ps4_code

AUTHOR
Derek Sollberger

5.5

a

$$\mathrm{E}(\lambda) = \frac{s}{r} = 5 \text{ and } \mathrm{Var}(\lambda) = \frac{s}{r^2} = (0.25)^2 \quad \rightarrow \quad s = 400, \quad r = 80$$

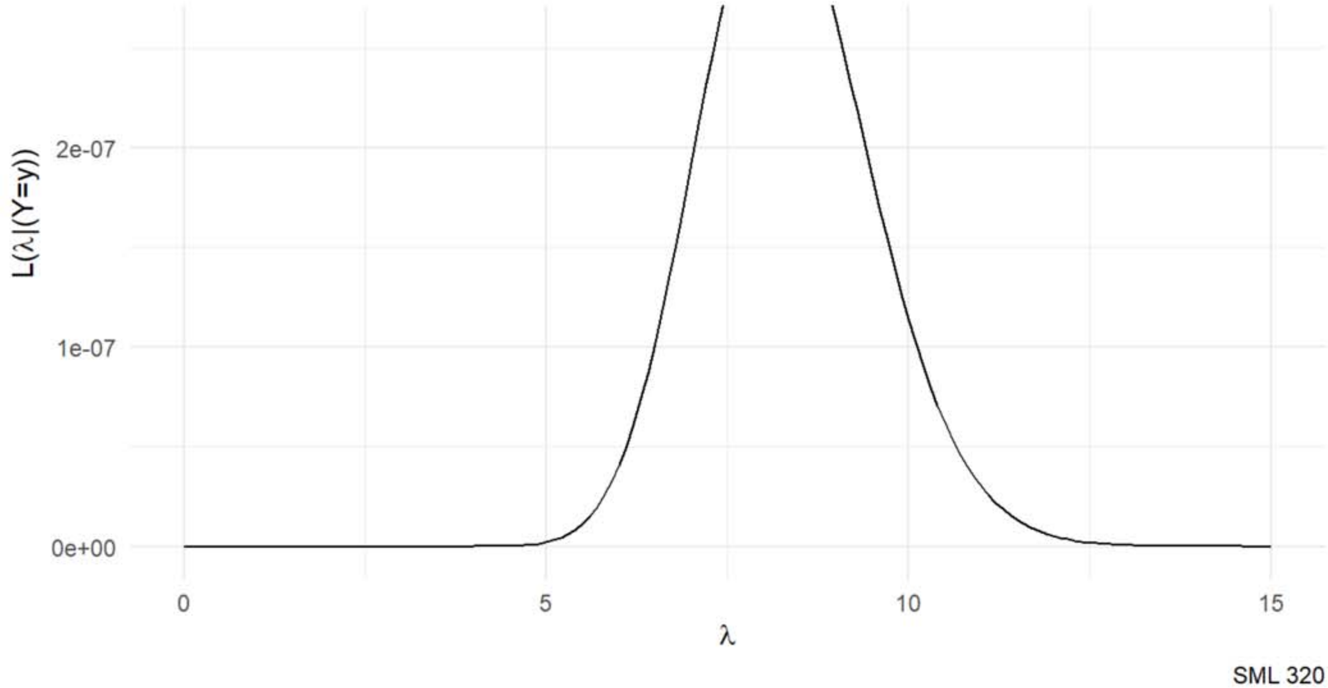
b

Since 10 is several standard deviations above the average value of 5, the prior probabliity is virtually zero.

