Using Bayseian Statistics to Analyse Kanye West's Effect on the 2020 US Elections

Candidate Number: 087074

Jaunary 2021

Question 1

Suppose that the the probability θ of voting for a candidate $j = 1 \dots K$ in all 51 US states is θ_j such that $\sum_{j=1}^{K} \theta_j = 1$. The distribution of votes in a state can be modeled as a multinomial distribution with probability mass function

$$p(y|\theta) = \frac{N!}{y_1! \dots y_K!} \theta_1^{y_1} \dots \theta_K^{y_K}$$

Assume that the probability of voting for a candidate is the same in all 51 states. The distribution of votes in each state can be multiplied together to obtain the likelihood function of votes in an election.

$$p(y|\theta) = \prod_{i=1}^{N_s} \frac{N!}{y_{i1}! \dots y_{iK}!} \theta_1^{y_{i1}} \dots \theta_K^{y_{iK}}$$

The $\frac{N!}{y_1! \dots y_K!}$ term does not vary with θ, therefore it can be treated as a constant.

$$p(y|\theta) \propto \prod_{i=1}^{N_s} \theta_1^{y_{i1}} \dots \theta_K^{y_{iK}}$$

$$p(y|\theta) \propto (\theta_1^{y_{11}} \dots \theta_1^{y_{N_s 1}}) \dots (\theta_K^{y_{11}} \dots \theta_K^{y_{N_s 1}})$$

$$p(y|\theta) \propto \theta_1^{\sum_{i=1}^{N_s} y_{ij}} \dots \theta_K^{\sum_{i=1}^{N_s} y_{iK}}$$

$$p(y|\theta) \propto \prod_{j=1}^{K} \theta_j^{\sum_{i=1}^{N_s} y_{ij}}$$

Question 2

The probability θ of voting for each candidate will not be the same in every state Therefore, we model the probabilities in a multinomial regression as functions of covariates. We have a linear predictor,

$$\eta = X\beta$$

where β is the regression coefficient and X contains the covariates. The softmax function maps η to the multinomial probabilities θ. The probability that a random voter votes candidate k is

$$\theta_{sk} = \frac{e^{\eta_{sk}}}{\sum_i \eta_{si}}$$

To ensure the model is identifiable, we fix η_{i1} to be 0 for all i.

The probability of voting for one of the candidates in each state, s, is 1.

$$\sum_{k=1}^{K} \theta_{sk} = 1$$

$$\sum_{k=1}^{K} \theta_{sk} = \frac{e^{\eta_{s1}}}{\sum_i \eta_{si}} + \cdots + \frac{e^{\eta_{sK}}}{\sum_i \eta_{si}}$$

$$\sum_{k=1}^{K} \theta_{sk} = \frac{\sum_{k=1}^{K} e^{\eta_{sk}}}{\sum_{i=1}^{K} e^{\eta_{si}}}$$

This implies that

$$\theta_{sk} = \frac{e^{\eta_{sk}}}{\sum_{i=1}^{K} e^{\eta_{si}}}$$

$$\theta_{s1} = \frac{e^{\eta_{s1}}}{\sum_{i=1}^{K} e^{\eta_{si}}}$$

Where $\eta_{s1} = 0$ for all s, therefore

$$\theta_{s1} = \frac{1}{\sum_{i=1}^{K} e^{\eta_{si}}}$$

$$\frac{\theta_{sk}}{\theta_{s1}} = \frac{\frac{e^{\eta_{s1}}}{\sum_{i=1}^{K} e^{\eta_{si}}}}{\frac{1}{\sum_{i=1}^{K} e^{\eta_{si}}}}$$

$$\frac{\theta_{sk}}{\theta_{s1}} = e^{\eta_{sk}}$$

$$\log\left(e^{\eta_{sk}}\right) = \log\left(\frac{\theta_{sk}}{\theta_{s1}}\right)$$

$$\eta_{sk} = \log\left(\frac{\theta_{sk}}{\theta_{s1}}\right)$$

Question 3

We can use Bayes Theorem to calculate the posterior distribution of β.

$$p(\beta|\mathbf{y}) = \frac{p(\mathbf{y}|\theta)\pi(\beta)}{p(\mathbf{y})}$$

$$p(\beta|\mathbf{y}) \propto p(\mathbf{y}|\beta)\pi(\beta)$$

The prior distribution β is $\beta_{i1} = 0$ for all i and the other values of the β matrix have independent normal distributions with constant variance σ^2.

$$\pi(\beta) = \frac{1}{\sigma\sqrt{2\pi}} exp\{-\frac{1}{2}(\frac{\beta - \mu}{\sigma^2})\}$$

We calculate the likelihood function of β, $p(\mathbf{y}|\beta)$ by applying the change of variable formula to $p(\mathbf{y}|\theta)$

$$f_\beta(\beta) = |\frac{\partial g^{-1}(\beta)}{\partial \beta}| f_\theta(g^{-1}(\beta))$$

We calculate the function $\theta = g^{-1}(\beta)$

$$\eta_{sk} = X_s B_k$$

$$log(\frac{\theta_{sk}}{\theta_{s1}}) = X_s B_k$$

where

$$\theta_{s1} = \frac{e^{\eta_{s1}}}{\sum_i e^{\eta_{si}}} = \frac{1}{\sum_i e^{\eta_{si}}}$$

$$log(\theta_{sk} \sum_i e^{\eta_{si}}) = X_s B_k$$

$$g^{-1}(\beta) = \theta_{sk} = \frac{e^{X_s B_k}}{\sum_i e^{\eta_{si}}}$$

We use this equation to calculate the change of variable formula,

$$|\frac{\partial g^{-1}(\beta)}{\partial \beta}| = \frac{X_s e^{X_s B_k}}{\sum_i e^{\eta_{si}}}$$

$$f_\theta(g^{-1}(\beta)) = \prod_{j=1}^{K} (\frac{e^{X_s B_k}}{\sum_i e^{\eta_{si}}})^{\sum_{i=1}^{N_s} y_{ij}}$$

$$p(\mathbf{y}|\beta) = \frac{X_s e^{X_s B_k}}{\sum_i e^{\eta_{si}}} \prod_{j=1}^{K} (\frac{e^{X_s B_k}}{\sum_i e^{\eta_{si}}})^{\sum_{i=1}^{N_s} y_{ij}}$$

By substituting this into the formula for the posterior, we obtain

$$p(\beta|\mathbf{y}) = \frac{X_s e^{X_s B_k}}{\sum_i e^{\eta_{si}}} \prod_{j=1}^{K} (\frac{e^{X_s B_k}}{\sum_i X_{sj} B_{ji}})^{\sum_{i=1}^{N_s} y_{ij}} \frac{1}{\sigma\sqrt{2\pi}} exp\{-\frac{1}{2}(\frac{\beta - \mu}{\sigma^2})\}$$

$$p(\beta|\mathbf{y}) = A \prod_{j=1}^{K} (\frac{e^{X_s B_k}}{\sum_i X_{sj} B_{ji}})^{\sum_{i=1}^{N_s} y_{ij}} exp\{-\frac{1}{2}(\frac{\beta}{\sigma^2})\}$$

$$\log p(\beta|\mathbf{y}) = \log(A \prod_{j=1}^{K} (\frac{e^{X_s B_k}}{\sum_i X_{sj} B_{ji}})^{\sum_{i=1}^{N_s} y_{ij}} e^{-\frac{\beta}{2\sigma^2}})$$

$$\log p(\beta|\mathbf{y}) = C + \log(\prod_{j=1}^{K} (\frac{e^{X_s B_k}}{\sum_i X_{sj} B_{ji}})^{\sum_{i=1}^{N_s} y_{ij}} e^{-\frac{\beta}{2\sigma^2}})$$

$$\log p(\beta|\mathbf{y}) = C + \log(e^{-\frac{\beta}{2\sigma^2}}) + \sum_{i=1}^{N_s} y_{ij} \log(\frac{e^{X_s B_k}}{\sum_i X_{sj} B_{ji}})$$

$$\log p(\beta|\mathbf{y}) = C + -\frac{1}{2\sigma^2} \sum_{i=1}^{p} \sum_{j=2}^{K} \beta_{ij}^2 + \sum_{s=1}^{N_s} \sum_{k=1}^{K} y_{sk}((\log(e^{X_s B_k}) - \log(\sum_i X_{sj} B_{ji}))$$

$$\log p(\beta|\mathbf{y}) = C + -\frac{1}{2\sigma^2} \sum_{i=1}^{p} \sum_{j=2}^{K} \beta_{ij}^2 + \sum_{s=1}^{N_s} \sum_{k=1}^{K} y_{sk}(\sum_{j=1} X_{sj} B_{jk}) - \sum_i exp\{\sum_j X_{sj} B_{ji}\})$$

where C is a constant that we can ignore during sampling

Question 4

I will sample from the log posterior distribution of β, $p(\beta|\mathbf{y})$ by using the Markov Chain Monte Carlo method.

