# Election Project By: Dae Woong Ham, Anci Sun, Jessica Hu, Winnie Yan, Jiayi Huang

# Introduction

This project is a geo-political project analyzing the elections and ultimately building two predictors related to the 2016 election results. To mainly outline this project, all of us applied our web scraping skills to extract the data necessary from online sources to build a final data frame with many variables necessary to analyze the elections including election results for four elections starting from 2004 on a county level. After this data frame was built one very specific map was coded in great detail to explore any variable or factor in the 2016 election. The one we chose for our project was exploring the relationship between white proportions in counties with the proportion of trump votes. This will be later elaborated on in part 3 of this report. Lastly, we built our predictors using classification trees for predicting the 2016 results and used the K-NN method to predictor the change from 2012 to 2016. Our general step to organize information was done through informative titles and detailed explanation before each code chunk including analysis for any plot given and logic used if unclear.

# Data Description

The final data frame had a total of 3082 rows which each corresponded to a county and many variables related to numerous elections and census data about each county. The columns are divided into three main sections: the geography of the county, the census data of the county, and the election information about the county.

The geography of the county includes the state in which the county is located in and the latitude and longtitude. The state variable is espicially important for merging purposes in part 1 of the project and the latitude and longtitude becomes important for drawing any sorts of map.

The census data takes the bulk of the variables with around 37 extracted variables. These variables describe statistics about the following things: employment, income, education, language spoken, native born, and white proportion. These things become espicially important when building our predicators and also become important when analyzing key variables that could have influenced the election or not. Later in part 3, the world map was constructed off the white proportion variable and how that impacted the election of 2016.

The last main section of the columns are the election votes themselves. The election results that were included were: 2004, 2008, 2012, and 2016. These four election votes all came from different sources, where some sources gave us counts and some gave us proportions. To standardize things, all proportions were given in a real number between 0-1 not in a % format. Moreoever, for sources that weren't given proportion, proportions were manually created by simple divisions from counts. 2012 and 2016 were espicially important election results to build predicators and then test these predicators for actual 2016 results.

# PART 3: Creating A Map

We were interested in exploring not just Democratic and Republican states on the map but correlation between each county's specific census variable and whether they more Republican (Trump) or more Democrat (Clinton) according to this variable. The variable we chose to explore was the proportion of whites in the county. Since many of Trump's policies and his action point towards a white supremacy and ignorance of minority Americans,

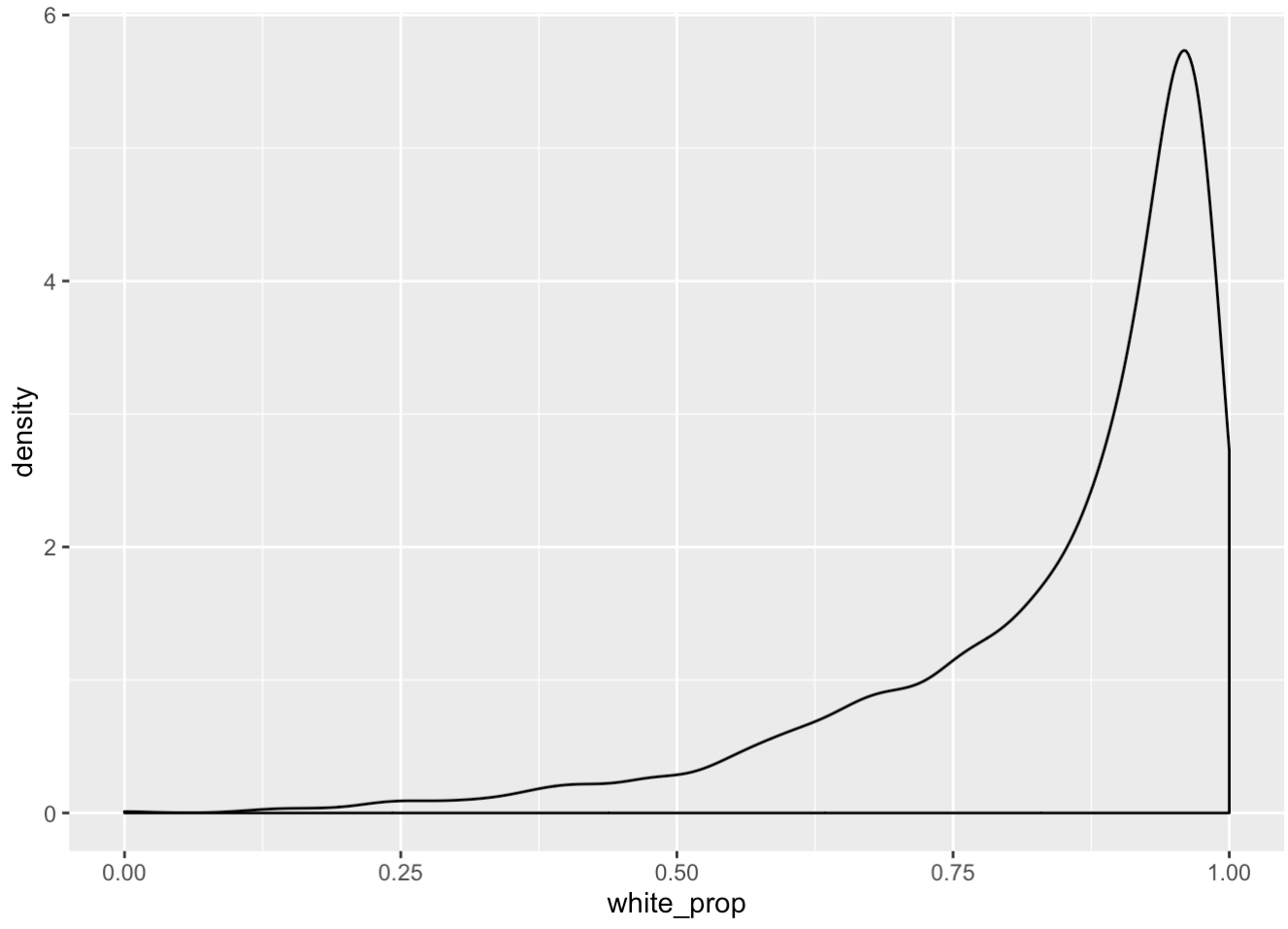
we expect that we will see a positive correlation where counties with bigger white proportion will have bigger proportion who voted for Trump. In order to achieve this, I didn't draw data from my final data frame instead I chose to use a specific merged data frame I created just for this map so I lose the least amount of data.

In order to create the map, I first used the maps package that was extremely useful in giving all the latitude and longtitude of each county and for each county allowing the whole map of the United States' with borders on each counties to be drawn (excluding hawaii). The drawing itself took place within ggplot using geom_polygon to draw the map. After drawing the map, I realized I needed a way to show the correlation, so I drew inspiration from previous homeworks and knew that the size and color are two ways to track down variables. However, since both proportions are continous variables, if I chose to just simply designate those variables in size and color I would end up with a continous scale that R makes for me, and for visibility sake it would be hard to see the correlation because there would be so many colors and so many sizes.

To solve this problem, I ended up creating factors which I then could designate the levels I wanted to use to differentiate the colors and sizes. Now all I needed to do was figure out these levels. For this purpose, I created exploratory plots using simple density curves, as these are best for analyzing one continous variable along with the summary function to see the statistics. I first found out that the proportion of whites had a heavily left skewed data where the average for each county was around 0.85. Therefore, I thought it would be best just to allocate the size of the circle to be the proportion of whites with only two levels, below 85% and above 85%. This binary factor would prove to be useful later in the map because then we would see a huge difference between the sizes of the circle hence very easy to differentiate counties with big white proportion and small white proportion. Next, the proportion who voted for Trump was a little more spread out so I broke it up into 5 parts. The reasoning behind breaking it up into 5 parts was directly affected by the fact that in the electoral vote system there are states that go heavily democratic, heavily republican, and swing states. Within these swing counties, for our case, there were those that went just slightly Republican and slightly Democrat. The "slightly" boundary I chose was if they're proportions were within 3 percent of each other. I thought this was reasonable enough. So these were the extra two factors I created.

