Gibss-sampling-Activity-Solution

Instructions

1. Download this `.Rmd` file and open it in RStudio.
2. Update the author name and date.
3. Answer the questions below, using LaTeX within R Markdown to type up your answers.
4. When you finish, `Knit to PDF`.

Markov Chains

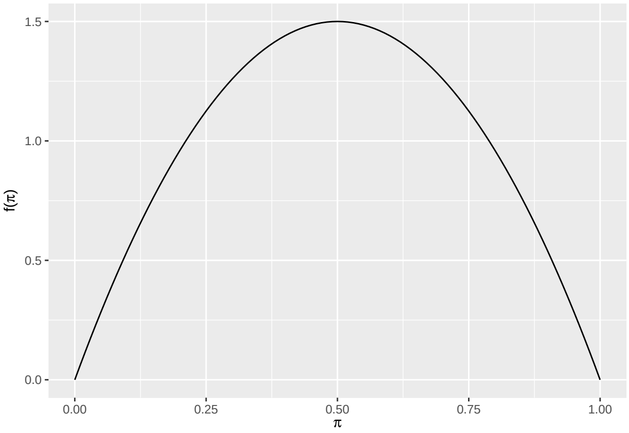
First we will implement a Markov chain. We will conduct the MCMC simulation using the rstan package (Guo and Weber 2020). There are two essential steps to all rstan analyses, first we define the Bayesian model structure and then simulate the posterior. We will use a generical Beta-Binomial example:

$$Y \mid \pi \sim Bin(10, \pi)$$
$$\pi \sim Beta(2, 2)$$

Where Y is the number of successes in 10 independent trials. Each trial has a probability of success π where our prior for π is captured by a $Beta(2, 2)$ model. If we observe 9 successes we have an updated posterior model of π with distribution $Beta(11, 3)$. Don't worry about how we arrived to this posterior for the moment being. Keep in mind our goal is to run an MCMC algorithm to produce an approximate sample from the Beta-binomial posterior $Beta(11, 3)$.

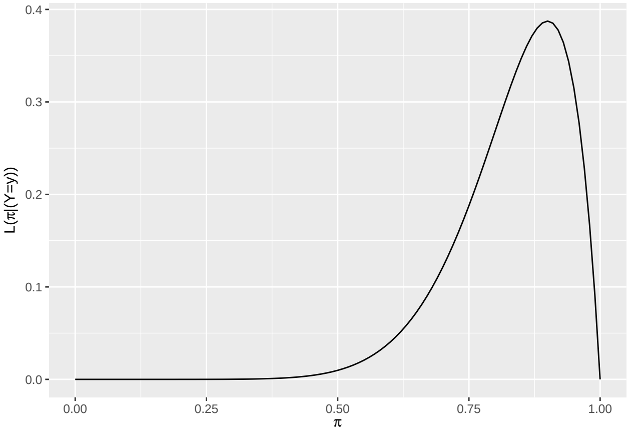
This plot represents our prior beliefs

```
plot_beta(2,2)
```



This plot shows our likelihood based on our two data points

```
plot_binomial_likelihood(y = 9,n = 10)
```



STEP 1: DEFINE the model

Observe there are 3 different elements of this code. You don't need to change anything, but we encourage you to identify the prior, posterior and parameters of the model in the code.

Data: Y is the observed number of success trials. We specify that Y is between 10 and 0.

Parameters: The model depends on π, therefore we must specify that π can be any real number from 0 to 1.

Model: We need to specify the model for the data and the model for the prior.

```
# 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(2, 2);
  }
"
```

STEP 2: Simulate the posterior

We simulate the posterior using the stan() function. This function designs and runs an MCMC algorithm to produce an approximate sample from the Beta-Binomial posterior.

The *model_code* argument requires a string that defines the model, which we created in the previous step.

The *data* argument requires a list of observed data.

The *chains* argument specifies how many parallel Markov Chains we are running. Since we are running four chains we will have four π values.

The *iter* argument specifies the number of iterations or length for each chain. The first half of this iterations are thrown out as “burn in” samples (samples that we use to calibrate our model).

We utilize the *seed* argument within the stan() function to keep our random results constant.

