Election Prediction

Michael Cambaliza

12/16/2020

```r
dim(election.raw)# dimension of election.raw
```

```
## [1] 31167     5
```

```r
which(is.na(election.raw)) # No missing values in election.raw
```

```
## integer(0)
```

```r
length(unique(election.raw$state)) # data does consist of all states and D.C.
```

```
## [1] 51
```

Data does consist of all states and D.C.

```
## [1] 3220   37
```

```
## [1] 58509
```

```
## [1] 1955
```

```
## [1] 2825
```

Our dimensions of census are 3220 observations by 37 variable (3220x37) and it shows us that 58509 missing values in census, with 1955 distinct county in census, and 2825 distinct county in election.raw.

```r
election.state <- election.raw %>%
  group_by(state, candidate) %>%
  summarise(votes = sum(votes))
```

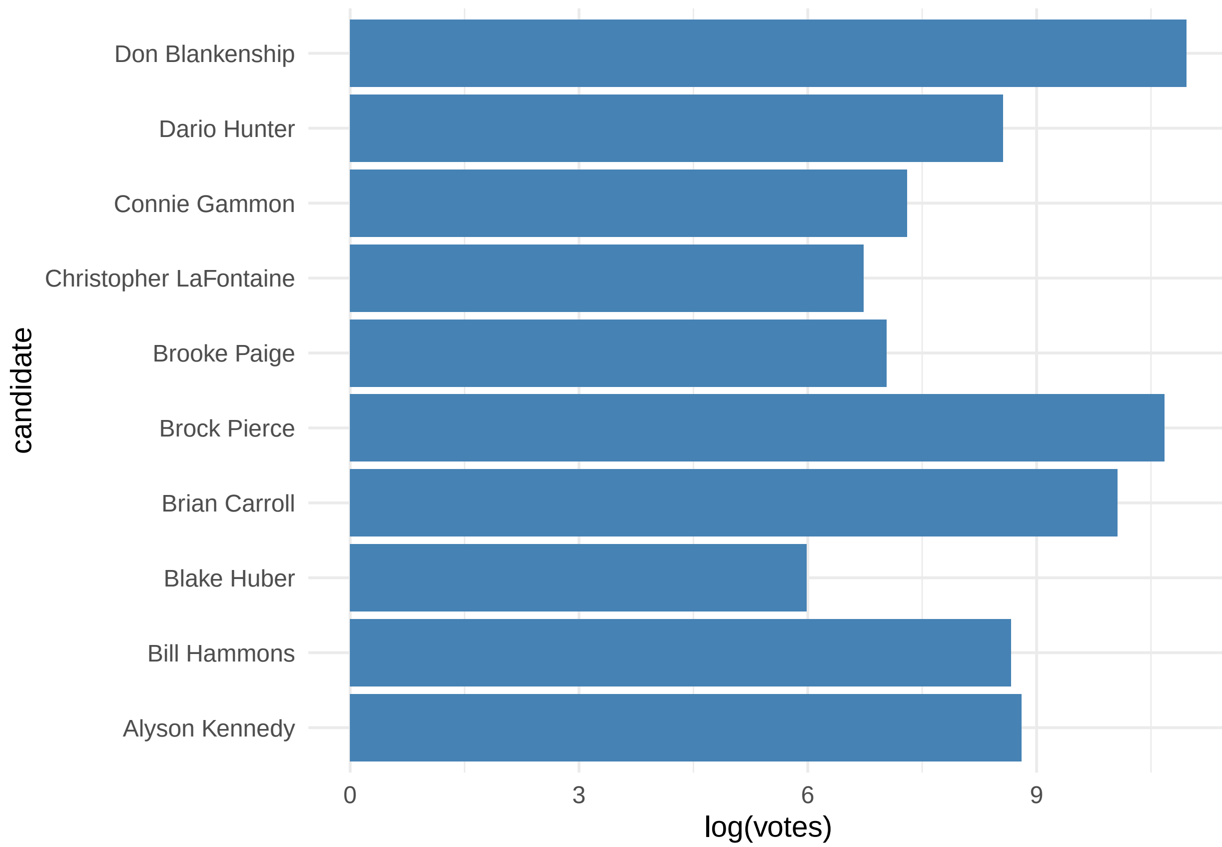
```
## `summarise()` regrouping output by 'state' (override with `.groups` argument)
```

```r
election.total <- election.raw %>%
  group_by(candidate) %>%
  summarise(votes = sum(votes))
```

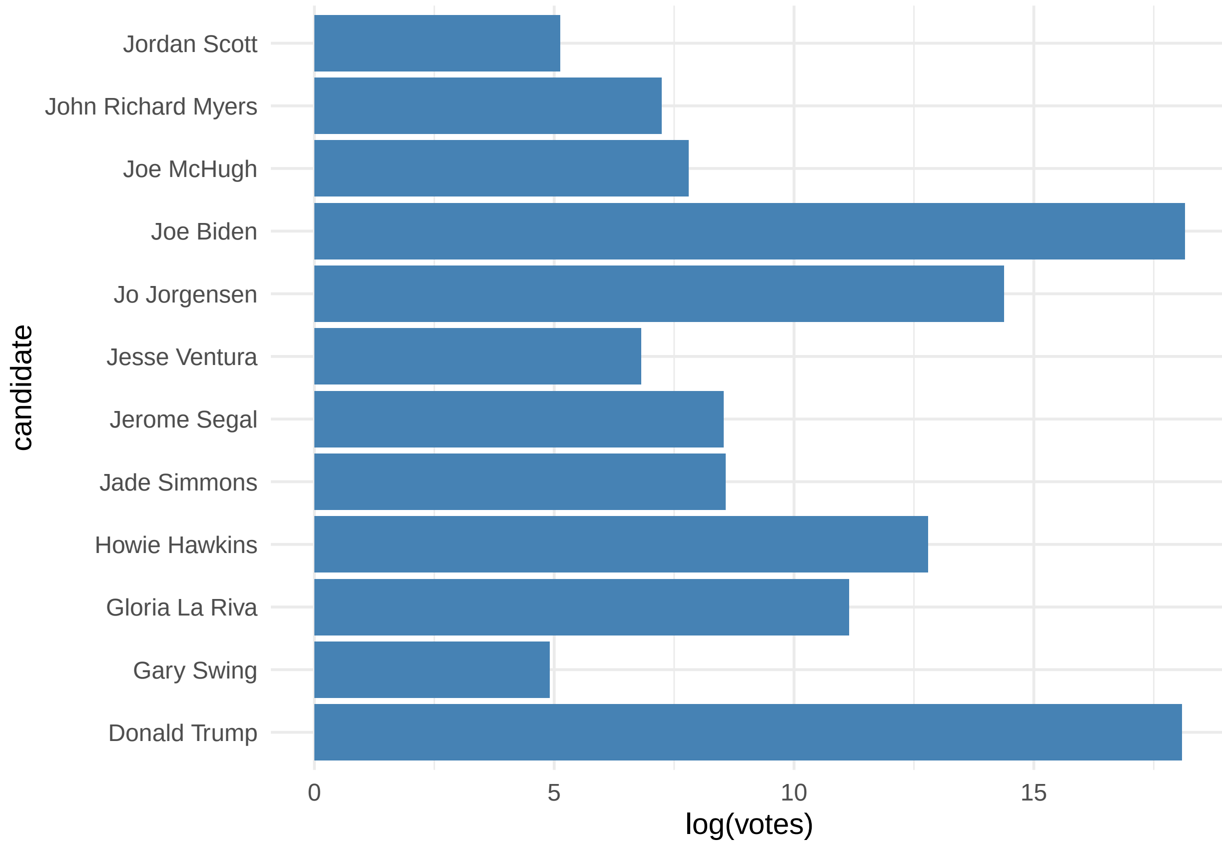
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

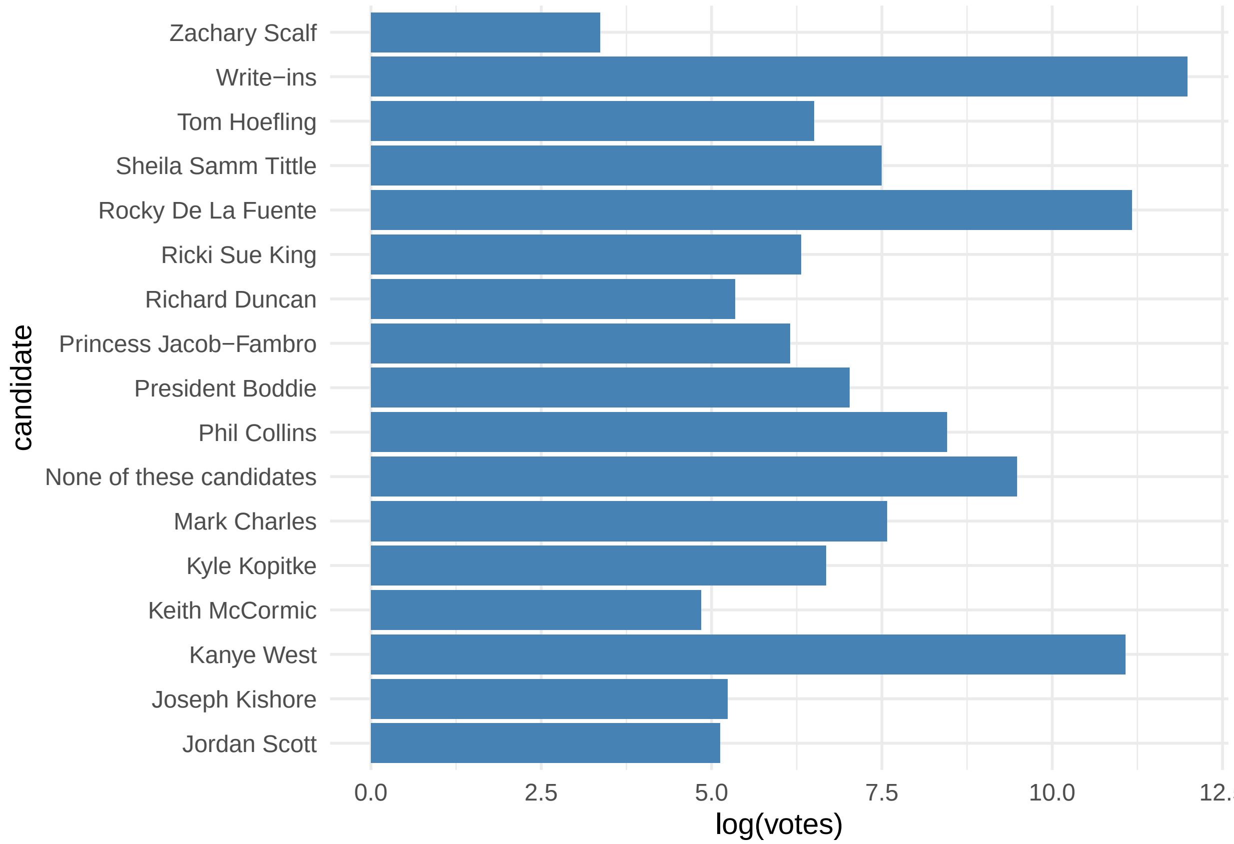
Above we create a state-level summary labeled as election.state and a federal-level summary labeled as election.total.

```
## [1] 38
```




```
## Coordinate system already present. Adding new coordinate system, which will replace the existing one
```