After making these factors, creating the map was all up to ggplot syntax. I used geom_jitter to jitter the points I made to look more visually pleasing. I then made sure my alpha value was enough to transparently see the dots. After that I just labelled the legend and the map itself, and chose the colors of the dots to reflect roughly democratic blue and republican blue depending on the proportion who voted for trump.

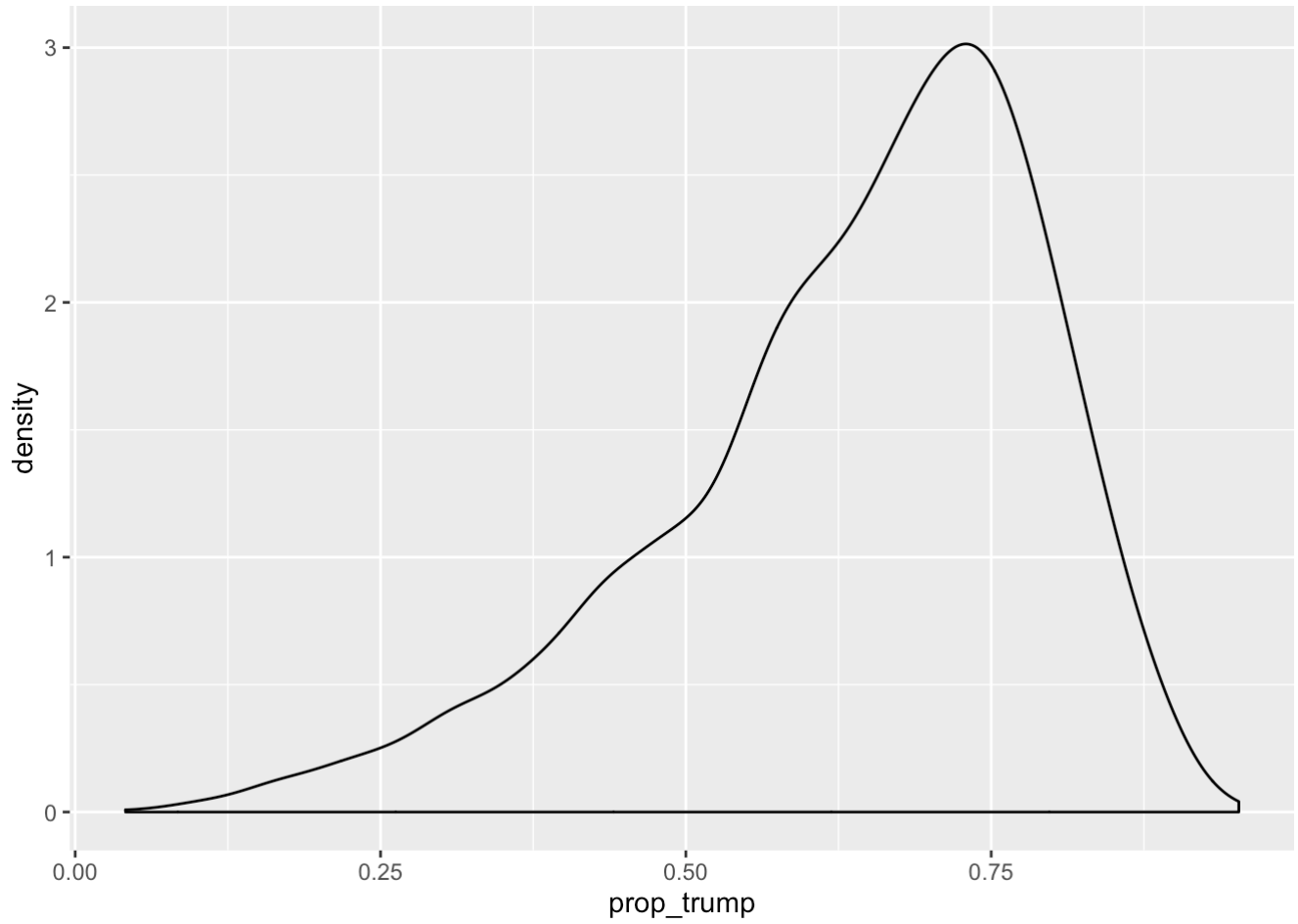
```
require(maps)
require(ggplot2)
all_county = map_data("county")

#for the purpose of this map let's just merge the census data and trumps and the longtit
ude and latitude data to lose the least possible information
#this merge will borrow parts from above
a = merge(merged_census_df, lat_long_df, by.x = c("state", "county"), by.y = c("state",
"county"))
a$county = tolower(gsub(" County", "", a$county))
#get rid of parish in data frame a
a$county = gsub(" parish", "", a$county)
#fix st. problem
a$county = gsub("st[.]", "st", a$county)
a$county = gsub("'", "", a$county)
trump_df = merge(a, election_2016_result, by.x = c("state", "county"), by.y = c("states"
, "county"))

#get rid of hawaii for mapping purposes
trump_df = trump_df[-(which(trump_df$state == "hawaii")), ]

#to create a visually informative graph I will not use every single continous level of p
roportion to differentiate but instead cut it off at certain level and use factors. To d
ecide the right factors let's explore the plots
ggplot(data = trump_df, aes(white_prop)) + geom_density()
```



```
summary(trump_df$white_prop)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.7700  0.9100  0.8419  0.9600  1.0000
```

```
ggplot(data = trump_df, aes(prop_trump)) + geom_density()
```

