practical_BCM_week17

FWJ Lewin

2025-02-28

```r
library(rstan)
```

```
## Loading required package: StanHeaders
```

```
##
## rstan version 2.32.6 (Stan version 2.32.2)
```

```
## 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)
```

```r
x <- c(1.20, 1.21, 3.06, 7.89, 5.67, 6.10, 3.90)
n <- length(x)

stan_data <- list(
  n = n,
  x = x
)

fit <- stan(file = "~/stan4_yehyeh.stan", data = stan_data,
            iter = 2000, chains = 4)
```

```
## Trying to compile a simple C file
```

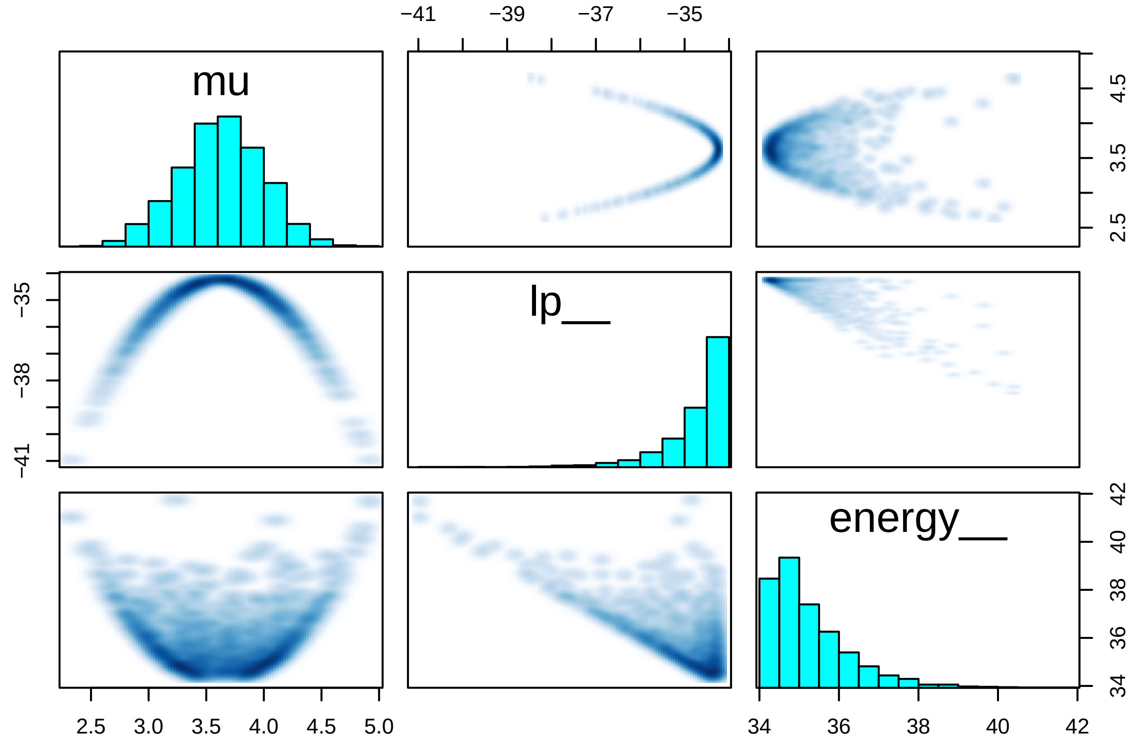
```
## Running /usr/lib/R/bin/R CMD SHLIB foo.c
## gcc -I"/usr/share/R/include" -DNDEBUG   -I"/usr/local/lib/R/site-library/Rcpp/include/"  -I"/usr/loca
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:19,
##                  from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1,
##                  from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp
##                  from <command-line>:
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:679:10: fatal error: cma
##   679 | #include <cmath>
##       |          ^~~~~~~
## compilation terminated.
## make: *** [/usr/lib/R/etc/Makeconf:168: foo.o] Error 1
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 6e-06 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.06 seconds.
## Chain 1: Adjust your expectations accordingly!
```

```
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.006 seconds (Warm-up)
## Chain 1:                0.006 seconds (Sampling)
## Chain 1:                0.012 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 2e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.02 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.006 seconds (Warm-up)
## Chain 2:                0.005 seconds (Sampling)
## Chain 2:                0.011 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 2e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.02 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.005 seconds (Warm-up)
## Chain 3:                0.005 seconds (Sampling)
## Chain 3:                0.01 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 2e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.02 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.005 seconds (Warm-up)
## Chain 4:                0.005 seconds (Sampling)
## Chain 4:                0.01 seconds (Total)
## Chain 4:
```

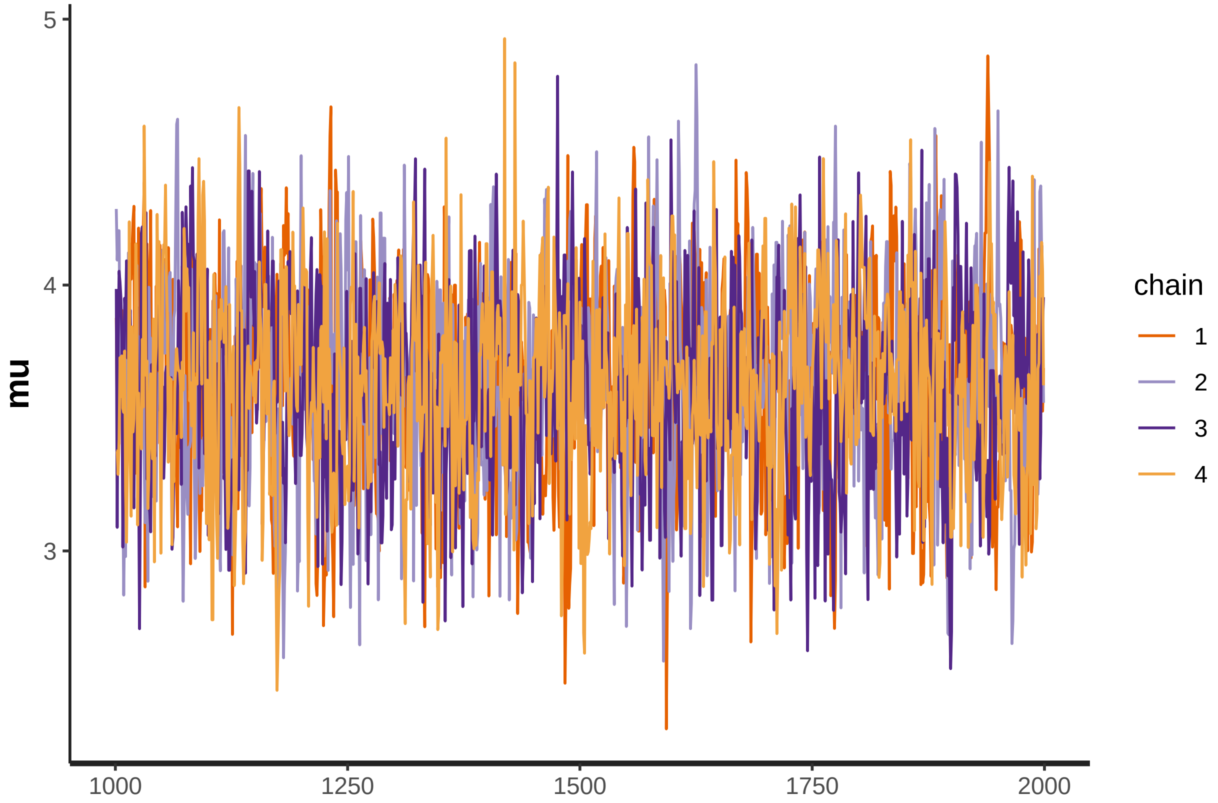
```r
print(fit)
```

```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##        mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## mu     3.63    0.01 0.37   2.90   3.39   3.64   3.88   4.32  1482    1
## lp__ -34.76    0.02 0.74 -36.83 -34.96 -34.47 -34.27 -34.23  1744    1
##
## Samples were drawn using NUTS(diag_e) at Fri Feb 28 16:49:06 2025.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```r
pairs(fit)
```



```r
traceplot(fit)
```



```r
mu_samples <- extract(fit)$mu

post_mean <- sum(x)/(n+1)
post_sd <- sqrt(1/(n+1))

# Compare MCMC results with theoretical values
cat("Theoretical posterior mean:", post_mean, "\n")
```

```
## Theoretical posterior mean: 3.62875
```

```r
cat("MCMC posterior mean:", mean(mu_samples), "\n")
```

```
## MCMC posterior mean: 3.629722
```

