Homework 3 Solutions

LDA & Bayes

In the first part of this assignment, you will be working with text from @realDonaldTrump Twitter. The text was taken from all tweets Donald Trump sent between 01/19/2016 and 01/19/2018. The goal is to use Latent Dirichlet Allocation in order to model the topics that the president tweeted about during this time. In the second part of this assignment, you are provided with data sets **dataset-2-train.txt** and **dataset-2-test.txt** containing details of contraceptive usage by 1934 Bangladeshi women. There are four attributes for each woman, along with a label indicating if she uses contraceptives. The attributes include:

- district: identifying code for the district the woman lives in
- urban: type of region of residence
- living.children: number of living children
- age-mean: age of women (in years, centred around mean)

The women are grouped into 60 districts. The task is to build a classification model that can predict if a given woman uses contraceptives.

1 Data Preperation

The tweet data is provided for you as **trump-tibble.csv**. After you read the data into R, you'll see that there are only two columns: document and text. The document column contains the date and time of the tweet, and the text column contains the actual text of the tweet. Before you begin, you'll want to cast the columns as characters rather than factors. You can do this with the following code:

```
# cast factors to characters
trump_tibble$document <- as.character(trump_tibble$document)
trump_tibble$text <- as.character(trump_tibble$text)
```

The following libraries will be of use for this problem:

```
#load libraries
library(topicmodels) #topic modeling functions
library(stringr) #common string functions
library(tidytext) #tidy text analysis
suppressMessages(library(tidyverse)) #data manipulation and visualization
## Source topicmodels2LDAvis & optimal_k functions
invisible(lapply(file.path("https://raw.githubusercontent.com/trinker/topicmodels_learning/master/functi
c("topicmodels2LDAvis.R", "optimal_k.R")),
devtools::source_url))
```

```
## SHA-1 hash of file is 5ac52af21ce36dfe8f529b4fe77568ced9307cf0
```

```
## SHA-1 hash of file is 7f0ab64a94948c8b60ba29dddf799e3f6c423435
```

```
## Loading required package: pacman
```

A. Use the `unnest_tokens` function to extract words from the tweets text.

B. Create a dataframe consisting of the document-word counts.

C. Create a document-term matrix using the `cast_dtm` function.

Solution

A.

```
#Read Data
trump_tibble<-read.csv('Data/trump_tibble.csv')

# cast factors to characters
trump_tibble$document <- as.character(trump_tibble$document)
trump_tibble$text <- as.character(trump_tibble$text)

#unnnest tokens
by_tweet_word <- trump_tibble %>%
       unnest_tokens(word, text)
```

B.

```
 # find document-word counts
word_counts <- by_tweet_word %>%
       anti_join(stop_words) %>%
       dplyr::count(document, word, sort = TRUE) %>%
       ungroup()
```

```
## Joining, by = "word"
```

C.

```
# create document-term matrix
tweets_dtm <- word_counts %>%
       cast_dtm(document, word, n)
```

2 LDA

A. Using the following control parameters, run the optimal-k function to search for the optimal number of topics. Be sure to set the "max.k" parameter equal to 30.

```
control<-list(burnin=500,iter=100,keep=100,seed=46)
```

B. Plot the results of the optimal-k function. What does this plot suggest about the number of topics in the text?

C. Run LDA on the document-term matrix using the optimal value of k. Print out the top 10 words for each of the k topics. Comment on the results and their plausibility. What does each topic seem to represent?

Solution

A.

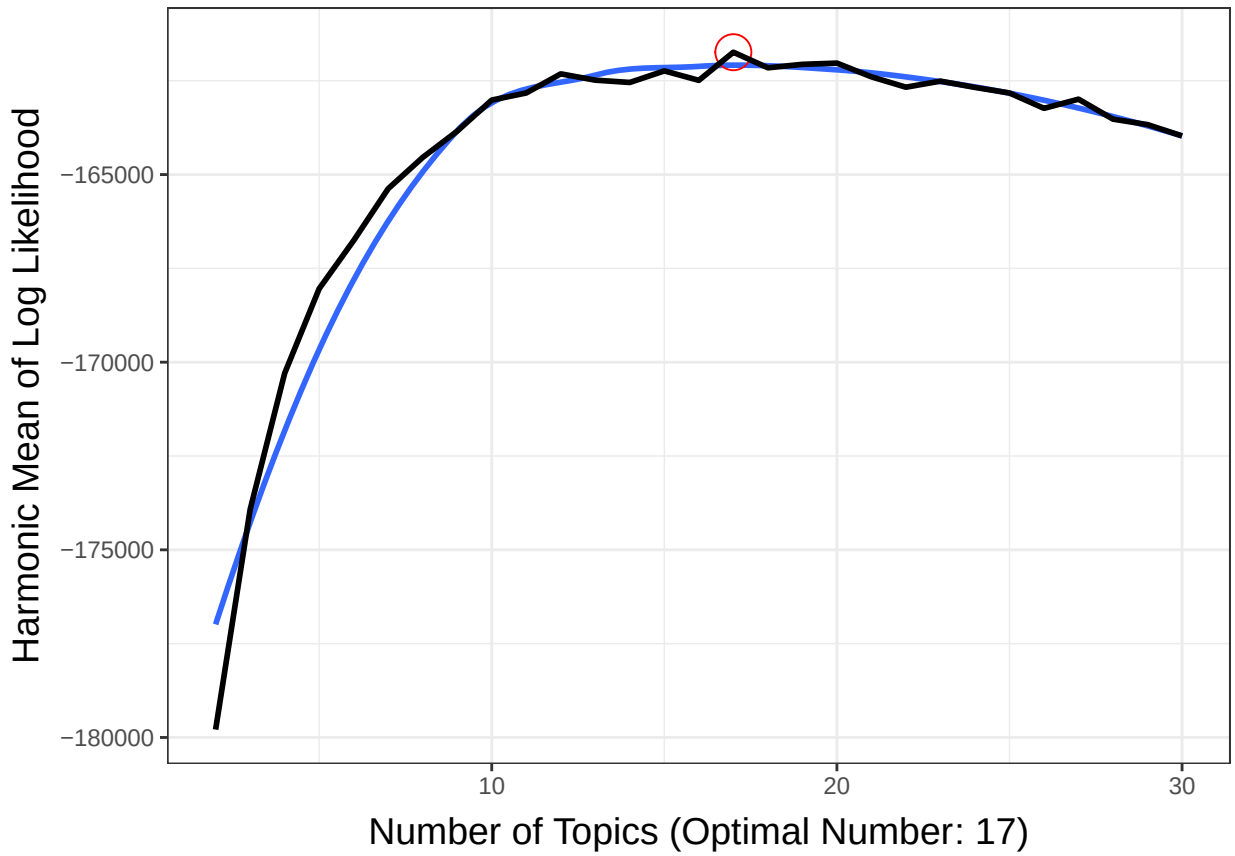
```
set.seed(100)
opt_k <- optimal_k(tweets_dtm,
                       max.k=30,
                       control=control,
                   drop.seed=F)
```

```
##
## Grab a cup of coffee this could take a while...
```

```
## 10 of 30 iterations (Current: 09:24:08; Elapsed: .1 mins)
## 20 of 30 iterations (Current: 09:24:22; Elapsed: .3 mins; Remaining: ~.4 mins)
## 30 of 30 iterations (Current: 09:24:40; Elapsed: .6 mins; Remaining: ~0 mins)
## Optimal number of topics = 17
```

B.

```
opt_k
```



This plot suggests that there are 17 different topics in the text, as a value of k = 17 maximized the harmonic mean of the log-likelihood of the model. Depending on what seed you set (or if you set one at all) your answer will vary.

C.

```
trump_lda = LDA(tweets_dtm,
                k=as.numeric(opt_k),
                method="Gibbs",
                control=control)

#look at topics
lda_inf = posterior(trump_lda)
topics.hp = topics(trump_lda,1)
terms.hp = terms(trump_lda, 10)
print(terms.hp[,])
```

```
##       Topic 1    Topic 2     Topic 3         Topic 4     Topic 5
##  [1,] "jobs"     "trump"     "enjoy"         "people"    "america"
##  [2,] "tax"      "president" "tonight"       "american"  "women"
##  [3,] "bill"     "obama"     "interviewed"   "happy"     "dollars"
```

```
##  [4,] "cuts"     "donald"    "7"             "meetings"  "united"
##  [5,] "record"   "vote"      "00"            "elizabeth" "god"
##  [6,] "market"   "hit"       "morning"       "york"      "day"
##  [7,] "stock"    "terrorism" "campaign"      "yesterday" "rally"
##  [8,] "economy"  "wonderful" "foxandfriends" "millions"  "wow"
##  [9,] "business" "1"         "change"        "warren"    "pay"
## [10,] "strong"   "usa"       "remember"      "goofy"     "world"
##       Topic 6                 Topic 7   Topic 8       Topic 9
##  [1,] "trump2016"             "hillary" "republican"  "time"
##  [2,] "makeamericagreatagain" "clinton" "obamacare"   "nice"
##  [3,] "ohio"                  "crooked" "house"       "job"
##  [4,] "votetrump"             "bad"     "healthcare"  "bad"
##  [5,] "love"                  "run"     "democrats"   "forward"
##  [6,] "california"            "bernie"  "republicans" "won"
##  [7,] "tough"                 "fbi"     "senate"      "law"
##  [8,] "supertuesday"          "beat"    "votes"       "total"
##  [9,] "words"                 "sanders" "repeal"      "taking"
## [10,] "florida"               "rigged"  "replace"     "low"
##       Topic 10    Topic 11   Topic 12          Topic 13      Topic 14
##  [1,] "north"     "country"  "u.s"             "cruz"        "join"
##  [2,] "korea"     "military" "election"        "ted"         "tomorrow"
##  [3,] "hampshire" "security" "time"            "lyin"        "arizona"
##  [4,] "carolina"  "border"   "watch"           "lost"        "maga"
##  [5,] "special"   "wall"     "russia"          "poll"        "americans"
##  [6,] "honor"     "massive"  "deal"            "kasich"      "day"
##  [7,] "china"     "stop"     "america"         "safe"        "talk"
##  [8,] "america"   "trade"    "congratulations" "incredible"  "supporters"
##  [9,] "meeting"   "national" "support"         "jeb"         "tickets"
## [10,] "hard"      "mexico"   "amazing"         "endorsement" "live"
##       Topic 15       Topic 16    Topic 17
##  [1,] "vote"         "news"      "people"
##  [2,] "win"          "fake"      "night"
##  [3,] "day"          "media"     "spent"
##  [4,] "alabama"      "cnn"       "press"
##  [5,] "luther"       "totally"   "senator"
##  [6,] "pennsylvania" "nytimes"   "ratings"
##  [7,] "illegal"      "failing"   "crowd"
##  [8,] "race"         "dishonest" "fight"
##  [9,] "strange"      "story"     "administration"
## [10,] "john"         "phony"     "governor"
```

There are multiple correct answers here, as long as justification is provided. Topic 1 seems to represent the state of the economy, the strong stock market, and jobs. Topics 2 and 7 deal with campaigning and the election. Topics 3 seems to deal with interviews and Trump's appearances on television. Topic 4 has to do with Trump's talk about border security. Topics 5, 8, and 15 can be seen as Trump's general "MAGA"" rhetoric. Topics 6 and 13 are both on Trump disparaging his political opponents. Topic 9 is all about the "fake news". Topics 10 and 12 are somewhat miscellaneous. Topic 11 has to do with Obamacare and discussion about the legislative branch. Topic 14 is somewhat focused on women. Overall, it appears that the LDA model did a good job of capturing the latent topics that are embedded within the president's tweets.

3 Bayesian Logistic Regresion

The first model we will fit to the contraceptives data is a varying-intercept logistic regression model, where the intercept varies by district:

- Prior distribution:

$$\beta_{0j} \sim N(\mu_0, \sigma_0), \text{ with } \mu_0 \sim N(0, 100) \text{ and } \sigma_0 \sim \text{Exponential}(.1)$$
$$\beta_1 \sim N(0, 100)\ ,\ \beta_2 \sim N(0, 100)\ ,\ \beta_3 \sim N(0, 100)$$

- Model for data:

$$Y_{ij} \sim \text{Bernoulli}(p_{ij})$$

$$\text{logit}\ p_{ij} = \beta_{0j} + \beta_1 * \text{urban} + \beta_2 * \text{living-children} + \beta_3 * \text{age-mean}$$

where Y_{ij} is 1 if woman i in district j uses contraception, and 0 otherwise, and where $i = 1, \ldots, N$ and $j = 1, \ldots, J$ (N is the number of observations in the data, and J is the number of districts). The above notation assumes $N(\mu, \sigma)$ is a normal distribution with mean μ and standard deviation σ. To verify the procedure, we will generate fake data with parameters of our choice and fit our model to it. This is done to ensure that the model can be fit correctly.