The rds file "USelectionData.rds" stores data on the 2020 US elections.

```
USelectionData <- readRDS("USelectionData.rds") # dataset
```

I have transformed the raw data into a dataset that is in a more useful format for analysis. The dataset only contains data on the candidates that I will be analysing.

```
# candidates used in analyses
MyCandidates <- c("Donald Trump", "Joe Biden", "Jo Jorgensen", "Kanye West")
USelectionOthers <- group_by(USelectionData, State) %>%
  mutate(Others = sum(TotalVotes[!(Candidate %in% MyCandidates)])) %>% ungroup()
USelectionOthers <- filter(USelectionOthers, Candidate=="Donald Trump")
USelectionOthers <- mutate(USelectionOthers, Party = "OTH", TotalVotes=Others,
                           Candidate="Others")
USelectionOthers <- dplyr::select(USelectionOthers, -Others)
USelectionDataNew <- rbind(USelectionData, USelectionOthers)
# create a dataset using only the key candidates
USelectionDataNew <- filter(USelectionDataNew, Candidate
                            %in% c(MyCandidates, "Others")) %>% arrange(State)
```

The statistics on the sex, race, income level and occupation of states are all in percentages, whereas Covid-19 cases and deaths are not. It is important that the variables are in the same order of magnitude to ensure an accurate analysis, therefore I have transformed the total number of cases and deaths of Covid-19 per state into the percentages of cases and deaths of Covid-19 for each state.

```
USelectionDataNew$CasesPerPopulationPercent <-
  (USelectionDataNew$Cases/USelectionDataNew$TotalPop)*100 # times a hundred for percent
USelectionDataNew$DeathsPerPopulationPercent <-
  (USelectionDataNew$Deaths/USelectionDataNew$TotalPop)*100
USelectionDataNew$EmploymentPercent <-
  (USelectionDataNew$Employed/USelectionDataNew$TotalPop)*100
```

There are two matrices in the log posterior that are constant, X and y. X is a Ns x p (states x features of model) matrix that contains the covariates. y is a Ns x K (number of states x candidates) matrix that contains the vote share outcome of the election. These matrices change depending on the features are chosen for analysis. Therefore, I have created functions that, given the features that are chosen, produce the X and y matrices.

```
ProduceX <- function(df, features){ # Creating X: Ns x p (states x features of model)
  ModelData <- df %>% select(features)
  ModelData2 <- unique(ModelData) # This ensures that one value is taken from each state
  X <- as.matrix(sapply(ModelData2, as.numeric))
  return(X)
}

ProduceY <- function(df){  # Create y: Ns x K (number of states x candidates)
  ModelData3 <- USelectionDataNew %>% select(State, Candidate, TotalVotes)
  # Create datasets that contain the number of votes for each candidate
  DT <- ModelData3[ModelData3$Candidate %in% c("Donald Trump"), ]
```

```
  JB <- ModelData3[ModelData3$Candidate %in% c("Joe Biden"), ]
  JJ <- ModelData3[ModelData3$Candidate %in% c("Jo Jorgensen"), ]
  OT <- ModelData3[ModelData3$Candidate %in% c("Others"), ]
  KW <- ModelData3[ModelData3$Candidate %in% c("Kanye West"), ]

  # Merge the datasets together
  DTJB <- merge(x = DT, y = JB, by = "State", all.x=TRUE)
  DTJBJJ <- merge(x = DTJB, y = JJ, by = "State", all.x=TRUE)
  # set column names to avoid confusion
  colnames(DTJBJJ) <- c("State", "Donald", "Donald_Vote", "Joe", "Joe_Vote",
                        "Jo", "Jo_Vote")
  DTJBJJOT <- merge(x = DTJBJJ, y = OT, by = "State", all.x=TRUE)
  DTJBJJOTKW <- merge(x = DTJBJJOT, y = KW, by = "State", all.x=TRUE)
  colnames(DTJBJJOTKW) <- c("State", "Donald", "Donald_Vote", "Joe", "Joe_Vote",
                            "Jo", "Jo_Vote", "Other", "Other_Vote", "Kanye",
                            "Kanye_Vote")
  # drop the columns that do not contain vote shares
  drops <- c("State", "Donald", "Joe", "Jo", "Other","Kanye")
  Tester <- DTJBJJOTKW[ , !(names(DTJBJJOTKW) %in% drops)]
  # set the vote share for states that Kanye West did not run to 0
  Tester[is.na(Tester)] <- 0
  y <- as.matrix(sapply(Tester, as.numeric))
  return(y)
}
```

The prior values of the β matrix are $\beta_{i1} = 0$ for all i, and the other entries have independent Normal priors with constant variance σ^2. β is a p X K (no. of features x no. of candidates) matrix, therefore each each feature adds 4 parameters to the model.

Here I create the initial parameter values of the model, the number of which depend on the number of features chosen. The Normal prior distributions have a mean of 0 to ensure that they are unbiased and a variance of 5, this is sufficiently large on the log scale to allow most types of effects.

```
ProduceParams <- function(df, features){
  ModelData <- df %>% select(features)
  ModelData2 <- unique(ModelData)
  p <- ncol(ModelData2)  # count the number of features, p
  K = 5 # the number of candidates is always 5
  mu <- 0
  sigma2 <- 5
  # I create p x 4 parameters with N(0,5)
  params <- rnorm((p*K-p),mean=0,sd=sqrt(5))
  return(params)
}
```

The log posterior of β, $p(\beta|\mathbf{y})$ is the distribution that I will be sampling from using the Markov Chain Monte Carlo algorithm. I have coded the log posterior distribution into a function that takes in the X and y matrices along with the parameters to calculate a value for the posterior.

```
LogPosterior <- function(X=initial_X, y=initial_y, params){
  # Create B using the parameter values
  shaper <- dim(X)[2]
  shaper
```

```
  params_shaped <- matrix(params, nrow = shaper, byrow = TRUE)
  zero_col <- rep(0, shaper)
  B <- cbind(zero_col, params_shaped)

  # This is the posterior distribution, created in two parts (Left and Right)
  C <- X %*% B
  C_vec <- colSums(exp(C), na.rm = FALSE, dims = 1)
  inside2 <- C - log(C_vec)
  Right <- sum(y*inside2)

  mu <- 0
  sigma2 <- 5
  sum_B <- sum(B*B)
  Left <- (-1/(2*sigma2))*sum_B

  Posterior <- abs(Left + Right)
  return(Posterior)
}
```

The parameter values are changed slightly in each iteration of the Markov Chain Monte Carlo algorithm. To do this, I proposed a random walk on each of the parameter values. The random walk is a Normal distribution with $\mu = 0$ and $\sigma^2 = 5$.

```
Proposal <- function(params, sigma2=0.25){
  K <- length(params)
  nudge <- rnorm(K,0,sigma2) # create the random walk values
  new_params <- params + nudge # add the random walks to the parameters
  return(new_params)
}
```

This function calculates the ratio between the posterior value of the new value and the posterior value of the old value.

```
MHratio <- function(paramold, paramnew, X=initial_X, y=initial_y){
  ratio <- LogPosterior(X=initial_X, y=initial_y, paramnew)/
    LogPosterior(X=initial_X,
  y=initial_y, paramold)
  return(ratio)
}
```

This function is my Monte Carlo Markov Chain sampler. Here I use the functions that I have previously made to sample from the posterior distribution using the MCMC algorithm. To run, the function requires the number of iterations to do and the parameter values. The output is a list of samples from the posterior distributions.

```
my_MCMC <- function(Niter, start_values){
  tNames <- c('1','2','3','4','5','6','7','8') # names of parameters
  # initialise an empty matrix to append values to
  mcmc.out <- matrix(NA, nrow=Niter+1, ncol=length(start_values))
  mcmc.out[1,] <- unlist(start_values)
  colnames(mcmc.out) <- tNames
  for (k in 2:(Niter+1)){
```

```
    paramsafter <- Proposal(paramsbefore) # calculate new parameter values
    mhr <- MHratio(paramold=paramsbefore, paramnew=paramsafter) # MH ratio
    rstart <- min(c(1,mhr))
    U <- runif(1,0,1)
    if(U<rstart){ # Accept/Reject step
      mcmc.out[k,] <- unlist(paramsafter)
    }
    else{
      mcmc.out[k,] <- mcmc.out[k-1,]
    }
  }
  as.mcmc.list(as.mcmc(mcmc.out)) # return a list of parameter samples
}
```

Question 5

The MCCM function that I reated allows me to choose features of the dataset that I think will have an impact on the number of votes Kanye West obtains in a state. I have a very limited knowledge of Kanye West's voter base and online material provides little information on the matter. Therfore I have run multiple models that include different parameters in order to see which parameters have the largest effect.

