Homework 4

Ginevra Carbone

Exercise 1

Above we found that the posterior mean is a weighted mean of the prior belief and the likelihood mean. Using some simple algebra, retrieve other two alternative expression and provide a nice interpretation.

$$\begin{aligned}
\mu^* &= \frac{n\tau^2\bar{y} + \sigma^2\mu}{n\tau^2 + \sigma^2} \\
&= \bar{y} + \frac{n\tau^2\bar{y} + \sigma^2\mu - n\tau^2\bar{y} - \sigma^2\bar{y}}{n\tau^2 + \sigma^2} \\
&= \bar{y} - (\bar{y} - \mu)\frac{\sigma^2}{n\tau^2 + \sigma^2}
\end{aligned}$$

Here the posterior mean is expressed as the sample mean plus an adjustment toward the prior mean. From this expression one can easily notice that, as previously pointed out, $\lim_{n\to\infty} \mu^* = \bar{y}$ and $\lim_{\tau\to 0} \mu^* = \mu$.

$$\begin{aligned}
\mu^* &= \frac{n\tau^2\bar{y} + \sigma^2\mu}{n\tau^2 + \sigma^2} \\
&= \mu + \frac{n\tau^2\bar{y} + \sigma^2\mu - n\tau^2\mu - \sigma^2\mu}{n\tau^2 + \sigma^2} \\
&= \mu + (\bar{y} - \mu)\frac{n\tau^2}{n\tau^2 + \sigma^2}
\end{aligned}$$

In this case the posterior mean is expressed as the prior mean plus an adjustment toward the sample mean.

Exercise 2

In `sim` in the code above, you find the MCMC output which allows for approximating the posterior distribution of our parameter of interest with S draws of θ. Please, produce an histogram for these random draws $\theta^{(1)}, \ldots, \theta^{(S)}$ compute the empirical quantiles, and overlap the true posterior distribution.

```r
#input values

#true mean
theta_sample <- 2
#likelihood variance
sigma2 <- 2
#sample size
n <- 10
#prior mean
mu <- 7
#prior variance
tau2 <- 2

#generate some data
set.seed(123)
y <- rnorm(n, theta_sample, sqrt(sigma2))
```

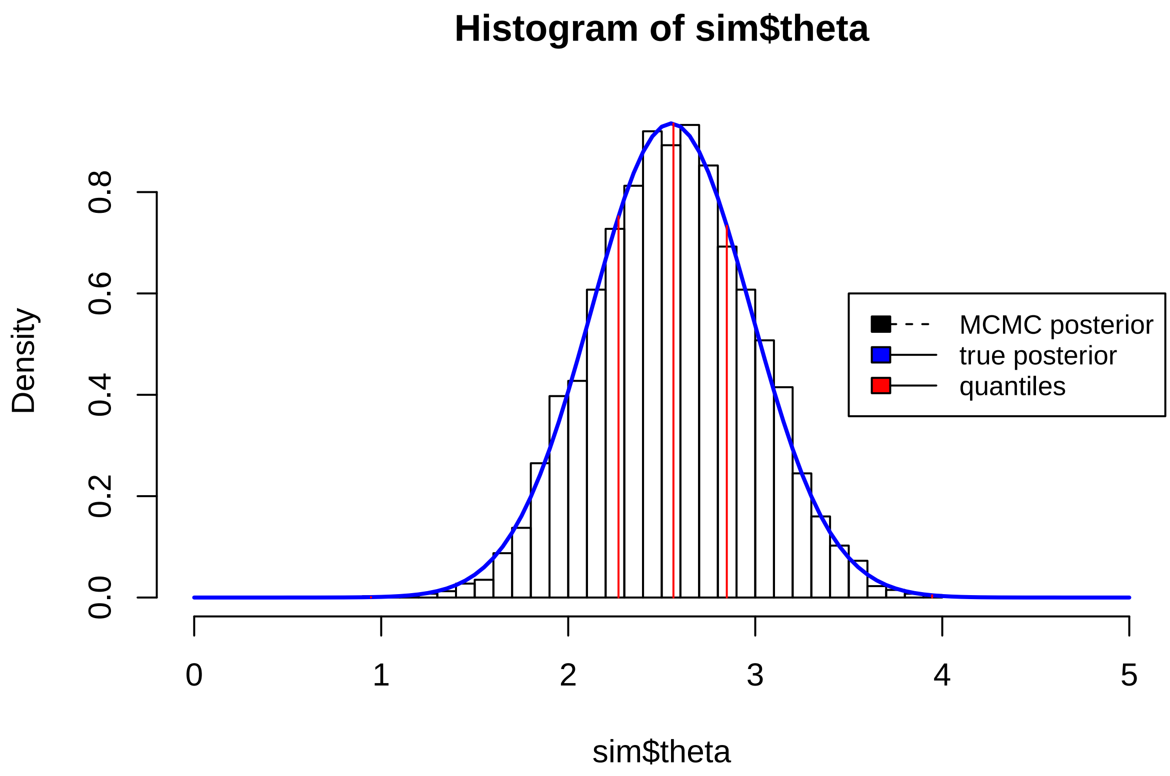
```r
#posterior mean
mu_star <- ((1/tau2)*mu+(n/sigma2)*mean(y))/((1/tau2)+(n/sigma2))

#posterior standard deviation
sd_star <- sqrt(1/( (1/tau2)+(n/sigma2)))

library(rstan)
data<- list(N=n, y=y, sigma =sqrt(sigma2), mu = mu, tau = sqrt(tau2))
fit <- stan(file="stan/normal.stan", data = data, chains = 4, iter=2000)
```

```
## In file included from /home/ginevracoal/R/x86_64-pc-linux-gnu-library/3.4/BH/include/boost/config.hpp
##                  from /home/ginevracoal/R/x86_64-pc-linux-gnu-library/3.4/BH/include/boost/math/tools
##                  from /usr/lib/R/site-library/StanHeaders/include/stan/math/rev/core/var.hpp:7,
##                  from /usr/lib/R/site-library/StanHeaders/include/stan/math/rev/core/gevv_vvv_vari.hp
##                  from /usr/lib/R/site-library/StanHeaders/include/stan/math/rev/core.hpp:12,
##                  from /usr/lib/R/site-library/StanHeaders/include/stan/math/rev/mat.hpp:4,
##                  from /usr/lib/R/site-library/StanHeaders/include/stan/math.hpp:4,
##                  from /usr/lib/R/site-library/StanHeaders/include/src/stan/model/model_header.hpp:4,
##                  from file47dc73651212.cpp:8:
## /home/ginevracoal/R/x86_64-pc-linux-gnu-library/3.4/BH/include/boost/config/compiler/gcc.hpp:186:0: w
##  #  define BOOST_NO_CXX11_RVALUE_REFERENCES
##  ^
## <command-line>:0:0: note: this is the location of the previous definition
##
## SAMPLING FOR MODEL 'normal' NOW (CHAIN 1).
##
## Gradient evaluation took 4e-06 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.04 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.006791 seconds (Warm-up)
##                0.006411 seconds (Sampling)
##                0.013202 seconds (Total)
##
##
## SAMPLING FOR MODEL 'normal' NOW (CHAIN 2).
##
## Gradient evaluation took 2e-06 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.02 seconds.
```

```
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.00639 seconds (Warm-up)
##                0.007528 seconds (Sampling)
##                0.013918 seconds (Total)
##
##
## SAMPLING FOR MODEL 'normal' NOW (CHAIN 3).
##
## Gradient evaluation took 2e-06 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.02 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.006403 seconds (Warm-up)
##                0.006052 seconds (Sampling)
##                0.012455 seconds (Total)
##
##
## SAMPLING FOR MODEL 'normal' NOW (CHAIN 4).
##
## Gradient evaluation took 2e-06 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.02 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
```

```
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.006365 seconds (Warm-up)
##                0.005831 seconds (Sampling)
##                0.012196 seconds (Total)
```

```r
#extract Stan output
sim <- extract(fit)

# MCMC posterior
hist(sim$theta, breaks=40, xlim = c(0,5), probability = TRUE)

# true posterior
curve(dnorm(x, mu_star, sd_star),
  xlab=expression(theta), ylab="", col="blue", lwd=2, add=T)

# compute empirical quantiles
qt <- quantile(sim$theta)
segments(qt, 0, qt, dnorm(qt, mu_star, sd_star),  col = 2)

legend(3.5, 0.6, c("MCMC posterior", "true posterior",
                   "quantiles"),
       c("black", "blue", "red" ),
       lty=c(2,1,1),lwd=c(1,1,1), cex=0.8)
```