After you read the train and test data into R, the following code will help with formatting:

```
# convert everything to numeric
for (i in 1:ncol(train)) {
  train[,i] <- as.numeric(as.character(train[,i]))
  test[,i] <- as.numeric(as.character(test[,i]))
}
# map district 61 to 54 (so that districts are in order)
train_bad_indices <- which(train$district == 61)
train[train_bad_indices, 1] <- 54
test_bad_indices <- which(test$district == 61)
test[test_bad_indices, 1] <- 54
```

A. To verify the procedure, simulate binary response data (using the `rbinom` function) assuming the following parameter values (and using the existing features and district information from the training data):

```
mu_beta_0 = 2
sigma_beta_0 = 1
set.seed(123)  # to ensure the next line is common to everyone
beta_0 = rnorm(n=60, mean=mu_beta_0, sd=sigma_beta_0)
beta_1 = 4
beta_2 = -3
beta_3 = -2
```

B. Fit the varying-intercept model specified above to your fake data

C. Plot the trace plots of the MCMC sampler for the parameters $\mu_{\beta_0}, \sigma_{\beta_0}, \beta_1, \beta_2, \beta_3$. Does it look like the samplers converged?

D. Plot histograms of the posterior distributions for the parameters $\beta_{0,10}, \beta_{0,20} \ldots \beta_{0,60}$. Are the actual parameters that you chose contained within these posterior distributions?

We now fit our model to the actual data. E. Fit the varying-intercept model to the real train data. Make sure to set a seed at 46 within the Stan function, to ensure that you will get the same results if you fit your model correctly.

F. Check the convergence by examining the trace plots, as you did with the fake data.

G. Based on the posterior means, women belonging to which district are most likely to use contraceptives? Women belonging to which district are least likely to use contraceptives?

H. What are the posterior means of μ_{β_0} and σ_{β_0}? Do these values offer any evidence in support of or against the varying-intercept model?

3 Solution

A.

```
# load libraries
library(plyr)
```

```
## ------------------------------------------------------------------------------
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
## ------------------------------------------------------------------------------
##
## Attaching package: 'plyr'
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
## The following object is masked from 'package:purrr':
##
##     compact
```

```
library(dplyr)
library(ggplot2)
library(rstan)
```

```
## Loading required package: StanHeaders
## rstan (Version 2.17.3, GitRev: 2e1f913d3ca3)
## 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)
##
## Attaching package: 'rstan'
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
# import data
train <- read.table('Data/dataset_2_train.txt', sep=",", header=T)
test <- read.table('Data/dataset_2_test.txt', sep=",", header=T)

# convert everything to numeric
for (i in 1:ncol(train)) {
  train[,i] <- as.numeric(as.character(train[,i]))
  test[,i] <- as.numeric(as.character(test[,i]))
}
# map district 61 to 54 (so that districts are in order)
train_bad_indices <- which(train$district == 61)
train[train_bad_indices, 1] <- 54
test_bad_indices <- which(test$district == 61)
```

```
test[test_bad_indices, 1] <- 54


# draw parameters to generate fake data
mu_beta_0 <- 2
sigma_beta_0 <- 1
set.seed(123)
beta_0 <- rnorm(n=60, mean=mu_beta_0, sd=sigma_beta_0)
beta_1 <- 4
beta_2 <- -3
beta_3 <- -2
   # create fake data
   fake_news <- rep(NA, nrow(train))
   fake_param<-rep(NA,nrow(train))
   for (ii in 1:nrow(train)) {
     district <- train$district[ii]
     district_param <- beta_0[district]
     urban <- train$urban[ii]
     living_children <- train$living.children[ii]
     age_mean <- train$age_mean[ii]
     fake_param_raw <- district_param + (beta_1 * urban) + (beta_2 * living_children) + (beta_3 * age_m
     fake_param[ii] <- exp(fake_param_raw) / (1 + exp(fake_param_raw))
     fake_news[ii] <- rbinom(n=1, size=1, prob=fake_param[ii])
   }


   summary(fake_param)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.1922  0.4754  1.0000  1.0000
```

```
   table(fake_news)
```

```
## fake_news
##   0   1
## 508 459
```

B.

```
 # model 1: varying intercept by district
   # create list
   stan_list <- c()
   stan_list$district <- train$district
   stan_list$urban <- train$urban
   stan_list$living_children <- train$living.children
   stan_list$age_mean <- train$age_mean
   stan_list$contraceptive_use <- fake_news
   stan_list$N <- length(stan_list$contraceptive_use)
   stan_list$num_districts <- length(unique(stan_list$district))

 # fit the model
   options(mc.cores = parallel::detectCores())
   fit <- stan(file = 'solution_3b.stan',
               data = stan_list,
               iter = 3000,
```

```
                chains = 4,
                refresh=0,
                control=list(adapt_delta=0.8))
```

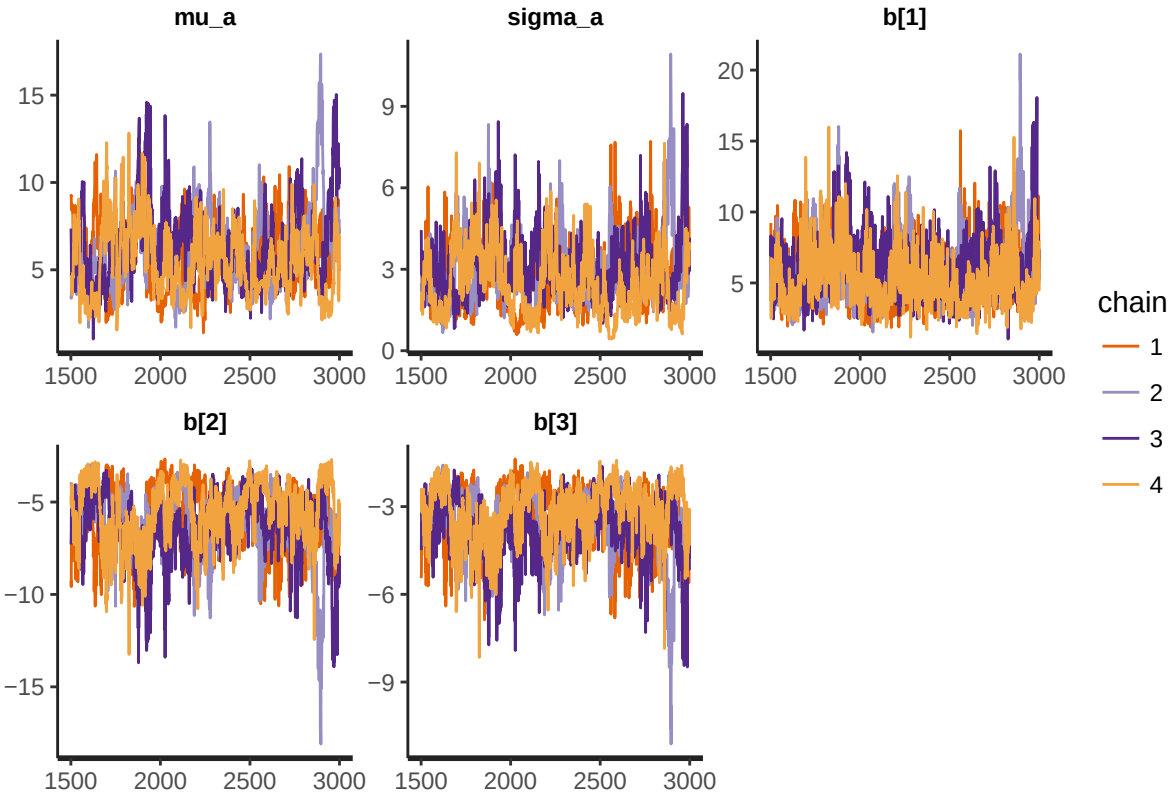
```
## Warning: There were 1 divergent transitions after warmup. Increasing adapt_delta above 0.8 may help.
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

```
## Warning: There were 4 chains where the estimated Bayesian Fraction of Missing Information was low. S
## http://mc-stan.org/misc/warnings.html#bfmi-low
```

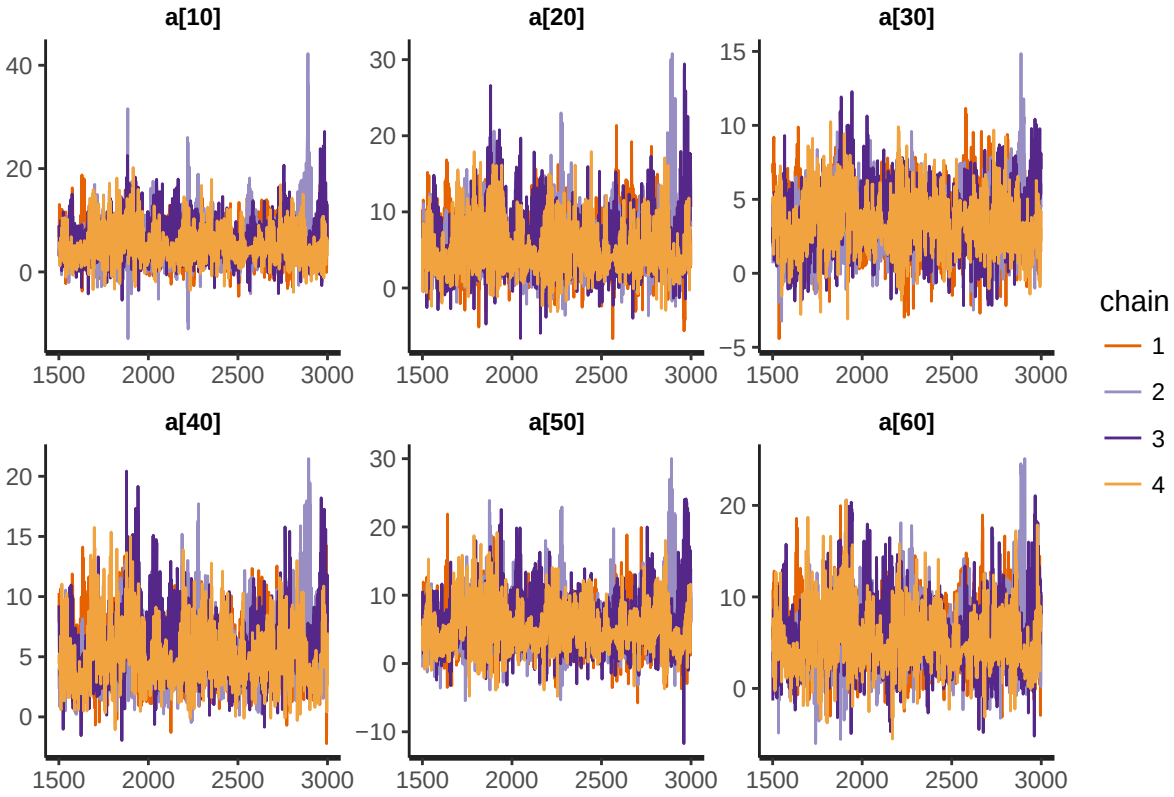
```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

C.

```
# look at convergence plots
plot(fit, plotfun="trace", pars=c('mu_a', 'sigma_a', 'b[1]', 'b[2]', 'b[3]'))
```



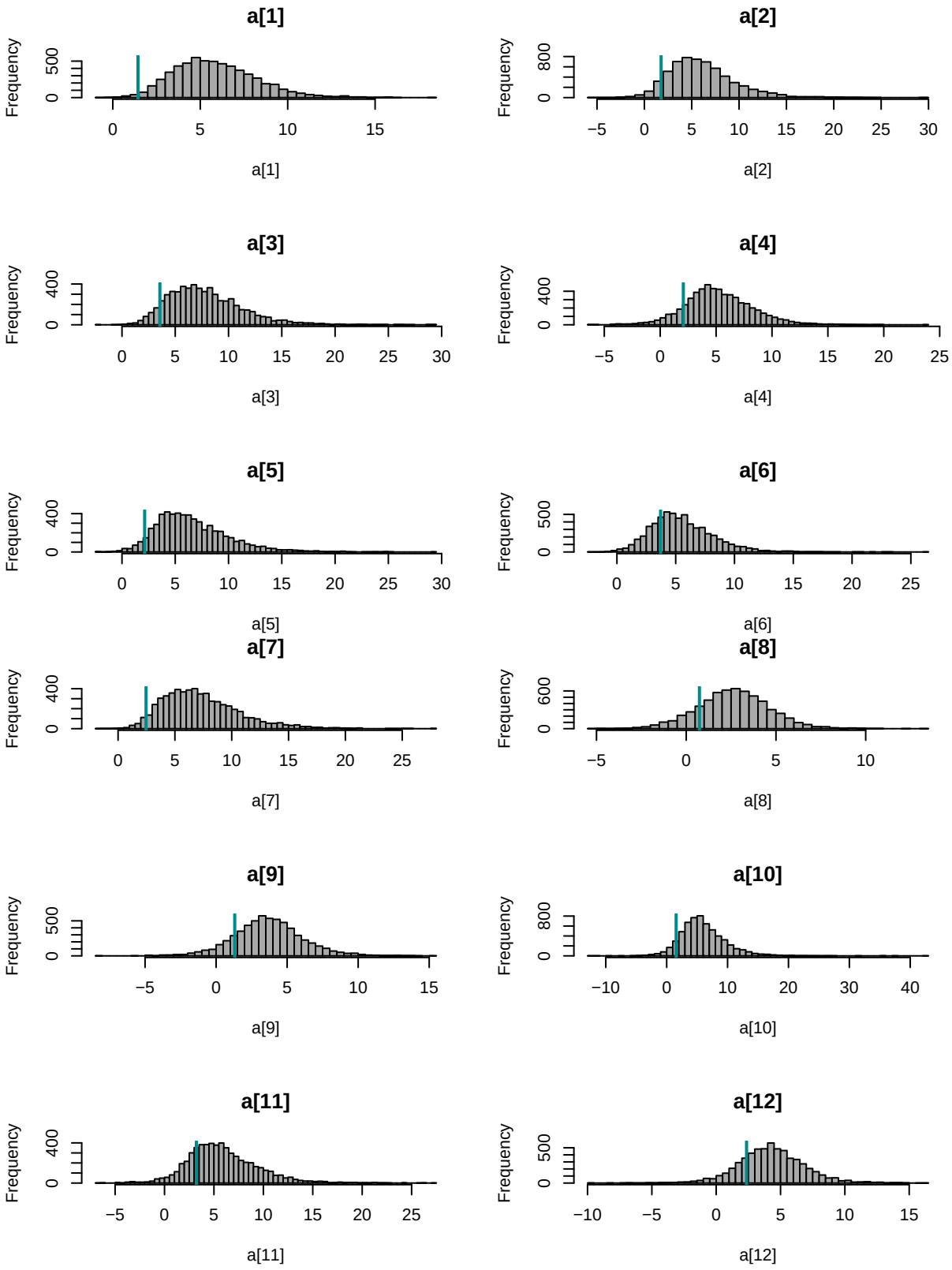

```
plot(fit, plotfun="trace", pars=c('a[10]', 'a[20]', 'a[30]',
                                  'a[40]', 'a[50]', 'a[60]'))
```