Each feature I include adds another 4 parameters to the model. Therefore, I have restricted the number of features in each model to 2. This is to ensure that the models have enough degrees of freedom so that the data suitably swamps the prior distributions.

I create 5 models, therefore analysing the effect of 10 different features. For each model, I sample from the posterior distribution using my MCMC sampler, evaluate traceplots and produce histograms of the posterior distributions. Using data on the posterior distributions, I create Monte Carlo estimates of the parameters and form the β matrix. I then calculate θ and have displayed the θ produced by the last model.

Black and Professional

Here I analyse the effect that the percentage of black people and professionals in a state has on voter distribution.

```
features <- c('Black', 'Professional') # chosen features
N <- length(features)
candidates <- c("Donald Trump", "Joe Biden", "Jo Jorgensen", "Others", "Kanye West")
# create X and y values
initial_X <- ProduceX(USelectionDataNew, features)
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(features)` instead of `features` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

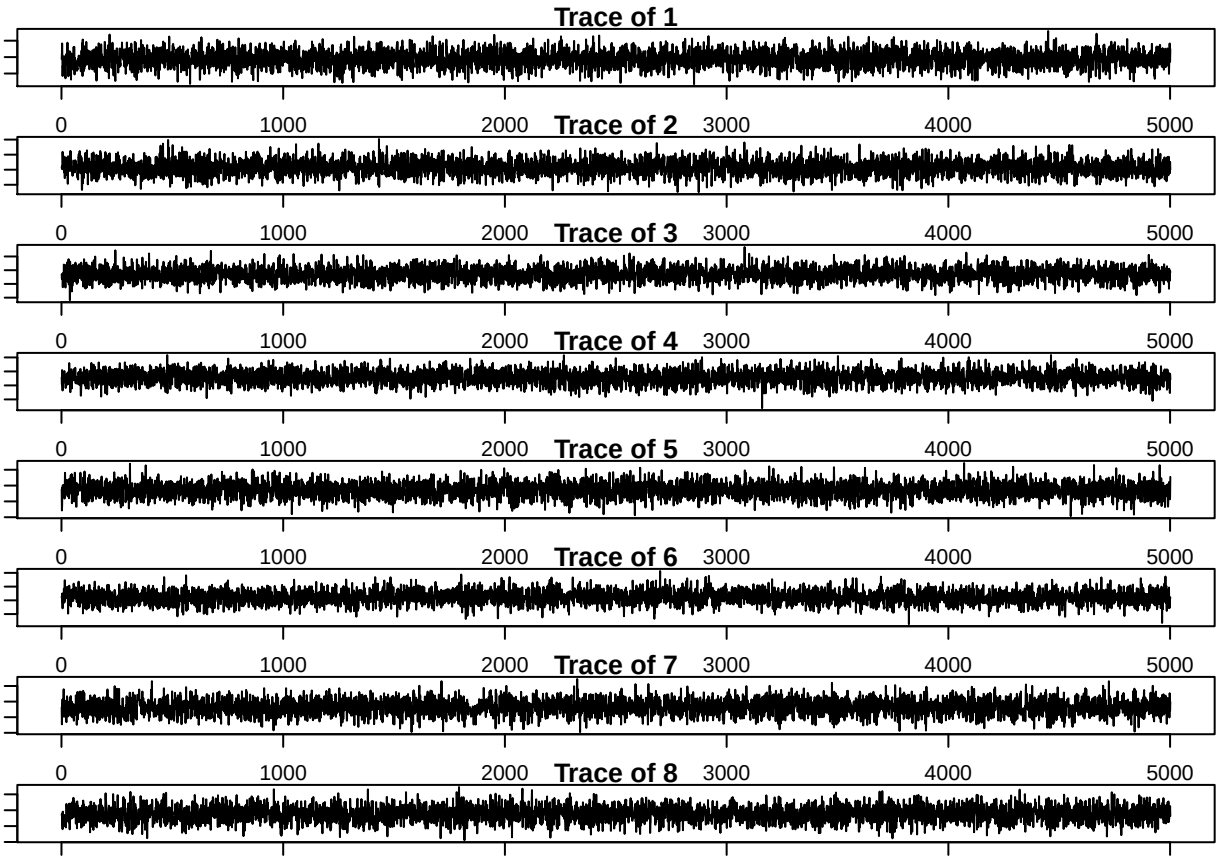
```
initial_y <- ProduceY(USelectionDataNew)
# initialise the parameter values
initial_params <- ProduceParams(USelectionDataNew, features)
paramsbefore <- initial_params
```

I use my MCMC sampler to sample 5000 times from the posterior distribution.

```
N_iterations <- 5000    # sample from the my_MCMC function
Samples <- my_MCMC(Niter=N_iterations, start_values = initial_params)
```

I use the sample data on the posterior distributions to create traceplots for the parameters. The traceplots are well mixed, indicating that the model has converged.

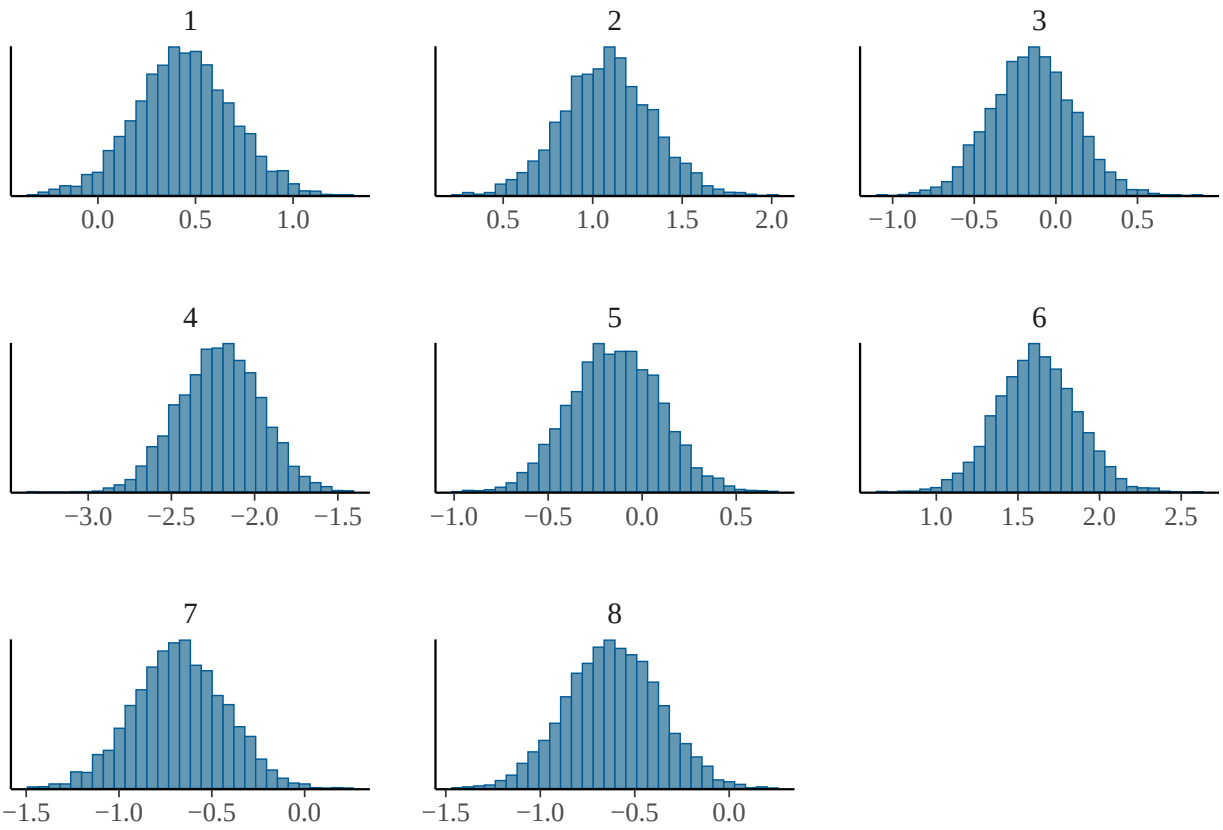
```
par(mfrow=c(8,1), mar=c(1,1,1,1))
traceplot(Samples) # draw the traceplot for all 8 parameters
```



Here I display histograms of the data on the posterior distributions. The histograms appear to be normally distributed with similar variances and different means.

