c98fed9c41' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.5e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.25 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: 0.083822 seconds (Warm-up)
## Chain 1:                0.096783 seconds (Sampling)
## Chain 1:                0.180605 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '6cc1e5f794ef111dd783f9c98fed9c41' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 9e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.09 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: 0.096243 seconds (Warm-up)
## Chain 2:                0.112077 seconds (Sampling)
## Chain 2:                0.20832 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '6cc1e5f794ef111dd783f9c98fed9c41' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.1 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: 0.087003 seconds (Warm-up)
## Chain 3:                0.083375 seconds (Sampling)
## Chain 3:                0.170378 seconds (Total)
## Chain 3:
```

b. Building trace plots of model

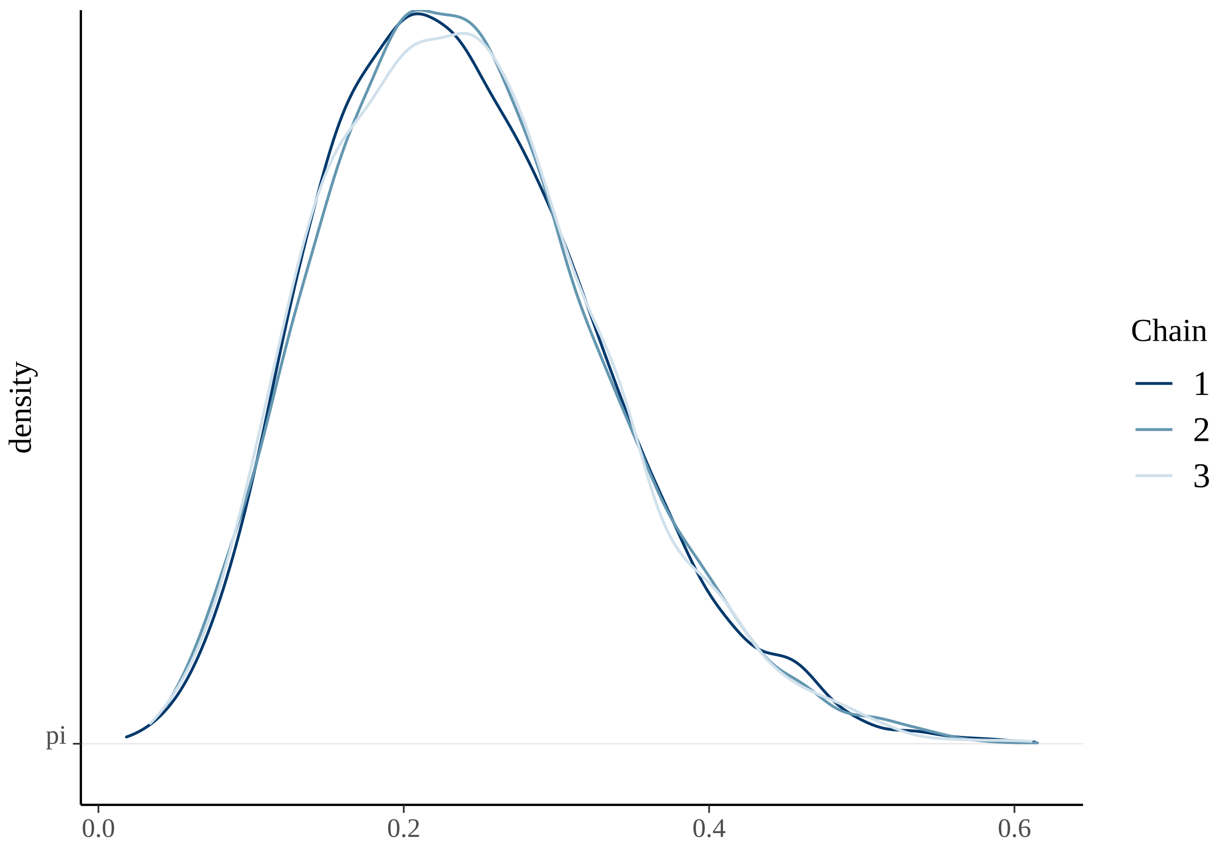
```
mcmc_trace(bb_sim, pars = "pi", size = 0.1)
```



c. The range of values on the trace plot x axis is 0 to 6000. The maximum value in this range is not 12000 because the half of the trials are thrown out; therefore, 6000 trials is the maximum.

d. The density plot functions for each of the three chains are as follows:

```
mcmc_dens_chains(bb_sim, pars = "pi") +
  yaxis_text(TRUE) +
  ylab("density")
```



e. The posterior model for this example is as follows:

$$Beta(5, 16)$$

The mean estimated by the posterior (0.3125) is slightly greater than that produced by the MCMC approximations (which hovers closer to 0.2).

Exercise 6.14

a. Building and simulating model

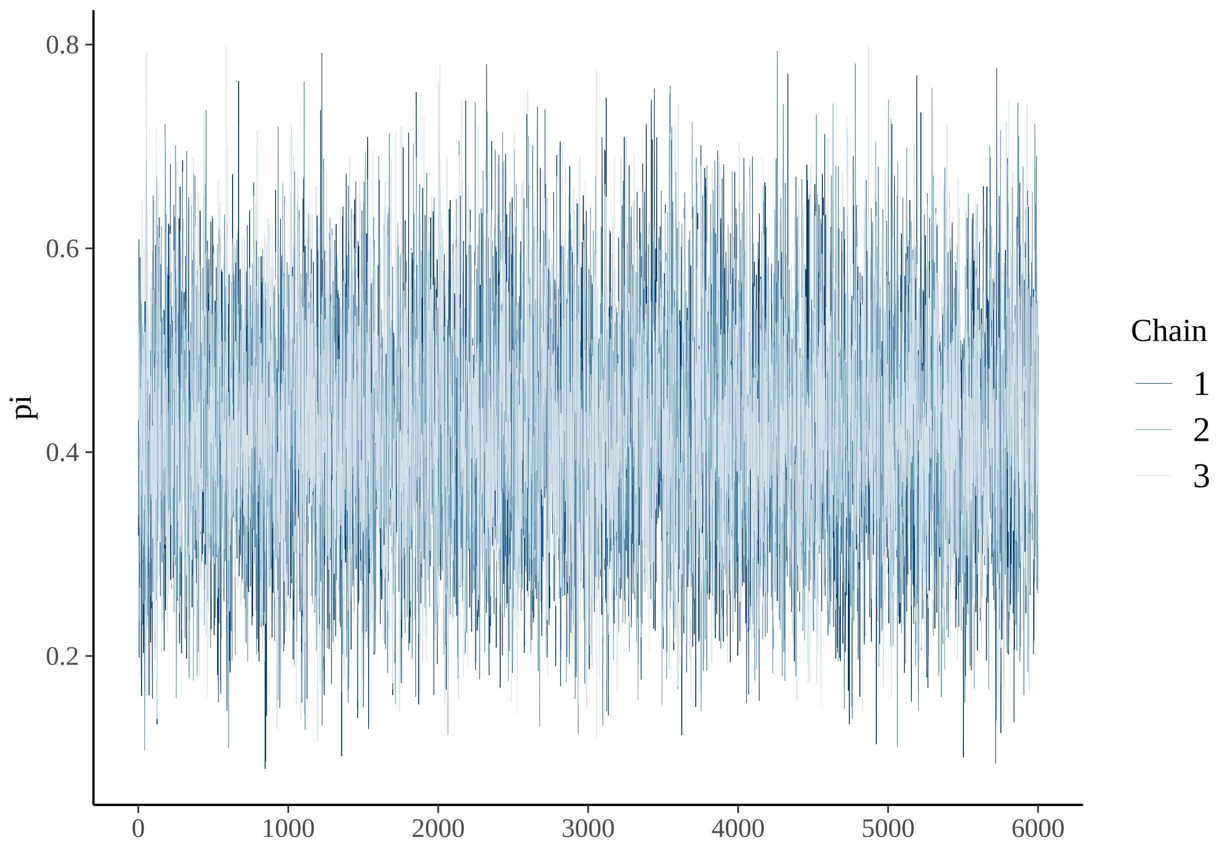
```
# STEP 1: DEFINE the model
bb_model <- "
  data {
    int<lower = 0, upper = 12> Y;
  }
  parameters {
    real<lower = 0, upper = 1> pi;
  }
  model {
    Y ~ binomial(12, pi);
    pi ~ beta(4, 3);
  }
"
#error somewhere in installation
# STEP 2: SIMULATE the posterior
bb_sim <- stan(model_code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

```
## 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: f
## #include <complex>
##          ^~~~~~~~~
## 3 errors generated.
## make: *** [foo.o] Error 1
##
## SAMPLING FOR MODEL '981ca33f2a8a2fb571d6627286410841' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.3e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.23 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: 0.082833 seconds (Warm-up)
## Chain 1:                0.083106 seconds (Sampling)
## Chain 1:                0.165939 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '981ca33f2a8a2fb571d6627286410841' 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 / 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: 0.083991 seconds (Warm-up)
## Chain 2:                0.093269 seconds (Sampling)
## Chain 2:                0.17726 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '981ca33f2a8a2fb571d6627286410841' 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 / 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: 0.084908 seconds (Warm-up)
## Chain 3:                0.084731 seconds (Sampling)
## Chain 3:                0.169639 seconds (Total)
## Chain 3:
```

