Final Project

Connor McNeill

The Ohio State University, Department of Statistics

Statistics 3303: Bayesian Analysis and Statistical Decision Making

Prof. Andrew Richards

April 26, 2022

I. Introduction

We have been asked to analyze data from a small clinical trial run by the World Health Organization to study the diagnostic ability of the EZK test: an inexpensive diagnostic test that detects the presence of a new strain of influenza, K9C9. The clinical trial was conducted in 10 countries, and in each country, 100 subjects were randomly selected and tested for K9C9 using both the EZK test and an accurate, more expensive test.

	EZK Negative	EZK Positive
K9C9 Negative	355	174
K9C9 Positive	139	332

Table I.1: Table shows the accuracy of the EZK test

As Table I.1 shows, the issue with the EZK test is that while it is inexpensive, there are a significant number of false positives and false negatives. We will evaluate the effectiveness of the EZK test in this report, and then determine for an insurance provider in Country D whether the EZK test is effective enough to predict whether an individual should be treated with the drug Neonicon.

II. Methods

II.1 Hierarchical Bayesian Model

Let Y_{ij} be the binary indicator of whether subject i in country j is infected (1) or not infected (0) according to the highly accurate diagnostic test. Let n_j be the total number of subjects tested in country j. Let x_{ij} be the indicator of whether subject i's EZK test was positive (1) or negative (0) in country j. We need to refactor the Country variable so that it is a number from 1 to 10 instead of the corresponding letter.

Let θ_{ij} be the true (but unknown) probability of subject i having K9C9 in country j.

I will use a random-effects model to model this due to potential variations in both test accuracy and the virus. Viruses change rapidly with frequent mutations. As such, certain countries may have different rates of prevalence and transmission. The model uses a hierarchical structure in order to account for this variation and the impact of both the accuracy of the EZK test and the country.

Let α_j be the parameter which represents the relationship between the country and the probability that an individual in that country is infected with K9C9. μ_α represents the true mean of α, and σ_α^2 represents the true variance of α.

Similarly, let β_j be the parameter which represents the relationship between testing positive on the EZK test and the probability that an individual in that country is infected with K9C9. μ_β represents the true mean of β, and σ_β^2 represents the true variance of β.

Let $\boldsymbol{Y} = (Y_1, \dots, Y_J), \boldsymbol{n} = (n_1, \dots, n_J), \boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_J), \boldsymbol{\beta} = (\beta_1, \dots, \beta_J);$ where J is the total number of countries. We know that $J = 10$ from the data set.

Likelihood: We assume conditional independence:

$$p(\boldsymbol{Y} = \boldsymbol{y}|\boldsymbol{\alpha}, \boldsymbol{\beta}) = \prod_{j=1}^{J} \prod_{i=1}^{n_j} p(Y_{ij}|\alpha_j, \beta_j)$$

where

$$Y_{ij} \,|\, \alpha_j, \beta_j \sim \text{Bernoulli}(\theta_{ij}) \text{ for } j = 1, \dots, J \text{ and } i = 1, \dots, n_j$$

and

$$\log\left(\frac{\theta_{ij}}{1-\theta_{ij}}\right) = \alpha_j + \beta_j x_{ij}$$

Priors: For $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ we assume:

$$p(\alpha, \beta|\mu_\alpha, \sigma_\alpha^2, \mu_\beta, \sigma_\beta^2) = \prod_{j=1}^{J} p(\alpha_j|\mu_\alpha, \sigma_\alpha^2) p(\beta_j|\mu_\beta, \sigma_\beta^2)$$

And for all $j = 1, \dots, J$:

$$\alpha_j \sim N(\mu_\alpha, \sigma_\alpha^2)$$

and

$$\beta_j \sim N(\mu_\beta, \sigma_\beta^2)$$

Finally,

$$p(\mu_\alpha, \sigma_\beta^2, \mu_\beta, \sigma_\beta^2) = p(\mu_\alpha) p(\sigma_\alpha^2) p(\mu_\beta) p(\sigma_\beta^2)$$

where

$$p(\mu_\alpha) \sim N(0,1), \quad p(\sigma_\alpha^2) \sim IG(2,1), \quad p(\mu_\beta) \sim N(0,1), \quad p(\sigma_\beta^2) \sim IG(2,1)$$

II.2 Modeling Probability

Our Bayesian Hierarchical Model utilizes the logit function to transform the parameters of the function (α and β) into the log-odds of a subject having the K9C9 virus.

$$\log\left(\frac{\theta_{ij}}{1-\theta_{ij}}\right) = \alpha_j + \beta_j x_{ij}$$

In order to get the probability from the log-odds, we can take the inverse logit of the above equation.

$$\theta_{ij} = \frac{\exp(\alpha_j + \beta_j x_{ij})}{1 + \exp(\alpha_j + \beta_j x_{ij})}$$

II.3 Assessing the Diagnostic Ability of EZK

Using our model and JAGS, we can determine the posterior distribution of each of our 24 parameters (10 alphas, 10 betas, and then the overall mean and variance of them). From this, we can then determine the posterior distributions of theta (the true probability) for each country by performing the transformation given above. Since $x_{ij} = 1$ for a positive EZK test, we can determine the probability of truly testing positive given the EZK test is positive for each positive with log-odds being equal to $\alpha_j + \beta_j$.

II.4 Model Fitting in JAGS

I performed MCMC using the JAGS interface in R using our model. I ran the MCMC through 50,000 iterations, with a burn-in of 5,000 iterations and 5,000 adaptive steps. This means that 40,000 iterations are used for the final posterior estimation. I only ran one MCMC chain to simplify the output; due to the large number of iterations, adaptive steps, and burn-in, the final MCMC is already rather accurate.

The starting values for all the alphas and betas, along with their overall mean was set to be zero. The starting value for the overall precision of the alphas and betas was set to be equal to one.

In order to evaluate if the MCMC algorithm converged, I first looked at the trace plots and the density plots for each of the 24 posterior distributions, along with the autocorrelation plots. These plots can all be found in the Appendix of the report. All the trace plots show no signs of any

patterns, all the density plots are smooth and without any skewness (with the exception of the two density plots for the variances which are skewed right, but that matches the expected inverse gamma posterior for them). Additionally, the autocorrelation plots show low amount of autocorrelation. Finally, looking at the effective sample size of each of the posteriors, all of them are greater than 9300, which is pretty decent considering the number of iterations being 50,000. All of this shows that the algorithm has converged to our final posterior estimates.