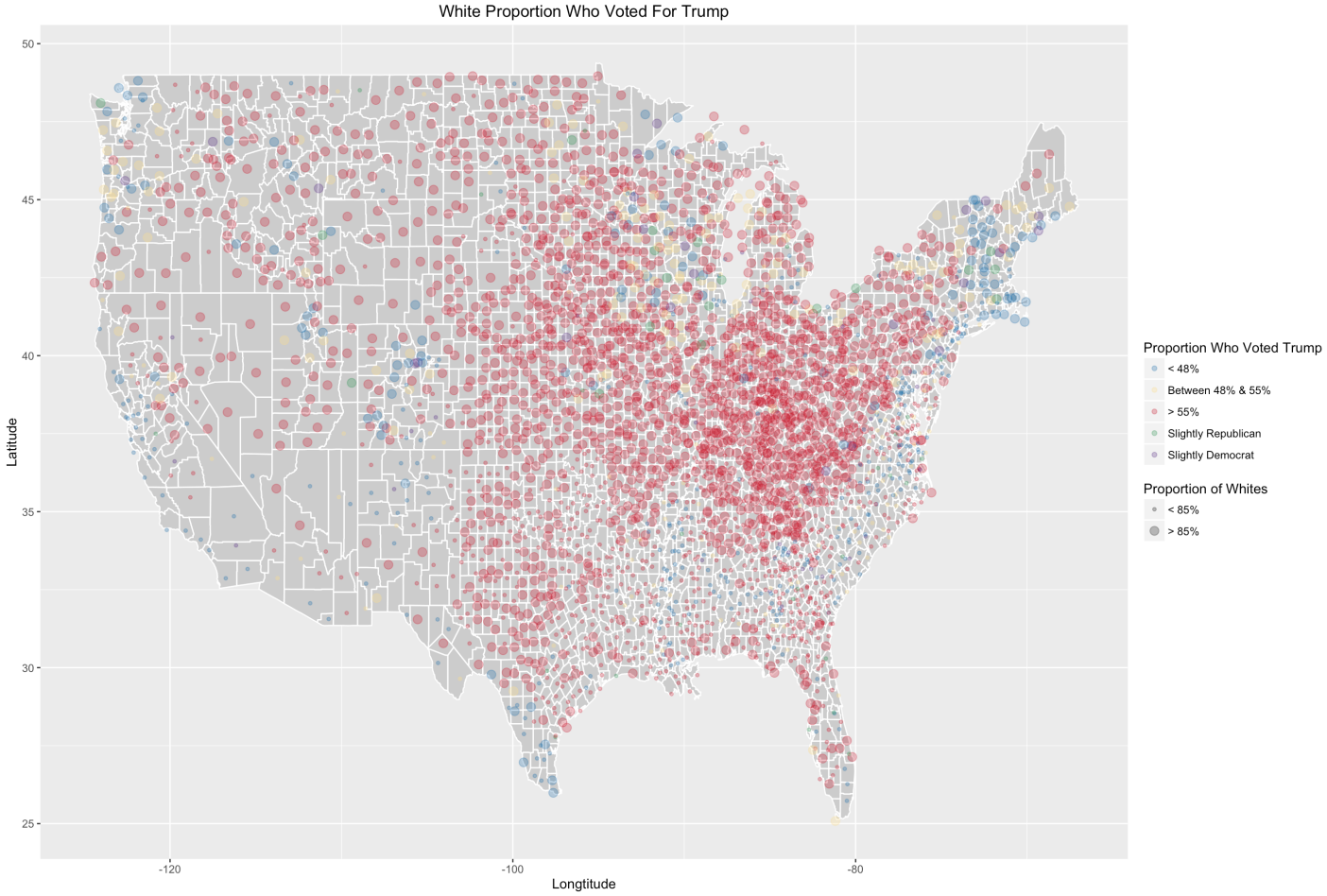



```
summary(trump_df$prop_trump)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.04122 0.55020 0.66820 0.63690 0.75160 0.95270
```

```
#let's create these factors, 2 levels for the white proportion one having less than 85%
 and one having more than 85% this will be used for the size of the circles for plot. Th
e reasoning I chose 85 is because that's around the mean
trump_df$white_prop_factor = rep(0, length(trump_df$white_prop))
trump_df$white_prop_factor[trump_df$white_prop >= 0.85] = "> 85%"
trump_df$white_prop_factor[trump_df$white_prop < 0.85] = "< 85%"
trump_df$white_prop_factor =
  factor(trump_df$white_prop_factor, levels = c("< 85%", "> 85%"), labels = c("< 85%",
"> 85%"))


#next for the proportion of trump let's create 4 factor levels one for below 0.55, for b
etween 0.55 and 0.75, and then finally above 0.75
trump_df$prop_trump_factor = rep(0, length(trump_df$white_prop))
trump_df$prop_trump_factor[trump_df$prop_trump < 0.48] = "< 48%"
trump_df$prop_trump_factor[(trump_df$prop_trump >= 0.48) & (trump_df$prop_trump <= 0.55)
] = "Between 48% & 55%"
trump_df$prop_trump_factor[trump_df$prop_trump > 0.55] = "> 55%"
trump_df$prop_trump_factor[(trump_df$prop_trump - trump_df$prop_clinton) < 0.03 & (trump
_df$prop_trump - trump_df$prop_clinton) > 0] = "Slightly Republican"
trump_df$prop_trump_factor[(trump_df$prop_clinton - trump_df$prop_trump) < 0.03 & (trump
_df$prop_clinton - trump_df$prop_trump) > 0] = "Slightly Democrat"
trump_df$prop_trump_factor =
  factor(trump_df$prop_trump_factor, levels = c("< 48%", "Between 48% & 55%", "> 55%",
"Slightly Republican", "Slightly Democrat"), labels = c("< 48%", "Between 48% & 55%", ">
 55%", "Slightly Republican", "Slightly Democrat"))

county_map = ggplot() + geom_polygon(data=all_county, aes(x=long, y=lat, group = group),
 colour="white", fill="grey80")

trump_white_prop_map =
  county_map + geom_jitter(data = trump_df, position=position_jitter(width=0.5, height=
0.5), aes(x = county_long, y = county_lat, size = white_prop_factor, color = prop_trump_
factor), alpha = 0.3) +
  scale_size_manual(name="Proportion of Whites", values = c(1,3)) +
  scale_colour_manual(name = "Proportion Who Voted Trump", values = c("#0571b0", "#fee09
0", "#ca0020", "#1a9850", "#542788")) +
  labs(title = "White Proportion Who Voted For Trump", x = "Longtitude", y = "Latitude")
```

```
trump_white_prop_map
```




# Part 3: Analyzing the map

As expected visually there is a story to tell. First of all, let's look at the two sides of the extreme. The small dots that are colored blue represents the counties with relatively small white proportion and relatively small percentage that voted for trump. These are the dots that represent the largely democratic states. And as expected, there are a lot more small blue dots than big blue dots, showing that out of all the strongly democratic counties who didn't vote more for Trump, tended to be the counties who had smaller white proportions. Next, the other end of the extreme says a similar but the opposite story. With the exception of small red dots in the mid-east south region of America, most red dots were big dots showing that counties who went strongly for Trump tended to be counties that had more white proportion. Also it's worthy to note how many more red dots there were than blue dots showing how Trump won a much larger number of counties than Hilary did.

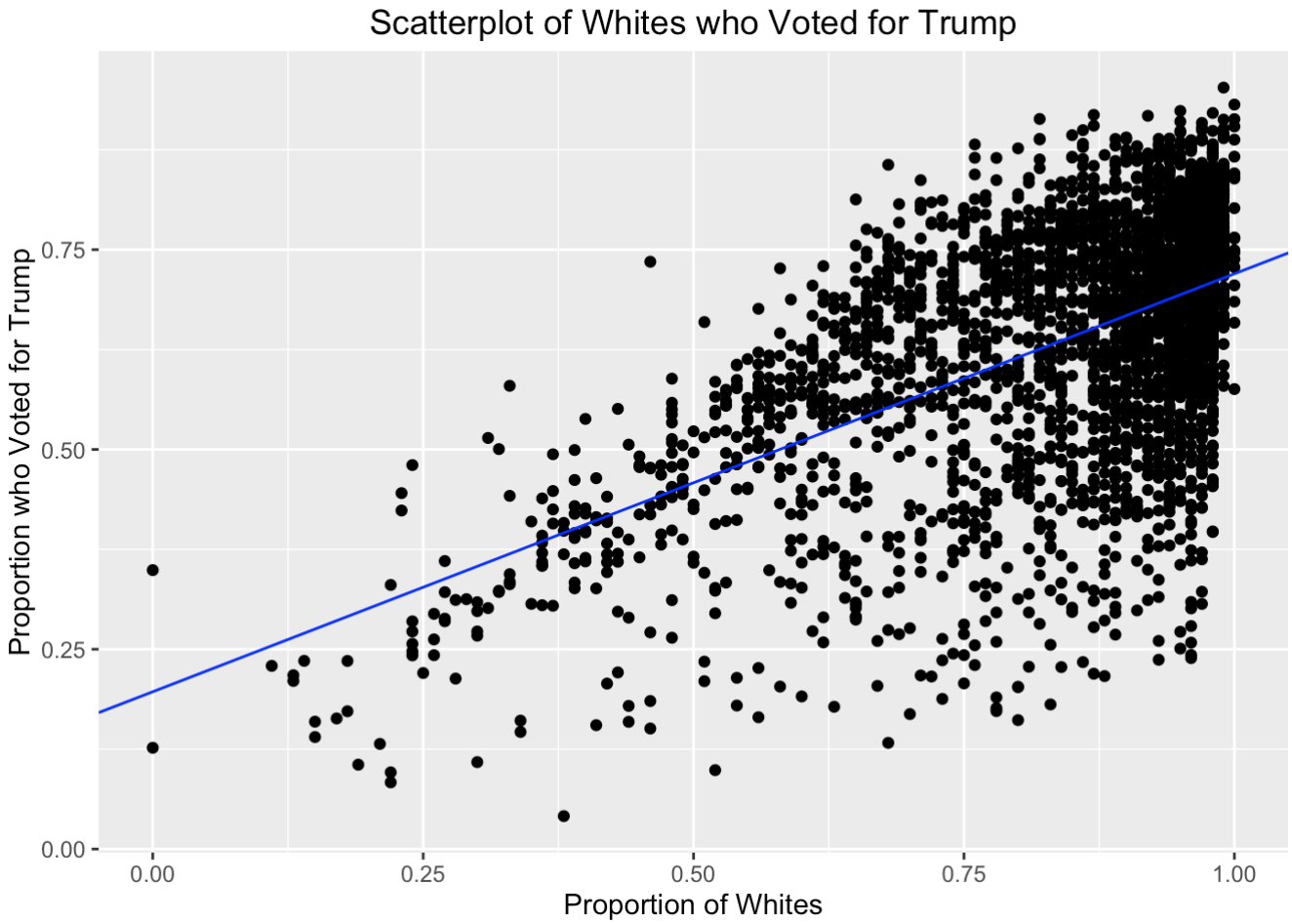
Now the middle, the yellow green and purple dots, is where the politically interesting data lies. Since, the swing states, or the states that were not in the extreme ends, were the ones that decided the election. First it's interesting to note that the green dots, which represent those swing counties that went just slightly Republican, tended to be mostly big dots. This shows that the white proportion might have had an association with deciding that small marging. Moreoever, many of these green dots are in very key states like Florida and Ohio. This is an extremely important place to analysis when seeing how the election got overturned. Also it's interesting to note that the yellow dots that represent just the counties inbetween the two extremes but not at a close margin, happen to be mostly big dots also.