b. Building trace plots of model

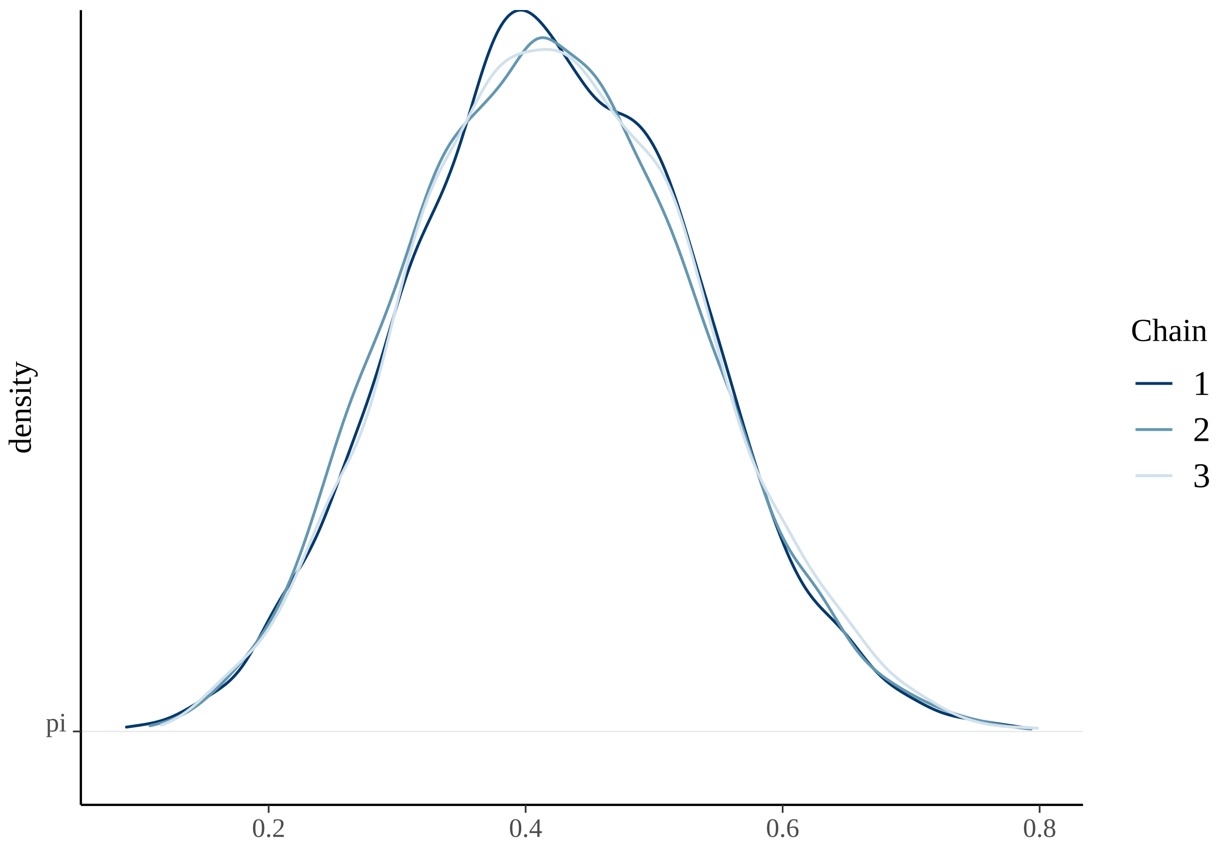
```
mcmc_trace(bb_sim, pars = "pi", size = 0.1)
```



c. The range of values on the trace plot x axis is 0 to 6000. The maximum value in this range is not 12000 because the half of the trials are thrown out; therefore, 6000 trials is the maximum.

d. The density plot functions for each of the three chains are as follows:

```r
mcmc_dens_chains(bb_sim, pars = "pi") +
  yaxis_text(TRUE) +
  ylab("density")
```



e. The posterior model for this example is as follows:

$$Beta(8, 11)$$

The mean estimated by the posterior (0.727) is slightly greater than that produced by the MCMC approximations (which hovers closer to 0.55).

Exercise 6.15

a. Building and simulating the model:

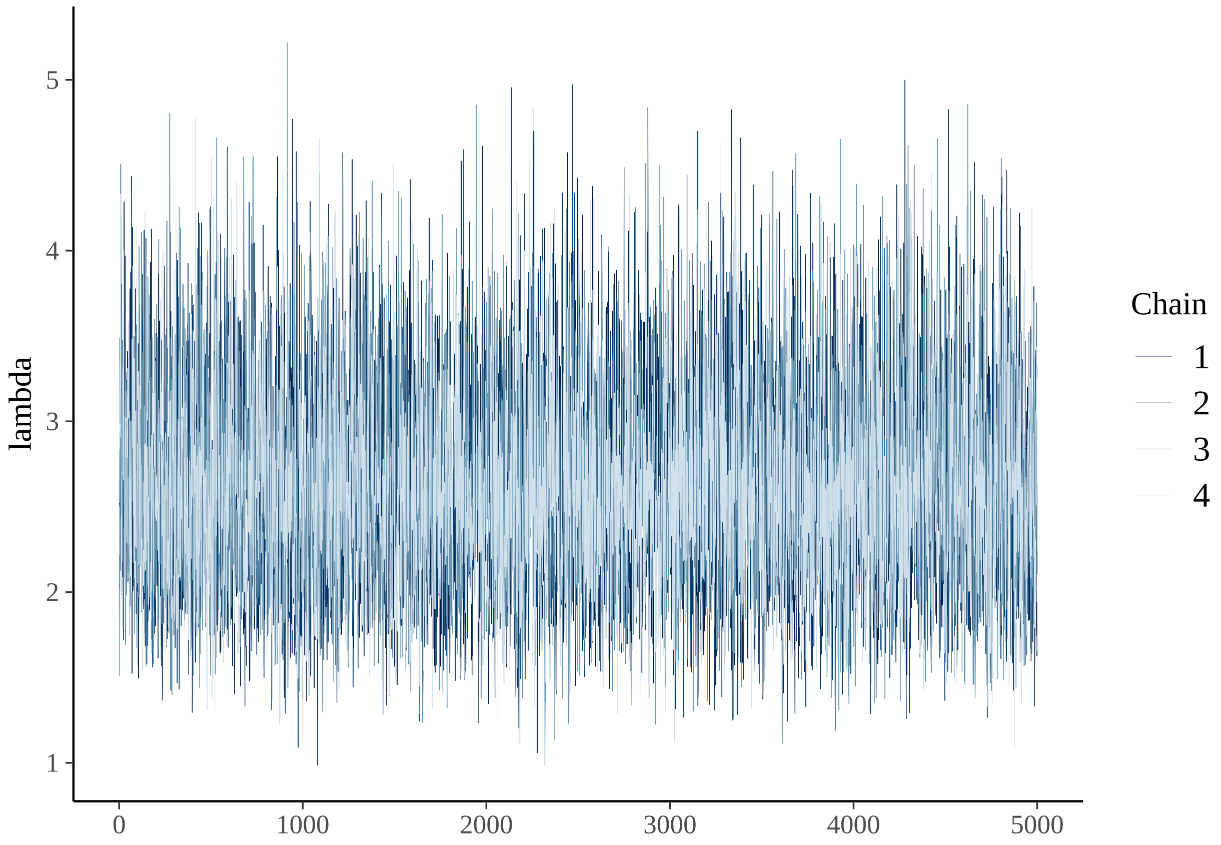
```
# STEP 1: DEFINE the model
gp_model <- "
  data {
    int<lower = 0> Y[3];
  }
  parameters {
    real<lower= 0> lambda;
  }
  model {
    Y ~ poisson(lambda);
    lambda ~ gamma(20, 5);
  }
"

# STEP 2: SIMULATE the posterior
gp_sim <- stan(model_code = gp_model, data = list(Y = c(0, 1, 0)),
               chains = 4, iter = 10000, seed = 84735)
```

```
## 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: f
## #include <complex>
##          ^~~~~~~~~
## 3 errors generated.
## make: *** [foo.o] Error 1
##
## SAMPLING FOR MODEL 'fc8d96254c1b5d3853c4e538836a1c66' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.9e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.19 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: 0.068542 seconds (Warm-up)
## Chain 1:                0.066594 seconds (Sampling)
## Chain 1:                0.135136 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'fc8d96254c1b5d3853c4e538836a1c66' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.1 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: 0.068202 seconds (Warm-up)
## Chain 2:                0.071925 seconds (Sampling)
## Chain 2:                0.140127 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'fc8d96254c1b5d3853c4e538836a1c66' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 7e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.07 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: 0.069137 seconds (Warm-up)
## Chain 3:                0.067757 seconds (Sampling)
## Chain 3:                0.136894 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'fc8d96254c1b5d3853c4e538836a1c66' 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: 0.067749 seconds (Warm-up)
## Chain 4:                0.070502 seconds (Sampling)
## Chain 4:                0.138251 seconds (Total)
## Chain 4:
```

