Bayesian Analysis of US Election Data

087074

January 2021

During the Covid-19 outbreak, Joe Biden beat Donald Trump to become the next President of the United States. In this research paper, I will be performing a Bayesian analysis on 2020 US election data to explore how important Covid-19 might have been in influencing state results in the elections.

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

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

I have transformed the raw data into a dataset that is in a more useful format for analysis. The new dataset contains information on a different state in each row. Each row also has a target variable which is the winner of the state i.e. Donald Trump or Joe Biden.

```
WithWinner <-  group_by(USelectionData,State) %>%  # data wrangling
  mutate(Winner=Candidate[which.max(TotalVotes)]) %>% ungroup() %>%
  dplyr::select(State, Candidate, Winner, TotalVotes, Party, Cases, Deaths, TotalPop,
                Men, Women, Hispanic, White, Black, Native, Asian, Pacific, Citizen,
                Poverty, IncomePerCap, Employed, Professional, Service, Office,
                Construction, Production)
# filter the data to only consider Biden vs Trump
USelectionDataBT <- filter(WithWinner, Candidate %in% c("Donald Trump", "Joe Biden"))
BidenTrumpWinner <- group_by(USelectionDataBT, State, Cases, Deaths, TotalPop, Men,
                             Women, Hispanic, White, Black, Native, Asian, Pacific,
                             Citizen, Poverty, IncomePerCap, Employed, Professional,
                             Service, Office, Construction, Production) %>%
  summarise(Winner = Winner[1], .groups = 'drop') %>% ungroup()
```

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.

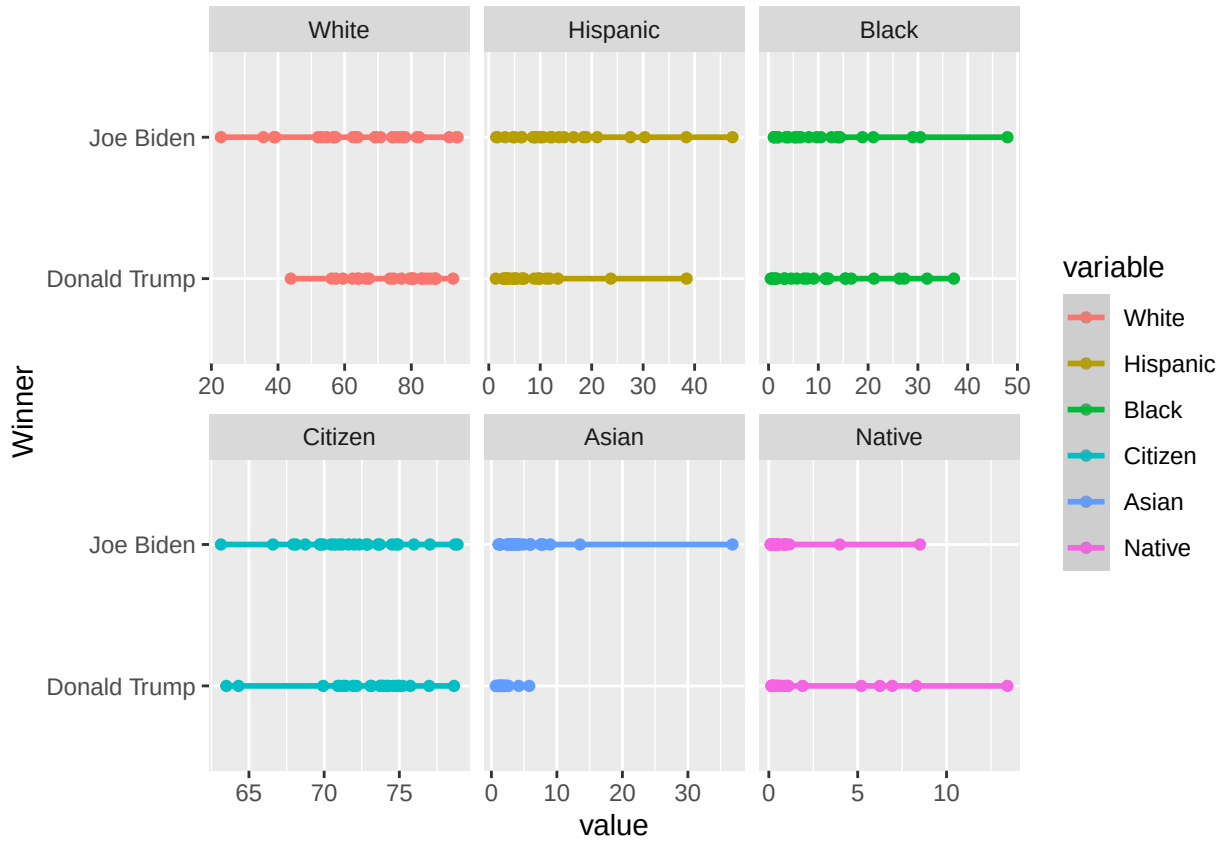
```
BidenTrumpWinner$CasesPerPopulationPercent <- # calculate the percentage of Covid-19
  (BidenTrumpWinner$Cases/BidenTrumpWinner$TotalPop)*100  # cases and deaths
BidenTrumpWinner$DeathsPerPopulationPercent <-             # in each state
  (BidenTrumpWinner$Deaths/BidenTrumpWinner$TotalPop)*100
```

To understand the effect that the variables have on who won each state, I have performed some preliminary data analysis. For this analysis, I have grouped the variables into four categories: those concerning race, income, profession and Covid-19. Through qualitatively analyzing graphs, we can gather an intuition as to which variables will be good indicators as to who won each state.

From this graph, we can see that states with a higher percentage of white people tend to be won by Donald Trump and that states with high levels of Hispanic and Asian people tend to be won by Joe Biden.

```
# select the variables relating to race
DataVisualisaionRace <- BidenTrumpWinner %>% select(Winner, White, Hispanic, Black,
                                                    Citizen, Asian, Native)
VisualisationRace <- melt(DataVisualisaionRace, id.vars="Winner")
ggplot(VisualisationRace) +    # plot a point graph of the data
  geom_point(aes(x=value, y=Winner, colour=variable)) +
  geom_smooth(aes(x=value, y=Winner, colour=variable), method=lm) +
  facet_wrap(~variable, scales="free_x")
```