Exercise 3

Suppose you receive $n = 15$ phone calls in a day, and you want to build a model for assessing their average length. Your likelihood for each call length is $y_i \sim Poisson(\lambda)$. Now, you have to choose the prior $\pi(\lambda)$. Please, tell which of these priors is adequate for describing the problem, and provide a short motivation for each of them:

1. $\pi(\lambda) = Beta(4, 2)$
2. $\pi(\lambda) = Normal(1, 2)$
3. $\pi(\lambda) = Gamma(4, 2)$

Now, compute your posterior as $\pi(\lambda|y) \propto L(\lambda; y)\pi(\lambda)$ for the selected prior. If your first choice was correct, you will be able to compute it analitically.

If $y_i \sim Poisson(\lambda)$, then a $Gamma$ prior on λ is a conjugate prior, because from

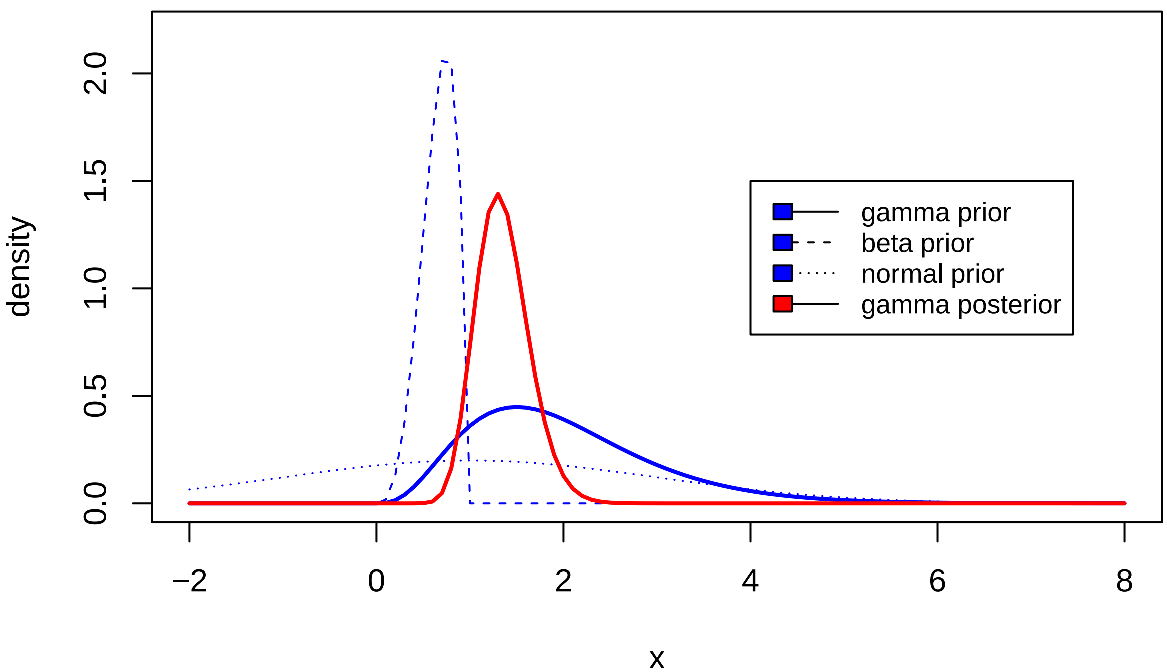
$$\pi(\lambda) = \frac{\beta^\alpha}{\Gamma(\alpha)}\lambda^{\alpha-1}e^{-\beta\lambda}$$

