BSDA: Assignment 6

Anonymous student

Contents

General Information to include

- **Time used for reading and self-study exercises**: ~ 9 hours.
- **Time used for the assignment**: ~8 hours
- **Good with assignment**: Redoing assignment 5 in Stan, it made a little clear the syntax and how to actually use Stan.
- **Things to improve in the assignment**: It was really difficult at the beginning to get the syntax right and get everything working. Once I started reading the syntax on binomial_logit and multi_normal it was a little easier to know which types of variables both functions took as input and then I was able to define the data. I would really appreciate some discussion about all the outputs we can get using Stan and their interpretation. I also had issues using RMarkdown and Stan, I have a mac aarch64. I used this solution to fix the issues: https://discourse.mc-stan.org/t/brms-inside-rmarkdown-r-cmd-shlib-foo-c-fails/17671 (installing `install.packages("StanHeaders")`). I was getting an error that said something like this:

```
> Creating a 'stanmodel' object model Running
> /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
> Trying to compile a simple C file rmarkdown

>Trying to compile a simple C file using C compiler: ‘Apple clang version 14.0.0
```

1. Generalized linear model: Bioassay with Stan

Replicate the computations for the bioassay example of section 3.7 (BDA3) using Stan.

1. Write down the model for the bioassay data in Stan syntax. For instructions in reporting your implementation, you can refer to parts 2 c) - g) in Assignment 5. More information on the bioassay data can be found in Section 3.7 of the course book. To get access to data, use the following code:

```r
library(bsda)
library(rjson)
library(rstan)
library(lubridate)
library(ggplot2)
library(tidyverse)
library(knitr)
library(kableExtra)
library(posterior)
library(bayesplot)
data("bioassay")
```

Use the Gaussian prior

$$\begin{bmatrix}\alpha\\ \beta\end{bmatrix} \sim \mathrm{N}\left(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0\right), \qquad \text{where} \quad \boldsymbol{\mu}_0 = \begin{bmatrix}0\\ 10\end{bmatrix} \quad \text{and} \quad \boldsymbol{\Sigma}_0 = \begin{bmatrix}2^2 & 12\\ 12 & 10^2\end{bmatrix}.$$

I will first show the model in stan syntax. This model is saved in `assignment6.stan` which is used later in `stan()` function.

```
data {
  // Biossay Data from BDA3
  int<lower=0> N;
  int<lower=0> y[N];
  vector[N] x;
  int<lower=0> n[N];

  // prior data
  vector[2] mu;
  matrix<lower=0>[2, 2] Sigma;
}


parameters {
  // alpha and beta
  vector[2] beta;
}


transformed parameters {
  vector[N] theta = beta[1] + beta[2] * x;
}

model {
  y ~ binomial_logit(n, theta);
  beta ~ multi_normal(mu, Sigma);
}
```

Now using `stan()` we can sample from the posterior.

```
options(mc.cores = parallel::detectCores())

fit1 <- stan(file = 'assignment6.stan', data = list(
  N = nrow(bioassay),
  y = bioassay$y,
  x = bioassay$x,
  n = bioassay$n,
  mu = c(0, 10),
  Sigma = matrix(c(4, 12, 12, 100), nrow = 2)
), chains=8, seed = 123)

fit1
```

```
## Inference for Stan model: anon_model.
## 8 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=8000.
##
##            mean se_mean   sd   2.5%    25%   50%   75% 97.5% n_eff Rhat
## beta[1]    0.96    0.02 0.90  -0.65   0.33  0.92  1.54  2.84  2198    1
## beta[2]   10.53    0.10 4.74   3.52   7.00  9.81 13.29 21.64  2078    1
## theta[1]  -8.09    0.07 3.59 -16.65 -10.14 -7.51 -5.48 -2.79  2365    1
## theta[2]  -2.20    0.02 1.12  -4.78  -2.81 -2.06 -1.42 -0.37  3804    1
## theta[3]   0.43    0.01 0.78  -1.02  -0.10  0.41  0.93  2.04  2687    1
## theta[4]   8.64    0.09 4.07   2.51   5.57  8.12 11.06 17.98  1930    1
## lp__      -7.14    0.02 1.05  -9.90  -7.50 -6.80 -6.41 -6.15  2234    1
##
## Samples were drawn using NUTS(diag_e) at Sun Oct  8 00:32:26 2023.
## 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).
```

