BDA - Assignment 6

Anonymous

24 10 2019

Contents

Used libraries:

```
library(dplyr)
library(ggplot2)
library(rstan)
library(gdata)
library(bayesplot)
library(aaltobda)
data("bioassay")
```

```
options(mc.cores = parallel::detectCores())
rstan_options(auto_write = TRUE)
```

Exercise 1

Write down the model for the bioassay data in Stan syntax.

Using Gaussian prior:

$$\begin{pmatrix}\alpha\\ \beta\end{pmatrix} \sim N(\mu_0, \Sigma_0)$$

, where

$$\mu_0 = \begin{pmatrix}0\\ 10\end{pmatrix} \Sigma_0 = \begin{pmatrix}2^2 & 10\\ 10 & 10^2\end{pmatrix}$$

I will write the **stan** code to file, do the compilation and fitting from rstudio directly. Here's the stan code (creates stan_model1.stan in the working directory):

```
write("// I'm writing STAN model from rstudio

      data {
        int N[4]; //sample sizes
        int y[4];  // deaths
        vector[4] x; // dozage
        vector[2] mu; // mu_alpha, mu_beta
        matrix[2,2] sigma; // variance-covariance matrix
      }
```

```
      parameters {
        vector[2] theta; // sampled alpha, beta
      }

      model {
        theta ~ multi_normal(mu, sigma);
        y ~ binomial_logit(N, theta[1] + theta[2]*x);
      }

      generated quantities {
      }

      ",

      "stan_model1.stan"

      )

#compiles the code
stanc("stan_model1.stan")
```

NOTE: In the data block, it would be *in general* better to parameterize the number of groups (4 here) and give it as an argument with stan -call. But since I will only use this once, with set values, this works as well.

Here's the data:

```
mu <- c(0,10)
s <- matrix(c(4,10,10,100),nrow = 2)

stan_data <- list(mu=mu, sigma=s, N=bioassay$n, x=bioassay$x, y=bioassay$y)

stan_model1 <- "./stan_model1.stan"
```

And then the run; 1200 iterations per chain, 6 chains, i.e., 6000 'proper' samples (not necessarily effective sample size though):

```
fit <- stan(file = stan_model1, data = stan_data, warmup = 200, iter = 1200, chains = 6, init = 'random

# I'll extract the posterior for Rhat and plotting
posterior <- extract(fit)
```

NOTE: The results of simulation are handled in the following sections (convergence and plots).

Exercise 2

Use Rhat for convergence analysis.

Rhat (potential scale reduction factor) for six chains, length 1000 (200 warmup excluded). Spesifically, I'll use the rstan's Rhat, which splits the chains instead of just comparing within chain variances with between variance (and rank-normalizes as well). In the rstan package documentation it is recommended not to use samples if Rhat is not less than 1.05.

```
# Extract( ) has already separated the warmup from the sample
alpha <- posterior$theta[,1]
beta <- posterior$theta[,2]
```

```
Rhat_alpha <- Rhat(cbind(alpha[1:1000],alpha[1001:2000],alpha[2001:3000],
            alpha[3001:4000],alpha[4001:5000],alpha[5001:6000]))

Rhat_beta <- Rhat(cbind(beta[1:1000],beta[1001:2000],beta[2001:3000],
           beta[3001:4000],beta[4001:5000],beta[5001:6000]))
```

Results: Rhat for alpha is `1.002`, and for beta `1.002`. It seems that the chains have converged (i.e., good samples were produced), since the within variances agree with the between variance. Let's also check the results with overloaded print function, since it also shows effective sample sizes:

```
print(fit)
```

```
## Inference for Stan model: stan_model1.
## 6 chains, each with iter=1200; warmup=200; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=6000.
##
##           mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## theta[1]  0.96    0.02 0.92 -0.74  0.32  0.90  1.55  2.94  1639    1
## theta[2] 10.54    0.13 4.73  3.41  7.03  9.86 13.20 21.67  1409    1
## lp__     -7.14    0.03 1.08 -9.98 -7.54 -6.79 -6.37 -6.11  1818    1
##
## Samples were drawn using NUTS(diag_e) at Fri Nov 01 21:47:05 2019.
## 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).
```