There should also be a bit of attention to the way I chose the colors. The resource I used for color choosing was color brewer, and this made more visually informative colors available. Red and blue were obvious choices for Democrat and Republican. However, the middle yellow color was just from the middle range that color brewer

website gave me. The slightly republican and slightly democrat colors I chose was a little tricky because initally I wanted to choose a light color of red and a light color of blue for slightly Republican and slightly Democrat respectively. However, because of the alpha transparency value, it was extremely hard to see the difference between a normal red color and a light red color, so I had to choose a completely different color for visual sake. These colors were again selected from color brewer.

To visually see this white proportion vs. trump proportion correlation not only on a map but also on a graph, I made the following scatterplot with a regression fitted line on it.

```
coef = coef(lm(prop_trump ~ white_prop, data = final_data_frame))
ggplot() + geom_point(data = final_data_frame, aes(x = white_prop, y = prop_trump)) + la
bs(title = "Scatterplot of Whites who Voted for Trump", x = "Proportion of Whites", y =
"Proportion who Voted for Trump") + geom_abline(intercept = coef[1], slope = coef[2], co
lor = "blue")
```




And not surprisingly, there seems to be a heavy positive correlation between these two variables further accentuating the map's claims on the two extremes. It seems that it was very fortunate for Trump that America is heavily dominated (according to the statistics) by an average of around 85% whites for each county. This caused him, with his white supremacy campaign and anti-immigration laws, to make many Americans heavily favor him. Hilary, on the other hand, seems to have gotten more minority counties. This, however, was not enough for her to win as she needed to win these key counties that were very close between her and Trump.

# PART 3: DONE

# PART 4: Building predictors

# Classification tree for the 2016 election results using 2012 election results as training set.

# Getting Data

First of all, I selected the data frames for my training set and test set. The training set includes all the 40 variables chosen from the census data as well as the proportion wins for Obama and Romney. I create two individual data frame in order to predict the proportion of votes for two parties separately. Then I created my test sets using the same procedure with 2016 data for Trump and Clinton proportion votes. This test set will not be used until I use my trainset to build a model to get a good CP value to predict and then the test set will be used at the very last.

```
#finding the total number of rows in the final_data_frame created in Part 1. All the tes
t and training sets have the same number of rows as the final_data_frame
nTrain = nrow(final_data_frame)

#select 2012 results from the final_data_frame as training set, separating by parties.
trainset = data.frame(sapply(final_data_frame[c(43, 45, 3:39)], as.numeric))
rep_trainset = trainset[-2]
dem_trainset = trainset[-1]
#select 2016 results from the final_data_frame as test set, separating by parties.
testset = data.frame(sapply(final_data_frame[c(51, 49, 3:39)], as.numeric))
rep_testset = testset[-2]
dem_testset = testset[-1]
```

# Cross-validation

For cross-validation, I chose 30 different complexity parameters (cps) and pass into my 2-fold cross-validation to test for accuracy. The reason the 2-fold CV was used is because we had 3082 rows and it wasn't divisble by 3, so hence 2-fold is best.

```
#partition our data into random non-overlapping pieces, taking 2 folds since we have 308
2 rows
permuteIndices = sample(nTrain)
folds = matrix(permuteIndices, ncol = 2)
# Choose cps
library(rpart)
cps = c(seq(0.0001, 0.001, by = 0.0001),
       seq(0.001, 0.01, by = 0.001),
       seq(0.01, 0.1, by = 0.01))
```

# Predict the Democratic proportion votes in 2016 first.

By randomly splitting the training sets into two different equal-sized parts as train fold and test fold, I created a double nested for-loops to loop through different cp values and for each of them, I use 'rpart' and 'predict' to test for classification. Then, for each set of the predicted winning party by comparing the proportion of votes for dem and rep, I compare them to the actual win in 2012 from the original data to test for accuracy.

```
#creating two empty matrices to store the prediction results using the training set
dem_preds = matrix(nrow = nTrain, ncol = length(cps))
rep_preds = matrix(nrow = nTrain, ncol = length(cps))
#using double for loop to loop over the folds and the complexity parameter values to get
 our predictions
for (i in 1:2) {
  trainFold = as.integer(folds[, -i])
  testFold = folds[, i]

  for (j in 1:length(cps)) {
    dem_tree = rpart(obama_prop_2012 ~ .,
            data = dem_trainset[trainFold,],
            control = rpart.control(cp = cps[j]))
    dem_preds[testFold, j] =
      predict(dem_tree,
              newdata = dem_trainset[testFold,])
    rep_tree = rpart(romney_prop ~ .,
            data = rep_trainset[trainFold,],
            control = rpart.control(cp = cps[j]))
    rep_preds[testFold, j] =
      predict(rep_tree,
              newdata = rep_trainset[testFold,])
  }
}
#creating a new col in trainset which incicates whether the dem party actually won or no
t in 2012
actual_dem_prop_train = trainset$obama_prop_2012
actual_rep_prop_train = trainset$romney_prop
trainset$actual_dem_won = actual_dem_prop_train > actual_rep_prop_train
#creating a matrix indicating whether the dem party won or not based on our predictions
 using the training set in 2012
pred_dem_won = dem_preds > rep_preds
#for each cp, calculate the proportion of correct predictions
cvRates = apply(pred_dem_won, 2, function(oneSet) {
  sum(oneSet == trainset$actual_dem_won) / length(oneSet)
})
```