5.6

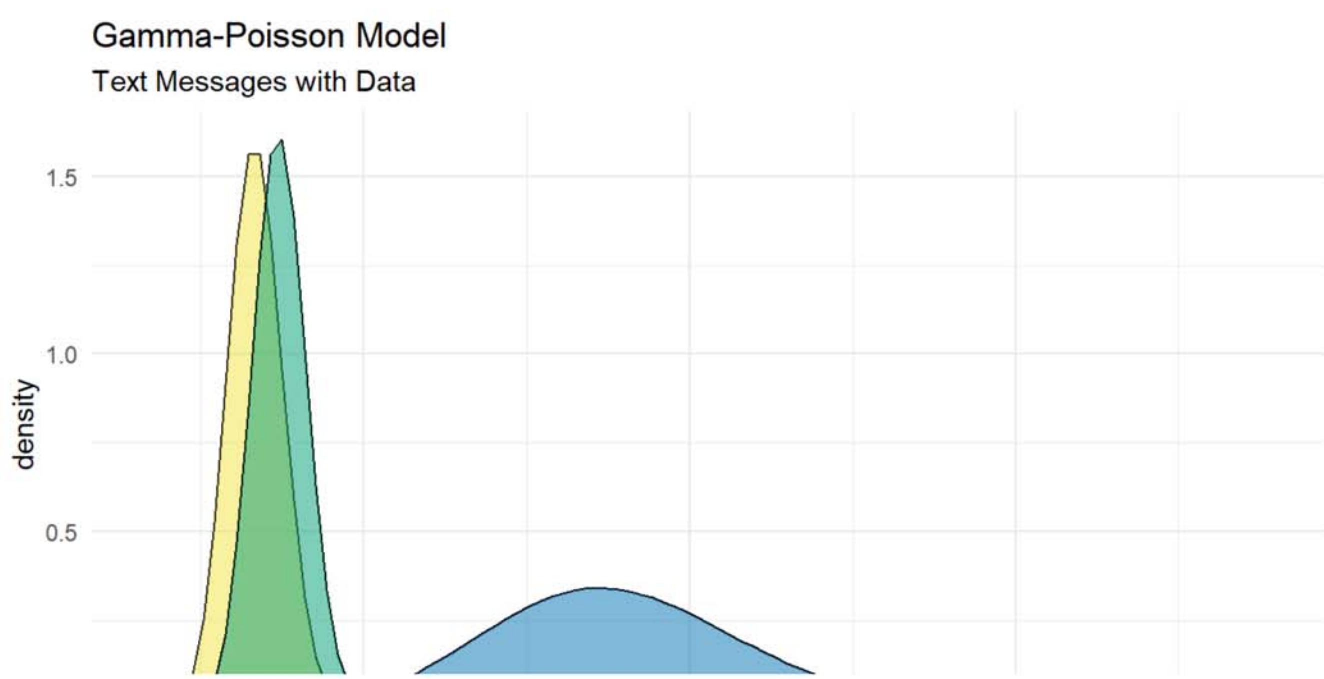
a

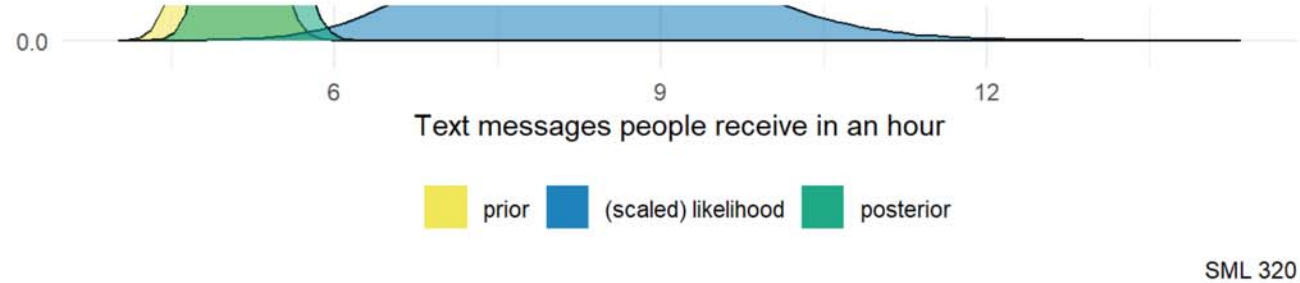
```
library("bayesrules")
library("bayesplot")
library("ggtext")
library("rstan")
library("tidyverse")

bayesrules::plot_poisson_likelihood(
  y = c(7, 3, 8, 9, 10, 12),
  lambda_upper_bound = 15
) +
  labs(title = "Likelihood Curve",
       subtitle = "Text messages people receive in an hour",
       caption = "SML 320") +
  theme_minimal()
```



b

```
bayesrules::plot_gamma_poisson(shape = 400, rate = 80,
                               sum_y = 49, n = 6) +
  labs(title = "Gamma-Poisson Model",
       subtitle = "Text Messages with Data",
       caption = "SML 320",
       x = "Text messages people receive in an hour") +
  theme_minimal() +
  theme(legend.position = "bottom")
```





c

```r
bayesrules::summarize_gamma_poisson(shape = 400, rate = 80,
                                    sum_y = 49, n = 6) |>
  mutate_if(is.numeric, round, digits = 4)
```

```
      model shape rate   mean   mode    var     sd
1     prior   400   80 5.0000 4.9875 0.0625 0.2500
2 posterior   449   86 5.2209 5.2093 0.0607 0.2464
```

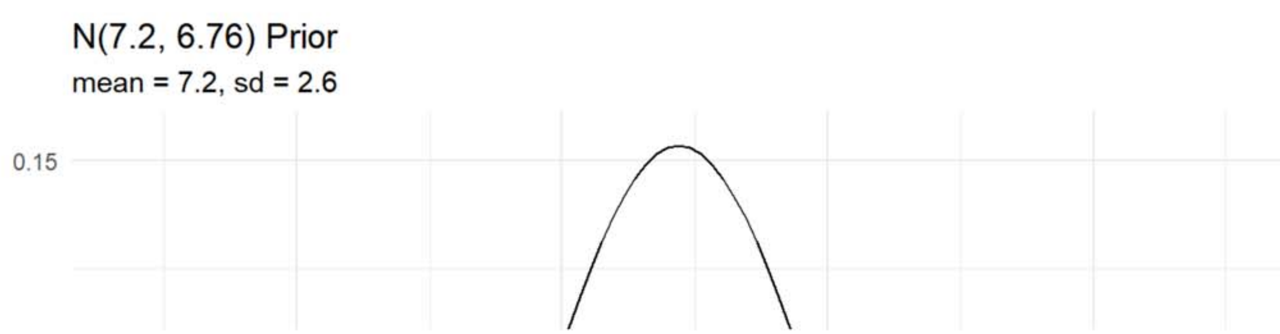
d

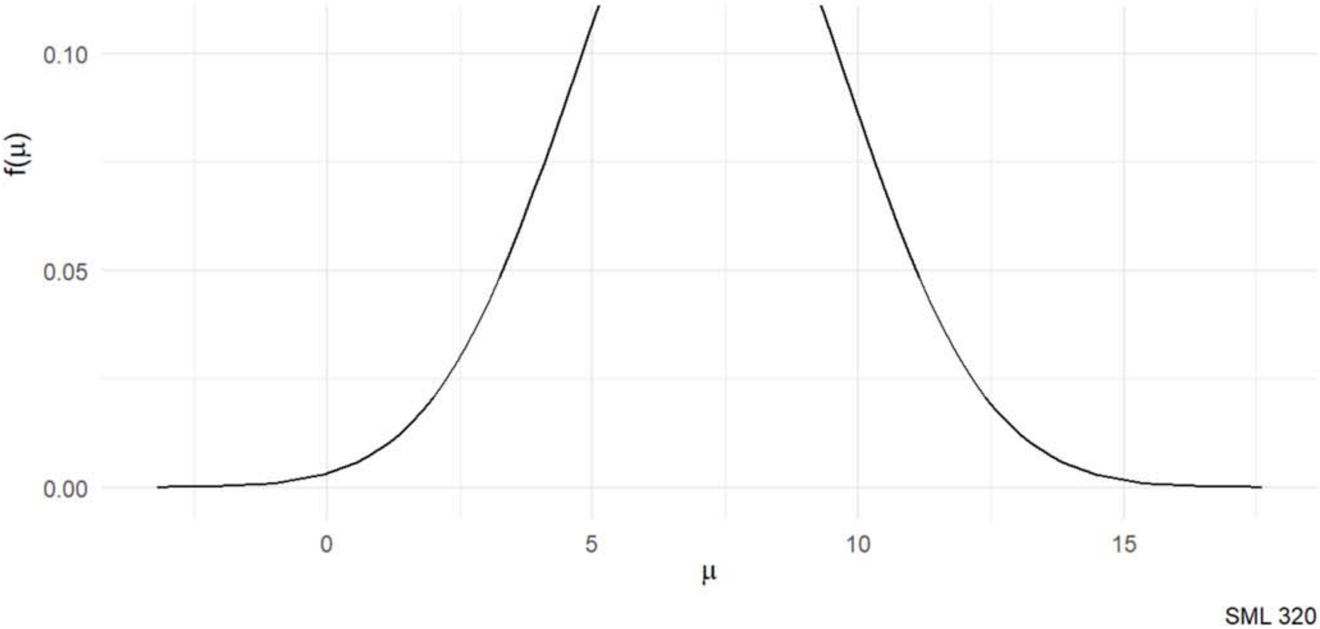
While the data from the six friends might be realistic, since we started with an overly informative prior (i.e. small variance), the application of the observed data barely changed the statistics from the prior distribution to the posterior distribution.

