BSDA: Assignment 6

October 8, 2023

```
knitr::opts_chunk$set(echo = TRUE, comment = '')
```

Times used for reading and self-study exercise: 5
Time used for the assignment: 6
Good with the assignment:
It is good that we got to work with stan function and syntax in this assignment.
Things to improve in the assignment:
One thing that can be improved for next year is when reporting the $\hat{R}$ it is unclear if we should report the digits of $\hat{R}$ based on MCSE or not because I would say that $\hat{R}$ is a posterior statistics of interest.

Question 1: Model in Stata syntax

For Question 1 we will present the stata syntax for the model in the example of section 3.7 and we are getting access to the bioassay data by using the following code. (Reference: Bayesian Data Analysis Third edition, chapter: 3, page: 75-76)

```
library(rstan)
data("bioassay")
setwd("~/Desktop/stan-demo-files")
```

Now when we have got access to the data, the chunk below is the stata syntax for calculating the generalized liner model from chapter 3.7 in the course literature.

```
data {
  real x[4];
  int <lower=0> n[4];
  int <lower=0> y[4];
  vector[2] mu;
  matrix[2,2] Sigma;
}
parameters {
  vector[2] theta;
}
transformed parameters {
  real alpha = theta[1];
  real beta = theta[2];
}
model {
  theta ~ multi_normal(mu, Sigma);
```

```
  for ( i in 1:4){
    y[i] ~ binomial_logit(n[i], alpha + beta * x[i]);
  }
}
```

so the stata syntax consist of four different parts. One part for the data, one for the model and two parts for the parameter vector $\boldsymbol{\theta} = (\alpha, \beta)$. As can be seen in the syntax we are using the following Gaussian prior in the model:

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

The reasoning behind setting function argument chains, iter and warmup to 4, 5000 and 1500 respectively can be found in part $c - g$ in Assignment 5. Below we replicate the computations for the bioassay example by using the stan-function. Further, because the fit object is a posterior statistics we decide to only report the number of digits for which is Monte Carlo standard error (MCSE) is zero.

```
mu_vector = c(0,10)
Sigma_vector = matrix(c(2^2,12,12,10^2),2,2)

data_list <- list(x = c(-0.86, -0.30,-0.05,0.73),
                  n = c(5,5,5,5),
                  y = c(0,1,3,5),
                  mu = mu_vector,
                  Sigma = Sigma_vector)
set.seed(123)

fit_model <- stan(file = "Assign6.stan",
                  data = data_list,
                  chains = 4, iter = 5000, warmup = 1500)
```

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
clang -mmacosx-version-min=10.13 -I"/Library/Frameworks/R.framework/Resources/include" -DNDE
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/StanHea
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEig
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEig
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/src/C
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/src/C
namespace Eigen {
               ^
               ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/StanHea
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEig
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/Core:
#include <complex>
         ^~~~~~~~~
3 errors generated.
make: *** [foo.o] Error 1
```

```
SAMPLING FOR MODEL 'Assign6' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000167 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.67 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 1: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 1: Iteration: 1501 / 5000 [ 30%]  (Sampling)
Chain 1: Iteration: 2000 / 5000 [ 40%]  (Sampling)
Chain 1: Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.081124 seconds (Warm-up)
Chain 1:                0.207382 seconds (Sampling)
Chain 1:                0.288506 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'Assign6' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 1.6e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.16 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 2: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 2: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 2: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 2: Iteration: 1501 / 5000 [ 30%]  (Sampling)
Chain 2: Iteration: 2000 / 5000 [ 40%]  (Sampling)
Chain 2: Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 2: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 2: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 2: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 2: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 2: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.079241 seconds (Warm-up)
Chain 2:                0.207411 seconds (Sampling)
Chain 2:                0.286652 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'Assign6' NOW (CHAIN 3).
```