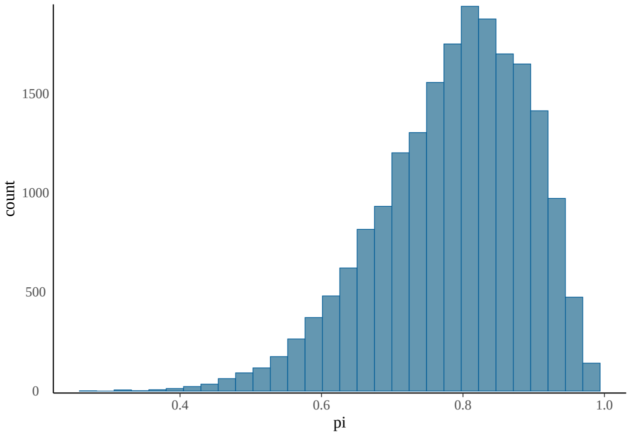
```
bb_sim <- stan(model_code = bb_model, #model_code argument
               data = list(Y = 9),  # data argument
               chains = 4, #we are running four chains
               iter = 5000*2, #number of iterations or length for each chain
               seed = 84735) #keep our random results constant
```

```
##
## SAMPLING FOR MODEL ’7b64678a3565f32e51f77686e11b9c04’ NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.1e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.21 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: 0.067504 seconds (Warm-up)
## Chain 1:                0.075371 seconds (Sampling)
## Chain 1:                0.142875 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL ’7b64678a3565f32e51f77686e11b9c04’ 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: 0.061774 seconds (Warm-up)
## Chain 2:                0.055056 seconds (Sampling)
## Chain 2:                0.11683 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '7b64678a3565f32e51f77686e11b9c04' 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: 0.062749 seconds (Warm-up)
## Chain 3:                0.075173 seconds (Sampling)
## Chain 3:                0.137922 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '7b64678a3565f32e51f77686e11b9c04' 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: 0.059229 seconds (Warm-up)
## Chain 4:                0.061673 seconds (Sampling)
## Chain 4:                0.120902 seconds (Total)
## Chain 4:
```

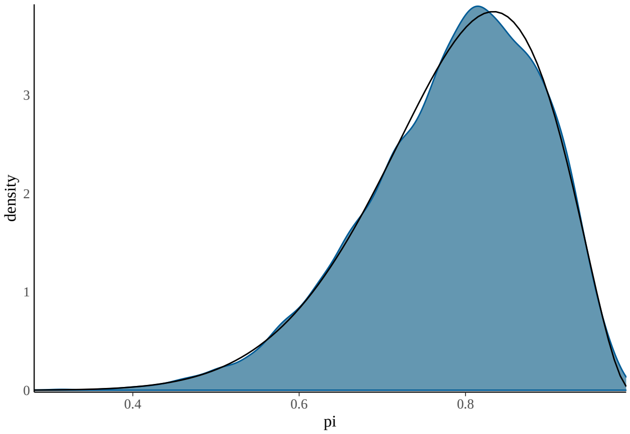
As you can see in Figure 1, when observing the distribution of the sampled π values we approximate the target Beta(11,3) posterior model of π. The target pdf superimposed to our simulated model. Excellent! Now you know how to approximate a posterior using Markov Chains (Alicia A. Johnson 2022).

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

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



```
# Density plot of the Markov chain values
p = seq(0, 1, length=100)
mcmc_dens(bb_sim, pars = "pi") +
  yaxis_text(TRUE) +
  ylab("density")+
  stat_function(fun = dbeta, args = list(11, 3))
```



Gibbs Sampling

Suppose we have data from a normal distribution where both the mean **and** variance are unknown. For convenience, we'll parameterize this model in terms of the *precision* $\gamma = \frac{1}{\sigma^2}$ instead of the variance σ^2.

$$Y \mid \mu, \gamma \sim N\left(\mu, \frac{1}{\gamma}\right)$$

Suppose we put the following *independent* priors on the mean μ and precision γ:

$$\mu \sim N(m, v)$$

$$\gamma \sim \text{Gamma}(a, b)$$

1. Write down the joint posterior distribution for μ, γ. Does this look like a recognizable probability distribution?

ANSWER: No, this is not a recognizable posterior:

$$\begin{aligned}
g(\mu, \gamma \mid y) &\propto f(y \mid \mu, \gamma) f(\mu, \gamma) \\
&= f(y \mid \mu, \gamma) f(\mu) f(\gamma), \text{ since } \mu, \gamma \text{ independent} \\
&= \left[(2\pi)^{-\frac{1}{2}} \gamma^{\frac{1}{2}} e^{-\frac{1}{2}\gamma(y-\mu)^2}\right] \left[(2\pi v)^{-\frac{1}{2}} e^{-\frac{1}{2v}(\mu-m)^2}\right] \left[\frac{b^a}{\Gamma(a)} \gamma^{a-1} e^{-b\gamma}\right] \\
&\propto \gamma^{\frac{1}{2}} e^{-\frac{1}{2}\gamma(y-\mu)^2} e^{-\frac{1}{2v}(\mu-m)^2} \gamma^{a-1} e^{-b\gamma} \\
&= \gamma^{\frac{1}{2}+a-1} e^{-\frac{1}{2}\gamma(y-\mu)^2 + -\frac{1}{2v}(\mu-m)^2 - b\gamma} \\
&= \gamma^{\frac{1}{2}+a-1} e^{-\frac{1}{2}\left[\gamma(y-\mu)^2 + \frac{1}{v}(\mu-m)^2 + 2b\gamma\right]} \\
&= \gamma^{\frac{1}{2}+a-1} e^{-\frac{1}{2}\left[\gamma y^2 - 2\mu y\gamma + \gamma\mu^2 + \mu^2/v - 2m\mu/v + m^2/v + 2b\gamma\right]} \\
&= \gamma^{\frac{1}{2}+a-1} e^{-\frac{1}{2}\left[\gamma(y^2+2b) - 2\mu(y\gamma + m/v) + \mu^2(\gamma + 1/v) + m^2/v\right]} \\
&\propto \gamma^{\frac{1}{2}+a-1} e^{-\frac{1}{2}\left[\gamma(y^2+2b) - 2\mu(y\gamma + \frac{m}{v}) + \mu^2(\frac{1}{v} + \gamma)\right]}
\end{aligned}$$

You should have answered "no" to Question 1, meaning that we can't use our usual techniques here to find Bayes estimators for μ or γ since we don't have a recognizable posterior distribution. Instead, we'll use a computational technique known as *Gibbs Sampling* to generate samples from this posterior distribution. Gibbs Sampling is particularly useful when we have more than one parameter, and the basic idea involves reducing our problem to a series of calculations involving one parameter at a time. In order to perform Gibbs Sampling, we need to find the conditional distributions

$$g(\mu \mid y, \gamma) \propto f(y \mid \mu, \gamma) f(\mu)$$

$$g(\gamma \mid y, \mu) \propto f(y \mid \mu, \gamma) f(\gamma)$$

We will use these conditional distributions to sample from the joint posterior $g(\mu, \gamma \mid y)$ according to the following algorithm:

(1) Start with initial values $\mu^{(0)}, \gamma^{(0)}$.

(2) Sample $\mu^{(t+1)} \sim g(\mu \mid y, \gamma = \gamma^{(t)})$.

(3) Sample $\gamma^{(t+1)} \sim g(\gamma \mid y, \mu = \mu^{(t+1)})$.

(4) Repeat many times.

It turns out that the resulting $\mu^{(0)}, \mu^{(1)}, \ldots, \mu^{(N)}$ and $\gamma^{(0)}, \gamma^{(1)}, \ldots, \gamma^{(N)}$ are samples from the joint posterior distribution $g(\mu, \gamma \mid Y)$, and we can use these sampled values to estimate quantities such as the posterior mean of each parameter $\hat{E}(\mu \mid y) = \frac{1}{N}\sum_{i=1}^{N} \mu^{(i)}, \;\; \hat{E}(\gamma \mid y) = \frac{1}{N}\sum_{i=1}^{N} \gamma^{(i)}$. Note that in practice we typically remove the initial iterations, known as the "burn-in" period: e.g., $\hat{E}(\mu \mid y) = \frac{1}{N-B}\sum_{i=B}^{N} \mu^{(i)}$.

