STAT 3303 Final

Jacob Depinet.23

2024-04-20

Written Report

Context

K9C9 is a growing strand of influenza in 10 countries across the world. In order to test if K9C9 is present within an individual an inexpensive test called EZK was developed as well as a perfect diagnostic test. Within each of the 10 countries, 100 subjects were randomly selected and were administered both tests. In order to estimate the probability of a subject having the virus, a Bayesian hierarchical model using the EZK results as a parametric function of the perfect test.

Modeling

In this context, a Random Effects Model (REM) was most appropriate. REMs are types of hierarchical models where the unknown states of random variables depend on an "overall" parameter that is common. Given that the outcome of perfect diagnostic test is conditioned on α_c and β_c, and α_c and β_c are also random variables that can be assumed to have some overarching mechanism that determines their distribution, a REM was the clear choice. Therefore the testing scenario can be modeled as follows:

$$Y_{sc}|\alpha_c, \beta_c \sim Bernoulli(\theta_c)$$

$$logit(\theta_c) = \alpha_c + \beta_c x_{sc}$$

$$p(\mathbf{y}|\alpha_c, \beta_c) = \prod_{c=1}^{N_c} \prod_{s=1}^{N_s} p(y_{sc}|\alpha_c, \beta_c)$$

$$p(\alpha_c, \beta_c | \mu_\alpha, \mu_\beta, \tau_\alpha, \tau_\beta) = \prod_{c=1}^{N_c} p(\alpha_c | \mu_\alpha, \tau_\alpha) p(\beta_c | \mu_\beta, \tau_\beta)$$

Where

$$\alpha_c = (\alpha_1, ..., \alpha_{N_c}) \quad \text{and} \quad \beta_c = (\beta_1, ..., \beta_{Nc})$$

$$\mu_\alpha \sim Normal(-0.5, 0.5)$$

$$\mu_\beta \sim Normal(1.5, 0.5)$$

$$\tau_\alpha \sim Uniform(0, 10)$$

$$\tau_\beta \sim Uniform(0, 10)$$

$$\alpha_c \sim Normal(\mu_\alpha, \tau_\alpha)$$

$$\beta_c \sim Normal(\mu_\beta, \tau_\beta)$$

Where y_{sc} is the outcome of the perfect diagnostic test for subject s in country c, x_{sc} is the result of the EZK test (0 for negative, 1 for positive), α_c is false positive rate of the EZK test for country c, β_c is the difference between the false positive rate of country c and the true positive rate of country c, θ_c is the true positive rate of country c, N_c is the number of countries which is 10, and N_s is the number of subject which is 10. μ_α is the expected average false positive rate governing all countries and τ_α is the precision of the false positive rate governing all countries. μ_β is the expected difference between false positive rate and true positive rate governing all countries and τ_β is the precision of this difference governing all countries.

When examining the prior distributions, μ_α and β_α are normally distributed. I have the means where I would imagine the false positive rate (α) and where I would expect the difference between the false positive rate and the true positive rate (β). I am not a subject area expert, so these are mere guesses. Their respective variances are chosen to be diffuse in this context, where the domain of θ_c ranges from 0 to 1. With τ_α and τ_β, a very diffuse distribution is given as I am not a subject matter expert.

The assumption is made that all the α values are independent of one another and all the β values are independent of one another; the α values are conditionally dependent on μ_α and τ_α; and the β values are conditionally dependent on μ_β and τ_β

Directed Acyclic Graph (DAG)

Directed Acyclic Graphs (DAGs) are a useful visualization technique to show the conditional dependence relationships between variables. To visualize the relationships in the hierarchical model, the following DAG is provided on the next page.

Model Set-Up and Computation

Using Just Another Gibbs Sampler (JAGS), the model was ran. α_c and β_c were initialized to 0, μ_α to 0, μ_β to 1, and τ_α and τ_β to 0.1. This was done on 15000 iterations with 5000 burn in iterations. 5000 samples were used as adaptation samples. 2 Markov chains were used on each α_c and β_c.

Trace plots are provided in the appendix. Upon there examination, they are mixing well. There appears to be a constant variance and constant mean within each trace plot for α_c and β_c. They appear to pass the fat marker test. Of the two chains that were run for each prior, they converge to the same point which is also an indicator of the Markov chain working well. The density plots also appear appropriate as they appear normally distributed.