Jordan Scott
John Richard Myers
Joe McHugh
Joe Biden
Jo Jorgensen
Jesse Ventura
Jerome Segal
Jade Simmons
Howie Hawkins
Gloria La Riva
Gary Swing
Donald Trump
candidate
0
5
10
15
log(votes)



There was 38 presidential candidates in the 2020 election and above are three barchart of all votes recieved by each candidate using a log transformation on our total votes.

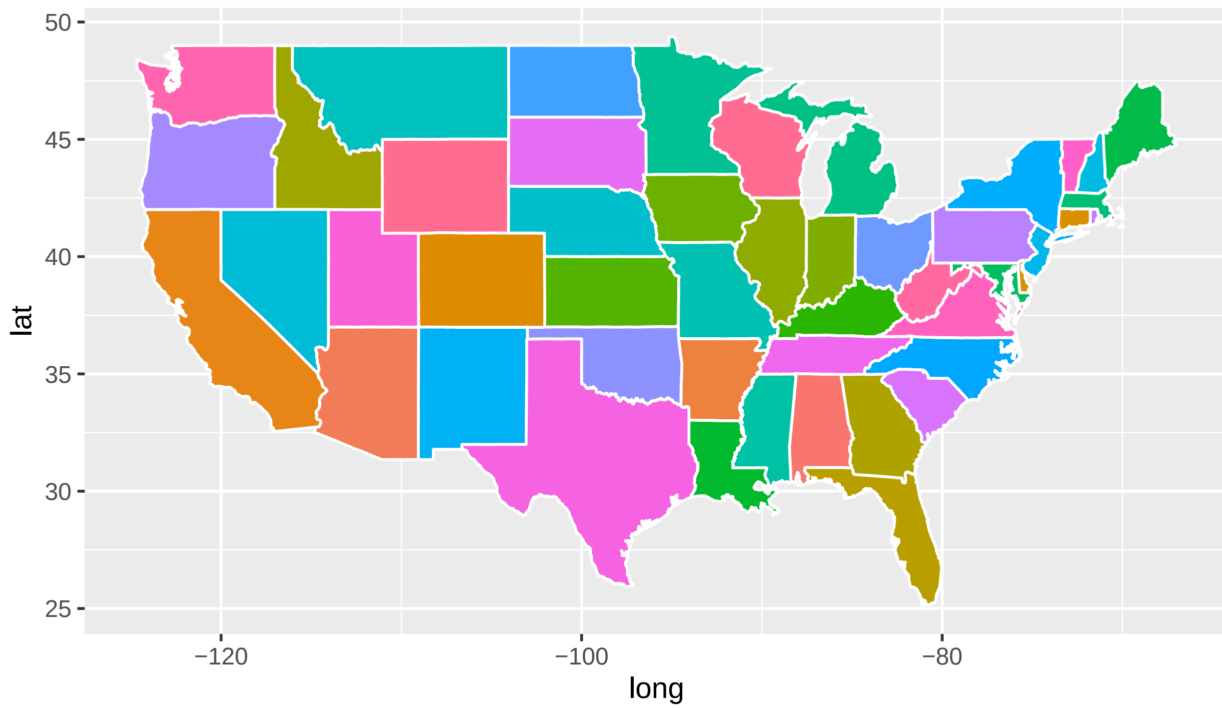
```
county.winner <-election.raw %>%
  group_by(state,county) %>%
  mutate(total = sum(votes), pct = votes/total)
county.winner = top_n(county.winner,1)
```

```
## Selecting by pct
```

```
state.winner <- election.state %>%
  group_by(state) %>%
  mutate(total = sum(votes), pct = votes/total)
state.winner = top_n(state.winner,1)
```

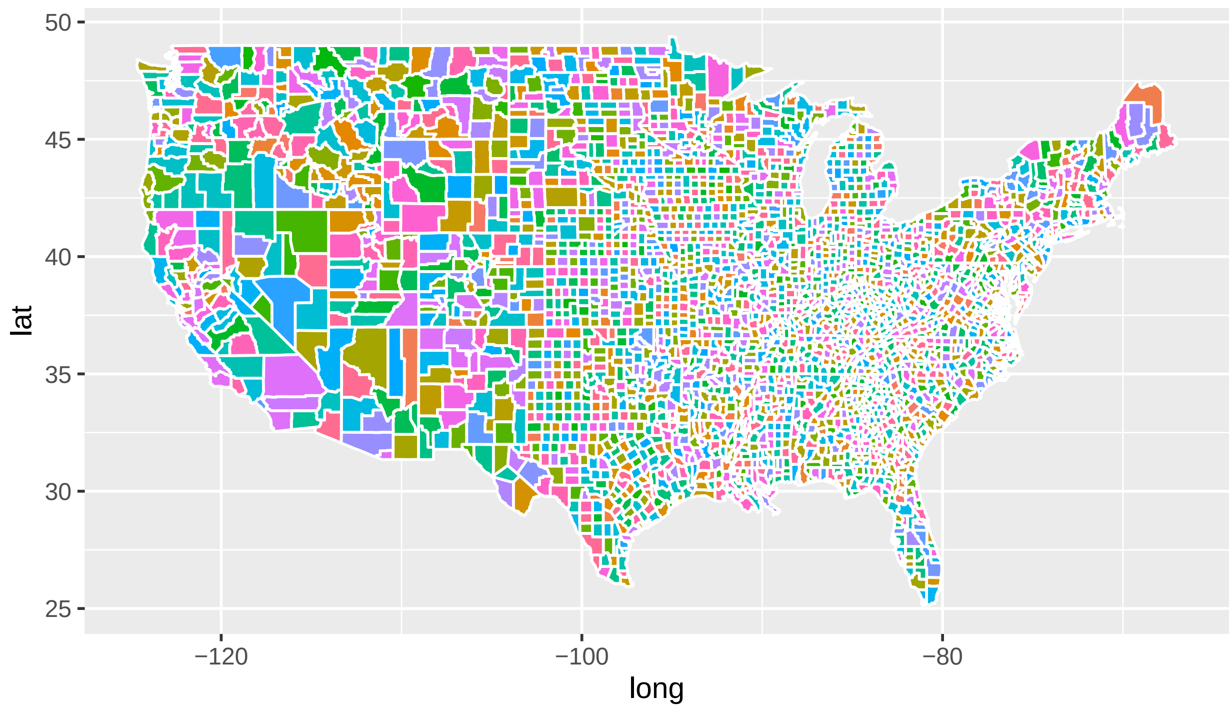
```
## Selecting by pct
```

Above we created two data sets, one set for county.winner which represents who gained the highest proportion of votes at the county level and another data set labeled state.winner which represents who gained the highest proportion of votes at the state level.

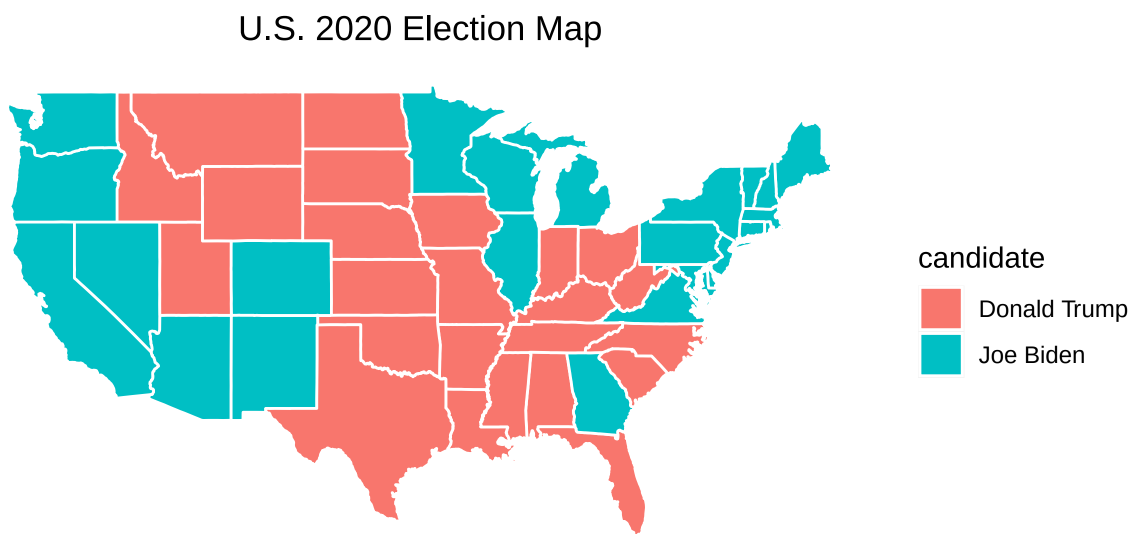


Visual of the U.S. map is shown above.

```
counties <- map_data("county")
ggplot(data = counties) +
  geom_polygon(aes(x = long, y = lat, fill = subregion, group = group),
               color = "white") +
  coord_fixed(1.3) +
  guides(fill=FALSE)
```