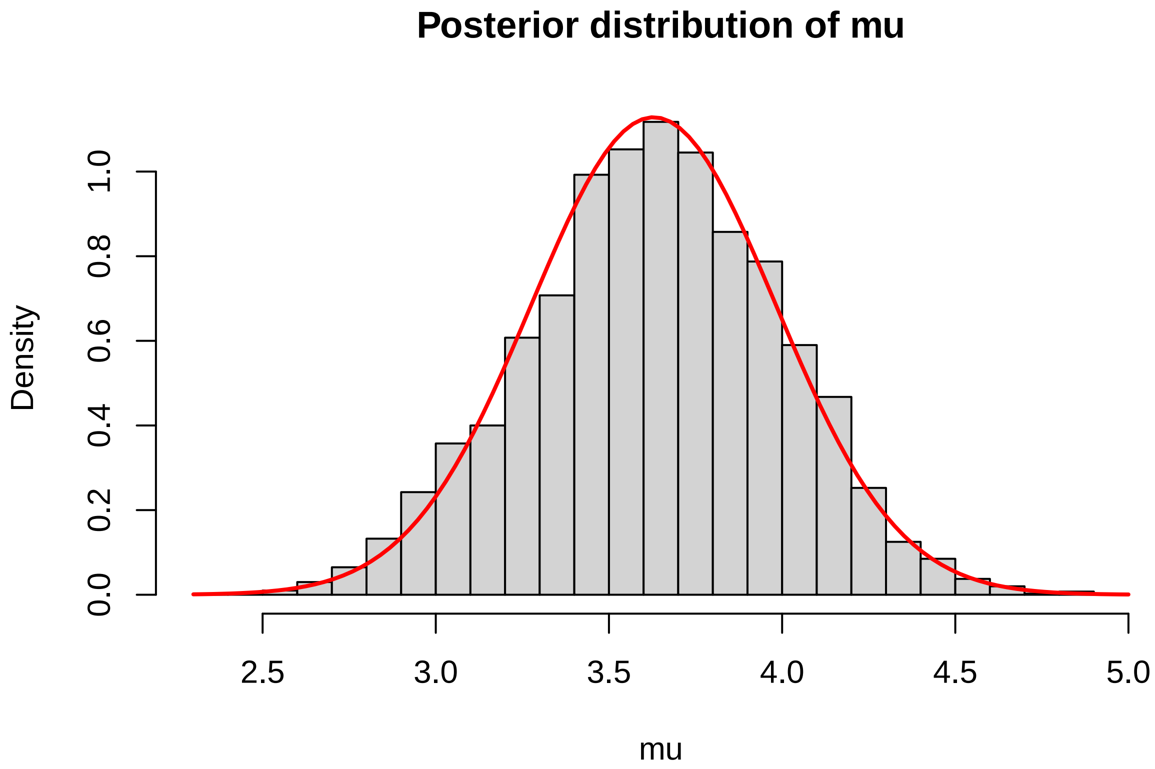
```r
cat("Theoretical posterior SD:", post_sd, "\n")
```

```
## Theoretical posterior SD: 0.3535534
```

```r
cat("MCMC posterior SD:", sd(mu_samples), "\n")
```

```
## MCMC posterior SD: 0.3654664
```

```r
# Plot histogram of samples with theoretical density overlay
hist(mu_samples, freq = FALSE, main = "Posterior distribution of mu",
     xlab = "mu", breaks = 30)
curve(dnorm(x, post_mean, post_sd), add = TRUE, col = "red", lwd = 2)
```



now for a stranger prior dist

```
our_data <- list(n = 20,
                 p = c(0.25057883, 0.16862872, 0.11989827, 0.28149519, 0.20427907,
                       0.16859187, 0.40253736, 0.09341611, 0.14762340, 0.14047014,
                       0.29998209, 0.19349593, 0.21179227, 0.24900885, 0.32570937,
                       0.12341203, 0.20488021, 0.33726469, 0.08214418, 0.41775598))
fit2 <- stan(file = "~/stan5_spawnpoint.stan", data = our_data,
            iter = 4000, warmup = 1000, chains = 4)
```

```
## Warning in readLines(file, warn = TRUE): incomplete final line found on
## '/home/rstudio/stan5_spawnpoint.stan'
```

```
## Trying to compile a simple C file
```

```
## Running /usr/lib/R/bin/R CMD SHLIB foo.c
## gcc -I"/usr/share/R/include" -DNDEBUG   -I"/usr/local/lib/R/site-library/Rcpp/include/"  -I"/usr/loca
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:19,
##                  from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1,
##                  from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp
##                  from <command-line>:
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:679:10: fatal error: cma
##   679 | #include <cmath>
##       |          ^~~~~~~
## compilation terminated.
## make: *** [/usr/lib/R/etc/Makeconf:168: foo.o] Error 1
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
```