2. Show that the conditional distributions $g(\mu \mid y, \gamma), g(\gamma \mid y, \mu)$ are proportional to $f(y \mid \mu, \gamma)f(\mu), f(y \mid \mu, \gamma)f(\gamma)$, respectively, as stated above.

$$\begin{aligned} g(\mu \mid y, \gamma) &= \frac{f(\mu, y, \gamma)}{f(y, \gamma)} \\ &\propto f(\mu, y, \gamma), \text{ since } f(y, \gamma) \text{ doesn't depend on } \mu \\ &= f(y \mid \mu, \gamma) f(\mu, \gamma) \\ &= f(y \mid \mu, \gamma) f(\mu) f(\gamma), \text{ since } \mu, \gamma \text{ independent} \\ &\propto f(y \mid \mu, \gamma) f(\mu), \text{ since} f(\gamma) \text{ doesn't depend on } \mu \end{aligned}$$

3. The next set of operations show that $\mu \mid y, \gamma \sim N\left(\frac{y\gamma + \frac{m}{v}}{\gamma + \frac{1}{v}}, \left[\gamma + \frac{1}{v}\right]^{-1}\right)$ and $\gamma \mid y, \mu \sim \text{Gamma}\left(\frac{1}{2} + a, \frac{1}{2}(y-\mu)^2 + b\right)$ based on the previous result. Complete the missing steps in the following operations.

ANSWER:

$$g(\mu \mid y, \gamma) \propto f(y \mid \mu, \gamma) f(\mu)$$

$$= \left[(2\pi)^{-\frac{1}{2}} \gamma^{\frac{1}{2}} e^{-\frac{1}{2}\gamma(y-\mu)^2}\right] \left[(2\pi v)^{-\frac{1}{2}} e^{-\frac{1}{2v}(\mu-m)^2}\right]$$

$$\propto e^{-\frac{1}{2}\gamma(y-\mu)^2 - \frac{1}{2v}(\mu-m)^2}$$

$$= e^{-\frac{1}{2}\gamma(y^2-2\mu y+\mu^2) - \frac{1}{2v}(\mu^2-2\mu m+m^2)}$$

$$\propto e^{-\frac{1}{2}\gamma(-2\mu y+\mu^2) - \frac{1}{2v}(\mu^2-2\mu m)}$$

$$\propto e^{-\frac{1}{2}[\gamma(-2\mu y+\mu^2) - \frac{1}{v}(\mu^2-2\mu m)]}$$

$$\propto e^{-\frac{1}{2}[-2\gamma\mu y+\gamma\mu^2) - \frac{1}{v}\mu^2 - \frac{1}{v}2\mu m)]}$$

$$\propto e^{-\frac{1}{2}[\mu^2(\gamma+\frac{1}{v}) - 2\mu - [y\gamma+\frac{m}{v}]}$$

$$= e^{-\frac{1}{2}\left[\mu^2(\gamma+\frac{1}{v}) - 2\mu(y\gamma+\frac{m}{v})\right]}$$

$$= e^{-\frac{1}{2}(\gamma+\frac{1}{v})\left[\mu^2 - 2\mu\left(\frac{y\gamma+\frac{m}{v}}{\gamma+\frac{1}{v}}\right)\right]}$$

$$\propto e^{-\frac{1}{2}(\gamma+\frac{1}{v})\left[\mu^2 - 2\mu\left(\frac{y\gamma+\frac{m}{v}}{\gamma+\frac{1}{v}}\right) + \left(\frac{y\gamma+\frac{m}{v}}{\gamma+\frac{1}{v}}\right)^2\right]}$$

$$\text{because of inverse proportionality} \propto e^{-\frac{1}{2\left(\gamma+\frac{1}{v}\right)^{-1}}\left[\mu - \left(\frac{y\gamma+\frac{m}{v}}{\gamma+\frac{1}{v}}\right)\right]^2}$$

$$\implies \mu \mid y, \gamma \sim N\left(\frac{y\gamma + \frac{m}{v}}{\gamma + \frac{1}{v}}, \left[\gamma + \frac{1}{v}\right]^{-1}\right)$$

$$\begin{aligned}
g(\gamma \mid y, \mu) &\propto f(y \mid \mu, \gamma) f(\gamma) \\
&= \left[(2\pi)^{-\frac{1}{2}} \gamma^{\frac{1}{2}} e^{-\frac{1}{2}\gamma(y-\mu)^2}\right] \left[\frac{b^a}{\Gamma(a)} \gamma^{a-1} e^{-b\gamma}\right] \\
&\propto \gamma^{\frac{1}{2}} \gamma^{a-1} e^{-\frac{1}{2}\gamma(y-\mu)^2} e^{-b\gamma} \\
&= \gamma^{\frac{1}{2}+a-1} e^{-\frac{1}{2}\gamma(y-\mu)^2 - b\gamma} \\
&= \gamma^{\frac{1}{2}+a-1} e^{-\gamma\left(\frac{1}{2}(y-\mu)^2 + b\right)}
\end{aligned}$$

$$\implies \gamma \mid y, \mu \sim \text{Gamma}\left(\frac{1}{2} + a, \frac{1}{2}(y-\mu)^2 + b\right)$$

4. Suppose that we choose the following hyperparameters for our prior distributions—$m = 0, v = 1, a = 1, b = 1$—and that we observe $y = 2$. Uncomment and complete the code to implement this Gibbs Sampler.