III. Results

III.1 Estimated Posterior Distributions

Our JAGS MCMC model produced twenty-four estimated posterior distributions for all of our parameters. The following two boxplots summarize the estimated posterior distributions of the components of our two parameters: alpha and beta (which vary between each of the ten countries).

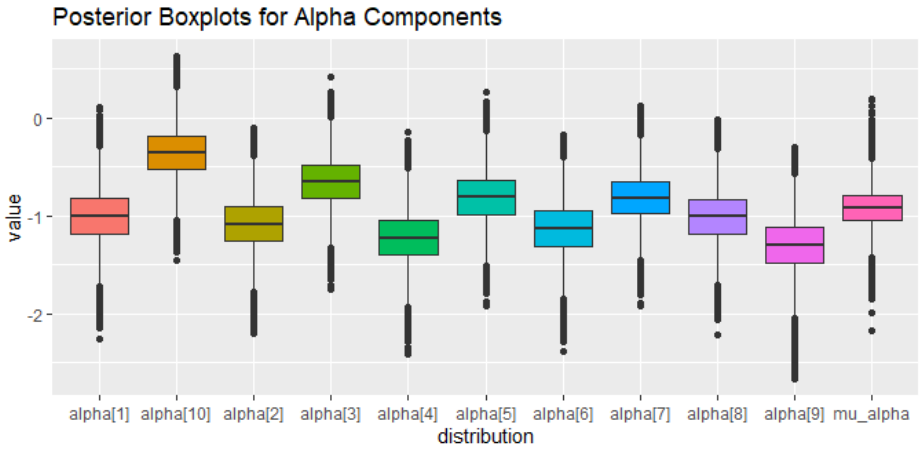


Figure III.1: This figure shows the boxplot of each of the 10 posterior distributions of alpha, along with that of the overall mean of the alphas.

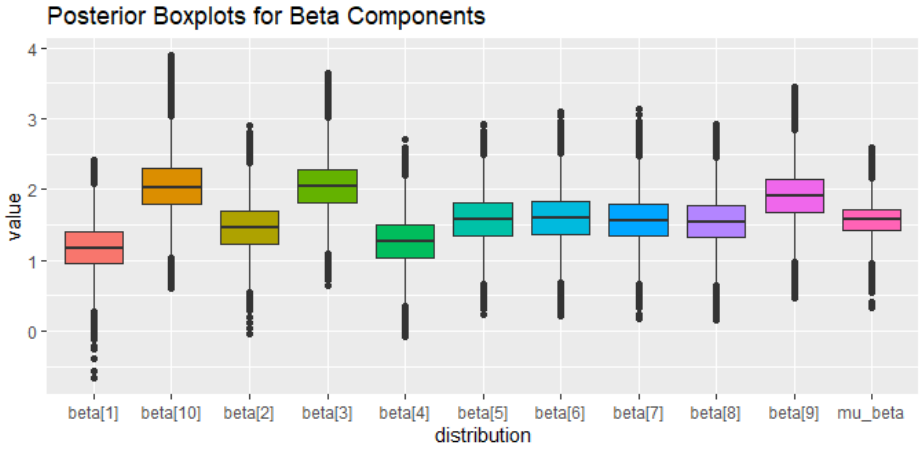


Figure III.2: This figure shows the boxplot of each of the 10 posterior distributions of beta, along with that of the overall mean of the betas.

We see that the alpha parameter captures more variation between the countries than the beta parameter. This is somewhat expected as the model design was built so that alpha captured the variation of the virus in each country while the beta component captured the variation of the accuracy of the EZK test in each country. We would expect that the EZK test positivity does not vary as greatly in each country, which is displayed by the posterior boxplots of beta. Now, let's look at the estimated theta posterior distributions after transforming the parameters as discussed earlier.

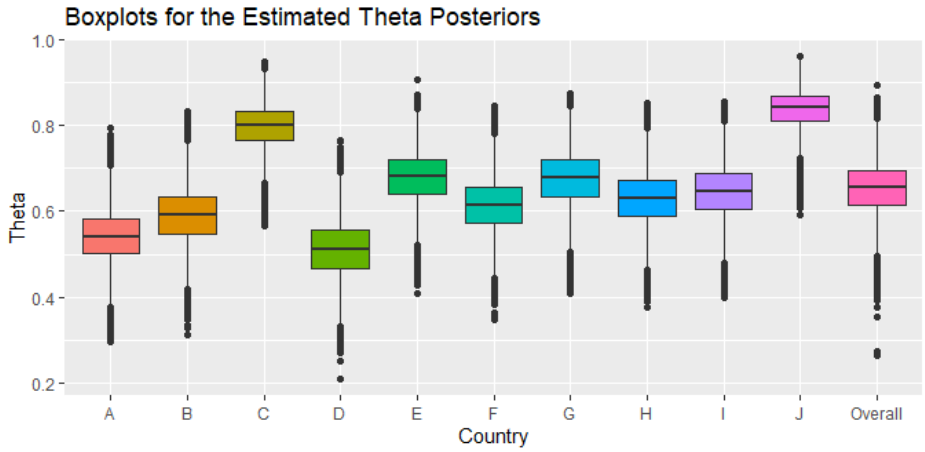


Figure III.3: This figure shows the estimated posteriors for theta for each of the ten countries and the overall mean. Note: the ordering is different than in figures III.1 and III.2 since index 10 corresponds to country J.

We see that the overall expected value of theta – in this case, given that the EZK test is positive – is approximately 66.34%. Likewise, we know that the log-odds of theta given the EZK test being negative is estimated by alpha. As such, we can calculate that expected log-odds to be approximately -0.9400, which corresponds to a theta of approximately 28.09%. Specifically looking at Country D, we see that the estimated theta distribution is much smaller than the overall. Our estimated value of theta (given a Positive EZK) is 51.10% and 22.70% (given a negative EZK).

III.2 Loss Function

The following table shows the costs incurred with each decision of the insurance company of whether or not to treat someone with the drug Neonicon, given that their EZK test is positive.

	Infected	Not Infected
Treat with Drug	$457	$457
Do Not Treat with Drug	$1490	$0

Table III.1: Shows the expected costs to the insurance company in country D of the decision of whether to treat the patient with the drug Neonicon.