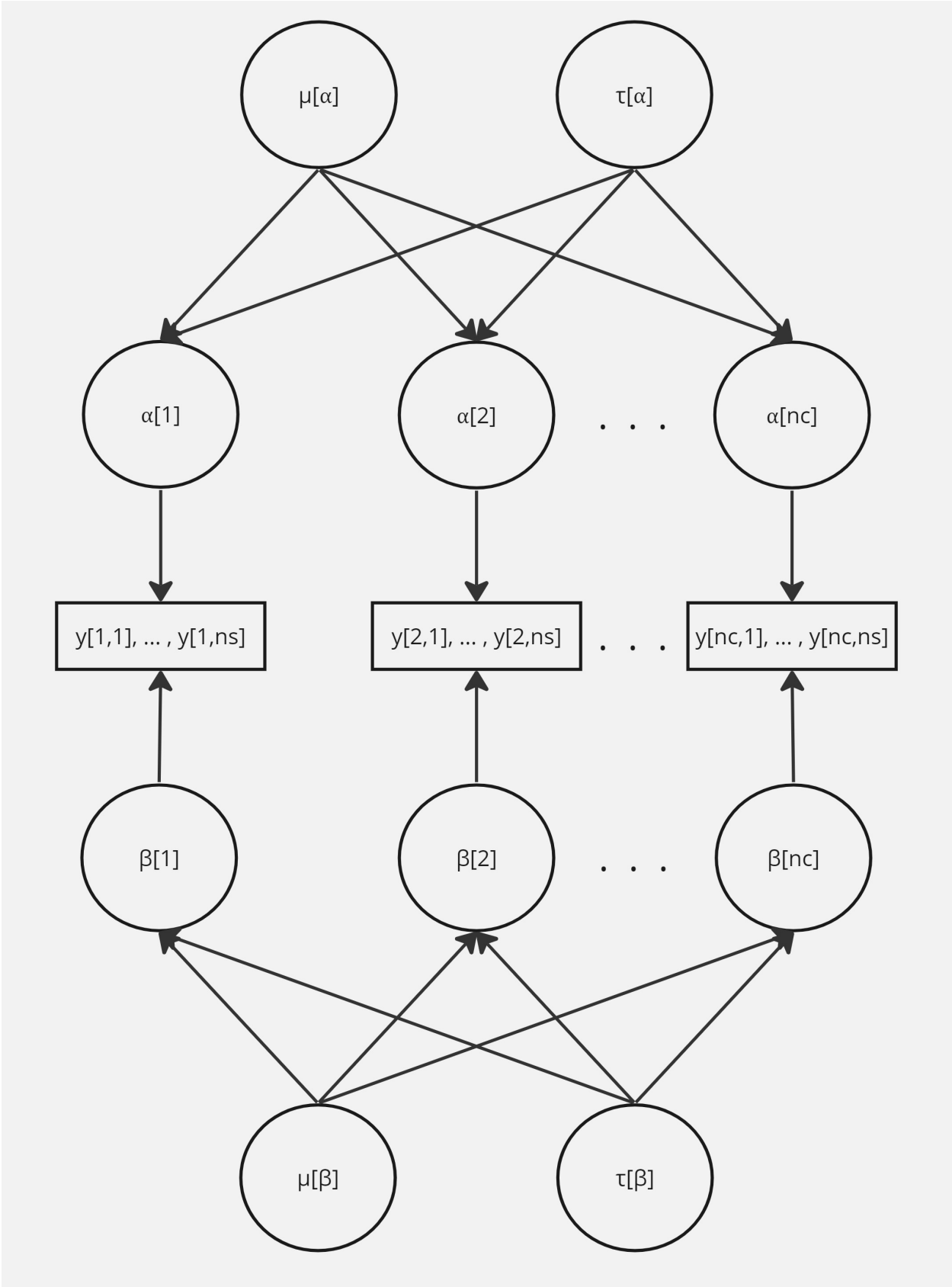


Figure 1: DAG

Interpretations

When looking at addressing the question of how effective the diagnostic ability is of EZK, lower α_c values are desired as these represent the false positives, and higher β_c are desired as these represent the different between the true positivity rate and the high positivity rate.

