Computing Homework 1

Code

Due: Wednesday 11/8 at 11:59pm, Canvas submission (.nb.html file only)

Honor Pledge

On my honor, I have neither received nor given any unauthorized assistance on this Homework.

- SIGNED: 218007361

Hide

```
# Load packages
library(ggplot2)
library(gridExtra)
library(rstan)
library(bayesplot)
```

In this exercise we consider two methods of simulating from the posterior distribution of a Beta-Binomial Bayesian model. The prior distribution is

$$\pi \sim Beta(1, 2),$$

and the data model is

$$y_i \mid \pi \overset{ind}{\sim} Bin(6, \pi).$$

Two observations are made:

$$\vec{y} = (y_1, y_2) = (3, 4).$$

Complete the following questions.

Question 1: analytical posterior (1pt)

Fill in the two blanks with numbers: the analytical posterior for π is

$$\pi \mid \vec{y} \sim Beta(8, 7).$$

Question 2: Posterior simulation in `rstan` (5 pts total)

2 - 1. (1pt)

Fill in the code chunk named `model` below, to define the Bayesian model structure in `rstan`. Use the following notation in your definition:

- `Y` for the data vector $\vec{y}$;
- `pi` for the parameter.

Be sure to specify the data type, range, vector length (if applicable), etc.

Hide

```
the_model <- "
  data {
    int<lower=0, upper=6> Y[2];
  }
  parameters {
    real<lower=0, upper=1> pi;
  }
  model {
    Y ~ binomial(6, pi);
    pi ~ beta(1, 2);
  }
"
```

2 - 2. (0pt)

Complete the code below to simulate from the posterior using the model defined above. Use the following settings:

- number of `chains` : 4
- number of `iter` : 10,000.

Hide

```
the_sim <- stan(model_code = the_model, data = list(Y = c(3,4)),
                chains = 4, iter = 10000, seed = 1000)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 5e-06 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.05 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: 0.029 seconds (Warm-up)
Chain 1:                0.034 seconds (Sampling)
Chain 1:                0.063 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 3e-06 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.03 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: 0.028 seconds (Warm-up)
Chain 2:                0.178 seconds (Sampling)
Chain 2:                0.206 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
```

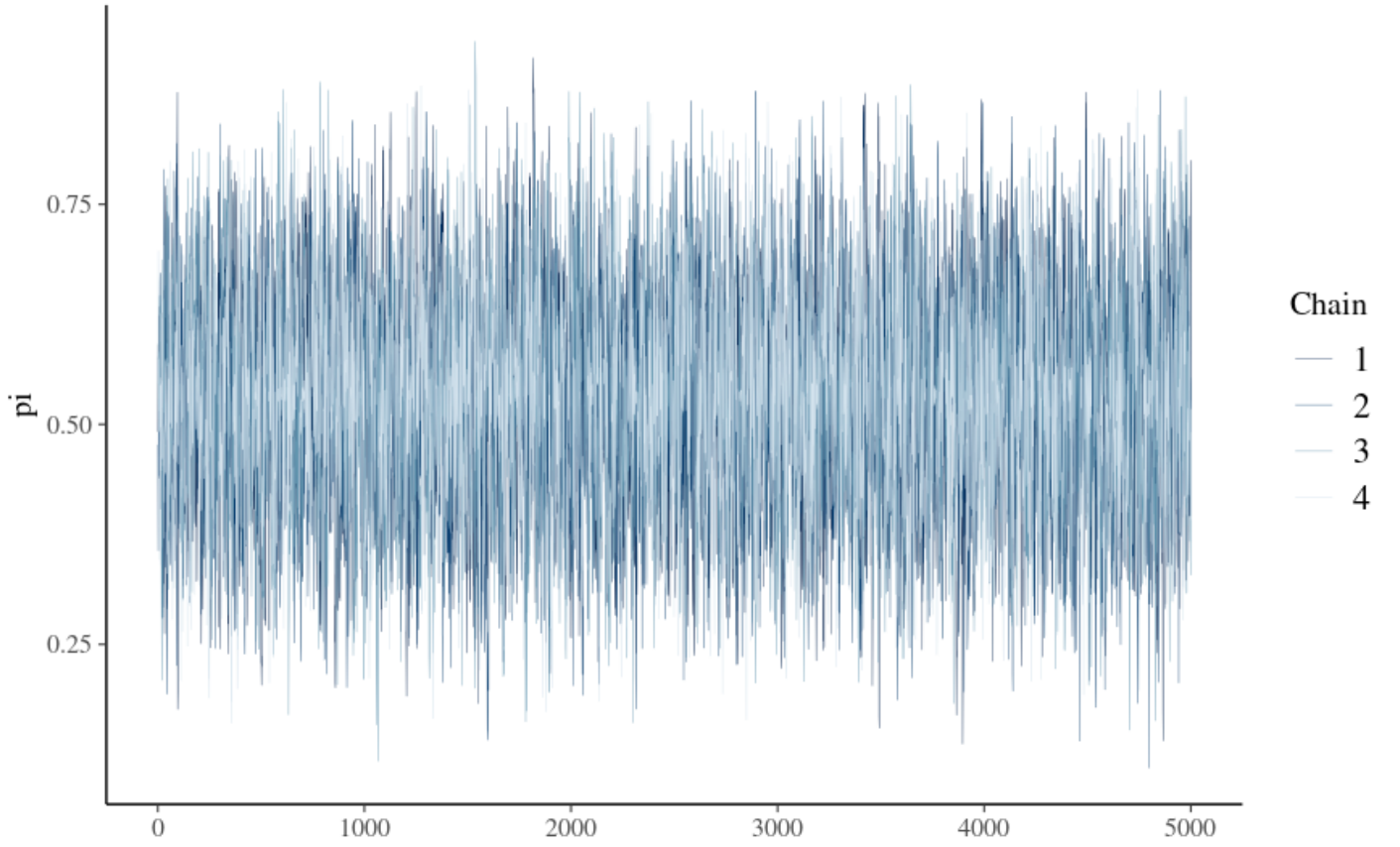
```
Chain 3:
Chain 3: Gradient evaluation took 3e-06 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.03 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 3: Iteration: 1000 / 10000 [ 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: 0.029 seconds (Warm-up)
Chain 3:                0.029 seconds (Sampling)
Chain 3:                0.058 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 3e-06 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.03 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: 0.029 seconds (Warm-up)
Chain 4:                0.03 seconds (Sampling)
Chain 4:                0.059 seconds (Total)
Chain 4:
```