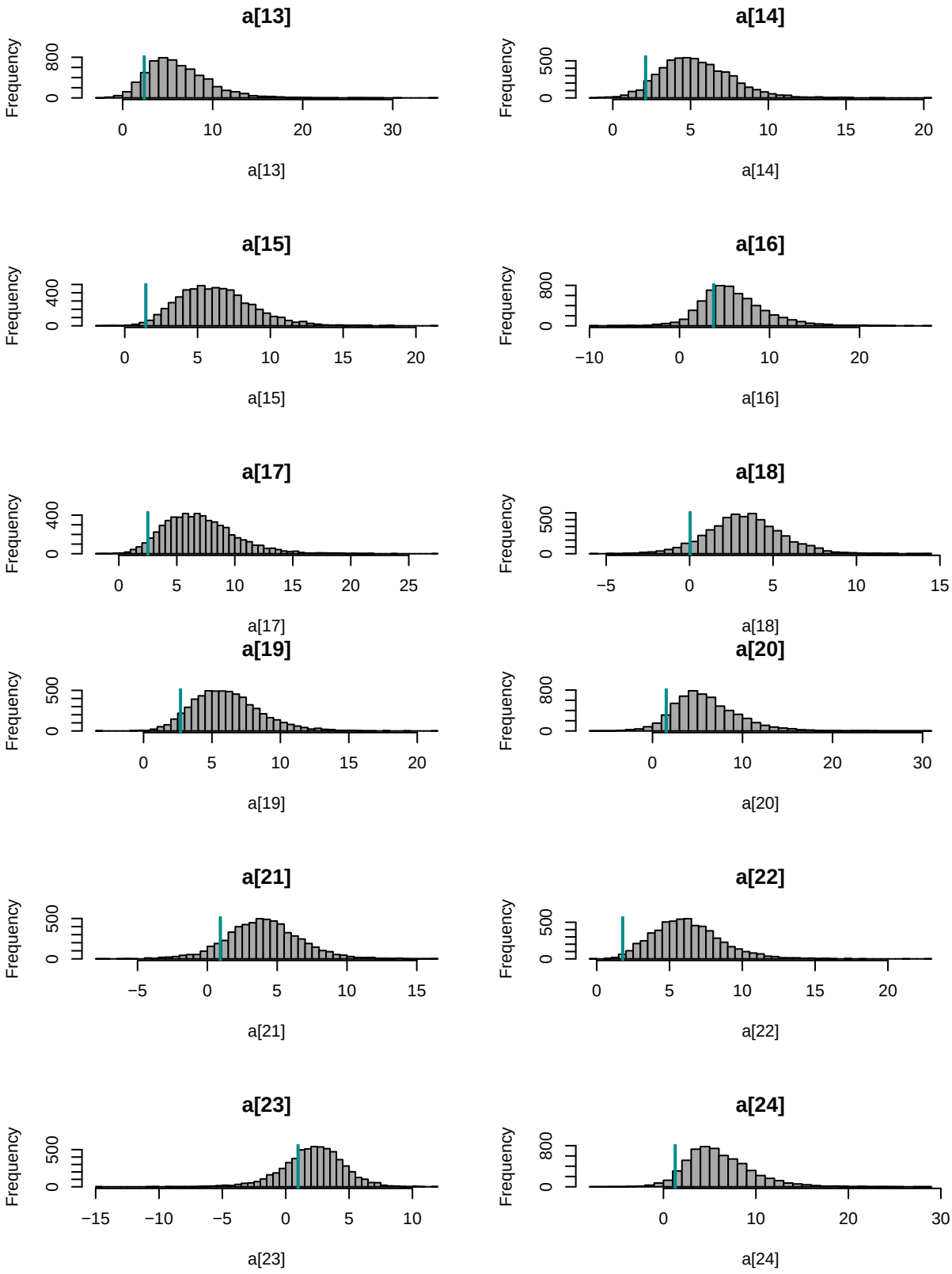


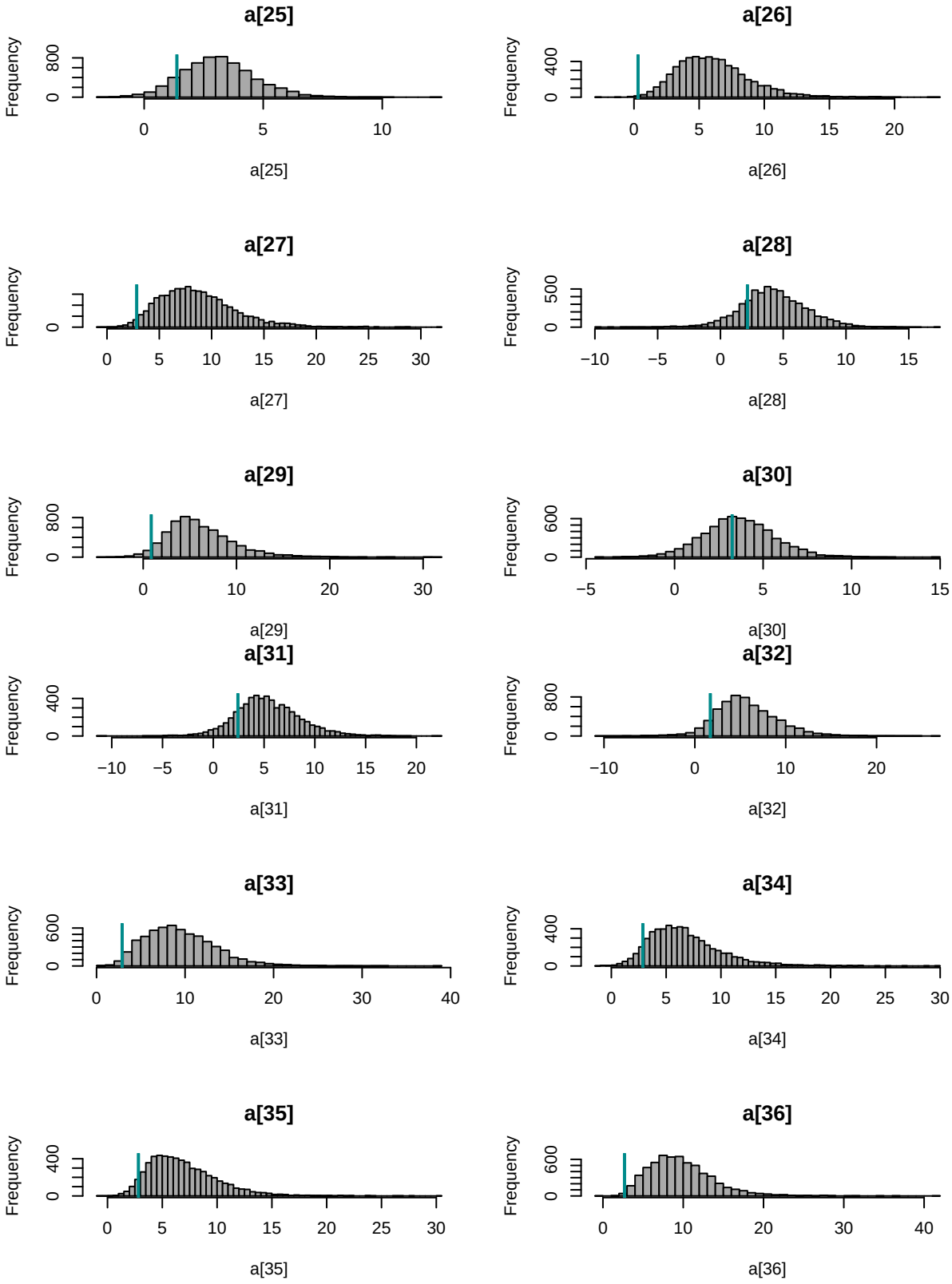
Yes, the trace plots of the MCMC sampler suggest that the sampler did converge. The average of each chain looks roughly the same. Although there is a little wandering within the chian, there is no evidence of divergent chains.

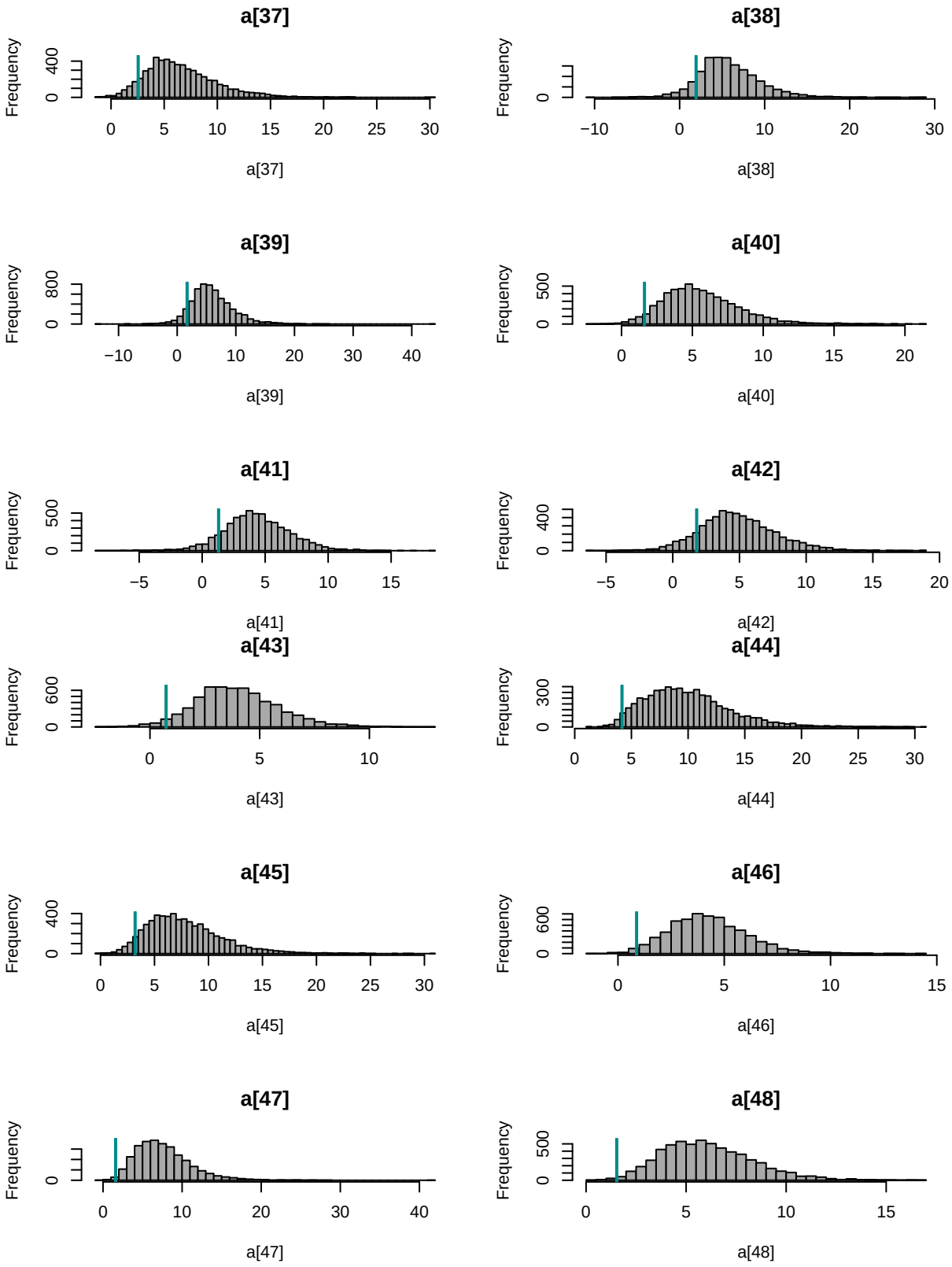
D.