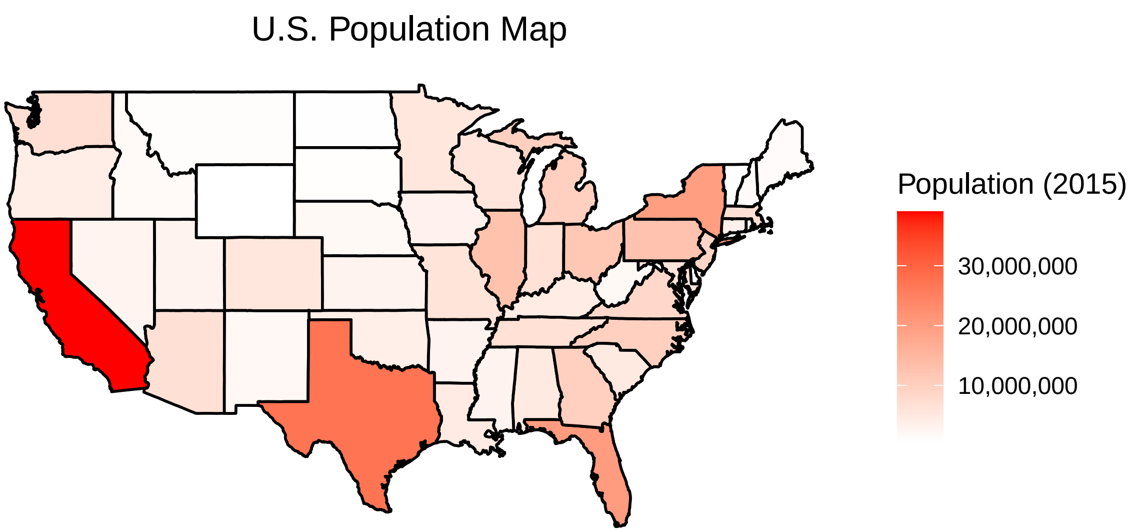


Visual of the U.S. map but at the county-level.



Map of which states were won by each candidate.

Map of California based on which counties each candidate won.



Many people during the election were confused on why Biden won the presidency eventhough it looked like Trump won the majority of each state. I created a heatmap of the U.S. using population as my indicator. This map shows which states contribute the most to our electoral college because of how many electoral votes they carry. The darker the state, the more votes it has and vice-versa. The map shows why Florida was such a big battleground state and has been for the past couple elections. Florida is one of the most populous states in the U.S. and is tied for 3rd with New York for electoral college votes.

```
census <- read_csv("census_county.csv")
```

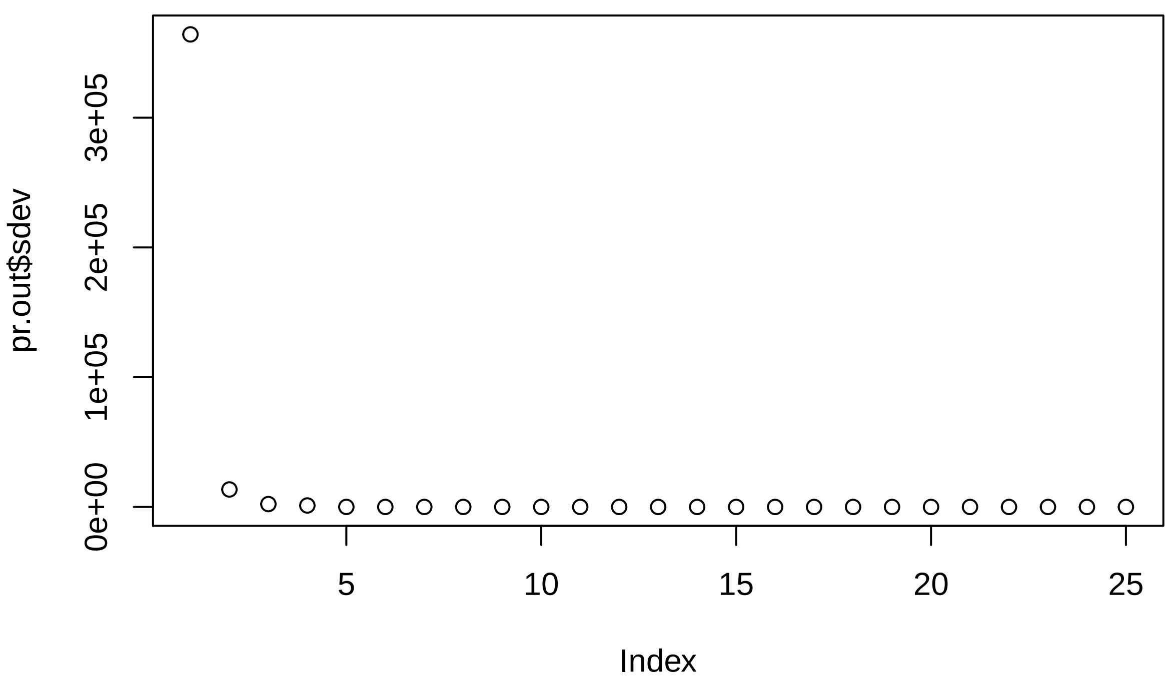
```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double(),
##   State = col_character(),
##   County = col_character()
## )
## i Use `spec()` for the full column specifications.
```

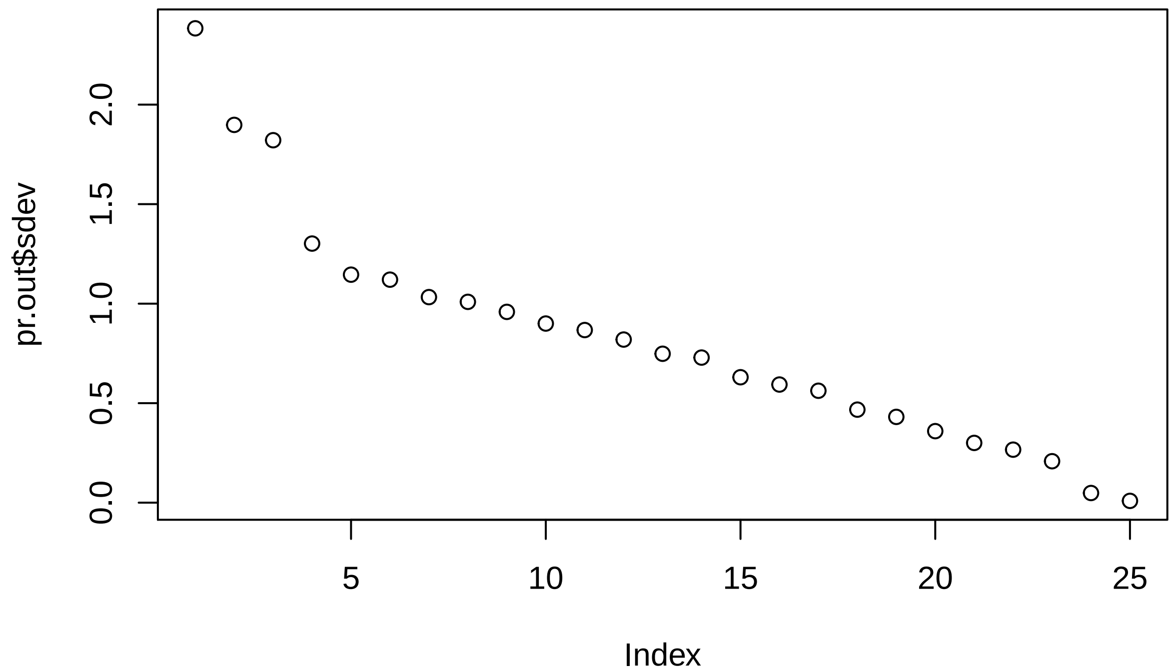
```
census.clean <- census %>%
  na.omit() %>%
  mutate(Men = Men/TotalPop, Employed = Employed/TotalPop,
         VotingAgeCitizen = VotingAgeCitizen/TotalPop,
         Minority = Hispanic + Black + Native + Asian + Pacific)

census.clean <- subset(census.clean,
       select = -c(Hispanic,Black,Native,Asian,Pacific,IncomeErr,IncomePerCap,Walk,
```

```r
                   PublicWork,Construction))
head(census.clean,5)
```

```
## # A tibble: 5 x 28
##   CountyId State County TotalPop   Men  Women White VotingAgeCitizen Income
##      <dbl> <chr> <chr>     <dbl> <dbl>  <dbl> <dbl>            <dbl>  <dbl>
## 1     1001 Alab~ Autau~    55036 0.489  28137  75.4            0.745  55317
## 2     1003 Alab~ Baldw~   203360 0.489 103833  83.1            0.764  52562
## 3     1005 Alab~ Barbo~    26201 0.533  12225  45.7            0.774  33368
## 4     1007 Alab~ Bibb ~    22580 0.543  10329  74.6            0.782  43404
## 5     1009 Alab~ Bloun~    57667 0.494  29177  87.4            0.737  47412
## # ... with 19 more variables: IncomePerCapErr <dbl>, Poverty <dbl>,
## #   ChildPoverty <dbl>, Professional <dbl>, Service <dbl>, Office <dbl>,
## #   Production <dbl>, Drive <dbl>, Carpool <dbl>, Transit <dbl>,
## #   OtherTransp <dbl>, WorkAtHome <dbl>, MeanCommute <dbl>, Employed <dbl>,
## #   PrivateWork <dbl>, SelfEmployed <dbl>, FamilyWork <dbl>,
## #   Unemployment <dbl>, Minority <dbl>
```

Above we cleaned and aggregated the data and labeled it has census.clean and the first 5 rows were printed.