```
ParameterData <- as.matrix(Samples)
mcmc_hist(ParameterData) # plot histograms of posterior distribution samples
```

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



I have calculated Monte Carlo Estimates of the parameters using the data on the posterior distributions in order to establish an estimate of β.

```
# apply the MC estimate formula
Param_sum <- colSums(ParameterData)
Param_n <- rep(N_iterations, length(Param_sum))
MC_estimate <- Param_sum/Param_n # divided the sum of each parameter samples by 2000
# Create B from MC estimates
B <- cbind(matrix(rep(0, N), N, 1), t(matrix(MC_estimate, ncol = N)))
colnames(B) <- candidates
rownames(B) <- features
B
```

```
##              Donald Trump  Joe Biden Jo Jorgensen     Others Kanye West
## Black                   0  0.4406908     1.079638 -0.1432130 -2.2105947
## Professional            0 -0.1432647     1.614817 -0.6751322 -0.6157547
```

I now calculate the θ matrix by applying the formula $\theta_{sk} = \frac{e^{\eta_{sk}}}{\sum_i \eta_{si}}$.

```
n <- initial_X %*% B
theta <- exp(n)/rowSums(exp(n))
```

Poverty and Office

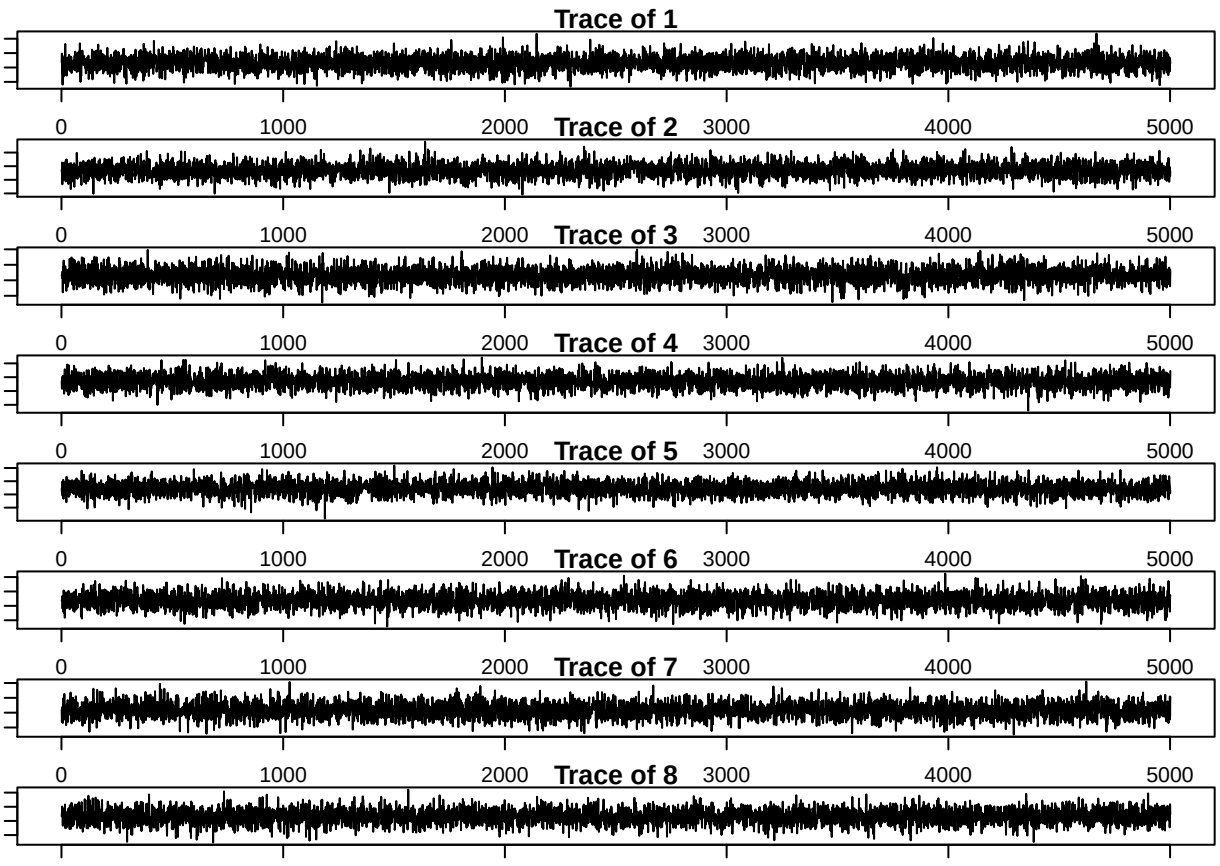
I repeat my analysis using two different features. The features that I have chosen is the poverty level is states and the percentage of office workers.

```
features <- c('Poverty', 'Office') # chosen features
N <- length(features)
candidates <- c("Donald Trump", "Joe Biden", "Jo Jorgensen", "Others", "Kanye West")
# create X and y values
initial_X <- ProduceX(USelectionDataNew, features)
initial_y <- ProduceY(USelectionDataNew)
# initialise the parameter values
initial_params <- ProduceParams(USelectionDataNew, features)
paramsbefore <- initial_params
```

```
N_iterations <- 5000 # sample from the my_MCMC function
Samples <- my_MCMC(Niter=N_iterations, start_values = initial_params)
```

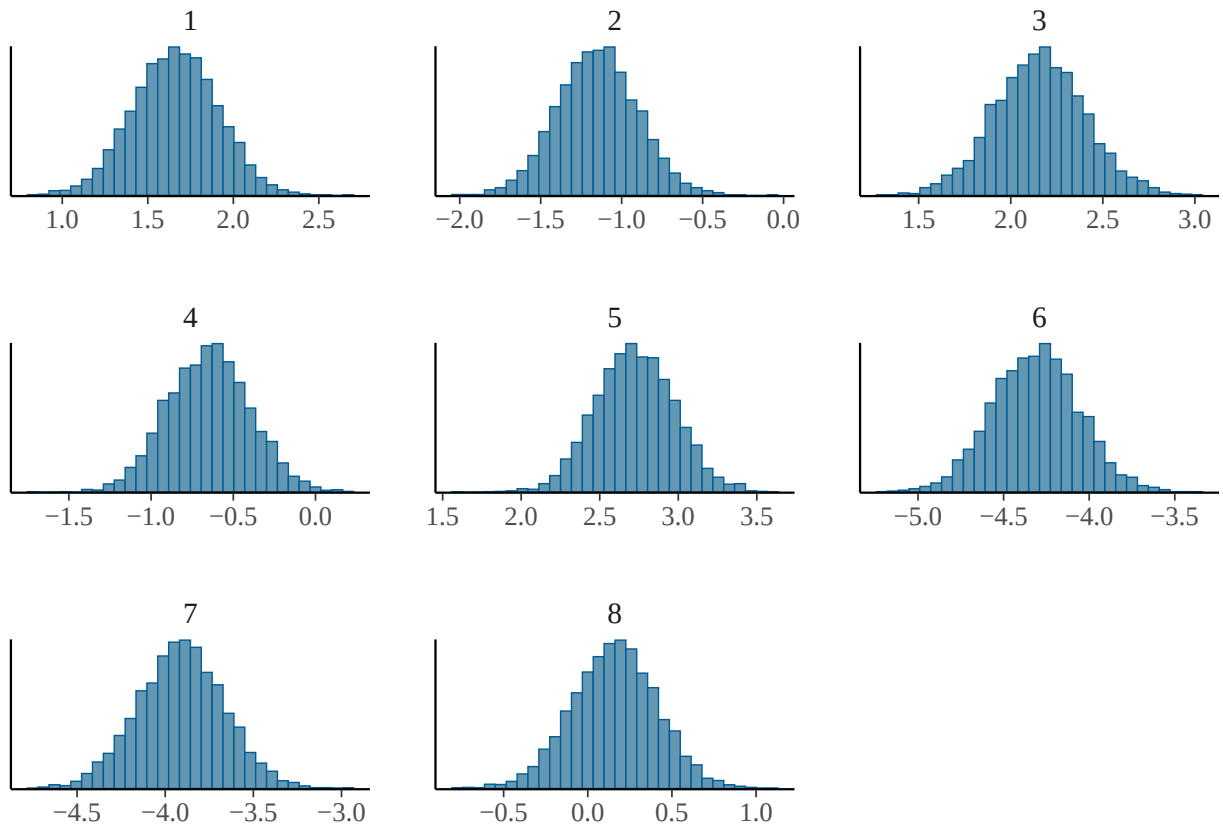
The well mixed traceplots imply that the model has converged.