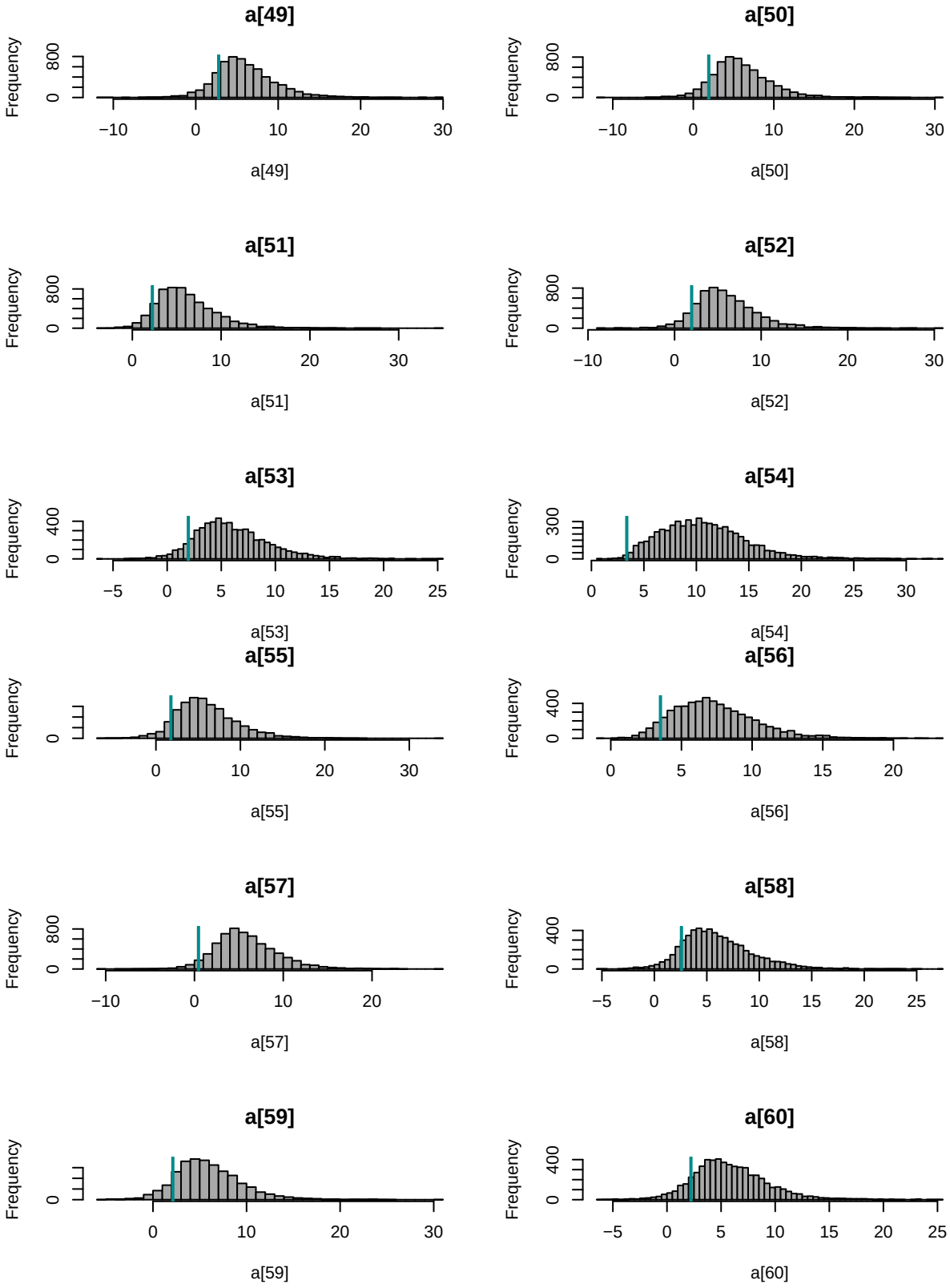
```
# look at posterior distributions
fit.extract<-rstan::extract(fit)

for(ii in 1:60){
  if(ii%%6==1){par(mfrow=c(3:2))}
  pars<-paste('a[',ii,']',sep='')
  hist(fit.extract$a[,ii],main=pars,xlab=pars,col='darkgrey',breaks = 50)
  abline(v=beta_0[ii],lwd=2,col='darkcyan')
}
```


a[1]
Frequency
0
500
0
5
10
15
a[1]
a[2]
Frequency
0
800
−5
0
5
10
15
20
25
30
a[2]
a[3]
Frequency
0
400
0
5
10
15
20
25
30
a[3]
a[4]
Frequency
0
400
−5
0
5
10
15
20
25
a[4]
a[5]
Frequency
0
400
0
5
10
15
20
25
30
a[5]
a[6]
Frequency
0
500
0
5
10
15
20
25
a[6]
a[7]
Frequency
0
400
0
5
10
15
20
25
a[7]
a[8]
Frequency
0
600
−5
0
5
10
a[8]
a[9]
Frequency
0
500
−5
0
5
10
15
a[9]
a[10]
Frequency
0
800
−10
0
10
20
30
40
a[10]
a[11]
Frequency
0
400
−5
0
5
10
15
20
25
a[11]
a[12]
Frequency
0
500
−10
−5
0
5
10
15
a[12]


a[13]
Frequency
0
800
0
10
20
30
a[13]
a[14]
Frequency
0
500
0
5
10
15
20
a[14]
a[15]
Frequency
0
400
0
5
10
15
20
a[15]
a[16]
Frequency
0
800
−10
0
10
20
a[16]
a[17]
Frequency
0
400
0
5
10
15
20
25
a[17]
a[18]
Frequency
0
500
−5
0
5
10
15
a[18]
a[19]
Frequency
0
500
0
5
10
15
20
a[19]
a[20]
Frequency
0
800
0
10
20
30
a[20]
a[21]
Frequency
0
500
−5
0
5
10
15
a[21]
a[22]
Frequency
0
500
0
5
10
15
20
a[22]
a[23]
Frequency
0
500
−15
−10
−5
0
5
10
a[23]
a[24]
Frequency
0
800
0
10
20
30
a[24]


a[25]
Frequency
0
800
0
5
10
a[25]
a[26]
Frequency
0
400
0
5
10
15
20
a[26]
a[27]
Frequency
0
0
5
10
15
20
25
30
a[27]
a[28]
Frequency
0
500
−10
−5
0
5
10
15
a[28]
a[29]
Frequency
0
800
0
10
20
30
a[29]
a[30]
Frequency
0
600
−5
0
5
10
15
a[30]
a[31]
Frequency
0
400
−10
−5
0
5
10
15
20
a[31]
a[32]
Frequency
0
800
−10
0
10
20
a[32]
a[33]
Frequency
0
600
0
10
20
30
40
a[33]
a[34]
Frequency
0
400
0
5
10
15
20
25
30
a[34]
a[35]
Frequency
0
400
0
5
10
15
20
25
30
a[35]
a[36]
Frequency
0
600
0
10
20
30
40
a[36]


a[37]
Frequency
0
400
0
5
10
15
20
25
30
a[37]
a[38]
Frequency
0
−10
0
10
20
30
a[38]
a[39]
Frequency
0
800
−10
0
10
20
30
40
a[39]
a[40]
Frequency
0
500
0
5
10
15
20
a[40]
a[41]
Frequency
0
500
−5
0
5
10
15
a[41]
a[42]
Frequency
0
400
−5
0
5
10
15
20
a[42]
a[43]
Frequency
0
600
0
5
10
a[43]
a[44]
Frequency
0
300
0
5
10
15
20
25
30
a[44]
a[45]
Frequency
0
400
0
5
10
15
20
25
30
a[45]
a[46]
Frequency
0
600
0
5
10
15
a[46]
a[47]
Frequency
0
0
10
20
30
40
a[47]
a[48]
Frequency
0
500
0
5
10
15
a[48]



Yes, all of the true parameters are contained within the posterior distributions from our model. This suggests that we can move forward with fitting the model to the actual train data. You may have noticed that the predictions are slightly biased. This is because we lose some information when converting probabilities to

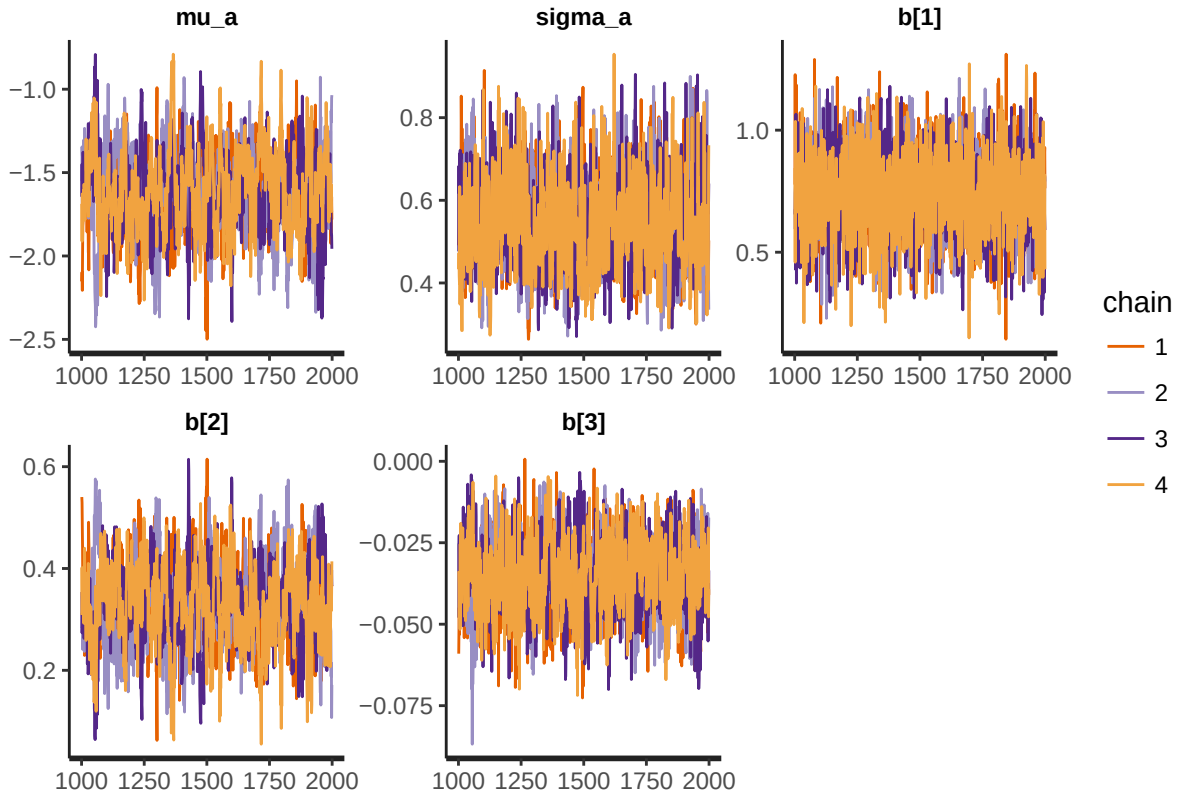
binary variables, however the estimates are still valid draws given the posterior distribution.

E.

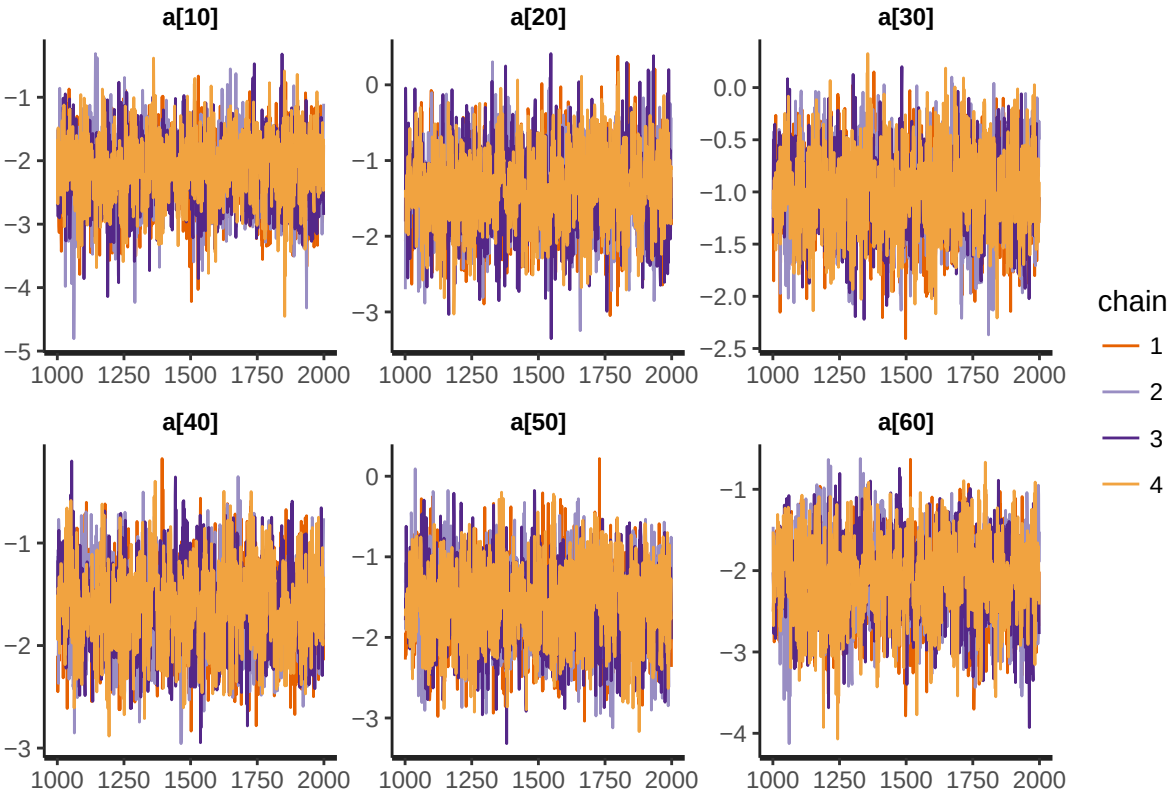
```
# fit the model to real data
     stan_list$contraceptive_use <- train$contraceptive_use
     stan_list$N <- length(stan_list$contraceptive_use)
     fit <- stan(file = "solution_3b.stan",
                 data = stan_list,
                 iter = 2000,
                 chains = 4,
                 seed = 46)
```