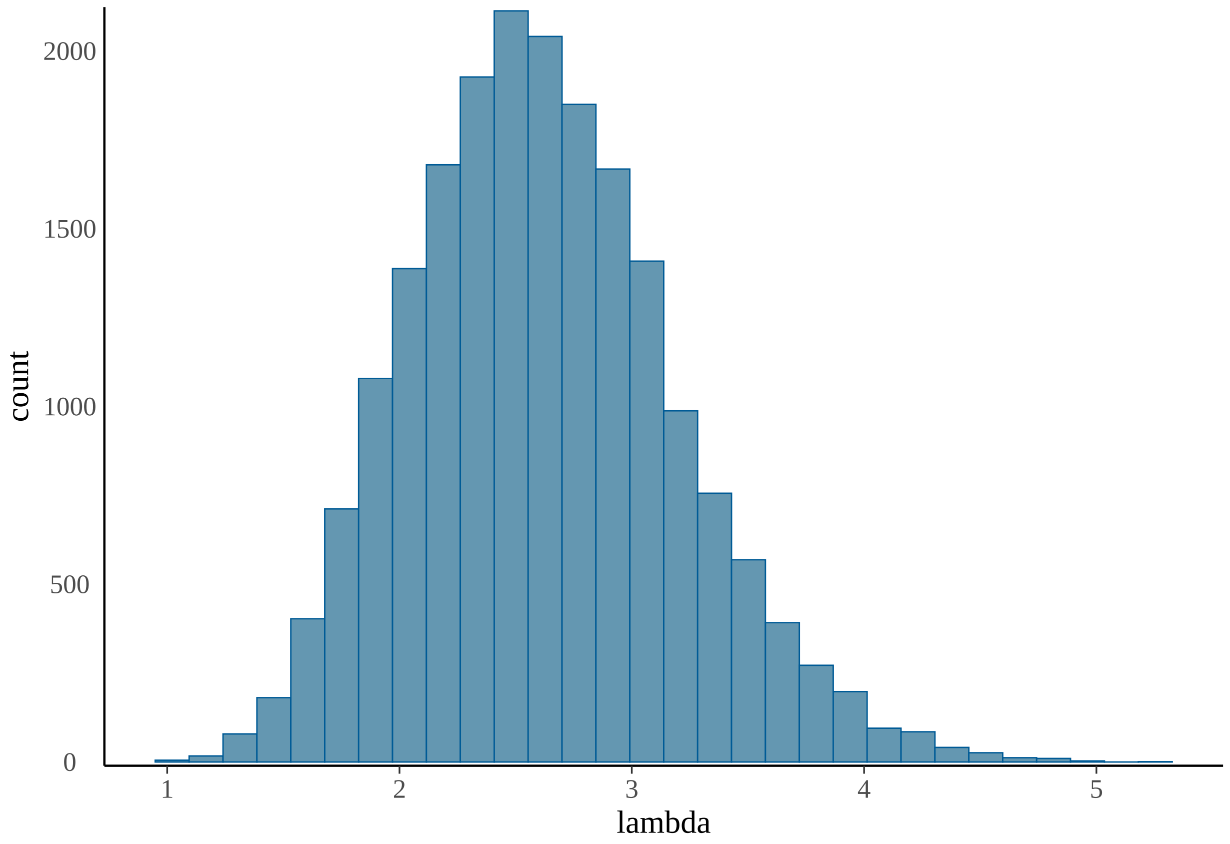
b. Building trace plots and density plots

```
# Trace plots of the 4 Markov chains
mcmc_trace(gp_sim, pars = "lambda", size = 0.1)
```

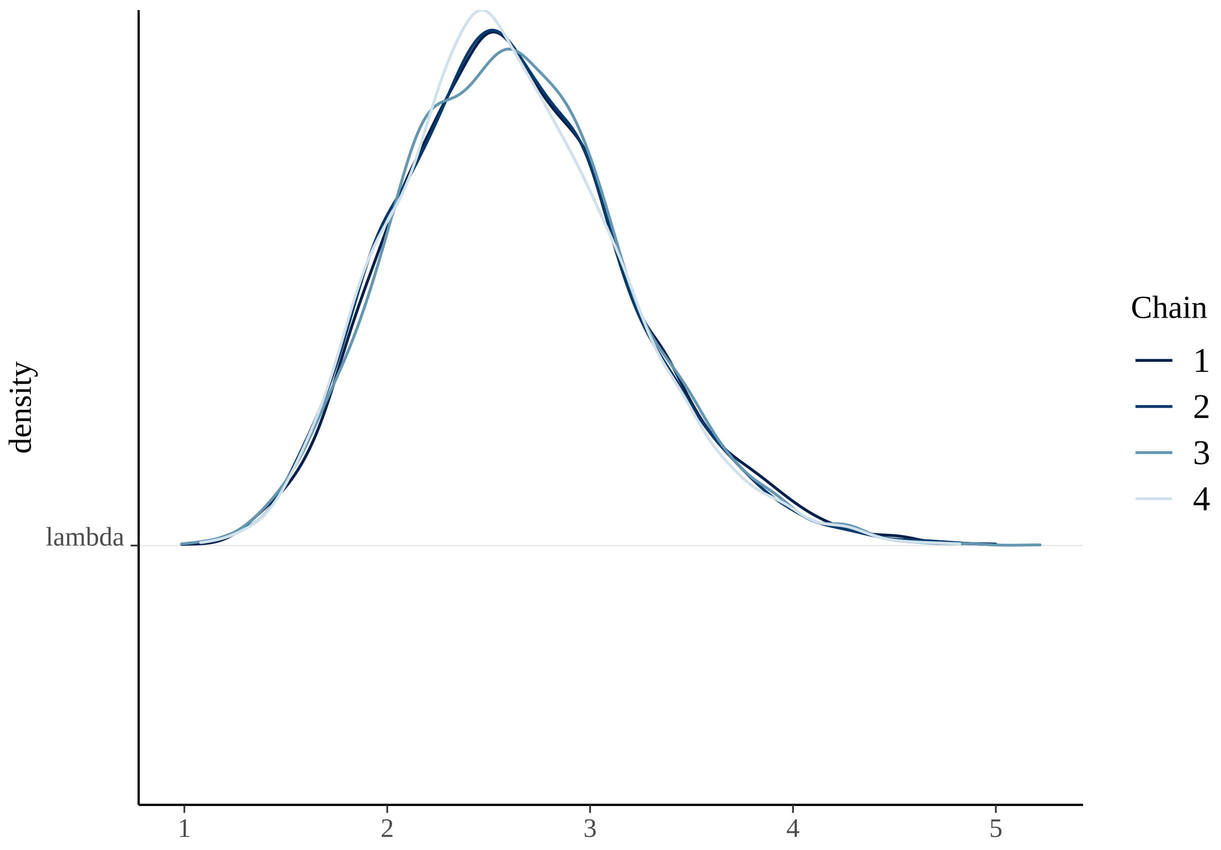


```
# Histogram of the Markov chain values
mcmc_hist(gp_sim, pars = "lambda") +
  yaxis_text(TRUE) +
  ylab("count")
```

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



```
# Density plot of the Markov chain values
mcmc_dens_chains(gp_sim, pars = "lambda") +
  yaxis_text(TRUE) +
  ylab("density")
```



c. Based on the density plots, the most plausible posterior value of lambda seems to be around 2.5.

d. The posterior model is as follows:

$$Gamma(21, 8)$$

The mean lambda produced by the specified model is 21/8==2.625. This is slightly higher than our MCMC approximated average for lambda (2.5ish).