```
par(mfrow=c(8,1), mar=c(1,1,1,1))
traceplot(Samples)
```




```
ParameterData <- as.matrix(Samples)
mcmc_hist(ParameterData) # plot histograms of posterior distribution samples
```

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



I have obtained estimates for β using the parameters Poverty and Office.

```
Param_sum <- colSums(ParameterData)
Param_n <- rep(N_iterations, length(Param_sum))
MC_estimate <- Param_sum/Param_n # divided the sum of each parameter samples by 2000
# Create B from MC estimates
B <- cbind(matrix(rep(0, N), N, 1), t(matrix(MC_estimate, ncol = N)))
colnames(B) <- candidates
rownames(B) <- features
B
```

```
##         Donald Trump Joe Biden Jo Jorgensen    Others Kanye West
## Poverty            0  1.662209    -1.150816  2.157354 -0.6490268
## Office             0  2.726682    -4.313340 -3.911697  0.1586585
```

```
n <- initial_X %*% B
theta <- exp(n)/rowSums(exp(n))
```

Asian and Construction

I repeat my analysis using the features Asian and Construction.

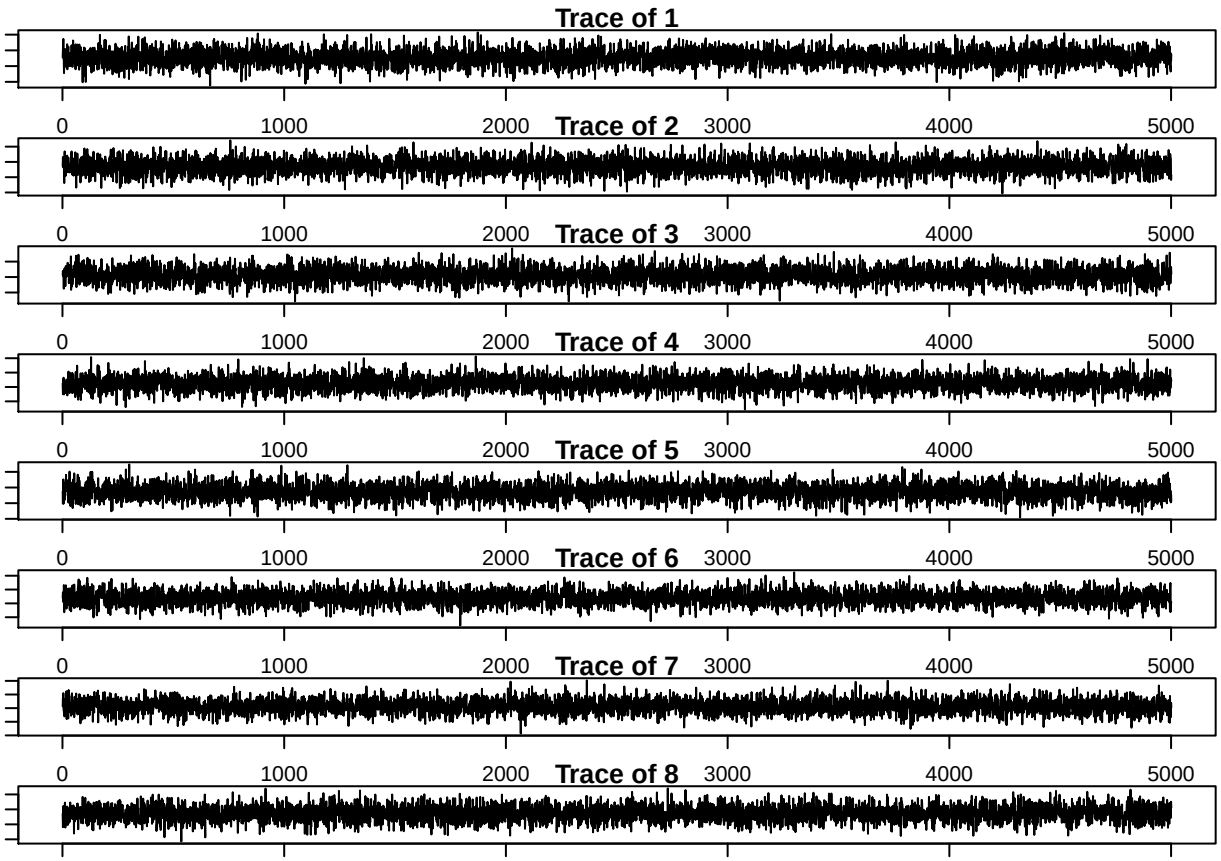
```
features <- c('Asian', 'Construction') # chosen features
N <- length(features)
candidates <- c("Donald Trump", "Joe Biden", "Jo Jorgensen", "Others", "Kanye West")
```

```
# create X and y values
initial_X <- ProduceX(USelectionDataNew, features)
initial_y <- ProduceY(USelectionDataNew)
# initialise the parameter values
initial_params <- ProduceParams(USelectionDataNew, features)
paramsbefore <- initial_params
```

```
N_iterations <- 5000 # sample from the my_MCMC function
Samples <- my_MCMC(Niter=N_iterations, start_values = initial_params)
```

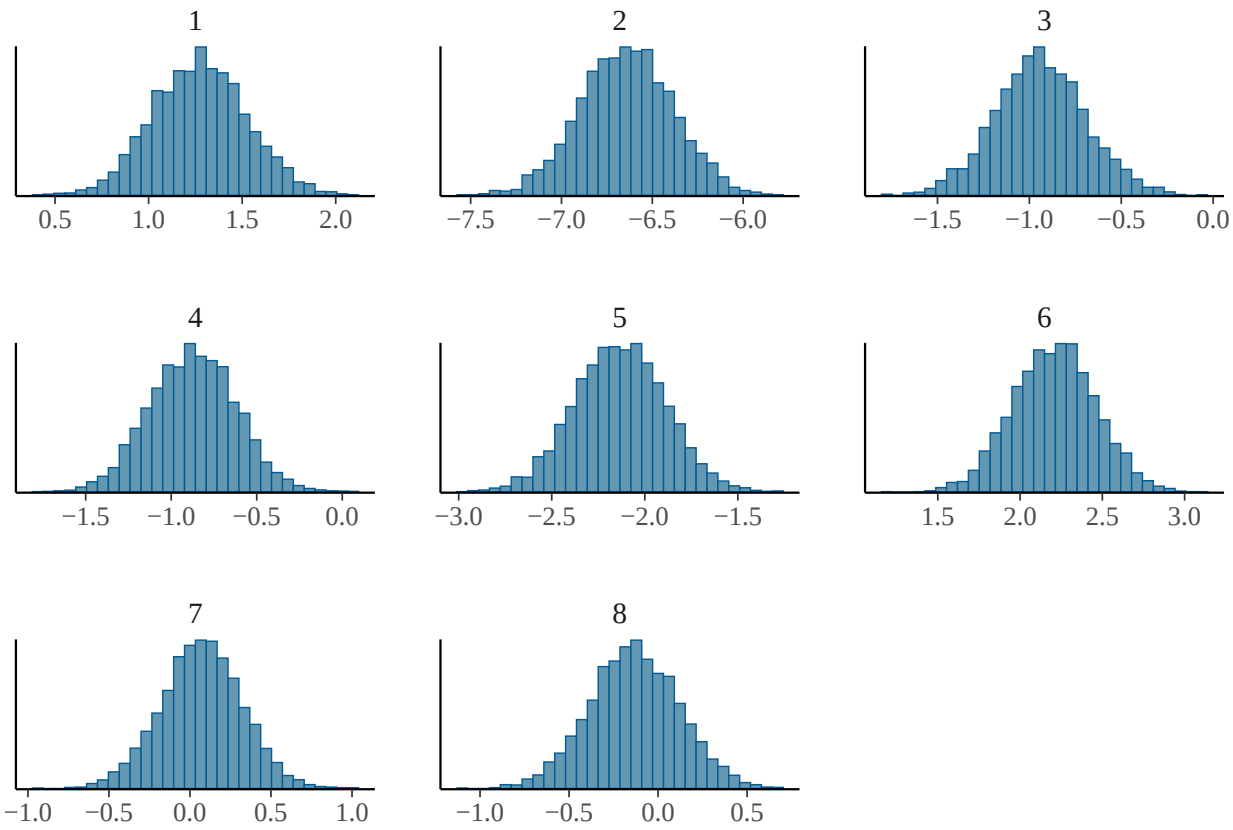
The well mixed traceplots imply that the model has converged.

