Computational Statistics & Probability

Problem Set 4 - HMC and Generalized Linear Models

Author: Neelesh Bhalla
Collaborators: Nils Marthiensen, Chia-Jung Chang

2022-12-09

1. Log-odds

a) If an event has probability 0.3, what are the log-odds of this event?

```r
p_1a <- 0.3
# The odds are defined as the probability that the event will occur divided by
# the probability that the event will not occur
odds_1a <- p_1a / (1 - p_1a)
log_odds_1a <- log(odds_1a)
log_odds_1a
```

```
## [1] -0.8472979
```

b) If an event has log-odds of 1, what is the probability of that event?

```r
log_odds_1b <- 1
odds_1b <- exp(log_odds_1b)
# reversing the mathematical formula of odds to derive probability
p_1b <- odds_1b / (1 + odds_1b)
p_1b
```

```
## [1] 0.7310586
```

c) If a logistic regression coefficient has value -0.70, what does this imply about the proportional change in odds of the outcome? Briefly explain your answer.

```r
# By definition, the logistic regression coefficient (let's say  ) associated with a
# predictor (X) is the expected change in log odds of having the outcome per unit change in X.
# So increasing the predictor by 1 unit (or going from 1 level to the next) multiplies
# the odds of having the outcome by e^.

# In our case, (logistic regression coefficient)   = -0.70
# So every unit change in the predictor multiplies the odds of having the outcome
# by e^  = e^(-0.70) which is evaluated below

exp(-0.70)
```

```
## [1] 0.4965853
```

```r
# So every unit change in the predictor multiplies the odds of having the outcome by 0.4965853
```

2. HMC, Interactions and Robust Priors

Recall the interaction model m8.3, which is a varying-slope regression model assessing the effect of a country being inside or outside Africa on relationship between the ruggedness of its terrain and its GDP.

```
# The following code will load the data and reduce it down to cases (nations) that
# have the outcome variable of interest
library(rethinking)
```

```
## Loading required package: rstan

## Loading required package: StanHeaders

##
## rstan version 2.26.13 (Stan version 2.26.1)

## 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)
## For within-chain threading using `reduce_sum()` or `map_rect()` Stan functions,
## change `threads_per_chain` option:
## rstan_options(threads_per_chain = 1)

## Loading required package: cmdstanr

## This is cmdstanr version 0.5.3

## - CmdStanR documentation and vignettes: mc-stan.org/cmdstanr

## - CmdStan path: /Users/neelesh/.cmdstan/cmdstan-2.30.1

## - CmdStan version: 2.30.1

##
## A newer version of CmdStan is available. See ?install_cmdstan() to install it.
## To disable this check set option or environment variable CMDSTANR_NO_VER_CHECK=TRUE.

## Loading required package: parallel

## rethinking (Version 2.21)

##
## Attaching package: 'rethinking'

## The following object is masked from 'package:rstan':
##
##     stan

## The following object is masked from 'package:stats':
##
##     rstudent
```

```
data(rugged)
d <- rugged
d$log_gdp <- log(d$rgdppc_2000)
dd <- d[ complete.cases(d$rgdppc_2000) , ]
dd$log_gdp_std <- dd$log_gdp / mean(dd$log_gdp)
dd$rugged_std <- dd$rugged / max(dd$rugged)
dd$cid <- ifelse( dd$cont_africa==1 , 1 , 2 )
```

```
# This model aims to predict log GDP with terrain ruggedness, continent, and the
# interaction of the two (making use of quap)
```

```
m8.3 <- quap(
alist(
log_gdp_std ~ dnorm(mu, sigma),
mu <- a[cid] + b[cid]*(rugged_std - 0.215) ,
a[cid] ~ dnorm(1, 0.1),
b[cid] ~ dnorm(0, 0.3),
sigma ~ dexp(1)
), data = dd)
precis( m8.3 , depth=2 )
```

```
##             mean          sd        5.5%       94.5%
## a[1]   0.8865640 0.015675785  0.86151108  0.91161694
## a[2]   1.0505697 0.009936664  1.03468896  1.06645037
## b[1]   0.1325065 0.074204851  0.01391282  0.25110019
## b[2]  -0.1425755 0.054749715 -0.23007611 -0.05507487
## sigma  0.1094948 0.005935386  0.10000887  0.11898065
```

a) Now fit this same model using Hamiltonian Monte Carlo (HMC). The code to do this is in the book, beginning with R code 9.13. You should use the ulam convenience function provided by the rethinking package.

```
# Presenting a slim list of the variables that will be used
dat_slim <- list(
log_gdp_std = dd$log_gdp_std,
rugged_std = dd$rugged_std,
cid = as.integer( dd$cid )
)
str(dat_slim)
```

```
## List of 3
##  $ log_gdp_std: num [1:170] 0.88 0.965 1.166 1.104 0.915 ...
##  $ rugged_std : num [1:170] 0.138 0.553 0.124 0.125 0.433 ...
##  $ cid        : int [1:170] 1 2 2 2 2 2 2 2 2 1 ...
```

```
# Sampling from the posterior. Getting samples from the posterior distribution with this code:
m2a_1c <- ulam(
alist(
log_gdp_std ~ dnorm( mu , sigma ) ,
mu <- a[cid] + b[cid]*( rugged_std - 0.215 ) ,
a[cid] ~ dnorm( 1 , 0.1 ) ,
b[cid] ~ dnorm( 0 , 0.3 ) ,
sigma ~ dexp( 1 )
) , data=dat_slim , chains=1 )
```

```
## Warning in '/var/folders/wx/1_76tj0s15gc4yxmndvw4l_00000gn/T/RtmpGmhclW/model-9fc066600f0a.stan', lin
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc
```

```
## Running MCMC with 1 chain, with 1 thread(s) per chain...
##
## Chain 1 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 1 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1 Iteration: 800 / 1000 [ 80%]  (Sampling)

## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected because of th

## Chain 1 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/var/folders/wx/1_7

## Chain 1 If this warning occurs sporadically, such as for highly constrained variable types like cova

## Chain 1 but if this warning occurs often then your model may be either severely ill-conditioned or m

## Chain 1

## Chain 1 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1 finished in 0.1 seconds.
```

```
# 'show' tells us about the model formula and also about how long each chain took to run
show( m2a_1c )
```

```
## Hamiltonian Monte Carlo approximation
## 500 samples from 1 chain
##
## Sampling durations (seconds):
##         warmup sample total
## chain:1   0.07   0.05  0.12
##
## Formula:
## log_gdp_std ~ dnorm(mu, sigma)
## mu <- a[cid] + b[cid] * (rugged_std - 0.215)
## a[cid] ~ dnorm(1, 0.1)
## b[cid] ~ dnorm(0, 0.3)
## sigma ~ dexp(1)
```

```
# The estimates are very similar to the quadratic approximation as observed above in this
# very question itself
precis( m2a_1c , depth=2 )
```

```
##             mean          sd        5.5%       94.5%     n_eff     Rhat4
## a[1]   0.8867281 0.015496291  0.86165442  0.91246430  545.0963 1.0016816
## a[2]   1.0505379 0.009990952  1.03463685  1.06612435 1018.0751 0.9979980
## b[1]   0.1343638 0.076017800  0.01018565  0.25253980  497.7832 1.0013727
## b[2]  -0.1401292 0.057636380 -0.22926991 -0.04442097  717.2951 0.9986055
## sigma  0.1115824 0.006183011  0.10169845  0.12207975  981.4474 1.0008789
```

```
# n_eff column provide diagnostic criteria, to help quantify how well the sampling worked.
# n_eff is a crude estimate of the number of independent samples the model managed to get

# Rhat is a complicated estimate of the convergence of the Markov chains to the target
# distribution. For this model, as desired, it approaches 1.00 from above, when all is well.
```

```
# Now sampling again but this time in parallel. Running all 4 chains at the same time,
# instead of in sequence.
m2a_4c <- ulam(
alist(
log_gdp_std ~ dnorm( mu , sigma ) ,
mu <- a[cid] + b[cid]*( rugged_std - 0.215 ) ,
a[cid] ~ dnorm( 1 , 0.1 ) ,
b[cid] ~ dnorm( 0 , 0.3 ) ,
sigma ~ dexp( 1 )
) , data=dat_slim , chains=4 , cores=4 )
```

```
## Warning in '/var/folders/wx/1_76tj0s15gc4yxmndvw4l_00000gn/T/RtmpGmhclW/model-9fc0671bbbb3.stan', li
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc

## Running MCMC with 4 parallel chains, with 1 thread(s) per chain...
##
## Chain 1 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 1 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1 Iteration: 501 / 1000 [ 50%]  (Sampling)

## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected because of th

## Chain 1 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/var/folders/wx/1_7

## Chain 1 If this warning occurs sporadically, such as for highly constrained variable types like cova

## Chain 1 but if this warning occurs often then your model may be either severely ill-conditioned or m

## Chain 1

## Chain 2 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2 Iteration: 300 / 1000 [ 30%]  (Warmup)

## Chain 2 Informational Message: The current Metropolis proposal is about to be rejected because of th

## Chain 2 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/var/folders/wx/1_7

## Chain 2 If this warning occurs sporadically, such as for highly constrained variable types like cova

## Chain 2 but if this warning occurs often then your model may be either severely ill-conditioned or m

## Chain 2

## Chain 3 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3 Iteration: 300 / 1000 [ 30%]  (Warmup)

## Chain 3 Informational Message: The current Metropolis proposal is about to be rejected because of th

## Chain 3 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/var/folders/wx/1_7

## Chain 3 If this warning occurs sporadically, such as for highly constrained variable types like cova

## Chain 3 but if this warning occurs often then your model may be either severely ill-conditioned or m

## Chain 3
## Chain 4 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 4 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 4 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 4 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 4 Iteration: 400 / 1000 [ 40%]  (Warmup)
```

```
## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected because of the
```

```
## Chain 4 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/var/folders/wx/1_7
```

```
## Chain 4 If this warning occurs sporadically, such as for highly constrained variable types like cova
```

```
## Chain 4 but if this warning occurs often then your model may be either severely ill-conditioned or m
```

```
## Chain 4
```

```
## Chain 1 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 2 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 2 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 2 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 2 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 3 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 3 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 3 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 3 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 3 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 4 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 4 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 4 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 4 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 4 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1 finished in 0.3 seconds.
## Chain 2 finished in 0.3 seconds.
## Chain 3 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 4 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 3 finished in 0.3 seconds.
## Chain 4 finished in 0.3 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.3 seconds.
## Total execution time: 0.5 seconds.
```

```r
# 'show' tells us about the model formula and also about how long each chain took to run
show( m2a_4c )
```

```
## Hamiltonian Monte Carlo approximation
## 2000 samples from 4 chains
```

```
##
## Sampling durations (seconds):
##         warmup sample total
## chain:1   0.14   0.13  0.26
## chain:2   0.18   0.08  0.27
## chain:3   0.17   0.10  0.27
## chain:4   0.17   0.09  0.26
##
## Formula:
## log_gdp_std ~ dnorm(mu, sigma)
## mu <- a[cid] + b[cid] * (rugged_std - 0.215)
## a[cid] ~ dnorm(1, 0.1)
## b[cid] ~ dnorm(0, 0.3)
## sigma ~ dexp(1)
```

```
# There were 2000 samples from all 4 chains, because each 1000 sample chain uses by default
# the fist half of the samples to adapt
```

```
precis( m2a_4c , 2 )
```

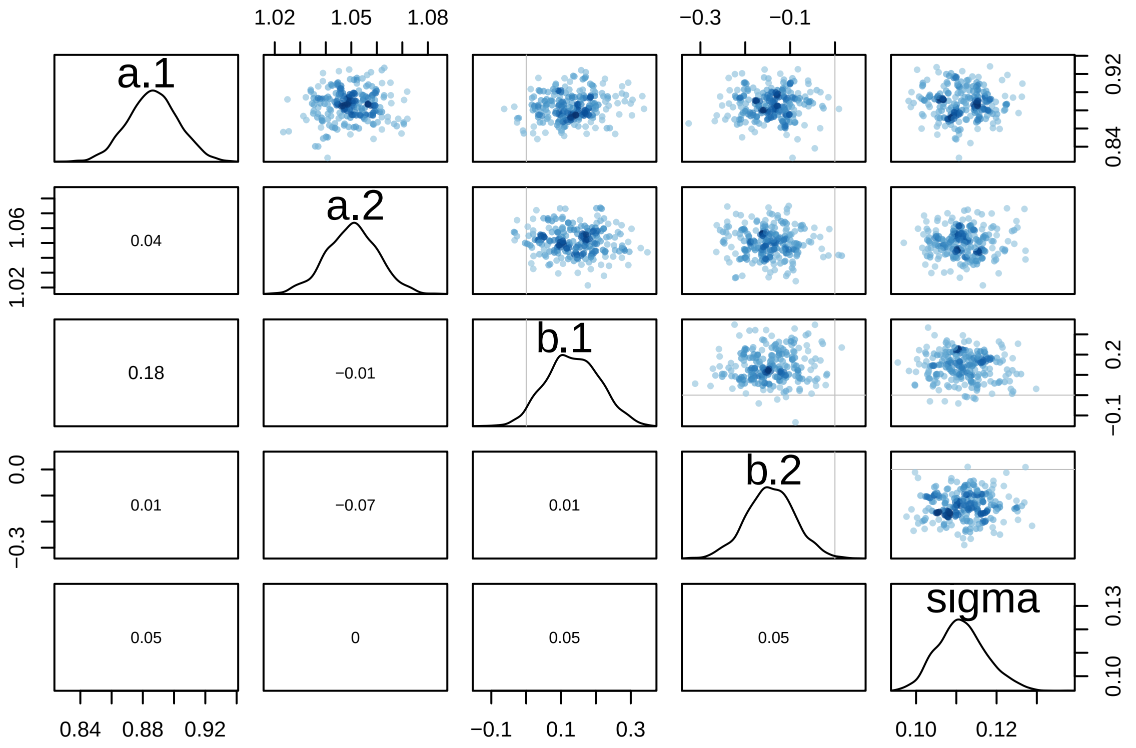
```
##              mean         sd        5.5%       94.5%    n_eff     Rhat4
## a[1]    0.8865176 0.01610918  0.86107339  0.91237403 2514.805 0.9998059
## a[2]    1.0503422 0.01002216  1.03408230  1.06610495 3629.352 0.9985990
## b[1]    0.1344924 0.07657230  0.01393065  0.25810567 2880.408 1.0004629
## b[2]   -0.1429057 0.05441947 -0.23150839 -0.05409341 2336.317 1.0000203
## sigma   0.1114291 0.00611479  0.10219773  0.12201222 2557.027 0.9991181
```

```
# It is intriguing that we have more than 2000 effective samples for each parameter.
# The book explains that "the adaptive sampler that Stan uses is so good, it can
# actually produce sequential samples that are better than uncorrelated. They are
# anti-correlated. This means it can explore the posterior distribution so efficiently
# that it can beat random."
```

b) Check your chains with traceplots and tankplots. Interpret these graphs to explain why, or why not, your HMC model is suitable for inference.

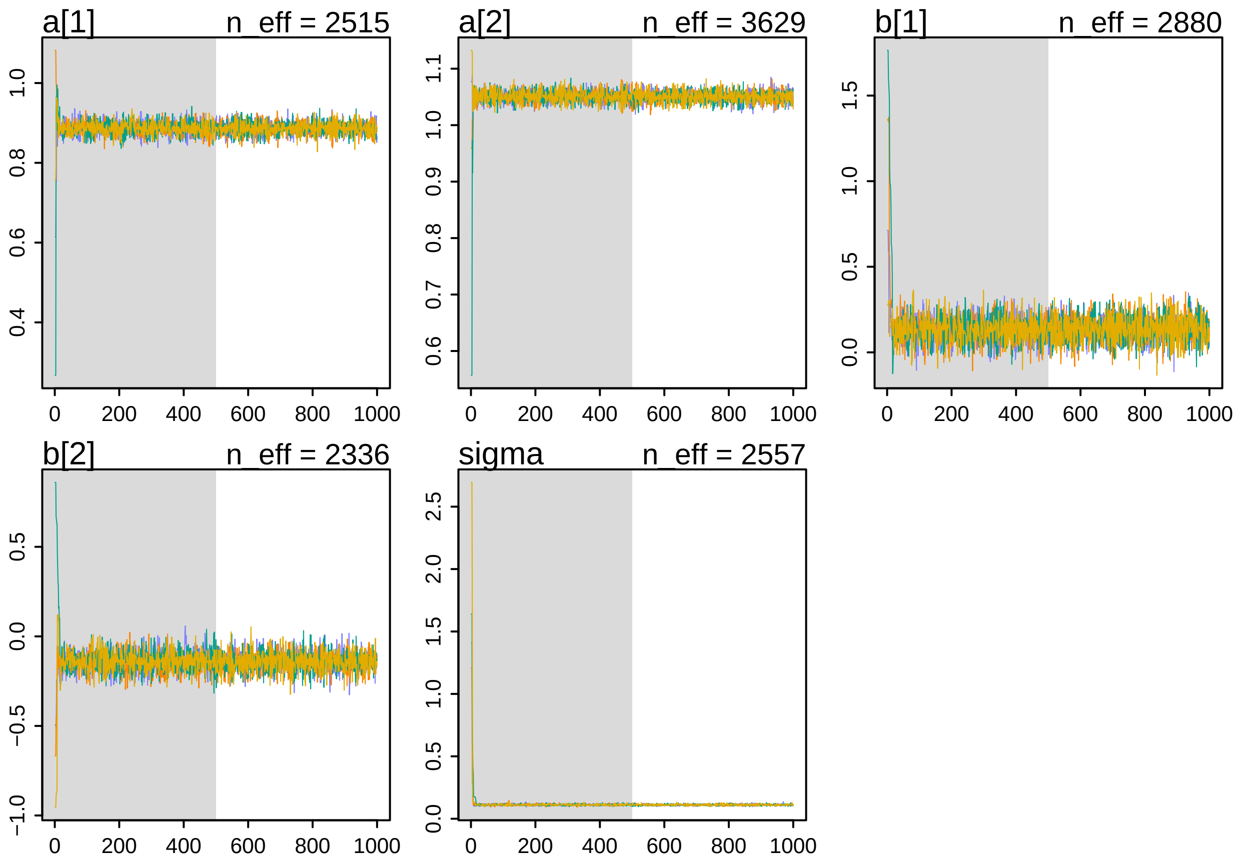
```
# Using 'pairs' directly on the model object, so that R knows to display parameter names
# and parameter correlations:
pairs( m2a_4c )
```

```
## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter
```

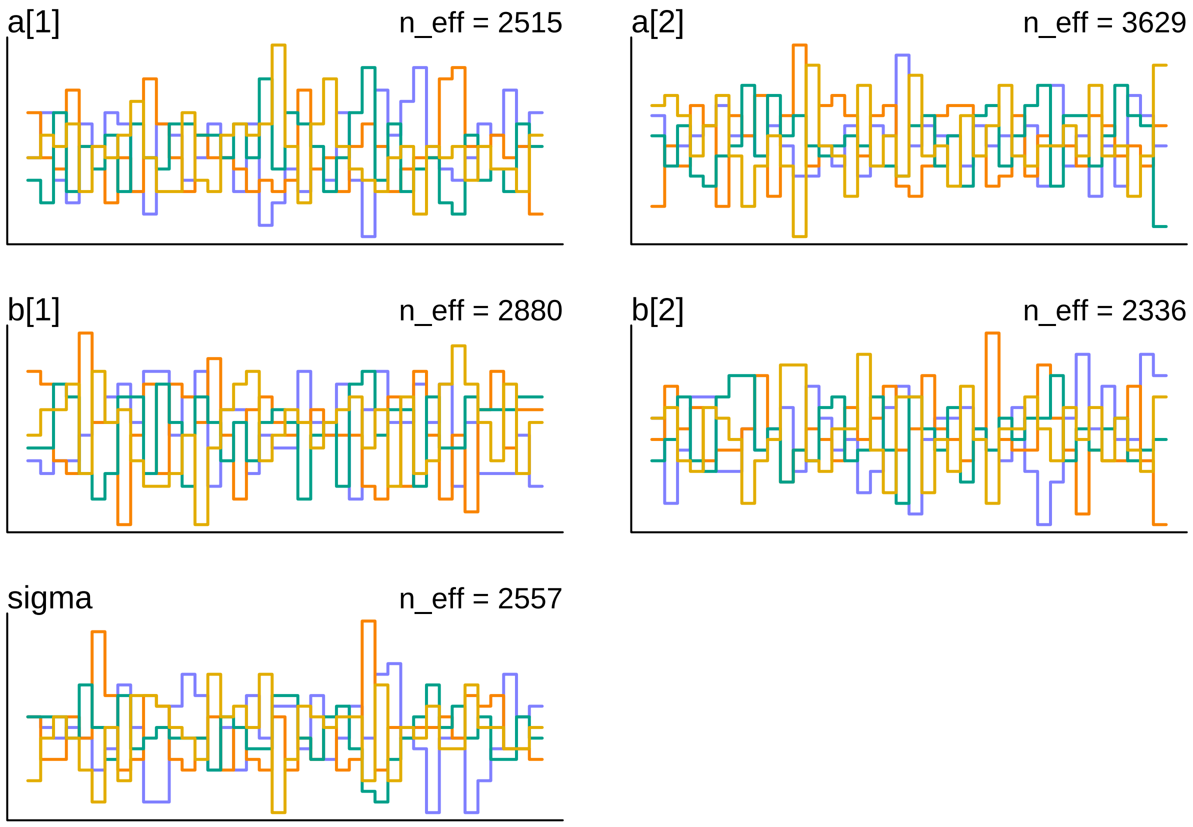


```
# This is a matrix of bivariate scatter plots.
# The smoothed histogram of each parameter is shown along the diagonal with its name.
# And the correlation between each pair of parameters is shown in the lower triangle
# of the matrix

# For this model and these data, the resulting posterior distribution is quite nearly
# multivariate Gaussian. The density for sigma is skewed in the expected direction.
# But otherwise the quadratic approximation does almost as well as Hamiltonian
# Monte Carlo.
```

```
traceplot( m2a_4c )
# This is a clean, healthy Markov chain - stationary, wellmixing and converged. The gray
# region is warmup (500 samples), during which the Markov chain was adapting to improve sampling
# efficiency. The white region contains the samples used for inference.
```



```
trankplot( m2a_4c, n_cols=2 )
# The horizontal axis is rank, from 1 to the number of samples across all chains (2000 in
# this example). The vertical axis is the frequency of ranks in each bin of the histogram.
# This trank plot is what we hope for: Histograms that overlap and stay within the same range.
```



c) Now fit your HMC model with a flat prior for sigma, sigma ~ dunif(0,1). What effect does this prior have on your posterior distribution? Explain your answer.

```
# fitting HMC with a flat prior for sigma
m2c <- ulam(
alist(
log_gdp_std ~ dnorm( mu , sigma ) ,
mu <- a[cid] + b[cid]*( rugged_std - 0.215 ) ,
a[cid] ~ dnorm( 1 , 0.1 ) ,
b[cid] ~ dnorm( 0 , 0.3 ) ,
sigma ~ dunif(0,1)
) , data=dat_slim , chains=4 , cores=4 )
```

```
## Warning in '/var/folders/wx/1_76tj0s15gc4yxmndvw4l_00000gn/T/RtmpGmhclW/model-9fc07617e3fb.stan', li
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc

## Running MCMC with 4 parallel chains, with 1 thread(s) per chain...
##
## Chain 1 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1 Iteration: 400 / 1000 [ 40%]  (Warmup)

## Chain 1 Informational Message: The current Metropolis proposal is about to be rejected because of th

## Chain 1 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/var/folders/wx/1_7
## Chain 1 If this warning occurs sporadically, such as for highly constrained variable types like cova

## Chain 1 but if this warning occurs often then your model may be either severely ill-conditioned or m

## Chain 1

## Chain 2 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 3 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 4 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 4 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 4 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 4 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 2 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 2 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 2 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 3 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 3 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 3 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 3 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 3 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 4 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 4 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 4 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 4 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 4 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1 finished in 0.3 seconds.
## Chain 2 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 3 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 4 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 4 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 2 finished in 0.3 seconds.
## Chain 3 finished in 0.3 seconds.
## Chain 4 finished in 0.3 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.3 seconds.
## Total execution time: 0.4 seconds.
```

```
show( m2c )
```

```
## Hamiltonian Monte Carlo approximation
## 2000 samples from 4 chains
##
## Sampling durations (seconds):
##         warmup sample total
## chain:1   0.16   0.11  0.28
## chain:2   0.17   0.11  0.29
## chain:3   0.19   0.10  0.28
## chain:4   0.18   0.09  0.28
##
## Formula:
## log_gdp_std ~ dnorm(mu, sigma)
## mu <- a[cid] + b[cid] * (rugged_std - 0.215)
## a[cid] ~ dnorm(1, 0.1)
## b[cid] ~ dnorm(0, 0.3)
## sigma ~ dunif(0, 1)
```

```
precis( m2c , 2 )
```

```
##             mean          sd        5.5%       94.5%    n_eff     Rhat4
## a[1]   0.8869145 0.016499281  0.86084295  0.91361731 2955.811 0.9988520
## a[2]   1.0505064 0.010122724  1.03399000  1.06656055 2784.535 1.0005109
## b[1]   0.1327646 0.075287181  0.01309351  0.25461736 2286.994 0.9994760
## b[2]  -0.1398656 0.054108173 -0.22651689 -0.05449416 2323.428 0.9986945
## sigma  0.1118889 0.006023511  0.10270167  0.12177111 2716.091 0.9984436
```

```
# here, we can observe that the distribution does not change much essentially. However,
# it will be more clear when we visualize the posterior distributions plot
```