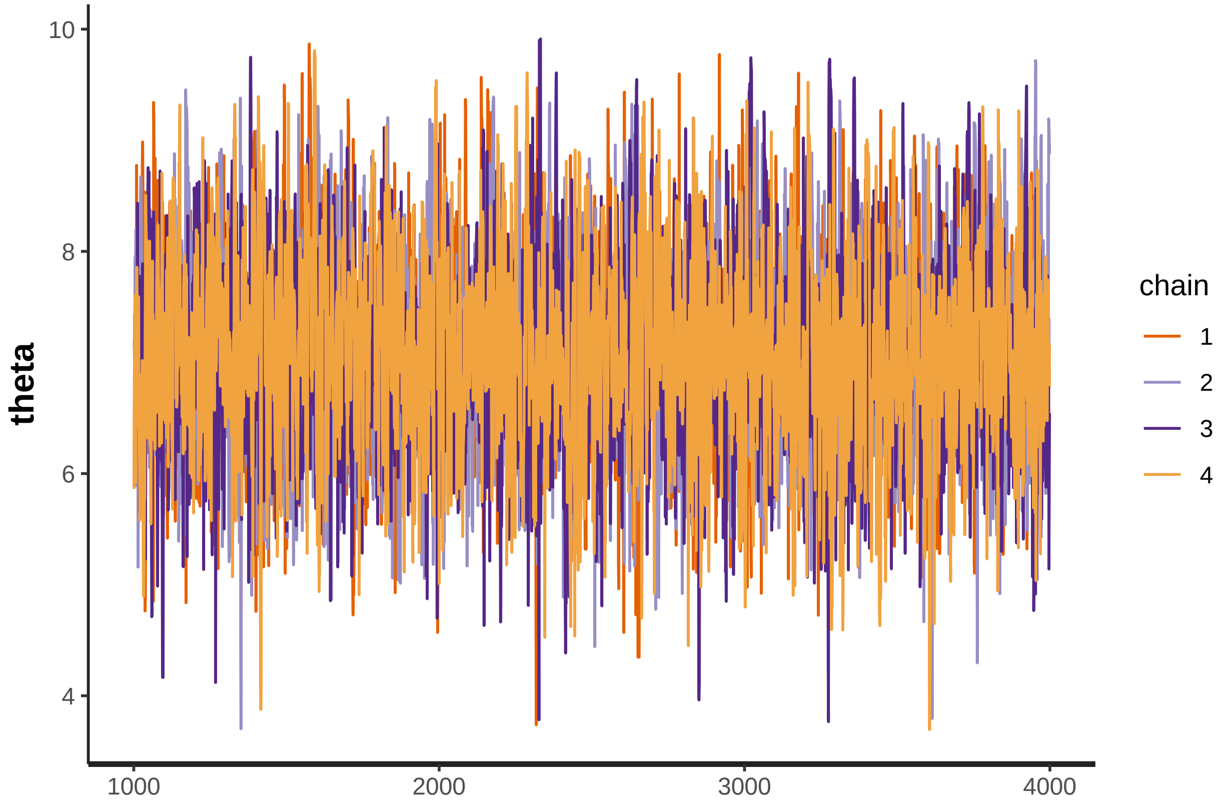
```
## Chain 1:
## Chain 1: Gradient evaluation took 6e-06 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.06 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 1: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 1: Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1001 / 4000 [ 25%]  (Sampling)
## Chain 1: Iteration: 1400 / 4000 [ 35%]  (Sampling)
## Chain 1: Iteration: 1800 / 4000 [ 45%]  (Sampling)
## Chain 1: Iteration: 2200 / 4000 [ 55%]  (Sampling)
## Chain 1: Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 1: Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 1: Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 1: Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 1: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.007 seconds (Warm-up)
## Chain 1:                0.018 seconds (Sampling)
## Chain 1:                0.025 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 4e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.04 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 2: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 2: Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 2: Iteration: 1001 / 4000 [ 25%]  (Sampling)
## Chain 2: Iteration: 1400 / 4000 [ 35%]  (Sampling)
## Chain 2: Iteration: 1800 / 4000 [ 45%]  (Sampling)
## Chain 2: Iteration: 2200 / 4000 [ 55%]  (Sampling)
## Chain 2: Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 2: Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 2: Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 2: Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 2: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.007 seconds (Warm-up)
## Chain 2:                0.018 seconds (Sampling)
## Chain 2:                0.025 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 3e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.03 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 3: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 3: Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 3: Iteration: 1001 / 4000 [ 25%]  (Sampling)
## Chain 3: Iteration: 1400 / 4000 [ 35%]  (Sampling)
## Chain 3: Iteration: 1800 / 4000 [ 45%]  (Sampling)
## Chain 3: Iteration: 2200 / 4000 [ 55%]  (Sampling)
## Chain 3: Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 3: Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 3: Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 3: Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 3: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.006 seconds (Warm-up)
## Chain 3:                0.02 seconds (Sampling)
## Chain 3:                0.026 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 3e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.03 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 4: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 4: Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 4: Iteration: 1001 / 4000 [ 25%]  (Sampling)
## Chain 4: Iteration: 1400 / 4000 [ 35%]  (Sampling)
## Chain 4: Iteration: 1800 / 4000 [ 45%]  (Sampling)
## Chain 4: Iteration: 2200 / 4000 [ 55%]  (Sampling)
## Chain 4: Iteration: 2600 / 4000 [ 65%]  (Sampling)
## Chain 4: Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 4: Iteration: 3400 / 4000 [ 85%]  (Sampling)
## Chain 4: Iteration: 3800 / 4000 [ 95%]  (Sampling)
## Chain 4: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.007 seconds (Warm-up)
## Chain 4:                0.023 seconds (Sampling)
## Chain 4:                0.03 seconds (Total)
## Chain 4:
```

```r
# Check diagnostics
print(fit2)
```