The results from the JAGS model show that countries 2, 9, and 10, which correspond to countries B, I, and J, have the lowest α_c values and countries and 1, 7, and 9, which correspond to countries A, G, and I, have the highest β_c values. These are countries then where the results from EZK are relatively reliable. Conversely, the countries that have the highest α_c values are 1, 6, and 7, which correspond to countries A, F, and G, and the lowest β_c values are 5, 6, and 8, which correspond to countries E, F, and H. These are the countries where the EZK tests are relatively most unreliable. It is important to note that I am interpreting these results relativistically and not directly. The mean values of α_c and β_c are on a logit scale and need to be transformed to get more direct results. However, because logit transformations are monotonic, these values can be compared amongst one another.

Specific mean values found from the JAGS model are provided in the appendix.

Appendix

```r
#Read Data
flu <- read.delim("flu.txt")
```

Jags

```r
#Libraries
require(coda)
```

```
## Loading required package: coda
```

```
## Warning: package 'coda' was built under R version 4.3.3
```

```r
require(rjags)
```

```
## Loading required package: rjags
```

```
## Warning: package 'rjags' was built under R version 4.3.3
```

```
## Linked to JAGS 4.3.1
```

```
## Loaded modules: basemod,bugs
```

```r
#initialize variables
nc <- 10
ns <- 100
y <- matrix(flu[,1], nrow = ns, ncol = nc, byrow = FALSE)
x1 <- matrix(flu[,2], nrow = ns, ncol = nc, byrow = FALSE)

#Initialize parameters
myinit <- list(alpha_c = rep(0, nc),
               beta_c = rep(0, nc),
```

```
               mu_alpha = 0,
               mu_beta = 1,
               tau_alpha = 0.1,
               tau_beta = 0.1)

#Iterations
niters <- 15000
nburns <- 5000
nadapt <- 5000
nchains <- 2

#Specify JAGS Model
mymodel <- "model {



#Likelihood
 for(i in 1:ns){
   for(j in 1:nc){
     y[i,j] ~ dbern(theta_c[i,j])
     logit(theta_c[i,j]) <- alpha_c[j] + beta_c[j] * x1[i,j]

   }
 }

#Priors make to vector,
mu_alpha ~ dnorm(0.1, 0.5)
mu_beta ~ dnorm(0.8, 0.5)
tau_alpha ~ dunif(0, 10)
tau_beta ~ dunif(0, 10)

for(j in 1:nc){
alpha_c[j] ~ dnorm(mu_alpha, tau_alpha)
```

```
beta_c[j] ~ dnorm(mu_beta, tau_beta)
}

}"

#fit and run jags
fit = jags.model(textConnection(mymodel),
                 data = list(x1 = x1, y = y, nc = nc, ns = ns),
                 inits = myinit,
                 n.chains = nchains,
                 n.adapt = nadapt)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1000
##    Unobserved stochastic nodes: 24
##    Total graph size: 2091
##
## Initializing model
```

```
fit.samples <- coda.samples(fit, c("alpha_c", "beta_c"), n.iter = niters)

#View results
summary(window(fit.samples, start = nburns + nadapt))
```

