2016 Presidential Election Debrief Report

Tinsley Zhu, Nicholas Kang, Daniel Ma, Anton Oyung, William Tang

December 8, 2016

```
setwd("~/classes/sophomore/stats 133/project/election/submissions")
```

Introduction

The 2016 Election was one for the ages. Almost every major national poll in the nation listed Hilary Clinton to be the next President of the United States, but in a stunning turn of events that marked one of the greatest political upsets since John Quincy Adams back in 1824, Donald J. Trump became the president elect. As UC Berkeley students enrolled in a statistics course analyzing data, it is only natural that our response is to analyze why this occurred and to make fabulous graphs, maps, and sharp use of ggplots. Buckle-up for a statistical adventure where we will describe data, make maps about them, and then talk about them even more.

Data Description

```
load("final.rda")
```

Our data frame contains the results of the 2004, 2008, 2012, and 2016 General Presidential Elections and a multitude of variables from 2010 census data. These variables include: -Total population -White population -Black population -Place of birth -Citizenship status -Educational attainment -Language Spoken at Home -Percentage of households with a given value for income and benefits Before we proceed to our analysis, we'll add some columns that will indicate whether a county voted Democrat or Republican. We'll also check to see how much missing data we have within each column.

```
final$Winner2004 = factor(final$`Kerry (Democrat) 2004 Votes` > final$`Bush (Republ
ican) 2004 Votes`,
                          levels = c(TRUE, FALSE), labels = c("D", "R"))
final$Winner2008 = factor(final$`Obama (Democrat) 2008 Votes` > final$`McCain (Rep
ublican) 2008 Votes`,
                          levels = c(TRUE, FALSE), labels = c("D", "R"))
final$Winner2012 = factor(final$`Obama (Democrat) 2012 Votes` > final$`Romney (Rep
ublican) 2012 Votes`,
                           levels = c(TRUE, FALSE), labels = c("D", "R"))
final$Winner2016 = factor(final$`Percent Hillary 2016` > final$`Percent Trump 2016`
,
                    levels = c(TRUE, FALSE), labels = c("D", "R"))
#Rearrange columns
final = final[, c(1:5, 48, 6:7, 49, 8:9, 50, 10:14, 51, 15:47)]
```

Analyzing the 2012 Election Results

With the following plots, we wanted to explore the following variables: 1. The percentage of White and Black voters. 2. How unemployment related to voting preferences. 3. How income and age influenced voting

preferences.

First, we prepared our final dataframe to create the variables that would be needed for this EDA:

Next, we organized the counties into two categories: those that had the majority vote for Obama versus those that had the majority vote for Romney.

```
# Creating data frames for counties where there is a majority of Obama voters vs Ro
mney voters.
obama_majority = final[final$`Obama (Democrat) 2012 Votes` > final$`Romney (Republi
can) 2012 Votes`, ]

romney_majority = final[final$`Obama (Democrat) 2012 Votes` < final$`Romney (Republ
ican) 2012 Votes`, ]

# Verifying that Obama won the popular vote in 2012.
sum(final$`Obama (Democrat) 2012 Votes`, na.rm = TRUE) >
  sum(final$`Romney (Republican) 2012 Votes`, na.rm = TRUE)
```

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

Looking at Percentages of White and Black Voters

Following this, the first plot we created analyzed the average percentage of White voters vs African American voters in each of the counties:

```
require(ggplot2)
```

```
## Loading required package: ggplot2
```