Exercise 6.16

a. Building and simulating the model:

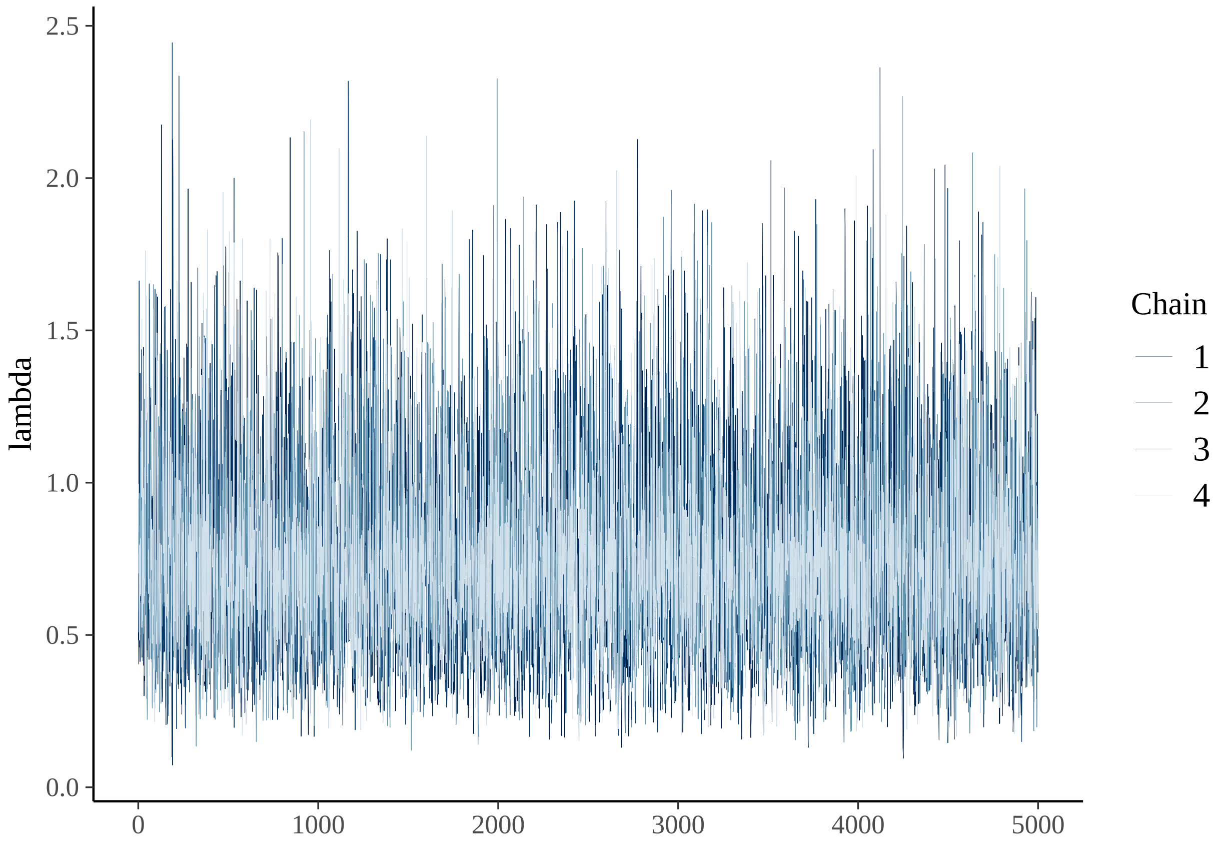
```
# STEP 1: DEFINE the model
gp_model <- "
  data {
    int<lower = 0> Y[3];
  }
  parameters {
    real<lower= 0> lambda;
  }
  model {
    Y ~ poisson(lambda);
    lambda ~ gamma(5, 5);
  }
"

# STEP 2: SIMULATE the posterior
gp_sim <- stan(model_code = gp_model, data = list(Y = c(0, 1, 0)),
               chains = 4, iter = 10000, seed = 84735)
```

```
## 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 '97765253d425f33463791f17d863ec5c' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.2 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: 0.072662 seconds (Warm-up)
## Chain 1:                0.070869 seconds (Sampling)
## Chain 1:                0.143531 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '97765253d425f33463791f17d863ec5c' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 7e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.07 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: 0.074488 seconds (Warm-up)
## Chain 2:                0.078412 seconds (Sampling)
## Chain 2:                0.1529 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '97765253d425f33463791f17d863ec5c' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 6e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.06 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: 0.073039 seconds (Warm-up)
## Chain 3:                0.070131 seconds (Sampling)
## Chain 3:                0.14317 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '97765253d425f33463791f17d863ec5c' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 7e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.07 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: 0.069417 seconds (Warm-up)
## Chain 4:                0.078049 seconds (Sampling)
## Chain 4:                0.147466 seconds (Total)
## Chain 4:
```

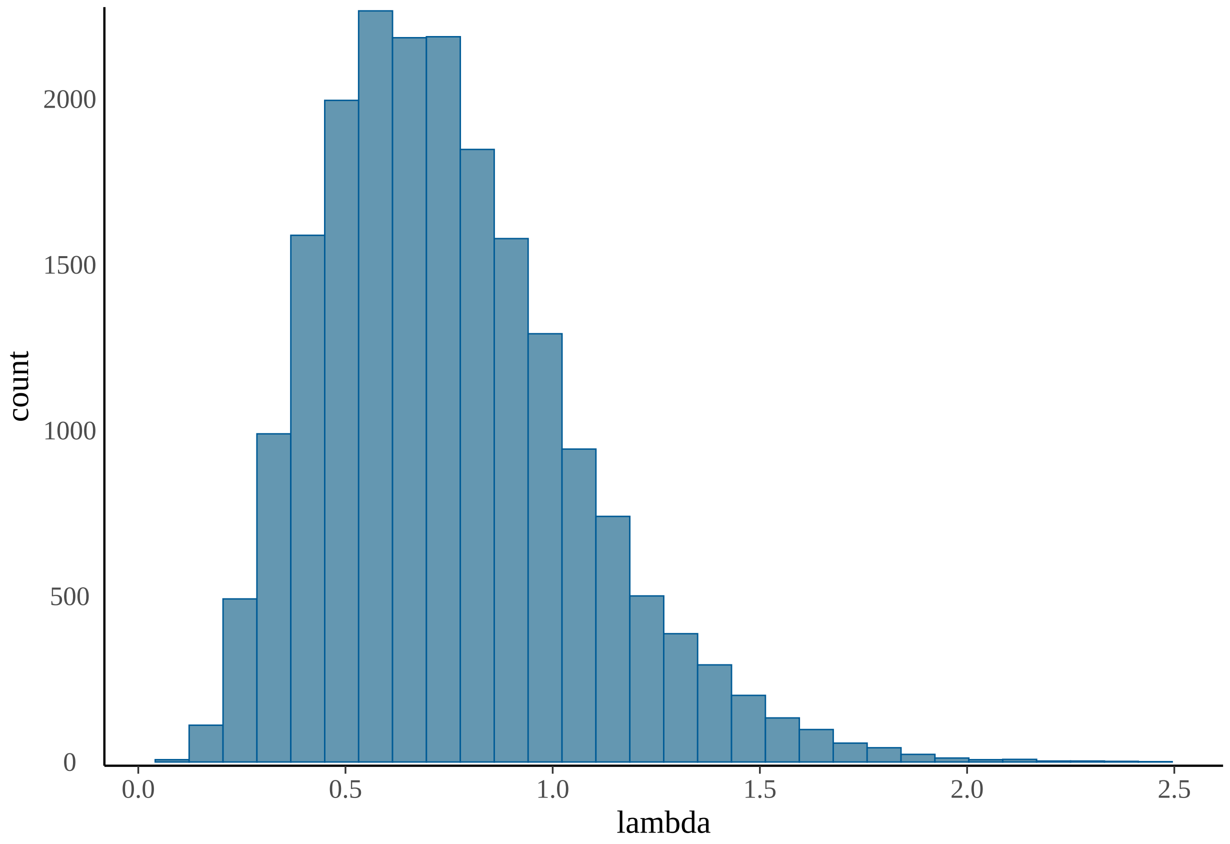
b. Building trace plots and density plots

```
# Trace plots of the 4 Markov chains
mcmc_trace(gp_sim, pars = "lambda", size = 0.1)
```