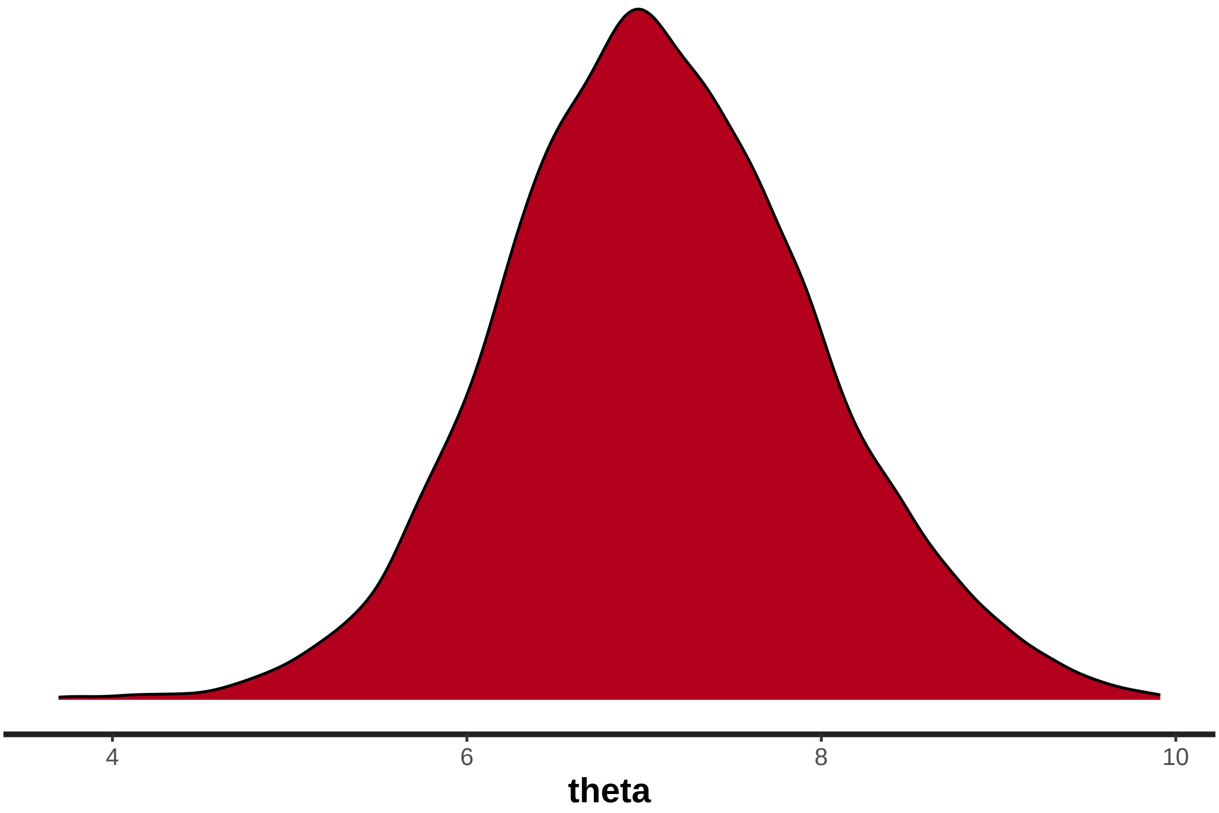
```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=4000; warmup=1000; thin=1;
## post-warmup draws per chain=3000, total post-warmup draws=12000.
##
##        mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## theta  7.09    0.01 0.89  5.39  6.50  7.06  7.67  8.91  3795    1
## lp__  16.99    0.02 0.87 14.40 16.83 17.33 17.53 17.59  2857    1
```

```
##
## Samples were drawn using NUTS(diag_e) at Fri Feb 28 16:50:02 2025.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
stan_trace(fit2)
```

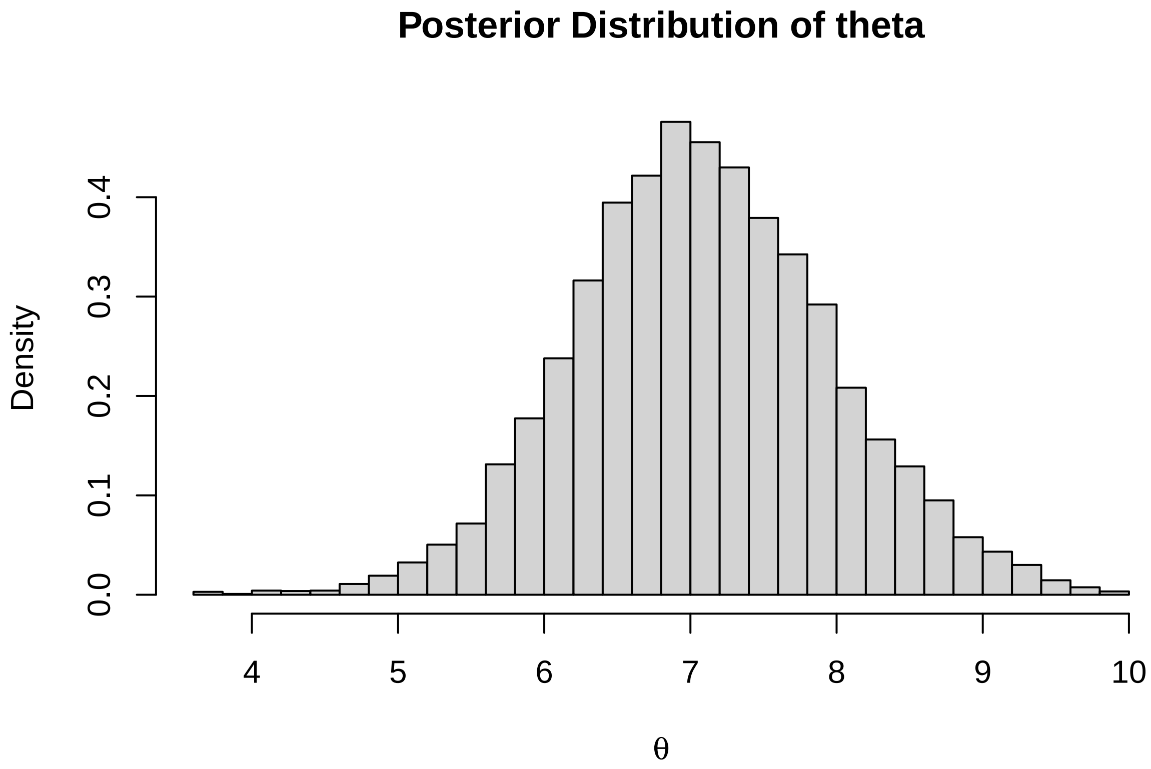


```
stan_dens(fit2)
```



```r
# Extract samples
theta_samples <- extract(fit2)$theta

# Plot histogram of posterior samples
hist(theta_samples, freq = FALSE, breaks = 30,
     main = "Posterior Distribution of theta",
     xlab = expression(theta))
```



```r
# Calculate posterior mean and 95% credible interval
post_mean <- mean(theta_samples)
post_ci <- quantile(theta_samples, c(0.025, 0.975))
cat("Posterior mean:", post_mean, "\n")
```

```
## Posterior mean: 7.090276
```