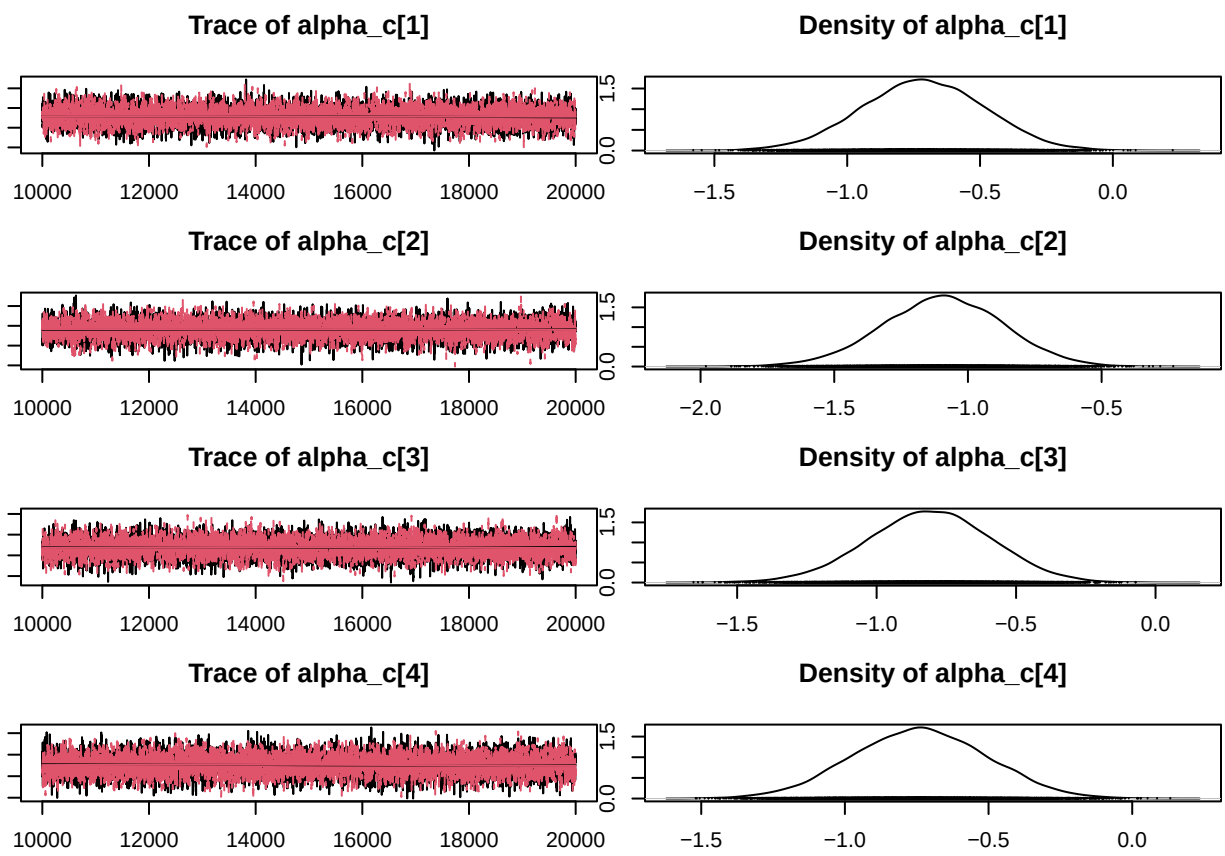
```
##
## Iterations = 10000:20000
## Thinning interval = 1
## Number of chains = 2
## Sample size per chain = 10001
##
```