F.

```
# look at convergence plots
plot(fit, plotfun="trace", pars=c('mu_a', 'sigma_a', 'b[1]', 'b[2]', 'b[3]'))
```




```
plot(fit, plotfun="trace", pars=c('a[10]', 'a[20]', 'a[30]',
                                  'a[40]', 'a[50]', 'a[60]'))
```

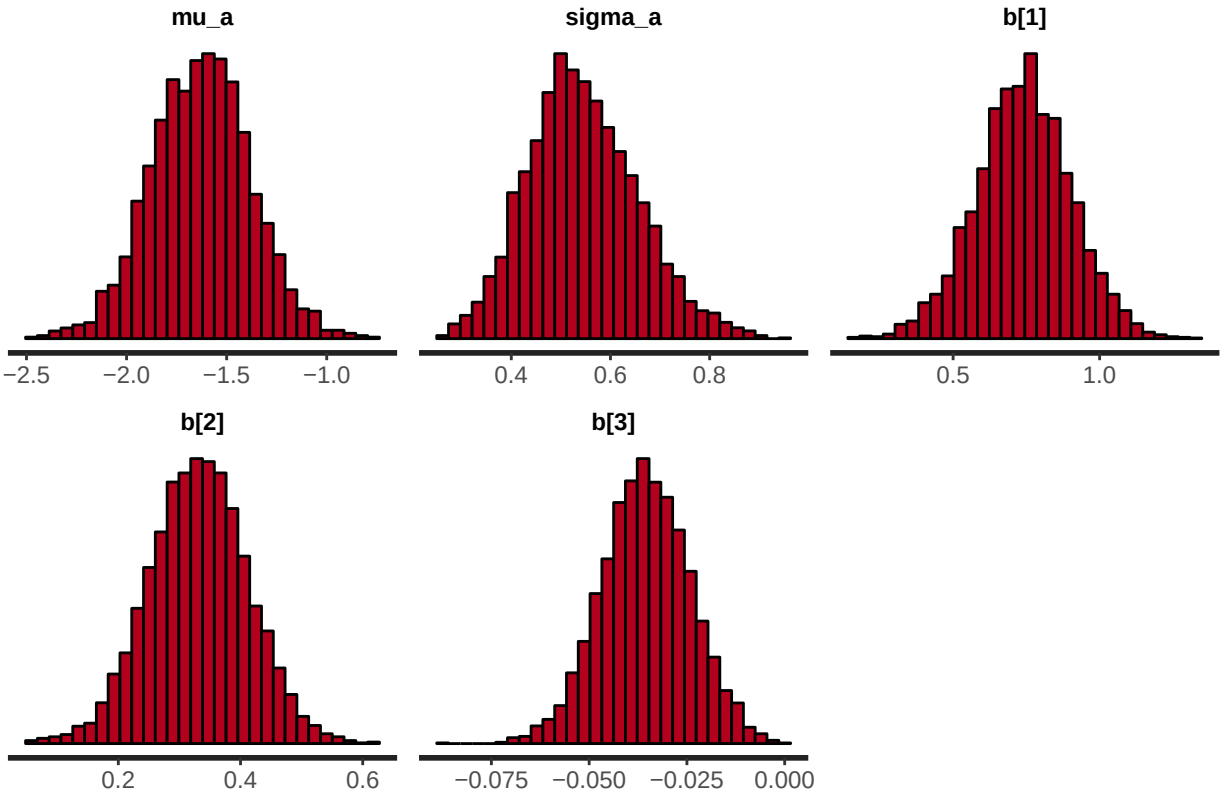


Yes, it looks as if the analysis has converged since none of the chains have appeared to diverge from one another.

G.

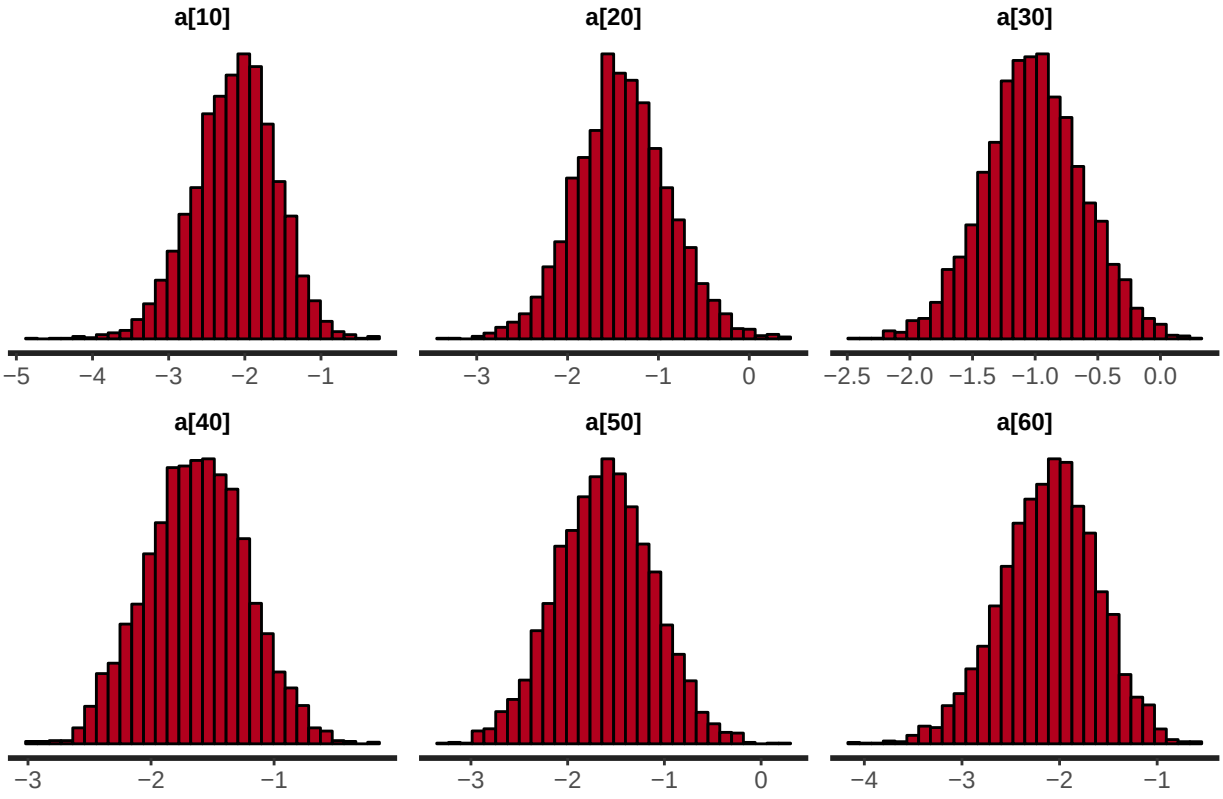
```
# look at posterior distributions
plot(fit, plotfun='hist', pars=c('mu_a', 'sigma_a', 'b[1]', 'b[2]', 'b[3]'))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
plot(fit, plotfun='hist', pars=c('a[10]', 'a[20]', 'a[30]',
                                 'a[40]', 'a[50]', 'a[60]'))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
# extract fit

     sims <- rstan::extract(fit)
     # we use rstan::extract because there is a collision with tidyr
     # find which intercepts have greatest/smallest posterior means

intercepts <- sims$a
     intercepts_pm <- colMeans(intercepts)
     max_index <- which(intercepts_pm == max(intercepts_pm)) #
     min_index <- which(intercepts_pm == min(intercepts_pm))

     paste('District most likely to use contraceptives:',max_index)
```

```
## [1] "District most likely to use contraceptives: 56"
```

```
     paste('District least likely to use contraceptives:',min_index)
```

```
## [1] "District least likely to use contraceptives: 11"
```

It appears that women from district 56 tend to be the most likely to use contraceptives, and women from district 11 tend to be the least likely to use contraceptives.

H.

```
# find posterior means of mu_a and sigma_a
     mu_a <- sims$mu_a
     sigma_a <- sims$sigma_a
     mu_a_pm <- mean(mu_a)
     sigma_a_pm <- mean(sigma_a)
     paste('Posterior mean of mu:',mu_a_pm)
```