Now I will report my implementation, based on parts 2 c) - g) in Assignment 5:

- The initial points of the MCMC chains are generated randomly by default in Stan. According to the documentation the default is to randomly generate initial values between -2 and 2 on the unconstrained support. The specific initial points randomly chosen for this particular assignment were:

```r
inits <- get_inits(fit1)
starting_points <- data.frame(alpha = numeric(0), beta = numeric(0))

for (i in 1:8) {
  alpha <- inits[[i]]$beta[1]
  beta <- inits[[i]]$beta[2]

  starting_points <- rbind(starting_points, data.frame(alpha = alpha, beta = beta))
}

knitr::kable(starting_points, format = "latex",
             booktabs = TRUE, align = "c",linesep = "",
             col.names = c("Alpha",
                           "Beta"),
             caption = "Intial points for each chain") %>%
  kable_styling(latex_options = "HOLD_position")
```

Table 1: Intial points for each chain

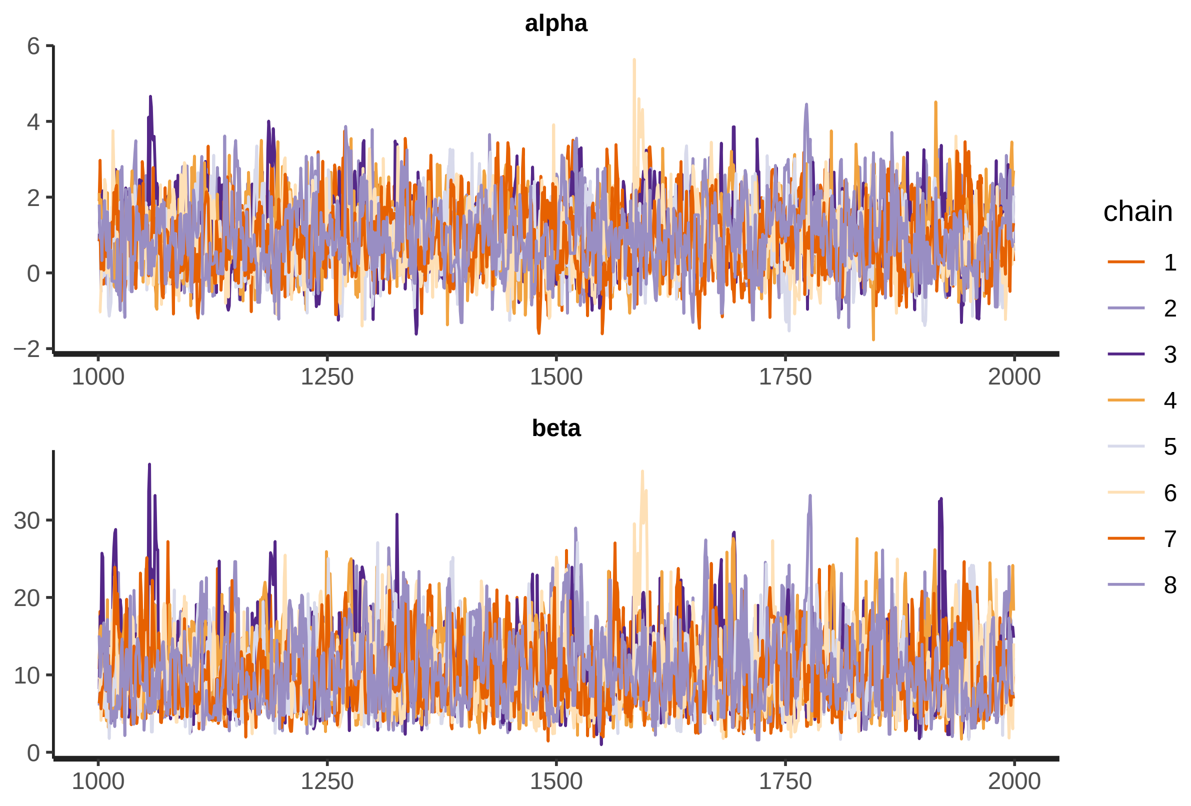
Alpha	Beta
0.6740347	0.7278982
-0.4211349	-1.1499405
0.9324464	-0.8094750
1.0430495	1.0563392
0.7632212	1.8586861
-0.1538331	1.9429743
-1.3372338	-1.1352896
-0.0138976	0.9994300