ANSWER:

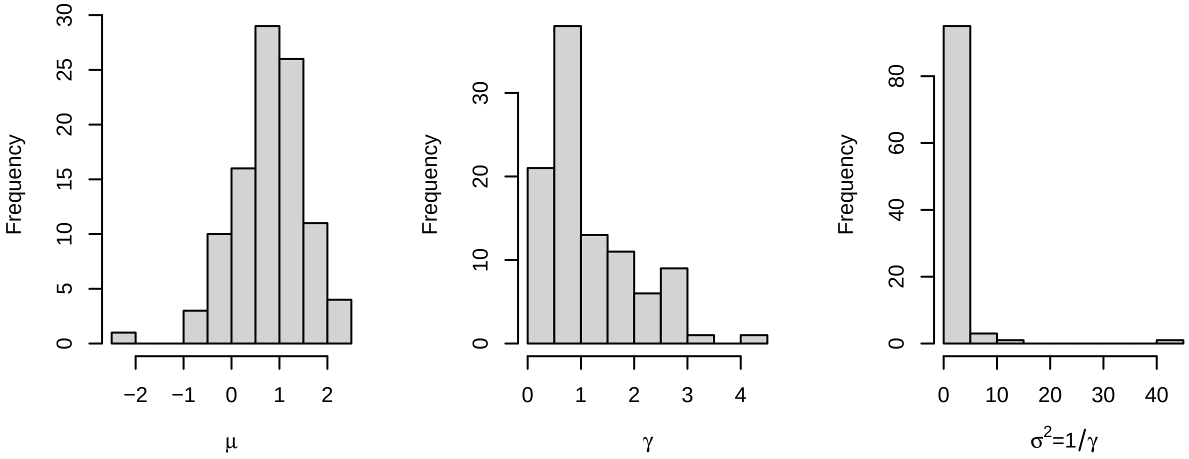
```r
# set up priors
m <-0
v <-1
a <-1
b <-1

# set up data
 y <- 2

# choose starting values by randomly sampling from our priors
# (this is just one possible way to choose starting values)
# (it's also useful to try out a few different starting values)
set.seed(1)
mu <- rnorm(1, mean = m, sd = sqrt(v))
gam <- rgamma(1, shape = a, rate = b)

# set up empty vectors to store samples
mus <- c()
gams <- c()

# store starting values in vectors of samples
mus[1] <- mu
gams[1] <- gam
```

```r
# choose number of iterations
# (we'll start with 100, but in practice you'd choose something much bigger)
N <-  100

# run through Gibbs Sampling for a total of N iterations
for(i in 2:N){
  # update mu
  numerator_for_mu <- y*gam + m/v
  denominator_for_mu <- gam + 1/v
  variance <- (1/denominator_for_mu)

  mu <- rnorm(n = 1, mean = (numerator_for_mu)/(denominator_for_mu), sd = sqrt(variance))

  # update gamma
  g1 <- (1.2) + a
  g2 <- (1/2)*(y-mu)^2 + b
  gam <- rgamma(n = 1, shape = g1, rate = g2)

  # store new samples
  mus[i] <- mu
  gams[i] <- gam
}
```

5. Look at a histogram of your posterior samples for μ, γ and $\sigma^2 = \frac{1}{\gamma}$.

ANSWER:

```
par(mfrow=c(1,3))
hist(mus, xlab = expression(mu), main = '')
hist(gams, xlab = expression(gamma), main = '')
hist(1/gams, xlab = expression(paste(sigma^2,'=',1/gamma)), main = '')
```



6. Estimate the posterior mean and median of μ. Find what the right parameters is for the mean and median functions below.