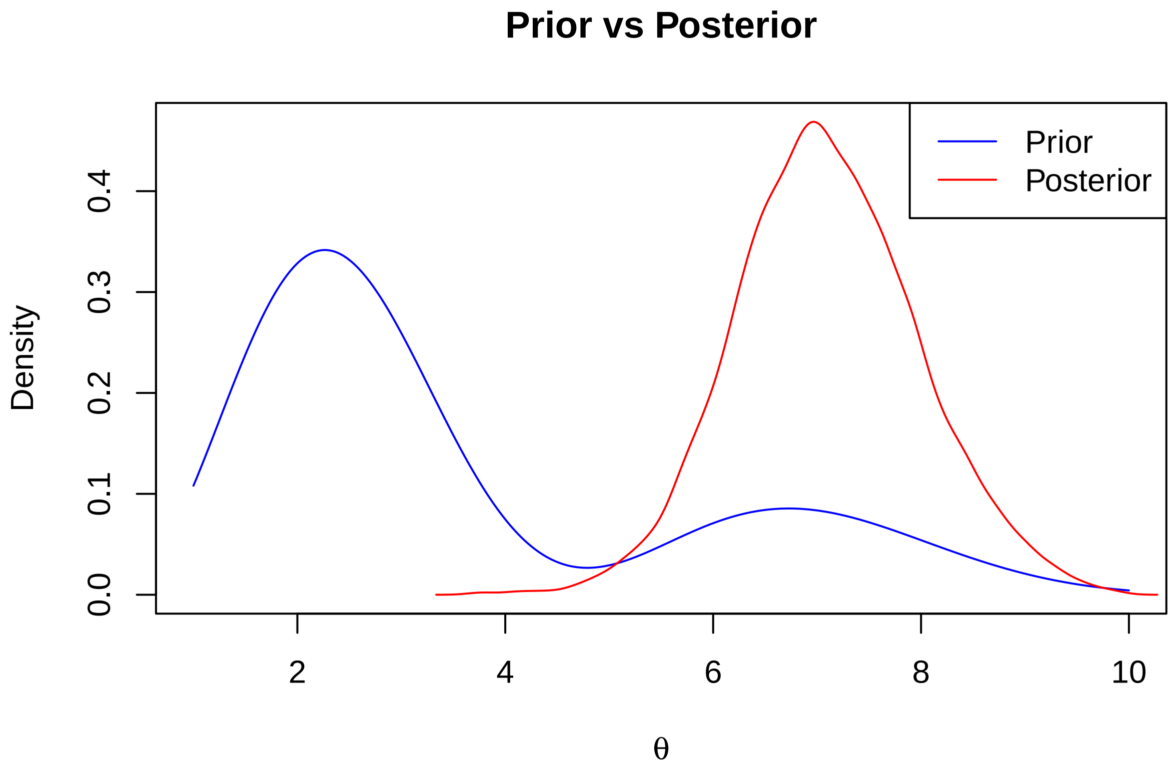
```r
cat("95% credible interval:", post_ci[1], "-", post_ci[2], "\n")
```

```
## 95% credible interval: 5.392537 - 8.906712
```

```r
# Plot prior and posterior together for comparison
theta_grid <- seq(1, 10, by = 0.01)
prior_unnorm <- sapply(theta_grid, function(t) {
  t^3 * exp(-t) * (sin(t) + 1.2)
})
prior_density <- prior_unnorm / sum(prior_unnorm * 0.01)

posterior_density <- density(theta_samples)

plot(theta_grid, prior_density, type = "l", col = "blue",
     xlab = expression(theta), ylab = "Density",
     main = "Prior vs Posterior", ylim = c(0, max(posterior_density$y)))
lines(posterior_density, col = "red")
legend("topright", legend = c("Prior", "Posterior"),
       col = c("blue", "red"), lty = 1)
```



```r
# Calculate mean of Beta(2, theta) for interpretation
expected_p <- 2 / (2 + post_mean)
cat("Expected value of Beta(2, theta) at posterior mean:", expected_p, "\n")
```

```
## Expected value of Beta(2, theta) at posterior mean: 0.2200153
```

hierarchical model now, with a ~ beta(2,1), b|a ~exp(a), lambda|b~exp(b), X|lambda ~ poisson(lambda)

```r
hie_data = list(N = 40,
                K = 2,
                J = 2,
                group = c(rep(1, 30),
                          rep(2, 10)),
                subgroup = c(sort(rep(1:2, 15)),
                             sort(rep(1:2, 5))),
                X = c(3, 2, 2, 2, 1,
                      2, 3, 4, 0, 0,
                      2, 5, 4, 4, 1,
                      3, 3, 3, 4, 4,
                      2, 5, 3, 4, 4,
                      1, 1, 3, 3, 5,
                      1, 0, 1, 1, 0,
                      1, 0, 1, 2, 0))
fit3 <- stan(file = "~/stan6_hierarchical.stan", data = hie_data,
            iter = 5000, warmup = 1000, chains = 4, cores = 4)
```