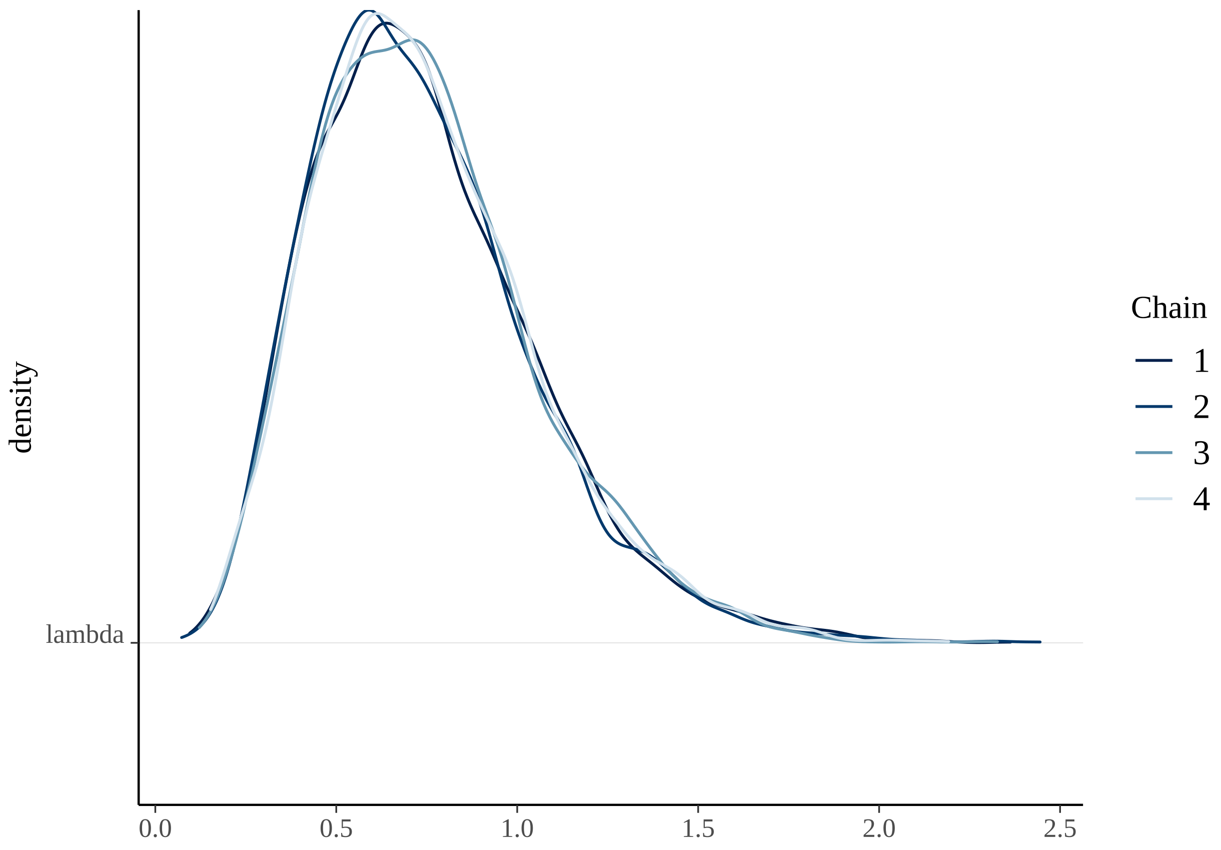


```
# Histogram of the Markov chain values
mcmc_hist(gp_sim, pars = "lambda") +
  yaxis_text(TRUE) +
  ylab("count")
```

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



```r
# Density plot of the Markov chain values
mcmc_dens_chains(gp_sim, pars = "lambda") +
  yaxis_text(TRUE) +
  ylab("density")
```



c. Based on the density plots, the most plausible posterior value of lambda seems to be around 0.6.

d. The posterior model is as follows:

$$Gamma(6, 8)$$

The mean lambda produced by the specified model is 6/8==0.75. This is higher than our MCMC approximated average for lambda.

Exercise 6.17

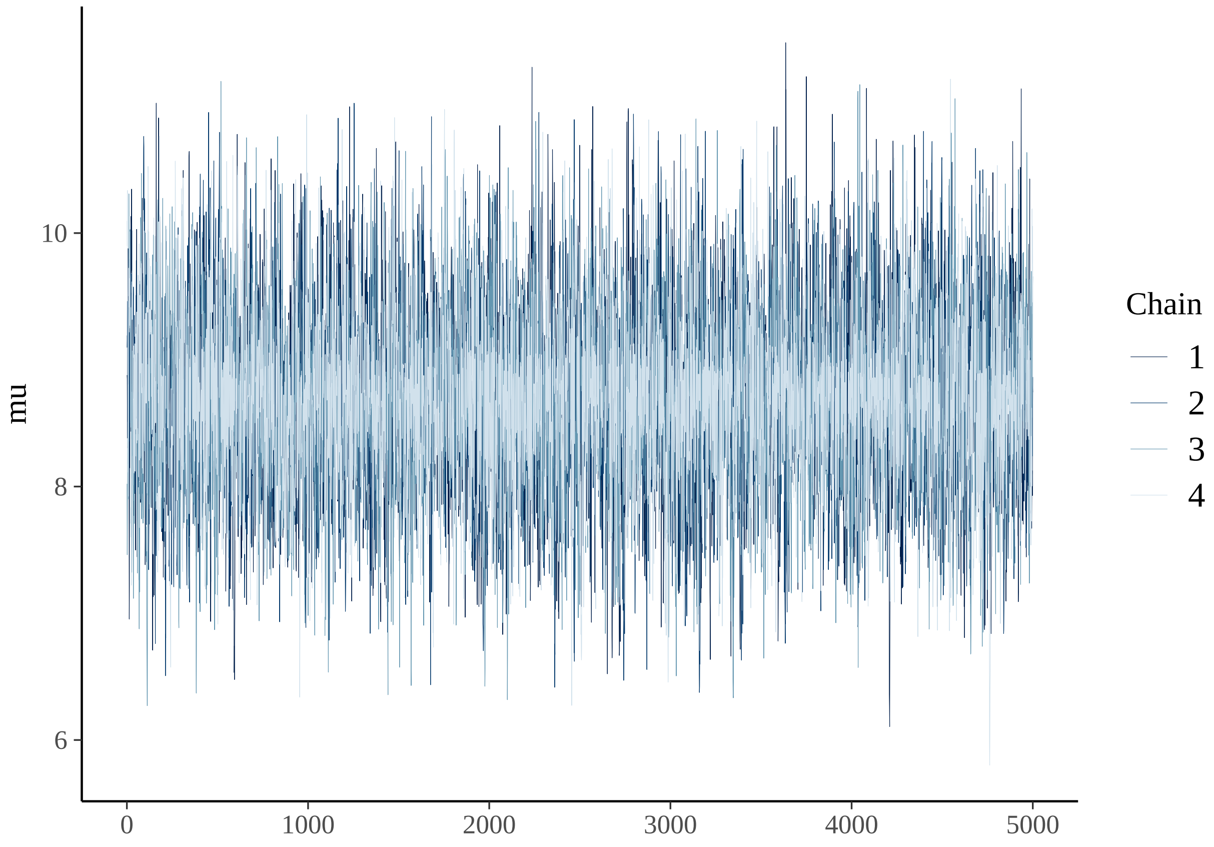
a. Building and simulating the model:

```r
# STEP 1: DEFINE the model
gp_model <- "
  data {
    real<lower = 0> Y[4];
  }
  parameters {
    real<lower= 0> mu;
  }
  model {
    Y ~ normal(mu, 1.69);
    mu ~ normal(10, 1.44);
  }
"

# STEP 2: SIMULATE the posterior
gp_sim <- stan(model_code = gp_model, data = list(Y = c(7.1, 8.9, 8.4, 8.6)), chains = 4, iter = 10000,
## 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 '3186728533482810a4aa478480e0d887' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 6e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.6 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: 0.068578 seconds (Warm-up)
## Chain 1:                0.07612 seconds (Sampling)
## Chain 1:                0.144698 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '3186728533482810a4aa478480e0d887' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 6e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.06 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: 0.068685 seconds (Warm-up)
## Chain 2:                0.08591 seconds (Sampling)
## Chain 2:                0.154595 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '3186728533482810a4aa478480e0d887' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 8e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.08 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: 0.069776 seconds (Warm-up)
## Chain 3:                0.075854 seconds (Sampling)
## Chain 3:                0.14563 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '3186728533482810a4aa478480e0d887' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 9e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.09 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 10000 [  0%]  (Warmup)
## Chain 4: Iteration: 1000 / 10000 [ 10%]  (Warmup)
## Chain 4: Iteration: 2000 / 10000 [ 20%]  (Warmup)
## Chain 4: Iteration: 3000 / 10000 [ 30%]  (Warmup)
## Chain 4: Iteration: 4000 / 10000 [ 40%]  (Warmup)
## Chain 4: Iteration: 5000 / 10000 [ 50%]  (Warmup)
## Chain 4: Iteration: 5001 / 10000 [ 50%]  (Sampling)
## Chain 4: Iteration: 6000 / 10000 [ 60%]  (Sampling)
## Chain 4: Iteration: 7000 / 10000 [ 70%]  (Sampling)
## Chain 4: Iteration: 8000 / 10000 [ 80%]  (Sampling)
## Chain 4: Iteration: 9000 / 10000 [ 90%]  (Sampling)
## Chain 4: Iteration: 10000 / 10000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.083333 seconds (Warm-up)
## Chain 4:                0.092898 seconds (Sampling)
## Chain 4:                0.176231 seconds (Total)
## Chain 4:
```