2 - 3. (1pt)

Visualize the trace plots of the chains in `the_sim` :

Hide

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

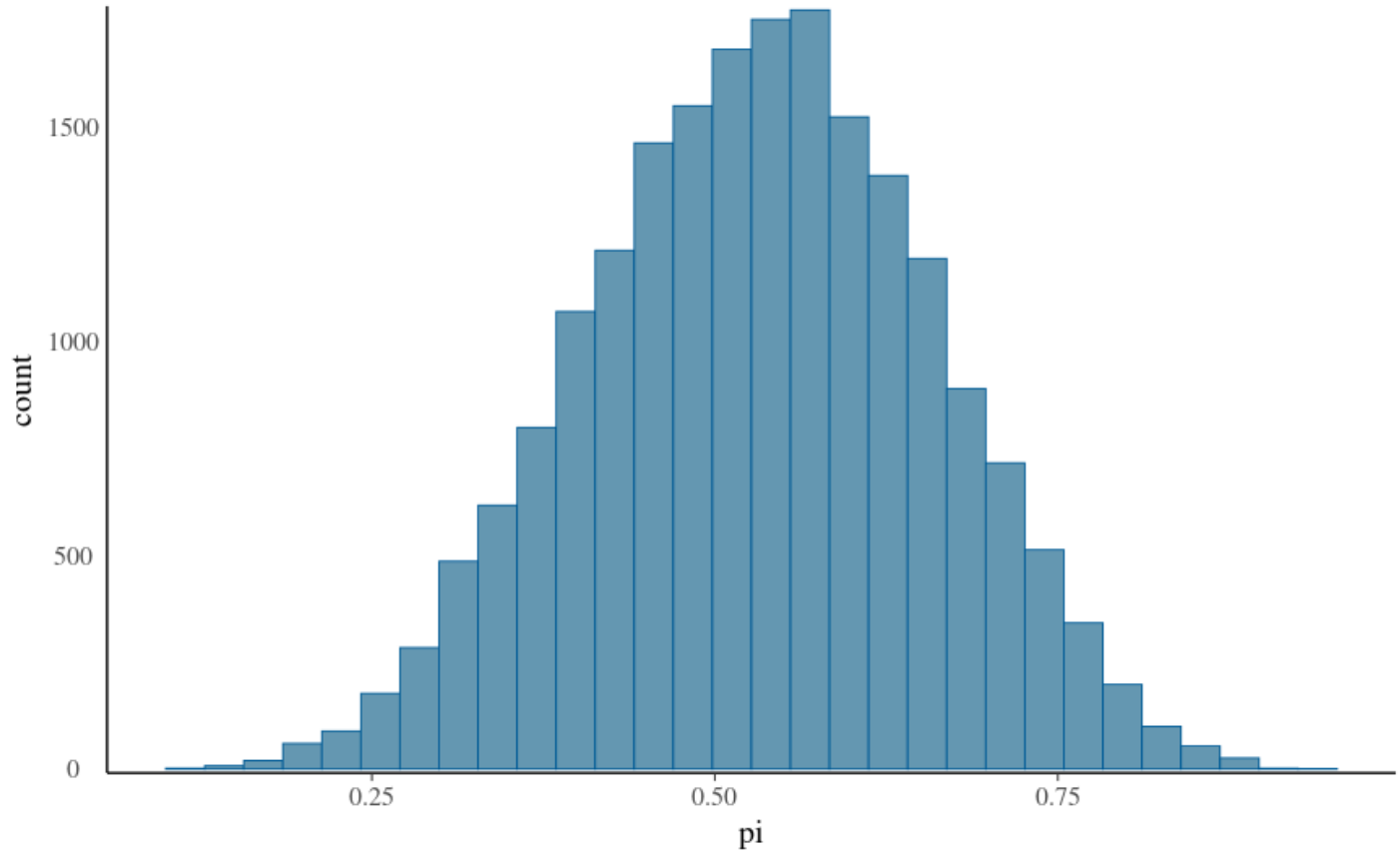


2 - 4. (1pt)

Visualize the histogram of the chains in `the_sim` :

Hide

```
mcmc_hist(the_sim, pars = "pi") +
  yaxis_text(TRUE) + ylab("count")
```



2 - 5. (1pt)

Calculate the effective sample size ratio for `the_sim` :

Hide

```
neff_ratio(the_sim, pars = c("pi"))
```

```
[1] 0.3701599
```

- Is this result satisfactory? Yes

2 - 6. (1pt)

Calculate the $\hat{R}$ for the chains in `the_sim` :

Hide

```
rhat(the_sim, pars = "pi")
```

```
[1] 1.000401
```

- Is this result satisfactory? Yes

Question 3: Posterior simulation via Independence Sampler (4 pts total)

Now forget about what you did using `rstan`. Instead, use the Independence Sampler algorithm to target the posterior distribution for π, **assuming you do not know its analytical distribution**. Use proposal distribution

$$\pi^* \sim Beta(a, b).$$

3 - 1. (1pt)

Complete the code chunk below to define the function, `one_iteration`, that produces one iteration of the independent sampler. Here,

- `a` : hyperparameter a;
- `b` : hyperparameter b;
- `current` : current value of the chain.

Hide

```
one_iteration <- function(a, b, current){
 # STEP 1: Propose the next chain location
 proposal <- rbeta(1, a, b)
 # STEP 2: Decide whether or not to go there
 proposal_plaus <- dbeta(proposal, 1, 2) * dbinom(7, 12, proposal)
 proposal_q     <- dbeta(proposal, a, b)
 current_plaus  <- dbeta(current, 1, 2) * dbinom(7, 12, current)
 current_q      <- dbeta(current, a, b)

 alpha <- min(1, proposal_plaus/current_plaus * current_q/proposal_q)
 next_stop <- sample(c(proposal, current),
                     size = 1, prob = c(alpha, 1-alpha))
 return(data.frame(proposal, alpha, next_stop))
}
```

3 - 2. (3 pts)

The function `whole_tour`, which produces `N` iterations of the independence sampler, has been defined for you below.