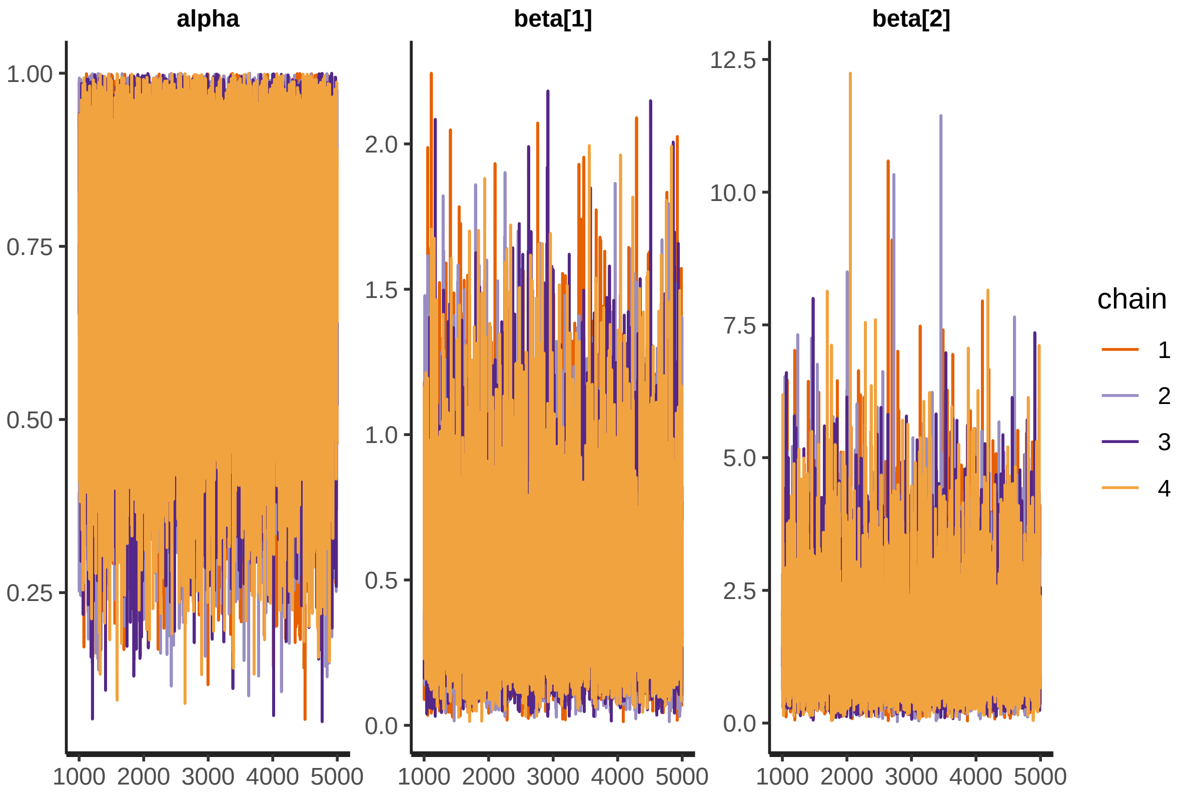
```
## Trying to compile a simple C file
## Running /usr/lib/R/bin/R CMD SHLIB foo.c
## gcc -I"/usr/share/R/include" -DNDEBUG   -I"/usr/local/lib/R/site-library/Rcpp/include/"  -I"/usr/loca
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:19,
##                  from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1,
##                  from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp
##                  from <command-line>:
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:679:10: fatal error: cma
##   679 | #include <cmath>
##       |          ^~~~~~~
## compilation terminated.
## make: *** [/usr/lib/R/etc/Makeconf:168: foo.o] Error 1
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 3e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.3 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 2.6e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.26 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 5000 [  0%]  (Warmup)
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 2.9e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.29 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 5000 [  0%]  (Warmup)
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 2.6e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.26 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 4: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 2: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 3: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 4: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 4: Iteration: 1001 / 5000 [ 20%]  (Sampling)
## Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
```

```
## Chain 1: Iteration: 1001 / 5000 [ 20%]  (Sampling)
## Chain 4: Iteration: 1500 / 5000 [ 30%]  (Sampling)
## Chain 2: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 2: Iteration: 1001 / 5000 [ 20%]  (Sampling)
## Chain 3: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 3: Iteration: 1001 / 5000 [ 20%]  (Sampling)
## Chain 1: Iteration: 1500 / 5000 [ 30%]  (Sampling)
## Chain 4: Iteration: 2000 / 5000 [ 40%]  (Sampling)
## Chain 2: Iteration: 1500 / 5000 [ 30%]  (Sampling)
## Chain 3: Iteration: 1500 / 5000 [ 30%]  (Sampling)
## Chain 1: Iteration: 2000 / 5000 [ 40%]  (Sampling)
## Chain 4: Iteration: 2500 / 5000 [ 50%]  (Sampling)
## Chain 2: Iteration: 2000 / 5000 [ 40%]  (Sampling)
## Chain 3: Iteration: 2000 / 5000 [ 40%]  (Sampling)
## Chain 1: Iteration: 2500 / 5000 [ 50%]  (Sampling)
## Chain 3: Iteration: 2500 / 5000 [ 50%]  (Sampling)
## Chain 4: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 2: Iteration: 2500 / 5000 [ 50%]  (Sampling)
## Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 3: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 4: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 2: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 3: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 4: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 2: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 3: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 4: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 2: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 4: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.114 seconds (Warm-up)
## Chain 4:                0.483 seconds (Sampling)
## Chain 4:                0.597 seconds (Total)
## Chain 4:
## Chain 3: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 2: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.162 seconds (Warm-up)
## Chain 1:                0.511 seconds (Sampling)
## Chain 1:                0.673 seconds (Total)
## Chain 1:
## Chain 3: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.157 seconds (Warm-up)
## Chain 3:                0.482 seconds (Sampling)
## Chain 3:                0.639 seconds (Total)
## Chain 3:
## Chain 2: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.162 seconds (Warm-up)
## Chain 2:                0.523 seconds (Sampling)
## Chain 2:                0.685 seconds (Total)
## Chain 2:
```

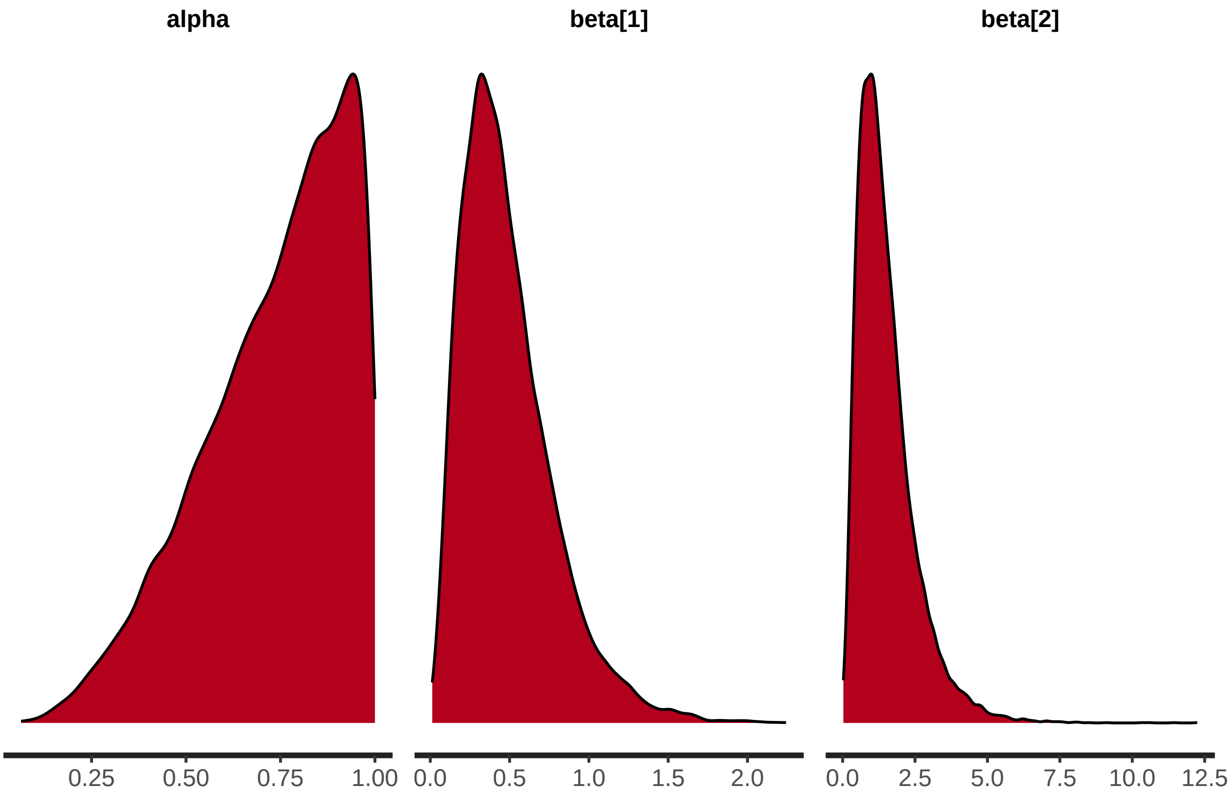
```
print(fit3, pars = c("alpha", "beta"))
```

```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=5000; warmup=1000; thin=1;
## post-warmup draws per chain=4000, total post-warmup draws=16000.
##
##         mean se_mean   sd 2.5%  25%  50%  75% 97.5% n_eff Rhat
## alpha   0.74    0.00 0.19 0.30 0.61 0.78 0.90  0.99 25258    1
## beta[1] 0.49    0.00 0.29 0.10 0.28 0.43 0.64  1.21 23269    1
## beta[2] 1.50    0.01 1.00 0.27 0.79 1.27 1.94  4.08 17853    1
##
## Samples were drawn using NUTS(diag_e) at Fri Feb 28 16:51:06 2025.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
traceplot(fit3, pars = c("alpha", "beta"))
```