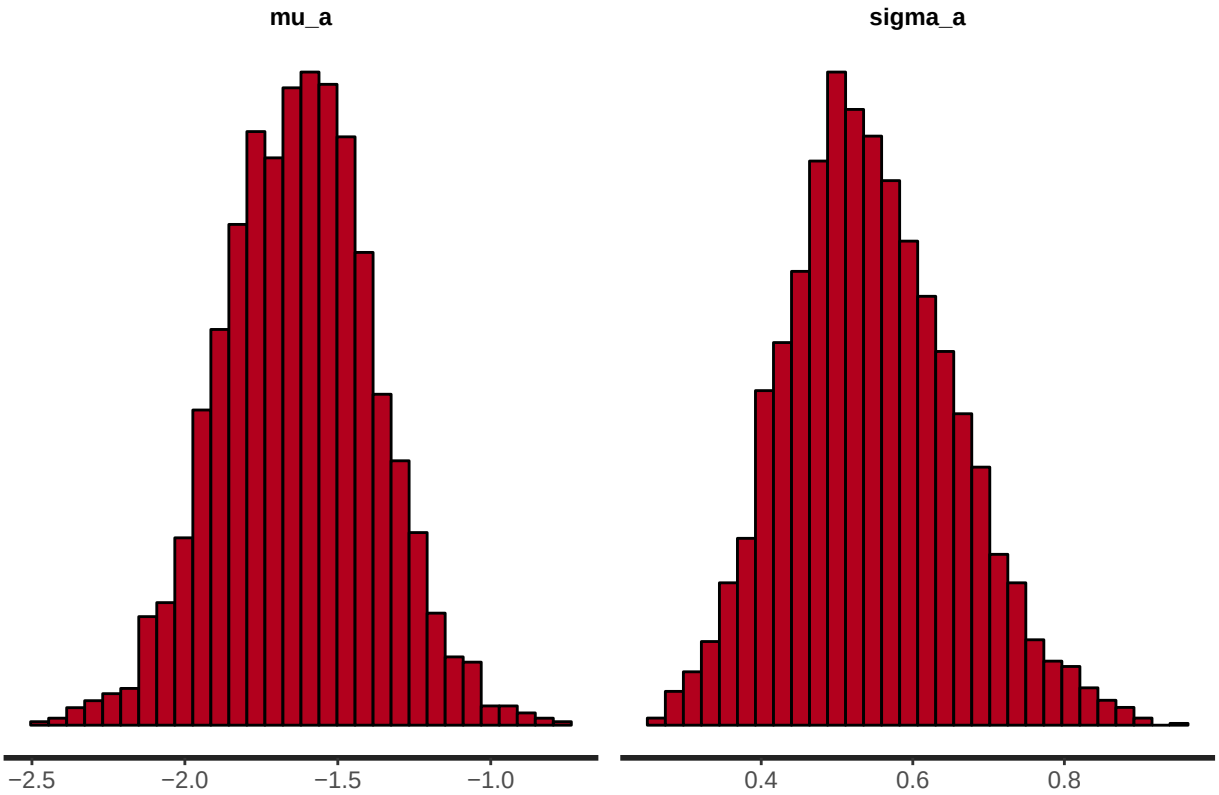
```
## [1] "Posterior mean of mu: -1.62787801690388"
```

```
     paste('Posterior mean of sigma:',sigma_a_pm)
```

```
## [1] "Posterior mean of sigma: 0.544910572692004"
```

```
     plot(fit,plotfun='hist',pars=c('mu_a','sigma_a'))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



{Since the posterior means of μ_{β_0} and σ_{β_0} are not 0,we have evidence in favor of the hypothesis that the rate of contraceptive usage varies by district. If we look at the histograms of these parameters, we can confirm that 0 appears to be far in the tails of the posterior distributions.}

4 Varying-Coefficients Model

In the next model we will fit to the contraceptives data is a varying-coefficients logistic regression model, where the coefficients on living-children, age-mean and urban vary by district:

Prior distribution:

$\beta_{0j} \sim N(\mu_0, \sigma_0)$, with $\mu_0 \sim N(0, 100)$ and $\sigma_0 \sim$ Exponential(0.1)

$\beta_{1j} \sim N(0, \sigma_1)$, with $\sigma_1 \sim$ Exponential(0.1)

$\beta_{2j} \sim N(0, \sigma_2)$, with $\sigma_2 \sim$ Exponential(0.1)

$\beta_{3j} \sim N(0, \sigma_3)$, with $\sigma_3 \sim$ Exponential(0.1)

Model for data:

$Y_{ij} \sim$ Bernoulli(p_{ij})

logit $p_{ij} = \beta_{0j} + \beta_{1j}$urban $+ \beta_{2j}$age-mean $+ \beta_{3j}$living-children

where $i = 1, \ldots, N$ and $j = 1, \ldots, J$ (N is the number of observations in the data, and J is the number of districts).

A. Fit the model to the real data. For each of the three coefficients to the predictors, plot vertical segments corresponding to the 95% central posterior intervals for the coefficient within each district. Thus you should have 60 parallel segments on each graph. If the segments are overlapping on the vertical scale, then the model fit suggests that the coefficient does not differ by district. What do you conclude from these graphs?

B. Use all of the information you've gleaned thus far to build a final Bayesian logistic regression classifier on the train set. Then, use your model to make predictions on the test set. Report your model's classification percentage.

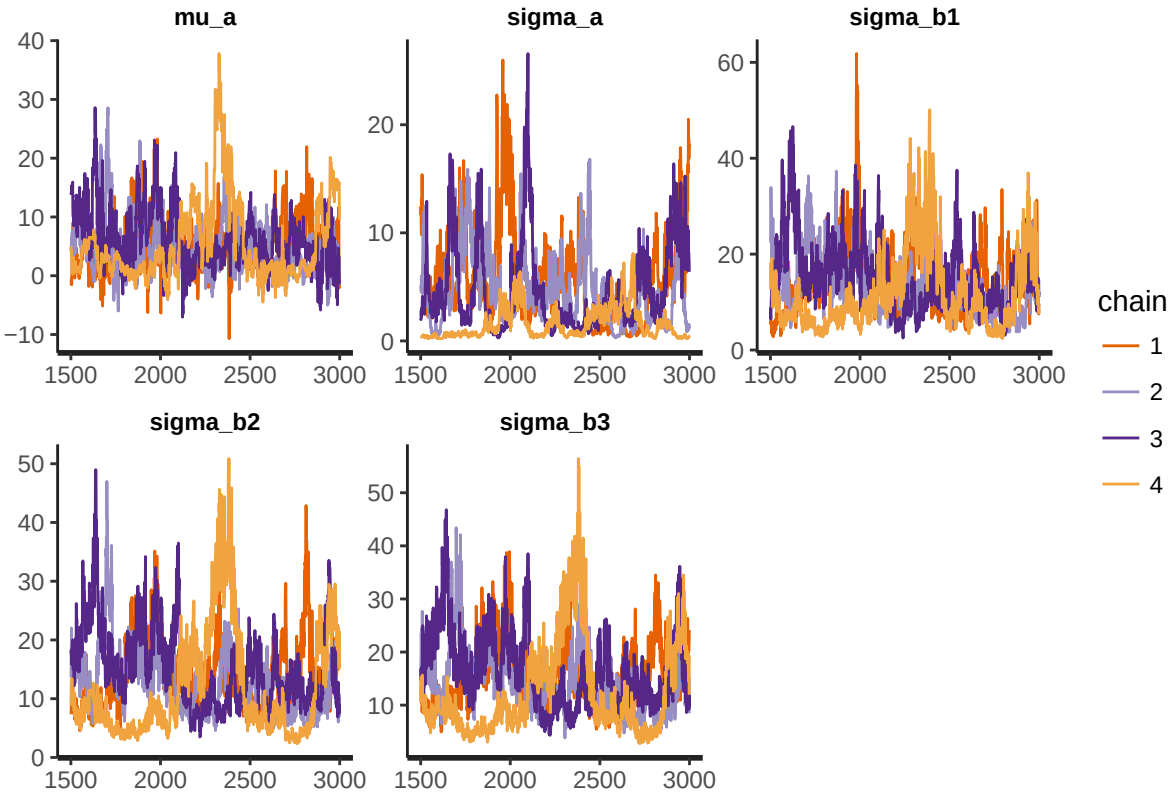
4 Solution

A.

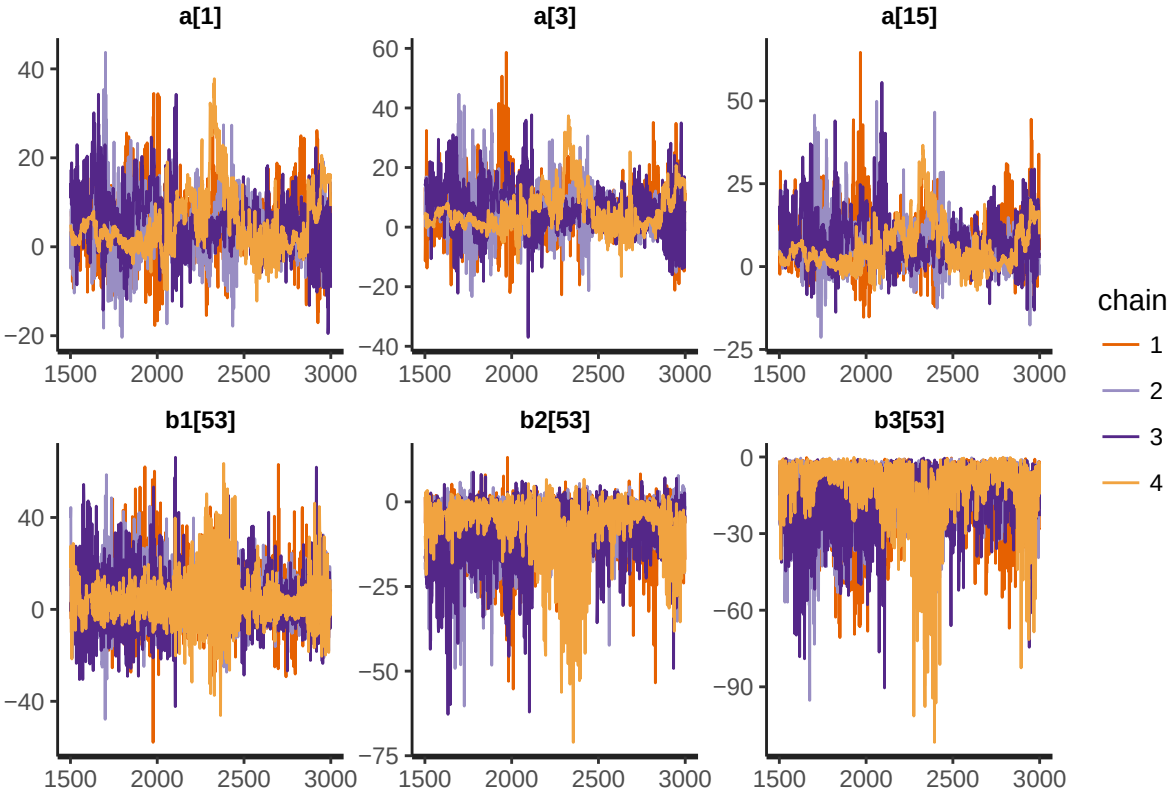
```
# create data list
   stan_list <- c()
   stan_list$district <- train$district
   stan_list$urban <- train$urban
   stan_list$living_children <- train$living.children
   stan_list$age_mean <- train$age_mean
   stan_list$contraceptive_use <- fake_news
   stan_list$N <- length(stan_list$contraceptive_use)
   stan_list$num_districts <- length(unique(stan_list$district))

# fit the model
options(mc.cores = parallel::detectCores())
fit.varcoef <- stan(file='solution_4a.stan',
            data = stan_list,
            iter = 3000,
            chains = 4,
            seed = 46)
```

```
## Warning: There were 19 divergent transitions after warmup. Increasing adapt_delta above 0.8 may help
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup

## Warning: There were 12 transitions after warmup that exceeded the maximum treedepth. Increase max_tr
## http://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded

## Warning: There were 4 chains where the estimated Bayesian Fraction of Missing Information was low. S
## http://mc-stan.org/misc/warnings.html#bfmi-low

## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
# look at convergence plots
plot(fit.varcoef, plotfun="trace",
     pars=c('mu_a', 'sigma_a', 'sigma_b1', 'sigma_b2', 'sigma_b3'))
```




```
plot(fit.varcoef, plotfun="trace",
     pars=c('a[1]', 'a[3]', 'a[15]', 'b1[53]', 'b2[53]','b3[53]'))
```



```r
# R Statistic for Convergence
fit.summary<-summary(fit.varcoef)
fit.summary$summary[1:241,'Rhat']
```

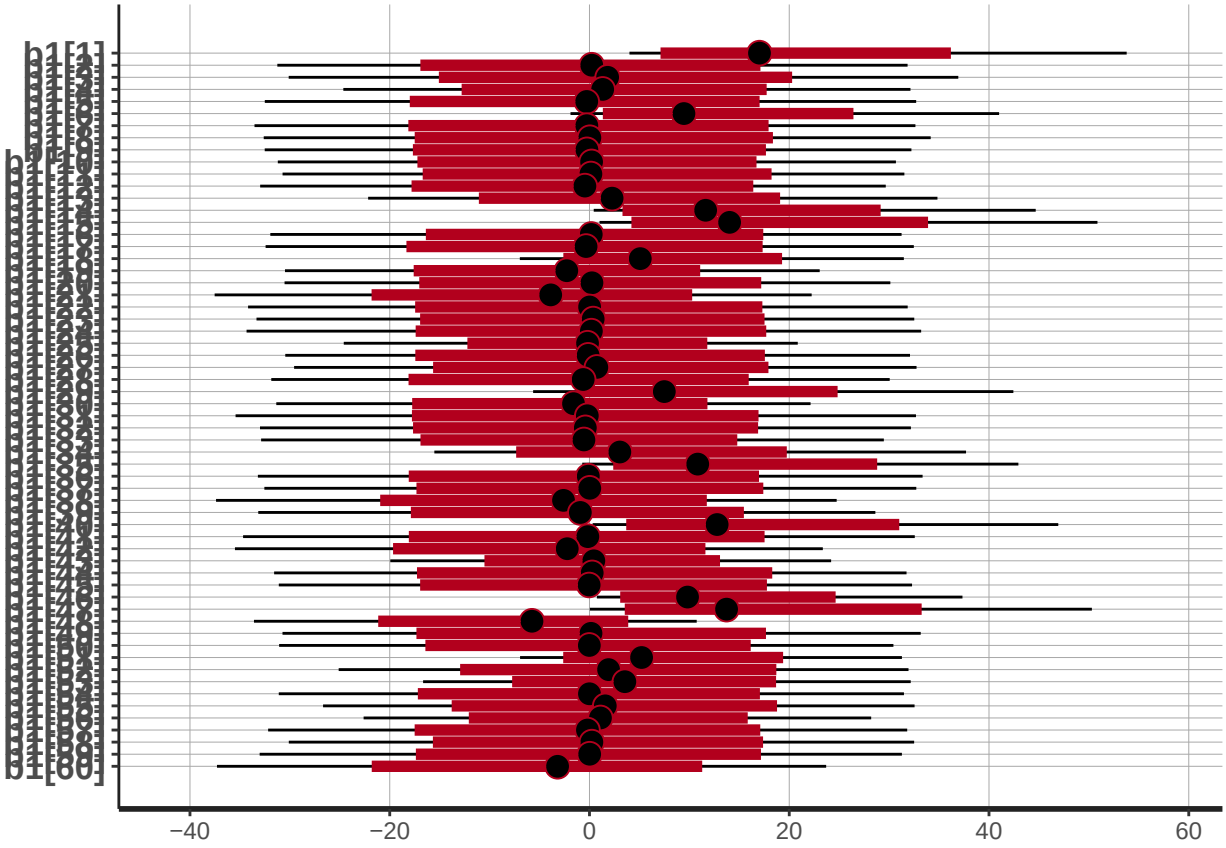
```
##      a[1]      a[2]      a[3]      a[4]      a[5]      a[6]      a[7]
## 1.0254147 1.0218581 1.0195826 1.0253478 1.0203138 1.0279659 1.0234831
##      a[8]      a[9]     a[10]     a[11]     a[12]     a[13]     a[14]
## 1.0289799 1.0208029 1.0275062 1.0249223 1.0256502 1.0266827 1.0260151
##     a[15]     a[16]     a[17]     a[18]     a[19]     a[20]     a[21]
## 1.0399542 1.0189074 1.0245924 1.0278585 1.0243528 1.0188150 1.0216994
##     a[22]     a[23]     a[24]     a[25]     a[26]     a[27]     a[28]
## 1.0269180 1.0258580 1.0225610 1.0291122 1.0239277 1.0376644 1.0239034
##     a[29]     a[30]     a[31]     a[32]     a[33]     a[34]     a[35]
## 1.0261923 1.0259864 1.0241266 1.0205597 1.0261583 1.0291464 1.0435358
##     a[36]     a[37]     a[38]     a[39]     a[40]     a[41]     a[42]
## 1.0221878 1.0239240 1.0184046 1.0205773 1.0310081 1.0223536 1.0239122
##     a[43]     a[44]     a[45]     a[46]     a[47]     a[48]     a[49]
## 1.0244412 1.0270227 1.0261376 1.0806378 1.0291780 1.0281784 1.0255583
##     a[50]     a[51]     a[52]     a[53]     a[54]     a[55]     a[56]
## 1.0247555 1.0283823 1.0184579 1.0258405 1.0259904 1.0204588 1.0602432
##     a[57]     a[58]     a[59]     a[60]     b1[1]     b1[2]     b1[3]
## 1.0218493 1.0199624 1.0219530 1.0207864 1.0522825 0.9998618 1.0012076
##     b1[4]     b1[5]     b1[6]     b1[7]     b1[8]     b1[9]    b1[10]
## 1.0002820 1.0006260 1.0215239 1.0002292 1.0002733 0.9995900 0.9999434
##    b1[11]    b1[12]    b1[13]    b1[14]    b1[15]    b1[16]    b1[17]
## 0.9999243 0.9999636 1.0015230 1.0376610 1.0327581 1.0002310 0.9997587
##    b1[18]    b1[19]    b1[20]    b1[21]    b1[22]    b1[23]    b1[24]
## 1.0098603 1.0008801 0.9998889 1.0035015 0.9998742 1.0011552 0.9999488
##    b1[25]    b1[26]    b1[27]    b1[28]    b1[29]    b1[30]    b1[31]
## 1.0000121 0.9998721 0.9995602 0.9996559 1.0139136 1.0004049 1.0000426
##    b1[32]    b1[33]    b1[34]    b1[35]    b1[36]    b1[37]    b1[38]
## 1.0003498 0.9998389 1.0028758 1.0251570 0.9999331 0.9996643 1.0015323
##    b1[39]    b1[40]    b1[41]    b1[42]    b1[43]    b1[44]    b1[45]
## 0.9997907 1.0289233 0.9997370 1.0019190 1.0015857 0.9998984 0.9999960
##    b1[46]    b1[47]    b1[48]    b1[49]    b1[50]    b1[51]    b1[52]
## 1.0368370 1.0243239 1.0148329 0.9996479 0.9997121 1.0076503 1.0004971
##    b1[53]    b1[54]    b1[55]    b1[56]    b1[57]    b1[58]    b1[59]
## 1.0039705 1.0002245 1.0001229 1.0003225 0.9999555 1.0004264 1.0000150
##    b1[60]     b2[1]     b2[2]     b2[3]     b2[4]     b2[5]     b2[6]
## 1.0010195 1.0551224 1.0439085 1.0053856 1.0393277 1.0001433 1.0491840
##     b2[7]     b2[8]     b2[9]    b2[10]    b2[11]    b2[12]    b2[13]
## 1.0098220 1.0345378 1.0429119 1.0048886 1.0257699 1.0470761 1.0477145
##    b2[14]    b2[15]    b2[16]    b2[17]    b2[18]    b2[19]    b2[20]
## 1.0486208 1.0617880 1.0015717 1.0445066 1.0501135 1.0425495 1.0334839
##    b2[21]    b2[22]    b2[23]    b2[24]    b2[25]    b2[26]    b2[27]
## 1.0388620 1.0455706 1.0409339 1.0019902 1.0515951 1.0421335 1.0053622
##    b2[28]    b2[29]    b2[30]    b2[31]    b2[32]    b2[33]    b2[34]
## 1.0458059 1.0464070 1.0489854 1.0453503 1.0371032 1.0259252 1.0412914
##    b2[35]    b2[36]    b2[37]    b2[38]    b2[39]    b2[40]    b2[41]
## 1.0534794 1.0063944 1.0013310 1.0327956 1.0029398 1.0565415 1.0378159
##    b2[42]    b2[43]    b2[44]    b2[45]    b2[46]    b2[47]    b2[48]
## 1.0077112 1.0248840 1.0105590 1.0448900 1.0735581 1.0129953 1.0338556
##    b2[49]    b2[50]    b2[51]    b2[52]    b2[53]    b2[54]    b2[55]
## 1.0000872 1.0048552 1.0519780 1.0337208 1.0268248 1.0045535 1.0019512
```