$$L(\lambda; y) = \frac{e^{-n\lambda}\lambda^{\sum y_i}}{\prod_{i=1}^{n}(y_i!)}$$

we have $\pi(\lambda|y) \propto Gamma(\sum y_i + \alpha, n + \beta)$. Therefore, the adequate prior is $Gamma(4, 2)$. We could also exclude the other two cases a priori, since a normal distribution also takes negative values, while a beta distribution is defined over the interval $[0, 1]$.

```
#generate some data from a poisson distribution
set.seed(123)
n = 15
lambda_sample = 1
```

```r
y <- rpois(n, lambda_sample)

# gamma prior
alpha = 4
beta = 2
curve(dgamma(x, alpha, beta), xlim=c(-2,8), col="blue",
      lty=1,lwd=2, ylim=c(0,2.2), ylab="density")

# beta prior
curve(dbeta(x, 4, 2), xlim=c(-2,8), col="blue", lty=2,
      lwd=1, add =T)

# normal prior
curve(dnorm(x, 1, 2), xlim=c(-2,8), col="blue", lty=3,
      lwd=1, add =T)

# gamma posterior
alpha_star = sum(y) + alpha
beta_star = n + beta
curve(dgamma(x, alpha_star, beta_star),
      xlab=expression(theta), ylab="", col="red", lwd=2, add=T)

legend(4, 1.5,
       c("gamma prior", "beta prior", "normal prior","gamma posterior"),
       c("blue", "blue","blue", "red" ),
       lty=c(1,2,3,1), lwd=c(1,1,1), cex=0.8)
```



Exercise 4

Open the file model called `biparametric.stan` *and replace the line* `target+=cauchy_lpdf(sigma|0,2.5);` *with the following one:* `target+=uniform_lpdf(sigma|0.1,10);`*.*