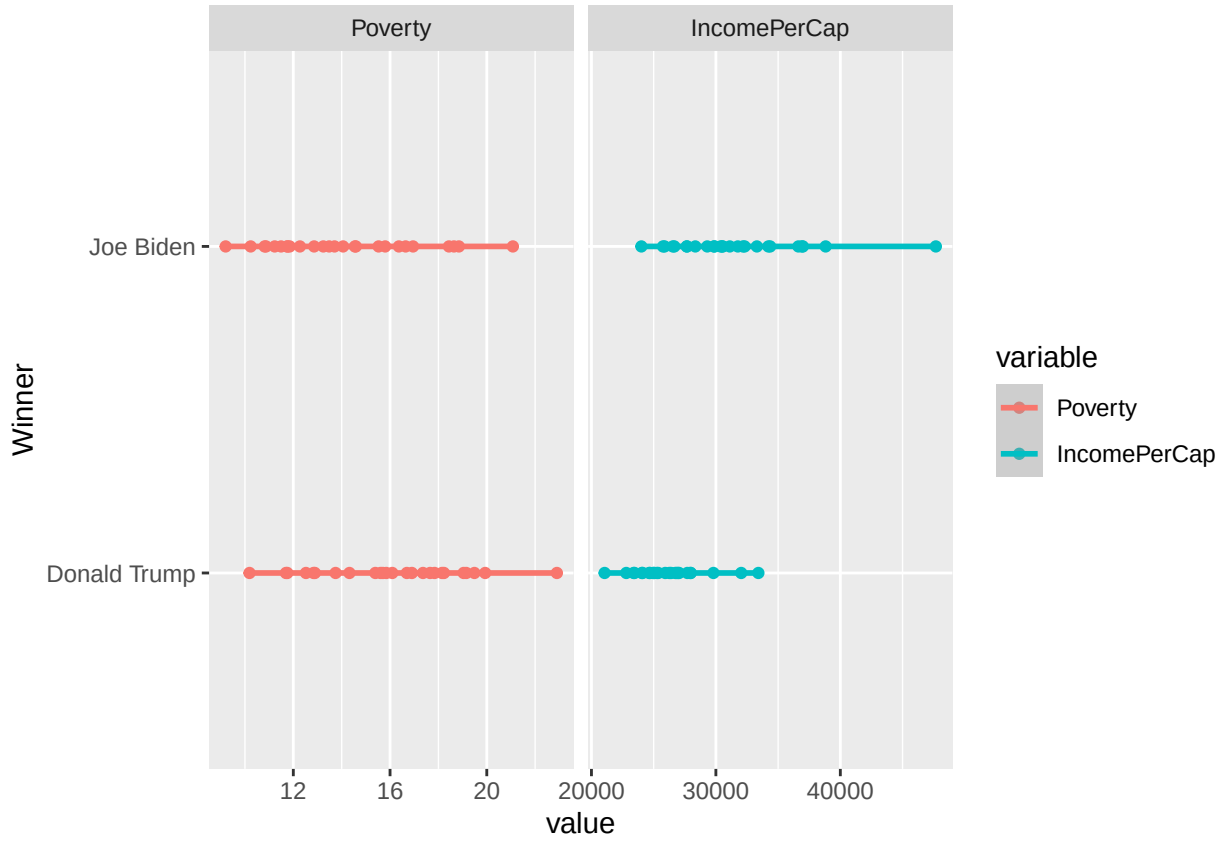
```
## `geom_smooth()` using formula 'y ~ x'
```



This graph strongly suggests that states with higher income per capita levels tend to be won by Joe Biden.

```
# select the variables relating to income
DataVisualisaionIncome <- BidenTrumpWinner %>% select(Winner, Poverty, IncomePerCap)
VisualisationIncome <- melt(DataVisualisaionIncome, id.vars="Winner")
ggplot(VisualisationIncome) +  # plot a point graph of the data
  geom_point(aes(x=value, y=Winner, colour=variable)) +
  geom_smooth(aes(x=value, y=Winner, colour=variable), method=lm) +
  facet_wrap(~variable, scales="free_x")
```

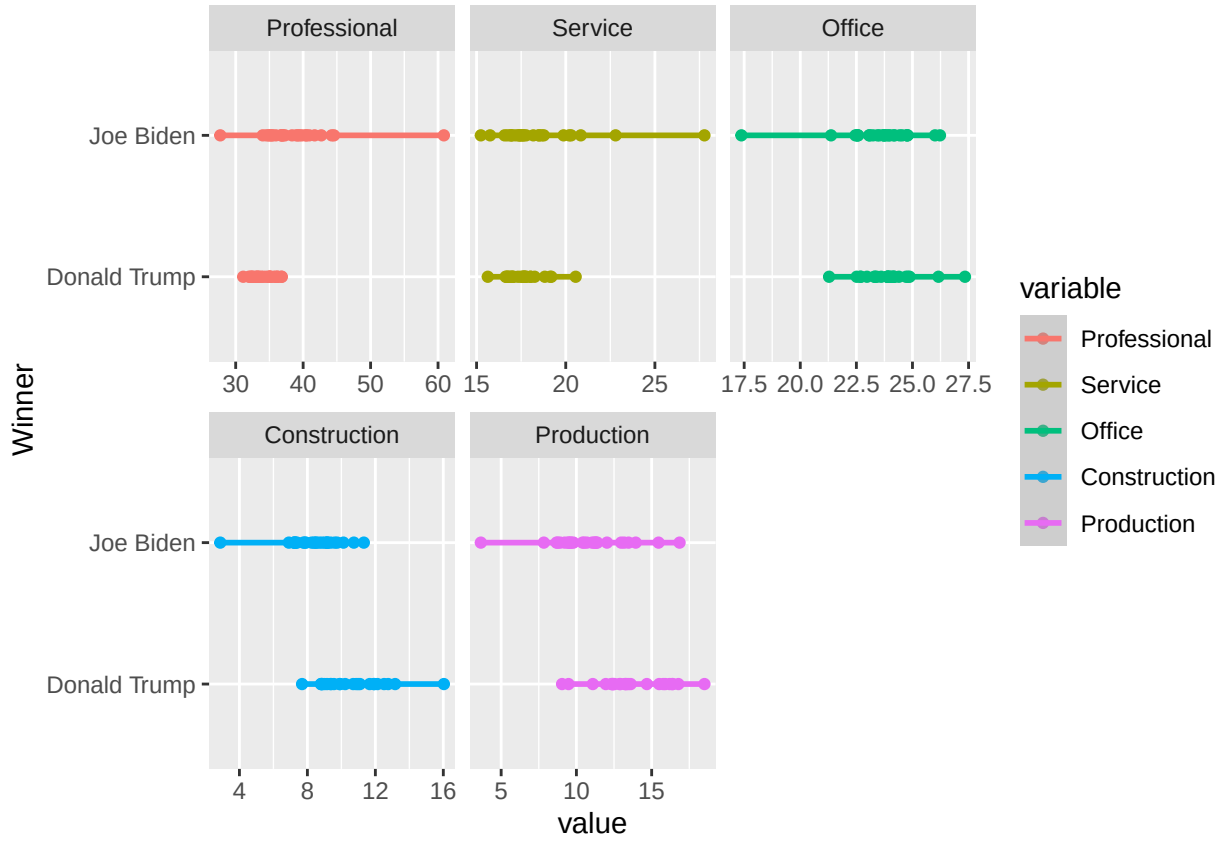
```
## `geom_smooth()` using formula 'y ~ x'
```



This graph shows that states with a higher percentage of people employed in professional jobs tend to be won by Joe Biden. States with high levels of construction and production jobs are likely to be won by Donald Trump.

```
# select the variables relating to occupation
DataVisualisaionJobs <- BidenTrumpWinner %>% select(Winner, Professional, Service,
                                                    Office, Construction, Production)
VisualisationJobs <- melt(DataVisualisaionJobs, id.vars="Winner")
ggplot(VisualisationJobs) +   # plot a point graph of the data
  geom_point(aes(x=value, y=Winner, colour=variable)) +
  geom_smooth(aes(x=value, y=Winner, colour=variable), method=lm) +
  facet_wrap(~variable, scales="free_x")
```