```
par(mfrow=c(8,1), mar=c(1,1,1,1))
traceplot(Samples) # draw the traceplot for all 8 parameters
```




```
ParameterData <- as.matrix(Samples)
mcmc_hist(ParameterData) # plot histograms of posterior distribution samples
```

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



I have obtained estimates for β using the parameters Asian and Construction.

```
Param_sum <- colSums(ParameterData)
Param_n <- rep(N_iterations, length(Param_sum))
MC_estimate <- Param_sum/Param_n # divided the sum of each parameter samples by 2000
# Create B from MC estimates
B <- cbind(matrix(rep(0, N), N, 1), t(matrix(MC_estimate, ncol = N)))
colnames(B) <- candidates
rownames(B) <- features
B
```

```
##              Donald Trump Joe Biden Jo Jorgensen      Others Kanye West
## Asian                   0  1.273623    -6.643781 -0.94073379 -0.8738541
## Construction            0 -2.135175     2.207626  0.07123638 -0.1334380
```

```
n <- initial_X %*% B
theta <- exp(n)/rowSums(exp(n))
```

CasesPerPopulationPercent and DeathsPerPopulationPercent

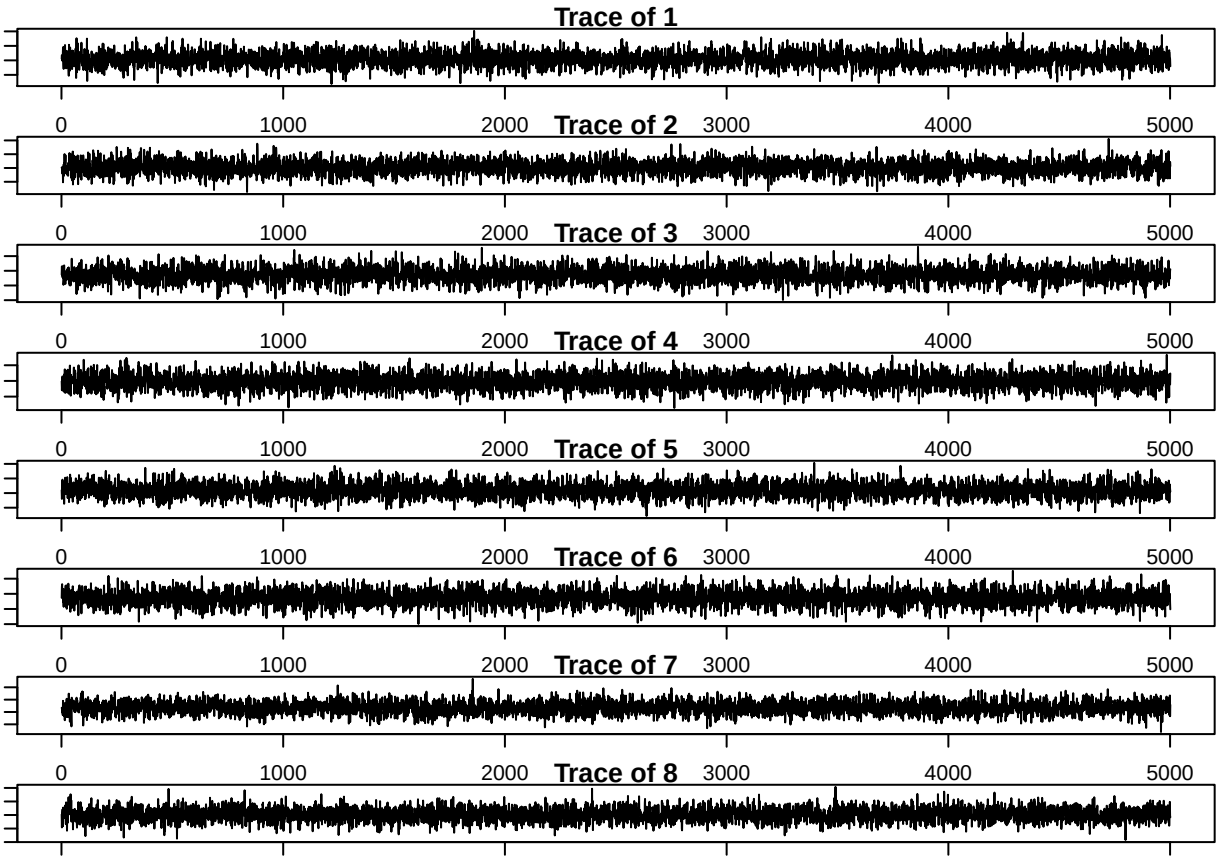
I repeat my analysis using the features CasesPerPopulationPercent and DeathsPerPopulationPercent to analyse the effect that Covid-19 had on Kanye West's number of votes in states.

```
# chosen features
features <- c("CasesPerPopulationPercent", "DeathsPerPopulationPercent")
N <- length(features)
candidates <- c("Donald Trump", "Joe Biden", "Jo Jorgensen", "Others", "Kanye West")
# create X and y values
initial_X <- ProduceX(USelectionDataNew, features)
initial_y <- ProduceY(USelectionDataNew)
# initialise the parameter values
initial_params <- ProduceParams(USelectionDataNew, features)
paramsbefore <- initial_params
```

```
N_iterations <- 5000 # sample from the my_MCMC function
Samples <- my_MCMC(Niter=N_iterations, start_values = initial_params)
```

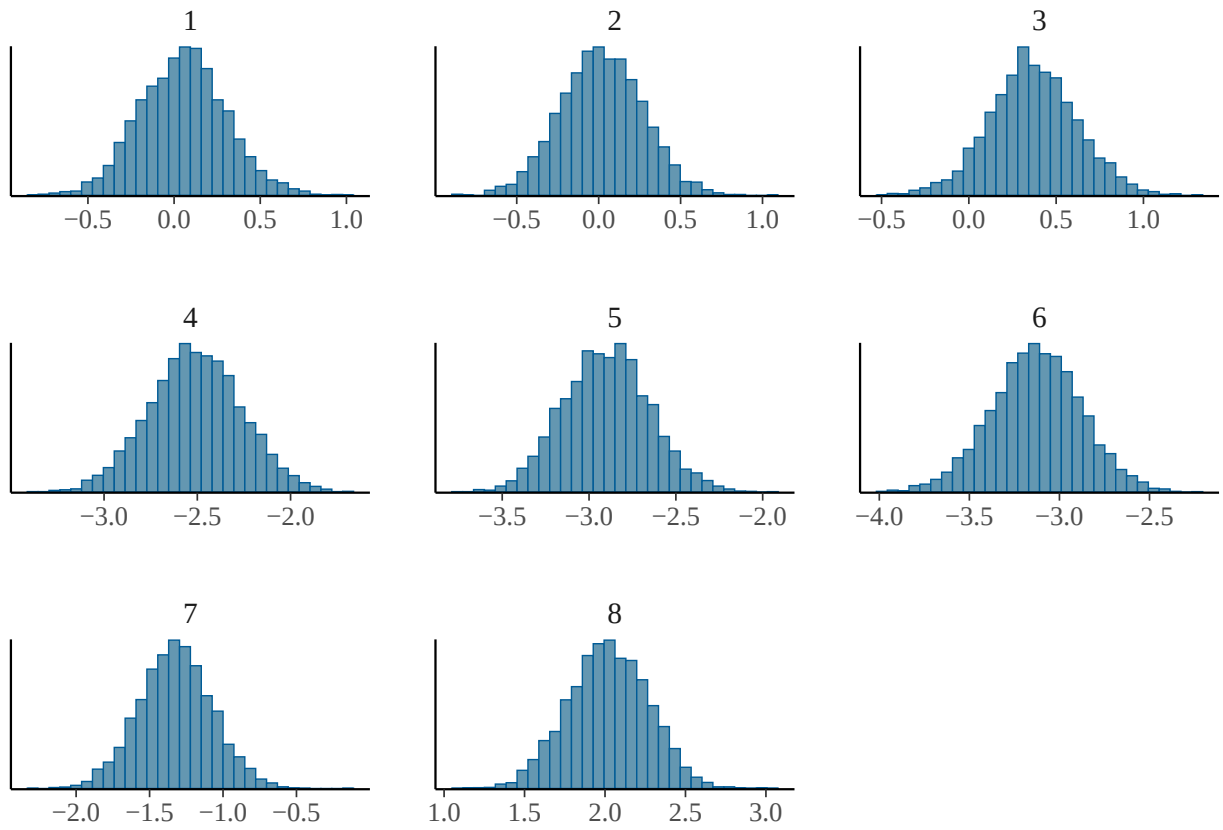
The well mixed traceplots imply that the model has converged.