```
Chain 3:
Chain 3: Gradient evaluation took 1.6e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.16 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 3: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 3: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 3: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 3: Iteration: 1501 / 5000 [ 30%]  (Sampling)
Chain 3: Iteration: 2000 / 5000 [ 40%]  (Sampling)
Chain 3: Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 3: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 3: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 3: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 3: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 3: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.083191 seconds (Warm-up)
Chain 3:                0.165689 seconds (Sampling)
Chain 3:                0.24888 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'Assign6' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 1.5e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.15 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 5000 [  0%]  (Warmup)
Chain 4: Iteration:  500 / 5000 [ 10%]  (Warmup)
Chain 4: Iteration: 1000 / 5000 [ 20%]  (Warmup)
Chain 4: Iteration: 1500 / 5000 [ 30%]  (Warmup)
Chain 4: Iteration: 1501 / 5000 [ 30%]  (Sampling)
Chain 4: Iteration: 2000 / 5000 [ 40%]  (Sampling)
Chain 4: Iteration: 2500 / 5000 [ 50%]  (Sampling)
Chain 4: Iteration: 3000 / 5000 [ 60%]  (Sampling)
Chain 4: Iteration: 3500 / 5000 [ 70%]  (Sampling)
Chain 4: Iteration: 4000 / 5000 [ 80%]  (Sampling)
Chain 4: Iteration: 4500 / 5000 [ 90%]  (Sampling)
Chain 4: Iteration: 5000 / 5000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0.080705 seconds (Warm-up)
Chain 4:                0.162421 seconds (Sampling)
Chain 4:                0.243126 seconds (Total)
Chain 4:
```

```
print(fit_model)
```

```
Inference for Stan model: Assign6.
```

```
4 chains, each with iter=5000; warmup=1500; thin=1;
post-warmup draws per chain=3500, total post-warmup draws=14000.

          mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
theta[1]  0.97    0.01 0.89 -0.65  0.36  0.93  1.54  2.88  4147    1
theta[2] 10.54    0.07 4.56  3.46  7.12  9.99 13.42 20.97  3999    1
alpha     0.97    0.01 0.89 -0.65  0.36  0.93  1.54  2.88  4147    1
beta     10.54    0.07 4.56  3.46  7.12  9.99 13.42 20.97  3999    1
lp__     -7.12    0.01 1.01 -9.84 -7.51 -6.82 -6.41 -6.15  4719    1

Samples were drawn using NUTS(diag_e) at Sun Oct  8 20:17:55 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).
```

```
MCSE.alpha <- 0.9184943 / sqrt(4321)
MCSE.alpha # 0.01397283
```

```
[1] 0.01397283
```

```
MCSE.beta <- 4.6165485 / sqrt(4054)
MCSE.beta # 0.07250626
```

```
[1] 0.07250626
```

```
print(fit_model, digits = 1)
```

```
Inference for Stan model: Assign6.
4 chains, each with iter=5000; warmup=1500; thin=1;
post-warmup draws per chain=3500, total post-warmup draws=14000.

         mean se_mean  sd 2.5%  25%  50%  75% 97.5% n_eff Rhat
theta[1]  1.0     0.0 0.9 -0.7  0.4  0.9  1.5   2.9  4147    1
theta[2] 10.5     0.1 4.6  3.5  7.1 10.0 13.4  21.0  3999    1
alpha     1.0     0.0 0.9 -0.7  0.4  0.9  1.5   2.9  4147    1
beta     10.5     0.1 4.6  3.5  7.1 10.0 13.4  21.0  3999    1
lp__     -7.1     0.0 1.0 -9.8 -7.5 -6.8 -6.4  -6.1  4719    1

Samples were drawn using NUTS(diag_e) at Sun Oct  8 20:17:55 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).
```

Question 2: $\hat{R}$ for convergence analysis

The $\hat{R}$ is a estimate that is often being used when looking at the convergence of several Markov chains. This estimate uses the between-sequence information and the within-sequence information when assessing the convergence to a common proposal distribution $p(\theta|y)$. Hence, $\hat{R}$ are using the mean and the variance for W and B to test if each individual sequence has reached stationary at a common distribution.

We will compute the $\hat{R}$ by using the Rhat-function from the rstan-package, but first we will extract all values for each chain and parameter for the fit object from the Stans

sampling function. Also, the warm up period will be the same as for Assignment 5 so the warm up period will be set to the first 1500 iterations in the chains