5.9

a

```r
bayesrules::plot_normal(mean = 7.2, sd = 2.6) +
  labs(title = "N(7.2, 6.76) Prior",
       subtitle = "mean = 7.2, sd = 2.6",
       caption = "SML 320") +
  theme_minimal()
```





b

```
pnorm(7.6, 7.2, 2.6, lower.tail = FALSE)
```

```
[1] 0.4388655
```

Yes, it seems plausible that the stock can rise by 7.6 dollars.

c

```
pnorm(4, 7.2, 2.6, lower.tail = FALSE)
```

```
[1] 0.8907954
```

Yes, it seems plausible that the stock can rise by 4 dollars.

d

```
pnorm(0, 7.2, 2.6, lower.tail = TRUE)
```

```
[1] 0.002809441
```

The prior probability of a stock price decrease is about 0.3 percent.

e

```
pnorm(8, 7.2, 2.6, lower.tail = FALSE)
```

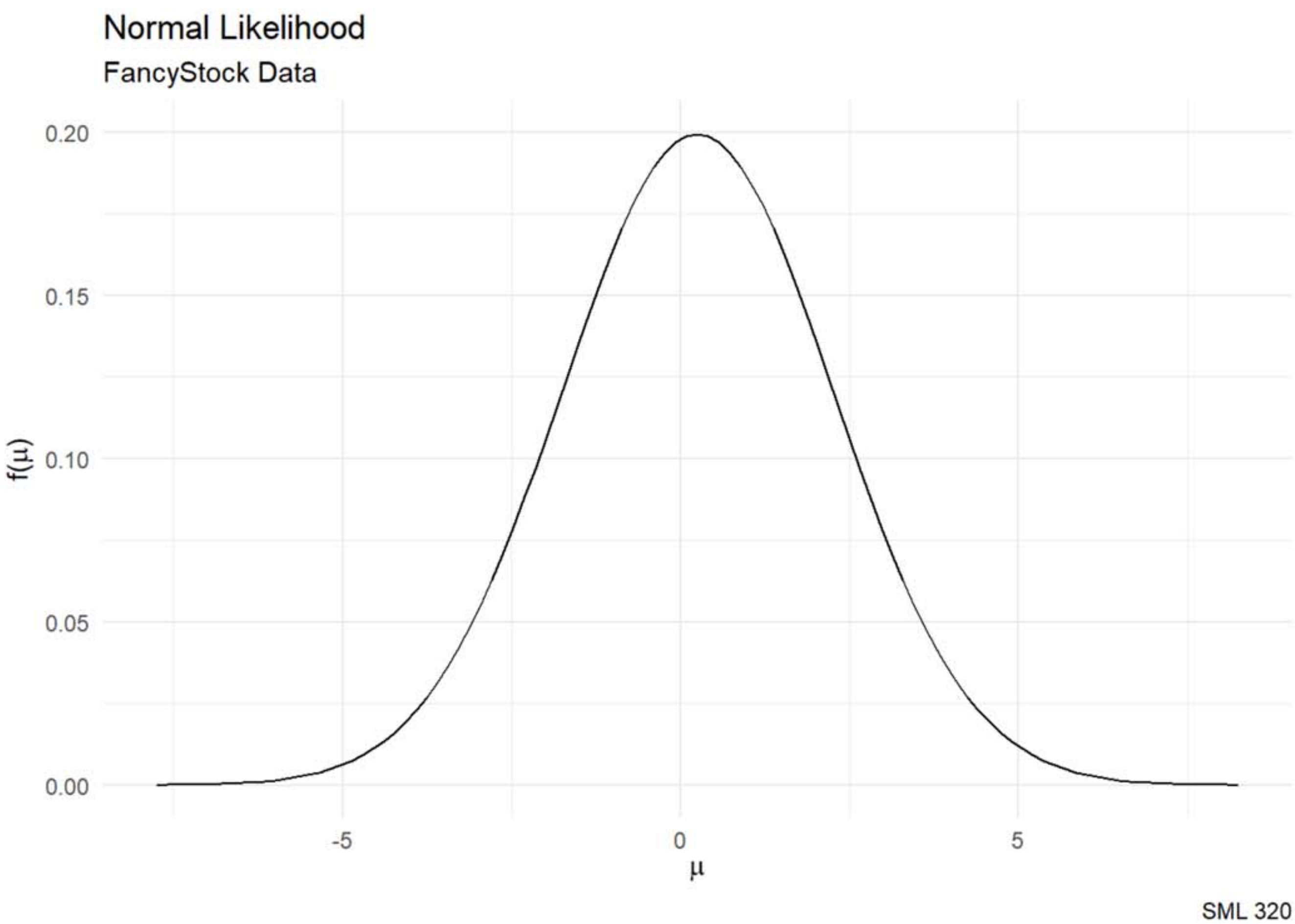
```
[1] 0.3791582
```

The prior probability that the stock rises by at least 8 dollars is about 38 percent.

5.10

a

```
obs_data <- c(-0.7, 1.2, 4.5, -4)

bayesrules::plot_normal(mean = mean(obs_data), sd = 2) +
  labs(title = "Normal Likelihood",
       subtitle = "FancyStock Data",
       caption = "SML 320") +
  theme_minimal()
```



.

b

```
bayesrules::plot_normal_normal(

  # from prior
  mean = 7.6, sd = 2.6,

  # from observations
  y_bar = mean(obs_data), sigma = 2, n = 4
) +
  labs(title = "Normal-Normal Model",
       subtitle = "Exercise 5.5b",
       caption = "SML 320",
       x = "change in stock price") +
  theme_minimal()
```

c

```
bayesrules::summarize_normal_normal(
  # from prior
  mean = 7.6, sd = 2.6,
```

```
  # from observations
  y_bar = mean(obs_data), sigma = 2, n = 4
) |>
  mutate_if(is.numeric, round, digits = 4)
```

```
      model   mean   mode    var     sd
1     prior 7.6000 7.6000 6.7600 2.6000
2 posterior 1.1972 1.1972 0.8711 0.9333
```

d

Now, in the posterior distribution, the view of the stock price change is more financially conservative with an average change of about 1.2 dollars.

e

```
pnorm(0, 1.1972, 0.9333, lower.tail = TRUE)
```

```
[1] 0.09978807
```

The posterior probability of a decrease in the stock price is about 10 percent.

f

```
pnorm(8, 1.1972, 0.9333, lower.tail = FALSE)
```

```
[1] 1.561615e-13
```

The posterior probability of the stock price increasing by over 8 dollars is virtually zero.