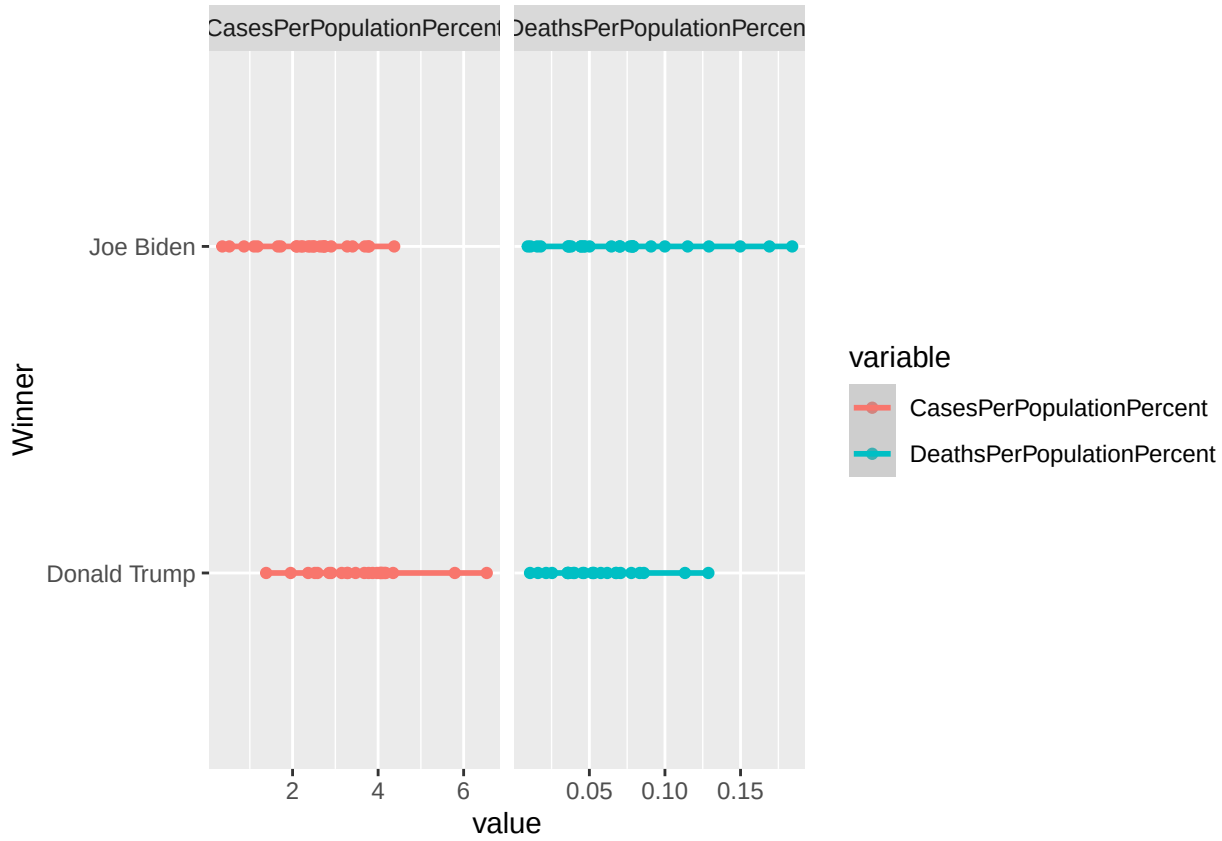
```
## `geom_smooth()` using formula 'y ~ x'
```



This graph suggests that states with higher levels of Covid-19 cases tend to be won by Donald Trump whereas states with higher levels of deaths from Covid-19 tend to be won by Joe Biden.

```
# select the variables relating to Covid-19, these are the variable I created
DataVisualisaionCovid <- BidenTrumpWinner %>% select(Winner, CasesPerPopulationPercent, DeathsPerPopula
VisualisationCovid <- melt(DataVisualisaionCovid, id.vars="Winner")
ggplot(VisualisationCovid) +
  geom_point(aes(x=value, y=Winner, colour=variable)) +
  geom_smooth(aes(x=value, y=Winner, colour=variable), method=lm) +
  facet_wrap(~variable, scales="free_x")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



I will perform a preliminary logistic regression to quantitatively understand which variables have the most effect on state outcome. For this logistic regression model, I will only be considering variables that appeared to influence the state outcome during data visualization.

First the winner column is turned into a Binary classifier.

```
BidenTrumpWinner$Winner[BidenTrumpWinner$Winner == "Donald Trump"] <- "0"
BidenTrumpWinner$Winner[BidenTrumpWinner$Winner == "Joe Biden"] <- "1"
```

A new dataset is created that only contains the target variable and the variables that will be used in the logistic regression model.

```
LogisticData1 <- BidenTrumpWinner %>% select(Winner, White, Asian, Hispanic,
                                             IncomePerCap, Professional, Construction,
                                             Production, CasesPerPopulationPercent,
                                             DeathsPerPopulationPercent)
```

The new dataset is split into train and test sets at an 80:20 ratio and the parameters of the logistic regression model are set.

```
train_df <- head(LogisticData1, n=0.8*(nrow(LogisticData1)))  # training dataset
valid_df <- tail(LogisticData1, n=0.2*(nrow(LogisticData1)))  # testing dataset

p <- 9   # number of model features
options(warn=-1)
train_X <- train_df[,c(2:10)]    # split the train and test sets into descriptive
```

```
train_y <- as.numeric(train_df$Winner)    # and target variables
valid_X <- valid_df[,c(2:10)]
valid_y <- as.numeric(valid_df$Winner)
tN <- length(train_y)    # obtain parameters for the logistic regression
tN_new <- length(valid_y)
tX_new <- valid_X
```

I have given each parameter a prior distribution ~ N(0,5). The distributions have a mean of 0 to ensure that they are unbiased. A variance of 5 is sufficiently large on the log scale to allow most types of effects. The stan linear regression model is then run using 2 chains.

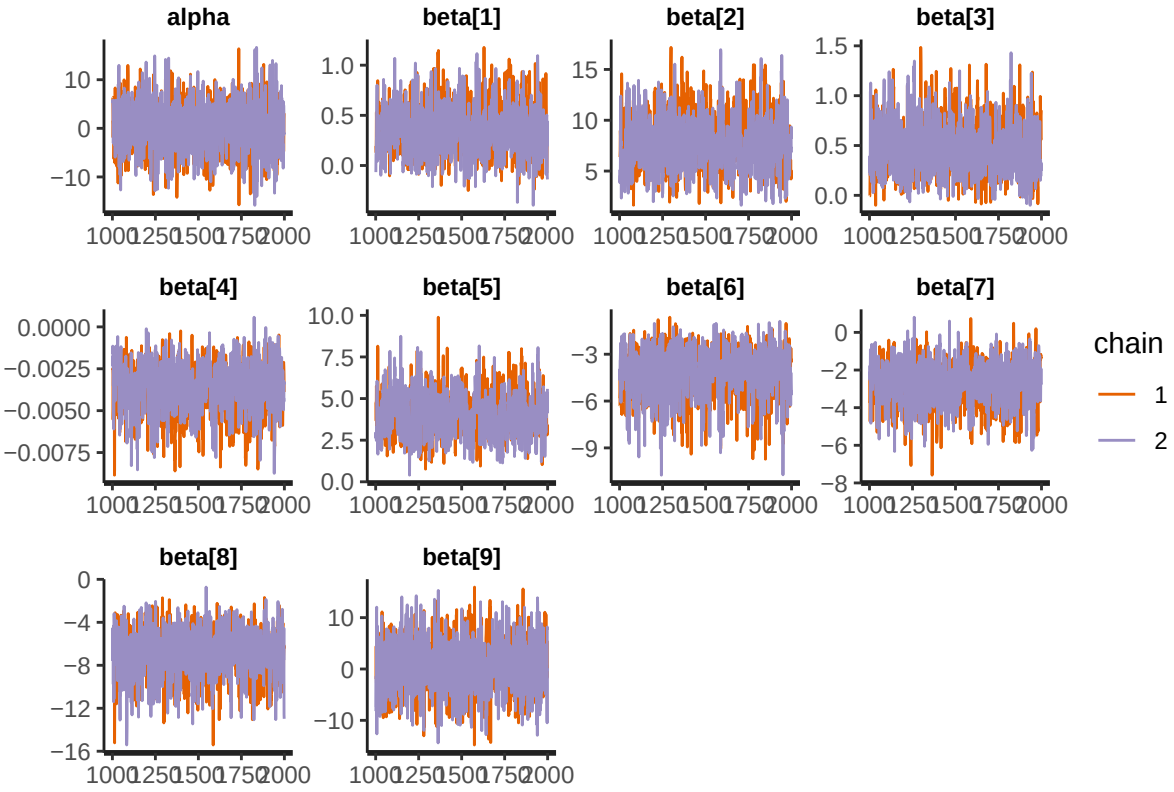
The Rhat values are all very close to 1 and, based on 2000 samples, the effective sample sizes vary between 400 and 1500 indicating that the model has converged.

