BDA - Assignment 6

Anonymous

Contents

Exercise 1

```
library(aaltobda)
data("bioassay")
library(cmdstanr)
options(mc.cores = 1)
library(posterior)
library(ggplot2)
```

$$\begin{bmatrix}\alpha \\ \beta\end{bmatrix} \sim N(\mu_0, \Sigma_0) \text{ where } \mu_0 = \begin{bmatrix}0 \\ 10\end{bmatrix} \text{ and } \Sigma_0 = \begin{bmatrix}2^2 & 12 \\ 12 & 10^2\end{bmatrix}$$

There will be 4 chains with 1000 iterations after 1000 warm ups.

```
bioassay_stan <- "
data {
  int<lower=0> N;
  vector[N] x;
  array[N] int y;
  array[N] int n;
  vector[2] mu0;
  matrix[2,2] Sigma;
}

// The parameters accepted by the model.
parameters {
  real alpha;
  real beta;
}
```

```
transformed parameters {
  // deterministic transformation of parameters and data
  vector[N] mu = alpha + beta * x; // linear model
}

// The model to be estimated.
model {
  [alpha, beta] ~ multi_normal(mu0, Sigma);
  y ~ binomial_logit(n, mu);
}"
data_list <- list(
  mu0 = c(0,10),
  Sigma = cbind(c(4,12),c(12,100)),
  N = nrow(bioassay),
  x = bioassay$x,
  n = bioassay$n,
  y = bioassay$y
  )

model <- cmdstan_model(write_stan_file(bioassay_stan))
fit <- model$sample(data = data_list, refresh = 1000)
```

```
## Running MCMC with 4 sequential chains...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 0.1 seconds.
## Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 finished in 0.0 seconds.
## Chain 3 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3 finished in 0.1 seconds.
## Chain 4 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.1 seconds.
## Total execution time: 0.7 seconds.
```

Exercise 2

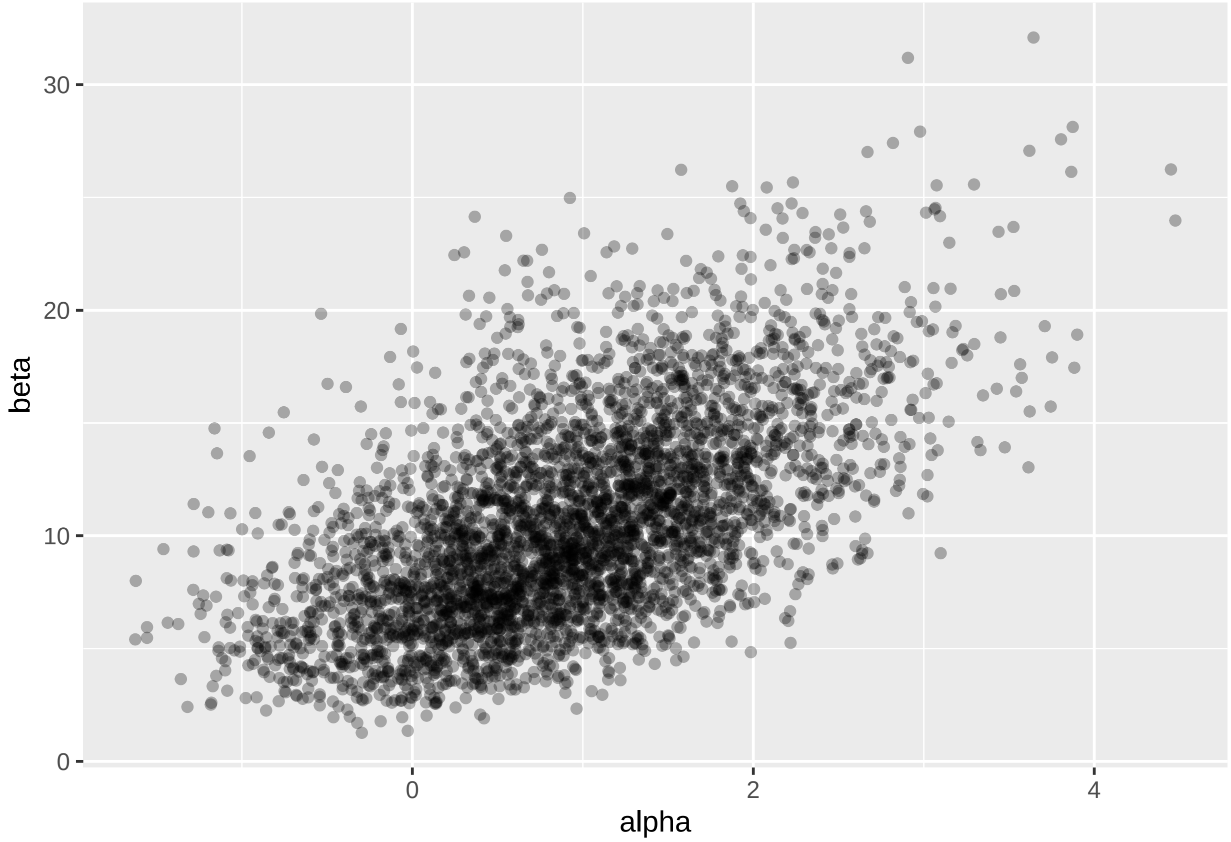
```
fit$summary()
```

```
## # A tibble: 7 x 10
##   variable   mean median    sd   mad      q5    q95  rhat ess_bulk ess_tail
##   <chr>     <dbl>  <dbl> <dbl> <dbl>   <dbl>  <dbl> <dbl>    <dbl>    <dbl>
## 1 lp__     -7.13  -6.83  0.998 0.742  -9.19  -6.17   1.00    1577.    1898.
## 2 alpha     0.925  0.911 0.892 0.888  -0.532  2.43   1.00    1202.    1600.
## 3 beta     10.6    9.98  4.53  4.48    4.15  18.8    1.00    1198.    1658.
## 4 mu[1]    -8.16  -7.70  3.44  3.32  -14.4   -3.38   1.00    1447.    1700.
## 5 mu[2]    -2.24  -2.12  1.10  1.06   -4.22  -0.649  1.00    2642.    1979.
## 6 mu[3]     0.397  0.402 0.780 0.771  -0.887  1.71   1.00    1470.    1794.
## 7 mu[4]     8.63   8.21  3.90  3.99    3.00  15.6    1.00    1078.    1618.
```

From the summary function we get $\hat{R}$ values of 1.003 for both α and β (The values show as 1.00 after knitting the file). Both of these are less than 1.02 which was given as an acceptable value in the lecture materials. This means our convergence is good, since adding iterations doesn't greatly change the degree of differences between our chains.

Exercise 3

```
bio_draws <- as_draws_df(fit$draws())
ggplot() +
  geom_jitter(data = bio_draws, width=0.05, height=0.05,
              aes(alpha, beta), alpha=0.3)+
  labs(x = 'alpha', y = 'beta')
```



Exercise 4

I used my own windows PC for the task
I used R as the programming environment
I used CmdStanR as the interface
I had some installation and compilation problems, such as the need to install cmdstan after cmdstanr but got it done with the help of google and zulip posts
I successfully installed locally
Learning new syntax is always a bit of trial and error at the start and I needed to google/check zulip a lot to get functional code. In the end it wasn't that difficult, but still time consuming.