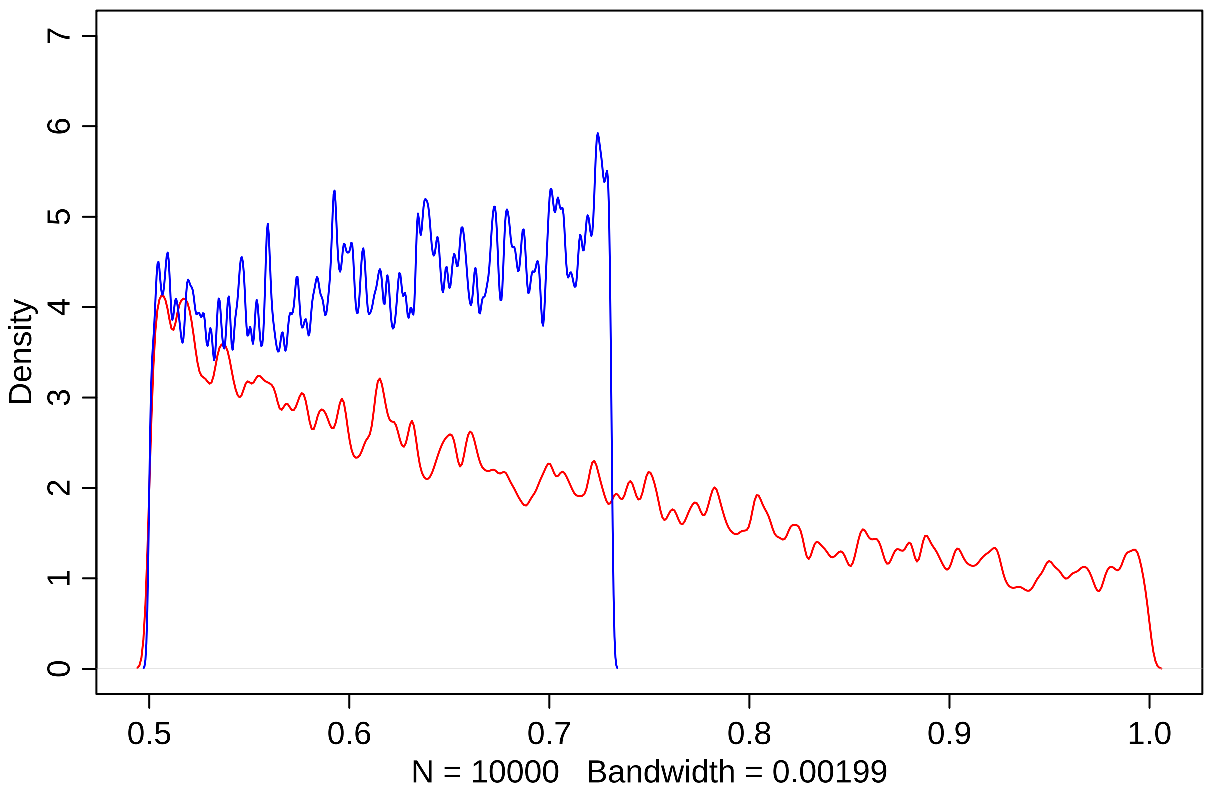
```
# comparing PRIOR distributions for 'sigma' in the two models
prior_m2a_4c <- extract.prior(m2a_4c, n=1e4)
```

```
## Warning in '/var/folders/wx/1_76tj0s15gc4yxmndvw4l_00000gn/T/RtmpGmhclW/model-9fc029e10aac.stan', li
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc

## Running MCMC with 1 chain, with 1 thread(s) per chain...
##
## Chain 1 Iteration:    1 / 20000 [  0%]  (Warmup)
## Chain 1 Iteration:  100 / 20000 [  0%]  (Warmup)
## Chain 1 Iteration:  200 / 20000 [  1%]  (Warmup)
## Chain 1 Iteration:  300 / 20000 [  1%]  (Warmup)
## Chain 1 Iteration:  400 / 20000 [  2%]  (Warmup)
## Chain 1 Iteration:  500 / 20000 [  2%]  (Warmup)
## Chain 1 Iteration:  600 / 20000 [  3%]  (Warmup)
## Chain 1 Iteration:  700 / 20000 [  3%]  (Warmup)
## Chain 1 Iteration:  800 / 20000 [  4%]  (Warmup)
## Chain 1 Iteration:  900 / 20000 [  4%]  (Warmup)
## Chain 1 Iteration: 1000 / 20000 [  5%]  (Warmup)
## Chain 1 Iteration: 1100 / 20000 [  5%]  (Warmup)
## Chain 1 Iteration:  1200 / 20000 [  6%]  (Warmup)
## Chain 1 Iteration:  1300 / 20000 [  6%]  (Warmup)
## Chain 1 Iteration:  1400 / 20000 [  7%]  (Warmup)
## Chain 1 Iteration:  1500 / 20000 [  7%]  (Warmup)
## Chain 1 Iteration:  1600 / 20000 [  8%]  (Warmup)
## Chain 1 Iteration:  1700 / 20000 [  8%]  (Warmup)
## Chain 1 Iteration:  1800 / 20000 [  9%]  (Warmup)
## Chain 1 Iteration:  1900 / 20000 [  9%]  (Warmup)
## Chain 1 Iteration:  2000 / 20000 [ 10%]  (Warmup)
## Chain 1 Iteration:  2100 / 20000 [ 10%]  (Warmup)
## Chain 1 Iteration:  2200 / 20000 [ 11%]  (Warmup)
## Chain 1 Iteration:  2300 / 20000 [ 11%]  (Warmup)
## Chain 1 Iteration:  2400 / 20000 [ 12%]  (Warmup)
## Chain 1 Iteration:  2500 / 20000 [ 12%]  (Warmup)
## Chain 1 Iteration:  2600 / 20000 [ 13%]  (Warmup)
## Chain 1 Iteration:  2700 / 20000 [ 13%]  (Warmup)
## Chain 1 Iteration:  2800 / 20000 [ 14%]  (Warmup)
## Chain 1 Iteration:  2900 / 20000 [ 14%]  (Warmup)
## Chain 1 Iteration:  3000 / 20000 [ 15%]  (Warmup)
## Chain 1 Iteration:  3100 / 20000 [ 15%]  (Warmup)
## Chain 1 Iteration:  3200 / 20000 [ 16%]  (Warmup)
## Chain 1 Iteration:  3300 / 20000 [ 16%]  (Warmup)
## Chain 1 Iteration:  3400 / 20000 [ 17%]  (Warmup)
## Chain 1 Iteration:  3500 / 20000 [ 17%]  (Warmup)
## Chain 1 Iteration:  3600 / 20000 [ 18%]  (Warmup)
## Chain 1 Iteration:  3700 / 20000 [ 18%]  (Warmup)
## Chain 1 Iteration:  3800 / 20000 [ 19%]  (Warmup)
## Chain 1 Iteration:  3900 / 20000 [ 19%]  (Warmup)
## Chain 1 Iteration:  4000 / 20000 [ 20%]  (Warmup)
## Chain 1 Iteration:  4100 / 20000 [ 20%]  (Warmup)
## Chain 1 Iteration:  4200 / 20000 [ 21%]  (Warmup)
## Chain 1 Iteration:  4300 / 20000 [ 21%]  (Warmup)
## Chain 1 Iteration:  4400 / 20000 [ 22%]  (Warmup)
## Chain 1 Iteration:  4500 / 20000 [ 22%]  (Warmup)
## Chain 1 Iteration:  4600 / 20000 [ 23%]  (Warmup)
## Chain 1 Iteration:  4700 / 20000 [ 23%]  (Warmup)
## Chain 1 Iteration:  4800 / 20000 [ 24%]  (Warmup)
## Chain 1 Iteration:  4900 / 20000 [ 24%]  (Warmup)
## Chain 1 Iteration:  5000 / 20000 [ 25%]  (Warmup)
## Chain 1 Iteration:  5100 / 20000 [ 25%]  (Warmup)
## Chain 1 Iteration:  5200 / 20000 [ 26%]  (Warmup)
## Chain 1 Iteration:  5300 / 20000 [ 26%]  (Warmup)
## Chain 1 Iteration:  5400 / 20000 [ 27%]  (Warmup)
## Chain 1 Iteration:  5500 / 20000 [ 27%]  (Warmup)
## Chain 1 Iteration:  5600 / 20000 [ 28%]  (Warmup)
## Chain 1 Iteration:  5700 / 20000 [ 28%]  (Warmup)
## Chain 1 Iteration:  5800 / 20000 [ 29%]  (Warmup)
## Chain 1 Iteration:  5900 / 20000 [ 29%]  (Warmup)
## Chain 1 Iteration:  6000 / 20000 [ 30%]  (Warmup)
## Chain 1 Iteration:  6100 / 20000 [ 30%]  (Warmup)
## Chain 1 Iteration:  6200 / 20000 [ 31%]  (Warmup)
## Chain 1 Iteration:  6300 / 20000 [ 31%]  (Warmup)
## Chain 1 Iteration:  6400 / 20000 [ 32%]  (Warmup)
## Chain 1 Iteration:  6500 / 20000 [ 32%]  (Warmup)
## Chain 1 Iteration:  6600 / 20000 [ 33%]  (Warmup)
## Chain 1 Iteration:  6700 / 20000 [ 33%]  (Warmup)
## Chain 1 Iteration:  6800 / 20000 [ 34%]  (Warmup)
## Chain 1 Iteration:  6900 / 20000 [ 34%]  (Warmup)
## Chain 1 Iteration:  7000 / 20000 [ 35%]  (Warmup)
## Chain 1 Iteration:  7100 / 20000 [ 35%]  (Warmup)
## Chain 1 Iteration:  7200 / 20000 [ 36%]  (Warmup)
## Chain 1 Iteration:  7300 / 20000 [ 36%]  (Warmup)
## Chain 1 Iteration:  7400 / 20000 [ 37%]  (Warmup)
## Chain 1 Iteration:  7500 / 20000 [ 37%]  (Warmup)
## Chain 1 Iteration:  7600 / 20000 [ 38%]  (Warmup)
## Chain 1 Iteration:  7700 / 20000 [ 38%]  (Warmup)
## Chain 1 Iteration:  7800 / 20000 [ 39%]  (Warmup)
## Chain 1 Iteration:  7900 / 20000 [ 39%]  (Warmup)
## Chain 1 Iteration:  8000 / 20000 [ 40%]  (Warmup)
## Chain 1 Iteration:  8100 / 20000 [ 40%]  (Warmup)
## Chain 1 Iteration:  8200 / 20000 [ 41%]  (Warmup)
## Chain 1 Iteration:  8300 / 20000 [ 41%]  (Warmup)
## Chain 1 Iteration:  8400 / 20000 [ 42%]  (Warmup)
## Chain 1 Iteration:  8500 / 20000 [ 42%]  (Warmup)
## Chain 1 Iteration:  8600 / 20000 [ 43%]  (Warmup)
## Chain 1 Iteration:  8700 / 20000 [ 43%]  (Warmup)
## Chain 1 Iteration:  8800 / 20000 [ 44%]  (Warmup)
## Chain 1 Iteration:  8900 / 20000 [ 44%]  (Warmup)
## Chain 1 Iteration:  9000 / 20000 [ 45%]  (Warmup)
## Chain 1 Iteration:  9100 / 20000 [ 45%]  (Warmup)
## Chain 1 Iteration:  9200 / 20000 [ 46%]  (Warmup)
## Chain 1 Iteration:  9300 / 20000 [ 46%]  (Warmup)
## Chain 1 Iteration:  9400 / 20000 [ 47%]  (Warmup)
## Chain 1 Iteration:  9500 / 20000 [ 47%]  (Warmup)
## Chain 1 Iteration:  9600 / 20000 [ 48%]  (Warmup)
## Chain 1 Iteration:  9700 / 20000 [ 48%]  (Warmup)
## Chain 1 Iteration:  9800 / 20000 [ 49%]  (Warmup)
## Chain 1 Iteration:  9900 / 20000 [ 49%]  (Warmup)
## Chain 1 Iteration: 10000 / 20000 [ 50%]  (Warmup)
## Chain 1 Iteration: 10001 / 20000 [ 50%]  (Sampling)
## Chain 1 Iteration: 10100 / 20000 [ 50%]  (Sampling)
## Chain 1 Iteration: 10200 / 20000 [ 51%]  (Sampling)
## Chain 1 Iteration: 10300 / 20000 [ 51%]  (Sampling)
## Chain 1 Iteration: 10400 / 20000 [ 52%]  (Sampling)
## Chain 1 Iteration: 10500 / 20000 [ 52%]  (Sampling)
## Chain 1 Iteration: 10600 / 20000 [ 53%]  (Sampling)
## Chain 1 Iteration: 10700 / 20000 [ 53%]  (Sampling)
## Chain 1 Iteration: 10800 / 20000 [ 54%]  (Sampling)
## Chain 1 Iteration: 10900 / 20000 [ 54%]  (Sampling)
## Chain 1 Iteration: 11000 / 20000 [ 55%]  (Sampling)
## Chain 1 Iteration: 11100 / 20000 [ 55%]  (Sampling)
## Chain 1 Iteration: 11200 / 20000 [ 56%]  (Sampling)
## Chain 1 Iteration: 11300 / 20000 [ 56%]  (Sampling)
## Chain 1 Iteration: 11400 / 20000 [ 57%]  (Sampling)
## Chain 1 Iteration: 11500 / 20000 [ 57%]  (Sampling)
## Chain 1 Iteration: 11600 / 20000 [ 58%]  (Sampling)
## Chain 1 Iteration: 11700 / 20000 [ 58%]  (Sampling)
## Chain 1 Iteration: 11800 / 20000 [ 59%]  (Sampling)
## Chain 1 Iteration: 11900 / 20000 [ 59%]  (Sampling)
## Chain 1 Iteration: 12000 / 20000 [ 60%]  (Sampling)
## Chain 1 Iteration: 12100 / 20000 [ 60%]  (Sampling)
## Chain 1 Iteration: 12200 / 20000 [ 61%]  (Sampling)
## Chain 1 Iteration: 12300 / 20000 [ 61%]  (Sampling)
## Chain 1 Iteration: 12400 / 20000 [ 62%]  (Sampling)
## Chain 1 Iteration: 12500 / 20000 [ 62%]  (Sampling)
## Chain 1 Iteration: 12600 / 20000 [ 63%]  (Sampling)
## Chain 1 Iteration: 12700 / 20000 [ 63%]  (Sampling)
## Chain 1 Iteration: 12800 / 20000 [ 64%]  (Sampling)
## Chain 1 Iteration: 12900 / 20000 [ 64%]  (Sampling)
## Chain 1 Iteration: 13000 / 20000 [ 65%]  (Sampling)
## Chain 1 Iteration: 13100 / 20000 [ 65%]  (Sampling)
## Chain 1 Iteration: 13200 / 20000 [ 66%]  (Sampling)
## Chain 1 Iteration: 13300 / 20000 [ 66%]  (Sampling)
## Chain 1 Iteration: 13400 / 20000 [ 67%]  (Sampling)
## Chain 1 Iteration: 13500 / 20000 [ 67%]  (Sampling)
## Chain 1 Iteration: 13600 / 20000 [ 68%]  (Sampling)
## Chain 1 Iteration: 13700 / 20000 [ 68%]  (Sampling)
## Chain 1 Iteration: 13800 / 20000 [ 69%]  (Sampling)
## Chain 1 Iteration: 13900 / 20000 [ 69%]  (Sampling)
## Chain 1 Iteration: 14000 / 20000 [ 70%]  (Sampling)
## Chain 1 Iteration: 14100 / 20000 [ 70%]  (Sampling)
## Chain 1 Iteration: 14200 / 20000 [ 71%]  (Sampling)
## Chain 1 Iteration: 14300 / 20000 [ 71%]  (Sampling)
## Chain 1 Iteration: 14400 / 20000 [ 72%]  (Sampling)
## Chain 1 Iteration: 14500 / 20000 [ 72%]  (Sampling)
## Chain 1 Iteration: 14600 / 20000 [ 73%]  (Sampling)
## Chain 1 Iteration: 14700 / 20000 [ 73%]  (Sampling)
## Chain 1 Iteration: 14800 / 20000 [ 74%]  (Sampling)
## Chain 1 Iteration: 14900 / 20000 [ 74%]  (Sampling)
## Chain 1 Iteration: 15000 / 20000 [ 75%]  (Sampling)
## Chain 1 Iteration: 15100 / 20000 [ 75%]  (Sampling)
## Chain 1 Iteration: 15200 / 20000 [ 76%]  (Sampling)
## Chain 1 Iteration: 15300 / 20000 [ 76%]  (Sampling)
## Chain 1 Iteration: 15400 / 20000 [ 77%]  (Sampling)
## Chain 1 Iteration: 15500 / 20000 [ 77%]  (Sampling)
## Chain 1 Iteration: 15600 / 20000 [ 78%]  (Sampling)
## Chain 1 Iteration: 15700 / 20000 [ 78%]  (Sampling)
## Chain 1 Iteration: 15800 / 20000 [ 79%]  (Sampling)
## Chain 1 Iteration: 15900 / 20000 [ 79%]  (Sampling)
## Chain 1 Iteration: 16000 / 20000 [ 80%]  (Sampling)
## Chain 1 Iteration: 16100 / 20000 [ 80%]  (Sampling)
## Chain 1 Iteration: 16200 / 20000 [ 81%]  (Sampling)
## Chain 1 Iteration: 16300 / 20000 [ 81%]  (Sampling)
## Chain 1 Iteration: 16400 / 20000 [ 82%]  (Sampling)
## Chain 1 Iteration: 16500 / 20000 [ 82%]  (Sampling)
## Chain 1 Iteration: 16600 / 20000 [ 83%]  (Sampling)
## Chain 1 Iteration: 16700 / 20000 [ 83%]  (Sampling)
## Chain 1 Iteration: 16800 / 20000 [ 84%]  (Sampling)
## Chain 1 Iteration: 16900 / 20000 [ 84%]  (Sampling)
## Chain 1 Iteration: 17000 / 20000 [ 85%]  (Sampling)
## Chain 1 Iteration: 17100 / 20000 [ 85%]  (Sampling)
## Chain 1 Iteration: 17200 / 20000 [ 86%]  (Sampling)
## Chain 1 Iteration: 17300 / 20000 [ 86%]  (Sampling)
## Chain 1 Iteration: 17400 / 20000 [ 87%]  (Sampling)
## Chain 1 Iteration: 17500 / 20000 [ 87%]  (Sampling)
## Chain 1 Iteration: 17600 / 20000 [ 88%]  (Sampling)
## Chain 1 Iteration: 17700 / 20000 [ 88%]  (Sampling)
## Chain 1 Iteration: 17800 / 20000 [ 89%]  (Sampling)
## Chain 1 Iteration: 17900 / 20000 [ 89%]  (Sampling)
## Chain 1 Iteration: 18000 / 20000 [ 90%]  (Sampling)
## Chain 1 Iteration: 18100 / 20000 [ 90%]  (Sampling)
## Chain 1 Iteration: 18200 / 20000 [ 91%]  (Sampling)
## Chain 1 Iteration: 18300 / 20000 [ 91%]  (Sampling)
## Chain 1 Iteration: 18400 / 20000 [ 92%]  (Sampling)
## Chain 1 Iteration: 18500 / 20000 [ 92%]  (Sampling)
## Chain 1 Iteration: 18600 / 20000 [ 93%]  (Sampling)
## Chain 1 Iteration: 18700 / 20000 [ 93%]  (Sampling)
## Chain 1 Iteration: 18800 / 20000 [ 94%]  (Sampling)
## Chain 1 Iteration: 18900 / 20000 [ 94%]  (Sampling)
## Chain 1 Iteration: 19000 / 20000 [ 95%]  (Sampling)
## Chain 1 Iteration: 19100 / 20000 [ 95%]  (Sampling)
## Chain 1 Iteration: 19200 / 20000 [ 96%]  (Sampling)
## Chain 1 Iteration: 19300 / 20000 [ 96%]  (Sampling)
## Chain 1 Iteration: 19400 / 20000 [ 97%]  (Sampling)
## Chain 1 Iteration: 19500 / 20000 [ 97%]  (Sampling)
## Chain 1 Iteration: 19600 / 20000 [ 98%]  (Sampling)
## Chain 1 Iteration: 19700 / 20000 [ 98%]  (Sampling)
## Chain 1 Iteration: 19800 / 20000 [ 99%]  (Sampling)
## Chain 1 Iteration: 19900 / 20000 [ 99%]  (Sampling)
## Chain 1 Iteration: 20000 / 20000 [100%]  (Sampling)
## Chain 1 finished in 2.1 seconds.
```

