Gibbs_Sampling_Activity

Ty Bruckner and Franco Salinas

```
library(rstan)
```

```
## Loading required package: StanHeaders

## Loading required package: ggplot2

## rstan (Version 2.21.3, 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)
```

```
library(bayesplot)
```

```
## This is bayesplot version 1.9.0

## - 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
```

Markov Chains Monte Carlo (MCMC)

MCMC is the application of Markov chains to simulate probability models. Two important characteristics are that MCMC samples aren't taken from the posterior pdf and that the samples aren't independent. The fact that the samples aren't independent reflects the "chain" fiture of the algorithm. For example in the $N - length$ MCMC sample (Markov chain) $\{\theta^{(1)}, \theta^{(2)}, ..., \theta^{(N)}\}$, when constructing the chain $\theta^{(2)}$ is drawn from some model that depends upon $\theta^{(1)}$, $\theta^{(3)}$ is drawn from some model that depends on $\theta^{(2)}$ and so on.

We can say that the (i+1)st chain value $\theta^{(i+1)}$ has a conditional pdf $f(\theta^{(i+1)}|\theta^{(i)}, y)$ is drawn from a model that depends on data y and the previous chain value $\theta^{(i)}$. It's important to note that by the Markov property, $\theta^{(i+1)}$ depends on the preceding chain values only through $\theta^{(i)}$, the most recent value. The only information we need to simulate $\theta^{(i+1)}$ is the value of $\theta^{(i)}$. Therefore, each value can be sampled from a different model, and none of these models are the target posterior. The pdf from which a Markov Chain value is simulated is not equivalent to the posterior pdf.

$$f(\theta^{(i+1)}|\theta^{(i)}, y) \neq f(\theta^{(i+1)}|y)$$

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 Beta-Binomial example.

###STEP 1: DEFINE the model

Data: Y is the observed number of success trials. We specify that Y is between 10 and 0. Parameters: The model depends on pi, therefore we must specify that pi can be any real number from 0 to 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. 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. To keep our random results constant we utilize the seed argument within the stan() function.

```
bb_sim <- stan(model_code = bb_model, data = list(Y = 9),
               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 {
```