```
samples <- extract(fit_model, permuted = F)

samples.alpha <- samples[1:3500,1:4,1]
Rhat(samples.alpha) #1.001476
```

```
[1] 1.001002
```

```
samples.beta <- samples[1:3500,1:4,2]
Rhat(samples.beta) # 1.000727
```

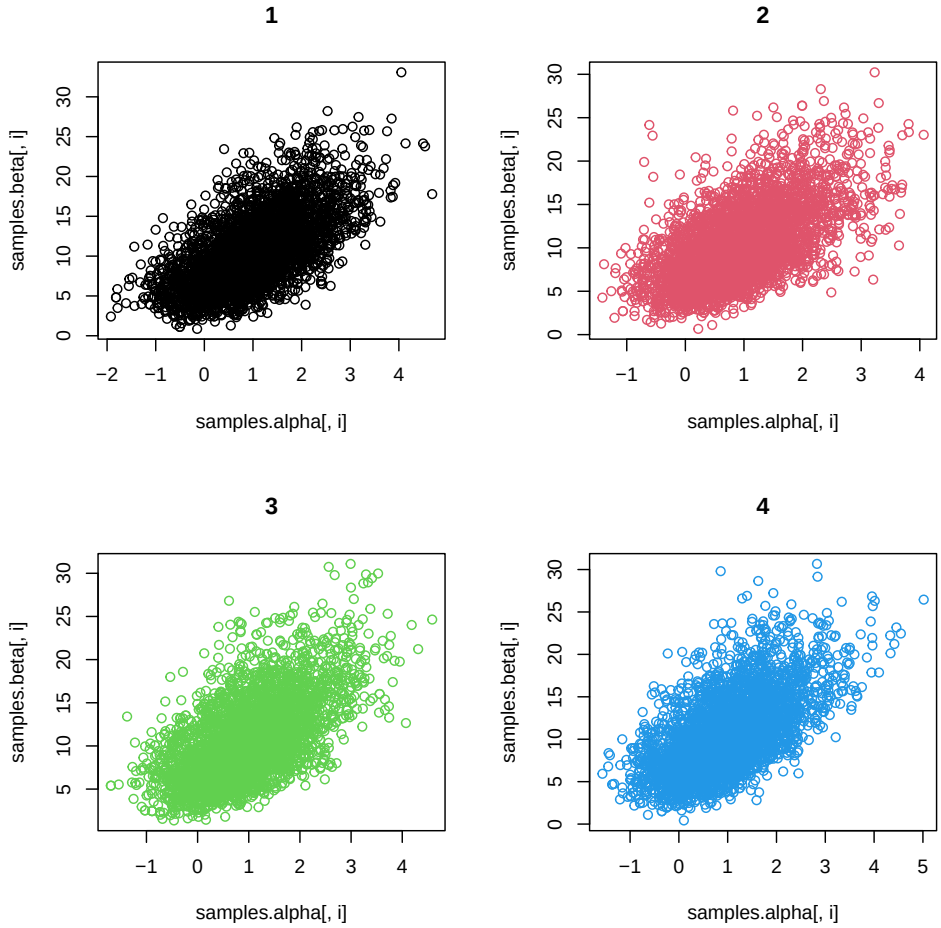
```
[1] 1.00022
```

and we got that $\hat{R}_\alpha = 1.001476$ and $\hat{R}_\beta = 1.000727$ when using n = 3500 iterations after removing the first 1500 iterations as the warm up period. Hence, we got a $\hat{R}$ close to 1 for both α and β which can be seen as a indicator for the chains to be converged. We are using the rule of thumb that there is still reason to believe that the chains has not converged yet when having $\hat{R} > 1.01$.

Question 3: Scatter plot for α and β

For Q3 we will plot the draws for α and β in a scatter plot because $\hat{R}$ is not, by itself, sufficient for convergence analysis regarding MCs. When plotting a scatter plot of Markov chains then we lose the dependence aspect of the sequence, hence if the plot looks like independent draws from a joint distribution then it indicates of the chain being converged.

```
par(mfrow=c(2,2))
n = 4
for (i in 1:n) {
  plot(x = samples.alpha[,i], y = samples.beta[,i],
       main = i,
       col = i)
}
```



Above, all four Markov Chains consist of 3500 iterations displayed on a scatter plot. The plots indicated that all four Markov chains has converged to a common joint distribution.