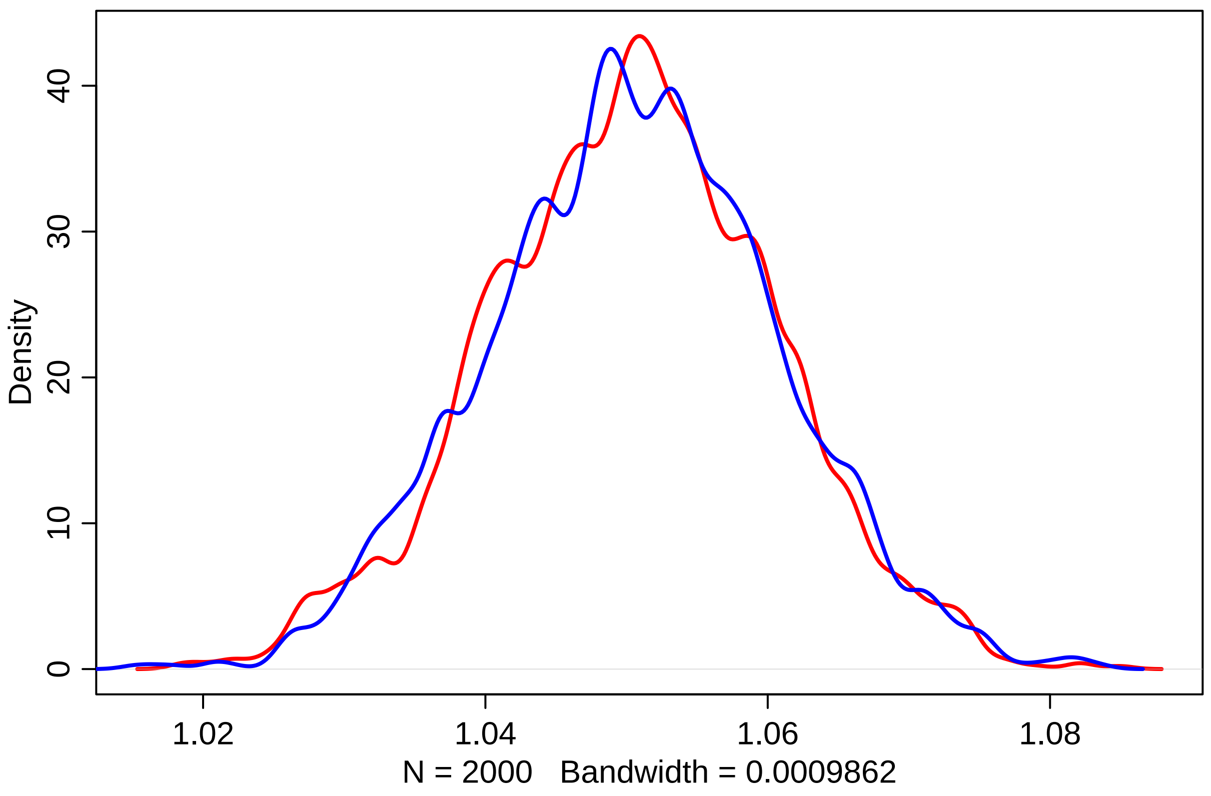
```
prior_m2c <- extract.prior(m2c, n=1e4)
```

```
## Warning in '/var/folders/wx/1_76tj0s15gc4yxmndvw4l_00000gn/T/RtmpGmhclW/model-9fc03f498a5c.stan', li
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc

## Running MCMC with 1 chain, with 1 thread(s) per chain...
##
## Chain 1 Iteration:     1 / 20000 [  0%]  (Warmup)
## Chain 1 Iteration:   100 / 20000 [  0%]  (Warmup)
## Chain 1 Iteration:   200 / 20000 [  1%]  (Warmup)
## Chain 1 Iteration:   300 / 20000 [  1%]  (Warmup)
## Chain 1 Iteration:   400 / 20000 [  2%]  (Warmup)
## Chain 1 Iteration:   500 / 20000 [  2%]  (Warmup)
## Chain 1 Iteration:   600 / 20000 [  3%]  (Warmup)
## Chain 1 Iteration:   700 / 20000 [  3%]  (Warmup)
## Chain 1 Iteration:   800 / 20000 [  4%]  (Warmup)
## Chain 1 Iteration:   900 / 20000 [  4%]  (Warmup)
## Chain 1 Iteration:  1000 / 20000 [  5%]  (Warmup)
## Chain 1 Iteration:  1100 / 20000 [  5%]  (Warmup)
## Chain 1 Iteration:  1200 / 20000 [  6%]  (Warmup)
## Chain 1 Iteration:  1300 / 20000 [  6%]  (Warmup)
## Chain 1 Iteration:  1400 / 20000 [  7%]  (Warmup)
## Chain 1 Iteration:  1500 / 20000 [  7%]  (Warmup)
## Chain 1 Iteration:  1600 / 20000 [  8%]  (Warmup)
## Chain 1 Iteration:  1700 / 20000 [  8%]  (Warmup)
## Chain 1 Iteration:  1800 / 20000 [  9%]  (Warmup)
## Chain 1 Iteration:  1900 / 20000 [  9%]  (Warmup)
## Chain 1 Iteration:  2000 / 20000 [ 10%]  (Warmup)
## Chain 1 Iteration:  2100 / 20000 [ 10%]  (Warmup)
## Chain 1 Iteration:  2200 / 20000 [ 11%]  (Warmup)
## Chain 1 Iteration:  2300 / 20000 [ 11%]  (Warmup)
## Chain 1 Iteration:  2400 / 20000 [ 12%]  (Warmup)
## Chain 1 Iteration:  2500 / 20000 [ 12%]  (Warmup)
## Chain 1 Iteration:  2600 / 20000 [ 13%]  (Warmup)
## Chain 1 Iteration:  2700 / 20000 [ 13%]  (Warmup)
## Chain 1 Iteration:  2800 / 20000 [ 14%]  (Warmup)
## Chain 1 Iteration:  2900 / 20000 [ 14%]  (Warmup)
## Chain 1 Iteration:  3000 / 20000 [ 15%]  (Warmup)
## Chain 1 Iteration:  3100 / 20000 [ 15%]  (Warmup)
## Chain 1 Iteration:  3200 / 20000 [ 16%]  (Warmup)
## Chain 1 Iteration:  3300 / 20000 [ 16%]  (Warmup)
## Chain 1 Iteration:  3400 / 20000 [ 17%]  (Warmup)
## Chain 1 Iteration:  3500 / 20000 [ 17%]  (Warmup)
## Chain 1 Iteration:  3600 / 20000 [ 18%]  (Warmup)
## Chain 1 Iteration:  3700 / 20000 [ 18%]  (Warmup)
## Chain 1 Iteration:  3800 / 20000 [ 19%]  (Warmup)
## Chain 1 Iteration:  3900 / 20000 [ 19%]  (Warmup)
## Chain 1 Iteration:  4000 / 20000 [ 20%]  (Warmup)
## Chain 1 Iteration:  4100 / 20000 [ 20%]  (Warmup)
## Chain 1 Iteration:  4200 / 20000 [ 21%]  (Warmup)
## Chain 1 Iteration:  4300 / 20000 [ 21%]  (Warmup)
## Chain 1 Iteration:  4400 / 20000 [ 22%]  (Warmup)
## Chain 1 Iteration:  4500 / 20000 [ 22%]  (Warmup)
## Chain 1 Iteration:  4600 / 20000 [ 23%]  (Warmup)
## Chain 1 Iteration:  4700 / 20000 [ 23%]  (Warmup)
## Chain 1 Iteration:  4800 / 20000 [ 24%]  (Warmup)
## Chain 1 Iteration:  4900 / 20000 [ 24%]  (Warmup)
## Chain 1 Iteration:  5000 / 20000 [ 25%]  (Warmup)
## Chain 1 Iteration:  5100 / 20000 [ 25%]  (Warmup)
## Chain 1 Iteration:  5200 / 20000 [ 26%]  (Warmup)
## Chain 1 Iteration:  5300 / 20000 [ 26%]  (Warmup)
## Chain 1 Iteration:  5400 / 20000 [ 27%]  (Warmup)
## Chain 1 Iteration:  5500 / 20000 [ 27%]  (Warmup)
## Chain 1 Iteration:  5600 / 20000 [ 28%]  (Warmup)
## Chain 1 Iteration:  5700 / 20000 [ 28%]  (Warmup)
## Chain 1 Iteration:  5800 / 20000 [ 29%]  (Warmup)
## Chain 1 Iteration:  5900 / 20000 [ 29%]  (Warmup)
## Chain 1 Iteration:  6000 / 20000 [ 30%]  (Warmup)
## Chain 1 Iteration:  6100 / 20000 [ 30%]  (Warmup)
## Chain 1 Iteration:  6200 / 20000 [ 31%]  (Warmup)
## Chain 1 Iteration:  6300 / 20000 [ 31%]  (Warmup)
## Chain 1 Iteration:  6400 / 20000 [ 32%]  (Warmup)
## Chain 1 Iteration:  6500 / 20000 [ 32%]  (Warmup)
## Chain 1 Iteration:  6600 / 20000 [ 33%]  (Warmup)
## Chain 1 Iteration:  6700 / 20000 [ 33%]  (Warmup)
## Chain 1 Iteration:  6800 / 20000 [ 34%]  (Warmup)
## Chain 1 Iteration:  6900 / 20000 [ 34%]  (Warmup)
## Chain 1 Iteration:  7000 / 20000 [ 35%]  (Warmup)
## Chain 1 Iteration:  7100 / 20000 [ 35%]  (Warmup)
## Chain 1 Iteration:  7200 / 20000 [ 36%]  (Warmup)
## Chain 1 Iteration:  7300 / 20000 [ 36%]  (Warmup)
## Chain 1 Iteration:  7400 / 20000 [ 37%]  (Warmup)
## Chain 1 Iteration:  7500 / 20000 [ 37%]  (Warmup)
## Chain 1 Iteration:  7600 / 20000 [ 38%]  (Warmup)
## Chain 1 Iteration:  7700 / 20000 [ 38%]  (Warmup)
## Chain 1 Iteration:  7800 / 20000 [ 39%]  (Warmup)
## Chain 1 Iteration:  7900 / 20000 [ 39%]  (Warmup)
## Chain 1 Iteration:  8000 / 20000 [ 40%]  (Warmup)
## Chain 1 Iteration:  8100 / 20000 [ 40%]  (Warmup)
## Chain 1 Iteration:  8200 / 20000 [ 41%]  (Warmup)
## Chain 1 Iteration:  8300 / 20000 [ 41%]  (Warmup)
## Chain 1 Iteration:  8400 / 20000 [ 42%]  (Warmup)
## Chain 1 Iteration:  8500 / 20000 [ 42%]  (Warmup)
## Chain 1 Iteration:  8600 / 20000 [ 43%]  (Warmup)
## Chain 1 Iteration:  8700 / 20000 [ 43%]  (Warmup)
## Chain 1 Iteration:  8800 / 20000 [ 44%]  (Warmup)
## Chain 1 Iteration:  8900 / 20000 [ 44%]  (Warmup)
## Chain 1 Iteration:  9000 / 20000 [ 45%]  (Warmup)
## Chain 1 Iteration:  9100 / 20000 [ 45%]  (Warmup)
## Chain 1 Iteration:  9200 / 20000 [ 46%]  (Warmup)
## Chain 1 Iteration:  9300 / 20000 [ 46%]  (Warmup)
## Chain 1 Iteration:  9400 / 20000 [ 47%]  (Warmup)
## Chain 1 Iteration:  9500 / 20000 [ 47%]  (Warmup)
## Chain 1 Iteration:  9600 / 20000 [ 48%]  (Warmup)
## Chain 1 Iteration:  9700 / 20000 [ 48%]  (Warmup)
## Chain 1 Iteration:  9800 / 20000 [ 49%]  (Warmup)
## Chain 1 Iteration:  9900 / 20000 [ 49%]  (Warmup)
## Chain 1 Iteration: 10000 / 20000 [ 50%]  (Warmup)
## Chain 1 Iteration: 10001 / 20000 [ 50%]  (Sampling)
## Chain 1 Iteration: 10100 / 20000 [ 50%]  (Sampling)
## Chain 1 Iteration: 10200 / 20000 [ 51%]  (Sampling)
## Chain 1 Iteration: 10300 / 20000 [ 51%]  (Sampling)
## Chain 1 Iteration: 10400 / 20000 [ 52%]  (Sampling)
## Chain 1 Iteration: 10500 / 20000 [ 52%]  (Sampling)
## Chain 1 Iteration: 10600 / 20000 [ 53%]  (Sampling)
## Chain 1 Iteration: 10700 / 20000 [ 53%]  (Sampling)
## Chain 1 Iteration: 10800 / 20000 [ 54%]  (Sampling)
## Chain 1 Iteration: 10900 / 20000 [ 54%]  (Sampling)
## Chain 1 Iteration: 11000 / 20000 [ 55%]  (Sampling)
## Chain 1 Iteration: 11100 / 20000 [ 55%]  (Sampling)
## Chain 1 Iteration: 11200 / 20000 [ 56%]  (Sampling)
## Chain 1 Iteration: 11300 / 20000 [ 56%]  (Sampling)
## Chain 1 Iteration: 11400 / 20000 [ 57%]  (Sampling)
## Chain 1 Iteration: 11500 / 20000 [ 57%]  (Sampling)
## Chain 1 Iteration: 11600 / 20000 [ 58%]  (Sampling)
## Chain 1 Iteration: 11700 / 20000 [ 58%]  (Sampling)
## Chain 1 Iteration: 11800 / 20000 [ 59%]  (Sampling)
## Chain 1 Iteration: 11900 / 20000 [ 59%]  (Sampling)
## Chain 1 Iteration: 12000 / 20000 [ 60%]  (Sampling)
## Chain 1 Iteration: 12100 / 20000 [ 60%]  (Sampling)
## Chain 1 Iteration: 12200 / 20000 [ 61%]  (Sampling)
## Chain 1 Iteration: 12300 / 20000 [ 61%]  (Sampling)
## Chain 1 Iteration: 12400 / 20000 [ 62%]  (Sampling)
## Chain 1 Iteration: 12500 / 20000 [ 62%]  (Sampling)
## Chain 1 Iteration: 12600 / 20000 [ 63%]  (Sampling)
## Chain 1 Iteration: 12700 / 20000 [ 63%]  (Sampling)
## Chain 1 Iteration: 12800 / 20000 [ 64%]  (Sampling)
## Chain 1 Iteration: 12900 / 20000 [ 64%]  (Sampling)
## Chain 1 Iteration: 13000 / 20000 [ 65%]  (Sampling)
## Chain 1 Iteration: 13100 / 20000 [ 65%]  (Sampling)
## Chain 1 Iteration: 13200 / 20000 [ 66%]  (Sampling)
## Chain 1 Iteration: 13300 / 20000 [ 66%]  (Sampling)
## Chain 1 Iteration: 13400 / 20000 [ 67%]  (Sampling)
## Chain 1 Iteration: 13500 / 20000 [ 67%]  (Sampling)
## Chain 1 Iteration: 13600 / 20000 [ 68%]  (Sampling)
## Chain 1 Iteration: 13700 / 20000 [ 68%]  (Sampling)
## Chain 1 Iteration: 13800 / 20000 [ 69%]  (Sampling)
## Chain 1 Iteration: 13900 / 20000 [ 69%]  (Sampling)
## Chain 1 Iteration: 14000 / 20000 [ 70%]  (Sampling)
## Chain 1 Iteration: 14100 / 20000 [ 70%]  (Sampling)
## Chain 1 Iteration: 14200 / 20000 [ 71%]  (Sampling)
## Chain 1 Iteration: 14300 / 20000 [ 71%]  (Sampling)
## Chain 1 Iteration: 14400 / 20000 [ 72%]  (Sampling)
## Chain 1 Iteration: 14500 / 20000 [ 72%]  (Sampling)
## Chain 1 Iteration: 14600 / 20000 [ 73%]  (Sampling)
## Chain 1 Iteration: 14700 / 20000 [ 73%]  (Sampling)
## Chain 1 Iteration: 14800 / 20000 [ 74%]  (Sampling)
## Chain 1 Iteration: 14900 / 20000 [ 74%]  (Sampling)
## Chain 1 Iteration: 15000 / 20000 [ 75%]  (Sampling)
## Chain 1 Iteration: 15100 / 20000 [ 75%]  (Sampling)
## Chain 1 Iteration: 15200 / 20000 [ 76%]  (Sampling)
## Chain 1 Iteration: 15300 / 20000 [ 76%]  (Sampling)
## Chain 1 Iteration: 15400 / 20000 [ 77%]  (Sampling)
## Chain 1 Iteration: 15500 / 20000 [ 77%]  (Sampling)
## Chain 1 Iteration: 15600 / 20000 [ 78%]  (Sampling)
## Chain 1 Iteration: 15700 / 20000 [ 78%]  (Sampling)
## Chain 1 Iteration: 15800 / 20000 [ 79%]  (Sampling)
## Chain 1 Iteration: 15900 / 20000 [ 79%]  (Sampling)
## Chain 1 Iteration: 16000 / 20000 [ 80%]  (Sampling)
## Chain 1 Iteration: 16100 / 20000 [ 80%]  (Sampling)
## Chain 1 Iteration: 16200 / 20000 [ 81%]  (Sampling)
## Chain 1 Iteration: 16300 / 20000 [ 81%]  (Sampling)
## Chain 1 Iteration: 16400 / 20000 [ 82%]  (Sampling)
## Chain 1 Iteration: 16500 / 20000 [ 82%]  (Sampling)
## Chain 1 Iteration: 16600 / 20000 [ 83%]  (Sampling)
## Chain 1 Iteration: 16700 / 20000 [ 83%]  (Sampling)
## Chain 1 Iteration: 16800 / 20000 [ 84%]  (Sampling)
## Chain 1 Iteration: 16900 / 20000 [ 84%]  (Sampling)
## Chain 1 Iteration: 17000 / 20000 [ 85%]  (Sampling)
## Chain 1 Iteration: 17100 / 20000 [ 85%]  (Sampling)
## Chain 1 Iteration: 17200 / 20000 [ 86%]  (Sampling)
## Chain 1 Iteration: 17300 / 20000 [ 86%]  (Sampling)
## Chain 1 Iteration: 17400 / 20000 [ 87%]  (Sampling)
## Chain 1 Iteration: 17500 / 20000 [ 87%]  (Sampling)
## Chain 1 Iteration: 17600 / 20000 [ 88%]  (Sampling)
## Chain 1 Iteration: 17700 / 20000 [ 88%]  (Sampling)
## Chain 1 Iteration: 17800 / 20000 [ 89%]  (Sampling)
## Chain 1 Iteration: 17900 / 20000 [ 89%]  (Sampling)
## Chain 1 Iteration: 18000 / 20000 [ 90%]  (Sampling)
## Chain 1 Iteration: 18100 / 20000 [ 90%]  (Sampling)
## Chain 1 Iteration: 18200 / 20000 [ 91%]  (Sampling)
## Chain 1 Iteration: 18300 / 20000 [ 91%]  (Sampling)
## Chain 1 Iteration: 18400 / 20000 [ 92%]  (Sampling)
## Chain 1 Iteration: 18500 / 20000 [ 92%]  (Sampling)
## Chain 1 Iteration: 18600 / 20000 [ 93%]  (Sampling)
## Chain 1 Iteration: 18700 / 20000 [ 93%]  (Sampling)
## Chain 1 Iteration: 18800 / 20000 [ 94%]  (Sampling)
## Chain 1 Iteration: 18900 / 20000 [ 94%]  (Sampling)
## Chain 1 Iteration: 19000 / 20000 [ 95%]  (Sampling)
## Chain 1 Iteration: 19100 / 20000 [ 95%]  (Sampling)
## Chain 1 Iteration: 19200 / 20000 [ 96%]  (Sampling)
## Chain 1 Iteration: 19300 / 20000 [ 96%]  (Sampling)
## Chain 1 Iteration: 19400 / 20000 [ 97%]  (Sampling)
## Chain 1 Iteration: 19500 / 20000 [ 97%]  (Sampling)
## Chain 1 Iteration: 19600 / 20000 [ 98%]  (Sampling)
## Chain 1 Iteration: 19700 / 20000 [ 98%]  (Sampling)
## Chain 1 Iteration: 19800 / 20000 [ 99%]  (Sampling)
## Chain 1 Iteration: 19900 / 20000 [ 99%]  (Sampling)
## Chain 1 Iteration: 20000 / 20000 [100%]  (Sampling)
## Chain 1 finished in 1.8 seconds.
```