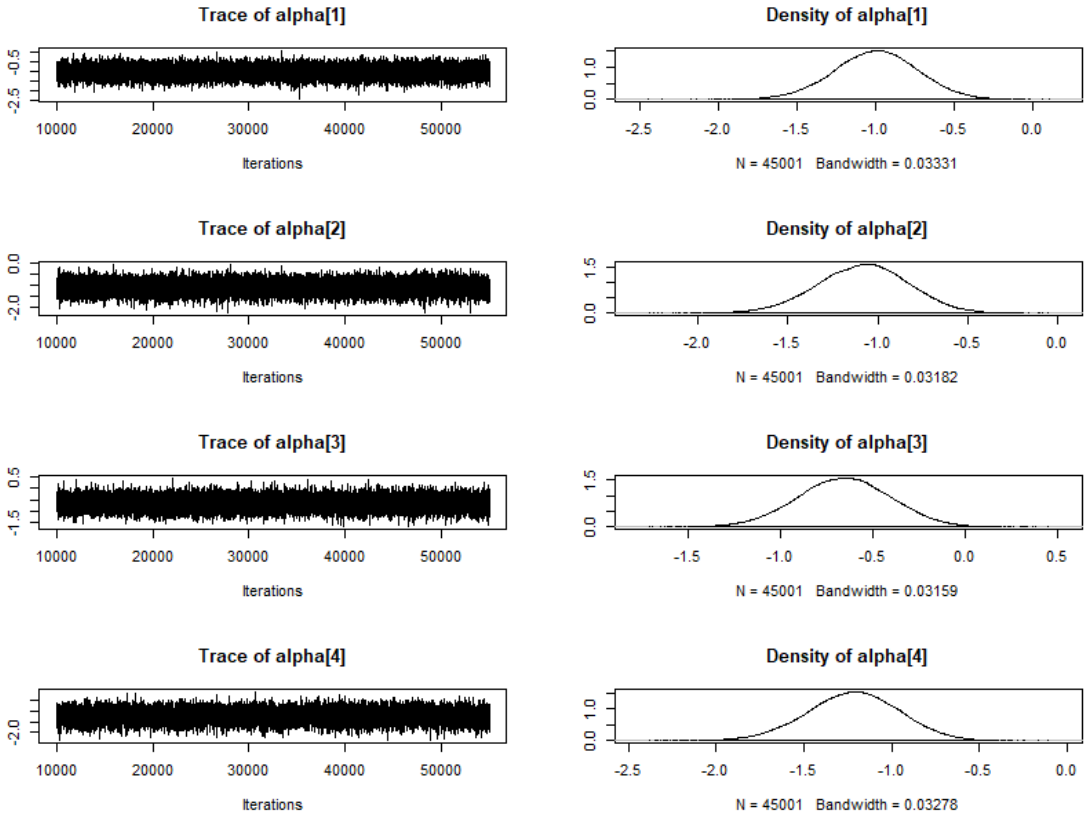
Let treating the patient with Neonicon be decision a_1 and not treating the patient be decision a_2. Based on Table III.1, we can develop an expected loss function for the decision set $\mathcal{A} = \{a_1, a_2\}$. The expected loss of $a_1 = 457\theta + 457(1-\theta) = 457$ and of $a_2 = 1490\theta + 0(1-\theta)$. We choose a_1 if $457 < 1490\theta \Rightarrow \theta > 457/1490$, and we choose a_2 if $\theta < 457/1490$. We can look at our estimated posterior function for theta in country D and find then percentage of times we would choose a_1 and a_2 based on the decision rule above. The result was 99.91778% for a_1 and 0.08222% for a_2.

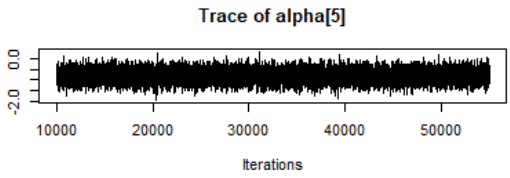
IV. Conclusion

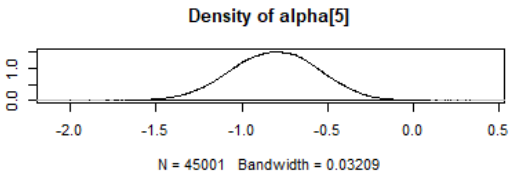
In conclusion, we found that the EZK test is approximately 66.34% effective in correctly predicting an infection of K9C9 overall across the ten countries. However, the effectiveness varies across the ten different countries due to expected variation in the virus and the accuracy of the test. Specifically, when looking at country D, the effectiveness of the EZK test drops to 51.10%, which is the unnormalized true positive rate. On the surface, this may mean that it is not effective enough to determine whether a patient should be treated with the drug Neonicon or not. However, when comparing the cost of not treating the patient if they actually are positive, then it makes sense that we should rather be safe than sorry (assuming that there aren't too many side effects). In fact, based on the decision theory criterion discussed in Section III.2, we found that 99.9% of the time, a positive result of the EZK test suggests that insurance company should permit the prescription of Neonicon. As such, the insurance company should approve Neonicon prescriptions for individuals who test positive for K9C9 on the EZK test.

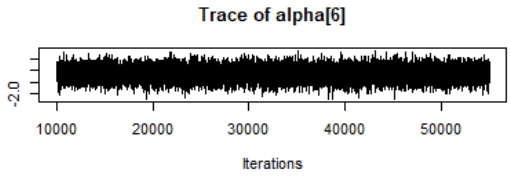
V. Appendix

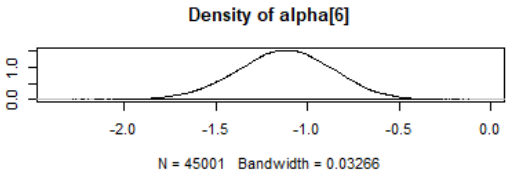
V.1 JAGS Model Plots: Trace, Density, Autocorrelation