```
summary(LogReg1)$summary[1:10,]  # summary statistics of the model
```

```
##                mean      se_mean          sd          2.5%          25%
## alpha     0.157658184 1.251394e-01 4.902447917  -9.267603450 -3.050194732
## beta[1]   0.365806170 9.238939e-03 0.240617867  -0.068626845  0.194185428
## beta[2]   7.781506339 1.146472e-01 2.616396100   3.165491756  5.929917284
## beta[3]   0.461035033 9.161358e-03 0.257866842   0.040221697  0.269331275
## beta[4]  -0.003759999 5.858086e-05 0.001435201  -0.006860304 -0.004638536
## beta[5]   4.047218743 5.806527e-02 1.295231567   1.756215686  3.112312787
## beta[6]  -4.214026901 6.520940e-02 1.551823365  -7.858518868 -5.136578475
## beta[7]  -2.729729806 4.971578e-02 1.159661212  -5.177142126 -3.477217900
## beta[8]  -6.980887139 8.680906e-02 2.205424193 -11.608123857 -8.417593797
## beta[9]   0.467852090 1.344380e-01 4.853953544  -9.240590209 -2.667336867
##                 50%          75%        97.5%     n_eff      Rhat
## alpha     0.064756411  3.544253159  9.388717908 1534.7513 1.0006775
## beta[1]   0.354173795  0.514804394  0.868784919  678.2839 1.0035486
## beta[2]   7.555816460  9.492276898 13.430878950  520.8110 1.0082156
## beta[3]   0.431081868  0.622260494  1.032828193  792.2664 1.0027766
## beta[4]  -0.003655372 -0.002754677 -0.001189855  600.2247 1.0087896
## beta[5]   3.966283875  4.907381011  6.785766066  497.5788 1.0085894
## beta[6]  -4.036820791 -3.098318501 -1.749735930  566.3229 1.0019323
## beta[7]  -2.625304553 -1.920216909 -0.709830652  544.0937 1.0051692
## beta[8]  -6.816642650 -5.372060122 -3.191126874  645.4374 1.0081842
## beta[9]   0.467868410  3.676948395 10.204869429 1303.6082 0.9991561
```

The traceplots shown here are well mixed, also indicating that the model has converged.

```
rstan::traceplot(LogReg1, pars=c("alpha","beta"))
```



Due to the restricted sample size of the dataset, the parameters that are kept in the model must be chosen carefully. The model must have enough degrees of freedom so that the data suitably swamps the prior distributions. Therefore I have chosen to follow the '10% rule of thumb' and only include the 6 most influential parameters from the previous regression model. I have formed a new dataset using these parameters. This dataset will form the basis of the final linear regression model.

```
LogisticData2 <- BidenTrumpWinner %>% select(Winner, Asian, Poverty, Professional,
                                             Production, Construction, CasesPerPopulationPercent)
```

It is useful to create a confusion matrix in order to visualize the performance of a classification model.

```
ConfusionMatrix <- function(Classifier, Truth){
  if(!(length(Classifier)==length(Truth))) # vectors must be the same size
    stop("Fix vector length")
  if(is.logical(Classifier))
    Classifier <- as.integer(Classifier)
  WhichClass0s <- which(Classifier < 1)
  ZeroCompare <- Truth[WhichClass0s]
  Predicted0 <- c(length(ZeroCompare)-sum(ZeroCompare), sum(ZeroCompare))
  WhichClass1s <- which(Classifier > 0)
  OnesCompare <- Truth[WhichClass1s]
  Predicted1 <- c(length(OnesCompare)-sum(OnesCompare), sum(OnesCompare))
  ConMatrix <- cbind(Predicted0, Predicted1)
  row.names(ConMatrix) <- c("Actual 0", "Actual 1")
  colnames(ConMatrix) <- c("Pred 0", "Pred 1")
  ConMatrix # return the confusion matrix of a binary classifier
}
```

I have defined an empty confusion matrix, the results of the linear regression model will be added to this. The 'folds' variable contains a 10-fold partition of the states. I have also stated a non-informative, N(0,5), prior distribution for all of the parameters.

```
FullConMatrix2 <- 0 # initialise an empty confusion matrix
# create the parameters of the logistic model
p <- 6
ta <- 0
Sigma <- 5
tSigma_a <- Sigma
beta0 <- rep(0,p)
tSigma_b <- rep(Sigma,p)
```

```
# create the indexes that will be used in the k-fold  cross validation
folds = list(c(1,2,3,4,5), c(6,7,8,9,10), c(11,12,13,14,15), c(16,17,18,19,20),
             c(21,22,23,24,25), c(26,27,28,29,30), c(31,32,33,34,35), c(36,37,38,39,40),
             c(41,42,43,44,45),c(46,47,48,49,50,51))
```

I now run the logistic regression model on the dataset using a 10-fold cross validation.

```
for (i in folds){  # iterate over each fold
  train_df <- LogisticData2[-i,]  # use each fold to create test and train sets
  valid_df <- LogisticData2[i,]
  train_X <- train_df[,c(2:7)]
  train_y <- as.numeric(train_df$Winner)
  valid_X <- valid_df[,c(2:7)]
  valid_y <- as.numeric(valid_df$Winner)

  tN <- length(train_y)   # create parameters for the logistic regression model
  tN_new <- length(valid_y)
  tX_new <- valid_X
  # create and run the model
  LogisticData <- list(N=tN, p=p, X=as.matrix(train_X), y=train_y, N_new=tN_new,
                       X_new=as.matrix(tX_new), a=ta, Sigma_a=tSigma_a, beta0=beta0,
                       Sigma_b=tSigma_b)
  LogReg2 <- stan("logisticRegression.stan", data=LogisticData, chains=2)
  # obtain the predictions
  Predictions <- rstan::extract(LogReg2, pars=c("alpha", "beta", "eta"), include=FALSE)