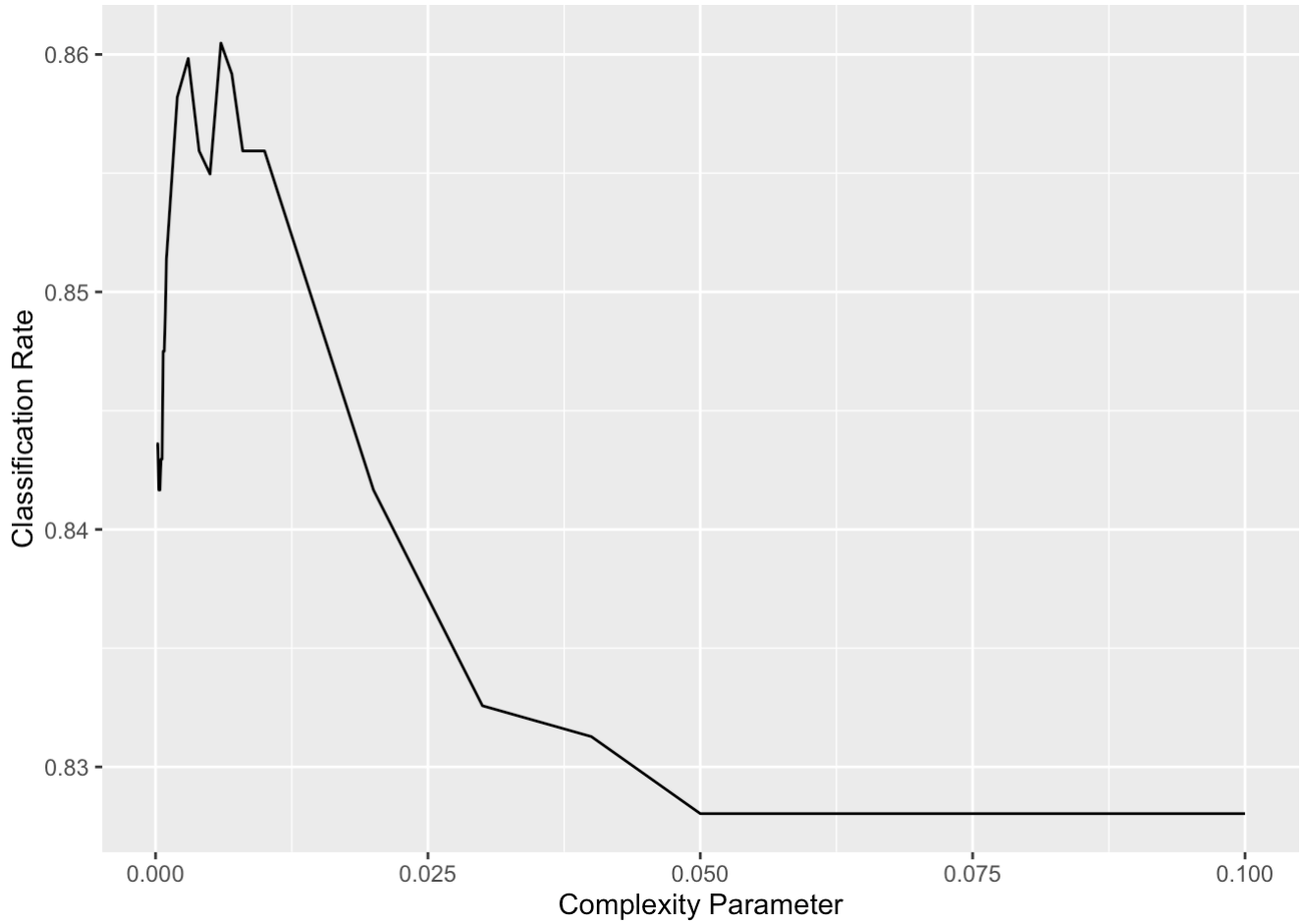
# Graphing the cp value

By plotting complexity parameters against their classification rate using a line graph, I am able to see that it peaks around 0.002 - 0.008. So I decided to choose the cp that gives the maximum classification rate for my final prediction tree.

```
library(ggplot2)
which.max(cvRates)
```

```
## [1] 16
```

```
cvRes = data.frame(cps, cvRates)
ggplot(data = cvRes, aes(x = cps, y = cvRates)) +
  geom_line() +
  labs(x = "Complexity Parameter", y = "Classification Rate")
```




# Getting the Classification Rate

After I chose the cp and created my final precision trees, I passed in the 2016 election results aka my test sets to predict the proportion wins for both parties separately. Again, by comparing the proportions and determining the winning party, I compare them to the actual win in 2016.

```
#choosing the cp that give sthe highest classification rate
cpChoice = cvRes$cps[which.max(cvRates)]
#using the cp that give sthe highest classification rate to do the final prediction usin
g the entire training set, then use the final prediction tree to predict the testset
dem_finalTree = rpart(obama_prop_2012~ .,
                  data = dem_trainset,
                  control = rpart.control(cp = cpChoice))

dem_testPreds = predict(dem_finalTree,
              newdata = dem_testset)

rep_finalTree = rpart(romney_prop~ .,
                  data = rep_trainset,
                  control = rpart.control(cp = cpChoice))

rep_testPreds = predict(rep_finalTree,
              newdata = rep_testset)
#creating a new col in testset which incicates whether the dem party actually won or not
 in 2016
actual_dem_prop_test = testset$prop_clinton
actual_rep_prop_test = testset$prop_trump
testset$actual_dem_won = actual_dem_prop_test > actual_rep_prop_test
#creating a matrix indicating whether the dem party won or not based on our predictions
 using the testset in 2016
final_pred_dem_won = dem_testPreds > rep_testPreds
#accessing the classfication rate of our predition on the testset
classRate = sum(final_pred_dem_won == testset$ac) / nrow(testset)
classRate
```

```
## [1] 0.9279689
```

```
#The classification rate is really high! That means our predictor is very accurate!
```

Surprising, the classification rate is quite high (>90%). The classification result can be shown more clearly using the classification_result data frame I created at the end which puts the predicted results side-by-side with the actual results to see to the difference. This was probably because of the variables that were chosen in the census data to predict the results. Most of the variables have high relevance and correlation with the election results. That is not too surprising given the fact the variables chosen in the census data were the following: employment, income, education level, US citizenship, language spoken, and demographic variables. These are all indications of whether one goes democrat or republican.

This final code chunk just combines the whole result of the classification tree built predicator into one data frame, with the missclassification rates included in the data frame column.

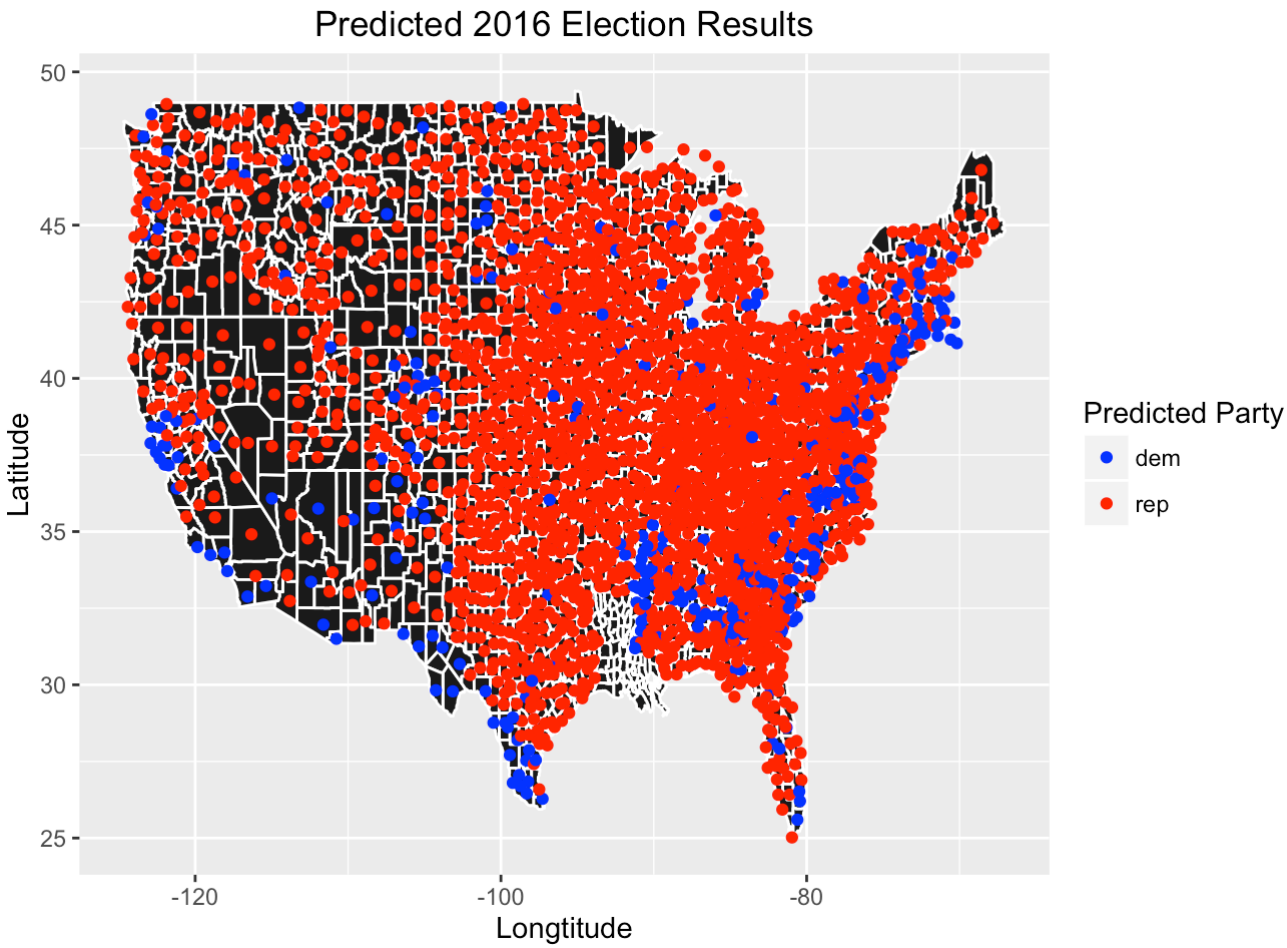
```
# combining our results into one table
classification_result = data.frame(state = final_data_frame$state, county = final_data_f
rame$county, actual_dem_prop_2012 = final_data_frame$obama_prop_2012, actual_rep_prop_20
12 = final_data_frame$romney_prop, predicted_dem_prop_2016 = unlist(dem_testPreds), actu
al_dem_prop_2016 = final_data_frame$prop_clinton, predicted_rep_prop_2016 = unlist(rep_t
estPreds), actual_dem_prop_2016 = final_data_frame$prop_trump, predicted_win_2016 = unli
st(final_pred_dem_won), actual_win_2016 = testset$actual_dem_won)
classification_result$misclassified =
  !(classification_result$actual_win_2016 == classification_result$predicted_win_2016)
#changing TRUE or FALSE to which party won in each county
classification_result$predicted_win_2016[classification_result$predicted_win_2016] = 'de
m'
classification_result$predicted_win_2016[classification_result$predicted_win_2016 == FAL
SE] = 'rep'
classification_result$actual_win_2016[classification_result$actual_win_2016] = 'dem'
classification_result$actual_win_2016[classification_result$actual_win_2016 == FALSE] =
'rep'
#this classfication rate should match the one in the previous code chunk, to make sure t
he result is still correct.
classificationRate = sum(classification_result$actual_win_2016 ==
classification_result$predicted_win_2016) / nrow(classification_result)
```