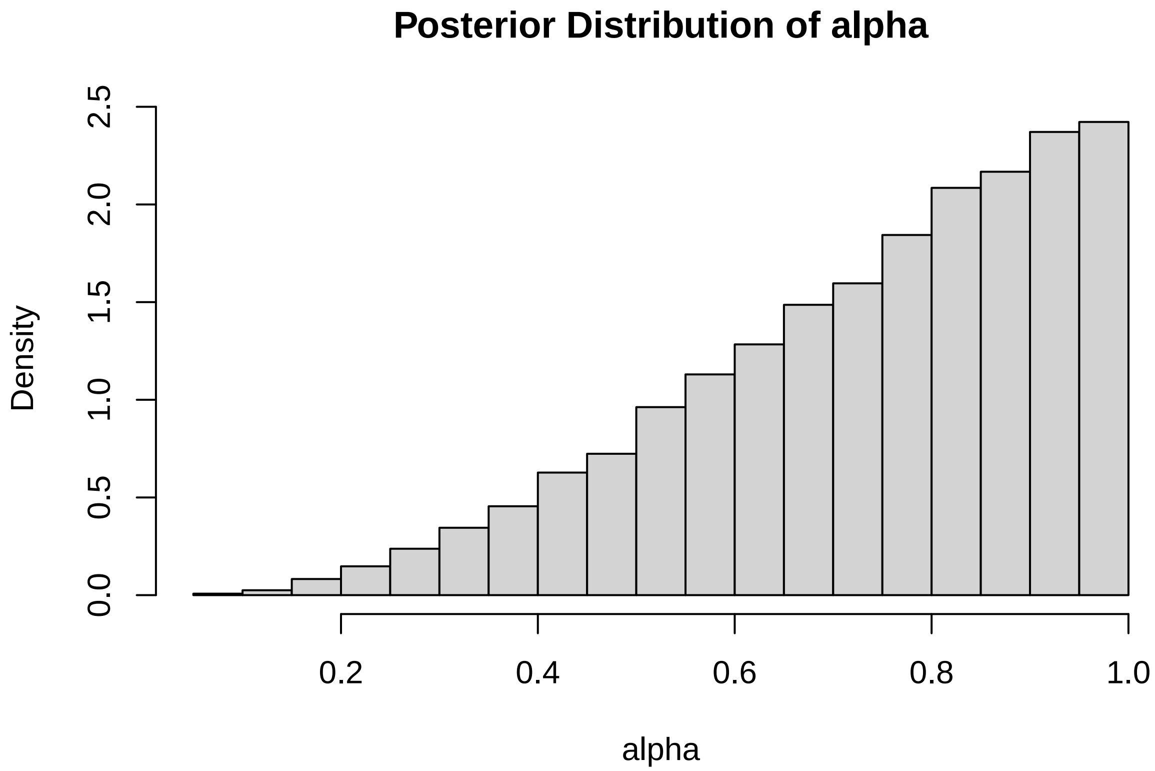


```
stan_dens(fit3, pars = c("alpha", "beta"))
```



```r
# Extract posterior samples
posterior <- extract(fit3)

# Investigate alpha
alpha_samples <- posterior$alpha
hist(alpha_samples, breaks = 30, main = "Posterior Distribution of alpha",
     xlab = "alpha", freq = FALSE)
```