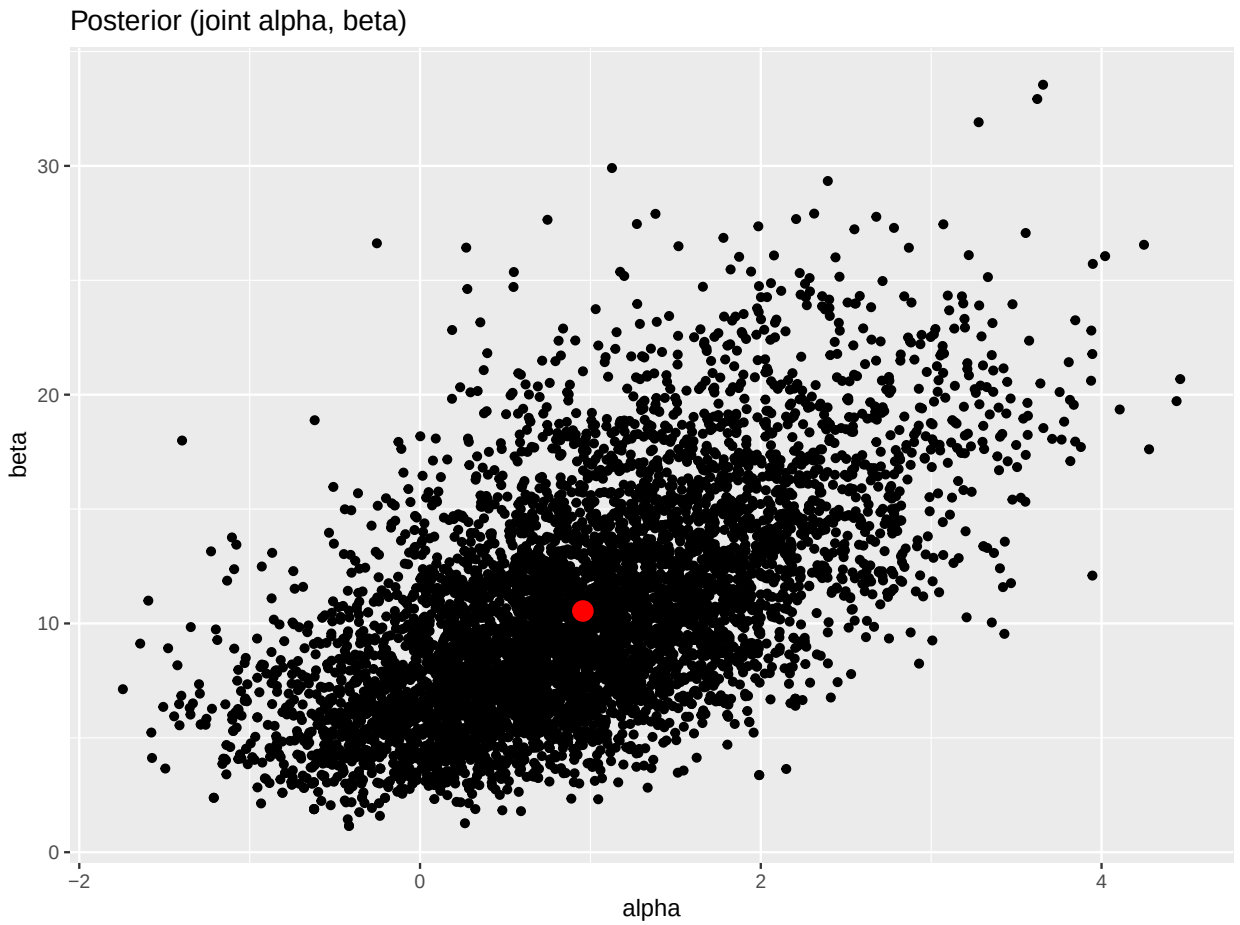
NOTE: theta[1]=alpha, theta[2]=beta.

Exercise 3

Plot the draws for alpha and beta (scatter plot) and include this plot in your report.

So, a simple scatter plot:

```
ggplot() +
  geom_point(aes(x=alpha, y=beta)) +
  geom_point(aes(x=mean(alpha), y=mean(beta)),color='red',size=4) +
  ggtitle('Posterior (joint alpha, beta)')
```



Seems reasonable. These do agree with the earlier results (also last weeks Metropolis had, for posterior, mean(alpha) ~ 1 and mean(beta) ~ 10).