  PostPredProbsJoint <- 1/(1+exp(-Predictions$eta_new))   # apply logit function
  postPredProbs <- apply(PostPredProbsJoint,2,mean)
  Classifier <- postPredProbs > 0.5  # classify each prediction
  # add the confusion matrix for each iteration to the total confusion matrix
  FullConMatrix2 <- FullConMatrix2 + ConfusionMatrix(Classifier=Classifier,
                                                     Truth=valid_y)
}
```

The Rhat values are all very close to 1. Based on 2000 samples, the effective sample sizes are all around 400, suggesting that the model has converged.

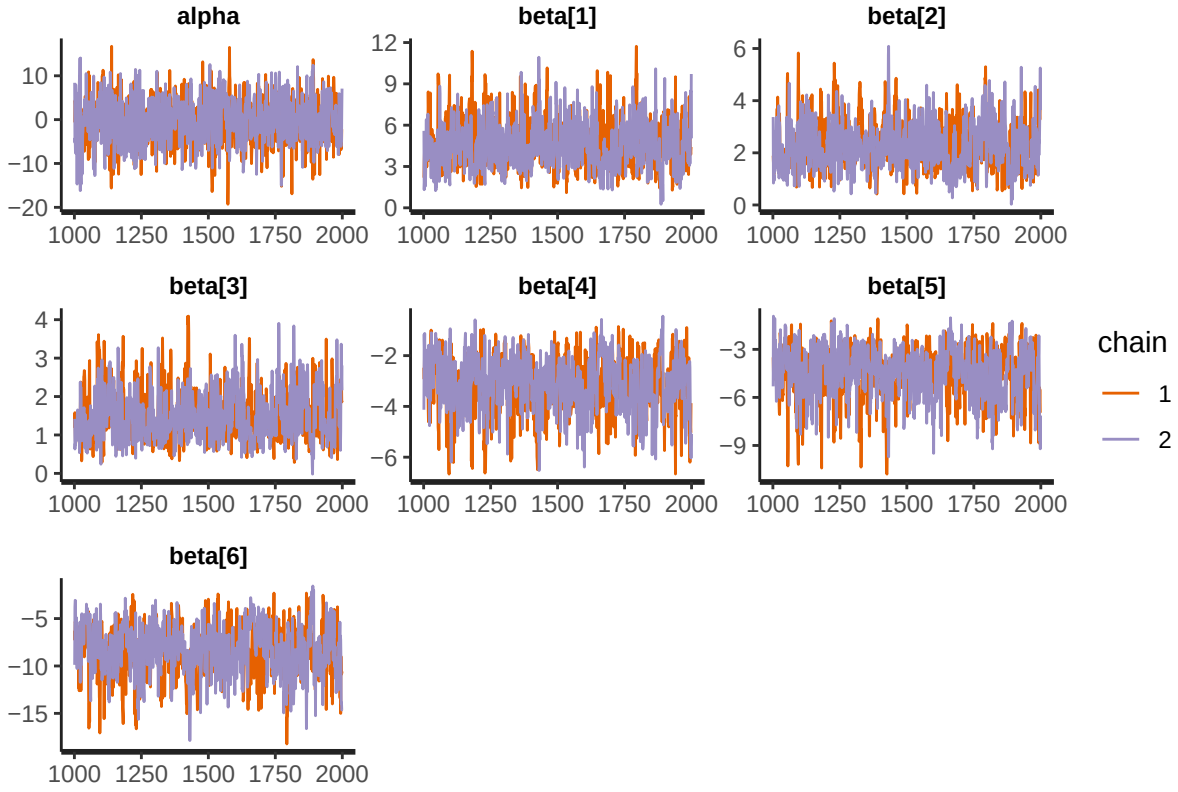
```
summary(LogReg2)$summary[1:7,]
```

```
##               mean     se_mean         sd        2.5%         25%         50%
```

```
## alpha   -0.3185396 0.15968444 5.0617681 -10.6281530 -3.564632 -0.3506635
## beta[1]  4.6819617 0.07789961 1.7105669   1.8585127  3.424766  4.5294423
## beta[2]  2.3219800 0.04365389 0.8803959   0.7974286  1.689155  2.2476423
## beta[3]  1.5132935 0.03121941 0.6285617   0.5548725  1.074973  1.3858217
## beta[4] -3.1354985 0.05833596 1.0655407  -5.4523868 -3.801035 -3.0783697
## beta[5] -4.5090764 0.08364111 1.5640281  -8.0201179 -5.414303 -4.3133484
## beta[6] -8.3411667 0.13710132 2.5679500 -13.9977353 -9.961861 -8.1895199
##               75%     97.5%     n_eff     Rhat
## alpha    3.069843  9.520947 1004.7992 1.001733
## beta[1]  5.749550  8.378278  482.1803 1.008107
## beta[2]  2.879330  4.222633  406.7337 1.005000
## beta[3]  1.904510  2.915087  405.3652 1.009585
## beta[4] -2.358841 -1.285798  333.6317 1.011164
## beta[5] -3.417320 -2.006462  349.6629 1.012802
## beta[6] -6.458189 -3.907841  350.8245 1.004330
```

The traceplots are well mixed indicating that the model has converged.

```
rstan::traceplot(LogReg2, pars=c("alpha","beta"))
```



The linear regression model is 86% accurate at predicting who won the state. We can therefore conclude that the variables that have been used in the model can be used as good predictors for state outcomes.

```
FullConMatrix2
```

```
##          Pred 0 Pred 1
```

```
## Actual 0     21      4
## Actual 1      3     23
```

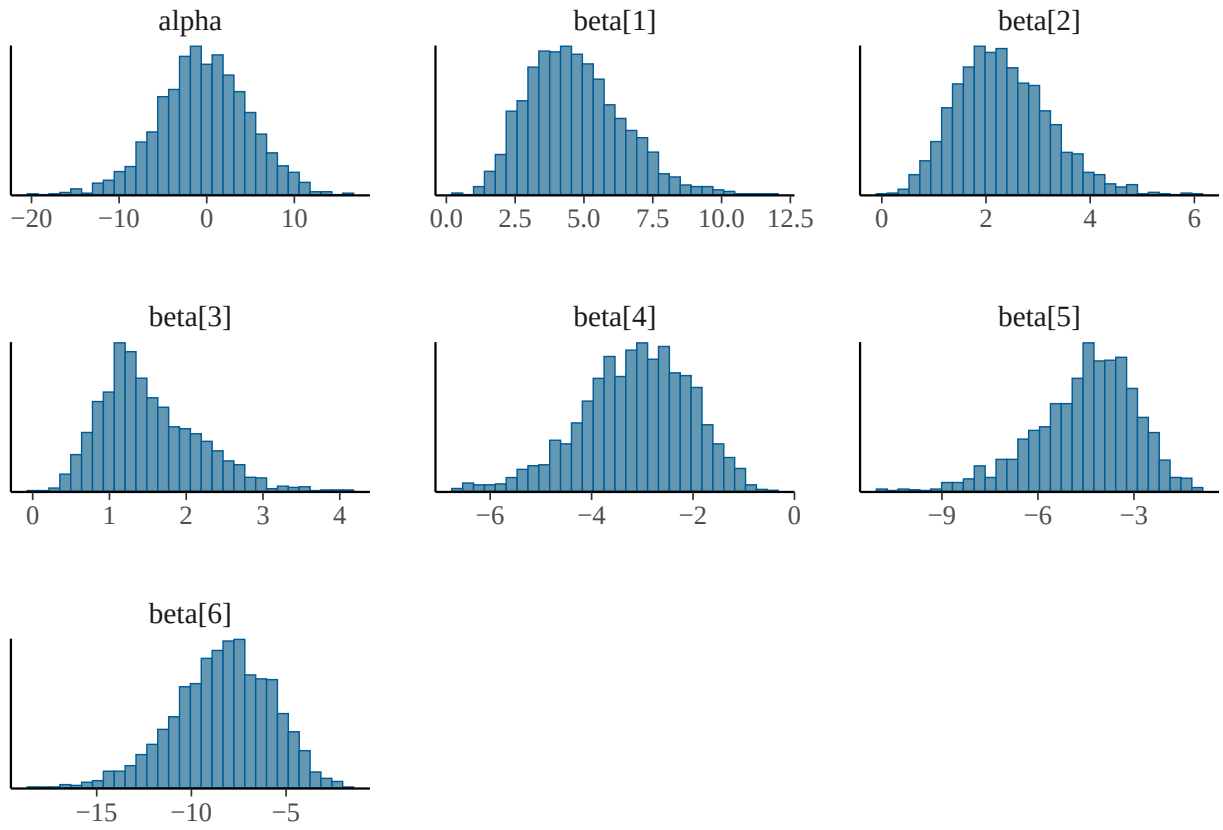
```
accuracy <- sum(diag(FullConMatrix2)/sum(FullConMatrix2))  # accuracy equation
cat('The accuracy of the model is:', accuracy)
```

```
## The accuracy of the model is: 0.8627451
```

It is useful to analyse the posterior distributions of the parameters to assess what affect the variables had on the model.

```
ParameterData2 <- as.matrix(LogReg2)[,1:7]  # extract the parameter information from model
mcmc_hist(ParameterData2, pars=c("alpha", "beta[1]", "beta[2]", "beta[3]", "beta[4]",
                                 "beta[5]", "beta[6]"))
```

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



The prior for each of the parameters of the model is a normal distribution that is centered on 0 with variance 5. This is to ensure that the priors are unbiased and can encompass all feasible possibilities for the posterior distributions. I will perform sensitivity analysis on the priors in order to analyse the robustness of the model. To do so, I will change the prior distributions of the model parameters to have a variance of 2.5 and 7.5 and analyse the effect that this has on the posterior distributions.

Here I set the variance to 2.5.