- The number of iterations for each chain (including warmup) is 2,000, which is Stan's default.

- The number of warmup iterations per chain is by default $\frac{\#\text{ iterations}}{2}$, so in this case we have 1,000 warmup iterations.
- The number of chains used were 8, which is the same number of chains I used in assignment 5. Stan uses 4 chains by default but I modified this parameter to 8.
- Figures below display all chains for α and β in a single line-plot

```
p <- traceplot(fit1, pars = c('beta[1]', 'beta[2]'), nrow=2)

p$data <- p$data %>%
  mutate(parameter = ifelse(parameter=='beta[1]', 'alpha','beta'))
p
```



- Finally here are two tables summarizing the results and showing the MCSEs respectively. Using both of them we can report the posterior statistics of interest by taking into account the MCSEs. We are deciding the digits to report by leaving out digits that are just random noise.

```
df_summary <- summarise_draws(fit1) %>%
  mutate(variable = case_when(
    variable == 'beta[1]' ~ "alpha",
    variable == 'beta[2]' ~ "beta",
    TRUE ~ variable
  ))

df_summary_mcse <- summarise_draws(fit1, default_mcse_measures()) %>%
  mutate(variable = case_when(
    variable == 'beta[1]' ~ "alpha",
    variable == 'beta[2]' ~ "beta",
    TRUE ~ variable
  ))

df_summary %>%  kbl(booktabs = T, linesep = "") %>%
  kable_styling(latex_options = c("striped", "scale_down", "HOLD_position"))
```

variable	mean	median	sd	mad	q5	q95	rhat	ess_bulk	ess_tail
alpha	0.9587426	0.9182864	0.8999463	0.8965249	-0.4365250	2.514810	1.003022	2260.786	3045.963
beta	10.5256423	9.8122278	4.7364295	4.5914713	4.2516746	19.179540	1.003253	2257.820	2631.970
theta[1]	-8.0933099	-7.5119661	3.5893587	3.4050041	-14.7511538	-3.316615	1.002558	2632.447	2872.844
theta[2]	-2.1989501	-2.0582472	1.1192628	1.0187317	-4.2565869	-0.616999	1.000592	4335.279	3664.654
theta[3]	0.4324604	0.4053246	0.7767167	0.7634955	-0.7849576	1.746120	1.002000	2733.743	3782.172
theta[4]	8.6424615	8.1231152	4.0743416	4.0508513	3.0827559	16.019328	1.003680	2058.414	2624.730
lp___	-7.1358986	-6.8030206	1.0512766	0.7030326	-9.2676034	-6.170162	1.001575	2717.966	2933.452

```
df_summary_mcse %>%  kbl(booktabs = T, linesep = "") %>%
  kable_styling(latex_options = c("striped", "HOLD_position"))
```

variable	mcse_mean	mcse_median	mcse_sd	mcse_q5	mcse_q95
alpha	0.0191361	0.0189975	0.0127632	0.0280081	0.0463592
beta	0.1032054	0.1129761	0.0960054	0.0677764	0.3654693
theta[1]	0.0732296	0.0823946	0.0718366	0.2179487	0.0578019
theta[2]	0.0180212	0.0169450	0.0188701	0.0604303	0.0231087
theta[3]	0.0149547	0.0166939	0.0096695	0.0247443	0.0257532
theta[4]	0.0923022	0.1190339	0.0809120	0.0637665	0.2206688
lp___	0.0221362	0.0178070	0.0372683	0.0849553	0.0027546