Normalization is important in PCA since it is a variance maximizing exercise. It projects your original data onto directions which maximize the variance.

The first plot above shows the amount of total variance explained in the different principal components where we have not normalized the data. As you can see, it seems like component one explains nearly all of the variance in the data.

If you look at the second picture, we have normalized the data first. Here it is clear that the other components contribute as well. From this structure, the PCA will select to project as much as possible in the direction of TotalPop since that variance is much greater.

```
pc.county <- pr.out$x[,1:2]
```

The First two principle components PC1 and PC2 are saved into a two-column data frame called pc.county.

```
x <- abs(pr.out$rotation[,1])
x <- sort(x, decreasing = TRUE)
head(x,3)
```

```
##       Poverty ChildPoverty     Employed
##     0.3778001    0.3777293    0.3466812
```

Poverty, ChildPoverty, and Employed with the largeest absolute values of the first principal component.

```
pr.out$rotation[,1]
```

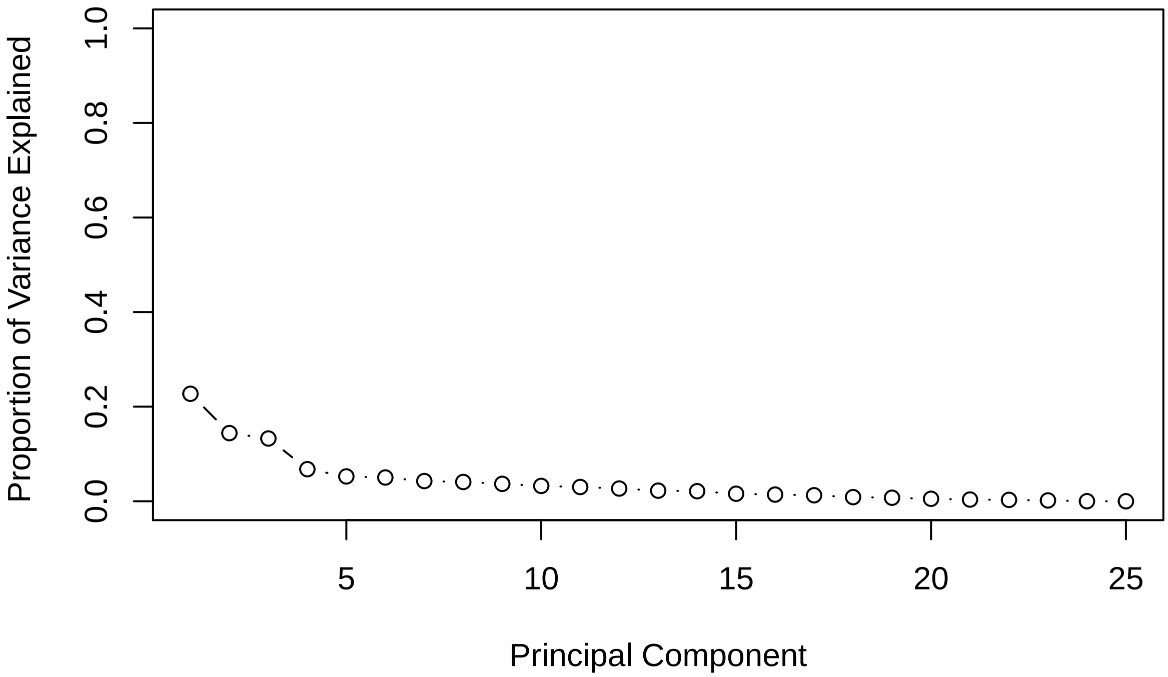
```
##         TotalPop             Men           Women           White
##       0.02771478      0.02689048      0.02746523      0.28919873
## VotingAgeCitizen          Income IncomePerCapErr         Poverty
##       0.01954140      0.31618303      0.09490944     -0.37780006
##     ChildPoverty    Professional         Service          Office
##      -0.37772932      0.22962431     -0.19922770     -0.07371964
##       Production           Drive         Carpool         Transit
##      -0.08521072     -0.11089482     -0.05893345      0.03809840
##      OtherTransp      WorkAtHome     MeanCommute        Employed
##      -0.02050837      0.21232689     -0.08258638      0.34668117
##      PrivateWork    SelfEmployed      FamilyWork    Unemployment
##       0.05463320      0.13492458      0.07247217     -0.33527551
##         Minority
##      -0.29265191
```

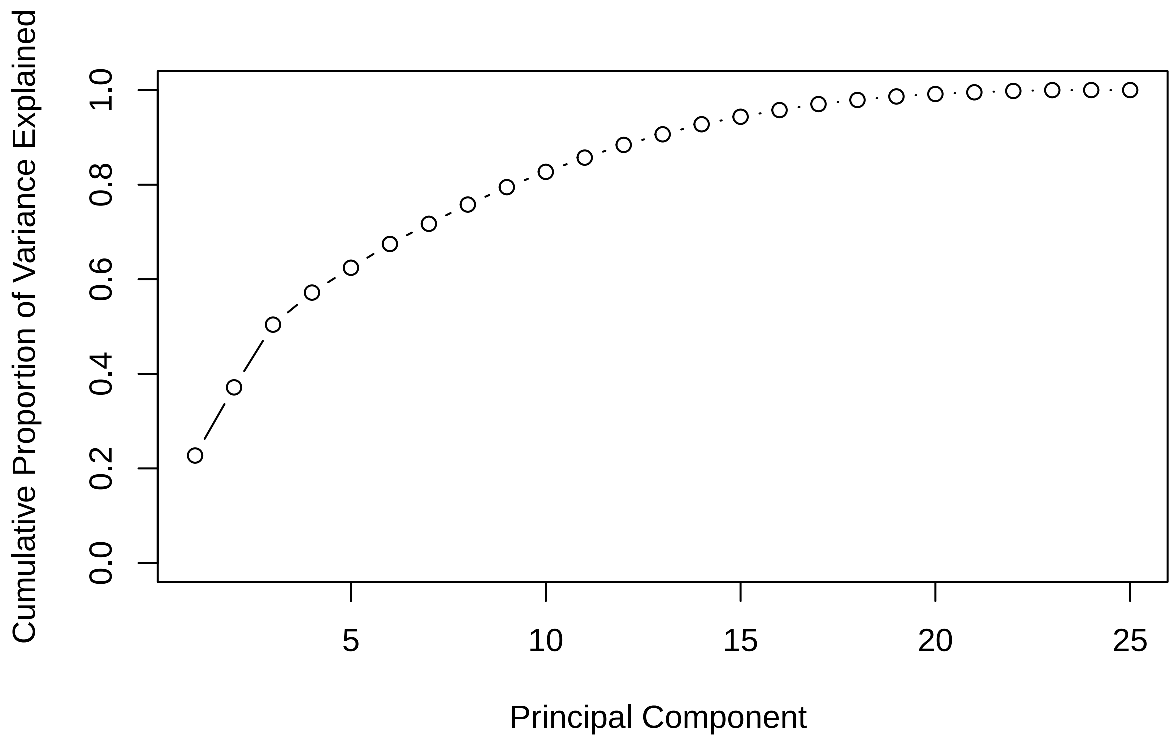
Production, MeanCommute, Minority, Poverty, Drive, ChildPoverty, Carpool, Service, OtherTrandp, Office, and Unemployment all have opposite signs and means there is a negative correlation between these features.

```r
summary(pr.out)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4    PC5     PC6     PC7
## Standard deviation     2.3835 1.8983 1.8213 1.3019 1.1457 1.12064 1.03283
## Proportion of Variance 0.2272 0.1441 0.1327 0.0678 0.0525 0.05023 0.04267
## Cumulative Proportion  0.2272 0.3714 0.5041 0.5718 0.6243 0.67459 0.71726
##                            PC8     PC9   PC10   PC11    PC12    PC13    PC14
## Standard deviation     1.00899 0.95871 0.9000 0.8674 0.81963 0.74815 0.72906
## Proportion of Variance 0.04072 0.03676 0.0324 0.0301 0.02687 0.02239 0.02126
## Cumulative Proportion  0.75798 0.79474 0.8272 0.8572 0.88411 0.90650 0.92776
##                           PC15    PC16    PC17    PC18    PC19    PC20   PC21
## Standard deviation     0.63028 0.59355 0.56233 0.46741 0.43096 0.35933 0.3002
## Proportion of Variance 0.01589 0.01409 0.01265 0.00874 0.00743 0.00516 0.0036
## Cumulative Proportion  0.94365 0.95775 0.97039 0.97913 0.98656 0.99173 0.9953
##                           PC22    PC23    PC24    PC25
## Standard deviation     0.26629 0.20824 0.04842 0.00898
## Proportion of Variance 0.00284 0.00173 0.00009 0.00000
## Cumulative Proportion  0.99817 0.99990 1.00000 1.00000
```

13 PC's required to capture 90% of the variance for the analysis.


Proportion of Variance Explained
0.0
0.2
0.4
0.6
0.8
1.0
5
10
15
20
25
Principal Component