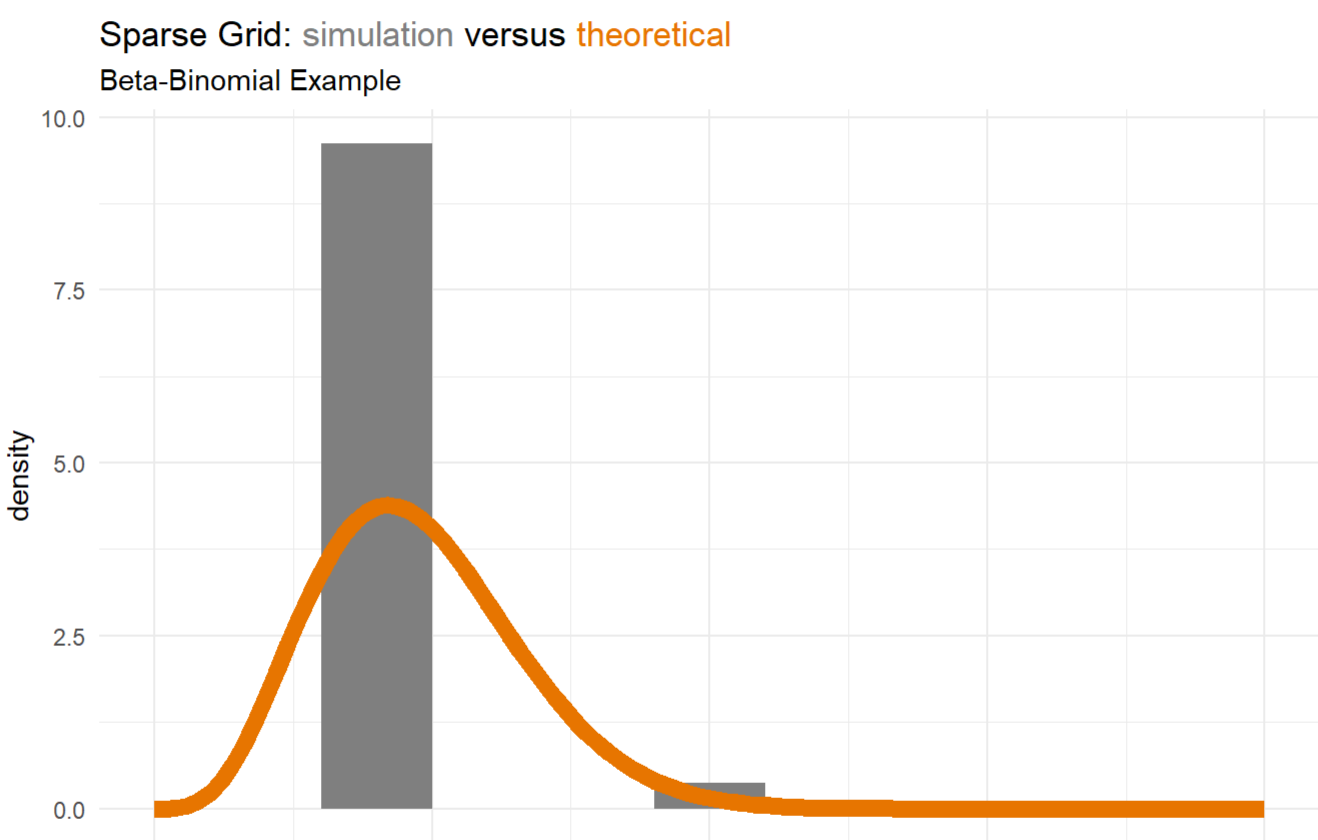
6.5

a

```
# Step 1: Define a grid of 6 pi values
grid_data <- data.frame(pi_grid = seq(from = 0, to = 1,
                                      length = 5))

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

```r
         likelihood = dbinom(2, 10, pi_grid))

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

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

ggplot(posterior_sample, aes(x = pi_grid)) +
  geom_histogram(aes(y = after_stat(density)),
                 binwidth = 0.1,
                 fill = "gray50") +
  stat_function(fun = dbeta, args = list(5, 16),
                color = "#E77500", linewidth = 3) +
  lims(x = c(0, 1)) +
  labs(title = "Sparse Grid: <span style='color:#7F7F7F'>simulation</span> versus <span style='
         subtitle = "Beta-Binomial Example",
         caption = "SML 320") +
  theme_minimal() +
  theme(plot.title = element_markdown())
```

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



0.00 0.25 0.50 0.75 1.00

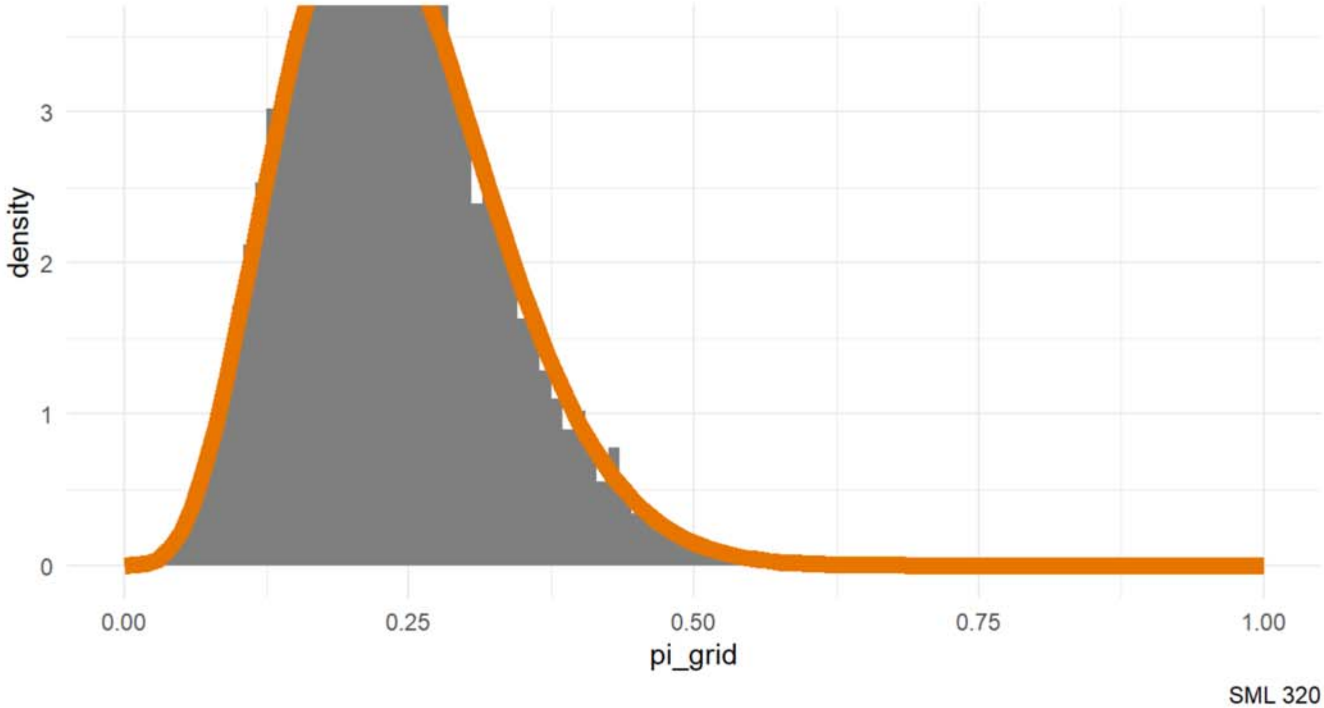
pi_grid

SML 320

b

```
# Step 1: Define a grid of 6 pi values
grid_data <- data.frame(pi_grid = seq(from = 0, to = 1,
                                      length = 501))