Then we can report the mean and quantiles estimates for α as:

$$E_{p(\alpha|y)}(\alpha) = 1.0$$
$$\text{quantiles 5\% and 95\%: } [-0.4, 2.5]$$

and the mean and quantiles estimates for β as:

$$E_{p(\beta|y)}(\beta) = 11$$
$$\text{quantiles 5\% and 95\%: } [4.3, 19]$$

2. Use $\widehat{R}$ for convergence analysis. You can either use Eq. (11.4) in BDA3 or the later version that can be found here. You should specify which $\widehat{R}$ you used. In R the best choice is to use function `Rhat` from package `rstan` (see `?rstan::Rhat`). To check $\widehat{R}$ and other diagnostics, you can also call `monitor(fit)`, where `fit` is the fit object returned by Stan's sampling function. Report the $\widehat{R}$ values both for α and β and discuss the convergence of the chains.

To obtain $\widehat{R}$ which in this case is the maximum of rank normalized split-$\widehat{R}$ and rank normalized folded-split-$\widehat{R}$ we can either use `rstan::Rhat()` function which requires a two-dimensional array whose rows are equal to the number of iterations of the Markov Chains and whose columns are equal to the number of Markov Chains or we can extract $\widehat{R}$ directly from the inference summary of our Stan model, both paths should give us the same answer.

```r
sample <- rstan::extract(fit1, permuted=FALSE)
df_summary_conv <- summarise_draws(fit1, default_convergence_measures()) %>%
  mutate(variable = case_when(
    variable == 'beta[1]' ~ "alpha",
    variable == 'beta[2]' ~ "beta",
    TRUE ~ variable
  ))

# Rhat Alpha
rstan::Rhat(sample[,,1])
```

```
## [1] 1.003021
```

```r
df_summary_conv$rhat[1]
```

```
## [1] 1.003022
```

```r
# Rhat Beta
rstan::Rhat(sample[,,2])
```

```
## [1] 1.003253
```

```r
df_summary_conv$rhat[2]
```

```
## [1] 1.003253
```

```r
df_summary_conv %>%  kbl(booktabs = T, linesep = "") %>%
  kable_styling(latex_options = c("striped", "HOLD_position"))
```

variable	rhat	ess_bulk	ess_tail
alpha	1.003022	2260.786	3045.963
beta	1.003253	2257.820	2631.970
theta[1]	1.002558	2632.447	2872.844
theta[2]	1.000592	4335.279	3664.654
theta[3]	1.002000	2733.743	3782.172
theta[4]	1.003680	2058.414	2624.730
lp___	1.001575	2717.966	2933.452