Plots of the proportion of variance explained (PVE) and cumulative PVE shown above.

```r
census.clean.dist <- dist(census.clean[4:28])
set.seed(1)
census.hclust <- hclust(census.clean.dist)
clus = cutree(census.hclust, 10)
table(clus)
```

```
## clus
##    1    2    3    4    5    6    7    8    9   10
## 3111   69    2    9   12    1    2    5    7    1
```

```r
clus[census.clean$County == "Santa Barbara County"]
```

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

```r
#Santa Barbara County is in cluster 1
pc.county.dist <- dist(pc.county)
set.seed(1)
pc.county.hclust <- hclust(pc.county.dist)
clus2 = cutree(pc.county.hclust, 10)
table(clus2)
```

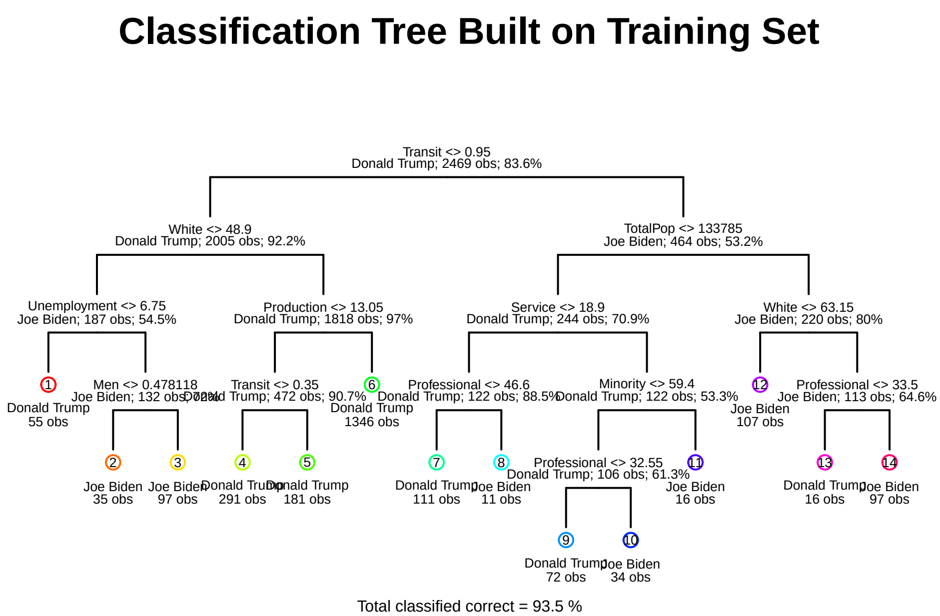
```
## clus2
##    1    2    3    4    5    6    7    8    9   10
## 2114  199  300  114  367   52    5   23    1   44
```

```r
clus2[census.clean$County == "Santa Barbara County"]
```

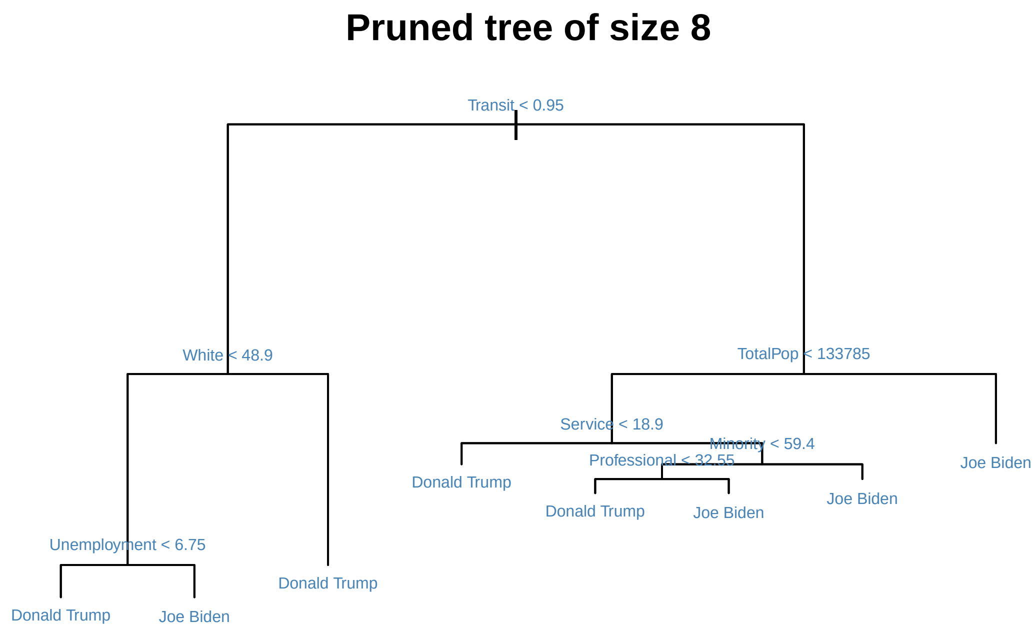
```
## [1] 3
```

I believe our second apporach put Santa Barbara County in a more appropriate cluster because before it was located in cluster 1 with another 3111 observations. Having over 3000 Counties similar to eachother is not very accurate and I am sure the second approach was more accurate than our first because it the observations are more spread out compared to the first method as you can see in the two tables shown.

Removing party affiliaiton from our dataset is encessary because it gives nothing to our analysis.It is not necessary for this variable to be included because the candidates are already being analyzed and their association with the party is already expected.



Question 15



Our pruned tree tells us a story about who voted for who and their current situation or demographic. Starting from left to right you can see that more than 50% of White individuals voted for Donald Trump. Out of these votes the counties that had more than a 6.75% unemployment rate tended to vote for Joe Biden.Going from one side to another when looking at the right side of our pruned tree you can see TotalPop as our first variable. Counties that were more populated voted for Biden and counties that were not voted for Trump. This is consisent with how most elections work because cities which are more populated usually lean democratic and rural areas tend to vote republican.Another interesting piece of information here is that more than 59% of minorities voted for Biden compared to Trump. This decision tree pretty much sums up how the election went. Trump won the White vote but Biden made up ground through the minority vote to win the election.

```
# Predict on test set
#Test error rate
candidate.test = election.te$candidate
test.pred = predict(pt.cv, election.te, type="class")
tree.test.rate <- calc_error_rate(test.pred,candidate.test)
#Train error rate
candidate.train = election.tr$candidate
train.pred = predict(pt.cv, election.tr, type="class")
tree.train.rate <- calc_error_rate(train.pred,candidate.train)
#Update records with tree.train.error and tree.test.error
records[1,1] = tree.train.rate
records[1,2] = tree.test.rate
records
```

```
##            train.error test.error
```

```
## tree      0.07371405  0.1003236
## logistic          NA         NA
## lasso             NA         NA
```

Update records with our rates.

```
glm.fit = glm(candidate ~. , data=election.tr, family=binomial)
prob.training = predict(glm.fit,election.tr, type="response")
election.logistic = election.tr %>%
  mutate(predy = as.factor(ifelse(prob.training<=0.5,"Donald Trump", "Joe Biden")))
table(pred = election.logistic$predy, true =election.tr$candidate)
```

```
##               true
## pred           Donald Trump Joe Biden
##   Donald Trump         2015       104
##   Joe Biden              50       300
```

```
prob.test = predict(glm.fit, election.te, type = "response")
election.logistic2 = election.te %>%
  mutate(predy = as.factor(ifelse(prob.test<=0.5,"Donald Trump", "Joe Biden")))
table(pred = election.logistic2$predy, true =election.te$candidate)
```

```
##               true
## pred           Donald Trump Joe Biden
##   Donald Trump          505        31
##   Joe Biden              14        68
```

Above is where a logistic regression is ran to predict the winning candiate for each county

```
logisitic.train.rate <- calc_error_rate(election.logistic$predy,election.tr$candidate)
logisitic.test.rate <- calc_error_rate(election.logistic2$predy,election.te$candidate)
records[2,1] = logisitic.train.rate
records[2,2] = logisitic.test.rate
records
```

```
##          train.error test.error
## tree      0.07371405 0.10032362
## logistic  0.06237343 0.07281553
## lasso             NA         NA
```

```
summary(glm.fit)
```

```
##
## Call:
## glm(formula = candidate ~ ., family = binomial, data = election.tr)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2406  -0.2230  -0.0855  -0.0269   3.3414
##
## Coefficients:
```

```
##                    Estimate Std. Error z value Pr(>|z|)    
## (Intercept)      -3.225e+01  9.261e+00  -3.483 0.000497 ***
## TotalPop          2.154e-05  2.716e-05   0.793 0.427744    
## Men               2.351e+00  4.897e+00   0.480 0.631211    
## Women            -4.019e-05  5.340e-05  -0.753 0.451670    
## White            -1.828e-01  7.077e-02  -2.584 0.009771 ** 
## VotingAgeCitizen  2.104e+01  2.846e+00   7.394 1.43e-13 ***
## Income           -1.318e-05  1.669e-05  -0.790 0.429694    
## IncomePerCapErr  -2.575e-04  1.402e-04  -1.836 0.066383 .  
## Poverty           5.250e-02  4.351e-02   1.207 0.227598    
## ChildPoverty      1.829e-03  2.550e-02   0.072 0.942827    
## Professional      2.780e-01  4.052e-02   6.860 6.89e-12 ***
## Service           3.139e-01  4.706e-02   6.671 2.55e-11 ***
## Office            7.796e-02  5.159e-02   1.511 0.130785    
## Production        1.371e-01  4.167e-02   3.290 0.001001 ** 
## Drive            -1.407e-01  4.177e-02  -3.367 0.000759 ***
## Carpool          -1.166e-01  5.330e-02  -2.188 0.028687 *  
## Transit           1.018e-01  9.194e-02   1.107 0.268221    
## OtherTransp       1.239e-01  9.840e-02   1.259 0.208036    
## WorkAtHome        3.123e-02  6.432e-02   0.486 0.627232    
## MeanCommute       2.512e-02  2.415e-02   1.040 0.298386    
## Employed          2.746e+01  3.367e+00   8.153 3.54e-16 ***
## PrivateWork       8.733e-02  2.213e-02   3.946 7.96e-05 ***
## SelfEmployed     -1.686e-02  4.783e-02  -0.353 0.724387    
## FamilyWork       -5.819e-01  3.202e-01  -1.818 0.069127 .  
## Unemployment      2.097e-01  4.710e-02   4.452 8.51e-06 ***
## Minority         -4.108e-02  6.915e-02  -0.594 0.552493    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2200.56  on 2468  degrees of freedom
## Residual deviance:  800.09  on 2443  degrees of freedom
## AIC: 852.09
##
## Number of Fisher Scoring iterations: 7
```

The significant variables are somewhat consistent with what I saw in the decision tree. There were a few that were not included such as TotalPop and Minority. Records updated with our rates from our logistic regression. For every unit of change in Unemployment the coefficient will change by 0.2097 and for Employed it will change by about 27.46.