# 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))

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

ggplot(posterior_sample, aes(x = pi_grid)) +
  geom_histogram(aes(y = after_stat(density)),
                 binwidth = 0.01,
                 fill = "gray50") +
  stat_function(fun = dbeta, args = list(5, 16),
                color = "#E77500", linewidth = 3) +
  lims(x = c(0, 1)) +
  labs(title = "Dense Grid: <span style='color:#7F7F7F'>simulation</span> versus <span style='c
       subtitle = "Beta-Binomial Example",
       caption = "SML 320") +
  theme_minimal() +
  theme(plot.title = element_markdown())
```

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

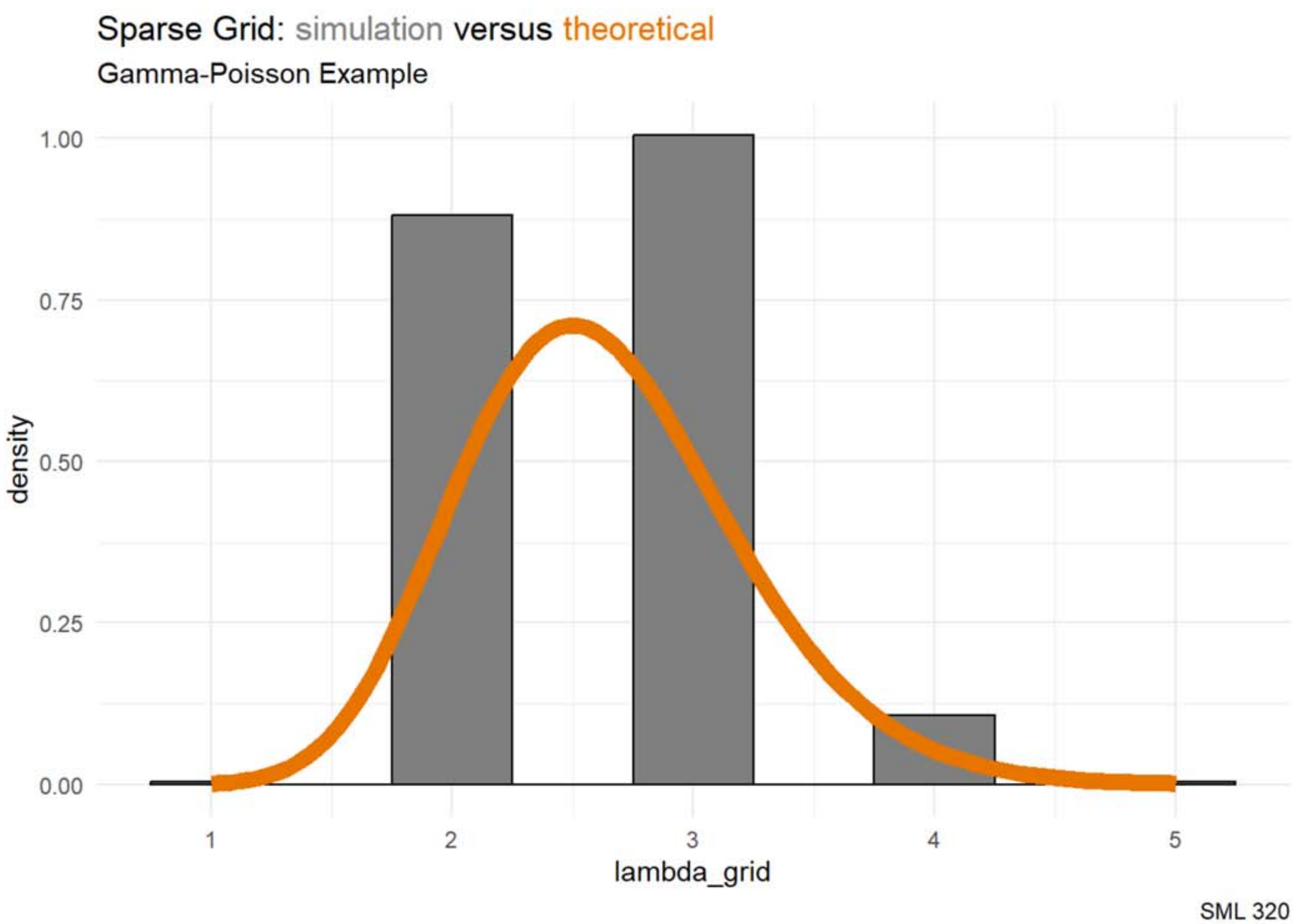


6.6

a

```
obs_counts <- c(0, 1, 0)

# Step 1: Define a grid of 11 pi values
grid_data <- data.frame(lambda_grid = seq(from = 0, to = 8,
                                          length = 9))

# Step 2: Evaluate the prior & likelihood at each pi
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))

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

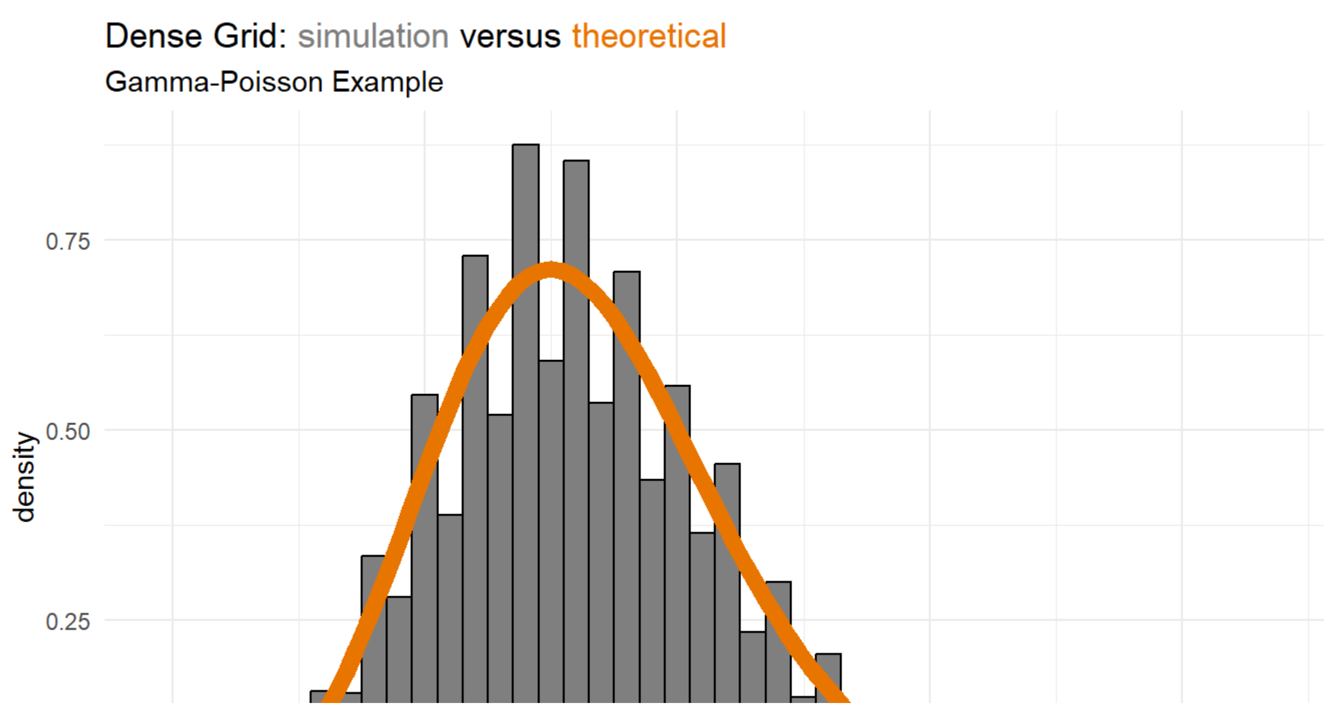
```
ggplot(posterior_sample, aes(x = lambda_grid)) +
  geom_histogram(aes(y = after_stat(density)),
                 binwidth = 0.5,
                 color = "black",
                 fill = "gray50") +
  stat_function(fun = dgamma, args = list(21, 8),
                color = "#E77500", linewidth = 3) +
  labs(title = "Sparse Grid: <span style='color:#7F7F7F'>simulation</span> versus <span style='
       subtitle = "Gamma-Poisson Example",
       caption = "SML 320") +
  theme_minimal() +
  theme(plot.title = element_markdown())
```