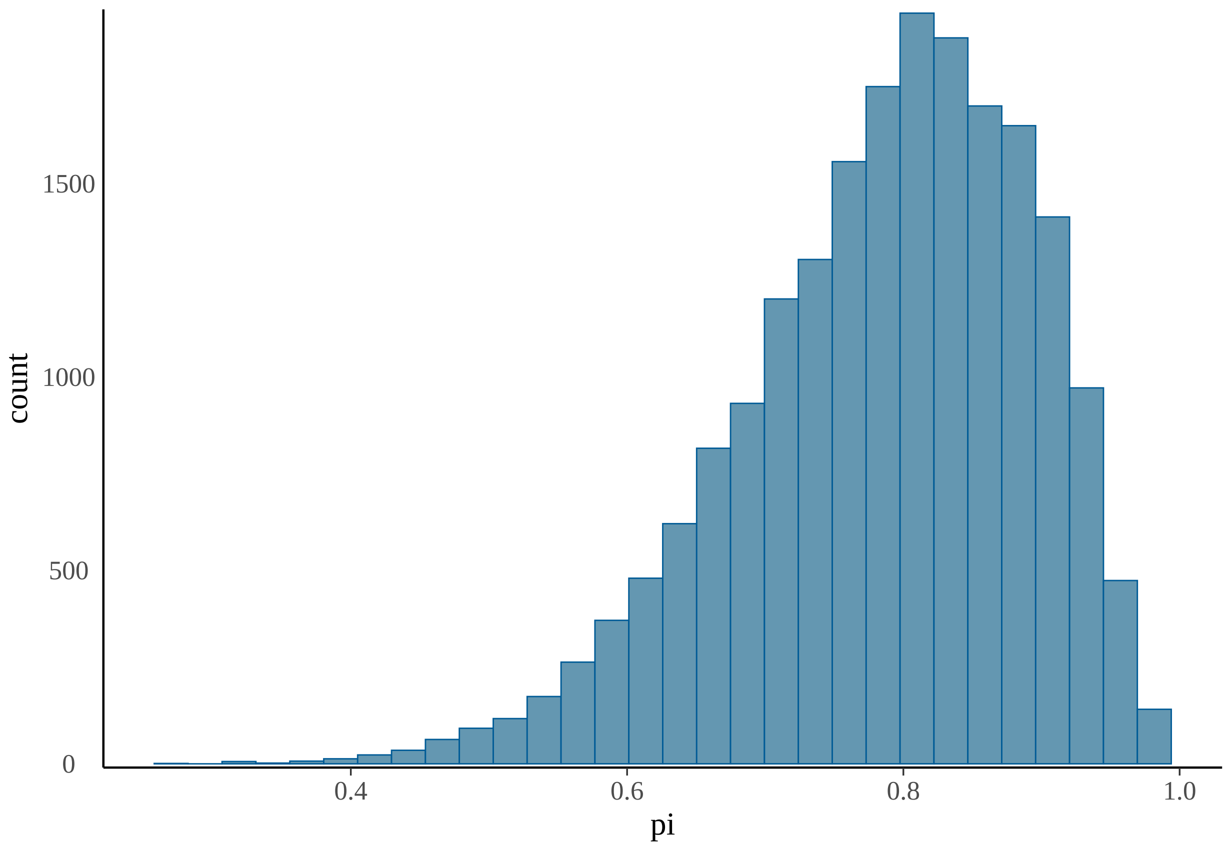
```
##                 ^
##                 ;
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/inclu
## /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/Core:96:10: f
## #include <complex>
##          ^~~~~~~~~
## 3 errors generated.
## make: *** [foo.o] Error 1
##
## SAMPLING FOR MODEL '7b64678a3565f32e51f77686e11b9c04' 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: 0.065823 seconds (Warm-up)
## Chain 1:                0.056285 seconds (Sampling)
## Chain 1:                0.122108 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: 0.054894 seconds (Warm-up)
## Chain 2:                0.056494 seconds (Sampling)
## Chain 2:                0.111388 seconds (Total)
## Chain 2: 
## 
## SAMPLING FOR MODEL '7b64678a3565f32e51f77686e11b9c04' NOW (CHAIN 3).
## Chain 3: 
## Chain 3: Gradient evaluation took 5e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.05 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: 0.057241 seconds (Warm-up)
## Chain 3:                0.073591 seconds (Sampling)
## Chain 3:                0.130832 seconds (Total)
## Chain 3: 
## 
## SAMPLING FOR MODEL '7b64678a3565f32e51f77686e11b9c04' NOW (CHAIN 4).
## Chain 4: 
## Chain 4: Gradient evaluation took 5e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.05 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: 0.066614 seconds (Warm-up)
## Chain 4:                0.051483 seconds (Sampling)
## Chain 4:                0.118097 seconds (Total)
## Chain 4:
```