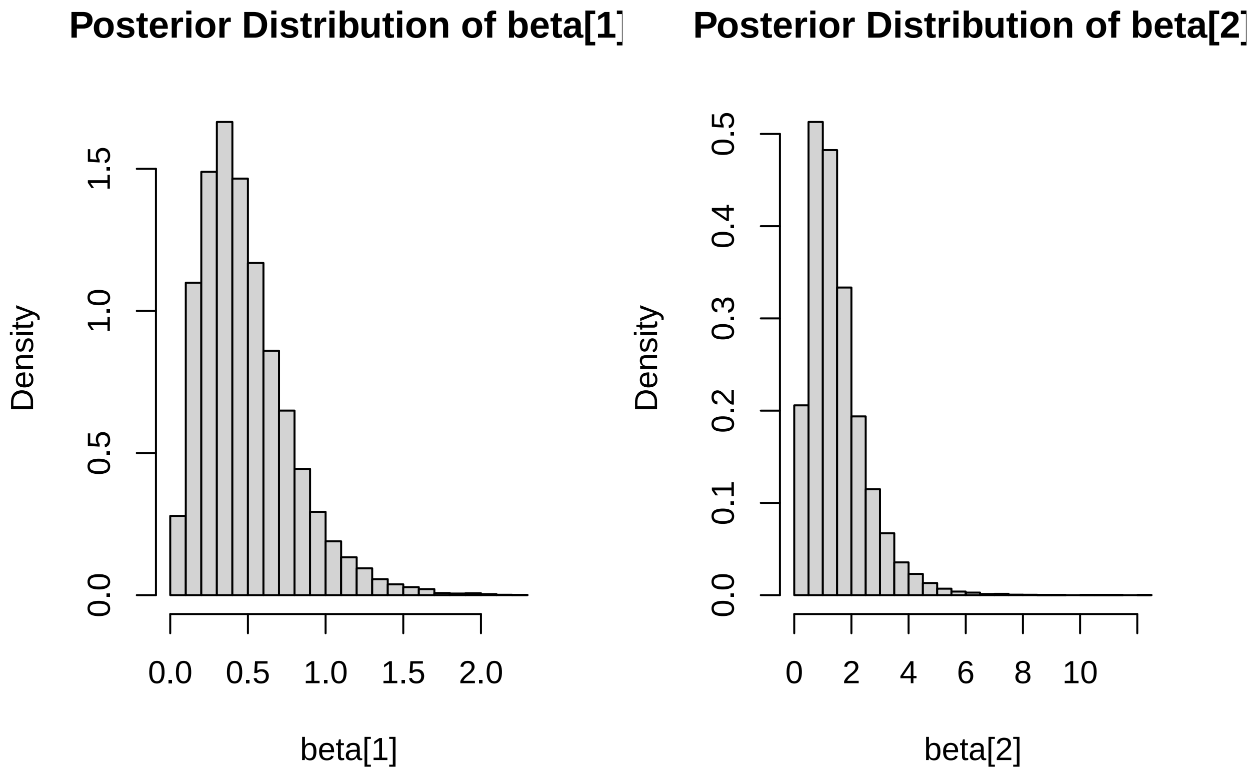
```
cat("Posterior mean of alpha:", mean(alpha_samples), "\n")
```

```
## Posterior mean of alpha: 0.7369387
```

```
cat("95% credible interval for alpha:", quantile(alpha_samples, c(0.025, 0.975)), "\n")
```

```
## 95% credible interval for alpha: 0.3001545 0.9888346
```

```
# Investigate beta
beta_samples <- posterior$beta
par(mfrow = c(1, 2))
for(k in 1:hie_data$K) {
  hist(beta_samples[,k], breaks = 30,
       main = paste("Posterior Distribution of beta[", k, "]", sep = ""),
       xlab = paste("beta[", k, "]", sep = ""), freq = FALSE)
}
```



```
par(mfrow = c(1, 1))

# Compare beta values
cat("Posterior mean of beta[1]:", mean(beta_samples[,1]), "\n")
```

```
## Posterior mean of beta[1]: 0.4867872
```

```
cat("Posterior mean of beta[2]:", mean(beta_samples[,2]), "\n")
```

```
## Posterior mean of beta[2]: 1.497617
```

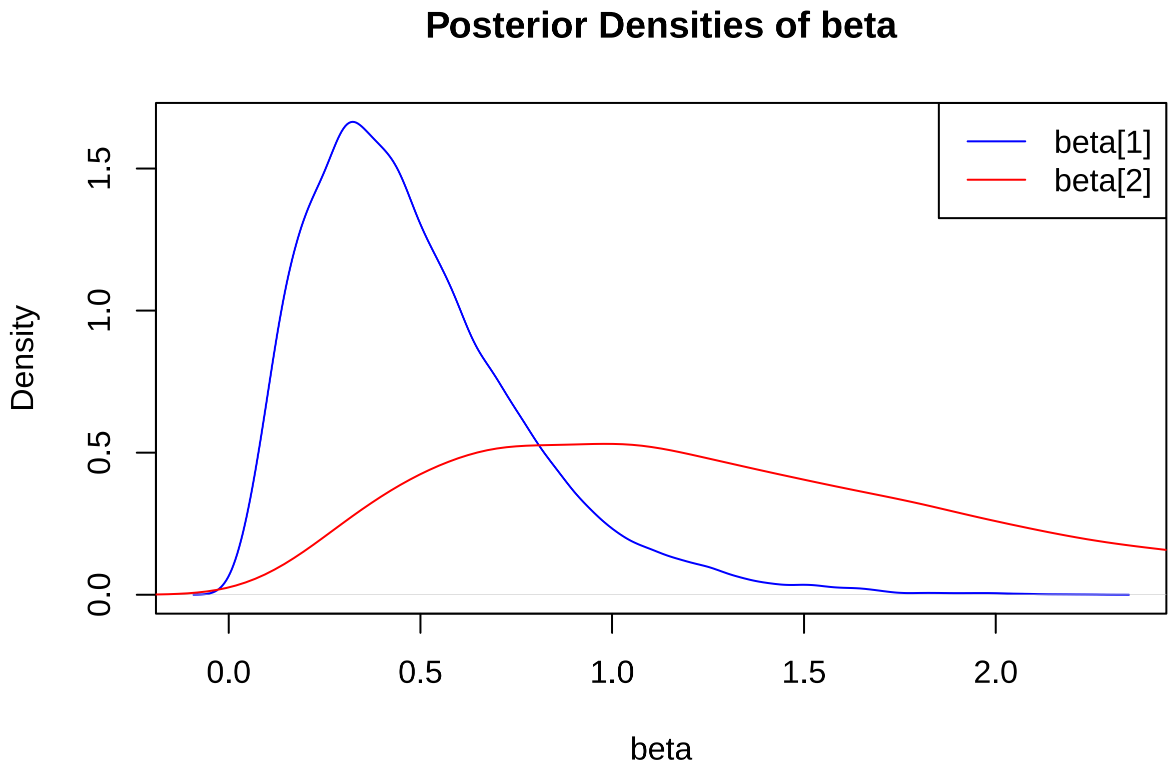
```
cat("95% CI for beta[1]:", quantile(beta_samples[,1], c(0.025, 0.975)), "\n")
```

```
## 95% CI for beta[1]: 0.09566209 1.212193
```

```
cat("95% CI for beta[2]:", quantile(beta_samples[,2], c(0.025, 0.975)), "\n")
```

```
## 95% CI for beta[2]: 0.273848 4.08386
```

```
# Compare the distributions
plot(density(beta_samples[,1]), col = "blue", main = "Posterior Densities of beta",
     xlab = "beta", ylim = c(0, max(c(density(beta_samples[,1])$y,
                                      density(beta_samples[,2])$y))))
lines(density(beta_samples[,2]), col = "red")
legend("topright", legend = c("beta[1]", "beta[2]"),
       col = c("blue", "red"), lty = 1)
```



```
# Look at lambda values
lambda_samples <- posterior$lambda
lambda_means <- matrix(0, nrow = hie_data$J, ncol = hie_data$K)
for(j in 1:hie_data$J) {
  for(k in 1:hie_data$K) {
    lambda_means[j,k] <- mean(lambda_samples[,j,k])
  }
}
print("Posterior means of lambda:")
```

```
## [1] "Posterior means of lambda:"
```

```
print(lambda_means)
```

```
##          [,1]      [,2]
## [1,] 2.324489 0.6284100
## [2,] 3.166393 0.7881051
```