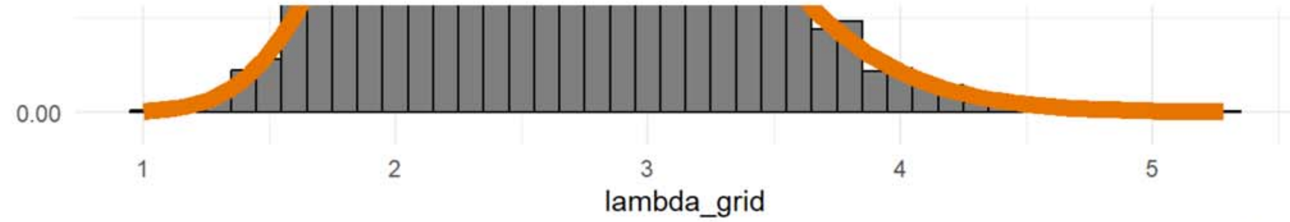


b

```
obs_counts <- c(0, 1, 0)

# Step 1: Define a grid of 11 pi values
grid_data <- data.frame(lambda_grid = seq(from = 0, to = 8,
                                          length = 201))
```

```
# Step 2: Evaluate the prior & likelihood at each pi
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))

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

ggplot(posterior_sample, aes(x = lambda_grid)) +
  geom_histogram(aes(y = after_stat(density)),
                 binwidth = 0.1,
                 color = "black",
                 fill = "gray50") +
  stat_function(fun = dgamma, args = list(21, 8),
                color = "#E77500", linewidth = 3) +
  labs(title = "Dense Grid: <span style='color:#7F7F7F'>simulation</span> versus <span style='
         subtitle = "Gamma-Poisson Example",
         caption = "SML 320") +
  theme_minimal() +
  theme(plot.title = element_markdown())
```





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(3, 8);
  }
"

# STEP 2: SIMULATE the posterior
bb_sim <- stan(model_code = bb_model, data = list(Y = 2),
               chains = 3, iter = 6000*2, seed = 84735)
```

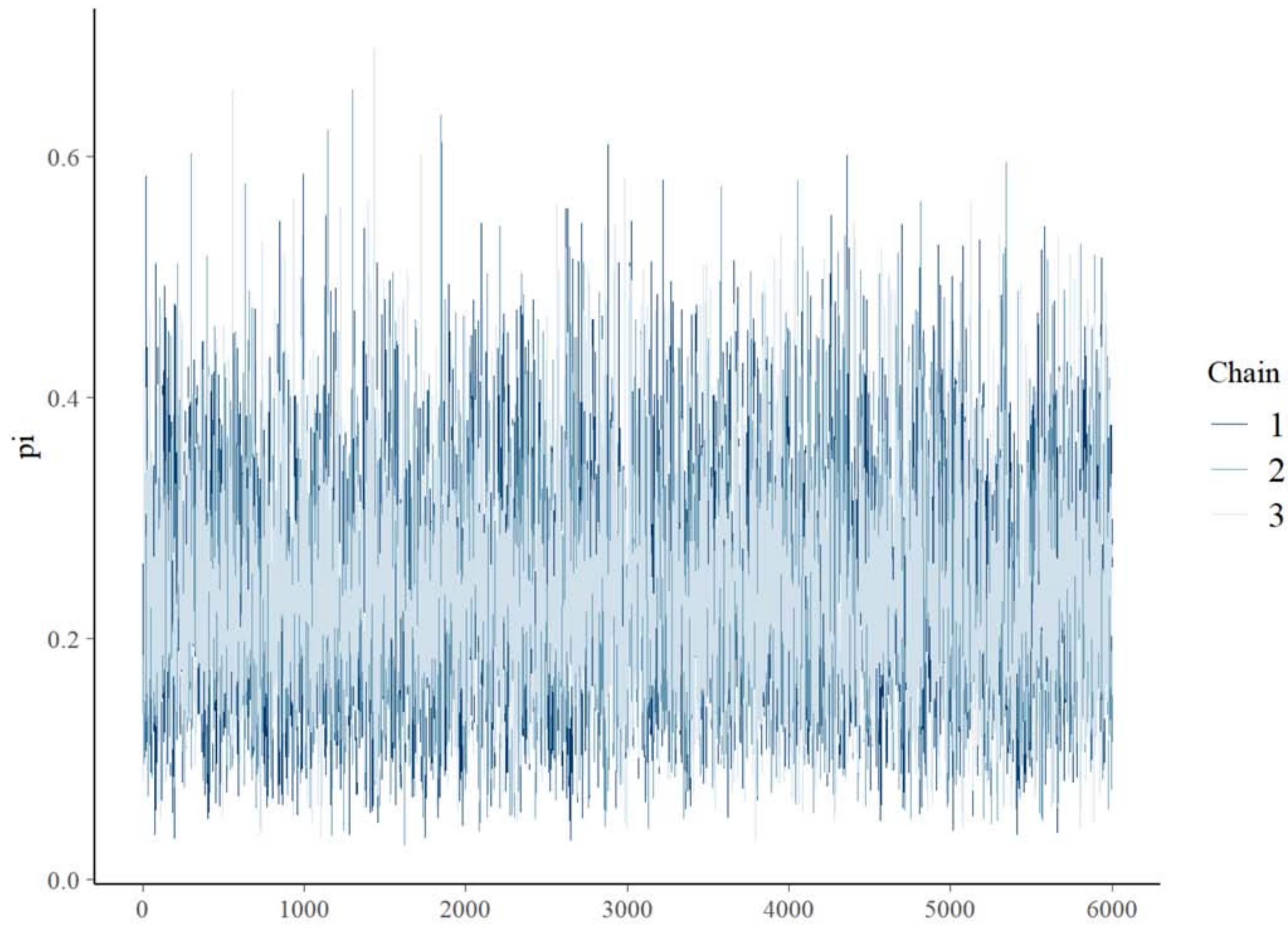
```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 1.1e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.11 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:     1 / 12000 [  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: 0.033 seconds (Warm-up)
Chain 1:                0.033 seconds (Sampling)
Chain 1:                0.066 seconds (Total)
Chain 1: 

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2: 
Chain 2: Gradient evaluation took 2e-06 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.02 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: 0.033 seconds (Warm-up)
Chain 2:                0.034 seconds (Sampling)
Chain 2:                0.067 seconds (Total)
Chain 2: 