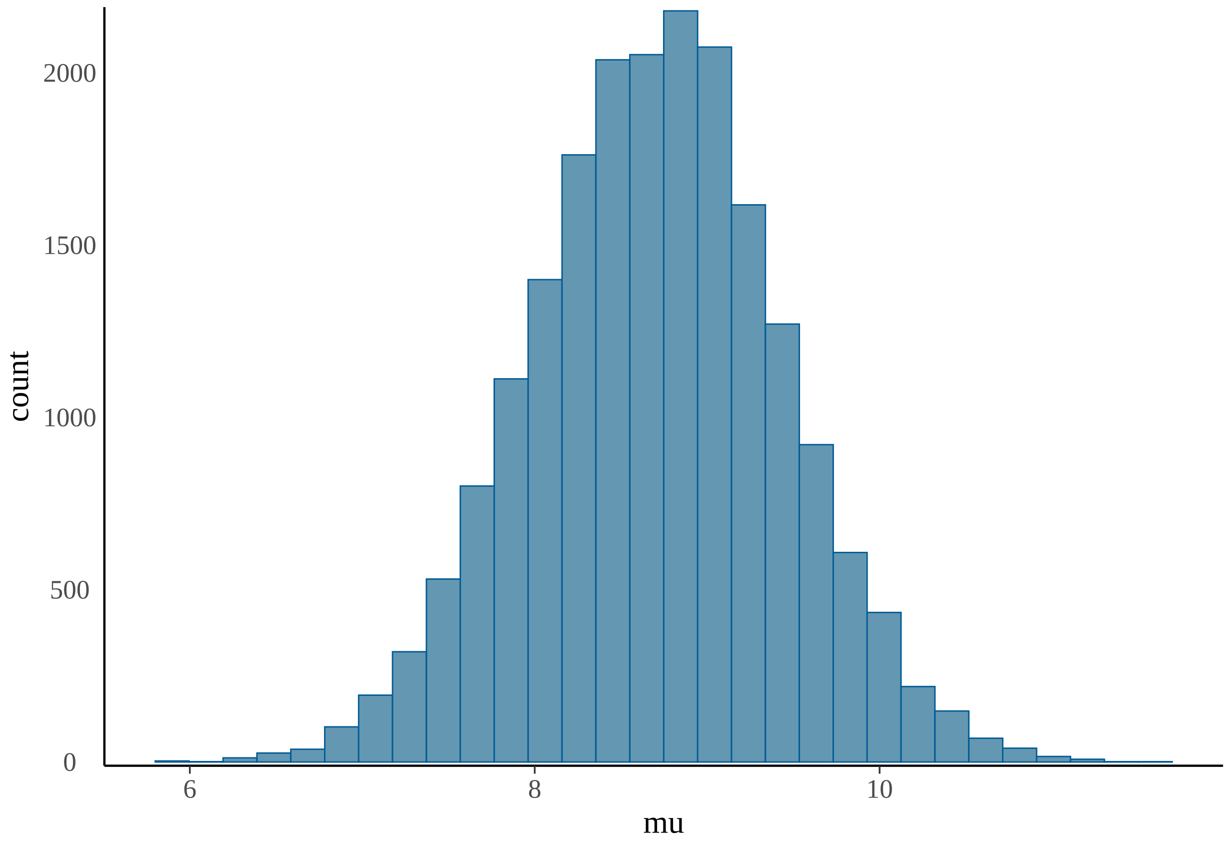
b. Building trace plots and density plots

```
# Trace plots of the 4 Markov chains
mcmc_trace(gp_sim, pars = "mu", size = 0.1)
```

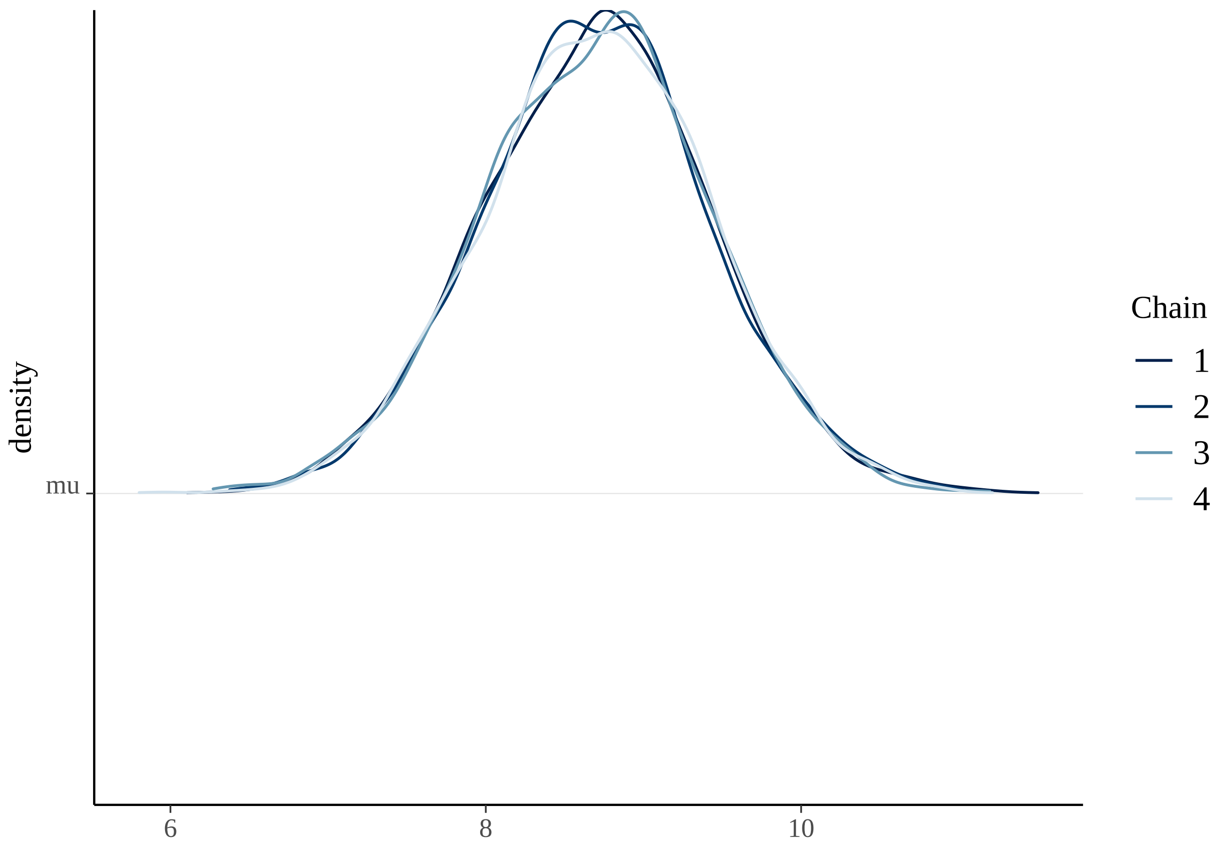


```
# Histogram of the Markov chain values
mcmc_hist(gp_sim, pars = "mu") +
  yaxis_text(TRUE) +
  ylab("count")
```