```
## variance = 2.5
FullConMatrix3 <- 0
p <- 6
ta <- 0
Sigma <- 2.5   # the only thing that has changed in the model is the variance
tSigma_a <- Sigma
beta0 <- rep(0,p)
tSigma_b <- rep(Sigma,p)
```

```
for (i in folds){   # repeat as done previously
  train_df <- LogisticData2[-i,]
  valid_df <- LogisticData2[i,]
  train_X <- train_df[,c(2:7)]
  train_y <- as.numeric(train_df$Winner)
  valid_X <- valid_df[,c(2:7)]
  valid_y <- as.numeric(valid_df$Winner)

  tN <- length(train_y)
  tN_new <- length(valid_y)
  tX_new <- valid_X

  LogisticData <- list(N=tN, p=p, X=as.matrix(train_X), y=train_y, N_new=tN_new,
                       X_new=as.matrix(tX_new), a=ta, Sigma_a=tSigma_a, beta0=beta0,
                       Sigma_b=tSigma_b)
  LogReg3 <- stan("logisticRegression.stan", data=LogisticData, chains=2)
  Predictions <- rstan::extract(LogReg3, pars=c("alpha", "beta", "eta"), include=FALSE)

  PostPredProbsJoint <- 1/(1+exp(-Predictions$eta_new))   # Logit function ?
  postPredProbs <- apply(PostPredProbsJoint,2,mean)
  Classifier <- postPredProbs > 0.5
  FullConMatrix3 <- FullConMatrix3 + ConfusionMatrix(Classifier=Classifier,
                                                     Truth=valid_y)
}
```

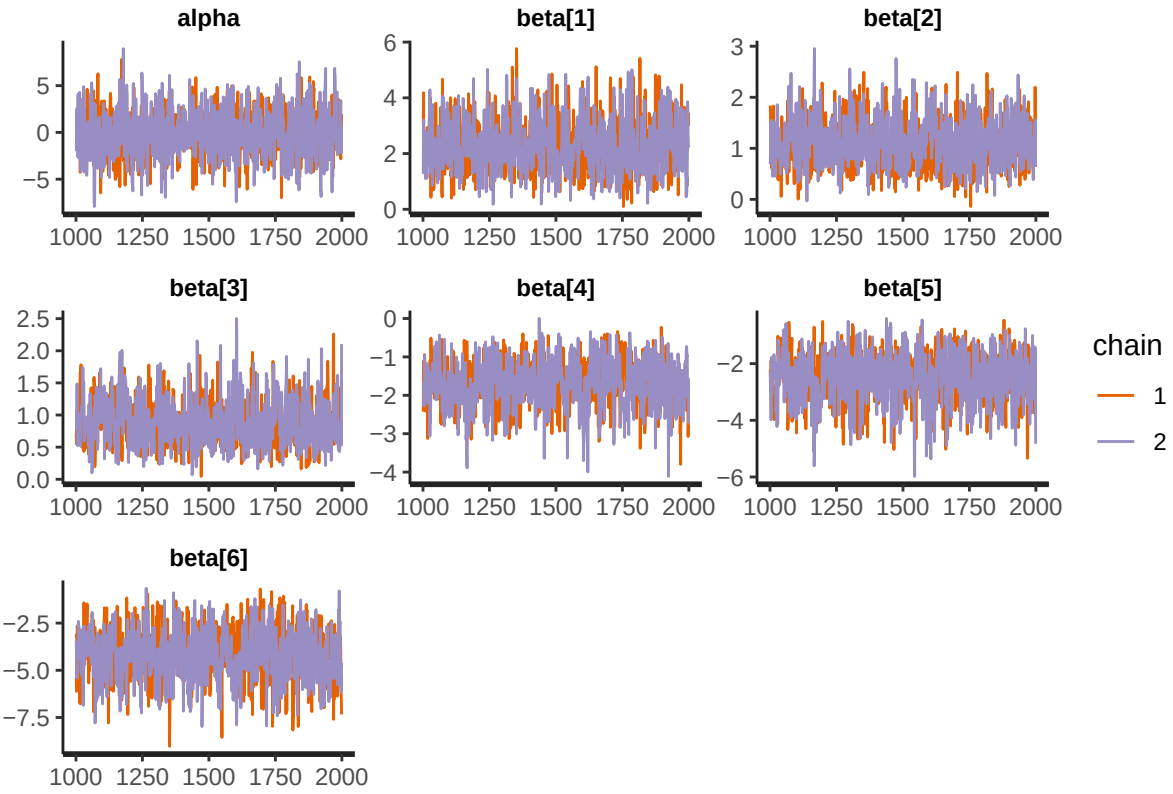
It is evident from the summary statistics and traceplots that the parameters of the model have converged.

```
summary(LogReg3)$summary[1:7,]
```

```
##               mean    se_mean        sd       2.5%        25%        50%
## alpha   -0.1273973 0.07680394 2.4198863 -4.7327625 -1.7421475 -0.1559716
## beta[1]  2.2643244 0.03341090 0.8909124  0.7166533  1.6202906  2.1731823
## beta[2]  1.1254979 0.01920099 0.4537301  0.3206565  0.7964974  1.1146999
## beta[3]  0.8894174 0.01475175 0.3459221  0.3028564  0.6400764  0.8618853
## beta[4] -1.6885017 0.02640882 0.5847547 -2.8933675 -2.0695613 -1.6628315
## beta[5] -2.4992010 0.03642893 0.8418038 -4.3178427 -3.0146204 -2.4274324
## beta[6] -4.1800466 0.05702067 1.3189337 -6.8709998 -5.0651821 -4.1580215
##               75%      97.5%    n_eff      Rhat
## alpha    1.489171  4.6413178 992.7112 1.0008740
## beta[1]  2.808751  4.2743536 711.0395 1.0002797
## beta[2]  1.430294  2.0554461 558.4030 0.9995853
## beta[3]  1.096635  1.6411346 549.8820 0.9991245
## beta[4] -1.280853 -0.6224515 490.2860 0.9991667
## beta[5] -1.901343 -1.0303435 533.9849 0.9993552
```