To handle overfitting in our logistic regression we will perform regularization. Running a 10-fold cross validations and selecting the paramater for the logistic regression with LASSO penalty should help us analyze more closely.

```r
bestlam = cv.out.lasso$lambda.min
bestlam
```

```
## [1] 0.0022
```

Our optimal value for lambda is 0.0022.

```r
out = glmnet(x,y,alpha=1,lambda = seq(1, 50) * 1e-4)
lasso.coef=predict(out,type="coefficients",s=bestlam)
lasso.coef
```

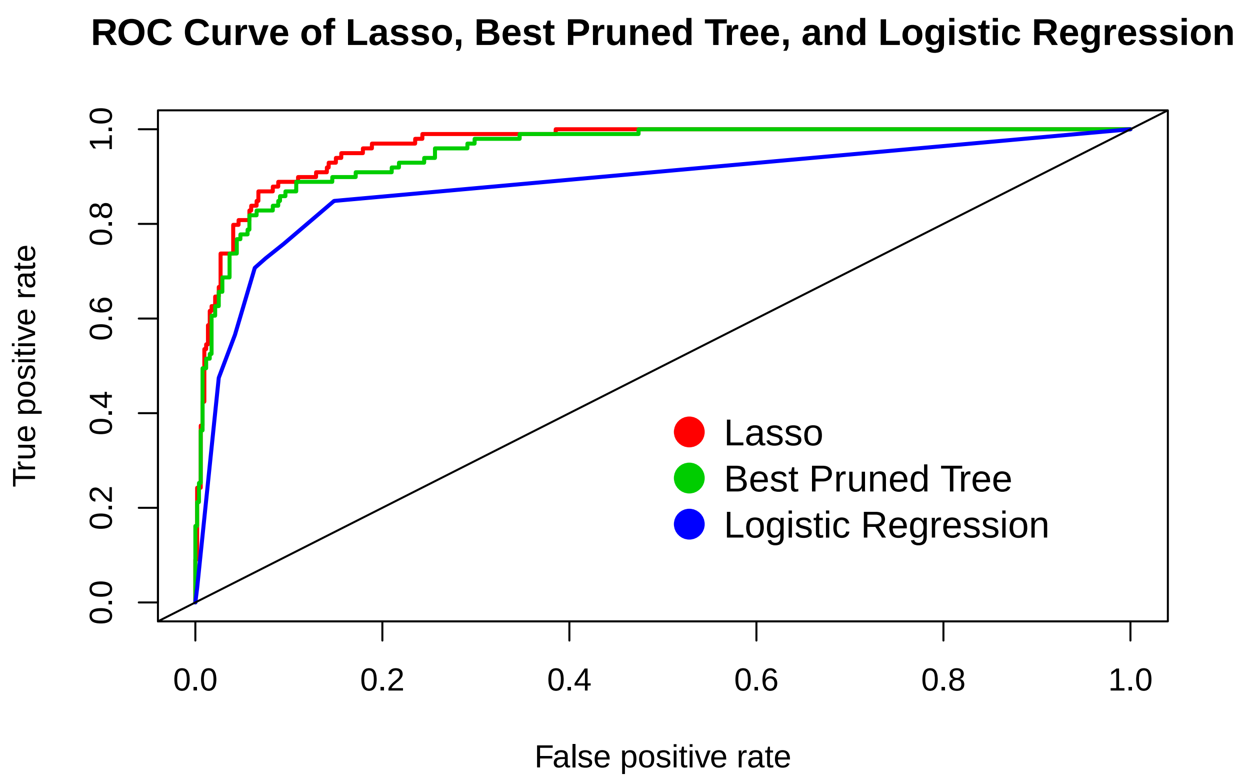
```
## 27 x 1 sparse Matrix of class "dgCMatrix"
##                              1
## (Intercept)      -1.834676e+00
## (Intercept)       .
## TotalPop          9.980334e-08
## Men               .
## Women             3.281371e-08
## White            -7.726437e-03
## VotingAgeCitizen  1.101209e+00
## Income            8.826819e-07
## IncomePerCapErr  -8.403289e-06
## Poverty           1.076907e-02
## ChildPoverty     -1.238740e-03
## Professional      1.847410e-02
## Service           1.634942e-02
## Office            4.743174e-03
## Production        6.216744e-03
## Drive            -7.733986e-03
## Carpool          -4.058022e-03
## Transit          -7.154089e-04
## OtherTransp       1.334428e-02
## WorkAtHome        .
## MeanCommute       3.065695e-03
## Employed          1.553542e+00
## PrivateWork       4.006498e-03
## SelfEmployed     -2.697595e-03
## FamilyWork       -1.557930e-02
## Unemployment      1.313481e-02
## Minority          1.236765e-03
```

The non-zero coefficients of our LASSO are given above.

```r
lasso.mod<- glmnet(x.train, y.train, alpha = 1, lambda = seq(1, 50) * 1e-4)
#test MSE
bestlam <- cv.out.lasso$lambda.min
lasso.pred <- predict(lasso.mod, s = bestlam, newx = x.test)
lasso.test.rate <- mean((lasso.pred-y.test)^2)
#training MSE
lasso.pred <- predict(lasso.mod, s = bestlam, newx = x.train)
lasso.train.rate <- mean((lasso.pred - y.train)^2)
#Update records
records[3,1] = lasso.train.rate
records[3,2] = lasso.test.rate
records
```

```
##          train.error test.error
## tree      0.07371405 0.10032362
## logistic  0.06237343 0.07281553
## lasso     0.06818199 0.07372460
```

Update records with our LASSO rates.



Above is our ROC curves for the decision tree, logistic regression, and LASSO logistic regression using predictions on our test data.

```
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
#SVM
svmfit <- svm(candidate~., data = election.tr, kernel = "radial", cost = 1)
candidate.test = election.te$candidate
yhat.svm = predict(svmfit, newdata = election.te, type = "prob")
table(yhat.svm, candidate.test)
```

```
##                 candidate.test
## yhat.svm         Donald Trump Joe Biden
##   Donald Trump          508        31
##   Joe Biden              11        68
```

```r
SVM.test.error <- calc_error_rate(yhat.svm,candidate.test)
SVM.test.error
```

```
## [1] 0.06796117
```

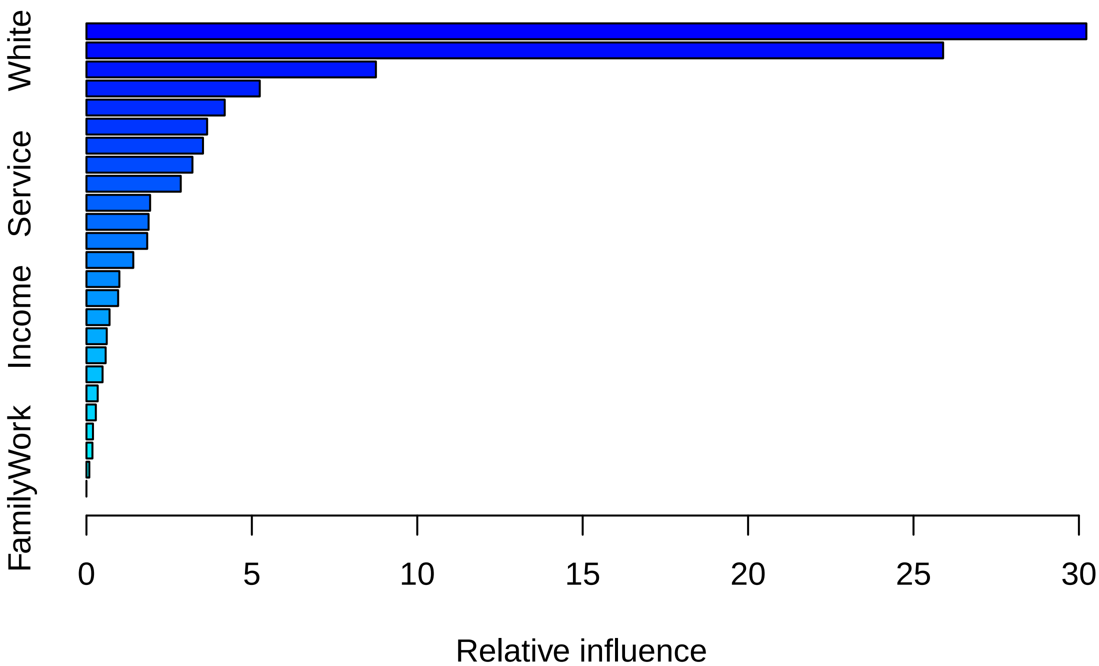
```r
candidate.train = election.tr$candidate
yhat.svm2 = predict(svmfit, newdata = election.tr, type = "prob")
table(yhat.svm2, candidate.train)
```

```
##                 candidate.train
## yhat.svm2        Donald Trump Joe Biden
##   Donald Trump         2041        64
##   Joe Biden              24       340
```

```r
SVM.train.error <- calc_error_rate(yhat.svm2,candidate.train)
SVM.train.error
```

```
## [1] 0.03564196
```

```r
#Boosting & Random Forest
boost.election = gbm(ifelse(candidate == "Joe Biden",1,0)~.,
                     data = election.cl,
                     distribution = "bernoulli", n.trees = 1000,
                     interaction.depth = 2, shrinkage = .01)
summary(boost.election)#Transit and White appeaer to be the most important
```




```
##                                 var     rel.inf
## Transit                     Transit 30.22497136
## White                         White 25.89558996
## Women                         Women  8.74978842
## Professional           Professional  5.23903316
## TotalPop                   TotalPop  4.18049724
## VotingAgeCitizen VotingAgeCitizen  3.64820708
## Employed                   Employed  3.52356588
## Minority                   Minority  3.20392810
## Service                     Service  2.85143405
## Unemployment           Unemployment  1.92504855
## Men                             Men  1.87864983
## Production               Production  1.83662534
## Poverty                     Poverty  1.41608404
## SelfEmployed           SelfEmployed  0.99542514
## OtherTransp             OtherTransp  0.95810868
## Income                       Income  0.69863316
## MeanCommute             MeanCommute  0.61428378
## Drive                         Drive  0.58079823
## IncomePerCapErr     IncomePerCapErr  0.48765232
## ChildPoverty           ChildPoverty  0.34119555
## Office                       Office  0.28440642
## WorkAtHome               WorkAtHome  0.19712891
## PrivateWork             PrivateWork  0.18156799
## Carpool                     Carpool  0.08737681
## FamilyWork               FamilyWork  0.00000000
```