# End of 2016 Classification Tree Predicator

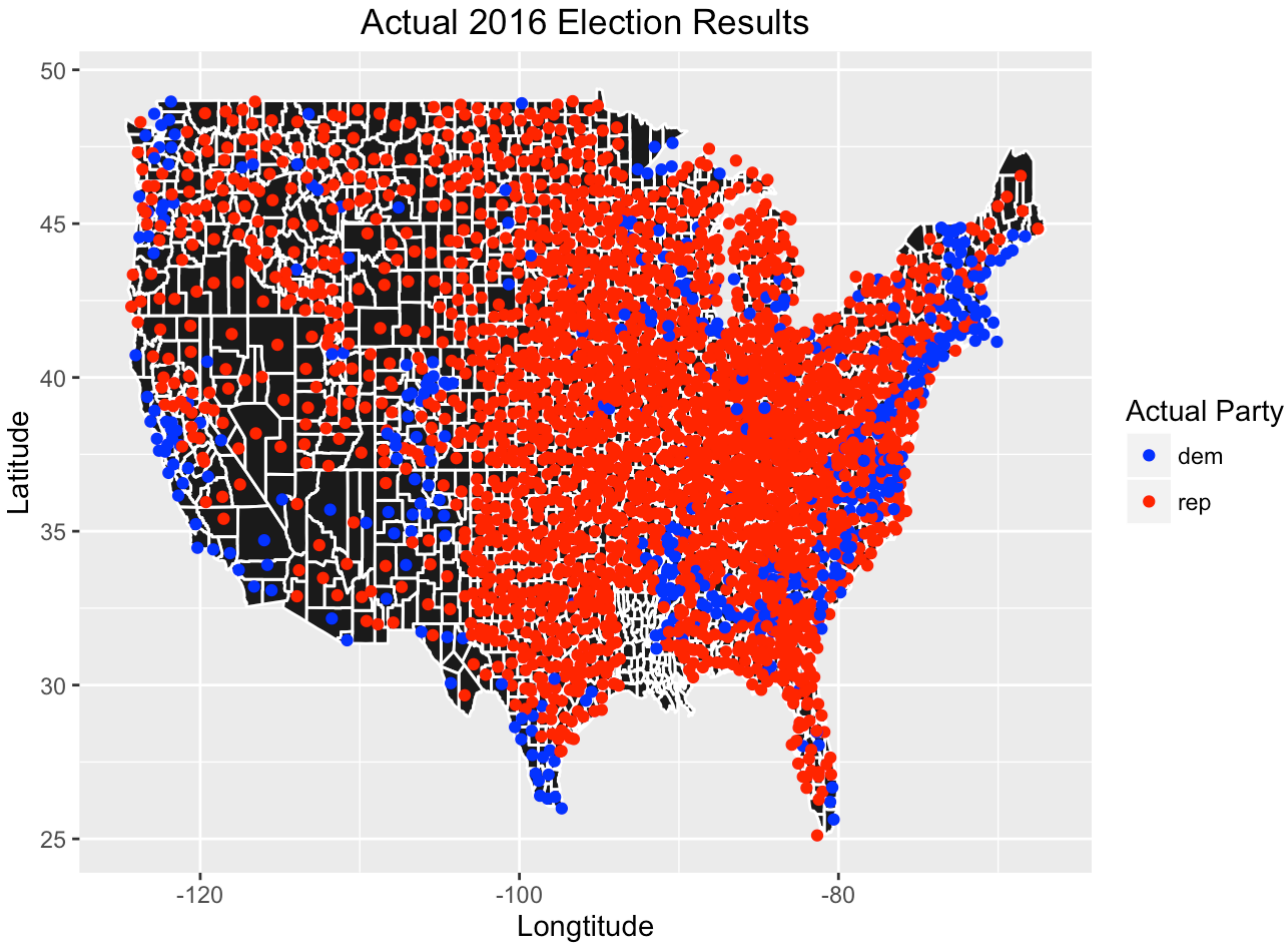
# Analysis of the 2016 Predicator

There were a total of three exploratory maps created. The first one is the predicted 2016 election results from our census variables. This creates a map showing which counties go republican or democrat, and each of the dots are colored accordingly. Then, to compare it with the actual results, we created an actual results map showing which counties actually went democratic or republican. Lastly, we created a map showing which counties had a misclassification, these dots were colored orange.