```
par(mfrow=c(8,1), mar=c(1,1,1,1))
traceplot(Samples) # draw the traceplot for all 8 parameters
```




```
ParameterData <- as.matrix(Samples)
mcmc_hist(ParameterData) # plot histograms of posterior distribution samples
```

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



I have obtained estimates for β using the parameters Asian and Construction.

```
Param_sum <- colSums(ParameterData)
Param_n <- rep(N_iterations, length(Param_sum))
MC_estimate <- Param_sum/Param_n # divided the sum of each parameter samples by 2000
# Create B from MC estimates
B <- cbind(matrix(rep(0, N), N, 1), t(matrix(MC_estimate, ncol = N)))
colnames(B) <- candidates
rownames(B) <- features
B
```

```
##                              Donald Trump   Joe Biden Jo Jorgensen     Others
## CasesPerPopulationPercent               0  0.05073199   0.01858085  0.3676444
## DeathsPerPopulationPercent              0 -2.90055391  -3.12987427 -1.3240128
##                              Kanye West
## CasesPerPopulationPercent    -2.510333
## DeathsPerPopulationPercent    2.019988
```

```
n <- initial_X %*% B
theta <- exp(n)/rowSums(exp(n))
```

EmploymentPercent and Service

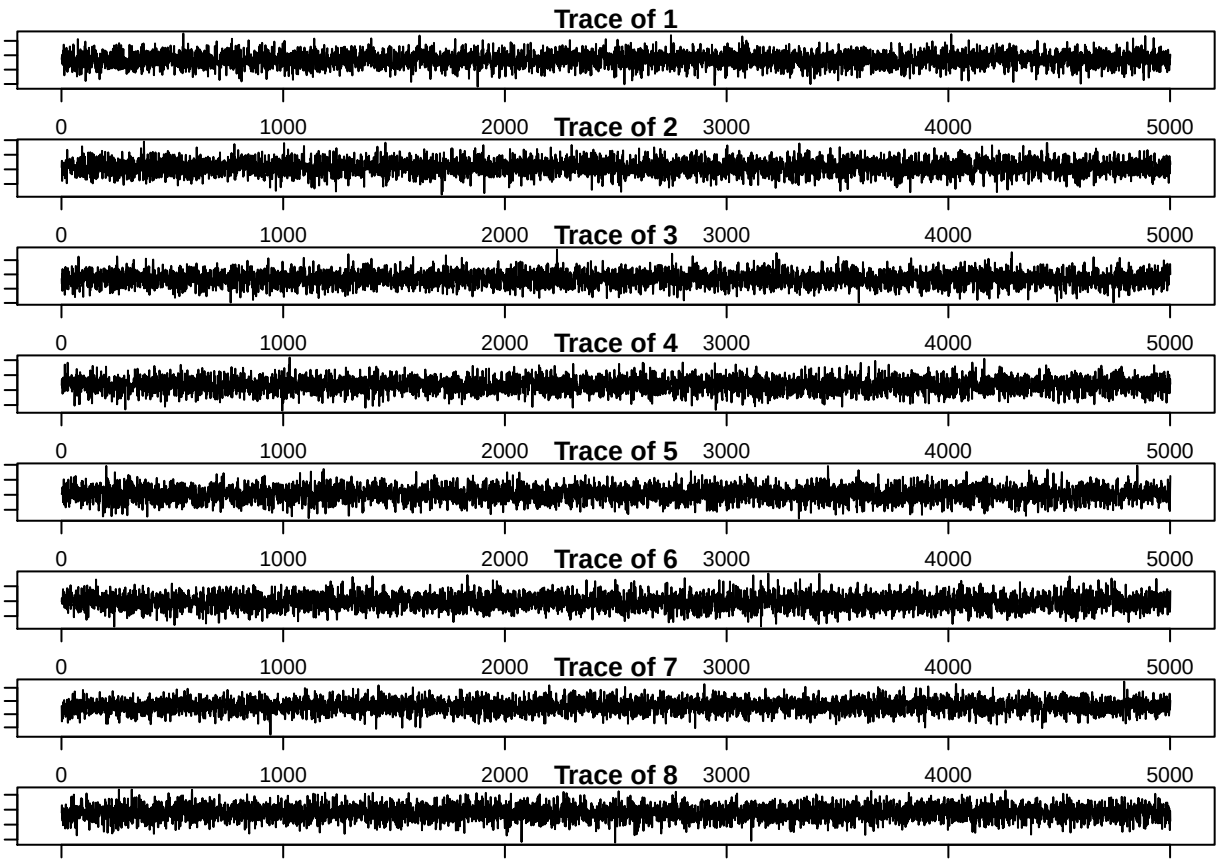
I repeat my analysis using the features EmploymentPercent and Service.

```
features <- c("EmploymentPercent", "Service") # chosen features
N <- length(features)
candidates <- c("Donald Trump", "Joe Biden", "Jo Jorgensen", "Others", "Kanye West")
# create X and y values
initial_X <- ProduceX(USelectionDataNew, features)
initial_y <- ProduceY(USelectionDataNew)
# initialise the parameter values
initial_params <- ProduceParams(USelectionDataNew, features)
paramsbefore <- initial_params
```

```
N_iterations <- 5000 # sample from the my_MCMC function
Samples <- my_MCMC(Niter=N_iterations, start_values = initial_params)
```

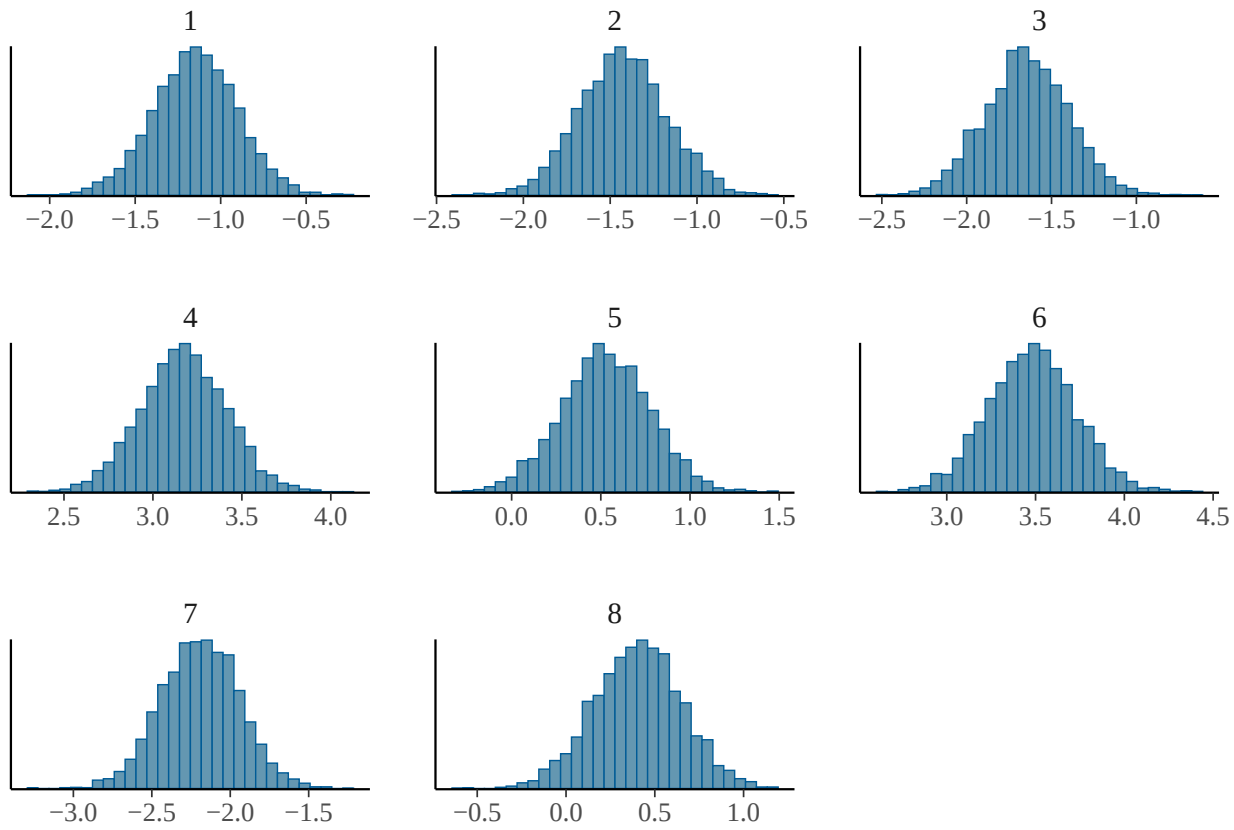
The well mixed traceplots imply that the model has converged.