Which prior are you now assuming for your parameter σ? Reproduce the same plots as above and briefly comment.

```r
library("bayesplot")
library("ggplot2")

#input values

#true mean
theta_sample <- 2
#likelihood variance
sigma2 <- 2
#sample size
n <- 10
#prior mean
mu <- 7
#prior variance
tau2 <- 2

#generate some data
set.seed(123)
y <- rnorm(n,theta_sample, sqrt(sigma2))

#launch biparametric Stan model
data3<- list(N=n, y=y, a=-10, b=10)
fit3 <- stan(file="stan/biparametric.stan", data = data3,
             chains = 4, iter=2000, refresh=-1)
```

```
## In file included from /home/ginevracoal/R/x86_64-pc-linux-gnu-library/3.4/BH/include/boost/config.hpp
##                  from /home/ginevracoal/R/x86_64-pc-linux-gnu-library/3.4/BH/include/boost/math/tools
##                  from /usr/lib/R/site-library/StanHeaders/include/stan/math/rev/core/var.hpp:7,
##                  from /usr/lib/R/site-library/StanHeaders/include/stan/math/rev/core/gevv_vvv_vari.hp
##                  from /usr/lib/R/site-library/StanHeaders/include/stan/math/rev/core.hpp:12,
##                  from /usr/lib/R/site-library/StanHeaders/include/stan/math/rev/mat.hpp:4,
##                  from /usr/lib/R/site-library/StanHeaders/include/stan/math.hpp:4,
##                  from /usr/lib/R/site-library/StanHeaders/include/src/stan/model/model_header.hpp:4,
##                  from file3bcc61b9a269.cpp:8:
## /home/ginevracoal/R/x86_64-pc-linux-gnu-library/3.4/BH/include/boost/config/compiler/gcc.hpp:186:0: w
##  #  define BOOST_NO_CXX11_RVALUE_REFERENCES
##  ^
## <command-line>:0:0: note: this is the location of the previous definition
##
## Gradient evaluation took 4e-06 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.04 seconds.
## Adjust your expectations accordingly!
##
##
##
##  Elapsed Time: 0.010612 seconds (Warm-up)
##                0.009511 seconds (Sampling)
##                0.020123 seconds (Total)
##
##
## Gradient evaluation took 2e-06 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.02 seconds.
```

```
## Adjust your expectations accordingly!
##
##
##
##  Elapsed Time: 0.0124 seconds (Warm-up)
##                0.010863 seconds (Sampling)
##                0.023263 seconds (Total)
##
##
## Gradient evaluation took 3e-06 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.03 seconds.
## Adjust your expectations accordingly!
##
##
##
##  Elapsed Time: 0.01058 seconds (Warm-up)
##                0.019632 seconds (Sampling)
##                0.030212 seconds (Total)
##
##
## Gradient evaluation took 2e-06 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.02 seconds.
## Adjust your expectations accordingly!
##
##
##
##  Elapsed Time: 0.010301 seconds (Warm-up)
##                0.016268 seconds (Sampling)
##                0.026569 seconds (Total)
```

```
## Warning: There were 2 divergent transitions after warmup. Increasing adapt_delta above 0.8 may help.
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

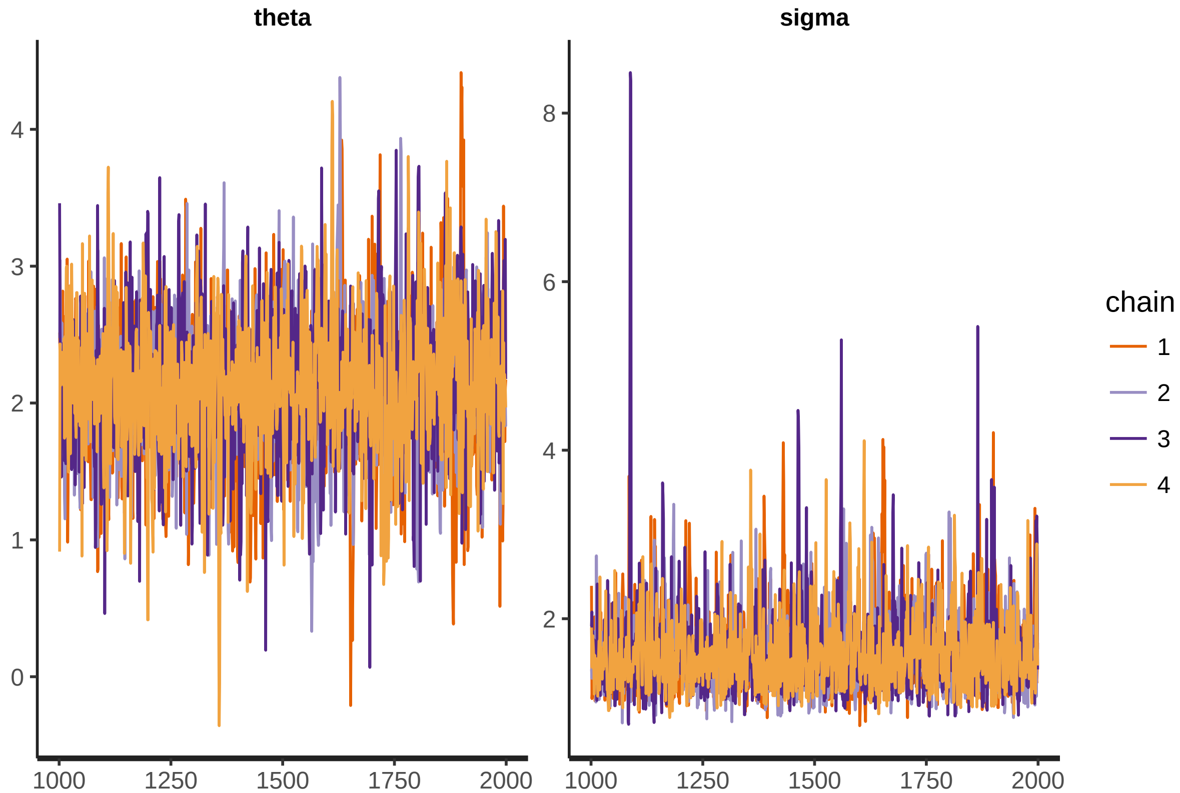
```
## Warning: There were 1 transitions after warmup that exceeded the maximum treedepth. Increase max_tre
## http://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