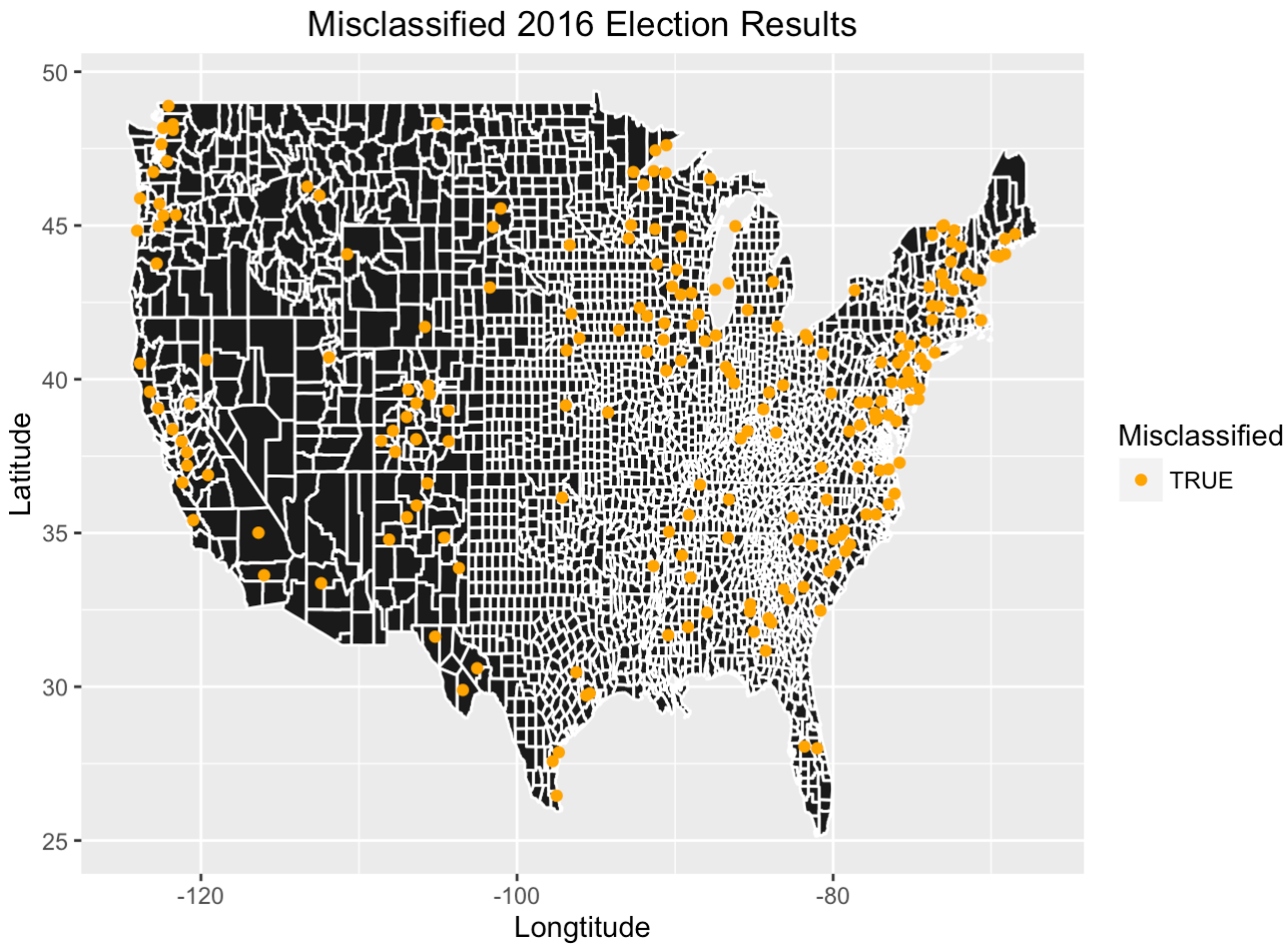
```
require(maps)
require(ggplot2)
all_county = map_data("county")

a = merge(merged_census_df, lat_long_df, by.x = c("state", "county"), by.y = c("state",
"county"))
a$county = tolower(gsub(" County", "", a$county))
classification2016_df = merge(a, classification_result, by.x = c("state", "county"), by.
y = c("state", "county"))

county_map = ggplot() + geom_polygon(data=all_county, aes(x=long, y=lat, group = group),
 colour="white", fill="grey10")

predicted2016_map =
  county_map +
  geom_jitter(data = classification2016_df, position=position_jitter(width=0.5, height=
0.5), aes(x = county_long, y = county_lat, color = predicted_win_2016 )) +
  scale_colour_manual(name = "Predicted Party", values = c("blue", "red")) +
  labs(title = "Predicted 2016 Election Results", x = "Longtitude", y = "Latitude")

predicted2016_map
```



```
actual2016_map =
  county_map +
  geom_jitter(data = classification2016_df, position=position_jitter(width=0.5, height=
0.5), aes(x = county_long, y = county_lat, color = actual_win_2016 )) +
  scale_colour_manual(name = "Actual Party", values = c("blue", "red")) +
  labs(title = "Actual 2016 Election Results", x = "Longtitude", y = "Latitude")

actual2016_map
```




```
#Getting the misclassified map
classification2016_misc = classification2016_df[classification2016_df$misclassified,]
mispredicted_map = county_map +
  geom_jitter(data = classification2016_misc, position=position_jitter(width=0.5, height
=0.5), aes(x = county_long, y = county_lat, color = misclassified)) +
  scale_colour_manual(name = "Misclassified", values = "orange") +
  labs(title = "Misclassified 2016 Election Results", x = "Longtitude", y = "Latitude")
mispredicted_map
```




Overall, the predictor we made worked very well. The predicted results of 2016 Election and the actual results are very similar. Both graphs show that the majority of the counties in our country voted for the republican party. The predictor was accurate in most the swing states, such as Florida, Pennsylvania, Michigan and Wisconsin, as these states all have majority red dots in the predicted graph. The predicator being accurate in these swing states are key because after all the prediactor should be predicating the ones that are unknown not the ones that we all know are going to go Democratic or Republican. The similarity between the predicted graph and the actual graph proves our high classification rate, which is above 90%.

By looking at the map where only contains the points where counties were misclassified. We see that once again, we have most of our mistakes in states like Gerogia, Iowa, Wisconsin, New York, Minnesota, California, and Michigan. These are the places where our prediction might not have worked very well. Realistically speaking it would be extremely weird to see our predicator predict almost every county correctly because this year's election had huge overturns and unexpected results. Many counties and even states went Republican when they were actually Democratic states. This definitly plays a part in our missclasification rate. However, it's worthy to note that there are only a few handful of orange dots in an over 3000 county map. #Analysis of 2016 Predictor Done

# K-Nearest-Neighbors on the Change From 2012 to 2016

```
# keep proportion data to train/test
drop_names = c("kerryVote", "other_count", "bushVote", "romney_count", "trump_count", "c
linton_count", "obama_count_2012")
full_census_knn = final_data_frame[ , !(names(final_data_frame) %in% drop_names)]

#we drop the vote proportions so that we can focus on the other features
drop = c("mccain_prop_2008", "romney_prop", "obama_prop_2008", "obama_prop_2012", "prop_
trump", "prop_clinton", "state", "county", "bush_prop", "ketty_prop")
dimensions = names(full_census_knn)[!(names(full_census_knn) %in% drop)]
```

```
#this block creates the training/test data by classifying changes in party loyalty for 2
004-2008, 2008-2012, 2012-2016
# D->D, D->R, R->D, R->R
census_knn = full_census_knn[,dimensions]
win_2004 = full_census_knn$kerry_prop - full_census_knn$bush_prop
win_2008 = full_census_knn$obama_prop_2008 - full_census_knn$mccain_prop_2008
win_2012 = full_census_knn$obama_prop_2012 - full_census_knn$romney_prop
win_2016 = full_census_knn$prop_clinton - full_census_knn$prop_trump
#1 = DD, 2 = RD, 3 = DR, 4 = RR
DD08 = as.numeric(win_2004 > 0 & win_2008 > 0)
DR08 = as.numeric(win_2004 > 0 & win_2008 < 0)*2
RD08 = as.numeric(win_2004 < 0 & win_2008 > 0)*3
RR08 = as.numeric(win_2004 < 0 & win_2008 < 0)*4
train_2008 = DD08+DR08+RD08+RR08

DD12 = as.numeric(win_2008 > 0 & win_2012 > 0)
DR12 = as.numeric(win_2008 > 0 & win_2012 < 0)*2
RD12 = as.numeric(win_2008 < 0 & win_2012 > 0)*3
RR12 = as.numeric(win_2008 < 0 & win_2012 < 0)*4
train_2012 = DD12+DR12+RD12+RR12

DD16 = as.numeric(win_2012 > 0 & win_2016 > 0)
DR16 = as.numeric(win_2012 > 0 & win_2016 < 0)*2
RD16 = as.numeric(win_2012 < 0 & win_2016 > 0)*3
RR16 = as.numeric(win_2012 < 0 & win_2016 < 0)*4
test_2016 = DD16+DR16+RD16+RR16
```

```
#testing the accuracies of knn for various values of k, using 2004-08 data, 2008-12 dat
a, and both in one dataframe
library(class)
full_train = rbind(census_knn, census_knn)
accuracy_08 = c(20)
accuracy_12 = c(20)
accuracy_08_12 = c(20)

for (i in 1:20) {
  accuracy_08_12[i] = sum(knn(full_train, census_knn, c(train_2008,train_2012), k = i) =
= test_2016)/length(test_2016)
  accuracy_08[i] = sum(knn(census_knn, census_knn, train_2008, k = i) == test_2016)/leng
th(test_2016)
  accuracy_12[i] = sum(knn(census_knn, census_knn, train_2012, k = i) == test_2016)/leng
th(test_2016)
}
accuracies = data.frame(k = 1:20, accuracy_08 = accuracy_08, accuracy_12 = accuracy_12,
accuracy_08_12 = accuracy_08_12)
accuracies
```

```
##     k accuracy_08 accuracy_12 accuracy_08_12
## 1   1   0.5869565   0.8659961      0.7229072
## 2   2   0.5678131   0.8338741      0.7232317
## 3   3   0.5944192   0.8595068      0.7774173
## 4   4   0.6012330   0.8601557      0.7725503
## 5   5   0.6099935   0.8695652      0.7942894
## 6   6   0.6038287   0.8640493      0.7852044
## 7   7   0.6148605   0.8653472      0.8011032
## 8   8   0.6161583   0.8650227      0.8072680
## 9   9   0.6129137   0.8663206      0.8205711
## 10 10   0.6083712   0.8669695      0.8179753
## 11 11   0.6073978   0.8656716      0.8192732
## 12 12   0.6093446   0.8624270      0.8225178
## 13 13   0.6048021   0.8630759      0.8277093
## 14 14   0.6086957   0.8656716      0.8228423
## 15 15   0.6090201   0.8634004      0.8267359
## 16 16   0.6109669   0.8624270      0.8273848
## 17 17   0.6093446   0.8614536      0.8299805
## 18 18   0.6103180   0.8608047      0.8299805
## 19 19   0.6073978   0.8614536      0.8345230
## 20 20   0.6096690   0.8569111      0.8364698
```