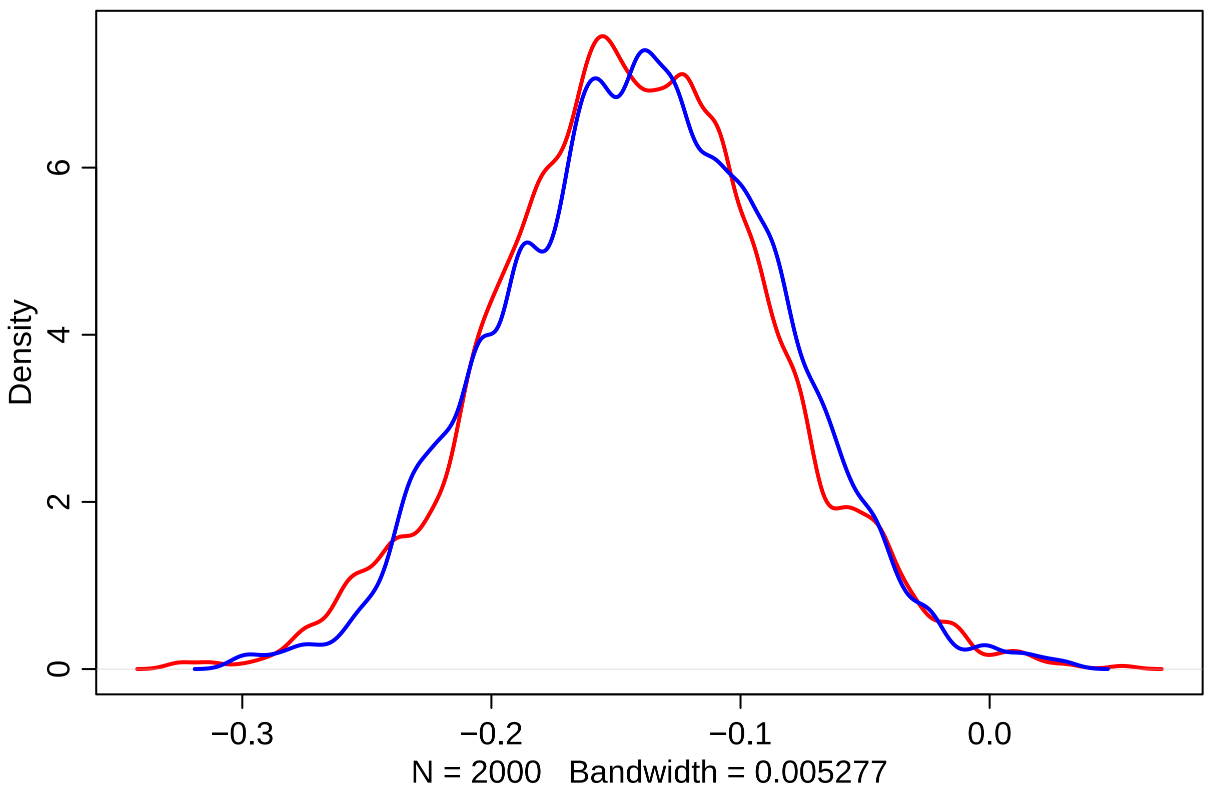
```r
pr_m2a_4c <- inv_logit(prior_m2a_4c$sigma)
pr_m2c <- inv_logit(prior_m2c$sigma)

dens(pr_m2a_4c, adj = 0.1, col='red', ylim = c(0, 7))
dens(pr_m2c, adj = 0.1, add = TRUE, col='blue')
```



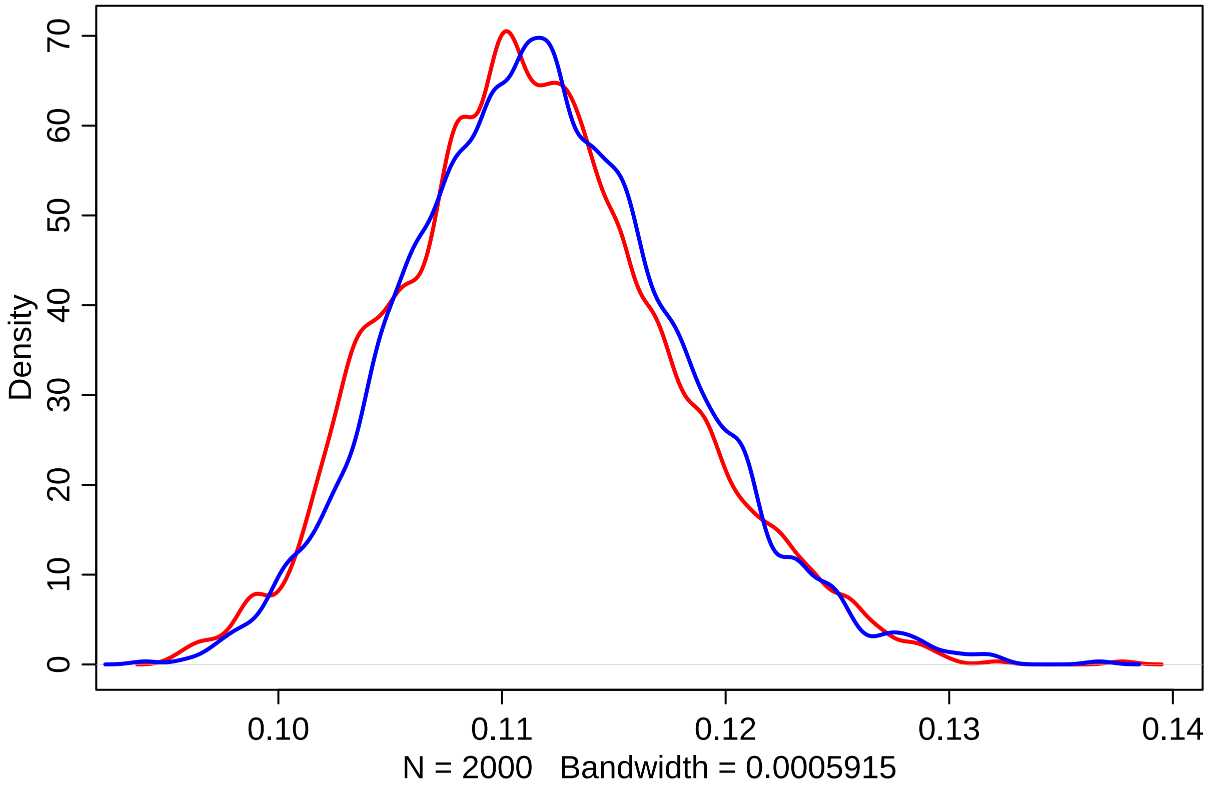
```
# comparing posterior distributions for 'a' in the two models
post_m2a_4c <- extract.samples(m2a_4c)
post_m2c <- extract.samples(m2c)
dens(post_m2a_4c$a[, 2], lwd = 2, col='red')
dens(post_m2c$a[, 2], add=TRUE, lwd = 2, col='blue')
```



```
# comparing posterior distributions for 'b' in the two models
post_m2a_4c <- extract.samples(m2a_4c)
post_m2c <- extract.samples(m2c)
dens(post_m2a_4c$b[, 2], lwd = 2, col='red')
dens(post_m2c$b[, 2], add=TRUE, lwd = 2, col='blue')
```

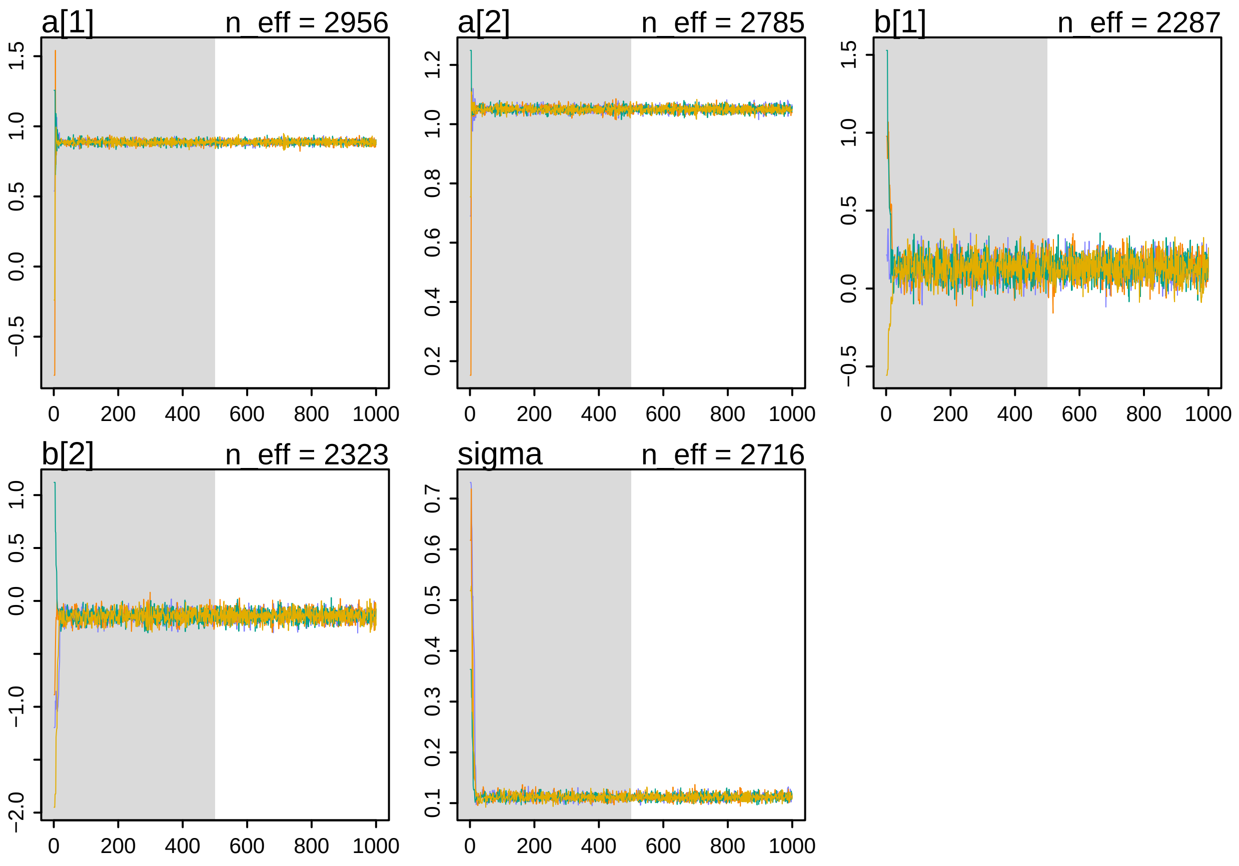


```
# comparing posterior distributions for 'sigma' in the two models
post_m2a_4c <- extract.samples(m2a_4c)
post_m2c <- extract.samples(m2c)
dens(post_m2a_4c$sigma, lwd = 2, col='red')
dens(post_m2c$sigma, add=TRUE, lwd = 2, col='blue')
```

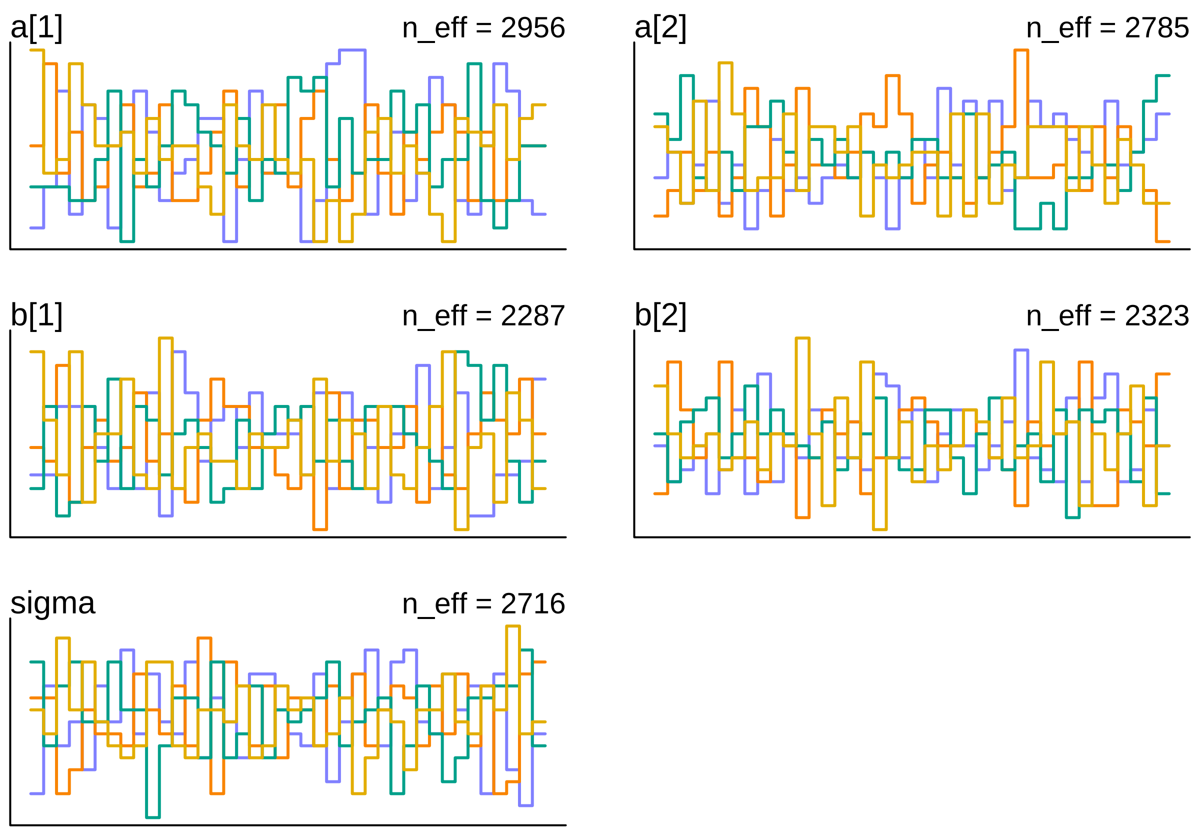


```
# Although very similar in precis tables, posterior distributions upon visualization appear to be
# skewed in direction of respective priors. For model m2a_4c, the prior of sigma is inclined
# little towards the left, thus the posterior is also very little oriented towards the left.
# For model m2c, since the prior is relatively flat, the posterior appears to be more uniformly
# centered.
```

```
# double checking the hygiene of the chains for model m2c using 'traceplot', just to be sure
traceplot( m2c )
```



```
# double checking the hygiene of the chains for model m2c using 'trankplot', just to be sure
trankplot( m2c , n_cols=2 )
```