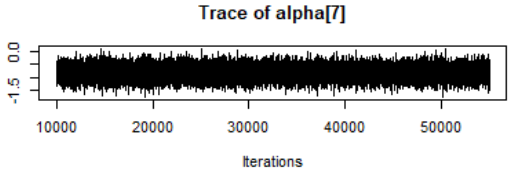


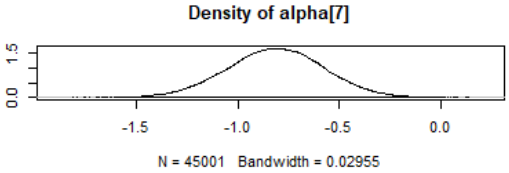

Trace of alpha[5]
Iterations

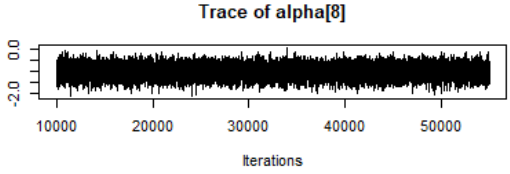

Density of alpha[5]
N = 45001 Bandwidth = 0.03209

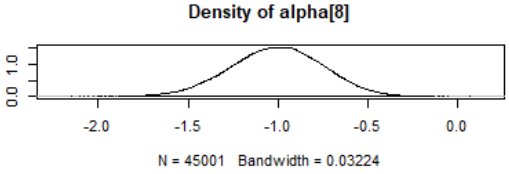

Trace of alpha[6]
Iterations

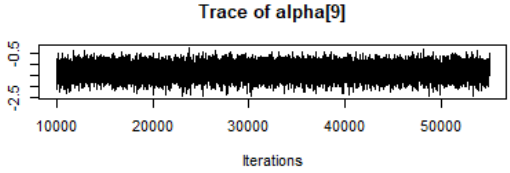

Density of alpha[6]
N = 45001 Bandwidth = 0.03266

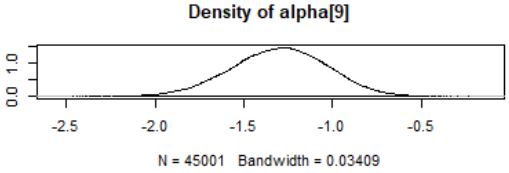

Trace of alpha[7]
Iterations

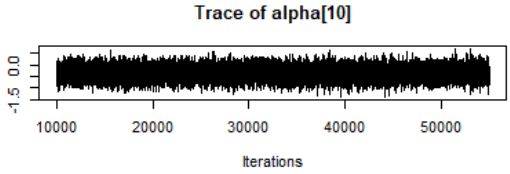

Density of alpha[7]
N = 45001 Bandwidth = 0.02955

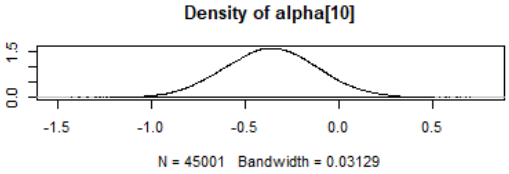

Trace of alpha[8]
Iterations

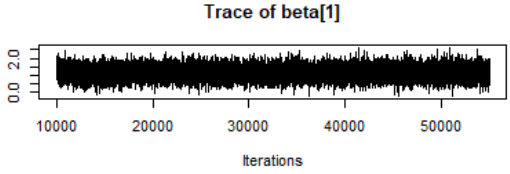

Density of alpha[8]
N = 45001 Bandwidth = 0.03224

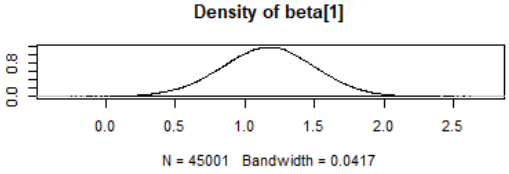

Trace of alpha[9]
Iterations

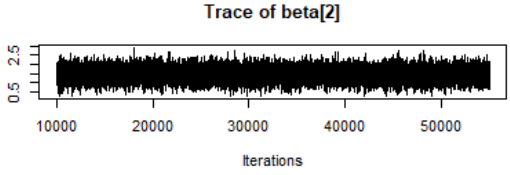

Density of alpha[9]
N = 45001 Bandwidth = 0.03409

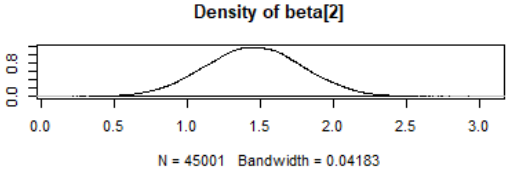

Trace of alpha[10]
Iterations

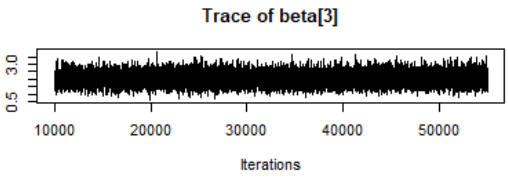

Density of alpha[10]
N = 45001 Bandwidth = 0.03129

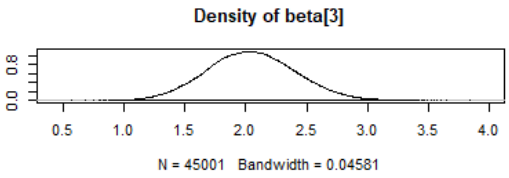

Trace of beta[1]
Iterations

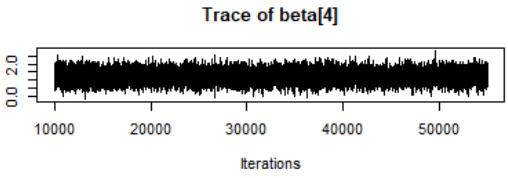

Density of beta[1]
N = 45001 Bandwidth = 0.0417

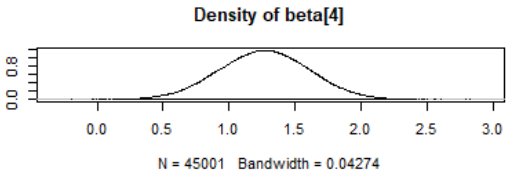

Trace of beta[2]
Iterations

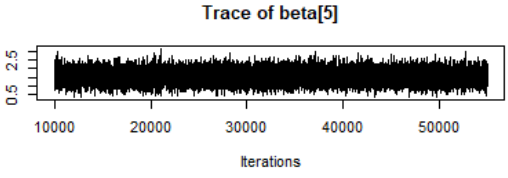

Density of beta[2]
N = 45001 Bandwidth = 0.04183

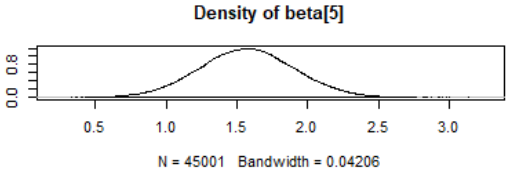

Trace of beta[3]
Iterations

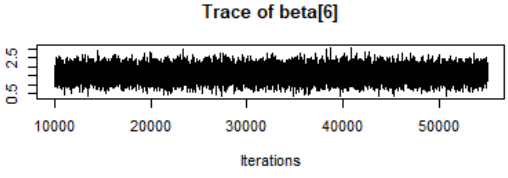

Density of beta[3]
N = 45001 Bandwidth = 0.04581

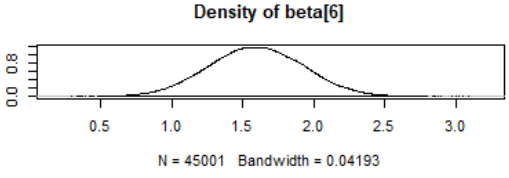

Trace of beta[4]
Iterations

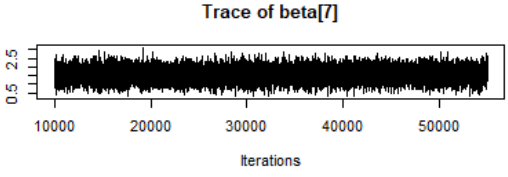

Density of beta[4]
N = 45001 Bandwidth = 0.04274

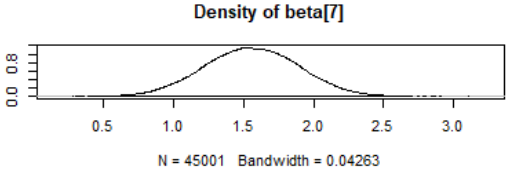

Trace of beta[5]
Iterations

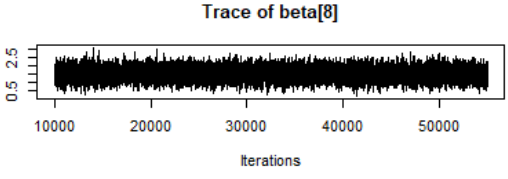

Density of beta[5]
N = 45001 Bandwidth = 0.04206

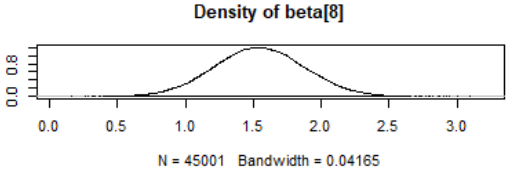

Trace of beta[6]
Iterations

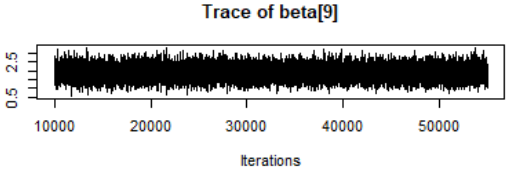

Density of beta[6]
N = 45001 Bandwidth = 0.04193

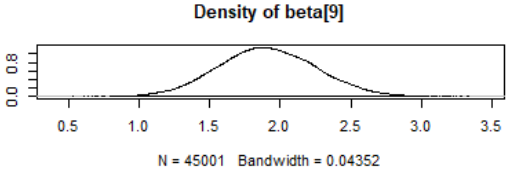

Trace of beta[7]
Iterations

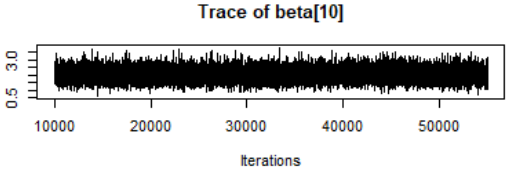

Density of beta[7]
N = 45001 Bandwidth = 0.04263

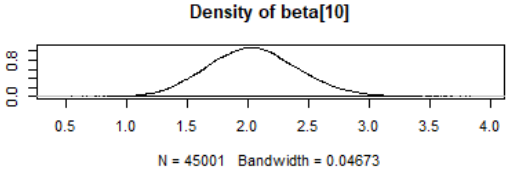

Trace of beta[8]
Iterations

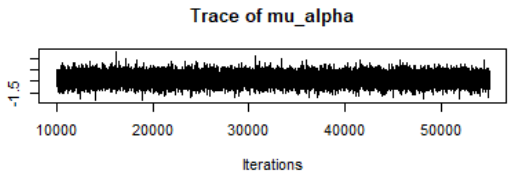

Density of beta[8]
N = 45001 Bandwidth = 0.04165

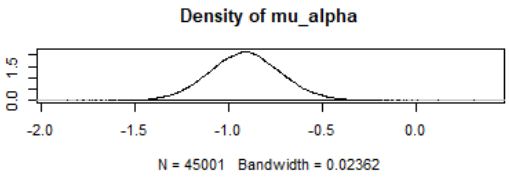

Trace of beta[9]
Iterations

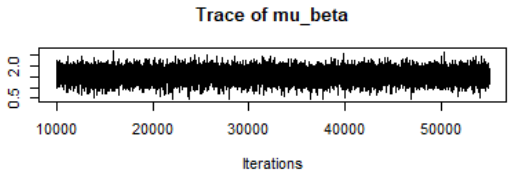