# More analysis

```
#identify which counties switched from D-R from 2008-12
length(which(train_2012 == 2))
```

```
## [1] 188
```

```
#identify how many of these counties stayed R in 2016
sum(test_2016[which(train_2012 == 2)]==4)/length(which(train_2012 == 2)) #~95% stayed re
publican
```

```
## [1] 0.9414894
```

```
#identify which counties switched from R-D from 2008-12
length(which(train_2012 == 3))
```

```
## [1] 10
```

```
#identify how many of these counties switched back to R
sum(test_2016[which(train_2012 == 3)]==1) #100% switched back
```

```
## [1] 0
```

```
# table of count of counties in states that switched R-D and switched back
ana3 = aggregate(county ~ state, full_census_knn[which(train_2012 == 3 & test_2016 == 2)
,c(1, 2)], length)
ana3 <- ana3[order(ana3$county, decreasing = TRUE),]
ana3    #spread all over
```

```
##            state county
## 1        alabama      2
## 6       new york      2
## 2       colorado      1
## 3        georgia      1
## 4       kentucky      1
## 5    mississippi      1
## 7 north carolina      1
## 8 south carolina      1
```

```
#identify which counties was consistently DD from 2008-12
length(which(train_2012 == 1))
```

```
## [1] 658
```

```
#from these, which were D in 2016
sum(test_2016[which(train_2012 == 1)]==2)/length(which(train_2012 == 1)) #~33% that were
 D in 2008-2012 flipped to R in 2016
```

```
## [1] 0.3100304
```

```
ana1 = aggregate(county ~ state, full_census_knn[which(train_2012 == 1 & test_2016 == 2)
,c(1, 2)], length)
ana1 <- ana1[order(ana1$county, decreasing = TRUE),]
ana1    #iowa, wisconsin, minnesota had the most counties that switched to R in 2016 aft
er DD
```

```
##             state county
## 10           iowa     31
## 35      wisconsin     21
## 14      minnesota     19
## 21       new york     16
## 13       michigan     12
## 8        illinois     11
## 24           ohio      9
## 12          maine      8
## 33       virginia      7
## 22 north carolina      6
## 7         georgia      5
## 28 south carolina      5
## 29   south dakota      5
## 34     washington      5
## 6          florida      4
## 9          indiana      4
## 23   north dakota      4
## 3         colorado      3
## 15    mississippi      3
## 16         montana      3
## 18  new hampshire      3
## 20     new mexico      3
## 26   pennsylvania      3
## 19     new jersey      2
## 25         oregon      2
## 1         arkansas      1
## 2       california      1
## 4      connecticut      1
## 5         delaware      1
## 11        kentucky      1
## 17        nebraska      1
## 27   rhode island      1
## 30      tennessee      1
## 31          texas      1
## 32        vermont      1
```

```
#identify which counties was consistently RR from 2008-12
length(which(train_2012 == 4))
```

```
## [1] 2221
```

```
#from these, which were D in 2016
sum(test_2016[which(train_2012 == 4)]==4) #most stayed republican
```

```
## [1] 2213
```

# Analysis of change in voter party

The most accurate predictor overall of a county's change in party loyalty from 2012 to 2016 was the county's change in party loyalty from 2008 to 2012, using 5 nearest neighbors. This K value was chosen by looking at all the accuracies from the accuracy table for several chosen K values and choosing the best accuracte value.

All of the counties that switched from voting Republican -> Democrat from 2008-12 switched back to voting Republican in the 2016 election. There were only 10 counties for which this was the case, and they were spread across 8 states.

While none of the counties that were Republican were classified as switching to Democrat, there were in actuality 19 counties that switched party loyalty from Republican to Democrat. These, however, were spread across the country, not confined to a single geographic area.

~40% of the counties that were predicted to stay Democratic in 2016 voted for the Republican candidate in the 2016 election. Many of these counties were located in midwestern states (the states with the most counties for which this was the case were Iowa, Winsonsin, and Minnesota)

Meanwhile, less than 8% of the counties that were consistently Republican in 2008 and 2012 elections voted for the Democratic candidate in 2016 This shows an overall trend toward voting Republican in 2016. Republican party loyalty in the past 3 elections have been very strong, while there were many counties that had voted Democrat in the last two elections that voted Republican for the 2016 election.

# End of Part 4

# Discussion

We learned through this project many machine learning tools and honestly a lot of geo-political information about the United States. It was absolutely necessary to know a bit of borderline cases when doing merges such as how some counties were written in different ways (like Brooklyn vs Kings). Moreover, the data merging part really taught us how much time and tedious work data scraping is. The reason is because in the world sources come from various places and standardization is rare. Obviously the project taught us some of the impacts and driving force of the 2016 election such as white proportions really affecting the vote.

We also learned that winning candidate can win the election with less than winning 1/3 of counties. The number of counties are disproportionally scattered across the US. Stated in the middle area of the US have lots of counties with very small populations. Also, by predicting the proportion of winning at county level, I learned how to apply the concepts of cross-validation as well as classification trees using complex real-world data. Through the process of data cleaning and merging, I realized that as a data scientist or a statistician, how important it is to have a well-formed, clean, easily workable data and how we spent almost half of our time simply putting data together with the minimum loss of information.

One of the more illuminating aspects of this project was that while the vast majority of the counties that switched from voting Democrat to Republican from 2008-12 continued to vote Republican in 2016, none of the counties that switched from Republican to Democrat from 2008-12 continued to vote Democrat. It also seemed that counties that had historically been Republican had more party loyalty than those counties that were more Democratic. While effectively all the counties that voted Republican in 2008 and 2012 continued to do so in 2016,

about 33% of the counties that voted Democratic in 2008 and 2012 switched to voting for the Republican candidate in 2016. In all, the process of working from the ground-up on this project exposed a lot of the issues that arise from working with a data set that wasn't clean, and the many different ways in which a single data set could be interpreted.

The last thing to mention in our discussion is how powerful visualization is. No amount of code can speak a similar story that some of our maps did. The visualization, espicially the specific ones done in part 3, really showed a powerful message if done properly and with good care. We realized how much a plot can show and really how important it is to pay attention to these small details like alpha values for transperency to tell a better story. After all, there's a huge market demand for people with good skills in producing good visualizing for a reason.

# References

Professor Nolan on getting the Virginia counties from Wikipedia.

Virginia counties wikipedia link:
"https://en.wikipedia.org/wiki/United_States_presidential_election_in_Virginia,_2004 (https://en.wikipedia.org/wiki/United_States_presidential_election_in_Virginia,_2004)"

Sources extracted from: 2016 election data: "https://github.com/tonmcg/County_Level_Election_Results_12-16/blob/master/2016_US_County_Level_Presidential_Results.csv" 2012 election data: "http://www.politico.com/2012-election/map/#/President/2012/ (http://www.politico.com/2012-election/map/#/President/2012/)" 2008 election data: "https://www.theguardian.com/news/datablog/2009/mar/02/us-elections-2008 (https://www.theguardian.com/news/datablog/2009/mar/02/us-elections-2008)" 2004 election data: "http://www.stat.berkeley.edu/users/nolan/data/voteProject/countyVotes2004.txt (http://www.stat.berkeley.edu/users/nolan/data/voteProject/countyVotes2004.txt)" 2010 census data: "http://factfinder2.census.gov/faces/nav/jsf/pages/searchresults.xhtml?refresh=t (http://factfinder2.census.gov/faces/nav/jsf/pages/searchresults.xhtml?refresh=t)" latitude and longtitude data: "http://www.stat.berkeley.edu/users/nolan/data/voteProject/counties.gml (http://www.stat.berkeley.edu/users/nolan/data/voteProject/counties.gml)"

All package used are citied within the library, these include: readr, XML, xlsx, RCurl, Rpart, class

http://colorbrewer2.org/ (http://colorbrewer2.org/)