d) Now fit your model with the log normal prior b[cid] ~ dlnorm(0,1) for b. What effect does this prior have on your posterior distribution? Explain your answer.

```r
m2d <- ulam(
alist(
log_gdp_std ~ dnorm( mu , sigma ) ,
mu <- a[cid] + b[cid]*( rugged_std - 0.215 ) ,
a[cid] ~ dnorm( 1 , 0.1 ) ,
b[cid] ~ dlnorm( 0 , 1 ) ,
sigma ~ dunif(0,1)
) , data=dat_slim , chains=4 , cores=4 )
```

```
## Warning in '/var/folders/wx/1_76tj0s15gc4yxmndvw4l_00000gn/T/RtmpGmhclW/model-9fc0563ebca5.stan', li
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc

## Running MCMC with 4 parallel chains, with 1 thread(s) per chain...
##
## Chain 1 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2 Iteration:   1 / 1000 [  0%]  (Warmup)

## Chain 2 Informational Message: The current Metropolis proposal is about to be rejected because of th

## Chain 2 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/var/folders/wx/1_7

## Chain 2 If this warning occurs sporadically, such as for highly constrained variable types like cova

## Chain 2 but if this warning occurs often then your model may be either severely ill-conditioned or m
## Chain 2

## Chain 3 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 4 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 4 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 4 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 1 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 2 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 2 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 2 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 3 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 3 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 3 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 4 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 4 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 4 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 4 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 4 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 2 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 3 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 3 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 4 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 4 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 4 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1 finished in 0.4 seconds.
## Chain 2 finished in 0.4 seconds.
## Chain 3 finished in 0.4 seconds.
## Chain 4 finished in 0.4 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.4 seconds.
## Total execution time: 0.5 seconds.
```

```
precis( m2d , 2 )
```

```
##             mean          sd       5.5%      94.5%    n_eff     Rhat4
## a[1]  0.88922812 0.016506861 0.86331858 0.91656752 2308.683 0.9991166
## a[2]  1.04761141 0.010073821 1.03069890 1.06362055 2459.535 0.9989680
## b[1]  0.18183365 0.066035791 0.08197540 0.28956615 1977.008 1.0005974
## b[2]  0.05070742 0.023607195 0.01959108 0.09401737 2034.892 1.0006229
## sigma 0.11557125 0.006766851 0.10506573 0.12683205 1599.145 1.0011244
```

```
# comparing PRIOR distributions for 'b' in the three models
prior_m2a_4c <- extract.prior(m2a_4c, n=1e4)
```

```
## Warning in '/var/folders/wx/1_76tj0s15gc4yxmndvw4l_00000gn/T/RtmpGmhclW/model-9fc0672033e6.stan', li
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc

## Running MCMC with 1 chain, with 1 thread(s) per chain...
##
## Chain 1 Iteration:     1 / 20000 [  0%]  (Warmup)
## Chain 1 Iteration:   100 / 20000 [  0%]  (Warmup)
## Chain 1 Iteration:   200 / 20000 [  1%]  (Warmup)
## Chain 1 Iteration:   300 / 20000 [  1%]  (Warmup)
## Chain 1 Iteration:   400 / 20000 [  2%]  (Warmup)
## Chain 1 Iteration:   500 / 20000 [  2%]  (Warmup)
## Chain 1 Iteration:   600 / 20000 [  3%]  (Warmup)
## Chain 1 Iteration:   700 / 20000 [  3%]  (Warmup)
## Chain 1 Iteration:   800 / 20000 [  4%]  (Warmup)
## Chain 1 Iteration:   900 / 20000 [  4%]  (Warmup)
## Chain 1 Iteration:  1000 / 20000 [  5%]  (Warmup)
## Chain 1 Iteration:  1100 / 20000 [  5%]  (Warmup)
## Chain 1 Iteration:  1200 / 20000 [  6%]  (Warmup)
## Chain 1 Iteration:  1300 / 20000 [  6%]  (Warmup)
## Chain 1 Iteration:  1400 / 20000 [  7%]  (Warmup)
## Chain 1 Iteration:  1500 / 20000 [  7%]  (Warmup)
## Chain 1 Iteration:  1600 / 20000 [  8%]  (Warmup)
## Chain 1 Iteration:  1700 / 20000 [  8%]  (Warmup)
## Chain 1 Iteration:  1800 / 20000 [  9%]  (Warmup)
## Chain 1 Iteration:  1900 / 20000 [  9%]  (Warmup)
## Chain 1 Iteration:  2000 / 20000 [ 10%]  (Warmup)
## Chain 1 Iteration:  2100 / 20000 [ 10%]  (Warmup)
## Chain 1 Iteration:  2200 / 20000 [ 11%]  (Warmup)
## Chain 1 Iteration:  2300 / 20000 [ 11%]  (Warmup)
## Chain 1 Iteration:  2400 / 20000 [ 12%]  (Warmup)
## Chain 1 Iteration:  2500 / 20000 [ 12%]  (Warmup)
## Chain 1 Iteration:  2600 / 20000 [ 13%]  (Warmup)
## Chain 1 Iteration:  2700 / 20000 [ 13%]  (Warmup)
## Chain 1 Iteration:  2800 / 20000 [ 14%]  (Warmup)
## Chain 1 Iteration:  2900 / 20000 [ 14%]  (Warmup)
## Chain 1 Iteration:  3000 / 20000 [ 15%]  (Warmup)
## Chain 1 Iteration:  3100 / 20000 [ 15%]  (Warmup)
## Chain 1 Iteration:  3200 / 20000 [ 16%]  (Warmup)
## Chain 1 Iteration:  3300 / 20000 [ 16%]  (Warmup)
## Chain 1 Iteration:  3400 / 20000 [ 17%]  (Warmup)
## Chain 1 Iteration:  3500 / 20000 [ 17%]  (Warmup)
## Chain 1 Iteration:  3600 / 20000 [ 18%]  (Warmup)
## Chain 1 Iteration:  3700 / 20000 [ 18%]  (Warmup)
## Chain 1 Iteration:  3800 / 20000 [ 19%]  (Warmup)
## Chain 1 Iteration:  3900 / 20000 [ 19%]  (Warmup)
## Chain 1 Iteration:  4000 / 20000 [ 20%]  (Warmup)
## Chain 1 Iteration:  4100 / 20000 [ 20%]  (Warmup)
## Chain 1 Iteration:  4200 / 20000 [ 21%]  (Warmup)
## Chain 1 Iteration:  4300 / 20000 [ 21%]  (Warmup)
## Chain 1 Iteration:  4400 / 20000 [ 22%]  (Warmup)
## Chain 1 Iteration:  4500 / 20000 [ 22%]  (Warmup)
## Chain 1 Iteration:  4600 / 20000 [ 23%]  (Warmup)
## Chain 1 Iteration:  4700 / 20000 [ 23%]  (Warmup)
## Chain 1 Iteration:  4800 / 20000 [ 24%]  (Warmup)
## Chain 1 Iteration:  4900 / 20000 [ 24%]  (Warmup)
## Chain 1 Iteration:  5000 / 20000 [ 25%]  (Warmup)
## Chain 1 Iteration:  5100 / 20000 [ 25%]  (Warmup)
## Chain 1 Iteration:  5200 / 20000 [ 26%]  (Warmup)
## Chain 1 Iteration:  5300 / 20000 [ 26%]  (Warmup)
## Chain 1 Iteration:  5400 / 20000 [ 27%]  (Warmup)
## Chain 1 Iteration:  5500 / 20000 [ 27%]  (Warmup)
## Chain 1 Iteration:  5600 / 20000 [ 28%]  (Warmup)
## Chain 1 Iteration:  5700 / 20000 [ 28%]  (Warmup)
## Chain 1 Iteration:  5800 / 20000 [ 29%]  (Warmup)
## Chain 1 Iteration:  5900 / 20000 [ 29%]  (Warmup)
## Chain 1 Iteration:  6000 / 20000 [ 30%]  (Warmup)
## Chain 1 Iteration:  6100 / 20000 [ 30%]  (Warmup)
## Chain 1 Iteration:  6200 / 20000 [ 31%]  (Warmup)
## Chain 1 Iteration:  6300 / 20000 [ 31%]  (Warmup)
## Chain 1 Iteration:  6400 / 20000 [ 32%]  (Warmup)
## Chain 1 Iteration:  6500 / 20000 [ 32%]  (Warmup)
## Chain 1 Iteration:  6600 / 20000 [ 33%]  (Warmup)
## Chain 1 Iteration:  6700 / 20000 [ 33%]  (Warmup)
## Chain 1 Iteration:  6800 / 20000 [ 34%]  (Warmup)
## Chain 1 Iteration:  6900 / 20000 [ 34%]  (Warmup)
## Chain 1 Iteration:  7000 / 20000 [ 35%]  (Warmup)
## Chain 1 Iteration:  7100 / 20000 [ 35%]  (Warmup)
## Chain 1 Iteration:  7200 / 20000 [ 36%]  (Warmup)
## Chain 1 Iteration:  7300 / 20000 [ 36%]  (Warmup)
## Chain 1 Iteration:  7400 / 20000 [ 37%]  (Warmup)
## Chain 1 Iteration:  7500 / 20000 [ 37%]  (Warmup)
## Chain 1 Iteration:  7600 / 20000 [ 38%]  (Warmup)
## Chain 1 Iteration:  7700 / 20000 [ 38%]  (Warmup)
## Chain 1 Iteration:  7800 / 20000 [ 39%]  (Warmup)
## Chain 1 Iteration:  7900 / 20000 [ 39%]  (Warmup)
## Chain 1 Iteration:  8000 / 20000 [ 40%]  (Warmup)
## Chain 1 Iteration:  8100 / 20000 [ 40%]  (Warmup)
## Chain 1 Iteration:  8200 / 20000 [ 41%]  (Warmup)
## Chain 1 Iteration:  8300 / 20000 [ 41%]  (Warmup)
## Chain 1 Iteration:  8400 / 20000 [ 42%]  (Warmup)
## Chain 1 Iteration:  8500 / 20000 [ 42%]  (Warmup)
## Chain 1 Iteration:  8600 / 20000 [ 43%]  (Warmup)
## Chain 1 Iteration:  8700 / 20000 [ 43%]  (Warmup)
## Chain 1 Iteration:  8800 / 20000 [ 44%]  (Warmup)
## Chain 1 Iteration:  8900 / 20000 [ 44%]  (Warmup)
## Chain 1 Iteration:  9000 / 20000 [ 45%]  (Warmup)
## Chain 1 Iteration:  9100 / 20000 [ 45%]  (Warmup)
## Chain 1 Iteration:  9200 / 20000 [ 46%]  (Warmup)
## Chain 1 Iteration:  9300 / 20000 [ 46%]  (Warmup)
## Chain 1 Iteration:  9400 / 20000 [ 47%]  (Warmup)
## Chain 1 Iteration:  9500 / 20000 [ 47%]  (Warmup)
## Chain 1 Iteration:  9600 / 20000 [ 48%]  (Warmup)
## Chain 1 Iteration:  9700 / 20000 [ 48%]  (Warmup)
## Chain 1 Iteration:  9800 / 20000 [ 49%]  (Warmup)
## Chain 1 Iteration:  9900 / 20000 [ 49%]  (Warmup)
## Chain 1 Iteration: 10000 / 20000 [ 50%]  (Warmup)
## Chain 1 Iteration: 10001 / 20000 [ 50%]  (Sampling)
## Chain 1 Iteration: 10100 / 20000 [ 50%]  (Sampling)
## Chain 1 Iteration: 10200 / 20000 [ 51%]  (Sampling)
## Chain 1 Iteration: 10300 / 20000 [ 51%]  (Sampling)
## Chain 1 Iteration: 10400 / 20000 [ 52%]  (Sampling)
## Chain 1 Iteration: 10500 / 20000 [ 52%]  (Sampling)
## Chain 1 Iteration: 10600 / 20000 [ 53%]  (Sampling)
## Chain 1 Iteration: 10700 / 20000 [ 53%]  (Sampling)
## Chain 1 Iteration: 10800 / 20000 [ 54%]  (Sampling)
## Chain 1 Iteration: 10900 / 20000 [ 54%]  (Sampling)
## Chain 1 Iteration: 11000 / 20000 [ 55%]  (Sampling)
## Chain 1 Iteration: 11100 / 20000 [ 55%]  (Sampling)
## Chain 1 Iteration: 11200 / 20000 [ 56%]  (Sampling)
## Chain 1 Iteration: 11300 / 20000 [ 56%]  (Sampling)
## Chain 1 Iteration: 11400 / 20000 [ 57%]  (Sampling)
## Chain 1 Iteration: 11500 / 20000 [ 57%]  (Sampling)
## Chain 1 Iteration: 11600 / 20000 [ 58%]  (Sampling)
## Chain 1 Iteration: 11700 / 20000 [ 58%]  (Sampling)
## Chain 1 Iteration: 11800 / 20000 [ 59%]  (Sampling)
## Chain 1 Iteration: 11900 / 20000 [ 59%]  (Sampling)
## Chain 1 Iteration: 12000 / 20000 [ 60%]  (Sampling)
## Chain 1 Iteration: 12100 / 20000 [ 60%]  (Sampling)
## Chain 1 Iteration: 12200 / 20000 [ 61%]  (Sampling)
## Chain 1 Iteration: 12300 / 20000 [ 61%]  (Sampling)
## Chain 1 Iteration: 12400 / 20000 [ 62%]  (Sampling)
## Chain 1 Iteration: 12500 / 20000 [ 62%]  (Sampling)
## Chain 1 Iteration: 12600 / 20000 [ 63%]  (Sampling)
## Chain 1 Iteration: 12700 / 20000 [ 63%]  (Sampling)
## Chain 1 Iteration: 12800 / 20000 [ 64%]  (Sampling)
## Chain 1 Iteration: 12900 / 20000 [ 64%]  (Sampling)
## Chain 1 Iteration: 13000 / 20000 [ 65%]  (Sampling)
## Chain 1 Iteration: 13100 / 20000 [ 65%]  (Sampling)
## Chain 1 Iteration: 13200 / 20000 [ 66%]  (Sampling)
## Chain 1 Iteration: 13300 / 20000 [ 66%]  (Sampling)
## Chain 1 Iteration: 13400 / 20000 [ 67%]  (Sampling)
## Chain 1 Iteration: 13500 / 20000 [ 67%]  (Sampling)
## Chain 1 Iteration: 13600 / 20000 [ 68%]  (Sampling)
## Chain 1 Iteration: 13700 / 20000 [ 68%]  (Sampling)
## Chain 1 Iteration: 13800 / 20000 [ 69%]  (Sampling)
## Chain 1 Iteration: 13900 / 20000 [ 69%]  (Sampling)
## Chain 1 Iteration: 14000 / 20000 [ 70%]  (Sampling)
## Chain 1 Iteration: 14100 / 20000 [ 70%]  (Sampling)
## Chain 1 Iteration: 14200 / 20000 [ 71%]  (Sampling)
## Chain 1 Iteration: 14300 / 20000 [ 71%]  (Sampling)
## Chain 1 Iteration: 14400 / 20000 [ 72%]  (Sampling)
## Chain 1 Iteration: 14500 / 20000 [ 72%]  (Sampling)
## Chain 1 Iteration: 14600 / 20000 [ 73%]  (Sampling)
## Chain 1 Iteration: 14700 / 20000 [ 73%]  (Sampling)
## Chain 1 Iteration: 14800 / 20000 [ 74%]  (Sampling)
## Chain 1 Iteration: 14900 / 20000 [ 74%]  (Sampling)
## Chain 1 Iteration: 15000 / 20000 [ 75%]  (Sampling)
## Chain 1 Iteration: 15100 / 20000 [ 75%]  (Sampling)
## Chain 1 Iteration: 15200 / 20000 [ 76%]  (Sampling)
## Chain 1 Iteration: 15300 / 20000 [ 76%]  (Sampling)
## Chain 1 Iteration: 15400 / 20000 [ 77%]  (Sampling)
## Chain 1 Iteration: 15500 / 20000 [ 77%]  (Sampling)
## Chain 1 Iteration: 15600 / 20000 [ 78%]  (Sampling)
## Chain 1 Iteration: 15700 / 20000 [ 78%]  (Sampling)
## Chain 1 Iteration: 15800 / 20000 [ 79%]  (Sampling)
## Chain 1 Iteration: 15900 / 20000 [ 79%]  (Sampling)
## Chain 1 Iteration: 16000 / 20000 [ 80%]  (Sampling)
## Chain 1 Iteration: 16100 / 20000 [ 80%]  (Sampling)
## Chain 1 Iteration: 16200 / 20000 [ 81%]  (Sampling)
## Chain 1 Iteration: 16300 / 20000 [ 81%]  (Sampling)
## Chain 1 Iteration: 16400 / 20000 [ 82%]  (Sampling)
## Chain 1 Iteration: 16500 / 20000 [ 82%]  (Sampling)
## Chain 1 Iteration: 16600 / 20000 [ 83%]  (Sampling)
## Chain 1 Iteration: 16700 / 20000 [ 83%]  (Sampling)
## Chain 1 Iteration: 16800 / 20000 [ 84%]  (Sampling)
## Chain 1 Iteration: 16900 / 20000 [ 84%]  (Sampling)
## Chain 1 Iteration: 17000 / 20000 [ 85%]  (Sampling)
## Chain 1 Iteration: 17100 / 20000 [ 85%]  (Sampling)
## Chain 1 Iteration: 17200 / 20000 [ 86%]  (Sampling)
## Chain 1 Iteration: 17300 / 20000 [ 86%]  (Sampling)
## Chain 1 Iteration: 17400 / 20000 [ 87%]  (Sampling)
## Chain 1 Iteration: 17500 / 20000 [ 87%]  (Sampling)
## Chain 1 Iteration: 17600 / 20000 [ 88%]  (Sampling)
## Chain 1 Iteration: 17700 / 20000 [ 88%]  (Sampling)
## Chain 1 Iteration: 17800 / 20000 [ 89%]  (Sampling)
## Chain 1 Iteration: 17900 / 20000 [ 89%]  (Sampling)
## Chain 1 Iteration: 18000 / 20000 [ 90%]  (Sampling)
## Chain 1 Iteration: 18100 / 20000 [ 90%]  (Sampling)
## Chain 1 Iteration: 18200 / 20000 [ 91%]  (Sampling)
## Chain 1 Iteration: 18300 / 20000 [ 91%]  (Sampling)
## Chain 1 Iteration: 18400 / 20000 [ 92%]  (Sampling)
## Chain 1 Iteration: 18500 / 20000 [ 92%]  (Sampling)
## Chain 1 Iteration: 18600 / 20000 [ 93%]  (Sampling)
## Chain 1 Iteration: 18700 / 20000 [ 93%]  (Sampling)
## Chain 1 Iteration: 18800 / 20000 [ 94%]  (Sampling)
## Chain 1 Iteration: 18900 / 20000 [ 94%]  (Sampling)
## Chain 1 Iteration: 19000 / 20000 [ 95%]  (Sampling)
## Chain 1 Iteration: 19100 / 20000 [ 95%]  (Sampling)
## Chain 1 Iteration: 19200 / 20000 [ 96%]  (Sampling)
## Chain 1 Iteration: 19300 / 20000 [ 96%]  (Sampling)
## Chain 1 Iteration: 19400 / 20000 [ 97%]  (Sampling)
## Chain 1 Iteration: 19500 / 20000 [ 97%]  (Sampling)
## Chain 1 Iteration: 19600 / 20000 [ 98%]  (Sampling)
## Chain 1 Iteration: 19700 / 20000 [ 98%]  (Sampling)
## Chain 1 Iteration: 19800 / 20000 [ 99%]  (Sampling)
## Chain 1 Iteration: 19900 / 20000 [ 99%]  (Sampling)
## Chain 1 Iteration: 20000 / 20000 [100%]  (Sampling)
## Chain 1 finished in 1.9 seconds.
```

```
prior_m2c <- extract.prior(m2c, n=1e4)
```

```
## Warning in '/var/folders/wx/1_76tj0s15gc4yxmndvw4l_00000gn/T/RtmpGmhclW/model-9fc06ed3d5ee.stan', li
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc
```

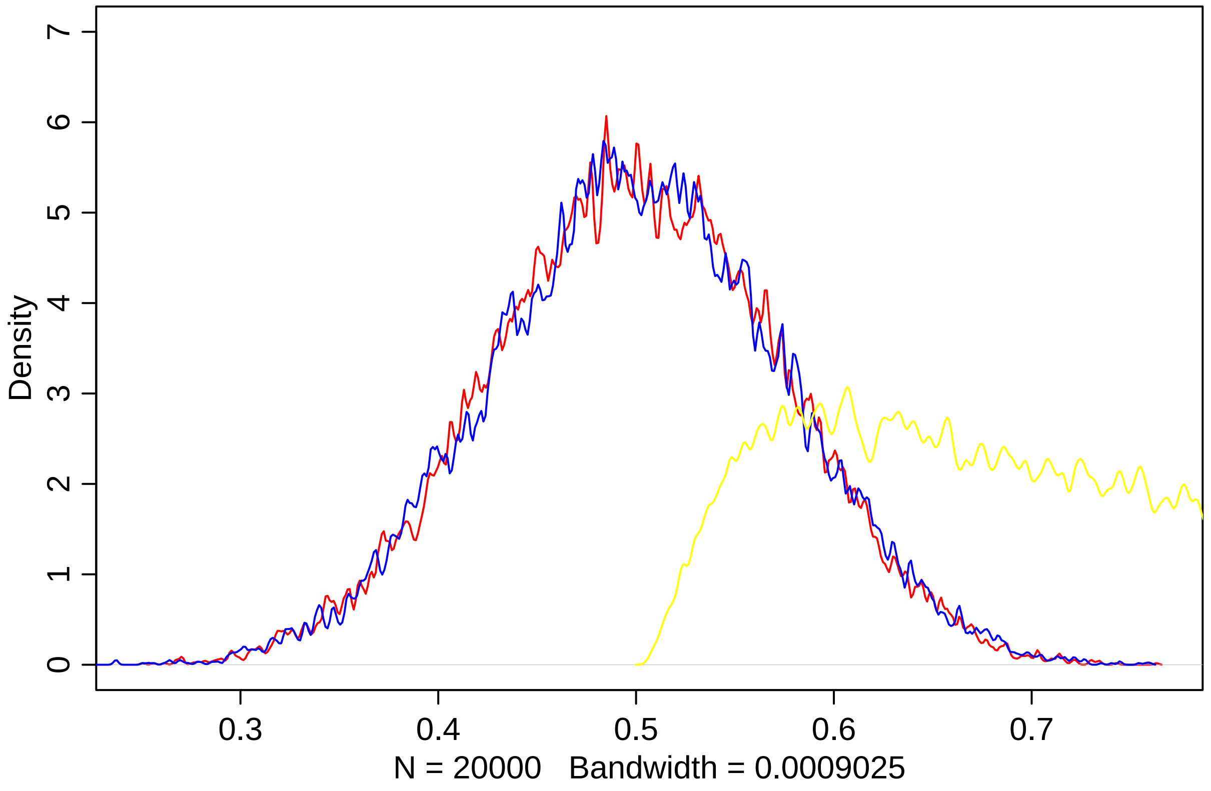
```
## Running MCMC with 1 chain, with 1 thread(s) per chain...
##
## Chain 1 Iteration:     1 / 20000 [  0%]  (Warmup)
## Chain 1 Iteration:   100 / 20000 [  0%]  (Warmup)
## Chain 1 Iteration:   200 / 20000 [  1%]  (Warmup)
## Chain 1 Iteration:   300 / 20000 [  1%]  (Warmup)
## Chain 1 Iteration:   400 / 20000 [  2%]  (Warmup)
## Chain 1 Iteration:   500 / 20000 [  2%]  (Warmup)
## Chain 1 Iteration:   600 / 20000 [  3%]  (Warmup)
## Chain 1 Iteration:   700 / 20000 [  3%]  (Warmup)
## Chain 1 Iteration:   800 / 20000 [  4%]  (Warmup)
## Chain 1 Iteration:   900 / 20000 [  4%]  (Warmup)
## Chain 1 Iteration:  1000 / 20000 [  5%]  (Warmup)
## Chain 1 Iteration:  1100 / 20000 [  5%]  (Warmup)
## Chain 1 Iteration:  1200 / 20000 [  6%]  (Warmup)
## Chain 1 Iteration:  1300 / 20000 [  6%]  (Warmup)
## Chain 1 Iteration:  1400 / 20000 [  7%]  (Warmup)
## Chain 1 Iteration:  1500 / 20000 [  7%]  (Warmup)
## Chain 1 Iteration:  1600 / 20000 [  8%]  (Warmup)
## Chain 1 Iteration:  1700 / 20000 [  8%]  (Warmup)
## Chain 1 Iteration:  1800 / 20000 [  9%]  (Warmup)
## Chain 1 Iteration:  1900 / 20000 [  9%]  (Warmup)
## Chain 1 Iteration:  2000 / 20000 [ 10%]  (Warmup)
## Chain 1 Iteration:  2100 / 20000 [ 10%]  (Warmup)
## Chain 1 Iteration:  2200 / 20000 [ 11%]  (Warmup)
## Chain 1 Iteration:  2300 / 20000 [ 11%]  (Warmup)
## Chain 1 Iteration:  2400 / 20000 [ 12%]  (Warmup)
## Chain 1 Iteration:  2500 / 20000 [ 12%]  (Warmup)
## Chain 1 Iteration:  2600 / 20000 [ 13%]  (Warmup)
## Chain 1 Iteration:  2700 / 20000 [ 13%]  (Warmup)
## Chain 1 Iteration:  2800 / 20000 [ 14%]  (Warmup)
## Chain 1 Iteration:  2900 / 20000 [ 14%]  (Warmup)
## Chain 1 Iteration:  3000 / 20000 [ 15%]  (Warmup)
## Chain 1 Iteration:  3100 / 20000 [ 15%]  (Warmup)
## Chain 1 Iteration:  3200 / 20000 [ 16%]  (Warmup)
## Chain 1 Iteration:  3300 / 20000 [ 16%]  (Warmup)
## Chain 1 Iteration:  3400 / 20000 [ 17%]  (Warmup)
## Chain 1 Iteration:  3500 / 20000 [ 17%]  (Warmup)
## Chain 1 Iteration:  3600 / 20000 [ 18%]  (Warmup)
## Chain 1 Iteration:  3700 / 20000 [ 18%]  (Warmup)
## Chain 1 Iteration:  3800 / 20000 [ 19%]  (Warmup)
## Chain 1 Iteration:  3900 / 20000 [ 19%]  (Warmup)
## Chain 1 Iteration:  4000 / 20000 [ 20%]  (Warmup)
## Chain 1 Iteration:  4100 / 20000 [ 20%]  (Warmup)
## Chain 1 Iteration:  4200 / 20000 [ 21%]  (Warmup)
## Chain 1 Iteration:  4300 / 20000 [ 21%]  (Warmup)
## Chain 1 Iteration:  4400 / 20000 [ 22%]  (Warmup)
## Chain 1 Iteration:  4500 / 20000 [ 22%]  (Warmup)
## Chain 1 Iteration:  4600 / 20000 [ 23%]  (Warmup)
## Chain 1 Iteration:  4700 / 20000 [ 23%]  (Warmup)
## Chain 1 Iteration:  4800 / 20000 [ 24%]  (Warmup)
## Chain 1 Iteration:  4900 / 20000 [ 24%]  (Warmup)
## Chain 1 Iteration:  5000 / 20000 [ 25%]  (Warmup)
## Chain 1 Iteration:  5100 / 20000 [ 25%]  (Warmup)
## Chain 1 Iteration:  5200 / 20000 [ 26%]  (Warmup)
## Chain 1 Iteration:  5300 / 20000 [ 26%]  (Warmup)
## Chain 1 Iteration:  5400 / 20000 [ 27%]  (Warmup)
## Chain 1 Iteration:  5500 / 20000 [ 27%]  (Warmup)
## Chain 1 Iteration:  5600 / 20000 [ 28%]  (Warmup)
## Chain 1 Iteration:  5700 / 20000 [ 28%]  (Warmup)
## Chain 1 Iteration:  5800 / 20000 [ 29%]  (Warmup)
## Chain 1 Iteration:  5900 / 20000 [ 29%]  (Warmup)
## Chain 1 Iteration:  6000 / 20000 [ 30%]  (Warmup)
## Chain 1 Iteration:  6100 / 20000 [ 30%]  (Warmup)
## Chain 1 Iteration:  6200 / 20000 [ 31%]  (Warmup)
## Chain 1 Iteration:  6300 / 20000 [ 31%]  (Warmup)
## Chain 1 Iteration:  6400 / 20000 [ 32%]  (Warmup)
## Chain 1 Iteration:  6500 / 20000 [ 32%]  (Warmup)
## Chain 1 Iteration:  6600 / 20000 [ 33%]  (Warmup)
## Chain 1 Iteration:  6700 / 20000 [ 33%]  (Warmup)
## Chain 1 Iteration:  6800 / 20000 [ 34%]  (Warmup)
## Chain 1 Iteration:  6900 / 20000 [ 34%]  (Warmup)
## Chain 1 Iteration:  7000 / 20000 [ 35%]  (Warmup)
## Chain 1 Iteration:  7100 / 20000 [ 35%]  (Warmup)
## Chain 1 Iteration:  7200 / 20000 [ 36%]  (Warmup)
## Chain 1 Iteration:  7300 / 20000 [ 36%]  (Warmup)
## Chain 1 Iteration:  7400 / 20000 [ 37%]  (Warmup)
## Chain 1 Iteration:  7500 / 20000 [ 37%]  (Warmup)
## Chain 1 Iteration:  7600 / 20000 [ 38%]  (Warmup)
## Chain 1 Iteration:  7700 / 20000 [ 38%]  (Warmup)
## Chain 1 Iteration:  7800 / 20000 [ 39%]  (Warmup)
## Chain 1 Iteration:  7900 / 20000 [ 39%]  (Warmup)
## Chain 1 Iteration:  8000 / 20000 [ 40%]  (Warmup)
## Chain 1 Iteration:  8100 / 20000 [ 40%]  (Warmup)
## Chain 1 Iteration:  8200 / 20000 [ 41%]  (Warmup)
## Chain 1 Iteration:  8300 / 20000 [ 41%]  (Warmup)
## Chain 1 Iteration:  8400 / 20000 [ 42%]  (Warmup)
## Chain 1 Iteration:  8500 / 20000 [ 42%]  (Warmup)
## Chain 1 Iteration:  8600 / 20000 [ 43%]  (Warmup)
## Chain 1 Iteration:  8700 / 20000 [ 43%]  (Warmup)
## Chain 1 Iteration:  8800 / 20000 [ 44%]  (Warmup)
## Chain 1 Iteration:  8900 / 20000 [ 44%]  (Warmup)
## Chain 1 Iteration:  9000 / 20000 [ 45%]  (Warmup)
## Chain 1 Iteration:  9100 / 20000 [ 45%]  (Warmup)
## Chain 1 Iteration:  9200 / 20000 [ 46%]  (Warmup)
## Chain 1 Iteration:  9300 / 20000 [ 46%]  (Warmup)
## Chain 1 Iteration:  9400 / 20000 [ 47%]  (Warmup)
## Chain 1 Iteration:  9500 / 20000 [ 47%]  (Warmup)
## Chain 1 Iteration:  9600 / 20000 [ 48%]  (Warmup)
## Chain 1 Iteration:  9700 / 20000 [ 48%]  (Warmup)
## Chain 1 Iteration:  9800 / 20000 [ 49%]  (Warmup)
## Chain 1 Iteration:  9900 / 20000 [ 49%]  (Warmup)
## Chain 1 Iteration: 10000 / 20000 [ 50%]  (Warmup)
## Chain 1 Iteration: 10001 / 20000 [ 50%]  (Sampling)
## Chain 1 Iteration: 10100 / 20000 [ 50%]  (Sampling)
## Chain 1 Iteration: 10200 / 20000 [ 51%]  (Sampling)
## Chain 1 Iteration: 10300 / 20000 [ 51%]  (Sampling)
## Chain 1 Iteration: 10400 / 20000 [ 52%]  (Sampling)
## Chain 1 Iteration: 10500 / 20000 [ 52%]  (Sampling)
## Chain 1 Iteration: 10600 / 20000 [ 53%]  (Sampling)
## Chain 1 Iteration: 10700 / 20000 [ 53%]  (Sampling)
## Chain 1 Iteration: 10800 / 20000 [ 54%]  (Sampling)
## Chain 1 Iteration: 10900 / 20000 [ 54%]  (Sampling)
## Chain 1 Iteration: 11000 / 20000 [ 55%]  (Sampling)
## Chain 1 Iteration: 11100 / 20000 [ 55%]  (Sampling)
## Chain 1 Iteration: 11200 / 20000 [ 56%]  (Sampling)
## Chain 1 Iteration: 11300 / 20000 [ 56%]  (Sampling)
## Chain 1 Iteration: 11400 / 20000 [ 57%]  (Sampling)
## Chain 1 Iteration: 11500 / 20000 [ 57%]  (Sampling)
## Chain 1 Iteration: 11600 / 20000 [ 58%]  (Sampling)
## Chain 1 Iteration: 11700 / 20000 [ 58%]  (Sampling)
## Chain 1 Iteration: 11800 / 20000 [ 59%]  (Sampling)
## Chain 1 Iteration: 11900 / 20000 [ 59%]  (Sampling)
## Chain 1 Iteration: 12000 / 20000 [ 60%]  (Sampling)
## Chain 1 Iteration: 12100 / 20000 [ 60%]  (Sampling)
## Chain 1 Iteration: 12200 / 20000 [ 61%]  (Sampling)
## Chain 1 Iteration: 12300 / 20000 [ 61%]  (Sampling)
## Chain 1 Iteration: 12400 / 20000 [ 62%]  (Sampling)
## Chain 1 Iteration: 12500 / 20000 [ 62%]  (Sampling)
## Chain 1 Iteration: 12600 / 20000 [ 63%]  (Sampling)
## Chain 1 Iteration: 12700 / 20000 [ 63%]  (Sampling)
## Chain 1 Iteration: 12800 / 20000 [ 64%]  (Sampling)
## Chain 1 Iteration: 12900 / 20000 [ 64%]  (Sampling)
## Chain 1 Iteration: 13000 / 20000 [ 65%]  (Sampling)
## Chain 1 Iteration: 13100 / 20000 [ 65%]  (Sampling)
## Chain 1 Iteration: 13200 / 20000 [ 66%]  (Sampling)
## Chain 1 Iteration: 13300 / 20000 [ 66%]  (Sampling)
## Chain 1 Iteration: 13400 / 20000 [ 67%]  (Sampling)
## Chain 1 Iteration: 13500 / 20000 [ 67%]  (Sampling)
## Chain 1 Iteration: 13600 / 20000 [ 68%]  (Sampling)
## Chain 1 Iteration: 13700 / 20000 [ 68%]  (Sampling)
## Chain 1 Iteration: 13800 / 20000 [ 69%]  (Sampling)
## Chain 1 Iteration: 13900 / 20000 [ 69%]  (Sampling)
## Chain 1 Iteration: 14000 / 20000 [ 70%]  (Sampling)
## Chain 1 Iteration: 14100 / 20000 [ 70%]  (Sampling)
## Chain 1 Iteration: 14200 / 20000 [ 71%]  (Sampling)
## Chain 1 Iteration: 14300 / 20000 [ 71%]  (Sampling)
## Chain 1 Iteration: 14400 / 20000 [ 72%]  (Sampling)
## Chain 1 Iteration: 14500 / 20000 [ 72%]  (Sampling)
## Chain 1 Iteration: 14600 / 20000 [ 73%]  (Sampling)
## Chain 1 Iteration: 14700 / 20000 [ 73%]  (Sampling)
## Chain 1 Iteration: 14800 / 20000 [ 74%]  (Sampling)
## Chain 1 Iteration: 14900 / 20000 [ 74%]  (Sampling)
## Chain 1 Iteration: 15000 / 20000 [ 75%]  (Sampling)
## Chain 1 Iteration: 15100 / 20000 [ 75%]  (Sampling)
## Chain 1 Iteration: 15200 / 20000 [ 76%]  (Sampling)
## Chain 1 Iteration: 15300 / 20000 [ 76%]  (Sampling)
## Chain 1 Iteration: 15400 / 20000 [ 77%]  (Sampling)
## Chain 1 Iteration: 15500 / 20000 [ 77%]  (Sampling)
## Chain 1 Iteration: 15600 / 20000 [ 78%]  (Sampling)
## Chain 1 Iteration: 15700 / 20000 [ 78%]  (Sampling)
## Chain 1 Iteration: 15800 / 20000 [ 79%]  (Sampling)
## Chain 1 Iteration: 15900 / 20000 [ 79%]  (Sampling)
## Chain 1 Iteration: 16000 / 20000 [ 80%]  (Sampling)
## Chain 1 Iteration: 16100 / 20000 [ 80%]  (Sampling)
## Chain 1 Iteration: 16200 / 20000 [ 81%]  (Sampling)
## Chain 1 Iteration: 16300 / 20000 [ 81%]  (Sampling)
## Chain 1 Iteration: 16400 / 20000 [ 82%]  (Sampling)
## Chain 1 Iteration: 16500 / 20000 [ 82%]  (Sampling)
## Chain 1 Iteration: 16600 / 20000 [ 83%]  (Sampling)
## Chain 1 Iteration: 16700 / 20000 [ 83%]  (Sampling)
## Chain 1 Iteration: 16800 / 20000 [ 84%]  (Sampling)
## Chain 1 Iteration: 16900 / 20000 [ 84%]  (Sampling)
## Chain 1 Iteration: 17000 / 20000 [ 85%]  (Sampling)
## Chain 1 Iteration: 17100 / 20000 [ 85%]  (Sampling)
## Chain 1 Iteration: 17200 / 20000 [ 86%]  (Sampling)
## Chain 1 Iteration: 17300 / 20000 [ 86%]  (Sampling)
## Chain 1 Iteration: 17400 / 20000 [ 87%]  (Sampling)
## Chain 1 Iteration: 17500 / 20000 [ 87%]  (Sampling)
## Chain 1 Iteration: 17600 / 20000 [ 88%]  (Sampling)
## Chain 1 Iteration: 17700 / 20000 [ 88%]  (Sampling)
## Chain 1 Iteration: 17800 / 20000 [ 89%]  (Sampling)
## Chain 1 Iteration: 17900 / 20000 [ 89%]  (Sampling)
## Chain 1 Iteration: 18000 / 20000 [ 90%]  (Sampling)
## Chain 1 Iteration: 18100 / 20000 [ 90%]  (Sampling)
## Chain 1 Iteration: 18200 / 20000 [ 91%]  (Sampling)
## Chain 1 Iteration: 18300 / 20000 [ 91%]  (Sampling)
## Chain 1 Iteration: 18400 / 20000 [ 92%]  (Sampling)
## Chain 1 Iteration: 18500 / 20000 [ 92%]  (Sampling)
## Chain 1 Iteration: 18600 / 20000 [ 93%]  (Sampling)
## Chain 1 Iteration: 18700 / 20000 [ 93%]  (Sampling)
## Chain 1 Iteration: 18800 / 20000 [ 94%]  (Sampling)
## Chain 1 Iteration: 18900 / 20000 [ 94%]  (Sampling)
## Chain 1 Iteration: 19000 / 20000 [ 95%]  (Sampling)
## Chain 1 Iteration: 19100 / 20000 [ 95%]  (Sampling)
## Chain 1 Iteration: 19200 / 20000 [ 96%]  (Sampling)
## Chain 1 Iteration: 19300 / 20000 [ 96%]  (Sampling)
## Chain 1 Iteration: 19400 / 20000 [ 97%]  (Sampling)
## Chain 1 Iteration: 19500 / 20000 [ 97%]  (Sampling)
## Chain 1 Iteration: 19600 / 20000 [ 98%]  (Sampling)
## Chain 1 Iteration: 19700 / 20000 [ 98%]  (Sampling)
## Chain 1 Iteration: 19800 / 20000 [ 99%]  (Sampling)
## Chain 1 Iteration: 19900 / 20000 [ 99%]  (Sampling)
## Chain 1 Iteration: 20000 / 20000 [100%]  (Sampling)
## Chain 1 finished in 2.0 seconds.
```

```
prior_m2d <- extract.prior(m2d, n=1e4)
```