Density of beta[9]
N = 45001 Bandwidth = 0.04352

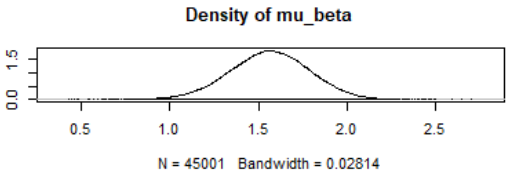

Trace of beta[10]
Iterations

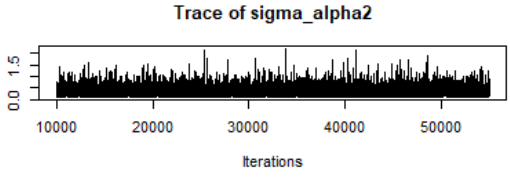

Density of beta[10]
N = 45001 Bandwidth = 0.04673

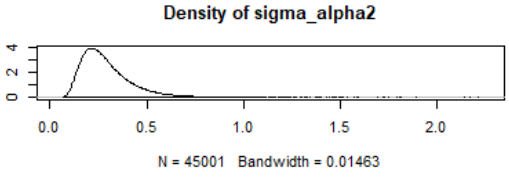

Trace of mu_alpha
Iterations

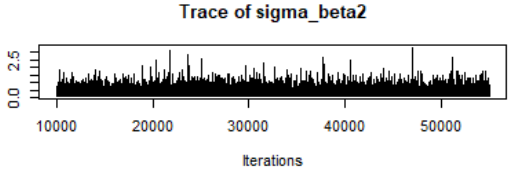

Density of mu_alpha
N = 45001 Bandwidth = 0.02362

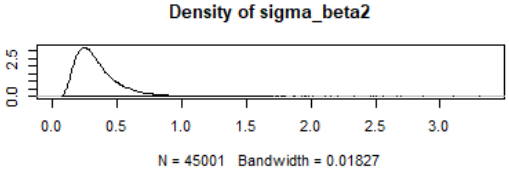

Trace of mu_beta
Iterations

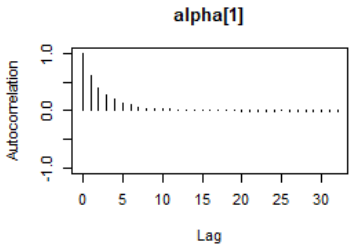

Density of mu_beta
N = 45001 Bandwidth = 0.02814

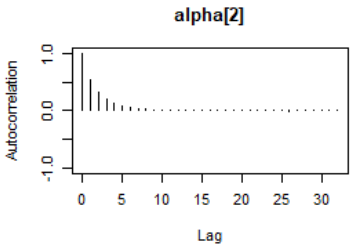

Trace of sigma_alpha2
Iterations

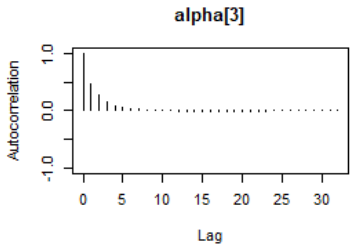

Density of sigma_alpha2
N = 45001 Bandwidth = 0.01463

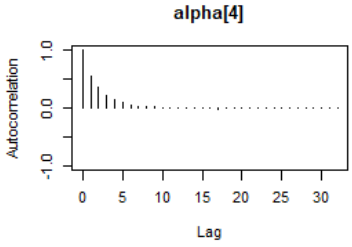

Trace of sigma_beta2
Iterations

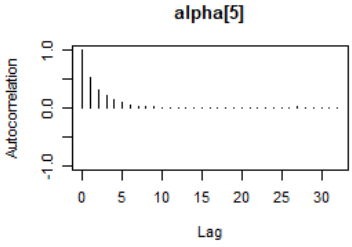

Density of sigma_beta2
N = 45001 Bandwidth = 0.01827

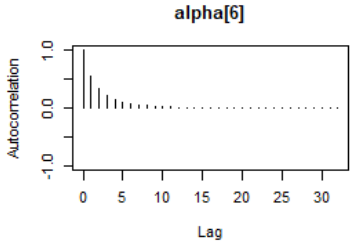

alpha[1]
Autocorrelation
Lag

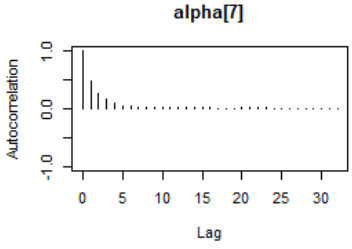

alpha[2]
Autocorrelation
Lag

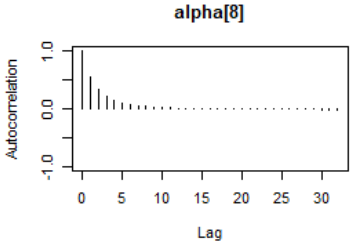

alpha[3]
Autocorrelation
Lag

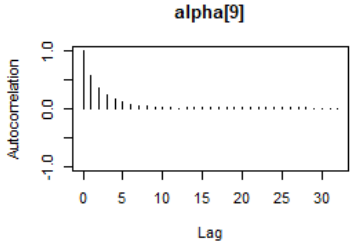

alpha[4]
Autocorrelation
Lag

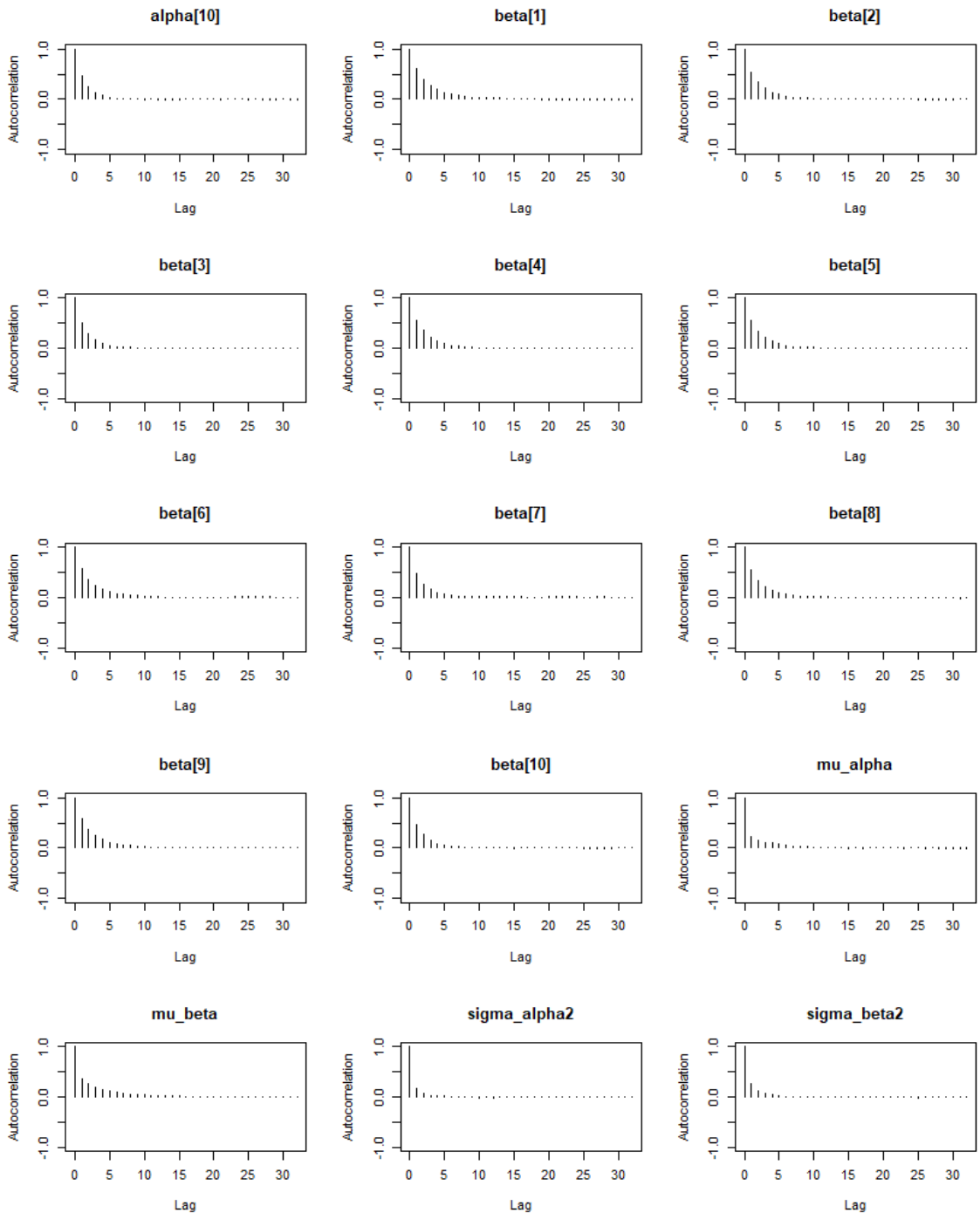

alpha[5]
Autocorrelation
Lag


alpha[6]
Autocorrelation
Lag


alpha[7]
Autocorrelation
Lag


alpha[8]
Autocorrelation
Lag


alpha[9]
Autocorrelation
Lag


alpha[10]
beta[1]
beta[2]
beta[3]
beta[4]
beta[5]
beta[6]
beta[7]
beta[8]
beta[9]
beta[10]
mu_alpha
mu_beta
sigma_alpha2
sigma_beta2
Autocorrelation
Lag
-1.0
0.0
1.0
0
5
10
15
20
25
30

V.2 R Code

```
# Import Libraries Needed
library(coda)
library(rjags)
library(tidyverse)

# Set seed
set.seed(3303)

# Import Dataset
flu <- read_csv("C:/Users/conno/OneDrive - The Ohio State University/~1222 -
SP22/Stat 3303/FINAL PROJECT/data 22")

# Summarize Rate of Infection & Positivity in Each Country
flu %>%
  group_by(Country) %>%
  summarize(infectionRate = mean(Infected),
            positiveRate = mean(EZK))

## # A tibble: 10 x 3
##    Country infectionRate positiveRate
##    <chr>           <dbl>        <dbl>
##  1 A                0.42         0.56
##  2 B                0.41         0.48
##  3 C                0.59         0.52
##  4 D                0.35         0.47
##  5 E                0.51         0.53
##  6 F                0.43         0.52
##  7 G                0.47         0.43
##  8 H                0.45         0.51
##  9 I                0.42         0.51
## 10 J                0.66         0.53

# Refactor Country
flu <- flu %>%