```
##    b2[56]    b2[57]    b2[58]    b2[59]    b2[60]     b3[1]     b3[2]
## 1.0625180 1.0367519 1.0426907 1.0068361 1.0316042 1.0403363 1.0372401
##     b3[3]     b3[4]     b3[5]     b3[6]     b3[7]     b3[8]     b3[9]
## 1.0065296 1.0313103 1.0423960 1.0403599 1.0445817 1.0237146 1.0358854
##    b3[10]    b3[11]    b3[12]    b3[13]    b3[14]    b3[15]    b3[16]
## 1.0359356 1.0301066 1.0365889 1.0385000 1.0403632 1.0263103 1.0409447
##    b3[17]    b3[18]    b3[19]    b3[20]    b3[21]    b3[22]    b3[23]
## 1.0423902 1.0445269 1.0443417 1.0263773 1.0368418 1.0465147 1.0143014
##    b3[24]    b3[25]    b3[26]    b3[27]    b3[28]    b3[29]    b3[30]
## 1.0352604 1.0519017 1.0304307 1.0404465 1.0344383 1.0349131 1.0457728
##    b3[31]    b3[32]    b3[33]    b3[34]    b3[35]    b3[36]    b3[37]
## 1.0405900 1.0243278 1.0365355 1.0297937 1.0352301 1.0379135 1.0350317
##    b3[38]    b3[39]    b3[40]    b3[41]    b3[42]    b3[43]    b3[44]
## 1.0246686 1.0397223 1.0389482 1.0344438 1.0399165 1.0272524 1.0320023
##    b3[45]    b3[46]    b3[47]    b3[48]    b3[49]    b3[50]    b3[51]
## 1.0473742 1.0594642 1.0173811 1.0606067 1.0214338 1.0262687 1.0349546
##    b3[52]    b3[53]    b3[54]    b3[55]    b3[56]    b3[57]    b3[58]
## 1.0352952 1.0318392 1.0361196 1.0392915 1.0322128 1.0285960 1.0363513
##    b3[59]    b3[60]      mu_a
## 1.0317226 1.0359278 1.0494345
```

```r
plot(fit.varcoef,pars=c('b1'),main='Urban : beta1')
```

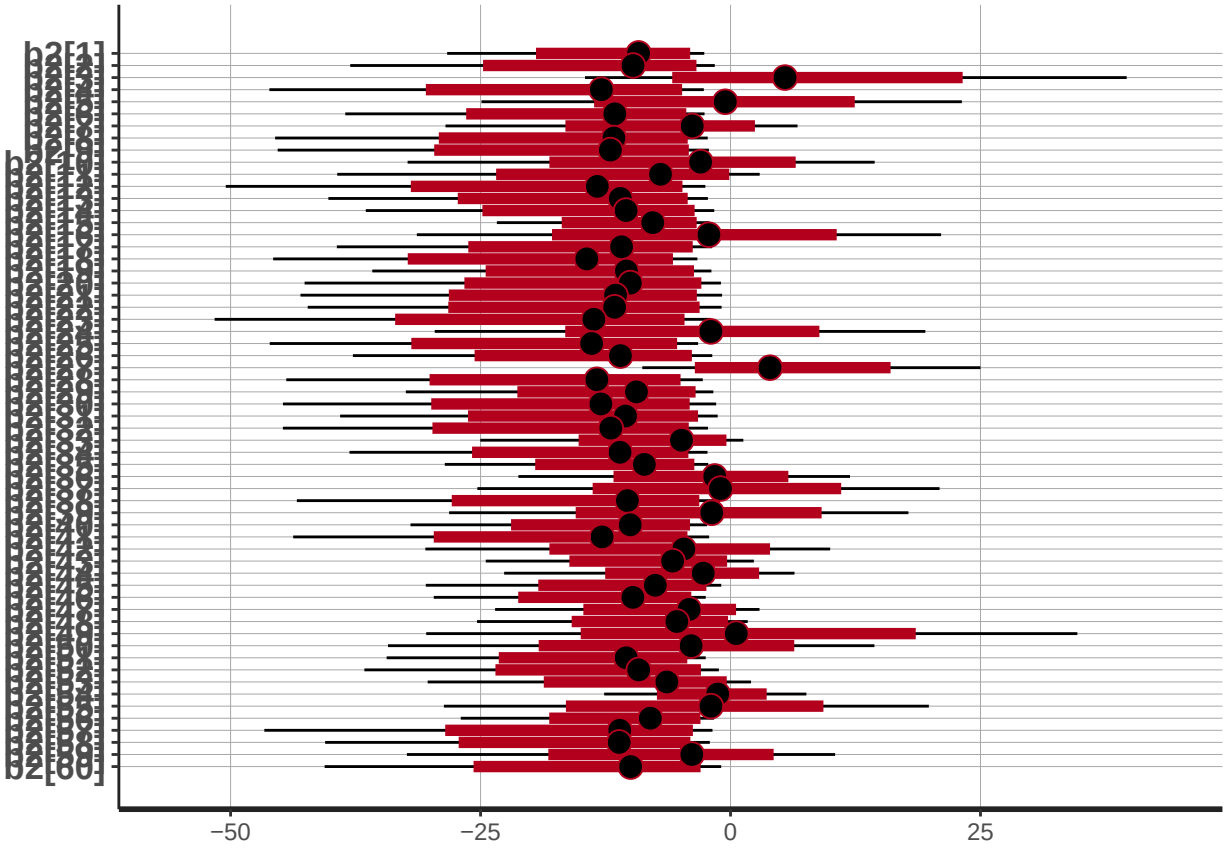
```
## ci_level: 0.8 (80% intervals)
```

```
## outer_level: 0.95 (95% intervals)
```



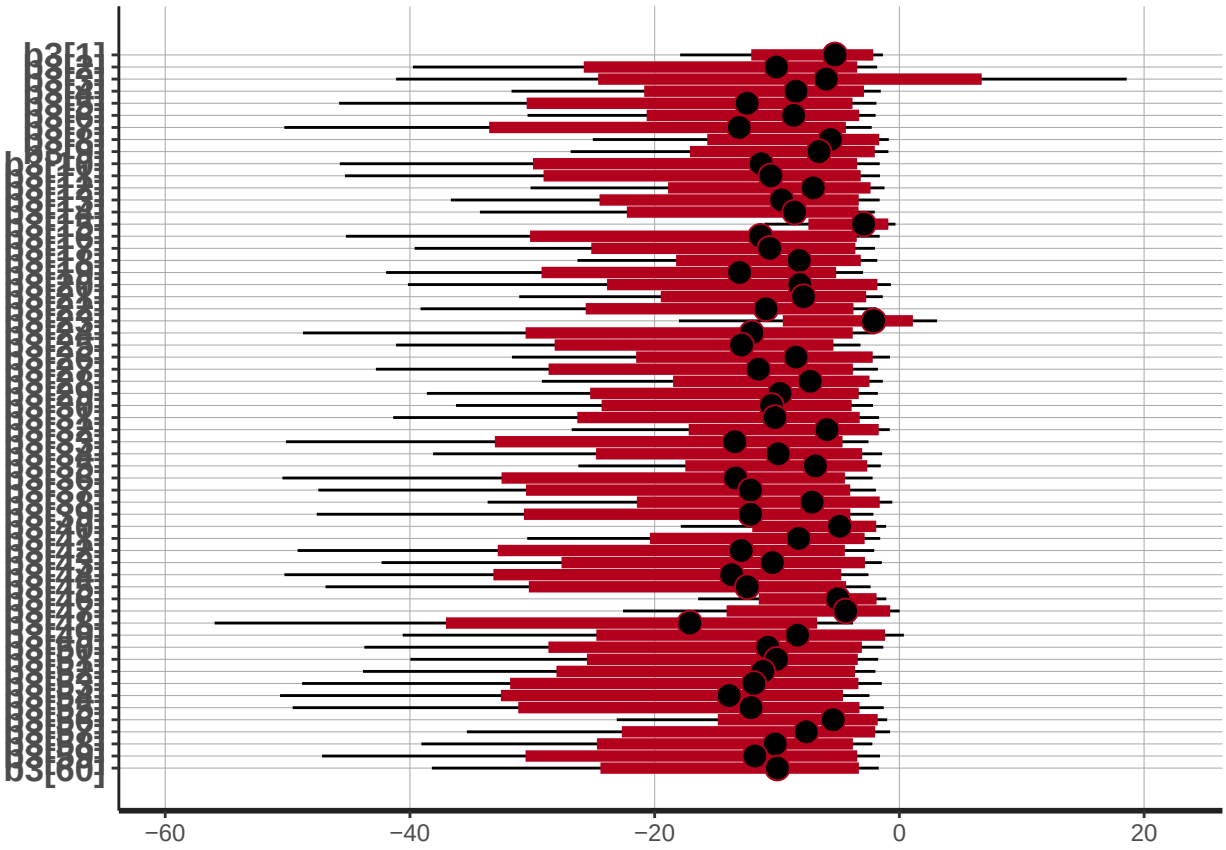

```r
plot(fit.varcoef,pars=c('b2'),main='Age-Mean : beta2')
```

```
## ci_level: 0.8 (80% intervals)
## outer_level: 0.95 (95% intervals)
```




```
plot(fit.varcoef,pars=c('b3'),main='Living-Children : beta3')
```

```
## ci_level: 0.8 (80% intervals)
## outer_level: 0.95 (95% intervals)
```



We should be a little wary of the convergence here. Although they roughly converge to the same area, the trains wander a bit. Looing at the R statistic for convergence, we can see that the values are only slightly larger than 1. (You do not have to calculate the R statistic for full credit.)

The inner interval is 80% and the outer interval is 95%. It looks like there is definitely variation by district for the coefficient on urban and age-mean. It doesn't look like there is much variation by district for the coefficient on living-children.

B.