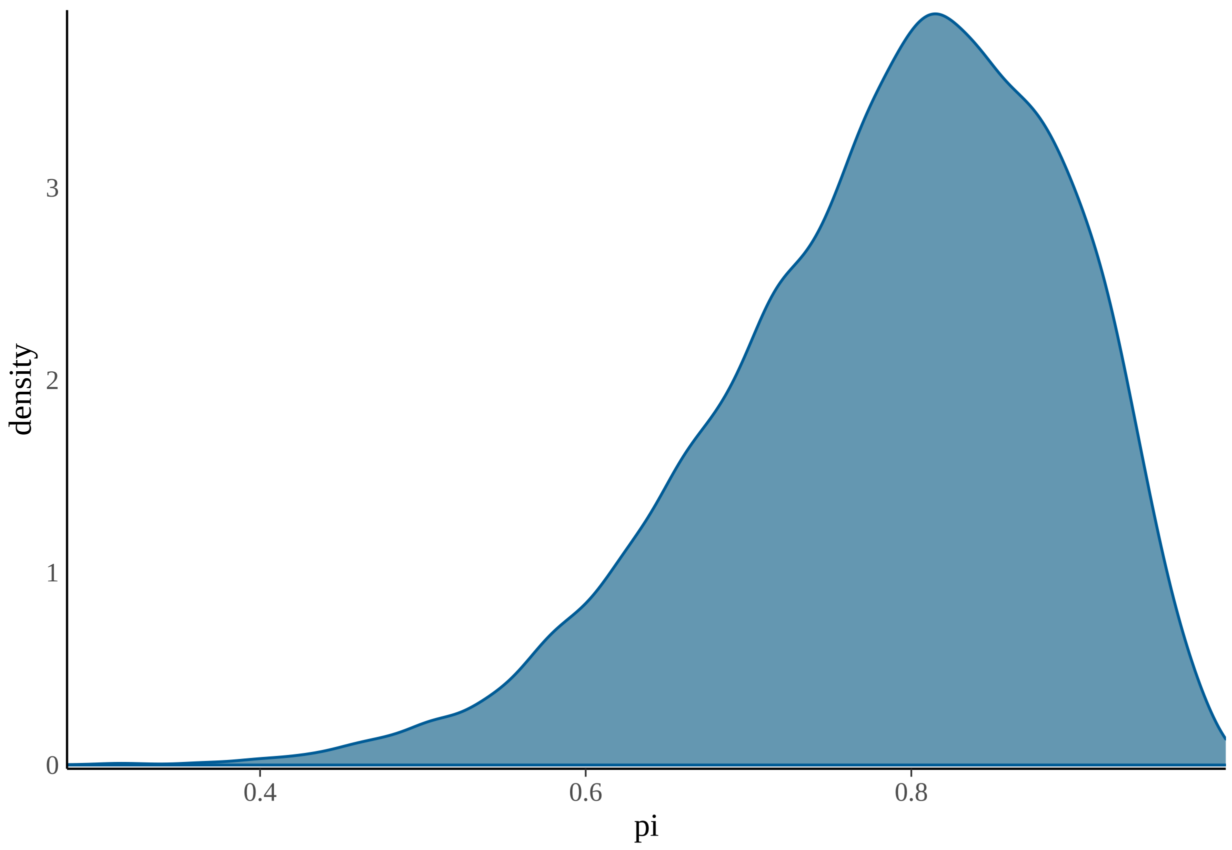
As you can see in Figure 1, when observing the distribution of the sampled π values we approximate the target Beta(11,3) posterior model of π. The target pdf is superimposed in black. (Alicia A. Johnson 2022)

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



Metropolis-Hastings algorithm

If we weren't able to recognize the posterior model of μ in a Normal-Normal model, we could approximate it using the MCMC simulation. Metropolis-Hastings algorithm helps automate the decision of what values of μ to sample and with what frequency. This algorithm iterates through a two step process. If we are in the location $\mu^{(i)} = \mu$ we select the next value to sample first by proposing a random location μ' and then we decide whether to stay at the current location or to stay at the current location $\mu^{(i+1)} = \mu$.

There are special cases of the Metropolis-Hastings that involve a different sampling decision criteria such as the Gibbs sampling, the Monte Carlo and the Metropolis algorithms. In this report we will be focusing on the Gibbs Sampling algorithm.

Gibbs Sampling

Now 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.

ANSWER:

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

A similar argument can be used to show $g(\gamma \mid y, \mu) \propto f(y|\mu, \gamma) f(\gamma)$.

3. Use this result to 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 $\gamma \mid y, \mu \sim \text{Gamma}\left(\frac{1}{2} + a, \frac{1}{2}(y-\mu)^2 + b\right)$.

ANSWER:

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

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

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

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

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

ANSWER:

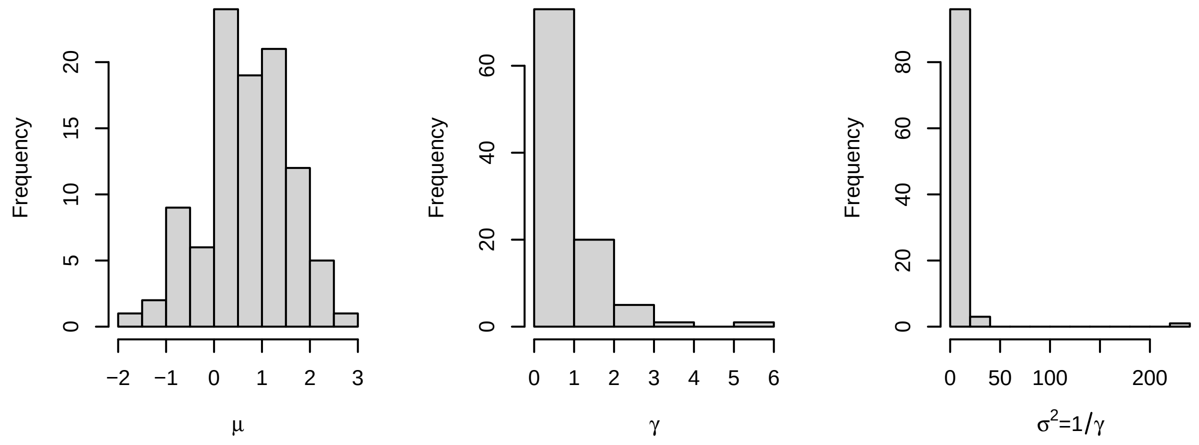
```r
# set up priors
m <- 0
v <- 1
a <- 1
b <- 1

# 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
  m1 <- y*gam + m/v
  m2 <- gam + 1/v
  mu <- rnorm(n = 1, mean = (m1)/(m2), sd = sqrt(1/m2))

  # update gamma
  g1 <- 0.5 + a
  g2 <- 0.5*(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:

```r
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.

ANSWER:

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

```
## [1] 0.6840056
```

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

```
## [1] 0.6525553
```

7. Find a 90% credible interval for μ, and estimate the probability that $\mu > 2$.

ANSWER:

```r
# 90% credible interval
quantile(mus, probs = c(0.05, 0.95))
```

```
##         5%        95%
## -0.8862182  2.0094145
```

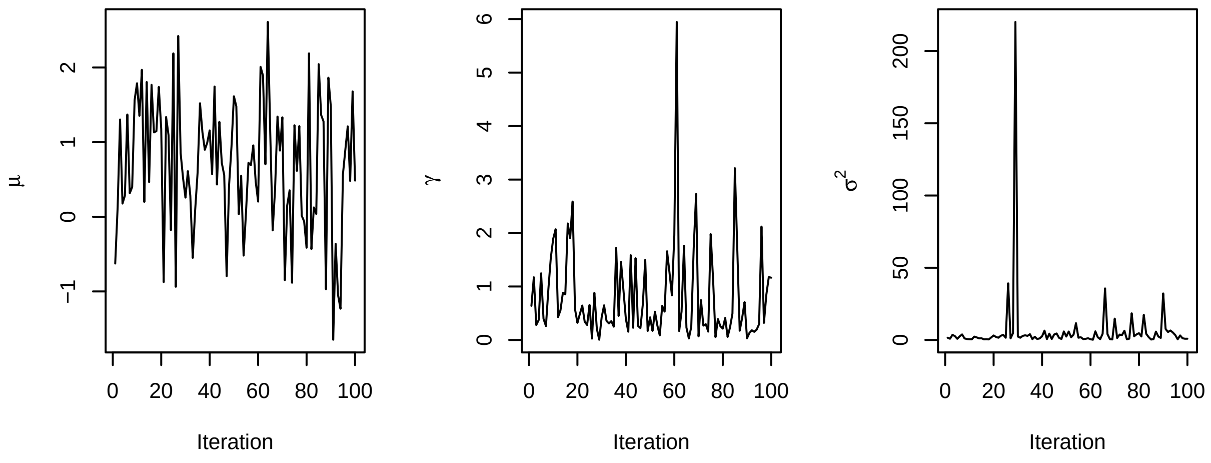
```r
# P(mu > 2 | y)
mean(mus > 2)
```

```
## [1] 0.06
```

8. Create a *trace plot* showing the behavior of the samples over the N iterations.

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')
```



9. 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.

Alicia A. Johnson, Mine Dogucu, Miles Q. Ott. 2022. "Bayes Rules!:an Introduction to Applied Bayesian Modeling."

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