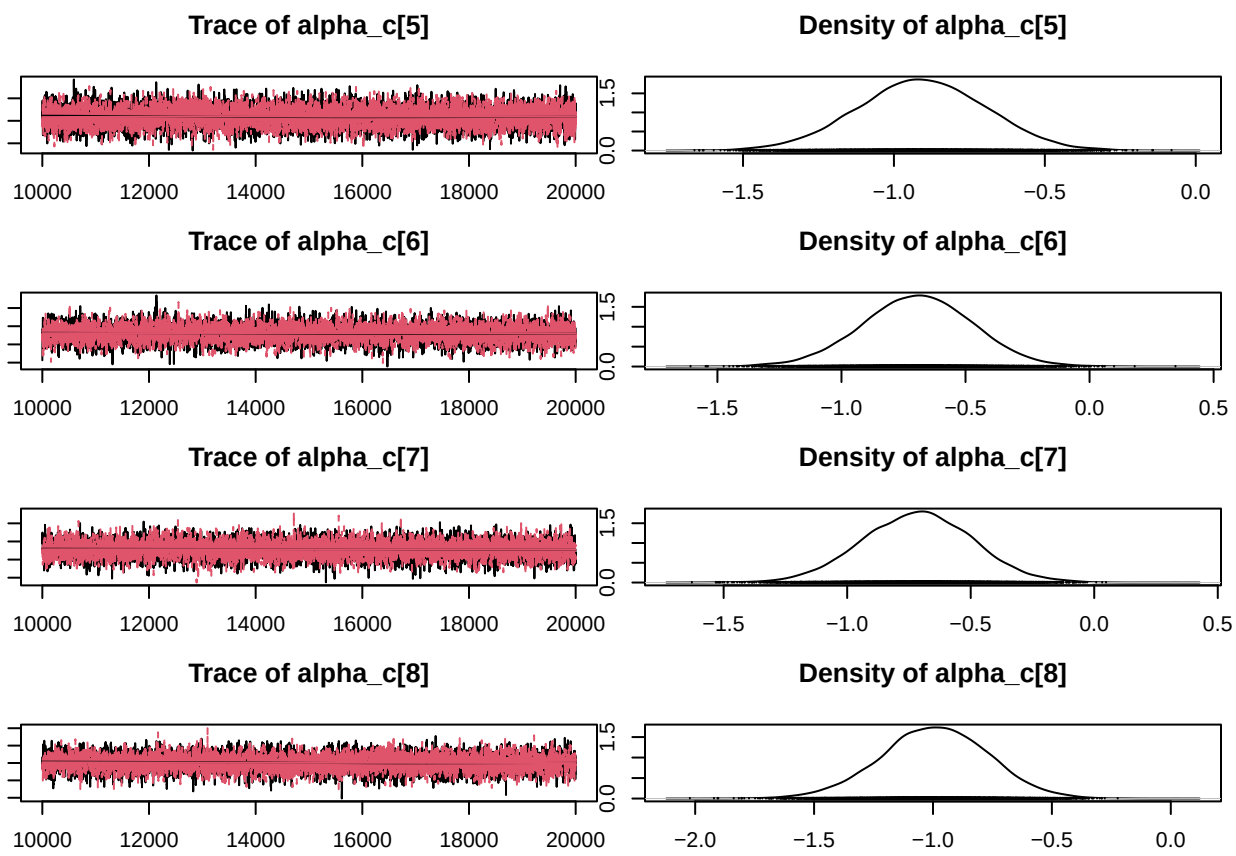
```
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                Mean     SD Naive SE Time-series SE
## alpha_c[1]  -0.7144 0.2311 0.001634       0.003068
## alpha_c[2]  -1.0950 0.2255 0.001595       0.003122
## alpha_c[3]  -0.8107 0.2221 0.001570       0.002935
## alpha_c[4]  -0.7397 0.2332 0.001649       0.003208
## alpha_c[5]  -0.9085 0.2125 0.001503       0.002561
## alpha_c[6]  -0.6940 0.2242 0.001585       0.003033
## alpha_c[7]  -0.7155 0.2236 0.001581       0.002885
## alpha_c[8]  -0.9935 0.2265 0.001601       0.003108
## alpha_c[9]  -0.9660 0.2221 0.001570       0.002945
## alpha_c[10] -1.0161 0.2277 0.001610       0.003115
## beta_c[1]    1.9186 0.2954 0.002088       0.004195
## beta_c[2]    1.6687 0.2993 0.002116       0.004188
## beta_c[3]    1.6604 0.2984 0.002110       0.004271
## beta_c[4]    1.9050 0.2963 0.002095       0.004266
## beta_c[5]    1.5988 0.3049 0.002156       0.004102
## beta_c[6]    1.4884 0.2938 0.002078       0.004246
## beta_c[7]    1.9136 0.3096 0.002189       0.004333
## beta_c[8]    1.5581 0.2917 0.002062       0.004190
## beta_c[9]    1.9406 0.3060 0.002164       0.004222
## beta_c[10]   1.6849 0.2899 0.002050       0.004262
##
## 2. Quantiles for each variable:
##
##                2.5%     25%     50%     75%   97.5%
## alpha_c[1]  -1.1717 -0.8700 -0.7144 -0.5574 -0.2688
## alpha_c[2]  -1.5462 -1.2457 -1.0921 -0.9404 -0.6601
## alpha_c[3]  -1.2533 -0.9580 -0.8091 -0.6607 -0.3822
## alpha_c[4]  -1.1960 -0.8987 -0.7398 -0.5806 -0.2859
## alpha_c[5]  -1.3310 -1.0499 -0.9090 -0.7646 -0.4936
```