```
# make predictions using posterior means of model 2
   # extract posterior means
   sims <- rstan::extract(fit.varcoef)
   a_sims <- sims$a
   a_pm <- colMeans(a_sims)
   b1_sims <- sims$b1
   b1_pm <- colMeans(b1_sims)
   b2_sims <- sims$b2
   b2_pm <- colMeans(b2_sims)
   b3_sims <- sims$b3
   b3_pm <- colMeans(b3_sims)
   predict_pm <- function(test) {
     preds <- rep(NA, nrow(test))
     for (i in 1:length(preds)) {
       # get data
       district <- as.numeric(as.character(test[i,1]))
       urban <- as.numeric(as.character(test[i,2]))
       living_children <- as.numeric(as.character(test[i,3]))
```

```
      age_mean <- as.numeric(as.character(test[i,4]))
      # get parameters
      beta_0 <- a_pm[district]
      beta_1 <- b1_pm[district]
      beta_2 <- b2_pm[district]
      beta_3 <- b3_pm[district]
      # convert to probability
      p_hat <- beta_0 + (beta_1 * urban) + (beta_2 * living_children) + (beta_3 * age_mean)
      pred <- exp(p_hat) / (1 + exp(p_hat))
      preds[i] <- pred
}
    return(round(preds))
  }
  preds <- predict_pm(test)
  cat("classification percentage on test set:", mean(preds == test$contraceptive_use))
```

```
## classification percentage on test set: 0.4446743
```

5 "Bayesball" (AC 209b Students Only)

In Major League Baseball (MLB), each team plays 162 games in the regular season. Under the Bradley-Terry model, each team i is assumed to have some underlying talent parameter π_i. The model states that the probability that team i defeats opponent j in any game is:

$$\Pr(\text{team } i \text{ defeats team } j) = \frac{\pi_i}{\pi_i + \pi_j}.$$

where $i, j \in (1, 2, ...30)$, since there are 30 teams in MLB. The parameter π_i is team i's "strength" parameter, and is required to be positive.

If we let V_{ij} be the number of times in a season that team i defeats team j, and n_{ij} to be the number of games between them, an entire season of MLB can be described with the following density:

$$p(V \mid \pi) = \prod_{i=1}^{N-1} \prod_{j=i+1}^{N} \binom{n_{ij}}{V_{ij}} (\frac{\pi_i}{\pi_i + \pi_j})^{V_{ij}} (\frac{\pi_j}{\pi_i + \pi_j})^{V_{ji}}$$

Team i's victories against team j follows a binomial distribution governed by the Bradley-Terry probability with the given strength parameters.

Rather than work with the π_i, we will transform the model by letting $\lambda_i = \log \pi_i$ for all i. Thus the probability i defeats j is

$$\Pr(\text{team } i \text{ defeats team } j) = \frac{\exp(\lambda_i)}{\exp(\lambda_i) + \exp(\lambda_j)}.$$

The advantage of this parameterization is that the λ_i are unconstrained real-valued parameters.

We now carry out a Bayesian analysis of the Bradley-Terry model. We will assume a hierarchical normal prior distribution on the λ_i, that is

$$\lambda_i \sim N(0, \sigma).$$

for all $i = 1, \ldots, N$. We will also assume that the standard deviation σ has a uniform prior distribution,

$$\sigma \sim \text{Uniform}(0, 50)$$

with a maximum value of 50.

Thus the full model is:

Prior distribution:

$\lambda_i \mid \sigma \sim N(0, \sigma)$, with $\sigma \sim \text{Uniform}(0, 50)$

Model for data:

$V_{ij} \mid \lambda_i, \lambda_j, n_{ij} \sim \text{Binomial}\left(n_{ij}, \frac{\exp(\lambda_i)}{\exp(\lambda_i)+\exp(\lambda_j)}\right)$

A. Why does this prior distribution on the λ_i and σ make sense? Briefly explain.

B. Implement the model in Stan.

C. Report the posterior means for each team's exponentiated strength parameters (that is, $\exp(\lambda_i)$).

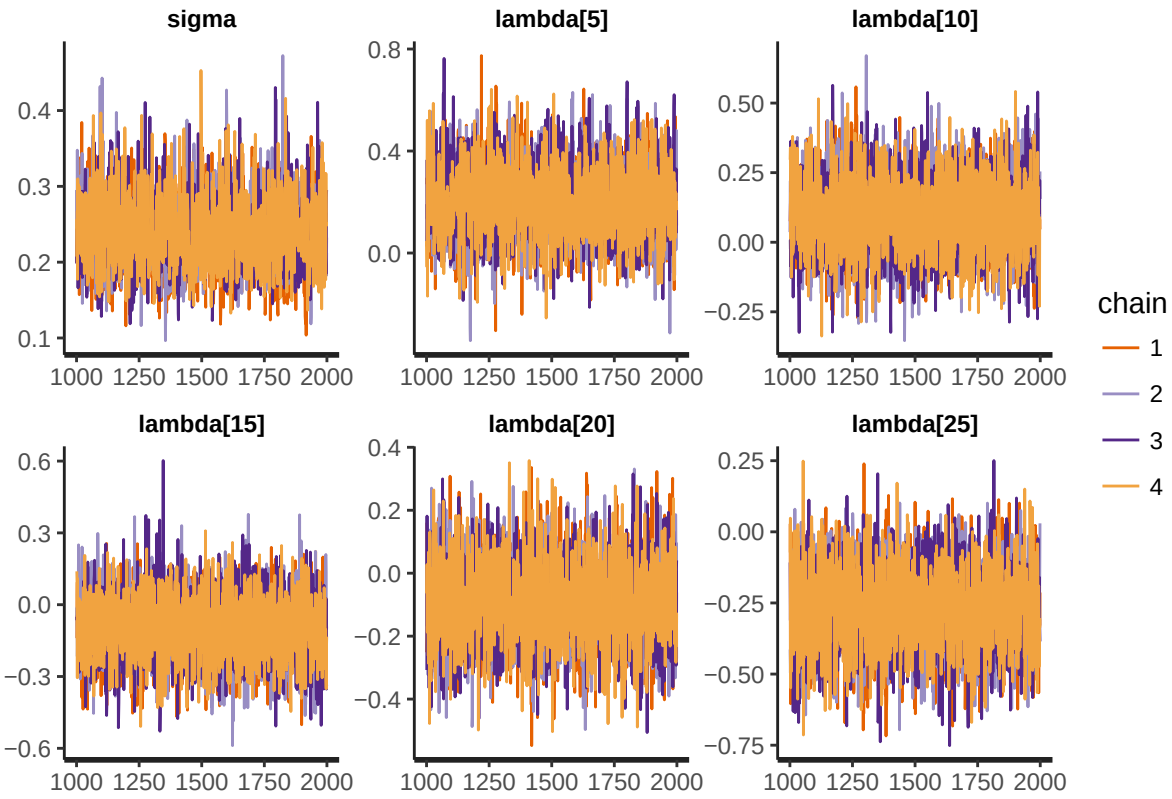
D. Using the posterior predictive distribution for the strengths of the Dodgers and Astros, simulate 1000 recreations of the 2017 World Series. That is, simulate 1000 separate series between the teams, where the series ends when either team gets to 4 wins. Based on your simulation, what was the probability that the Astros would have won the World Series last year?

5 Solution

A. It makes sense that the lambdas are drawn from a distribution centered around 0, as this is the parameter of a truly average team. It makes sense to use a uniform prior on sigma because have no data from prior seasons; however, we can constrain the parameter space to $[0, 50]$ because we don't expect the standard deviation to be outside of this range.

B.

```
games_2017 <- read.table('Data/Bayesball.txt', sep=",")
    # subset to needed data
    data_2017 <- games_2017[,c(7, 4, 11, 10)]
    names(data_2017) <- c("Home","Away","Home_Score","Away_Score")
    data_2017$score_diff <- data_2017$Home_Score - data_2017$Away_Score
    data_2017$home_win <- ifelse(data_2017$score_diff > 0, 1, 0)
    # add team IDs (for Stan)
    ids <- sort(unique(data_2017$Home))
    new_ids <- 1:length(ids)
    data_2017$Home_ID <- mapvalues(data_2017$Home, from=ids, to=new_ids)
    data_2017$Away_ID <- mapvalues(data_2017$Away, from=ids, to=new_ids)
    # create list (for Stan)
    ls_2017 <- c()
    ls_2017$team1 <- as.numeric(data_2017$Home_ID)
    ls_2017$team2 <- as.numeric(data_2017$Away_ID)
    ls_2017$team1_win <- data_2017$home_win
    ls_2017$nteams <- length(unique(data_2017$Home_ID))
    ls_2017$ngames <- nrow(data_2017)

    # fit the model
   options(mc.cores = parallel::detectCores())
   fit.bb <- stan(file='solution_5b.stan',
                  data = ls_2017,
                  iter = 2000,
                  chains = 4)

   # check chains
  plot(fit.bb, plotfun="trace",
     pars=c('sigma', 'lambda[5]', 'lambda[10]', 'lambda[15]', 'lambda[20]','lambda[25]'))
```



C.

```
# extract parameters
   sims <- rstan::extract(fit.bb)
   pi_sims <- sims$exp_lambda
   pi <- colMeans(pi_sims)
   sigma_sims <- sims$sigma
   sigma <- mean(sigma_sims)

# create final dataframe
  results <- data.frame(ids, new_ids, pi)
  names(results) <- c('Team_Id', 'Team_Num', 'Talent_Level')

print(results)
```

```
##    Team_Id Team_Num Talent_Level
## 1      ANA        1    1.0087122
## 2      ARI        2    1.2242332
## 3      ATL        3    0.8482862
## 4      BAL        4    0.9385835
## 5      BOS        5    1.2381919
## 6      CHA        6    0.8190983
## 7      CHN        7    1.1832089
## 8      CIN        8    0.8197699
## 9      CLE        9    1.4178969
## 10     COL       10    1.1120367
## 11     DET       11    0.7778525
## 12     HOU       12    1.4045141
## 13     KCA       13    1.0005714
```

```
## 14     LAN       14    1.4497591
## 15     MIA       15    0.9154590
## 16     MIL       16    1.0801986
## 17     MIN       17    1.0814792
## 18     NYA       18    1.2011318
## 19     NYN       19    0.8212110
## 20     OAK       20    0.9240164
## 21     PHI       21    0.7741400
## 22     PIT       22    0.9073038
## 23     SDN       23    0.8609208
## 24     SEA       24    0.9738552
## 25     SFN       25    0.7707441
## 26     SLN       26    1.0293915
## 27     TBA       27    1.0156382
## 28     TEX       28    0.9711192
## 29     TOR       29    0.9513069
## 30     WAS       30    1.2485916
```

D.

```
# simulate World Series
   # extract posterior sims for Astros and Dodgers talent levels
   Astros_talent_sims <- pi_sims[,12]
   Dodgers_talent_sims <- pi_sims[,14]

      # sample 1000 talents
   Astros_talent_samples <- sample(x=Astros_talent_sims, size=1000, replace=F)
   Dodgers_talent_samples <- sample(x=Dodgers_talent_sims, size=1000, replace=F)
  p_astros<-mapply(function(x,y){return(x/(x+y))},
                   Astros_talent_samples,Dodgers_talent_samples)
   # simulate World series
   Astros_win_series <- rep(NA, 1000)
for (ii in 1:1000) {

  p <- p_astros[ii]
  Astros_wins <- 0
  Dodgers_wins <- 0

  # keep playing until one team has wone 4 games
  while ((Astros_wins < 4) && (Dodgers_wins < 4)) {
    Astros_win <- rbinom(n=1, size=1, prob=p)
    if (Astros_win == 1) {
      Astros_wins <- Astros_wins + 1
    } else {
      Dodgers_wins <- Dodgers_wins + 1
    }
  }

  #Indicate when the Astros win
  if (Astros_wins == 4) {
    Astros_win_series[ii] <- 1
  } else {
    Astros_win_series[ii] <- 0
  }
```

```
}
cat("Probability Astros win World Series: ", mean(Astros_win_series))
```

```
## Probability Astros win World Series:  0.481
```

[This part was not required by the homework] The solution can also be simulated using a negative binomial distribution. The negative binomial distribution is a discrete distribution of the number of success in a sequence of bernoulli trials before a failure occurs. The probability mass function is:

$$f(k|r,p) = \binom{k+r-1}{k} p^k (1-p)^r$$

where k is the number of successes, r is the number of failures and p is a probability of success. In the Bayesball case, we can think of the Dodgers winning four games as a failure (r=4) . That is we want to draw the amount of games the Astros have won before the Dogders win 4. We then can count the number of times the Astros ahave won 4 or more games before the Dodgers win 4 to see how many out of the 1000 games the astros have won.

```
# simulate World Series
   # extract posterior sims for Astros and Dodgers talent levels
   Astros_talent_sims <- pi_sims[,12]
   Dodgers_talent_sims <- pi_sims[,14]

      # sample 1000 talents
   Astros_talent_samples <- sample(x=Astros_talent_sims, size=1000, replace=F)
   Dodgers_talent_samples <- sample(x=Dodgers_talent_sims, size=1000, replace=F)
  p_astros<-mapply(function(x,y){return(x/(x+y))},
                   Astros_talent_samples,Dodgers_talent_samples)

    #Draw from the negative binoimal distribtuion 1000 times:
    # (R parameterizes the neg binomial distribution differently, p corresponds to the probability of a
  astro_wins<-rnbinom(1000,size=4,p=(1-p_astros))


  cat("Probability Astros win World Series: ", sum(astro_wins>=4)/1000)
```

```
## Probability Astros win World Series:  0.493
```