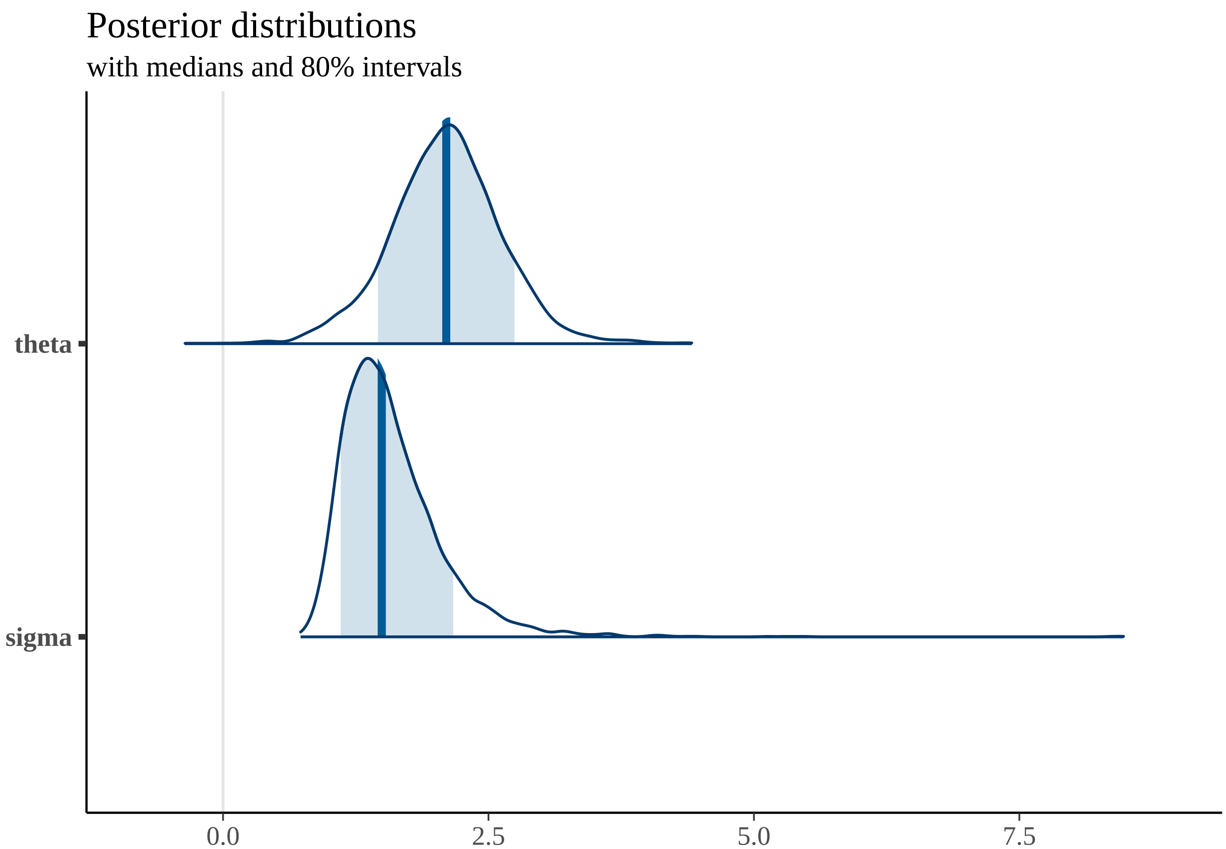
```r
#extract stan output for biparametric model
sim3 <- extract(fit3)
posterior_biv <- as.matrix(fit3)

theta_est <- mean(sim3$theta)
sigma_est <- mean(sim3$sigma)
c(theta_est, sigma_est)
```

```
## [1] 2.099920 1.588445
```

```r
traceplot(fit3, pars=c("theta", "sigma"))
```




```
plot_title <- ggtitle("Posterior distributions",
                      "with medians and 80% intervals")

mcmc_areas(posterior_biv,
           pars = c("theta","sigma"),
           prob = 0.8) + plot_title
```



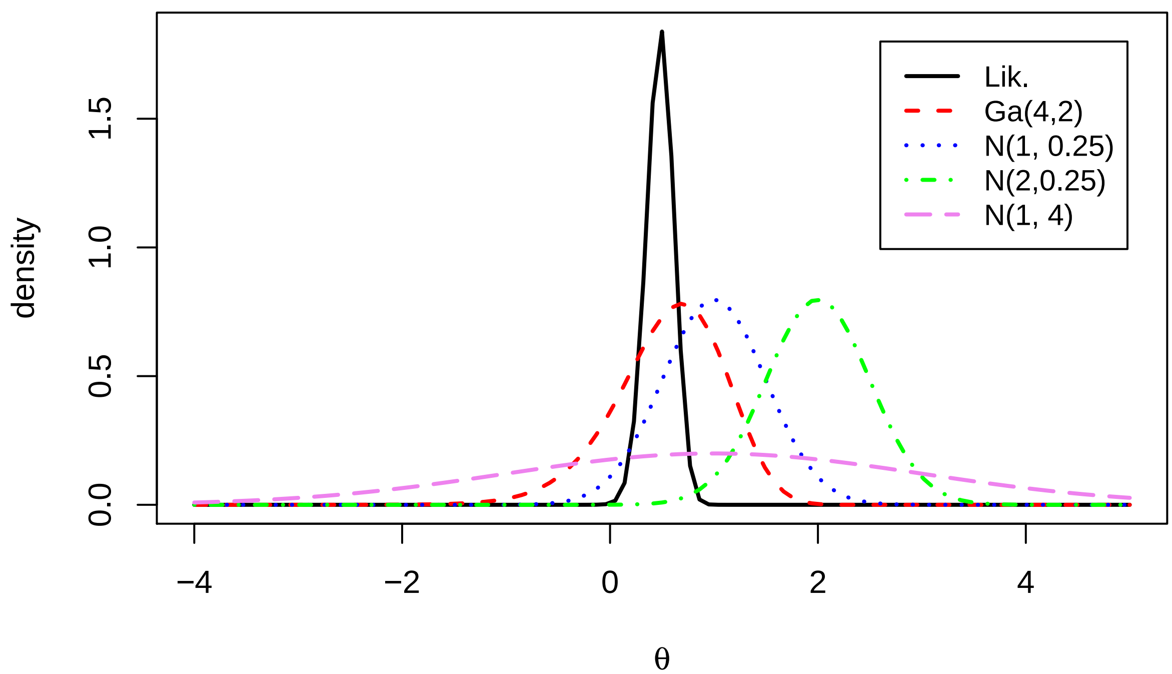
In this case we are assuming that $\sigma \sim Unif(0.1, 10)$, obtaining different mean and variance for both θ and σ posterior distributions.