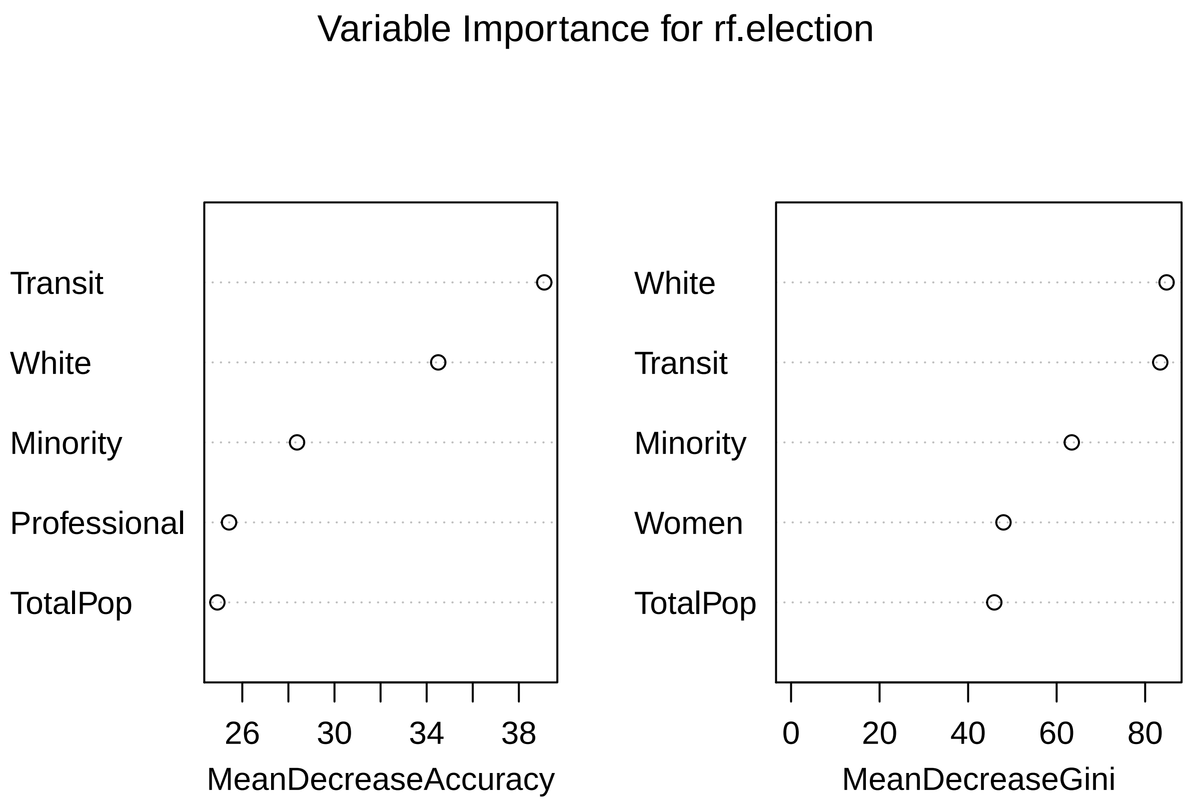
```r
rf.election = randomForest(candidate ~ ., data = election.tr, importance=TRUE)
rf.election
```

```
##
## Call:
##  randomForest(formula = candidate ~ ., data = election.tr, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 6.2%
## Confusion matrix:
##              Donald Trump Joe Biden class.error
## Donald Trump         2014        51  0.02469734
## Joe Biden             102       302  0.25247525
```

```r
#Out-of-bag estimate of error is 6.28%
#5 variables were randomly considered at each split
#500 trees were used to fit the data
importance(rf.election)
```

```
##                   Donald Trump  Joe Biden MeanDecreaseAccuracy MeanDecreaseGini
## TotalPop             19.949546 16.4841995            24.918483        45.918916
## Men                   5.296138 14.6472930            14.221996        18.266809
## Women                20.305219 16.6111697            24.579719        47.991490
## White                24.479583 33.3991852            34.502435        84.842260
## VotingAgeCitizen     19.732583  3.6906565            19.110011        21.151843
## Income               11.924349 14.1448953            19.413271        18.924748
## IncomePerCapErr      12.899036 13.1099098            18.878071        25.807458
## Poverty              11.959553  9.8138817            15.884351        19.281183
## ChildPoverty          6.829350 13.7861682            15.528052        18.934583
## Professional         15.906973 22.9014371            25.426535        33.724337
## Service              17.304659 13.3051983            22.992528        18.905691
## Office               13.208304 -0.1788336            12.694604        12.509494
## Production            9.667370 18.3638591            19.674967        25.392730
## Drive                11.685981 15.4806467            18.233329        19.313819
## Carpool               6.926136  3.6209230             7.911996         8.922278
## Transit              12.997920 39.5227457            39.101731        83.415096
## OtherTransp           3.508169 11.5897787            10.617453        13.294874
## WorkAtHome            4.138065  9.5307960             9.353283        10.175348
## MeanCommute          14.175618  4.3824496            13.950876        14.556479
## Employed             10.749843 18.3637121            20.429666        21.356368
## PrivateWork          14.372918  0.8148817            14.241959        10.305110
## SelfEmployed         10.706155  9.4167398            14.184880        14.327536
## FamilyWork            2.272994  4.6801742             5.406437         3.827222
## Unemployment         10.146679 15.4569957            18.920325        19.559554
## Minority             19.570915 27.7145566            28.375093        63.422564
```

```r
varImpPlot(rf.election, sort=T,
           main="Variable Importance for rf.election", n.var=5)
```




```r
#When viewing the variable importance plot you can see that the order of important variables are simila
#matrix for boosting
yhat.boost = predict(boost.election, newdata = election.te, n.trees=500,
                     type = "response")
yhat.boost = ifelse(yhat.boost > 0.2, "Joe Biden", "Donald Trump")
table(yhat.boost, candidate.test)
```

```
##                 candidate.test
## yhat.boost      Donald Trump Joe Biden
##   Donald Trump           480        15
##   Joe Biden               39        84
```

```r
calc_error_rate(yhat.boost,candidate.test)
```

```
## [1] 0.08737864
```

```r
#matrix for random forest
yhat.rf = predict(rf.election,newdata = election.te, type = "prob")
yhat.rf = ifelse(yhat.rf[, 2] > 0.2, "Joe Biden", "Donald Trump")
table(yhat.rf, candidate.test)
```

```
##                 candidate.test
## yhat.rf         Donald Trump Joe Biden
##   Donald Trump           473        11
##   Joe Biden               46        88
```

```r
calc_error_rate(yhat.rf,candidate.test)
```