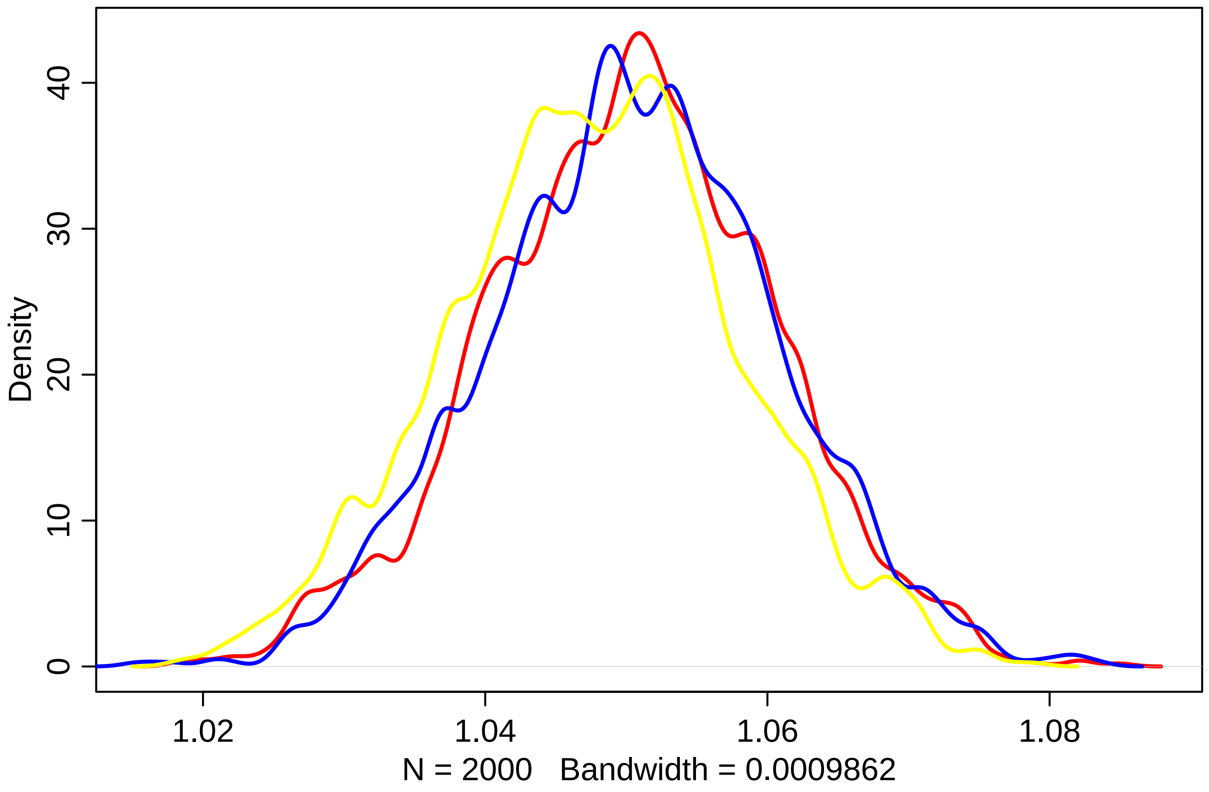
```
## Warning in '/var/folders/wx/1_76tj0s15gc4yxmndvw4l_00000gn/T/RtmpGmhclW/model-9fc027529bc9.stan', li
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc
```

```
## Running MCMC with 1 chain, with 1 thread(s) per chain...
##
## Chain 1 Iteration:     1 / 20000 [  0%]  (Warmup)
## Chain 1 Iteration:   100 / 20000 [  0%]  (Warmup)
## Chain 1 Iteration:   200 / 20000 [  1%]  (Warmup)
## Chain 1 Iteration:   300 / 20000 [  1%]  (Warmup)
## Chain 1 Iteration:   400 / 20000 [  2%]  (Warmup)
## Chain 1 Iteration:   500 / 20000 [  2%]  (Warmup)
## Chain 1 Iteration:   600 / 20000 [  3%]  (Warmup)
## Chain 1 Iteration:   700 / 20000 [  3%]  (Warmup)
## Chain 1 Iteration:   800 / 20000 [  4%]  (Warmup)
## Chain 1 Iteration:   900 / 20000 [  4%]  (Warmup)
## Chain 1 Iteration:  1000 / 20000 [  5%]  (Warmup)
## Chain 1 Iteration:  1100 / 20000 [  5%]  (Warmup)
## Chain 1 Iteration:  1200 / 20000 [  6%]  (Warmup)
## Chain 1 Iteration:  1300 / 20000 [  6%]  (Warmup)
## Chain 1 Iteration:  1400 / 20000 [  7%]  (Warmup)
## Chain 1 Iteration:  1500 / 20000 [  7%]  (Warmup)
## Chain 1 Iteration:  1600 / 20000 [  8%]  (Warmup)
## Chain 1 Iteration:  1700 / 20000 [  8%]  (Warmup)
## Chain 1 Iteration:  1800 / 20000 [  9%]  (Warmup)
## Chain 1 Iteration:  1900 / 20000 [  9%]  (Warmup)
## Chain 1 Iteration:  2000 / 20000 [ 10%]  (Warmup)
## Chain 1 Iteration:  2100 / 20000 [ 10%]  (Warmup)
## Chain 1 Iteration:  2200 / 20000 [ 11%]  (Warmup)
## Chain 1 Iteration:  2300 / 20000 [ 11%]  (Warmup)
## Chain 1 Iteration:  2400 / 20000 [ 12%]  (Warmup)
## Chain 1 Iteration:  2500 / 20000 [ 12%]  (Warmup)
## Chain 1 Iteration:  2600 / 20000 [ 13%]  (Warmup)
## Chain 1 Iteration:  2700 / 20000 [ 13%]  (Warmup)
## Chain 1 Iteration:  2800 / 20000 [ 14%]  (Warmup)
## Chain 1 Iteration:  2900 / 20000 [ 14%]  (Warmup)
## Chain 1 Iteration:  3000 / 20000 [ 15%]  (Warmup)
## Chain 1 Iteration:  3100 / 20000 [ 15%]  (Warmup)
## Chain 1 Iteration:  3200 / 20000 [ 16%]  (Warmup)
## Chain 1 Iteration:  3300 / 20000 [ 16%]  (Warmup)
## Chain 1 Iteration:  3400 / 20000 [ 17%]  (Warmup)
## Chain 1 Iteration:  3500 / 20000 [ 17%]  (Warmup)
## Chain 1 Iteration:  3600 / 20000 [ 18%]  (Warmup)
## Chain 1 Iteration:  3700 / 20000 [ 18%]  (Warmup)
## Chain 1 Iteration:  3800 / 20000 [ 19%]  (Warmup)
## Chain 1 Iteration:  3900 / 20000 [ 19%]  (Warmup)
## Chain 1 Iteration:  4000 / 20000 [ 20%]  (Warmup)
## Chain 1 Iteration:  4100 / 20000 [ 20%]  (Warmup)
## Chain 1 Iteration:  4200 / 20000 [ 21%]  (Warmup)
## Chain 1 Iteration:  4300 / 20000 [ 21%]  (Warmup)
## Chain 1 Iteration:  4400 / 20000 [ 22%]  (Warmup)
## Chain 1 Iteration:  4500 / 20000 [ 22%]  (Warmup)
## Chain 1 Iteration:  4600 / 20000 [ 23%]  (Warmup)
## Chain 1 Iteration:  4700 / 20000 [ 23%]  (Warmup)
## Chain 1 Iteration:  4800 / 20000 [ 24%]  (Warmup)
## Chain 1 Iteration:  4900 / 20000 [ 24%]  (Warmup)
## Chain 1 Iteration:  5000 / 20000 [ 25%]  (Warmup)
## Chain 1 Iteration:  5100 / 20000 [ 25%]  (Warmup)
## Chain 1 Iteration:  5200 / 20000 [ 26%]  (Warmup)
## Chain 1 Iteration:  5300 / 20000 [ 26%]  (Warmup)
## Chain 1 Iteration:  5400 / 20000 [ 27%]  (Warmup)
## Chain 1 Iteration:  5500 / 20000 [ 27%]  (Warmup)
## Chain 1 Iteration:  5600 / 20000 [ 28%]  (Warmup)
## Chain 1 Iteration:  5700 / 20000 [ 28%]  (Warmup)
## Chain 1 Iteration:  5800 / 20000 [ 29%]  (Warmup)
## Chain 1 Iteration:  5900 / 20000 [ 29%]  (Warmup)
## Chain 1 Iteration:  6000 / 20000 [ 30%]  (Warmup)
## Chain 1 Iteration:  6100 / 20000 [ 30%]  (Warmup)
## Chain 1 Iteration:  6200 / 20000 [ 31%]  (Warmup)
## Chain 1 Iteration:  6300 / 20000 [ 31%]  (Warmup)
## Chain 1 Iteration:  6400 / 20000 [ 32%]  (Warmup)
## Chain 1 Iteration:  6500 / 20000 [ 32%]  (Warmup)
## Chain 1 Iteration:  6600 / 20000 [ 33%]  (Warmup)
## Chain 1 Iteration:  6700 / 20000 [ 33%]  (Warmup)
## Chain 1 Iteration:  6800 / 20000 [ 34%]  (Warmup)
## Chain 1 Iteration:  6900 / 20000 [ 34%]  (Warmup)
## Chain 1 Iteration:  7000 / 20000 [ 35%]  (Warmup)
## Chain 1 Iteration:  7100 / 20000 [ 35%]  (Warmup)
## Chain 1 Iteration:  7200 / 20000 [ 36%]  (Warmup)
## Chain 1 Iteration:  7300 / 20000 [ 36%]  (Warmup)
## Chain 1 Iteration:  7400 / 20000 [ 37%]  (Warmup)
## Chain 1 Iteration:  7500 / 20000 [ 37%]  (Warmup)
## Chain 1 Iteration:  7600 / 20000 [ 38%]  (Warmup)
## Chain 1 Iteration:  7700 / 20000 [ 38%]  (Warmup)
## Chain 1 Iteration:  7800 / 20000 [ 39%]  (Warmup)
## Chain 1 Iteration:  7900 / 20000 [ 39%]  (Warmup)
## Chain 1 Iteration:  8000 / 20000 [ 40%]  (Warmup)
## Chain 1 Iteration:  8100 / 20000 [ 40%]  (Warmup)
## Chain 1 Iteration:  8200 / 20000 [ 41%]  (Warmup)
## Chain 1 Iteration:  8300 / 20000 [ 41%]  (Warmup)
## Chain 1 Iteration:  8400 / 20000 [ 42%]  (Warmup)
## Chain 1 Iteration:  8500 / 20000 [ 42%]  (Warmup)
## Chain 1 Iteration:  8600 / 20000 [ 43%]  (Warmup)
## Chain 1 Iteration:  8700 / 20000 [ 43%]  (Warmup)
## Chain 1 Iteration:  8800 / 20000 [ 44%]  (Warmup)
## Chain 1 Iteration:  8900 / 20000 [ 44%]  (Warmup)
## Chain 1 Iteration:  9000 / 20000 [ 45%]  (Warmup)
## Chain 1 Iteration:  9100 / 20000 [ 45%]  (Warmup)
## Chain 1 Iteration:  9200 / 20000 [ 46%]  (Warmup)
## Chain 1 Iteration:  9300 / 20000 [ 46%]  (Warmup)
## Chain 1 Iteration:  9400 / 20000 [ 47%]  (Warmup)
## Chain 1 Iteration:  9500 / 20000 [ 47%]  (Warmup)
## Chain 1 Iteration:  9600 / 20000 [ 48%]  (Warmup)
## Chain 1 Iteration:  9700 / 20000 [ 48%]  (Warmup)
## Chain 1 Iteration:  9800 / 20000 [ 49%]  (Warmup)
## Chain 1 Iteration:  9900 / 20000 [ 49%]  (Warmup)
## Chain 1 Iteration: 10000 / 20000 [ 50%]  (Warmup)
## Chain 1 Iteration: 10001 / 20000 [ 50%]  (Sampling)
## Chain 1 Iteration: 10100 / 20000 [ 50%]  (Sampling)
## Chain 1 Iteration: 10200 / 20000 [ 51%]  (Sampling)
## Chain 1 Iteration: 10300 / 20000 [ 51%]  (Sampling)
## Chain 1 Iteration: 10400 / 20000 [ 52%]  (Sampling)
## Chain 1 Iteration: 10500 / 20000 [ 52%]  (Sampling)
## Chain 1 Iteration: 10600 / 20000 [ 53%]  (Sampling)
## Chain 1 Iteration: 10700 / 20000 [ 53%]  (Sampling)
## Chain 1 Iteration: 10800 / 20000 [ 54%]  (Sampling)
## Chain 1 Iteration: 10900 / 20000 [ 54%]  (Sampling)
## Chain 1 Iteration: 11000 / 20000 [ 55%]  (Sampling)
## Chain 1 Iteration: 11100 / 20000 [ 55%]  (Sampling)
## Chain 1 Iteration: 11200 / 20000 [ 56%]  (Sampling)
## Chain 1 Iteration: 11300 / 20000 [ 56%]  (Sampling)
## Chain 1 Iteration: 11400 / 20000 [ 57%]  (Sampling)
## Chain 1 Iteration: 11500 / 20000 [ 57%]  (Sampling)
## Chain 1 Iteration: 11600 / 20000 [ 58%]  (Sampling)
## Chain 1 Iteration: 11700 / 20000 [ 58%]  (Sampling)
## Chain 1 Iteration: 11800 / 20000 [ 59%]  (Sampling)
## Chain 1 Iteration: 11900 / 20000 [ 59%]  (Sampling)
## Chain 1 Iteration: 12000 / 20000 [ 60%]  (Sampling)
## Chain 1 Iteration: 12100 / 20000 [ 60%]  (Sampling)
## Chain 1 Iteration: 12200 / 20000 [ 61%]  (Sampling)
## Chain 1 Iteration: 12300 / 20000 [ 61%]  (Sampling)
## Chain 1 Iteration: 12400 / 20000 [ 62%]  (Sampling)
## Chain 1 Iteration: 12500 / 20000 [ 62%]  (Sampling)
## Chain 1 Iteration: 12600 / 20000 [ 63%]  (Sampling)
## Chain 1 Iteration: 12700 / 20000 [ 63%]  (Sampling)
## Chain 1 Iteration: 12800 / 20000 [ 64%]  (Sampling)
## Chain 1 Iteration: 12900 / 20000 [ 64%]  (Sampling)
## Chain 1 Iteration: 13000 / 20000 [ 65%]  (Sampling)
## Chain 1 Iteration: 13100 / 20000 [ 65%]  (Sampling)
## Chain 1 Iteration: 13200 / 20000 [ 66%]  (Sampling)
## Chain 1 Iteration: 13300 / 20000 [ 66%]  (Sampling)
## Chain 1 Iteration: 13400 / 20000 [ 67%]  (Sampling)
## Chain 1 Iteration: 13500 / 20000 [ 67%]  (Sampling)
## Chain 1 Iteration: 13600 / 20000 [ 68%]  (Sampling)
## Chain 1 Iteration: 13700 / 20000 [ 68%]  (Sampling)
## Chain 1 Iteration: 13800 / 20000 [ 69%]  (Sampling)
## Chain 1 Iteration: 13900 / 20000 [ 69%]  (Sampling)
## Chain 1 Iteration: 14000 / 20000 [ 70%]  (Sampling)
## Chain 1 Iteration: 14100 / 20000 [ 70%]  (Sampling)
## Chain 1 Iteration: 14200 / 20000 [ 71%]  (Sampling)
## Chain 1 Iteration: 14300 / 20000 [ 71%]  (Sampling)
## Chain 1 Iteration: 14400 / 20000 [ 72%]  (Sampling)
## Chain 1 Iteration: 14500 / 20000 [ 72%]  (Sampling)
## Chain 1 Iteration: 14600 / 20000 [ 73%]  (Sampling)
## Chain 1 Iteration: 14700 / 20000 [ 73%]  (Sampling)
## Chain 1 Iteration: 14800 / 20000 [ 74%]  (Sampling)
## Chain 1 Iteration: 14900 / 20000 [ 74%]  (Sampling)
## Chain 1 Iteration: 15000 / 20000 [ 75%]  (Sampling)
## Chain 1 Iteration: 15100 / 20000 [ 75%]  (Sampling)
## Chain 1 Iteration: 15200 / 20000 [ 76%]  (Sampling)
## Chain 1 Iteration: 15300 / 20000 [ 76%]  (Sampling)
## Chain 1 Iteration: 15400 / 20000 [ 77%]  (Sampling)
## Chain 1 Iteration: 15500 / 20000 [ 77%]  (Sampling)
## Chain 1 Iteration: 15600 / 20000 [ 78%]  (Sampling)
## Chain 1 Iteration: 15700 / 20000 [ 78%]  (Sampling)
## Chain 1 Iteration: 15800 / 20000 [ 79%]  (Sampling)
## Chain 1 Iteration: 15900 / 20000 [ 79%]  (Sampling)
## Chain 1 Iteration: 16000 / 20000 [ 80%]  (Sampling)
## Chain 1 Iteration: 16100 / 20000 [ 80%]  (Sampling)
## Chain 1 Iteration: 16200 / 20000 [ 81%]  (Sampling)
## Chain 1 Iteration: 16300 / 20000 [ 81%]  (Sampling)
## Chain 1 Iteration: 16400 / 20000 [ 82%]  (Sampling)
## Chain 1 Iteration: 16500 / 20000 [ 82%]  (Sampling)
## Chain 1 Iteration: 16600 / 20000 [ 83%]  (Sampling)
## Chain 1 Iteration: 16700 / 20000 [ 83%]  (Sampling)
## Chain 1 Iteration: 16800 / 20000 [ 84%]  (Sampling)
## Chain 1 Iteration: 16900 / 20000 [ 84%]  (Sampling)
## Chain 1 Iteration: 17000 / 20000 [ 85%]  (Sampling)
## Chain 1 Iteration: 17100 / 20000 [ 85%]  (Sampling)
## Chain 1 Iteration: 17200 / 20000 [ 86%]  (Sampling)
## Chain 1 Iteration: 17300 / 20000 [ 86%]  (Sampling)
## Chain 1 Iteration: 17400 / 20000 [ 87%]  (Sampling)
## Chain 1 Iteration: 17500 / 20000 [ 87%]  (Sampling)
## Chain 1 Iteration: 17600 / 20000 [ 88%]  (Sampling)
## Chain 1 Iteration: 17700 / 20000 [ 88%]  (Sampling)
## Chain 1 Iteration: 17800 / 20000 [ 89%]  (Sampling)
## Chain 1 Iteration: 17900 / 20000 [ 89%]  (Sampling)
## Chain 1 Iteration: 18000 / 20000 [ 90%]  (Sampling)
## Chain 1 Iteration: 18100 / 20000 [ 90%]  (Sampling)
## Chain 1 Iteration: 18200 / 20000 [ 91%]  (Sampling)
## Chain 1 Iteration: 18300 / 20000 [ 91%]  (Sampling)
## Chain 1 Iteration: 18400 / 20000 [ 92%]  (Sampling)
## Chain 1 Iteration: 18500 / 20000 [ 92%]  (Sampling)
## Chain 1 Iteration: 18600 / 20000 [ 93%]  (Sampling)
## Chain 1 Iteration: 18700 / 20000 [ 93%]  (Sampling)
## Chain 1 Iteration: 18800 / 20000 [ 94%]  (Sampling)
## Chain 1 Iteration: 18900 / 20000 [ 94%]  (Sampling)
## Chain 1 Iteration: 19000 / 20000 [ 95%]  (Sampling)
## Chain 1 Iteration: 19100 / 20000 [ 95%]  (Sampling)
## Chain 1 Iteration: 19200 / 20000 [ 96%]  (Sampling)
## Chain 1 Iteration: 19300 / 20000 [ 96%]  (Sampling)
## Chain 1 Iteration: 19400 / 20000 [ 97%]  (Sampling)
## Chain 1 Iteration: 19500 / 20000 [ 97%]  (Sampling)
## Chain 1 Iteration: 19600 / 20000 [ 98%]  (Sampling)
## Chain 1 Iteration: 19700 / 20000 [ 98%]  (Sampling)
## Chain 1 Iteration: 19800 / 20000 [ 99%]  (Sampling)
## Chain 1 Iteration: 19900 / 20000 [ 99%]  (Sampling)
## Chain 1 Iteration: 20000 / 20000 [100%]  (Sampling)
## Chain 1 finished in 1.9 seconds.
```