Exercise 5

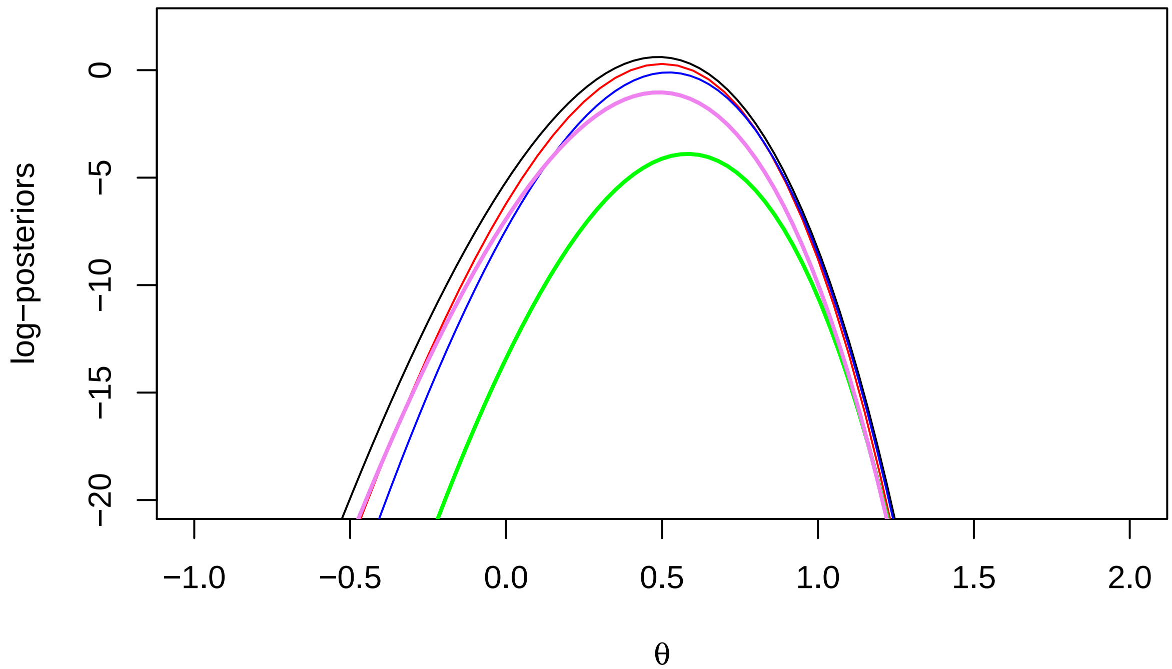
Reproduce the first plot above for the soccer goals, but this time by replacing Prior 1 with a `Gamma(2,4)`*. Then, compute the final Bayes factor matrix (*`BF_matrix`*) with this new prior and the other ones unchanged, and comment. Is still Prior 2 favorable over all the others?*

```
library(LearnBayes)
data(soccergoals)

y <- soccergoals$goals

#write the likelihood function via the gamma distribution
lik_pois<- function(data, theta){
  n <- length(data)
  lambda <- exp(theta)
  dgamma(lambda, shape =sum(data)+1, scale=1/n)
}

prior_gamma <- function(par, theta){
  lambda <- exp(theta)
  dgamma(lambda, par[1], rate=par[2])*lambda
}

prior_norm <- function(npar, theta){
```

```
  lambda=exp(theta)
  (dnorm(theta, npar[1], npar[2]))

}

lik_pois_v <- Vectorize(lik_pois, "theta")
prior_gamma_v <- Vectorize(prior_gamma, "theta")
prior_norm_v <- Vectorize(prior_norm, "theta")

#likelihood
curve(lik_pois_v(theta=x, data=y), xlim=c(-4,5),
      xlab=expression(theta), ylab = "density", lwd =2 )
#prior 1
curve(prior_gamma_v(theta=x, par=c(4, 2)), lty =2, col="red",
      add = TRUE, lwd =2)
#prior 2
curve(prior_norm_v(theta=x, npar=c(1, .5)), lty =3, col="blue",
      add =TRUE, lwd=2)
#prior 3
curve(prior_norm_v(theta=x, npar=c(2, .5)), lty =4, col="green",
      add =TRUE, lwd =2)
#prior 4
curve(prior_norm_v(theta=x, npar=c(1, 2)), lty =5, col="violet",
      add =TRUE, lwd =2)

legend(2.6, 1.8,
       c("Lik.", "Ga(4,2)", "N(1, 0.25)",
         "N(2,0.25)","N(1, 4)" ), lty=c(1,2,3,4,5),
       col=c("black", "red", "blue", "green", "violet"),
       lwd=2, cex=0.9)
```



```r
logpoissongamma <- function(theta, datapar){
   data <- datapar$data
   par <- datapar$par
   lambda <- exp(theta)
   log_lik <- log(lik_pois(data, theta))
   log_prior <- log(prior_gamma(par, theta))
   return(log_lik+log_prior)
}

logpoissongamma.v <- Vectorize( logpoissongamma, "theta")

logpoissonnormal <- function( theta, datapar){
 data <- datapar$data
 npar <- datapar$par
 lambda <- exp(theta)
 log_lik <- log(lik_pois(data, theta))
 log_prior <- log(prior_norm(npar, theta))
  return(log_lik+log_prior)
}
logpoissonnormal.v <- Vectorize( logpoissonnormal, "theta")

#log-likelihood
curve(log(lik_pois(y, theta=x)), xlim=c(-1,2),ylim=c(-20,2),
      lty =1, ylab="log-posteriors", xlab=expression(theta))
#log posterior 1
curve(logpoissongamma.v(theta=x, list(data=y, par=c(4, 2))),
      col="red", xlim=c(-1,4),ylim=c(-20,2), lty =1, add =TRUE)
#log posterior 2
curve(logpoissonnormal.v( theta=x, datapar <- list(data=y,
      par=c(1,.5))),
      lty =1, col="blue",  add =TRUE)
#log posterior 3
curve(logpoissonnormal.v( theta=x, datapar <- list(data=y,
      par=c(2, .5))), lty =1, col="green", add =TRUE, lwd =2)
#log posterior 4
curve(logpoissonnormal.v( theta=x, list(data=y, par=c(1, 2))),
      lty =1, col="violet", add =TRUE, lwd =2)

legend(2.6, 1.3, c( "loglik", "lpost 1", "lpost 2", "lpost 3",
                    "lpost 4" ),
       lty=1, col=c("black", "red", "blue", "green",
                    "violet"),lwd=2, cex=0.9)
```




```
datapar <- list(data=y, par=c(4, 2))
fit1 <- laplace(logpoissongamma, .5, datapar)
datapar <- list(data=y, par=c(1, .5))
fit2 <- laplace(logpoissonnormal, .5, datapar)
datapar <- list(data=y, par=c(2, .5))
fit3 <- laplace(logpoissonnormal, .5, datapar)
datapar <- list(data=y, par=c(1, 2))
fit4 <- laplace(logpoissonnormal, .5, datapar)

postmode <- c(fit1$mode, fit2$mode, fit3$mode, fit4$mode )
postsds <- sqrt(c(fit1$var, fit2$var, fit3$var, fit4$var))
logmarg <- c(fit1$int, fit2$int, fit3$int, fit4$int)
cbind(postmode, postsds, logmarg)
```

```
##       postmode   postsds    logmarg
## [1,] 0.4999512 0.1280372 -0.8440147
## [2,] 0.5207825 0.1260712 -1.2551710
## [3,] 0.5825195 0.1224723 -5.0763156
## [4,] 0.4899414 0.1320165 -2.1372163
```

```
BF_matrix <- matrix(1, 4,4)
for (i in 1:3){
  for (j in 2:4){
   BF_matrix[i,j]<- exp(logmarg[i]-logmarg[j])
   BF_matrix[j,i]=(1/BF_matrix[i,j])
  }
}

round_bf <- round(BF_matrix,3)
round_bf
```

```
##       [,1]  [,2]   [,3]  [,4]
## [1,] 1.000 1.509 68.876 3.644
```

```
## [2,] 0.663 1.000 45.656 2.416
## [3,] 0.015 0.022  1.000 0.053
## [4,] 0.274 0.414 18.899 1.000
```