```
## beta[6] -3.213522 -1.6965168 535.0339 1.0002799
```

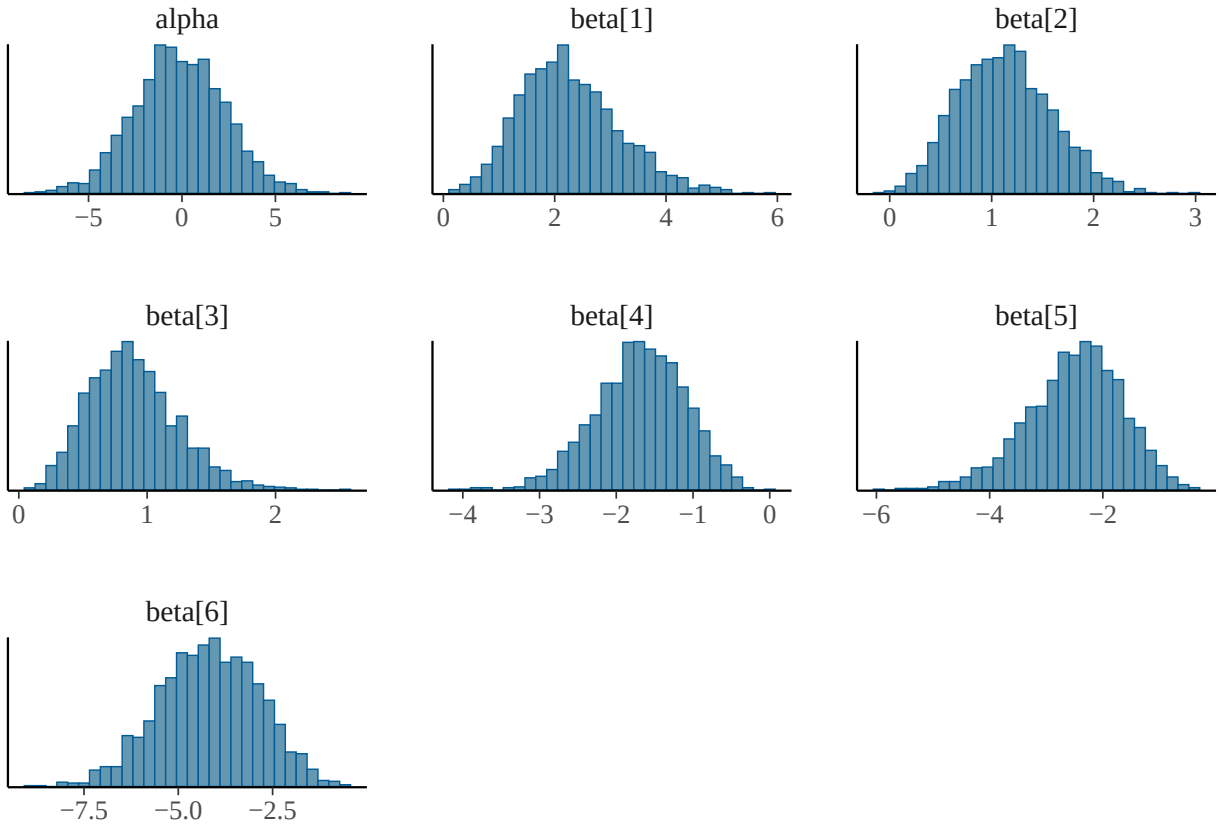
```
rstan::traceplot(LogReg3, pars=c("alpha","beta"))
```



The parameters' posterior distributions when the variance of the prior is 2.5 are very similar to the parameters' posterior distributions when the variance is 5.

```
ParameterData3 <- as.matrix(LogReg3)[,1:7]  # extract the posterior distributions
mcmc_hist(ParameterData3, pars=c("alpha", "beta[1]", "beta[2]", "beta[3]", "beta[4]",
                                 "beta[5]", "beta[6]"))
```

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



The accuracy of the model is unchanged.

```
FullConMatrix3
```

```
##          Pred 0 Pred 1
## Actual 0     21      4
## Actual 1      3     23
```

```
accuracy <- sum(diag(FullConMatrix3)/sum(FullConMatrix3))
cat('The accuracy of the model is:', accuracy)
```

```
## The accuracy of the model is: 0.8627451
```

In order to perform a symmetric sensitivity analysis, I change the variance of the prior distributions to 7.5.

```
FullConMatrix4 <- 0
p <- 6
ta <- 0
Sigma <- 7.5 # the model is the sam but variance now equals 7.5
tSigma_a <- Sigma
beta0 <- rep(0,p)
tSigma_b <- rep(Sigma,p)
```

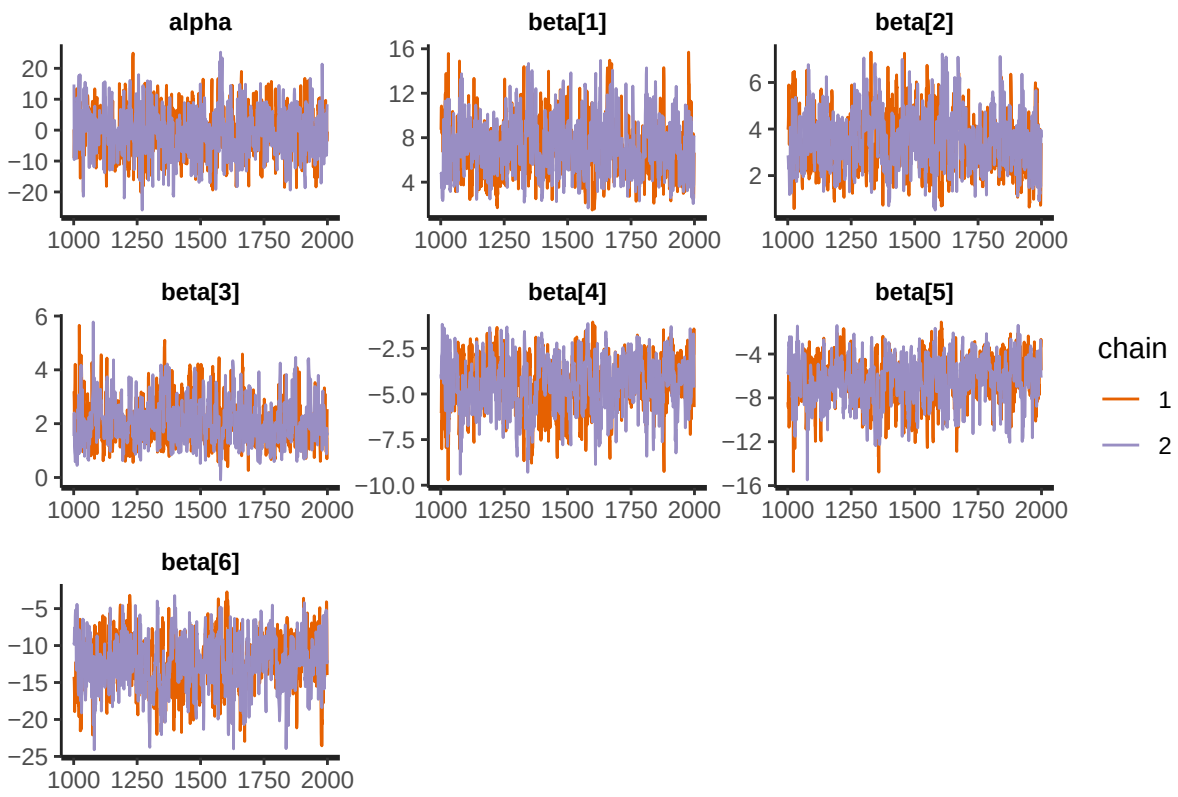
```
for (i in folds){
  train_df <- LogisticData2[-i,]
  valid_df <- LogisticData2[i,]
  train_X <- train_df[,c(2:7)]
  train_y <- as.numeric(train_df$Winner)
  valid_X <- valid_df[,c(2:7)]
  valid_y <- as.numeric(valid_df$Winner)

  tN <- length(train_y)
  tN_new <- length(valid_y)
  tX_new <- valid_X

  LogisticData <- list(N=tN, p=p, X=as.matrix(train_X), y=train_y, N_new=tN_new,
                       X_new=as.matrix(tX_new), a=ta, Sigma_a=tSigma_a, beta0=beta0,
                       Sigma_b=tSigma_b)
  LogReg4 <- stan("logisticRegression.stan", data=LogisticData, chains=2)
  Predictions <- rstan::extract(LogReg4, pars=c("alpha", "beta", "eta"), include=FALSE)

  PostPredProbsJoint <- 1/(1+exp(-Predictions$eta_new))   # Logit function ?
  postPredProbs <- apply(PostPredProbsJoint,2,mean)
  Classifier <- postPredProbs > 0.5
  FullConMatrix4 <- FullConMatrix4 + ConfusionMatrix(Classifier=Classifier,
                                                     Truth=valid_y)
}
```

The model has converged, this is evident from the summary statistics and the well mixed traceplots.