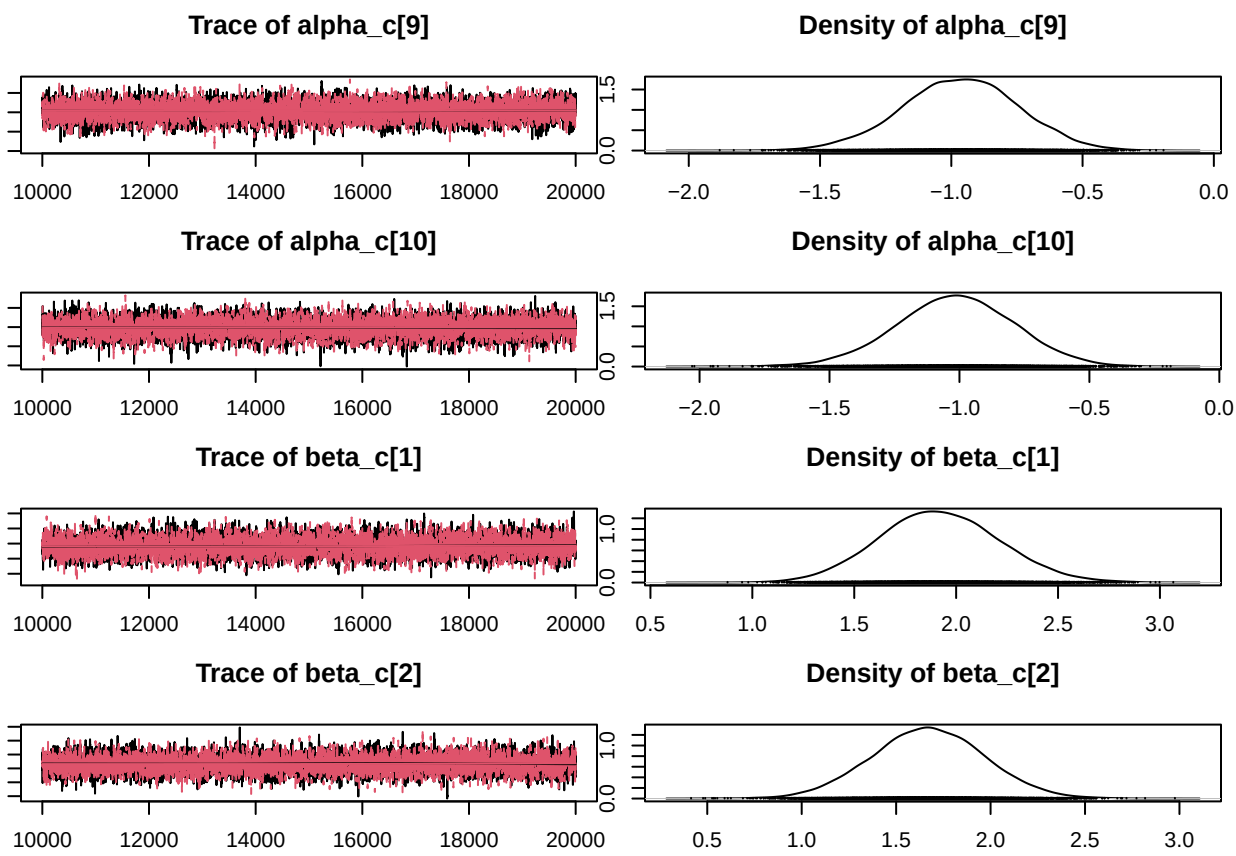
```
## alpha_c[6]  -1.1385 -0.8424 -0.6924 -0.5442 -0.2592
## alpha_c[7]  -1.1553 -0.8682 -0.7134 -0.5638 -0.2805
## alpha_c[8]  -1.4497 -1.1418 -0.9912 -0.8394 -0.5560
## alpha_c[9]  -1.4140 -1.1146 -0.9634 -0.8164 -0.5394
## alpha_c[10] -1.4726 -1.1655 -1.0138 -0.8625 -0.5749
## beta_c[1]    1.3562  1.7168  1.9127  2.1142  2.5082
## beta_c[2]    1.0857  1.4691  1.6695  1.8693  2.2546
## beta_c[3]    1.0784  1.4619  1.6558  1.8533  2.2614
## beta_c[4]    1.3389  1.7031  1.9012  2.1045  2.4867
## beta_c[5]    0.9888  1.3969  1.5962  1.8033  2.1986
## beta_c[6]    0.9063  1.2932  1.4878  1.6849  2.0658
## beta_c[7]    1.3150  1.7059  1.9111  2.1184  2.5291
## beta_c[8]    0.9857  1.3620  1.5565  1.7524  2.1254
## beta_c[9]    1.3634  1.7330  1.9364  2.1438  2.5535
## beta_c[10]   1.1261  1.4911  1.6824  1.8798  2.2599
```