```
par(mfrow=c(8,1), mar=c(1,1,1,1))
traceplot(Samples) # draw the traceplot for all 8 parameters
```




```
ParameterData <- as.matrix(Samples)
mcmc_hist(ParameterData) # plot histograms of posterior distribution samples
```

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



I have obtained estimates for β using the parameters EmploymentPercent and Service

```
Param_sum <- colSums(ParameterData)
Param_n <- rep(N_iterations, length(Param_sum))
MC_estimate <- Param_sum/Param_n # divided the sum of each parameter samples by 2000
# Create B from MC estimates
B <- cbind(matrix(rep(0, N), N, 1), t(matrix(MC_estimate, ncol = N)))
colnames(B) <- candidates
rownames(B) <- features
B
```

```
##                    Donald Trump  Joe Biden Jo Jorgensen    Others Kanye West
## EmploymentPercent             0 -1.1491112    -1.427410 -1.645991  3.1724263
## Service                       0  0.5284507     3.480629 -2.190983  0.4032891
```

This time, I have displayed the θ matrix. This shows the probability of voting for each of the candidates in the different states.

```
n <- initial_X %*% B
theta <- exp(n)/rowSums(exp(n))
theta
```

```
##       Donald Trump     Joe Biden Jo Jorgensen        Others Kanye West
##  [1,] 2.407351e-61  2.166784e-78 5.904309e-62 3.166537e-107          1
##  [2,] 9.396312e-70  1.245904e-89 5.508112e-73 1.260555e-120          1
##  [3,] 1.204853e-62  2.991125e-79 6.842655e-59 7.023442e-112          1
```

```
##  [4,] 3.207735e-62  1.727210e-79 1.027916e-62 7.854191e-109          1
##  [5,] 5.393698e-66  3.884365e-84 1.652738e-65 5.162853e-116          1
##  [6,] 1.692742e-72  2.205041e-93 4.872697e-77 9.383754e-125          1
##  [7,] 3.483814e-72  7.339888e-93 4.565450e-76 1.474319e-124          1
##  [8,] 1.771705e-68  1.323577e-87 1.440378e-69 1.545721e-119          1
##  [9,] 2.805455e-75  8.092979e-98 1.382022e-84 4.482795e-127          1
## [10,] 2.862081e-64  2.900480e-81 3.645390e-60 6.226763e-115          1
## [11,] 3.651082e-64  3.523606e-82 9.179245e-66 1.168689e-111          1
## [12,] 8.901645e-69  9.509403e-87 3.848264e-63 1.053689e-123          1
## [13,] 6.792140e-65  7.347219e-83 4.065714e-65 1.085028e-113          1
## [14,] 6.438225e-69  1.582729e-88 5.819111e-72 2.963272e-119          1
## [15,] 1.807717e-67  1.161960e-86 1.918147e-70 9.695493e-117          1
## [16,] 1.017492e-73  2.756137e-95 4.126602e-80 6.640475e-126          1
## [17,] 2.173981e-70  1.016908e-90 4.099387e-75 6.137015e-121          1
## [18,] 5.431556e-63  1.309625e-80 3.281235e-64 8.472363e-110          1
## [19,] 3.664554e-64  1.617296e-81 3.517913e-62 1.355759e-113          1
## [20,] 3.261407e-71  2.543143e-91 1.750787e-73 1.018309e-123          1
## [21,] 4.430872e-73  2.995714e-94 2.768103e-78 2.021004e-125          1
## [22,] 4.975157e-74  2.055354e-95 5.906255e-79 3.027152e-127          1
## [23,] 7.197888e-65  8.906846e-83 8.188283e-65 8.490729e-114          1
## [24,] 1.579146e-75  9.079980e-98 7.866279e-83 1.339020e-128          1
## [25,] 2.346181e-59  2.222330e-75 1.964517e-57 4.311888e-105          1
## [26,] 7.007283e-68  6.394975e-87 2.098736e-69 2.993679e-118          1
## [27,] 5.016216e-70  1.589297e-89 8.622799e-71 1.927715e-122          1
## [28,] 4.071574e-75  3.529229e-97 4.477589e-82 4.630024e-128          1
## [29,] 5.544979e-68  3.310006e-84 4.579827e-54 8.737925e-127          1
## [30,] 3.407372e-76  6.495114e-99 4.330963e-85 6.534640e-129          1
## [31,] 5.252360e-70  3.365413e-90 1.434656e-74 2.376928e-120          1
## [32,] 2.730293e-62  1.750111e-78 7.731176e-57 3.580357e-112          1
## [33,] 3.505144e-69  4.965766e-88 1.073624e-67 3.588957e-122          1
## [34,] 4.240654e-65  3.039127e-83 5.547408e-66 1.077283e-113          1
## [35,] 1.023048e-77 1.215345e-100 2.033034e-85 3.060933e-132          1
## [36,] 1.004681e-67  7.667811e-87 6.600501e-70 1.283850e-117          1
## [37,] 2.332985e-65  1.344769e-83 2.315616e-66 4.363691e-114          1
## [38,] 1.356786e-66  5.497329e-85 1.484826e-66 7.917173e-117          1
## [39,] 1.480774e-68  5.708487e-88 4.353126e-71 6.792924e-119          1
## [40,] 5.096809e-72  6.299345e-92 5.567213e-72 2.177719e-126          1
## [41,] 7.542596e-64  2.299498e-81 3.251771e-63 2.593953e-112          1
## [42,] 3.671620e-74  1.096289e-95 1.185507e-79 3.585046e-127          1
## [43,] 5.242915e-65  2.589306e-83 6.585338e-67 5.564340e-113          1
## [44,] 1.086759e-66  2.088215e-85 2.900305e-68 3.994999e-116          1
## [45,] 5.291276e-67  1.974803e-86 5.756253e-72 7.673877e-115          1
## [46,] 1.300602e-75  1.205851e-97 1.163514e-81 1.993000e-129          1
## [47,] 2.128621e-70  1.105707e-90 7.320793e-75 4.271500e-121          1
## [48,] 4.299162e-68  2.334542e-87 1.612324e-70 3.895837e-118          1
## [49,] 6.266306e-60  7.369695e-76 1.258116e-56 7.543072e-107          1
## [50,] 6.630500e-73  4.573771e-94 2.506556e-78 5.396769e-125          1
## [51,] 1.062863e-73  6.211321e-95 2.622894e-78 7.763282e-127          1
```

Question 6

My analysis shows that Kanye West obtains his highest vote share in states that have a high percentage of black people and a high level of employment. As Kanye West is a new candidate, his voters have most

likely switched from a previous party. The candidate that also performs well in states with a high percentage of black people and a high level of employment is Joe Biden, therefore Kanye West's voters have probably switched from voting for Biden to voting for Kanye West.

Kanye West obtained a very small number of votes, totaling 60,000, and the most votes that West obtained in a single state was Tennessee with 10,258 votes. Out of the 12 states that Kanye West ran in, the smallest number of votes to decide a state was Minnesota which Joe Biden won by a margin of 233,394 votes. This highlights the negligible difference that Kanye West had in the election. It is reasonable to assume that if Kanye West did run in all 51 states, he would also have had negligible effects and not impacted on the outcome on any of the states. However, if Kanye West does run in 2024, in all 51 states and with a full political campaign, he could then start to be a factor in deciding state outcomes.

Question 7

My Markov Chain Monte Carlo sampler allows me to select different features from the data and sample from the posterior distribution. The model produces well mixed traceplots, indicating that convergence has occurred. The model also produces reasonable posterior distributions for the parameters. Therefore, I am confident that the majority of the model is working correctly. However, the posterior distributions that I obtain for the parameters change every time that I run the model, this causes the Monte Carlo Estimate of β that I derive from these posteriors to also change every time I run the model. Therefore, any conclusions that I have made are not scientifically valid as they are not reproducible.

In order to calculate the multinomial probabilities for each state, I applied the formula $\eta = X\beta$ to calculate η and then applied the softmax function that links the η matrix to θ. The values of θ that I obtained were either very close to 0 or very close to 1, suggesting that a random voter's vote for a candidate in each state could be predicted with almost certain probability. This is not realistic as there is a high level of uncertainty in voter prediction.

The dataset that I have performed my analysis on gives features of each of the states. It would be useful to have a dataset that specifies features of the candidates in each state. For example, the percentage of Kanye West's voters that were employed in each state or the percentage of Donald Trump's voters that were Hispanic in each state. This would give an insight into the type of voters that are voting for each candidate in each state. Using this data, I could analyse how Kanye West's voters compare in similarity to other candidates to see which candidate's vote share he would be influencing most.