I suspect there's an error in the assignment, since if I replace Prior 1 with a $Gamma(4, 2)$ (instead of $Gamma(2, 4)$) It becomes the favorable one over the others.

Exercise 6

Let $y = (1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0)$ collect the results of tossing $n = 14$ times an unfair coin, where 1 denotes heads and 0 crosses, and $p = Prob(y_i = 1)$.

- *Looking at the **Stan** code for the other models, write a short **Stan** Beta-Binomial model, where p has a $Beta(a, b)$ prior with $a = 3, b = 3$;*
- *Extract the posterior distribution with the function **extract()**;*
- *Compute analitically the posterior distribution and compare it with the **Stan** distribution.*

```
# data
y <- c(1,0,0,1,0,0,0,0,0,1,0,0,1,0)
#sample size
n <- length(y)
# success probability
sample_prob <- sum(y)/n

# prior
alpha <- 3
beta <- 3
curve(dbeta(x, alpha, beta), xlim=c(-0.5,1.5), ylim=c(0,4), col="red", lty=1,lwd=2, ylab="density")

#launch Stan model
data <- list(N=n, y=sum(y), alpha=3, beta=3)
fit <- stan(file="stan/beta_binomial.stan", data = data,
            chains = 4, iter=2000)
```

```
## In file included from /home/ginevracoal/R/x86_64-pc-linux-gnu-library/3.4/BH/include/boost/config.hpp
##                  from /home/ginevracoal/R/x86_64-pc-linux-gnu-library/3.4/BH/include/boost/math/tools
##                  from /usr/lib/R/site-library/StanHeaders/include/stan/math/rev/core/var.hpp:7,
##                  from /usr/lib/R/site-library/StanHeaders/include/stan/math/rev/core/gevv_vvv_vari.hp
##                  from /usr/lib/R/site-library/StanHeaders/include/stan/math/rev/core.hpp:12,
##                  from /usr/lib/R/site-library/StanHeaders/include/stan/math/rev/mat.hpp:4,
##                  from /usr/lib/R/site-library/StanHeaders/include/stan/math.hpp:4,
##                  from /usr/lib/R/site-library/StanHeaders/include/src/stan/model/model_header.hpp:4,
##                  from file4a0160716894.cpp:8:
## /home/ginevracoal/R/x86_64-pc-linux-gnu-library/3.4/BH/include/boost/config/compiler/gcc.hpp:186:0: w
##  #  define BOOST_NO_CXX11_RVALUE_REFERENCES
##  ^
## <command-line>:0:0: note: this is the location of the previous definition
##
## SAMPLING FOR MODEL 'beta_binomial' NOW (CHAIN 1).
## Rejecting initial value:
##   Error evaluating the log probability at the initial value.
## Exception: binomial_lpmf: Probability parameter is -1.10367, but must be in the interval [0, 1]  (in
##
## Rejecting initial value:
```

```
##   Error evaluating the log probability at the initial value.
## Exception: binomial_lpmf: Probability parameter is 1.05437, but must be in the interval [0, 1]  (in
##
##
## Gradient evaluation took 4e-06 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.04 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.015636 seconds (Warm-up)
##                0.01165 seconds (Sampling)
##                0.027286 seconds (Total)
##
##
## SAMPLING FOR MODEL 'beta_binomial' NOW (CHAIN 2).
## Rejecting initial value:
##   Error evaluating the log probability at the initial value.
## Exception: binomial_lpmf: Probability parameter is -1.68811, but must be in the interval [0, 1]  (in
##
##
## Gradient evaluation took 3e-06 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.03 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.01335 seconds (Warm-up)
##                0.010486 seconds (Sampling)
##                0.023836 seconds (Total)
```