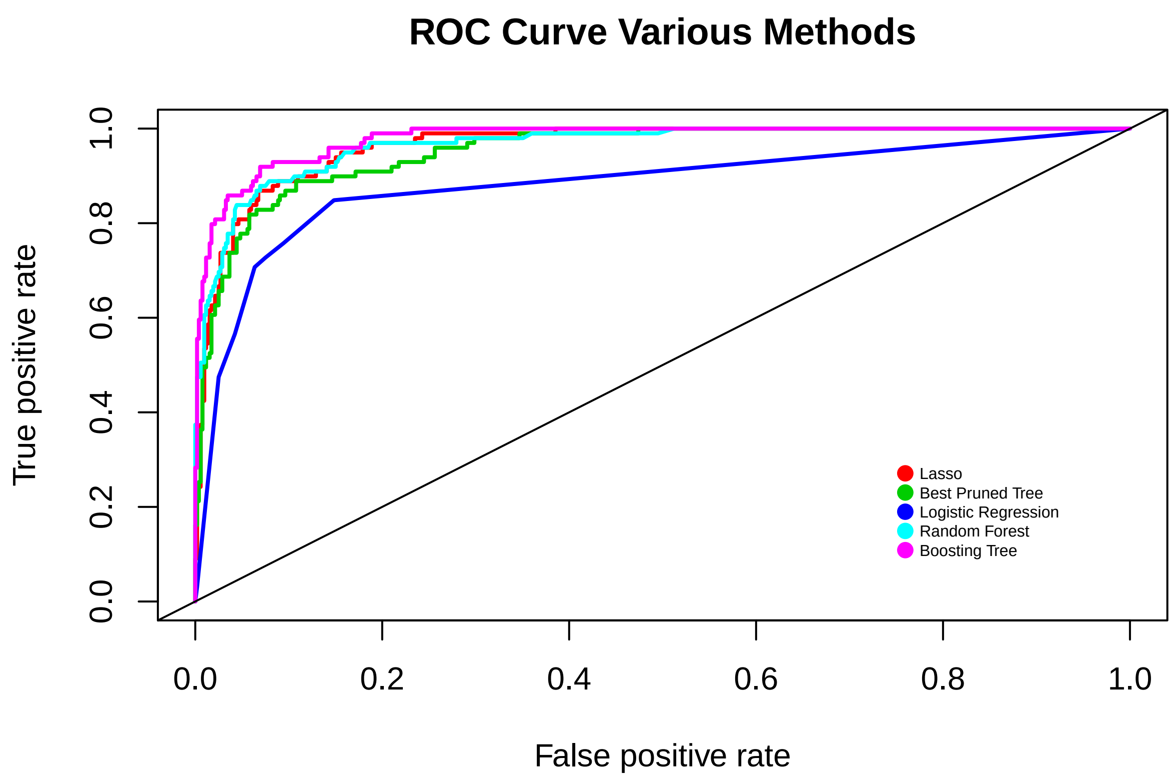
```
## [1] 0.09223301
```

```r
#ROC for RF
prob.training5 = predict(rf.election, newdata = election.te,
                         type = "prob")
pred5 = prediction(prob.training5[, 2], election.te$candidate)
perf5 = performance(pred5, measure="tpr", x.measure="fpr")
#ROC for Boosting
prob.training6 = predict(boost.election,newdata = election.te,
                         type = "response")
```

```
## Using 1000 trees...
```

```r
pred6 = prediction(prob.training6, election.te$candidate)
perf6 = performance(pred6, measure="tpr", x.measure="fpr")
```

Above we will view 3 more classification methods to get a better understanding of what we are looking at. The three methods we chose were a SVM, random forest, and a boosted tree. We will calculate their error rates and visualize how random forest and the boosted tree match up on the ROC curve plot alongside our previous methods. Our error rates for each method is calculated and shown above.



Above is our updated ROC curve that includes random forest and our boosted tree. the methods match up decently with our other methods.

```
training.samples <- election.cl$candidate %>%
  createDataPartition(p = 0.8, list = FALSE)
train.data <- election.cl[training.samples, ]
```

```
## Warning: The `i` argument of ``[`()` can't be a matrix as of tibble 3.0.0.
## Convert to a vector.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
test.data <- election.cl[-training.samples, ]
preproc.param <- train.data %>%
  preProcess(method = c("center", "scale"))
```

Estimate preprocessing parameters

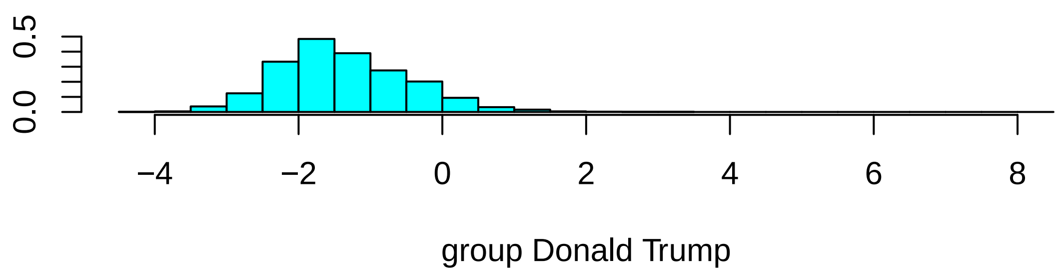
```
# Transform the data using the estimated parameters
train.transformed <- preproc.param %>% predict(train.data)
test.transformed <- preproc.param %>% predict(test.data)
```

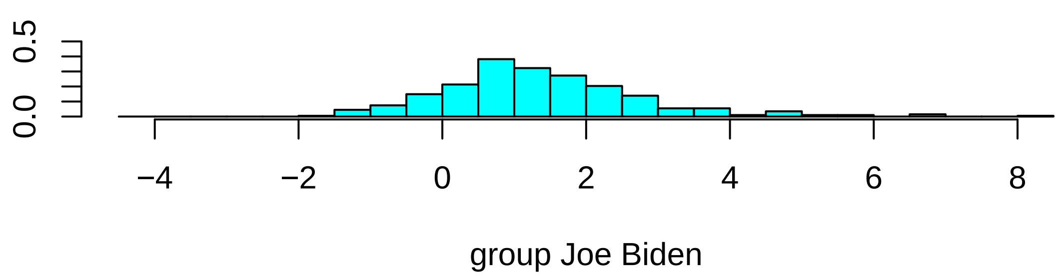
Transform the data using the estimated parameters

```
## [1] 0.9237013
```

```
## Call:
## lda(candidate ~ ., data = train.transformed)
##
## Prior probabilities of groups:
## Donald Trump    Joe Biden
##    0.8369081    0.1630919
##
## Group means:
##                TotalPop         Men      Women      White VotingAgeCitizen
## Donald Trump -0.1770307  0.04896083 -0.1773762  0.2174339       0.07431543
## Joe Biden     0.9084353 -0.25124319  0.9102085 -1.1157650      -0.38135065
##                   Income IncomePerCapErr    Poverty ChildPoverty Professional
## Donald Trump -0.05923605      0.07325937 -0.0753016  -0.05461986   -0.1272518
## Joe Biden     0.30397060     -0.37593148  0.3864112   0.28028256    0.6529943
##                 Service      Office Production       Drive     Carpool
## Donald Trump -0.0914435 -0.04117299  0.1172535  0.09544651  0.01628883
## Joe Biden     0.4692436  0.21127975 -0.6016881 -0.48978505 -0.08358634
##                 Transit OtherTransp   WorkAtHome MeanCommute    Employed
## Donald Trump -0.1455481  -0.0874138 -0.007479427 -0.02914272 -0.03011628
## Joe Biden     0.7468820   0.4485651  0.038380779  0.14954624  0.15454210
##               PrivateWork SelfEmployed  FamilyWork Unemployment   Minority
## Donald Trump  0.004372071   0.07948346  0.03599982  -0.09565376 -0.2140474
## Joe Biden    -0.022435344  -0.40787046 -0.18473356   0.49084860  1.0983872
##
## Coefficients of linear discriminants:
##                          LD1
## TotalPop         -1.14134979
## Men               0.02485773
```

```
## Women               1.41901298
## White              -0.83767902
## VotingAgeCitizen    0.35548089
## Income              0.11372692
## IncomePerCapErr    -0.08101481
## Poverty             0.60015266
## ChildPoverty       -0.23230756
## Professional        0.67887869
## Service             0.36022024
## Office              0.11189771
## Production          0.30891199
## Drive              -0.47896965
## Carpool            -0.14423104
## Transit            -0.21115464
## OtherTransp         0.06099535
## WorkAtHome         -0.08929833
## MeanCommute         0.14070934
## Employed            0.57629576
## PrivateWork         0.14604327
## SelfEmployed       -0.01511049
## FamilyWork         -0.05234425
## Unemployment        0.20698790
## Minority            0.20154646
```






```
## [1] "class"     "posterior" "x"
```

We will fit our model, make predictions, and produce plots of the linear discriminants, obtained by computing LD1 and LD2 for each of the training observations.

```
head(predictions$class, 6)
```

```
## [1] Joe Biden    Joe Biden    Joe Biden    Donald Trump Donald Trump
## [6] Donald Trump
## Levels: Donald Trump Joe Biden
```

```
# Predicted probabilities of class memebership.
head(predictions$posterior, 6)
```

```
##   Donald Trump   Joe Biden
## 1 0.0349178278 0.965082172
## 2 0.0002218756 0.999778124
## 3 0.3562617827 0.643738217
## 4 0.9966726129 0.003327387
## 5 0.8943715516 0.105628448
## 6 0.9975448801 0.002455120
```

```
# Linear discriminants
head(predictions$x, 3)
```

```
##        LD1
## 1 2.733356
## 2 4.596601
## 3 1.735671
```

We inspect our results with the head() function.

```
#MODEL ACCURACY
mean(predictions$class==test.transformed$candidate)
```

```
## [1] 0.9237013
```

```
#QDA
library(MASS)
# Fit the model
model <- qda(candidate~., data = train.transformed)
# Make predictions
predictions <- model %>% predict(test.transformed)
# Model accuracy
mean(predictions$class == test.transformed$candidate)
```

```
## [1] 0.8912338
```

For our method we wil use a Linear discriminant analysis (LDA) to predict the class of candidate. Before we perform our LDA we will consider the univariate distributions of each variable and make sure they are normally distributed. If our data was skewed we would perform some kind of transformation on it. We will also consider removing outliers as well to standardize our data.

LDA determines group means and computes, for each individual, the probability of belonging to the different groups. The individual is then affected to the group with the highest probability score.

We finish this analysis up by computing our model accuracy which is 0.9123377.

In this analysis it shows us that all models can be used to predict the winner and what you choose to tackle this problem is your choice. The error rates for a good portion of the methods implemented in this analysis were relatively close to one another. Using a logistic or linear regression may not differ so much from using a SVM or LDA. Predicting election winners is tough and you could look at 2016 when every big media news outlet had Clinton by a landslide. Data collection is not consistent either because one party refuses to participate in polling and census data because of the fear of lying and mistrust with media. Collecting more data fgor any analysis is great but for this one especially it would make our predictions alot more helpful and useful. This project shows the different methods and angles an analysis could take in trying to predict winners of future elections but no matter which you choose, you can not predict with 100% certainty of who will be the net U.S. President.