```
summary(LogReg4)$summary[1:7,]
```

```
##                mean    se_mean        sd        2.5%        25%         50%
## alpha    -0.7338171 0.30445172 7.5744468 -15.3708283  -5.961392  -0.5569086
## beta[1]   7.1524762 0.13410387 2.5109547   2.8576859   5.356470   6.9434507
## beta[2]   3.4784757 0.06771721 1.2385524   1.3123729   2.562043   3.3944265
## beta[3]   2.0609946 0.04036775 0.8292274   0.7381389   1.474047   1.9373925
## beta[4]  -4.4707774 0.07921775 1.4651478  -7.5551925  -5.411402  -4.3608755
## beta[5]  -6.3671458 0.11555845 2.1702720 -11.1421077  -7.651211  -6.1392868
## beta[6] -12.2177956 0.21639688 3.6936519 -20.0154823 -14.696909 -11.9995059
##                75%     97.5%    n_eff      Rhat
## alpha    4.116409 14.389321 618.9634 0.9996586
## beta[1]  8.763827 12.626516 350.5865 1.0053671
## beta[2]  4.318178  5.990427 334.5266 1.0022546
## beta[3]  2.528946  3.986720 421.9667 1.0060588
## beta[4] -3.395959 -1.970842 342.0723 1.0038806
## beta[5] -4.845511 -2.730068 352.7158 1.0083484
## beta[6] -9.485177 -5.521160 291.3466 1.0054725
```

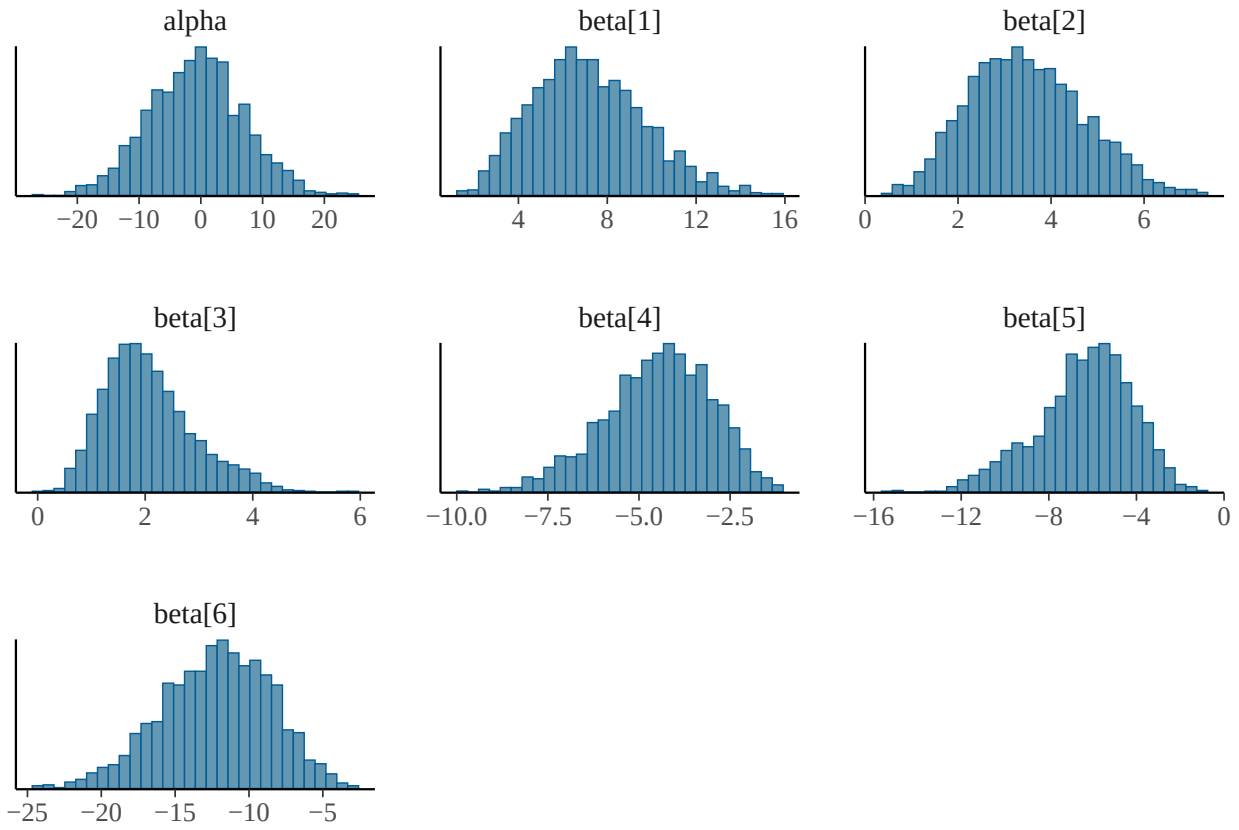
```
rstan::traceplot(LogReg4, pars=c("alpha","beta"))
```



The parameters' posterior distributions are similar to those obtained with different variances.

```
ParameterData4 <- as.matrix(LogReg4)[,1:7] # extract and plot the posterior distributions
mcmc_hist(ParameterData4, pars=c("alpha", "beta[1]", "beta[2]", "beta[3]", "beta[4]",
                                 "beta[5]", "beta[6]"))
```

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



The same accuracy score is achieved.

```
FullConMatrix4
```

```
##          Pred 0 Pred 1
## Actual 0     21      4
## Actual 1      3     23
```

```
accuracy <- sum(diag(FullConMatrix4)/sum(FullConMatrix4))
cat('The accuracy of the model is:', accuracy)
```

```
## The accuracy of the model is: 0.8627451
```

The model converges for a prior variance of 2.5, 5 and 7.5, showing that the model is robust enough to cope with a range of different prior distributions. Thus, it can be concluded that the model is insensitive to the prior distribution.

In order to calculate the Monte Carlo estimate of the parameters, the sum of the parameters must be divided by the sample size (2000).

```
Param_n <- c(2000, 2000, 2000, 2000, 2000, 2000, 2000)  # a list of sample sizes
Param_sum <- colSums(ParameterData2)
MC_estimate <- Param_sum/Param_n # divided the sum of each parameter samples by 2000
MC_estimate
```

```
##      alpha    beta[1]    beta[2]    beta[3]    beta[4]    beta[5]    beta[6]
## -0.3185396  4.6819617  2.3219800  1.5132935 -3.1354985 -4.5090764 -8.3411667
```

To calculate the Monte Carlo errors of the parameters, the standard deviations of the parameters are divided by the square root of their effective sample sizes. The Monte Carlo errors are small for all of the parameters. Therefore, we can be confident that the Monte Carlo estimates are accurate representations of the parameter values.

```
Param_n_eff <- summary(LogReg2)$summary[1:7,][,"n_eff"] # obtain the effective sample
Param_sd <- apply(ParameterData2, 2, sd)                  # size
MC_error <- Param_sd/sqrt(Param_n_eff)   # apply MC error formula
MC_error
```

```
##      alpha    beta[1]    beta[2]    beta[3]    beta[4]    beta[5]    beta[6]
## 0.15968444 0.07789961 0.04365389 0.03121941 0.05833596 0.08364111 0.13710132
```

The winner of the state is a binary classifier, with Joe Biden winning being represented by 1 and Donald Trump winning being represented by 0. Therefore, negative Monte Carlo expectations of parameter values mean that an increase in that feature increases the likelihood that a state will be won by Donald Trump and vice versa. The beta[6] variable is the most influential model feature with a MC estimate of almost double any other parameter. The variables all have the same order of magnitude, therefore we can conclude that the beta[6] variable is almost twice as important in deciding state outcome as any other variable.

The beta[6] variable is the percentage of people with Covid-19 in each state. Therefore, rates of Covid-19 did have a significant impact on state outcome in the election. The MC estimate of the beta[6] parameter is negative, therefore high rates of Covid-19 were a contributing factor in helping Donald Trump win states. Logistic regression is a non-linear model, therefore it is difficult to quantify the direct effects that Covid-19 had on state outcome.

Joe Biden won despite the detrimental effects that Covid-19 rates had on his performance in certain states. Therefore, although rates of Covid-19 effected individual state outcomes, it did not effect who won the election.