Hide

```
whole_tour <- function(N, a, b){
  # 1. Start the chain at location 0.5
  current <- 0.5
  # 2. Initialize the simulation
  pi <- rep(0, N)
  # 3. Simulate N Markov chain stops
  for(i in 1:N){
    # Simulate one iteration
    sim <- one_iteration(a = a, b = b, current = current)
    # Record next location
    pi[i] <- sim$next_stop
    # Reset the current location
    current <- sim$next_stop
  }
  # 4. Return the chain locations
  return(data.frame(iteration = c(1:N), pi))
}
```

Complete the following chunks of code to:

- (1pt) Use the `whole_tour` function to produce the object `independence_sim`, consisting of `N = 5000` iterations from the independent sampler, with prior hyperparameters `a = 1` and `b = 2`;

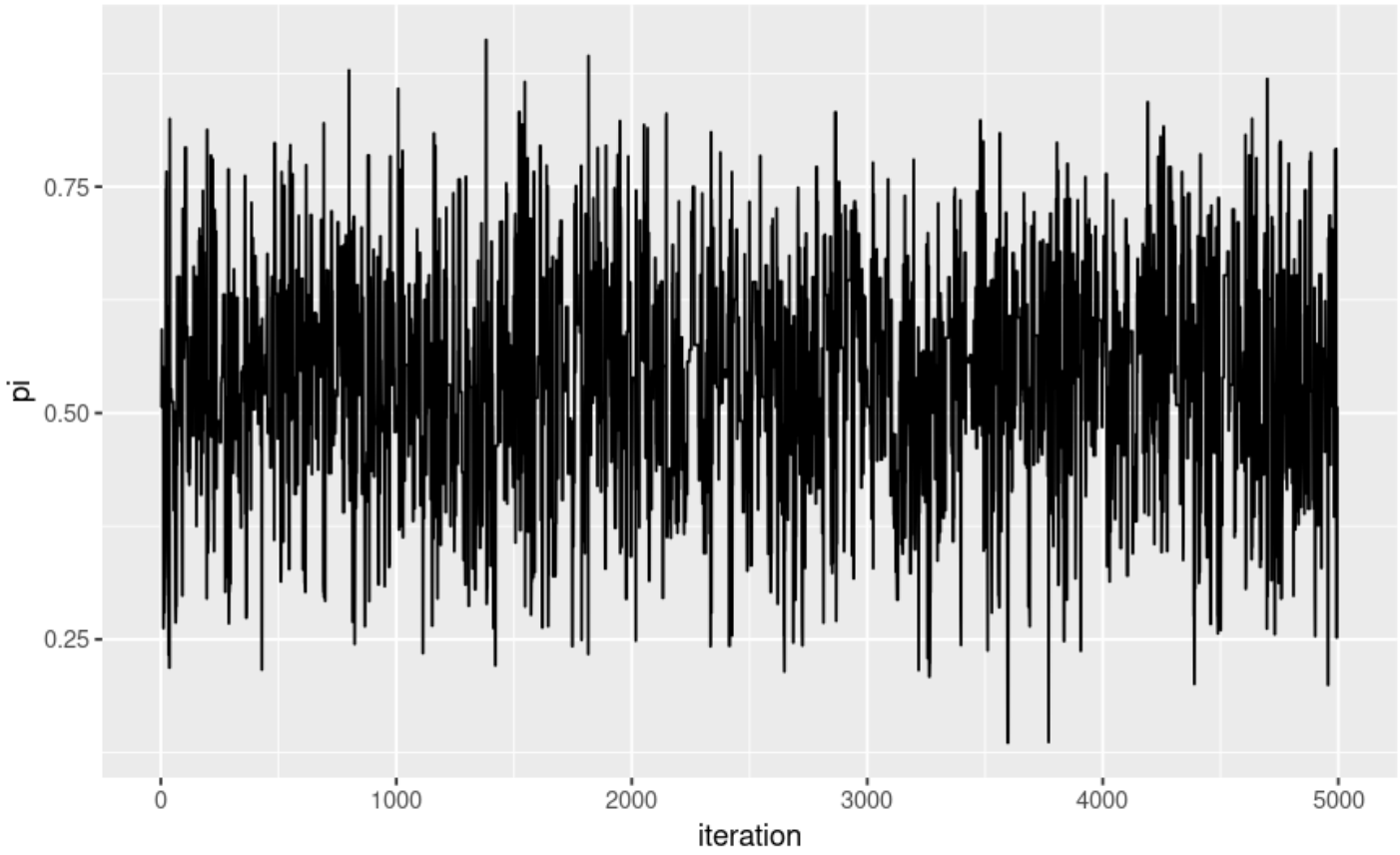
Hide

```
set.seed(1000)
independence_sim <- whole_tour(N=5000, a=1, b=2)
```

- (1pt) Produce the trace plot of the chain in `independence_sim`:

Hide

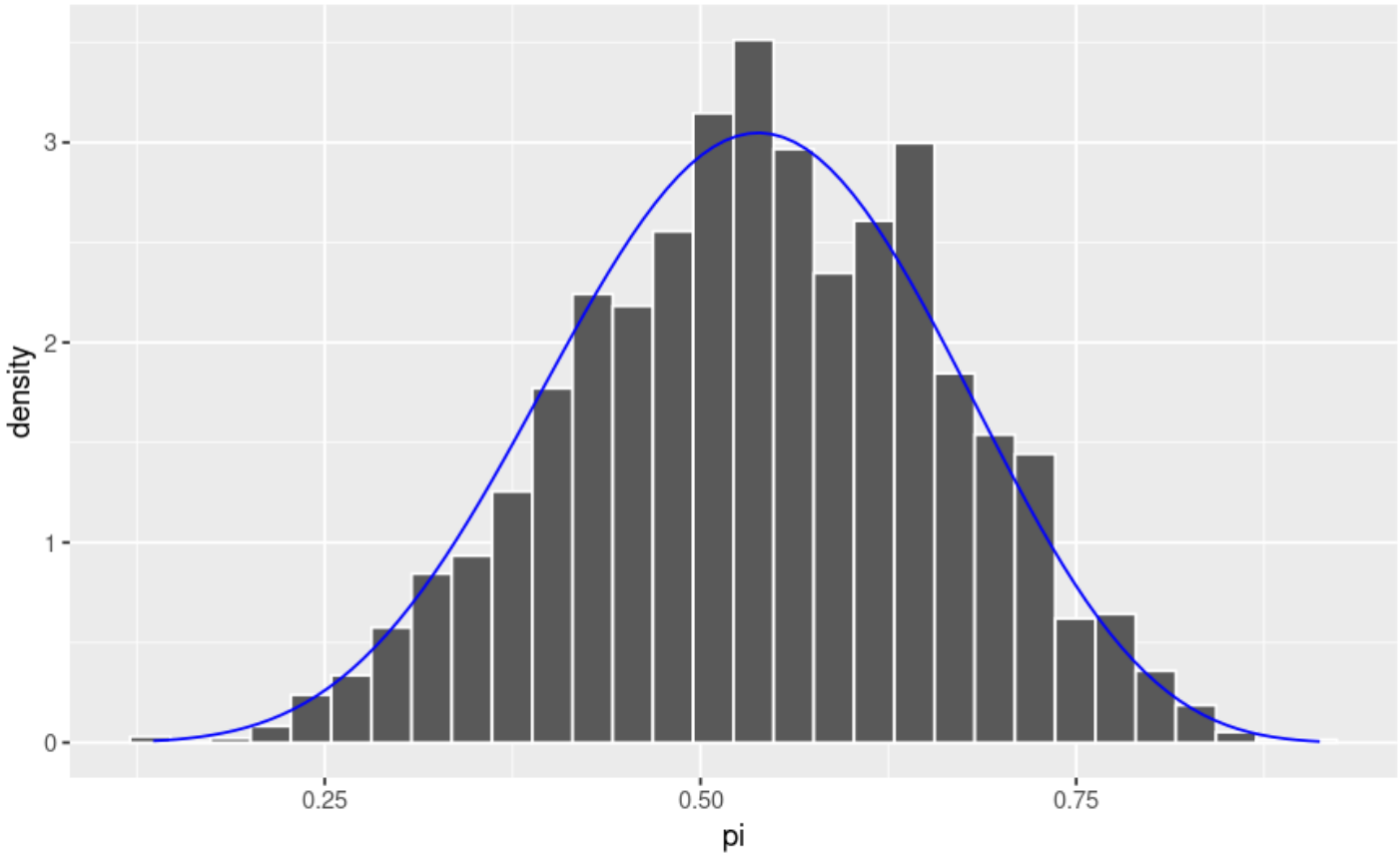
```
ggplot(independence_sim, aes(x = iteration, y =  pi)) +
  geom_line()
```



- (1pt) Produce the histogram of the chain in `independence_sim` , and overlay it with the analytical posterior for π which you derived from Question 1.

Hide

```
ggplot(independence_sim, aes(x = pi)) +
  geom_histogram(aes(y = ..density..), bins=30, color = "white") +
  stat_function(fun = dbeta, args = list(8,7), color = "blue")
```


density
3
2
1
0
0.25
0.50
0.75
pi