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3: 
Chain 3: Gradient evaluation took 2e-06 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.02 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: 0.034 seconds (Warm-up)
Chain 3:                0.035 seconds (Sampling)
Chain 3:                0.069 seconds (Total)
Chain 3:
```

b

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



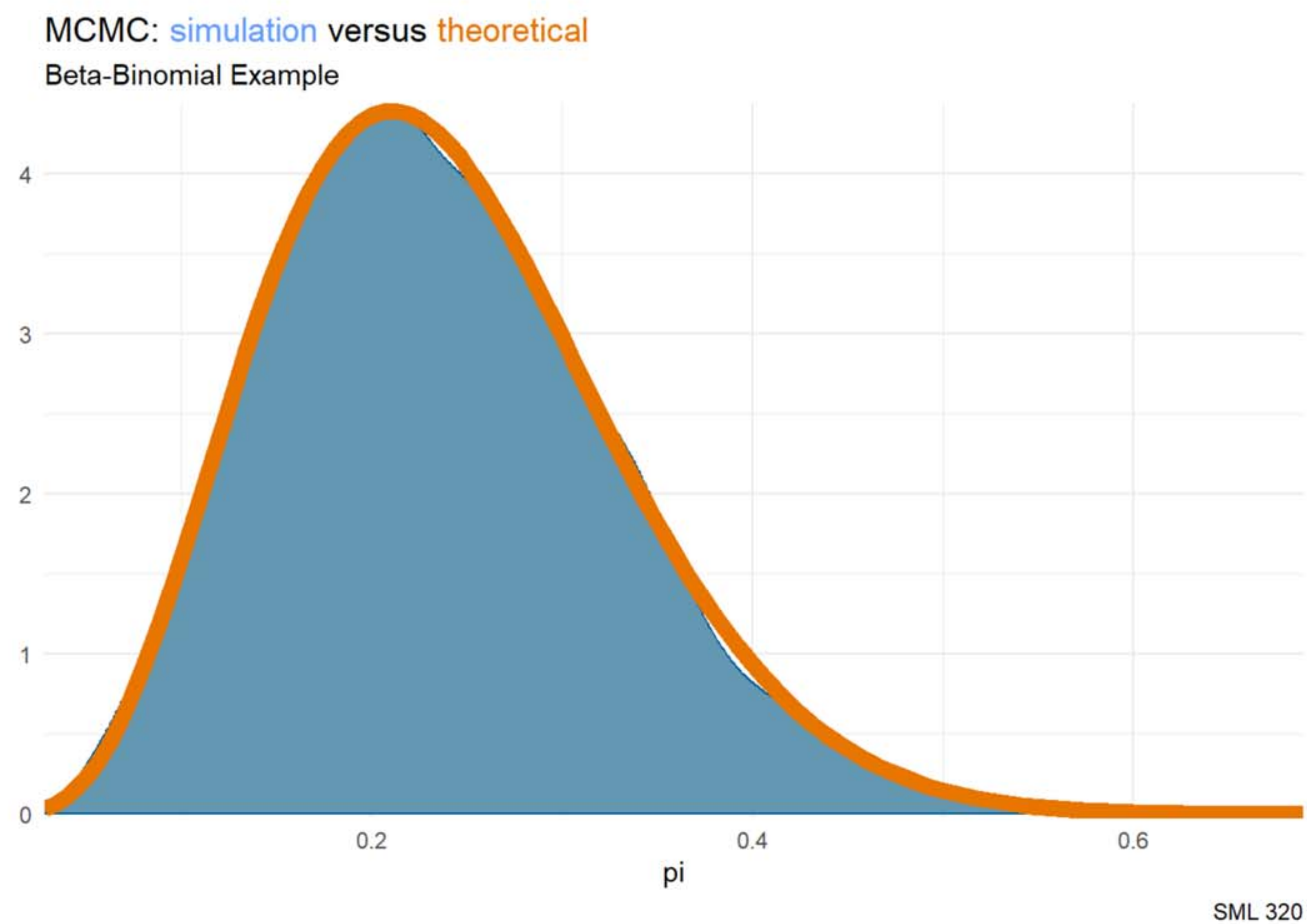
c

The trace plot displays only the last 6000 elements of each chain because we are disregarding the "burn-in" start of the MCMC.

d

d

```
bayesplot::mcmc_dens(bb_sim, pars = "pi") +
  stat_function(fun = dbeta, args = list(5, 16),
                color = "#E77500", linewidth = 3) +
  labs(title = "MCMC: <span style='color:#619CFF'>simulation</span> versus <span style='color:#
       subtitle = "Beta-Binomial Example",
       caption = "SML 320") +
  theme_minimal() +
  theme(plot.title = element_markdown())
```



e

The simulation seems to align with the Beta(5,16) posterior model that we expect.

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

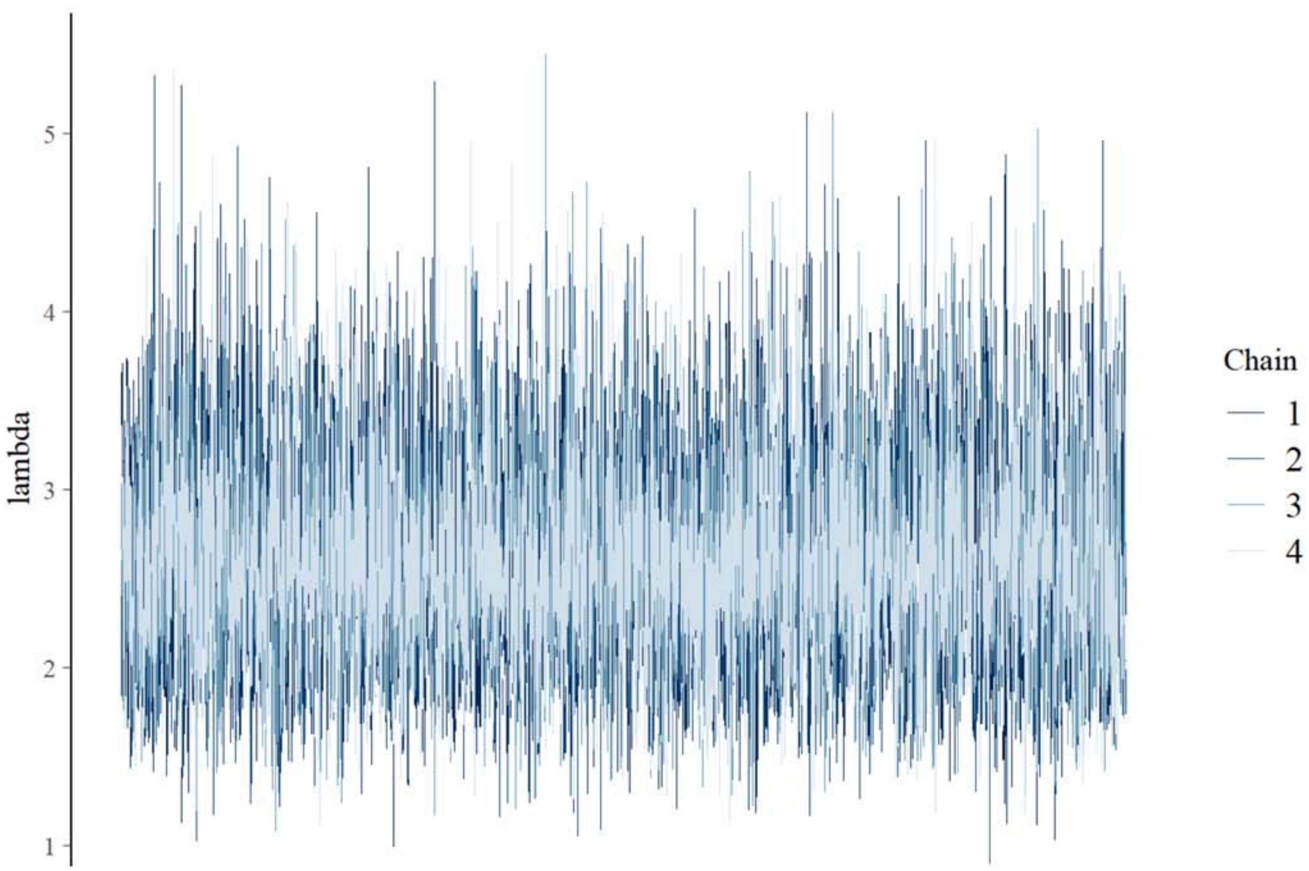
```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 1.5e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.15 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 10000 [  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: 0.026 seconds (Warm-up)
Chain 1:                0.03 seconds (Sampling)
Chain 1:                0.056 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 2e-06 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.02 seconds.
```