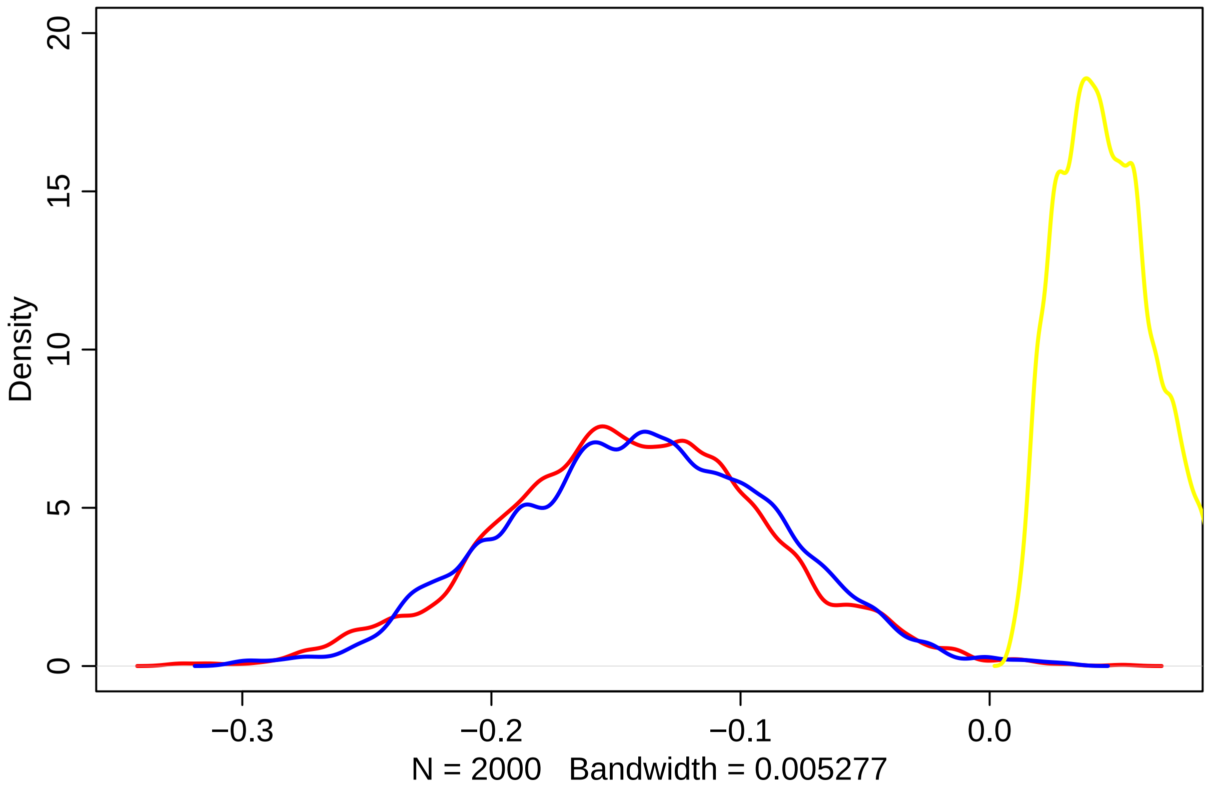
```
pr_m2a_4c <- inv_logit(prior_m2a_4c$b)
pr_m2c <- inv_logit(prior_m2c$b)
pr_m2d <- inv_logit(prior_m2d$b)

dens(pr_m2a_4c, adj = 0.1, col='red', ylim = c(0, 7))
dens(pr_m2c, adj = 0.1, add = TRUE, col='blue')
dens(pr_m2d, adj = 0.1, add = TRUE, col='yellow')
```



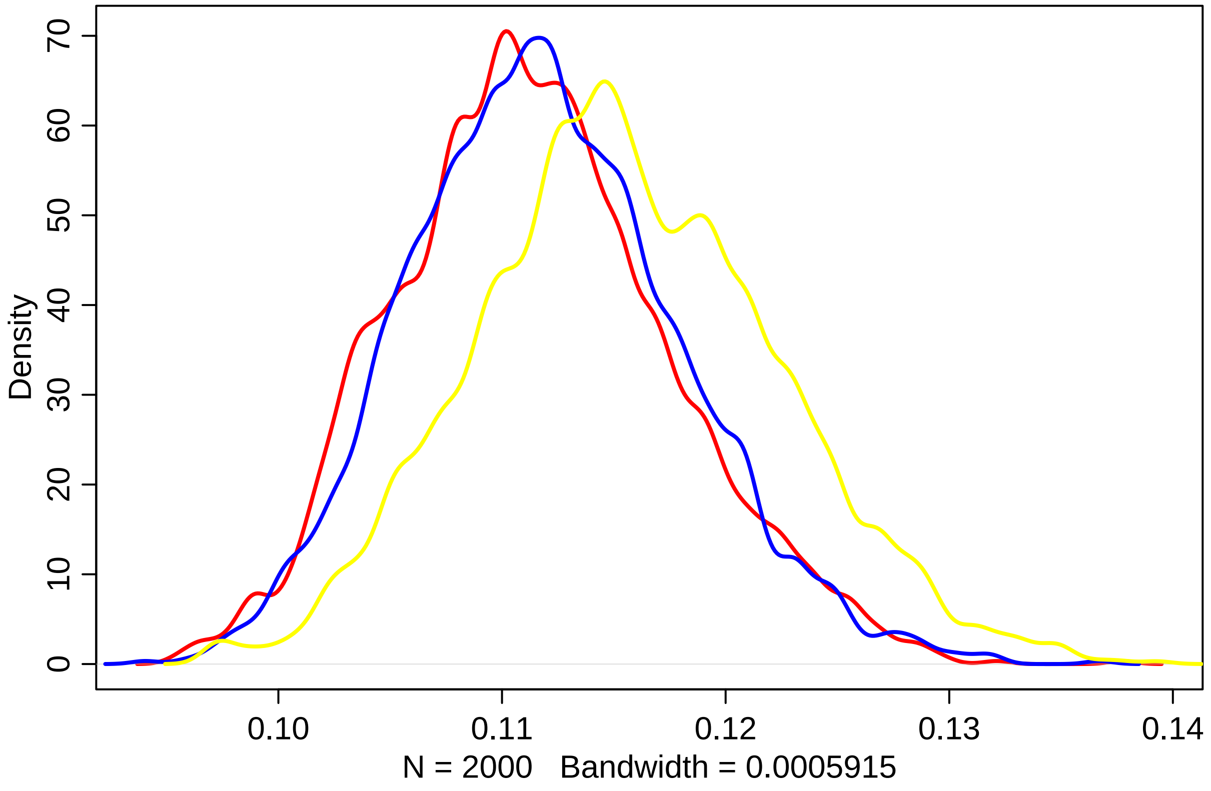
```
# comparing posterior distributions for 'a' in the three models
post_m2d <- extract.samples(m2d)

dens(post_m2a_4c$a[, 2], lwd = 2, col='red')
dens(post_m2c$a[, 2], add=TRUE, lwd = 2, col='blue')
dens(post_m2d$a[, 2], add=TRUE, lwd = 2, col='yellow')
```



```
# comparing posterior distributions for 'b' in the three models
post_m2d <- extract.samples(m2d)

dens(post_m2a_4c$b[, 2], lwd = 2, col='red', ylim = c(0, 20))
dens(post_m2c$b[, 2], add=TRUE, lwd = 2, col='blue')
dens(post_m2d$b[, 2], add=TRUE, lwd = 2, col='yellow')
```

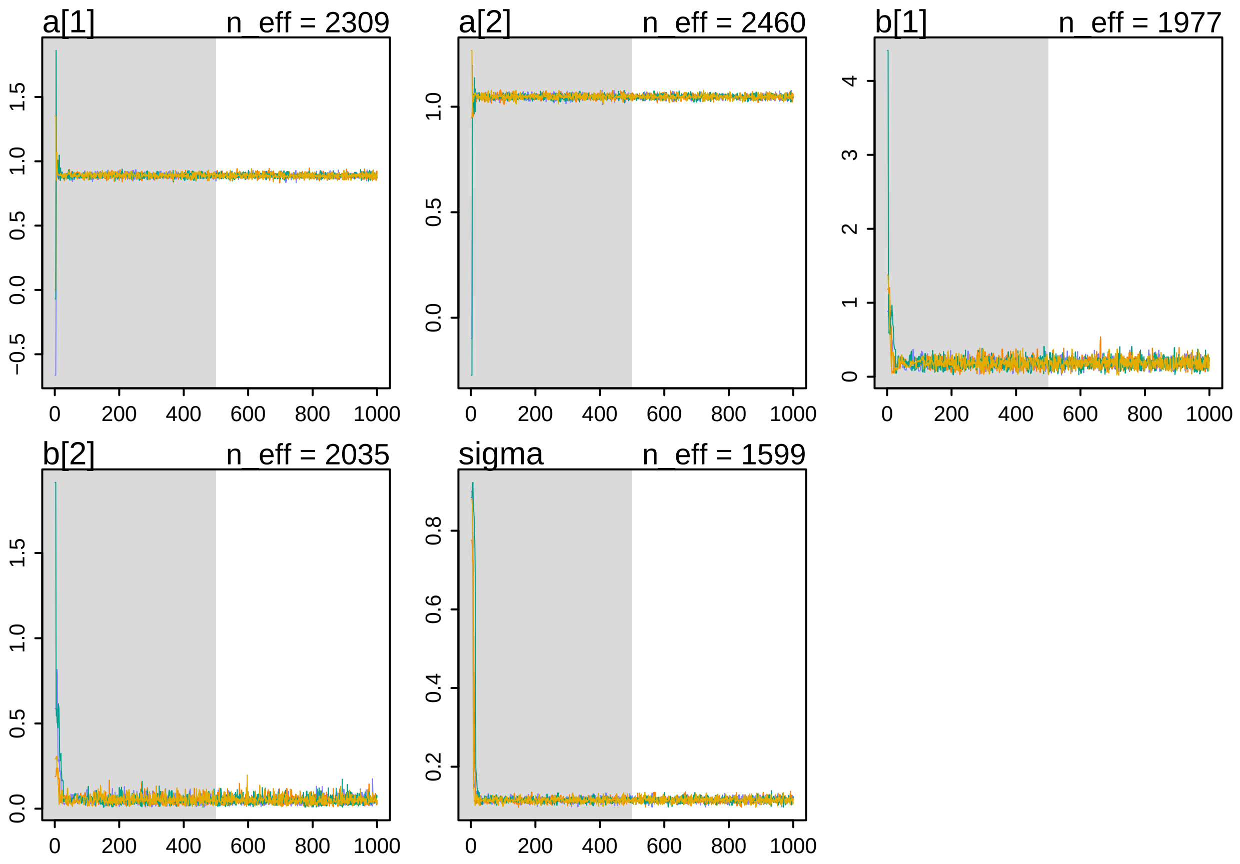


```
# comparing posterior distributions for 'sigma' in the three models
post_m2d <- extract.samples(m2d)

dens(post_m2a_4c$sigma, lwd = 2, col='red')
dens(post_m2c$sigma, add=TRUE, lwd = 2, col='blue')
dens(post_m2d$sigma, add=TRUE, lwd = 2, col='yellow')
```

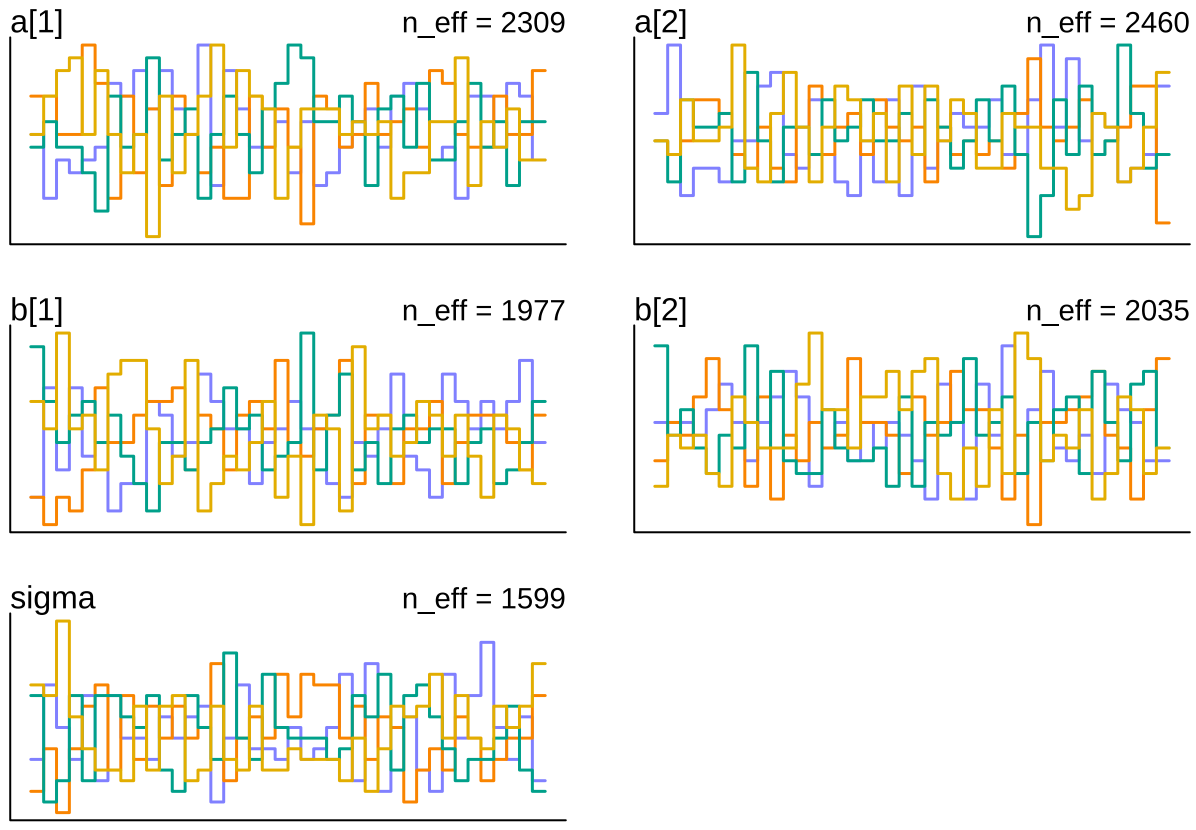


```
# The direction of LOG-NORMAL priors for model m2d as we can see from the prior plot above is
# towards the right.
# For this reason, Posterior distributions of 'b' values upon visualization also appear to be
# centered at higher values towards the right in model m2d in comparison to models m2a_4c and m2c.

# The posterior of sigma is also centered towards the right in comparison to the other two
# models that show overlap.

# Posterior of 'a' is almost nill (or little) affected
```

```
# double checking the hygiene of the chains for model m2d using 'traceplot', just to be sure
traceplot( m2d )
```



```
# double checking the hygiene of the chains for model m2d using 'trankplot', just to be sure
trankplot( m2d , n_cols=2 )
```



3. Binomial Regression

We started the course sampling marbles from a bucket to estimate its contents and tossing a globe to estimate the proportion of its surface covered in water. Each made use of the binomial distribution and was ideal to introduce the fundamentals of Bayesian inference. Nevertheless, Binomial regression – which is any type of GLM using a binomial mean-variance relationship – introduces complications that we needed to postpone until now. Return to the prosocial chimpanzee experiment in section §11.1 of the textbook, and the HMC model that features individual chimpanzee (actor) parameters actor and individual treatment parameters:

```
data(chimpanzees)
d <- chimpanzees

d$treatment <- 1 + d$prosoc_left + 2*d$condition

# prior trimmed data list
dat_list <- list(
pulled_left = d$pulled_left,
actor = d$actor,
treatment = as.integer(d$treatment) )
```

```
m11.4 <- ulam(
alist(
pulled_left ~ dbinom( 1, p ),
logit(p) <- a[actor] + b[treatment] ,
a[actor] ~ dnorm( 0 , 1.5 ),
b[treatment] ~ dnorm( 0, 0.5 )
), data = dat_list, chains=4, log_lik =TRUE) # See sec 11.1 to prepare dat_list
```

```
## Warning in '/var/folders/wx/1_76tj0s15gc4yxmndvw4l_00000gn/T/RtmpGmhclW/model-9fc01b5b901c.stan', li
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc
## Warning in '/var/folders/wx/1_76tj0s15gc4yxmndvw4l_00000gn/T/RtmpGmhclW/model-9fc01b5b901c.stan', li
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc
## Warning in '/var/folders/wx/1_76tj0s15gc4yxmndvw4l_00000gn/T/RtmpGmhclW/model-9fc01b5b901c.stan', li
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc
```

```
## Running MCMC with 4 sequential chains, with 1 thread(s) per chain...
##
## Chain 1 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 1 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1 finished in 1.2 seconds.
## Chain 2 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 2 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 2 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 2 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 2 finished in 1.2 seconds.
## Chain 3 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 3 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 3 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 3 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 3 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 3 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 3 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 3 finished in 1.3 seconds.
## Chain 4 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 4 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 4 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 4 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 4 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 4 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 4 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 4 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 4 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 4 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 4 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 4 finished in 1.2 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 1.2 seconds.
## Total execution time: 5.4 seconds.
```

```
precis( m11.4 , 2 )
```

```
##             mean        sd         5.5%       94.5%     n_eff    Rhat4
## a[1] -0.43501145 0.3336959 -0.958290190  0.08650944  884.2202 1.002139
## a[2]  3.91867199 0.7452340  2.786400200  5.17299685 1372.6797 1.000249
## a[3] -0.72760523 0.3431561 -1.279529950 -0.19017385  862.2777 1.004625
## a[4] -0.73497870 0.3415706 -1.295135950 -0.19890931  936.4909 1.003805
## a[5] -0.43836890 0.3317364 -0.950776795  0.09068255  794.0829 1.000653
## a[6]  0.49194936 0.3319405 -0.046120778  1.01449770  895.3917 1.000024
## a[7]  1.96146430 0.4191940  1.298867350  2.64294025 1062.7185 1.002338
## b[1] -0.05249907 0.2921596 -0.523759010  0.42312383  812.5213 1.000806
## b[2]  0.46763316 0.2932638 -0.002325271  0.95087633  791.9927 1.002260
## b[3] -0.39032643 0.2978968 -0.879605820  0.07980406  747.6746 1.002363
## b[4]  0.35248845 0.2897704 -0.113752225  0.81854004  784.1275 1.002254
```

a) Compare m11.4 to a Laplacian quadratic approximate posterior distribution, constructed using quap(), that also includes individual parameters for actor and treatment. What are the differences and similarities between the two approximate posteriors? Explain your answer.

```
m3a <- quap(
alist(
pulled_left ~ dbinom( 1, p ),
logit(p) <- a[actor] + b[treatment],
a[actor] ~ dnorm( 0 , 1.5 ),
b[treatment] ~ dnorm( 0, 0.5 )
) , data=dat_list )
precis(m3a,depth=2)
```

```
##             mean        sd        5.5%       94.5%
## a[1] -0.43919313 0.3276014 -0.96276350  0.08437724
## a[2]  3.70607160 0.7217542  2.55256895  4.85957425
## a[3] -0.73272774 0.3329749 -1.26488593 -0.20056956
## a[4] -0.73277182 0.3329760 -1.26493171 -0.20061193
## a[5] -0.43920497 0.3276016 -0.96277558  0.08436564
## a[6]  0.46895451 0.3317749 -0.06128581  0.99919483
## a[7]  1.90505686 0.4136438  1.24397418  2.56613954
## b[1] -0.04065249 0.2837319 -0.49411080  0.41280582
## b[2]  0.47214126 0.2842163  0.01790879  0.92637373
## b[3] -0.37834280 0.2852440 -0.83421781  0.07753222
## b[4]  0.36201240 0.2838328 -0.09160719  0.81563199
```

```
# comparing models using PSIS
compare( m11.4, m3a, func=PSIS)
```