  mutate(CountryFac = case_when(
    Country == 'A' ~ 1,
    Country == 'B' ~ 2,
    Country == 'C' ~ 3,
    Country == 'D' ~ 4,
    Country == 'E' ~ 5,
    Country == 'F' ~ 6,
    Country == 'G' ~ 7,
    Country == 'H' ~ 8,
    Country == 'I' ~ 9,
    Country == 'J' ~ 10
  ))
```

```
# Summarize Overall Results
table(flu$Infected, flu$EZK)

##
##       0   1
##   0 355 174
##   1 139 332

# Find length
n <- length(flu$Infected)

# Setup input data list for JAGS:
mydata = list(n=n, y=flu$Infected, ezk=flu$EZK, country=flu$CountryFac)

# Setup parameter initialization for JAGS:
myinit = list(alpha = rep(0,10),
              beta = rep(0,10),
              mu_alpha = 0,
              tau_alpha2 = 1,
              mu_beta = 0,
              tau_beta2 = 1)

# Setup MCMC options for JAGS:
niters = 50000
nburns = 5000
nadapt = 5000
nchains = 1

# Specify JAGS model:
model <- " model {
  # likelihood
  for (i in 1:n) {
    y[i] ~ dbern(theta[i])
    logit(theta[i]) <- alpha[country[i]] + beta[country[i]]*ezk[i]
  }

  # priors
  mu_alpha ~ dnorm(0, 1)
  tau_alpha2 ~ dgamma(2, 1)
  mu_beta ~ dnorm(0, 1)
  tau_beta2 ~ dgamma(2, 1)

  for (j in 1:10) {
    alpha[j] ~ dnorm(mu_alpha, tau_alpha2)
    beta[j] ~ dnorm(mu_beta, tau_beta2)
  }

  # sigmas
  sigma_alpha2 <- 1/tau_alpha2
  sigma_beta2 <- 1/tau_beta2
```

```
}"

# Fit JAGS Model
fit=jags.model(textConnection(model),
               data=mydata, 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: 3090
##
## Initializing model

# Run MCMC
fit.samples = coda.samples(fit, c('alpha', 'beta', 'mu_alpha', 'sigma_alpha2'
,
                                  'mu_beta', 'sigma_beta2'),
                           n.iter=niters)

# Summarize MCMC output
summary(window(fit.samples,start=nburns+nadapt))

##
## Iterations = 10000:55000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 45001
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                 Mean     SD  Naive SE Time-series SE