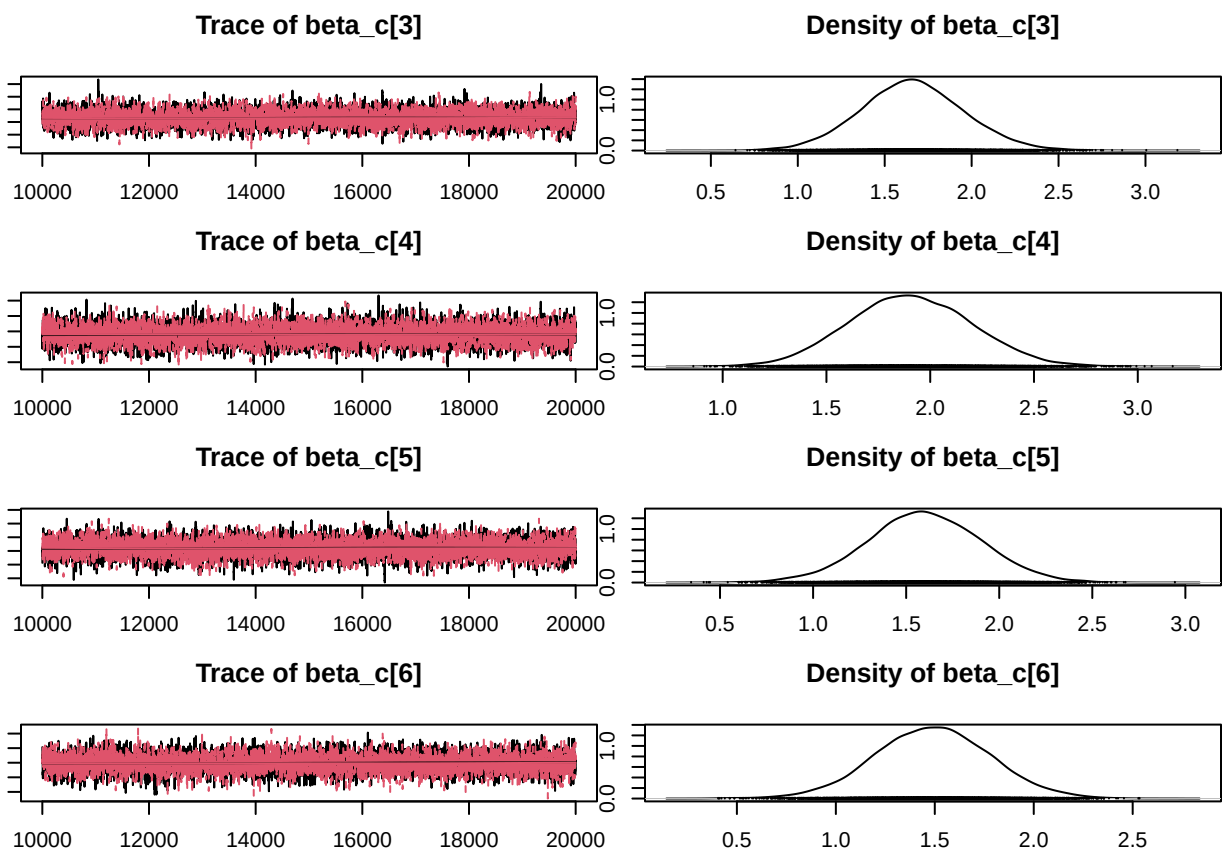
Plots

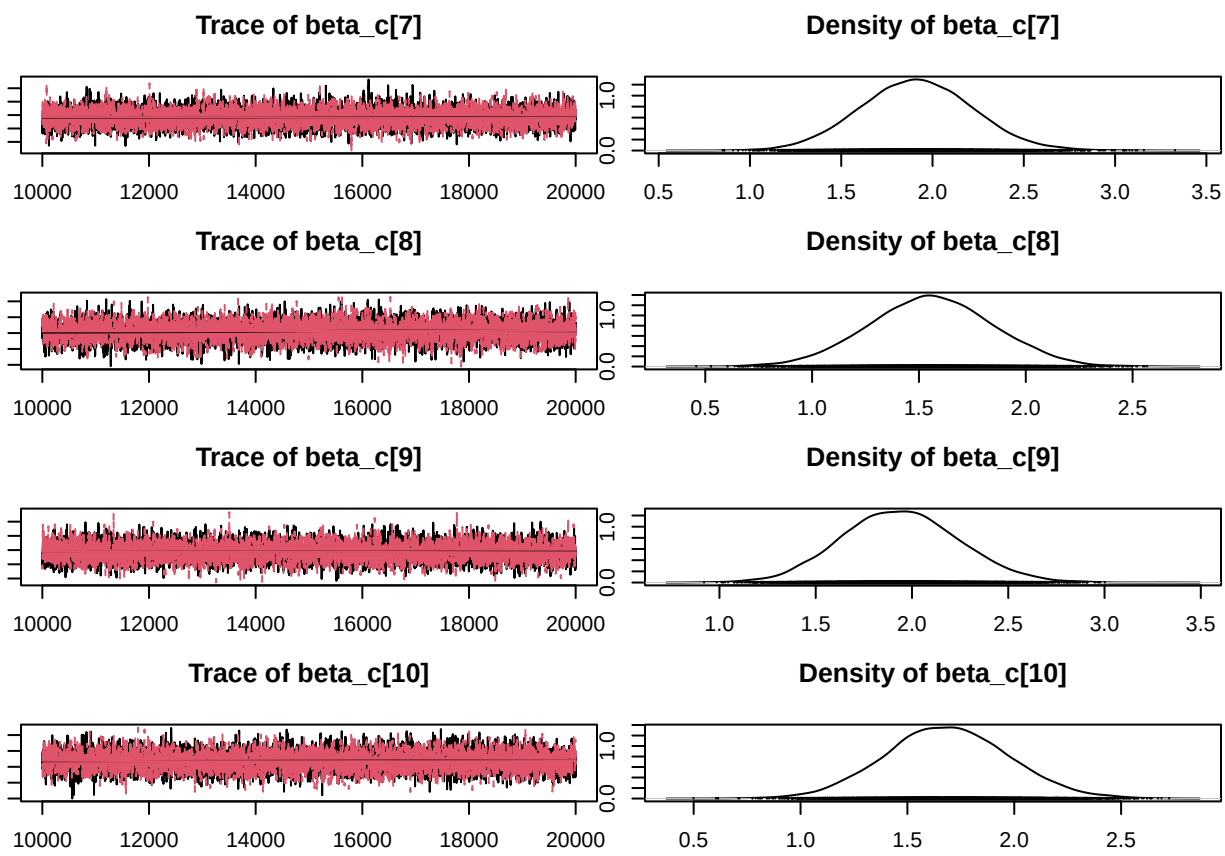
```r
# Adjust plot sizes
par(pin = c(3, 0.4))

# Plot
plot(window(fit.samples, start = nburns + nadapt))
```


Trace of alpha_c[1]
Density of alpha_c[1]
Trace of alpha_c[2]
Density of alpha_c[2]
Trace of alpha_c[3]
Density of alpha_c[3]
Trace of alpha_c[4]
Density of alpha_c[4]
10000
12000
14000
16000
18000
20000
0.0
1.5
−2.0
−1.5
−1.0
−0.5
0.0


Trace of alpha_c[5]
Density of alpha_c[5]
Trace of alpha_c[6]
Density of alpha_c[6]
Trace of alpha_c[7]
Density of alpha_c[7]
Trace of alpha_c[8]
Density of alpha_c[8]
10000
12000
14000
16000
18000
20000
0.0
1.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5


Trace of alpha_c[9]
Density of alpha_c[9]
Trace of alpha_c[10]
Density of alpha_c[10]
Trace of beta_c[1]
Density of beta_c[1]
Trace of beta_c[2]
Density of beta_c[2]


Trace of beta_c[3]
Density of beta_c[3]
Trace of beta_c[4]
Density of beta_c[4]
Trace of beta_c[5]
Density of beta_c[5]
Trace of beta_c[6]
Density of beta_c[6]
10000
12000
14000
16000
18000
20000
0.0
1.0
0.5
1.0
1.5
2.0
2.5
3.0


Trace of beta_c[7]
Density of beta_c[7]
Trace of beta_c[8]
Density of beta_c[8]
Trace of beta_c[9]
Density of beta_c[9]
Trace of beta_c[10]
Density of beta_c[10]
10000
12000
14000
16000
18000
20000
0.0
1.0
0.5
1.5
2.0
2.5
3.0
3.5