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



```r
# Density plot of the Markov chain values
mcmc_dens_chains(gp_sim, pars = "mu") +
  yaxis_text(TRUE) +
  ylab("density")
```



c. Based on the density plots, the most plausible posterior value of mu seems to be around 9.

d. The posterior model is as follows:

```
prior_mean <- 10
prior_sd <- 1.44
obsv_mean <- mean(c(7.1, 8.9, 8.4, 8.6))
obsv_sd <- 1.69
n <- 4
posterior_mu <- prior_mean*obsv_sd/(n*prior_sd+obsv_sd)+obsv_mean * ((n*prior_sd)/(n*prior_sd+obsv_sd))
posterior_sigma <- (prior_sd*obsv_sd)/(n*prior_sd+obsv_sd)

posterior_mu
```

```
## [1] 8.64698
```

```
posterior_sigma
```

```
## [1] 0.3266577
```

$$Normal(8.69, 0.53)$$

The mean mu produced by the specified model is 8.69. This is slightly than our MCMC approximated average of 9.

Exercise 6.18

a. Building and simulating the model:

```
# STEP 1: DEFINE the model
gp_model <- "
  data {
    real<lower = -20> Y[5];
  }
  parameters {
    real<lower= -20> mu;
  }
  model {
    Y ~ normal(mu, 64);
    mu ~ normal(-14, 4);
  }
"

# STEP 2: SIMULATE the posterior
gp_sim <- stan(model_code = gp_model, data = list(Y = c(-10.1, 5.5, 0.1, -1.4, 11.5)), chains = 4, iter
```

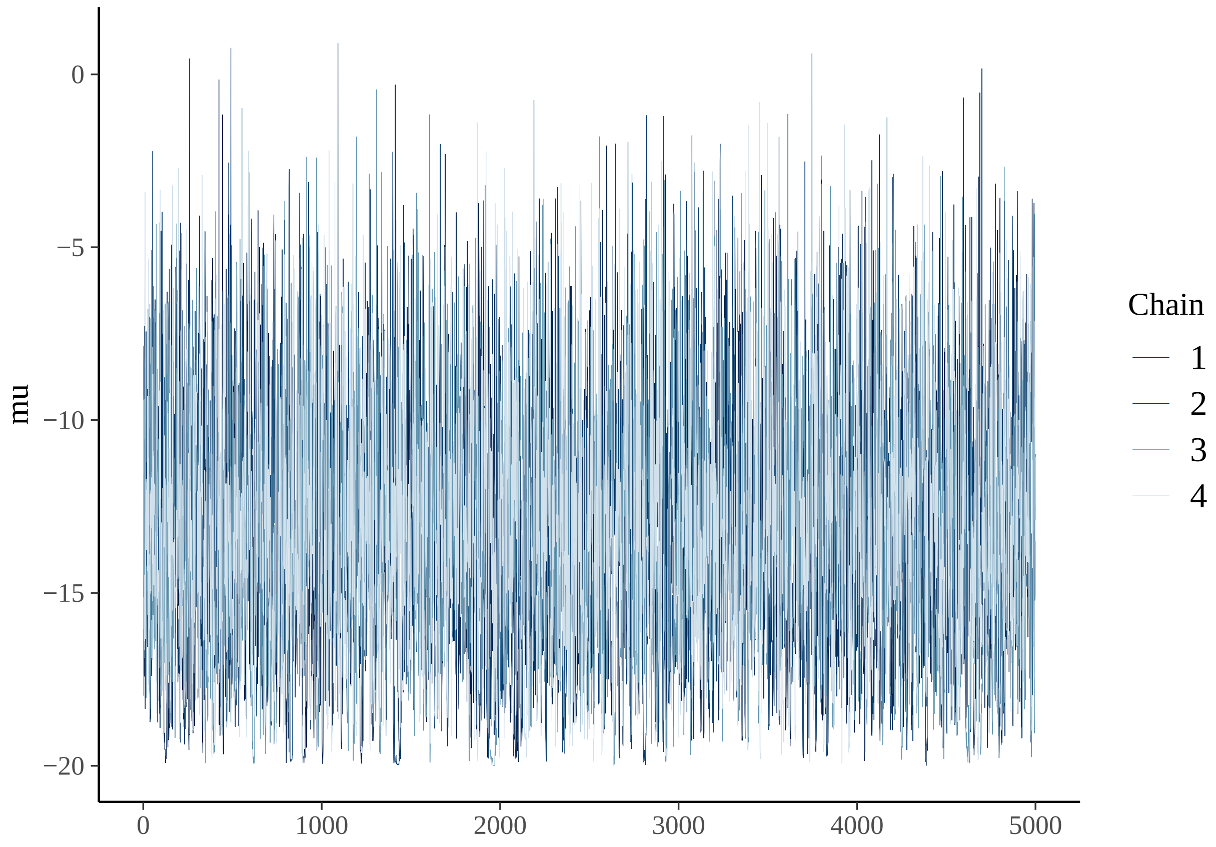
```
## Trying to compile a simple C file
```

```
## 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: f
## #include <complex>
##          ^~~~~~~~~
## 3 errors generated.
## make: *** [foo.o] Error 1
##
## SAMPLING FOR MODEL 'f1f467c2aafeeaa24ae37fde3928fda8' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.3e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.23 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: 0.081631 seconds (Warm-up)
## Chain 1:                0.083205 seconds (Sampling)
## Chain 1:                0.164836 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'f1f467c2aafeeaa24ae37fde3928fda8' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 7e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.07 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: 0.078069 seconds (Warm-up)
## Chain 2:                0.10879 seconds (Sampling)
## Chain 2:                0.186859 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'f1f467c2aafeeaa24ae37fde3928fda8' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 7e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.07 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: 0.076199 seconds (Warm-up)
## Chain 3:                0.07794 seconds (Sampling)
## Chain 3:                0.154139 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'f1f467c2aafeeaa24ae37fde3928fda8' 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: 0.076945 seconds (Warm-up)
## Chain 4:                0.088663 seconds (Sampling)
## Chain 4:                0.165608 seconds (Total)
## Chain 4:
```