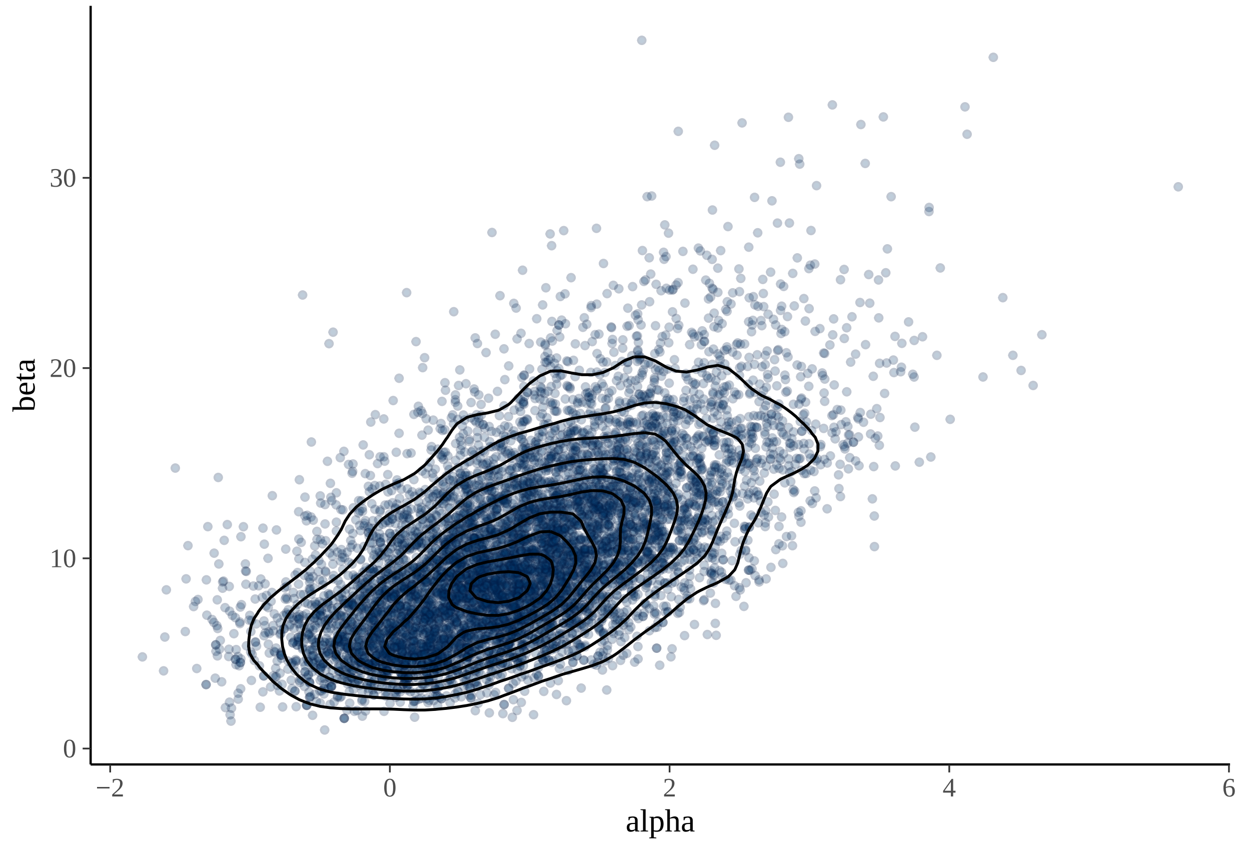
Having an $\widehat{R} = 1$ means that the between and within variance in our simulated sequences is the same and that our chains are mixed and have converged. in this case we obtained values for α and β pretty close to 1, specifically we got: 1.003, and 1.003 respectively meaning that the between and within chain variance are about equal. According to `rstan::Rhat` documentation the convention is that if $\widehat{R}$ is big (e.g., $\widehat{R} > 1.05$) we should keep sampling, in this case both values are less than 1.05 and greater than 1, indicating that the chains have converged and mixed.

Plot the draws for α and β (scatter plot) and include this plot in your report. You can compare the results to Figure~3.3b in BDA3 to verify that your code gives sensible results. Notice though that the results in Figure~3.3b are generated from posterior with a uniform prior, so even when your algorithm works perfectly, the results will look slightly different (although fairly similar).

The figure below is a scatter plot showing the the draws from the posterior for α and β.

```r
p1 <- mcmc_scatter(fit1, pars = c("beta[1]", "beta[2]"), size = 1, alpha = 0.25)
```

```
p1 + stat_density_2d(color = "black", linewidth = .5) +
  labs(y = "beta", x = "alpha")
```


beta
alpha