## alpha[1]     -0.9972 0.2689 0.0012675      0.0027952
## alpha[2]     -1.0812 0.2588 0.0012199      0.0024682
## alpha[3]     -0.6472 0.2529 0.0011921      0.0020939
## alpha[4]     -1.2263 0.2627 0.0012385      0.0025511
## alpha[5]     -0.8086 0.2534 0.0011946      0.0023184
## alpha[6]     -1.1234 0.2683 0.0012646      0.0025907
## alpha[7]     -0.8162 0.2403 0.0011329      0.0020632
## alpha[8]     -1.0031 0.2573 0.0012131      0.0024070
## alpha[9]     -1.3046 0.2756 0.0012993      0.0027632
## alpha[10]    -0.3504 0.2537 0.0011959      0.0021102
## beta[1]       1.1683 0.3388 0.0015970      0.0037023
## beta[2]       1.4533 0.3380 0.0015933      0.0031733
## beta[3]       2.0376 0.3637 0.0017145      0.0032188
## beta[4]       1.2693 0.3413 0.0016090      0.0033336
## beta[5]       1.5714 0.3335 0.0015723      0.0030568
```

```
## beta[6]       1.5909 0.3400 0.0016026      0.0033159
## beta[7]       1.5612 0.3428 0.0016157      0.0030180
## beta[8]       1.5396 0.3347 0.0015780      0.0031703
## beta[9]       1.9087 0.3512 0.0016557      0.0035063
## beta[10]      2.0335 0.3800 0.0017915      0.0032983
## mu_alpha     -0.9086 0.1972 0.0009295      0.0015905
## mu_beta       1.5588 0.2287 0.0010781      0.0021509
## sigma_alpha2  0.3019 0.1542 0.0007268      0.0009466
## sigma_beta2   0.3484 0.1912 0.0009014      0.0013508
##
## 2. Quantiles for each variable:
##
##                 2.5%     25%     50%     75%   97.5%
## alpha[1]     -1.5369 -1.1750 -0.9949 -0.8135 -0.4817
## alpha[2]     -1.6013 -1.2525 -1.0771 -0.9073 -0.5806
## alpha[3]     -1.1492 -0.8172 -0.6456 -0.4762 -0.1565
## alpha[4]     -1.7517 -1.3988 -1.2240 -1.0484 -0.7223
## alpha[5]     -1.3143 -0.9782 -0.8061 -0.6372 -0.3210
## alpha[6]     -1.6644 -1.2975 -1.1193 -0.9419 -0.6151
## alpha[7]     -1.2931 -0.9771 -0.8150 -0.6547 -0.3491
## alpha[8]     -1.5198 -1.1750 -0.9985 -0.8292 -0.5094
## alpha[9]     -1.8662 -1.4831 -1.2978 -1.1175 -0.7826
## alpha[10]    -0.8426 -0.5213 -0.3514 -0.1803  0.1485
## beta[1]       0.4971  0.9432  1.1712  1.3980  1.8263
## beta[2]       0.7823  1.2277  1.4539  1.6823  2.1076
## beta[3]       1.3506  1.7890  2.0286  2.2743  2.7777
## beta[4]       0.5945  1.0426  1.2718  1.5004  1.9321
## beta[5]       0.9224  1.3463  1.5674  1.7937  2.2328
## beta[6]       0.9312  1.3636  1.5890  1.8145  2.2675
## beta[7]       0.8895  1.3326  1.5614  1.7871  2.2391
## beta[8]       0.8987  1.3115  1.5345  1.7623  2.2013
## beta[9]       1.2354  1.6708  1.9024  2.1405  2.6141
## beta[10]      1.3123  1.7737  2.0230  2.2809  2.8091
## mu_alpha     -1.2918 -1.0383 -0.9088 -0.7810 -0.5181
## mu_beta       1.1040  1.4097  1.5604  1.7095  2.0027
## sigma_alpha2  0.1240  0.2008  0.2661  0.3602  0.6930
## sigma_beta2   0.1322  0.2238  0.3029  0.4179  0.8381
```