ANSWER:

```
# posterior mean
mean(mus)
```

```
## [1] 0.8248975
```

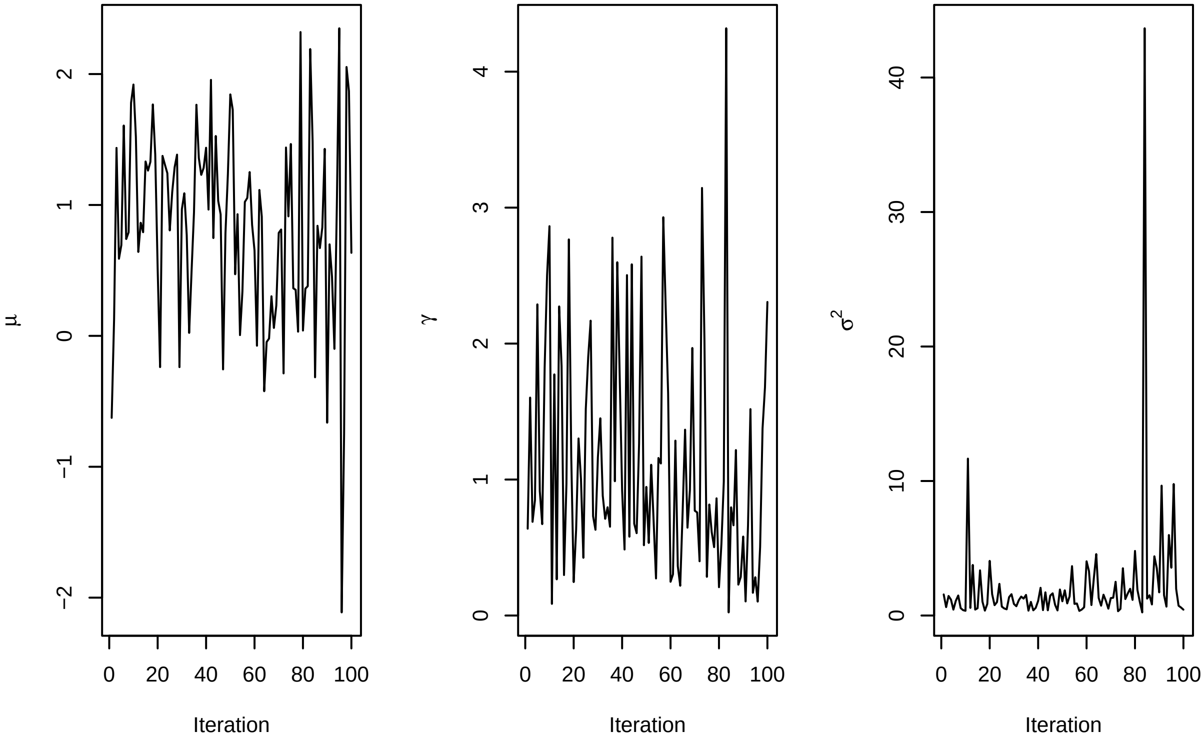
```r
# posterior median
median(mus)
```

```
## [1] 0.856095
```

7. Uncomment the code to create a *trace plot* showing the behavior of the samples over the N iterations.Describe what these trace plots tell us, do you think this is a good chain?

ANSWER:

```r
iterations <- 1:N

par(mfrow=c(1,3))
plot(mus ~ iterations, xlab = 'Iteration', ylab = expression(mu), type = 'l')
plot(gams ~ iterations, xlab = 'Iteration', ylab = expression(gamma), type = 'l')
plot(1/gams ~ iterations, xlab = 'Iteration', ylab = expression(sigma^2), type = 'l')
```



This is a good chain because we see that when touring the different values for our parameters we cover a lot of different values in the area of interest. Also, there are not trends, meaning that our chain is stable and knows what area to explore. Lastly, we don't see that there are values that are being sampled (toured) multiple times in a row, which would imply that we are oversampling certain values making our estimate non-proportional to the posterior.

8. As mentioned above, in practice we usually pick a burn-in period of initial iterations to remove. This decision is often motivated by the fact that, depending on your choice of starting value, it may take awhile for your chain of samples to look like it is "mixing" well. Play around with your choice of starting value above to see if you can find situations in which a burn-in period might be helpful.

References

Alicia A. Johnson, Mine Dogucu, Miles Q. Ott. 2022. *Bayes Rules!:an Introduction to Applied Bayesian Modeling*. Chapman; Hall/CRC.

Guo, Jonah Gabry, Jiqiang, and Sebastian Weber. 2020. "Rstan: R Interface to Stan."