```
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.02 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: 0.027 seconds (Warm-up)
Chain 2:                0.025 seconds (Sampling)
Chain 2:                0.052 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 2e-06 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.02 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: 0.026 seconds (Warm-up)
Chain 3:                0.026 seconds (Sampling)
Chain 3:                0.052 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 2e-06 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.02 seconds.
```

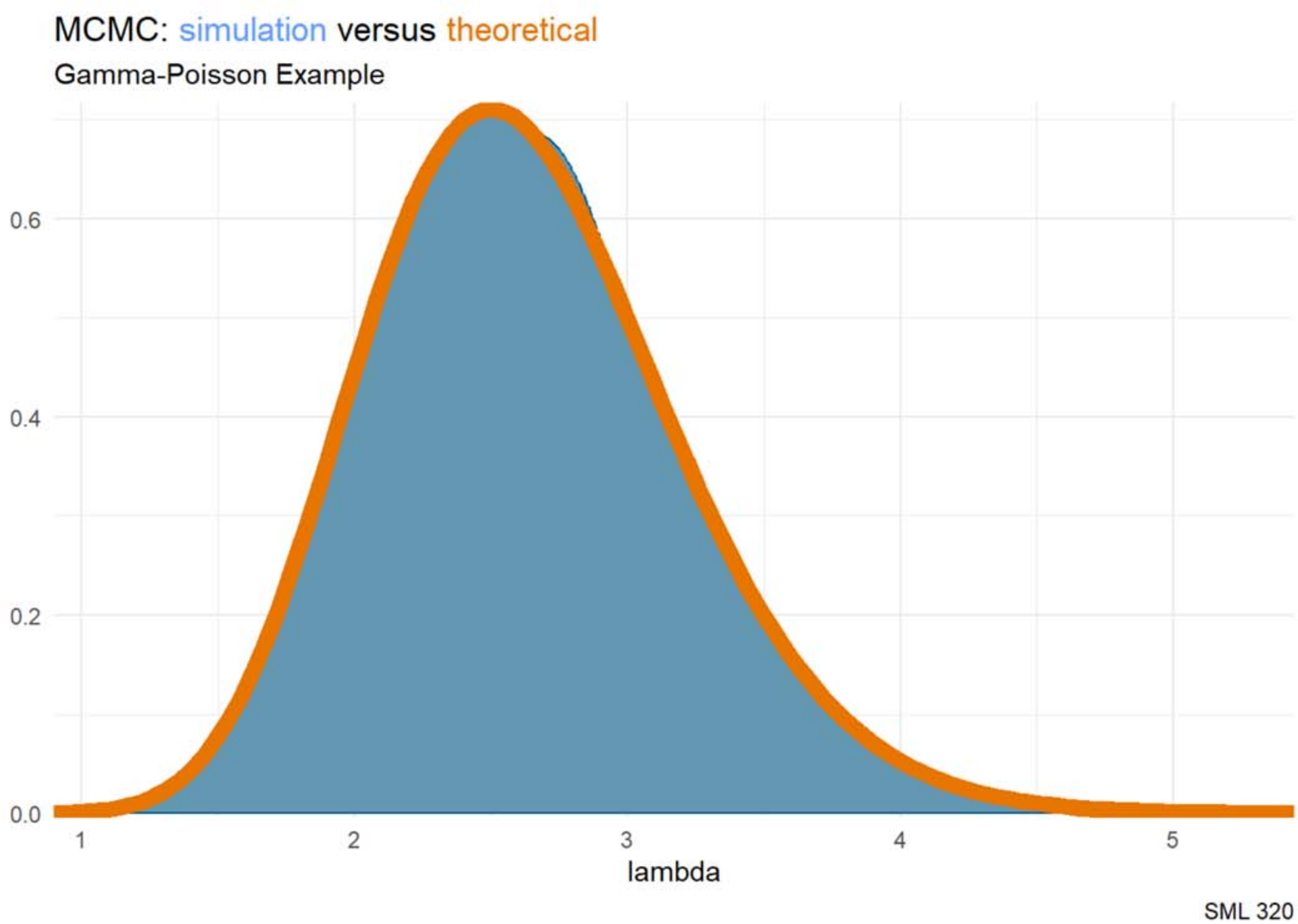
```
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.02 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 4: Iteration: 1000 / 10000 [ 10%]  (Warmup)
Chain 4: Iteration: 2000 / 10000 [ 20%]  (Warmup)
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: 0.026 seconds (Warm-up)
Chain 4:                0.027 seconds (Sampling)
Chain 4:                0.053 seconds (Total)
Chain 4:
```

b

```
bayesplot::mcmc_trace(gp_sim, pars = "lambda", size = 0.1)
```




```
bayesplot::mcmc_dens(gp_sim, pars = "lambda") +
  stat_function(fun = dgamma, args = list(21, 8),
                color = "#E77500", linewidth = 3) +
  labs(title = "MCMC: <span style='color:#619CFF'>simulation</span> versus <span style='color:#
       subtitle = "Gamma-Poisson Example",
       caption = "SML 320") +
  theme_minimal() +
  theme(plot.title = element_markdown())
```



C

From our density plots, the mode from the MCMC seems to be about 2.5

The simulation seems to align with the Gamma(21,8) posterior model that we expect.