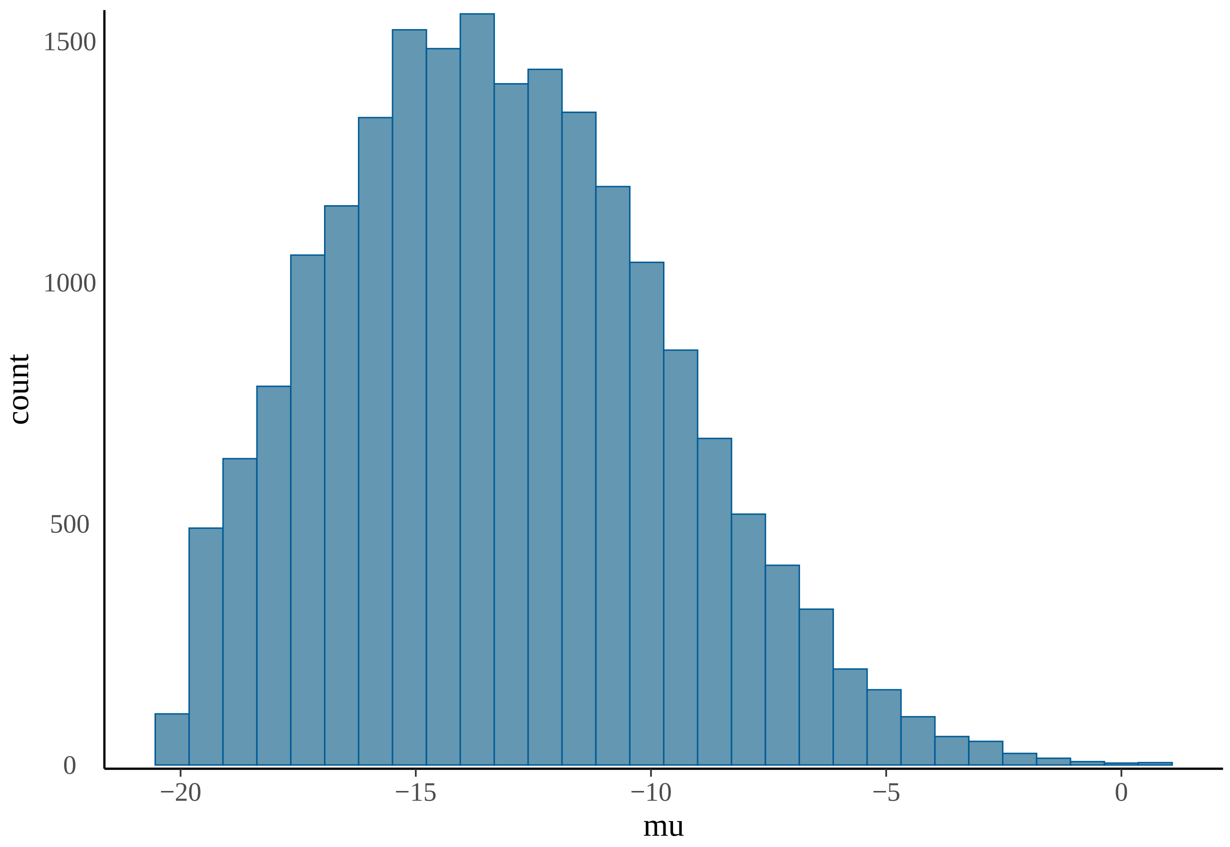
b. Building trace plots and density plots

```r
# Trace plots of the 4 Markov chains
mcmc_trace(gp_sim, pars = "mu", size = 0.1)
```

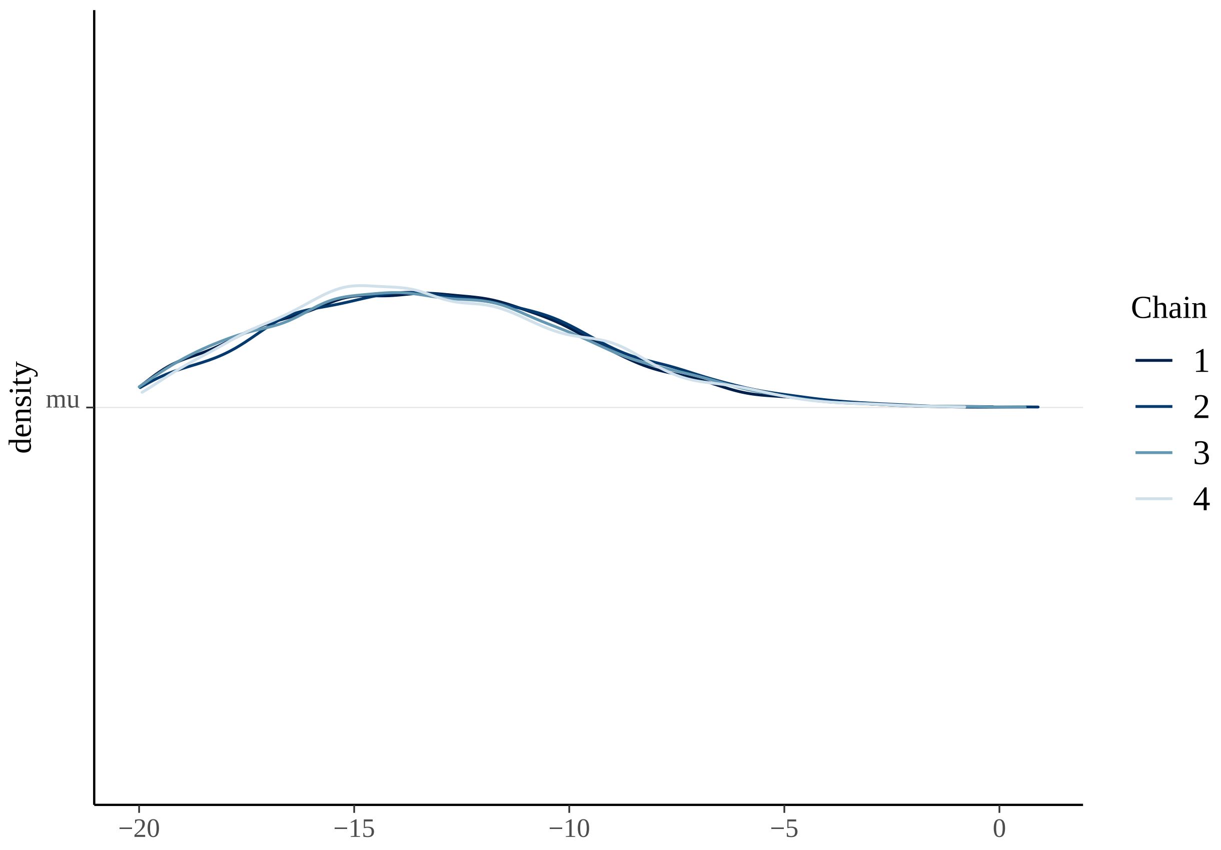


```r
# Histogram of the Markov chain values
mcmc_hist(gp_sim, pars = "mu") +
  yaxis_text(TRUE) +
  ylab("count")
```

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



```r
# Density plot of the Markov chain values
mcmc_dens_chains(gp_sim, pars = "mu") +
  yaxis_text(TRUE) +
  ylab("density")
```



c. Based on the density plots, the most plausible posterior value of mu seems to be around -13.

d. The posterior model is as follows:

```
prior_mean <- -14
prior_sd <- 4
obsv_mean <- mean(c(-10.1, 5.5, 0.1, -1.4, 11.5))
obsv_sd <- 64
n <- 5
posterior_mu <- prior_mean*obsv_sd^2/(n*prior_sd^2+obsv_sd^2)+obsv_mean * ((n*prior_sd^2)/(n*prior_sd^2
posterior_sigma <- (prior_sd^2*obsv_sd^2)/(n*prior_sd^2+obsv_sd^2)

posterior_mu
```

```
## [1] -13.71034
```

```
posterior_sigma
```

```
## [1] 15.69349
```

$$Normal(-13.71, 15.69)$$

```
summarize_normal_normal(mean=-14, sd=4, sigma=64, y_bar=c(-10.1, 5.5, 0.1, -1.4, 11.5), n = 5)
```

```
##       model      mean      mode      var       sd
## 1     prior -14.00000 -14.00000 16.00000 4.000000
## 2 posterior -13.92529 -13.92529 15.69349 3.961501
## 3     prior -13.62644 -13.62644 16.00000 4.000000
## 4 posterior -13.72989 -13.72989 15.69349 3.961501
## 5     prior -13.75862 -13.75862 16.00000 4.000000
## 6 posterior -13.51149 -13.51149 15.69349 3.961501
```

The mean mu produced by the specified model is -13.7. This is very close to our MCMC approximated average for mu (-13).