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

effectiveSize(window(fit.samples,start=nburns+nadapt))
```

```
##  alpha[1]     alpha[2]     alpha[3]     alpha[4]     alpha[5]     alpha[6]

##  9253.244    10994.109    14586.318    10605.914    11947.452    10723.215

##  alpha[7]     alpha[8]     alpha[9]    alpha[10]      beta[1]      beta[2]

## 13568.477    11430.825     9950.161    14452.587     8373.029    11344.810
```

```
##   beta[3]      beta[4]      beta[5]      beta[6]      beta[7]      beta[8]

## 12767.278    10483.284    11906.298    10511.628    12897.706    11148.166

##   beta[9]     beta[10]     mu_alpha      mu_beta sigma_alpha2  sigma_beta2

## 10034.051    13276.433    15370.639    11304.939    26531.926    20041.517
```

```
# Make MCMC Posteriors Data Frame from the MCMC Output
matrix = as.matrix(window(fit.samples,start=nburns+nadapt))
posteriors = as_tibble(matrix)

# Select the Posteriors for the Alphas, and Create an Alphas Boxplot
alphas <- posteriors %>%
  select(contains('alpha') & !contains('sigma'))
alphas <- pivot_longer(alphas, cols = names(alphas), names_to = "distribution
")
alphas %>%
  ggplot() +
  geom_boxplot(mapping = aes(x=distribution, y=value, fill = distribution),
               show.legend=FALSE) +
  labs(title = 'Posterior Boxplots for Alpha Components')

# Select the Posteriors for the Betas, and Create a Betas Boxplot
betas <- posteriors %>%
  select(contains('beta') & !contains('sigma'))
betas <- pivot_longer(betas, cols = names(betas), names_to = "distribution")
betas %>%
  ggplot() +
  geom_boxplot(mapping = aes(x=distribution, y=value, fill = distribution),
               show.legend=FALSE) +
  labs(title = 'Posterior Boxplots for Beta Components')

# Calculate the Log-Odds of Being Infected Given + EZK Test
logoddsvec <- alphas$value + betas$value
names_vec = c("A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "Overall")
logoddsvecnames <- rep(names_vec, length(logoddsvec)/11)
log_odds <- bind_cols(logoddsvecnames, logoddsvec)
names(log_odds) <- c("Country", "Log-Odds")

# Calculate Theta from Transformation and Create Boxplot
log_odds <- log_odds %>%
  mutate(Theta = exp(`Log-Odds`)/(1+exp(`Log-Odds`)))
log_odds %>%
  ggplot() +
  geom_boxplot(mapping = aes(x=Country, y=Theta, fill=Country),
               show.legend=FALSE) +
  labs(title = 'Boxplots for the Estimated Theta Posteriors')
```

```r
# Calculate overall mean for alpha posteriors
alphavec <- alphas %>%
  filter(distribution != 'mu_alpha')
alpha_mean <- mean(alphavec$value)

# Calculate overall mean for beta posteriors
betavec <- betas %>%
  filter(distribution != 'mu_beta')
beta_mean <- mean(betavec$value)

# Calculate Log-Odds for Testing Positive, given either EZK + or -
log_odds_0 <- alpha_mean
log_odds_1 <- alpha_mean + beta_mean

# Calculate P(Theta_Hat|X=0) and P(Theta_Hat|X=1)
exp(log_odds_0)/(1+exp(log_odds_0))
```

```
## [1] 0.2817466
```

```r
exp(log_odds_1)/(1+exp(log_odds_1))
```

```
## [1] 0.6631949
```

```r
# Calculate Log-Odds for Testing Positive, given either EZK + or - for
Country D
log_odds_0_d <- mean(posteriors$`alpha[4]`)
log_odds_1_d <- mean(posteriors$`alpha[4]` + posteriors$`beta[4]`)


# Calculate P(Theta_Hat|X=0) and P(Theta_Hat|X=1) for Country D
exp(log_odds_0_d)/(1+exp(log_odds_0_d))
```

```
## [1] 0.2268314
```

```r
exp(log_odds_1_d)/(1+exp(log_odds_1_d))
```

```
## [1] 0.5107542
```

```r
# Determine the Correct Decision Based on Decision Rule
theta_vec <- log_odds %>%
  filter(Country == 'D') %>%
  select(Theta) %>%
  as_vector()

mean(theta_vec > (457/1490))
```

```
## [1] 0.9990222
```

```r
mean(theta_vec < (457/1490))
```

```
## [1] 0.000977756
```