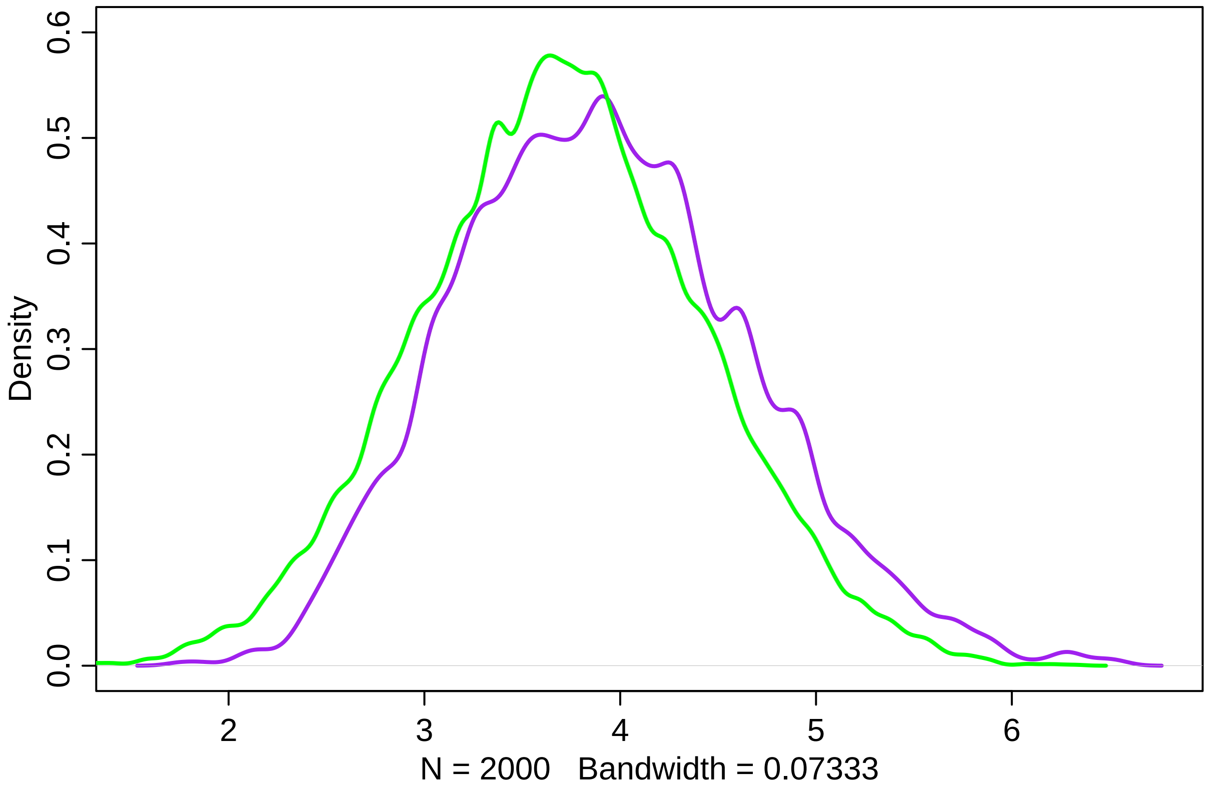
```
## Warning in compare(m11.4, m3a, func = PSIS): Not all model fits of same class.
## This is usually a bad idea, because it implies they were fit by different algorithms.
## Check yourself, before you wreck yourself.
```

```
##           PSIS       SE     dPSIS       dSE    pPSIS    weight
## m11.4 532.3654 18.89532 0.0000000        NA 8.524922 0.5286727
## m3a   532.5950 18.62595 0.2296333 0.3682554 8.236797 0.4713273
```

```
# PSIS suggests that both models behave almost with similar precision. The PSIS values of both the
# models are same
```

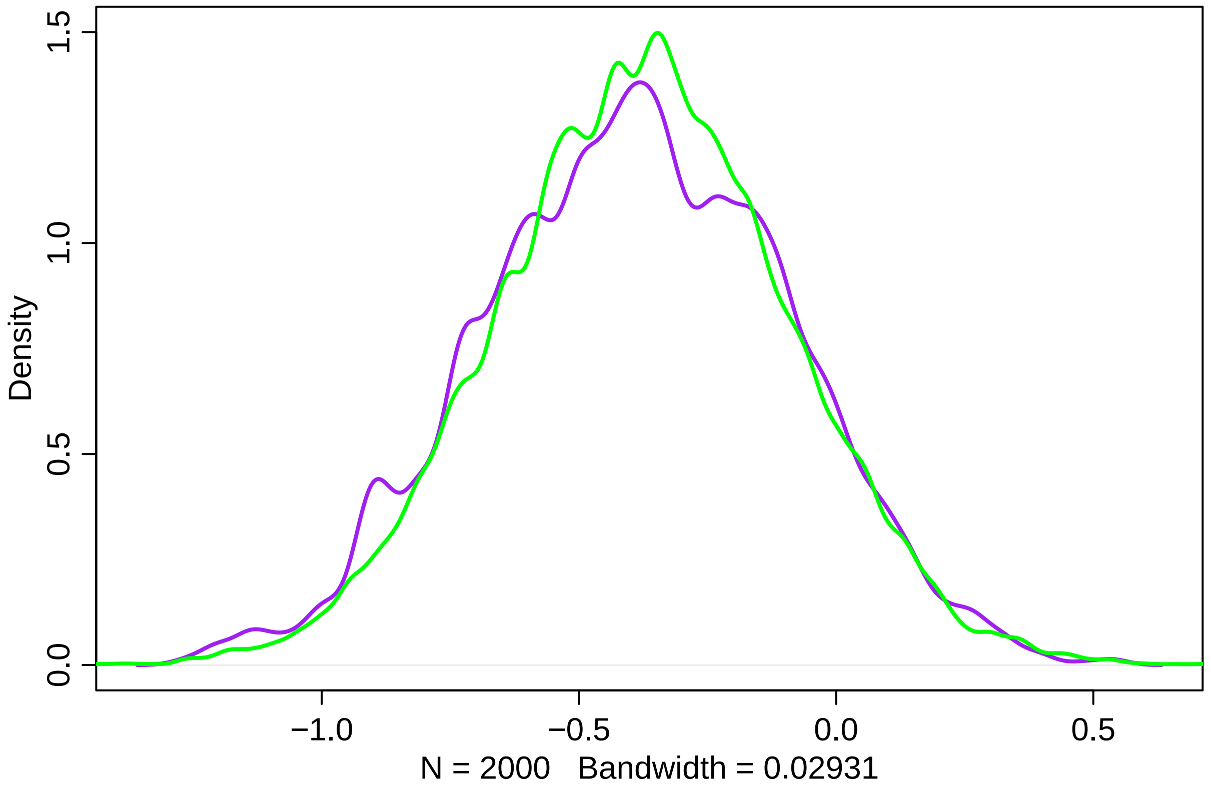
```
# There is some difference in the a[2] values - there is a higher estimate with the ulam model.
```

```
# now comparing posterior distributions for 'a' in the two models
post_m11.4 <- extract.samples(m11.4)
post_m3a <- extract.samples(m3a)

dens(post_m11.4$a[, 2], lwd = 2, col='purple', ylim = c(0, 0.6))
dens(post_m3a$a[, 2], add=TRUE, lwd = 2, col='green')
```



```
# now comparing posterior distributions for 'b' in the two models
post_m11.4 <- extract.samples(m11.4)
post_m3a <- extract.samples(m3a)
```

```
dens(post_m11.4$b[, 3], lwd = 2, col='purple', ylim = c(0, 1.5))
dens(post_m3a$b[, 3], add=TRUE, lwd = 2, col='green')
```



```
# From the above visualization of posteriors, it is clear that both the models 'ulan' and 'quap'
# generate similar posterior distributions for 'a' and 'b'. However, for 'a', the ulam
# model (purple) placed more probability mass in the upper end of the tail which ends up pushing
# the mean of this posterior distribution further to the right when compared to that of the
# quadratic approximation model.
# The reason behind this is that quadratic approximation is assuming the posterior distribution
# to be Gaussian and as a consequence producing a symmetric distribution with less
# probability mass in the upper tail.
```

b) Change the prior on the variable intercept to dnorm(0 , 10) and estimate the posterior distribution with both ulam() and quap(). Do the differences between the two estimations increase, decrease, or stay the same? Explain your answer.

```
m11.4_3b <- ulam(
alist(
pulled_left ~ dbinom( 1, p ),
logit(p) <- a[actor] + b[treatment] ,
a[actor] ~ dnorm( 0 , 10),
b[treatment] ~ dnorm( 0, 0.5 )
), data = dat_list, chains=4, log_lik =TRUE)
```

```
## Warning in '/var/folders/wx/1_76tj0s15gc4yxmndvw41_00000gn/T/RtmpGmhclW/model-9fc04736fd47.stan', li
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc
## Warning in '/var/folders/wx/1_76tj0s15gc4yxmndvw4l_00000gn/T/RtmpGmhclW/model-9fc04736fd47.stan', lin
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc
## Warning in '/var/folders/wx/1_76tj0s15gc4yxmndvw4l_00000gn/T/RtmpGmhclW/model-9fc04736fd47.stan', lin
##     of arrays by placing brackets after a variable name is deprecated and
##     will be removed in Stan 2.32.0. Instead use the array keyword before the
##     type. This can be changed automatically using the auto-format flag to
##     stanc
```

```
## Running MCMC with 4 sequential chains, with 1 thread(s) per chain...
##
## Chain 1 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 1 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1 finished in 1.5 seconds.
## Chain 2 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 2 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 2 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 2 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 2 finished in 1.5 seconds.
## Chain 3 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 3 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 3 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 3 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 3 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 3 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 3 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 3 finished in 1.5 seconds.
## Chain 4 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 4 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 4 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 4 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 4 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 4 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 4 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 4 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 4 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 4 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 4 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 4 finished in 1.4 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 1.5 seconds.
## Total execution time: 6.3 seconds.
```

```
precis( m11.4_3b , 2 )
```

```
##             mean        sd        5.5%      94.5%    n_eff     Rhat4
## a[1] -0.40954725 0.3647949 -0.99631522  0.1716224 490.1168 1.0007128
## a[2] 11.10273279 5.3721507  4.77601895 21.0216485 666.8335 1.0013403
## a[3] -0.71702286 0.3663233 -1.31856745 -0.1648423 496.6573 1.0002778
## a[4] -0.71952122 0.3691109 -1.31815200 -0.1238644 502.7711 0.9999588
## a[5] -0.41642251 0.3624445 -0.99004221  0.1590387 491.0808 0.9999525
## a[6]  0.54412650 0.3647713 -0.02620972  1.1195497 496.5600 1.0009248
## a[7]  2.13035885 0.4866378  1.38126030  2.9111144 645.5419 1.0009864
## b[1] -0.09125543 0.3169864 -0.58570331  0.4208727 480.1287 0.9990013
## b[2]  0.43709727 0.3174793 -0.05099625  0.9467423 448.0083 0.9994851
## b[3] -0.45108872 0.3073809 -0.93980987  0.0144581 436.7485 1.0017192
## b[4]  0.31875698 0.3034859 -0.14778250  0.8040126 446.2078 1.0018479
```

```
m3b <- quap(
alist(
pulled_left ~ dbinom( 1, p ),
logit(p) <- a[actor] + b[treatment],
a[actor] ~ dnorm( 0 , 10),
b[treatment] ~ dnorm( 0, 0.5 )
) , data=dat_list )
precis(m3b,depth=2)
```

```
##            mean        sd        5.5%        94.5%
## a[1] -0.3519984 0.3477654 -0.90779469  0.203797804
## a[2]  6.9924004 3.5460942  1.32505687 12.659743832
## a[3] -0.6546467 0.3537049 -1.21993540 -0.089357905
## a[4] -0.6546971 0.3537062 -1.21998795 -0.089406259
## a[5] -0.3519976 0.3477654 -0.90779385  0.203798608
## a[6]  0.5812395 0.3522762  0.01823414  1.144244843
## a[7]  2.1186246 0.4523564  1.39567168  2.841577559
## b[1] -0.1418952 0.3011500 -0.62319103  0.339400627
## b[2]  0.3815868 0.3009913 -0.09945547  0.862629094
## b[3] -0.4901542 0.3030962 -0.97456042 -0.005747904
## b[4]  0.2695535 0.3007597 -0.21111856  0.750225653
```

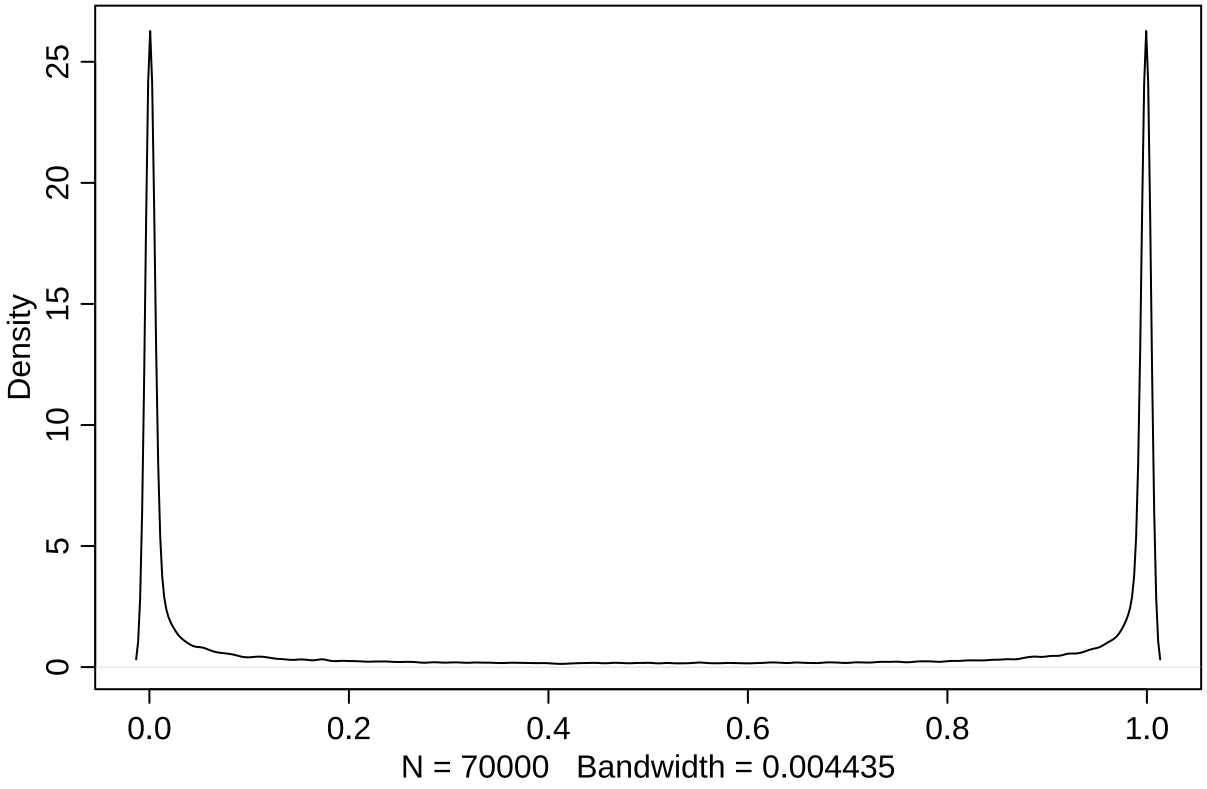
```
# comparing models using PSIS
compare( m11.4_3b, m3b, func=PSIS)
```

```
## Warning in compare(m11.4_3b, m3b, func = PSIS): Not all model fits of same class.
## This is usually a bad idea, because it implies they were fit by different algorithms.
## Check yourself, before you wreck yourself.

## Some Pareto k values are very high (>1). Set pointwise=TRUE to inspect individual points.

##              PSIS       SE    dPSIS      dSE     pPSIS       weight
## m11.4_3b 529.7405 19.77912  0.00000       NA  8.988327 1.000000e+00
## m3b      573.4947 17.58633 43.75427 5.200684 27.342918 3.154145e-10
```

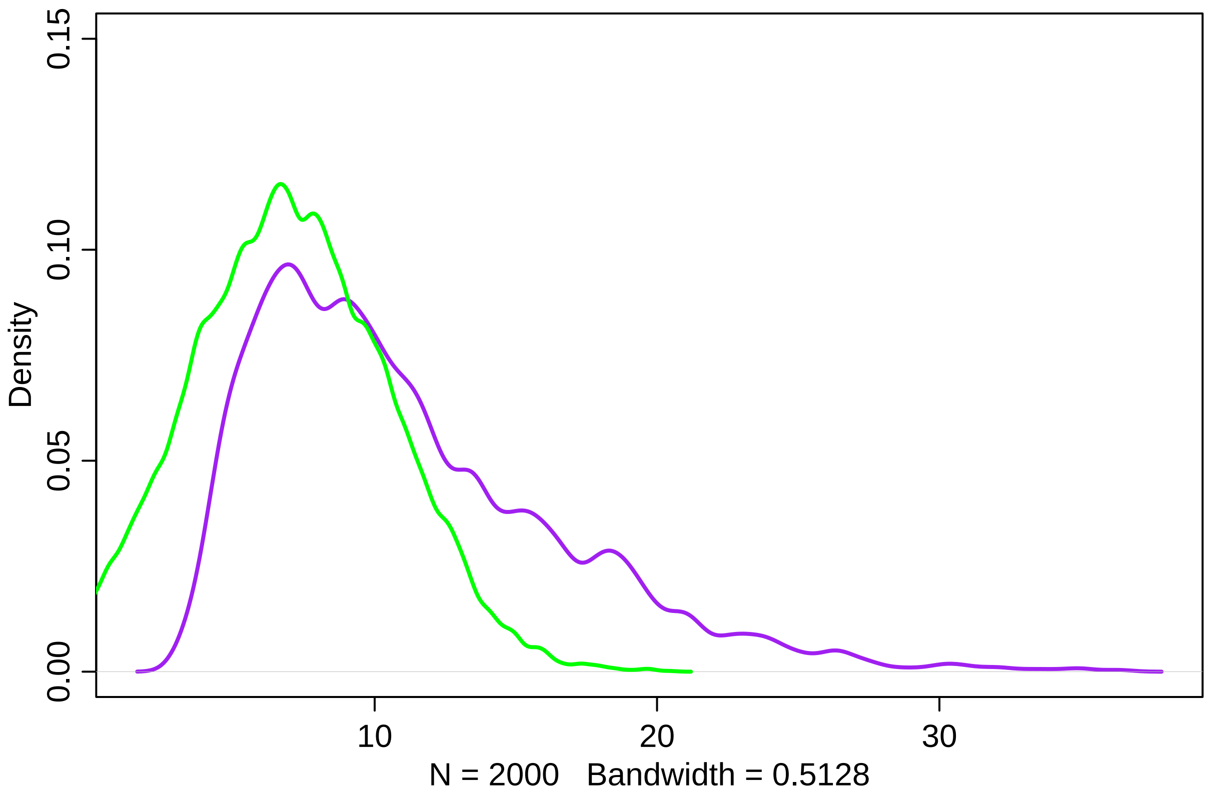
```
# This time, PSIS suggests that both models behave with different precision. The PSIS values of
# both the models are different. It is clear that 'ulan' is a better model for this choice of
# prior intercept.
```

```
# This time, there is NOTICEABLE difference in the a[2] values - there is again a higher
# estimate with the ulam model.
```

```
# visualizing PRIOR of 'a' used here
prior <- extract.prior( m3b , n=1e4 )
p <- inv_logit( prior$a )
dens( p , adj=0.1 )
```



```
# now comparing posterior distributions for 'a' in the two models
post_m11.4_3b <- extract.samples(m11.4_3b)
post_m3b <- extract.samples(m3b)

dens(post_m11.4_3b$a[, 2], lwd = 2, col='purple', ylim = c(0, 0.15))
dens(post_m3b$a[, 2], add=TRUE, lwd = 2, col='green')
```



```
# It is noteworthy that the difference in 'a' values get bigger as we move towards a choice of
# flatter priors.

# A flat Normal(0,10) prior on the intercept produces a very non-flat prior distribution on
# the outcome scale.

# Most of the probability mass is piled up near zero and one. The model thinks, before it sees the
# data, that chimpanzees either never or always pull the left lever. This generates unnecessary
# inference error.
# A flat prior in the logit space is not a flat prior in the outcome probability space.
```