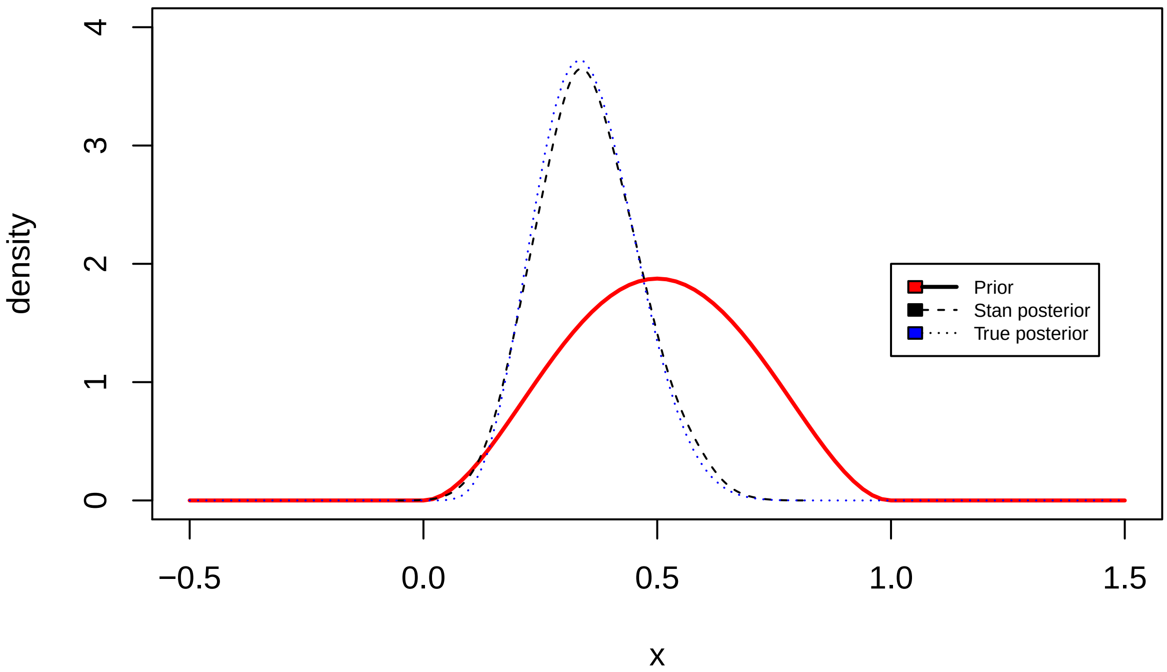
```
##
##
## SAMPLING FOR MODEL 'beta_binomial' NOW (CHAIN 3).
## Rejecting initial value:
##   Error evaluating the log probability at the initial value.
## Exception: binomial_lpmf: Probability parameter is -0.664732, but must be in the interval [0, 1]  (i
##
##
## Gradient evaluation took 3e-06 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.03 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.01165 seconds (Warm-up)
##                0.009774 seconds (Sampling)
##                0.021424 seconds (Total)
##
##
## SAMPLING FOR MODEL 'beta_binomial' NOW (CHAIN 4).
## Rejecting initial value:
##   Error evaluating the log probability at the initial value.
## Exception: binomial_lpmf: Probability parameter is 1.97672, but must be in the interval [0, 1]  (in
##
## Rejecting initial value:
##   Error evaluating the log probability at the initial value.
## Exception: binomial_lpmf: Probability parameter is -1.95316, but must be in the interval [0, 1]  (in
##
## Rejecting initial value:
##   Error evaluating the log probability at the initial value.
## Exception: binomial_lpmf: Probability parameter is 1.53795, but must be in the interval [0, 1]  (in
##
## Rejecting initial value:
##   Error evaluating the log probability at the initial value.
## Exception: binomial_lpmf: Probability parameter is -1.97424, but must be in the interval [0, 1]  (in
##
## Rejecting initial value:
##   Error evaluating the log probability at the initial value.
## Exception: binomial_lpmf: Probability parameter is -0.368488, but must be in the interval [0, 1]  (i
##
## Rejecting initial value:
##   Error evaluating the log probability at the initial value.
```

```
## Exception: binomial_lpmf: Probability parameter is -0.952131, but must be in the interval [0, 1]  (ir
##
##
## Gradient evaluation took 3e-06 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.03 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.014232 seconds (Warm-up)
##                0.00996 seconds (Sampling)
##                0.024192 seconds (Total)
```

```
## Warning: There were 13 divergent transitions after warmup. Increasing adapt_delta above 0.8 may help
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
#extract Stan output
sim <- extract(fit)

#stan simulated posterior
lines(density(sim$p, adj=2), col ="black", lwd=1, lty =2)

# true posterior
alpha_star <- alpha + sum(y)
beta_star <- beta + n - sum(y)
curve(dbeta(x, alpha_star , beta_star), lty=3, lwd=1,
      col="blue", add=T)

legend(1, 2, c("Prior", "Stan posterior", "True posterior"),
       c("red", "black", "blue"), lty=c(1,2,3),
       lwd=c(2,1,1), cex=0.6)
```


Prior
Stan posterior
True posterior
density
x
−0.5
0.0
0.5
1.0
1.5
0
1
2
3
4