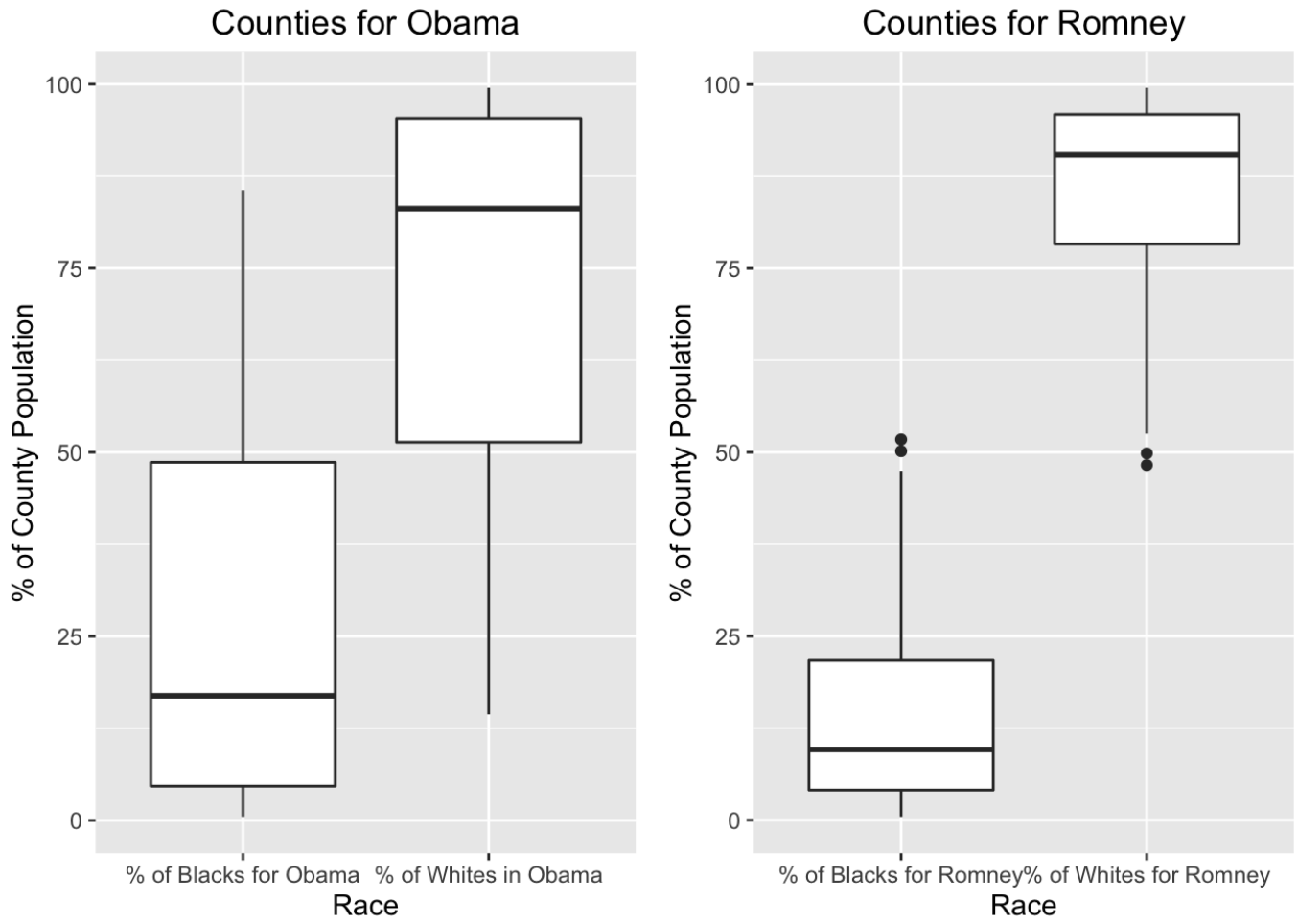
```
require(gridExtra)
```

```
## Loading required package: gridExtra
```

```
obama_race_percent = ggplot(data = obama_majority) +
  geom_boxplot(mapping = aes(x = "% of Whites in Obama", y = `% White`)) +
  geom_boxplot(mapping = aes(x = "% of Blacks for Obama", y = `% Black`)) +
  scale_y_continuous(name = "% of County Population") +
  scale_x_discrete(name = "Race") +
  ggtitle("Counties for Obama")

romney_race_percent = ggplot(data = romney_majority) +
  geom_boxplot(mapping = aes(x = "% of Whites for Romney", y = `% White`)) +
  geom_boxplot(mapping = aes(x = "% of Blacks for Romney", y = `% Black`)) +
  scale_y_continuous(name = "% of County Population") +
  scale_x_discrete(name = "Race") +
  ggtitle("Counties for Romney")

grid.arrange(obama_race_percent, romney_race_percent, ncol = 2)
```

```
## Warning: Removed 208 rows containing non-finite values (stat_boxplot).

## Warning: Removed 208 rows containing non-finite values (stat_boxplot).
```

```
## Warning: Removed 1336 rows containing non-finite values (stat_boxplot).

## Warning: Removed 1336 rows containing non-finite values (stat_boxplot).
```

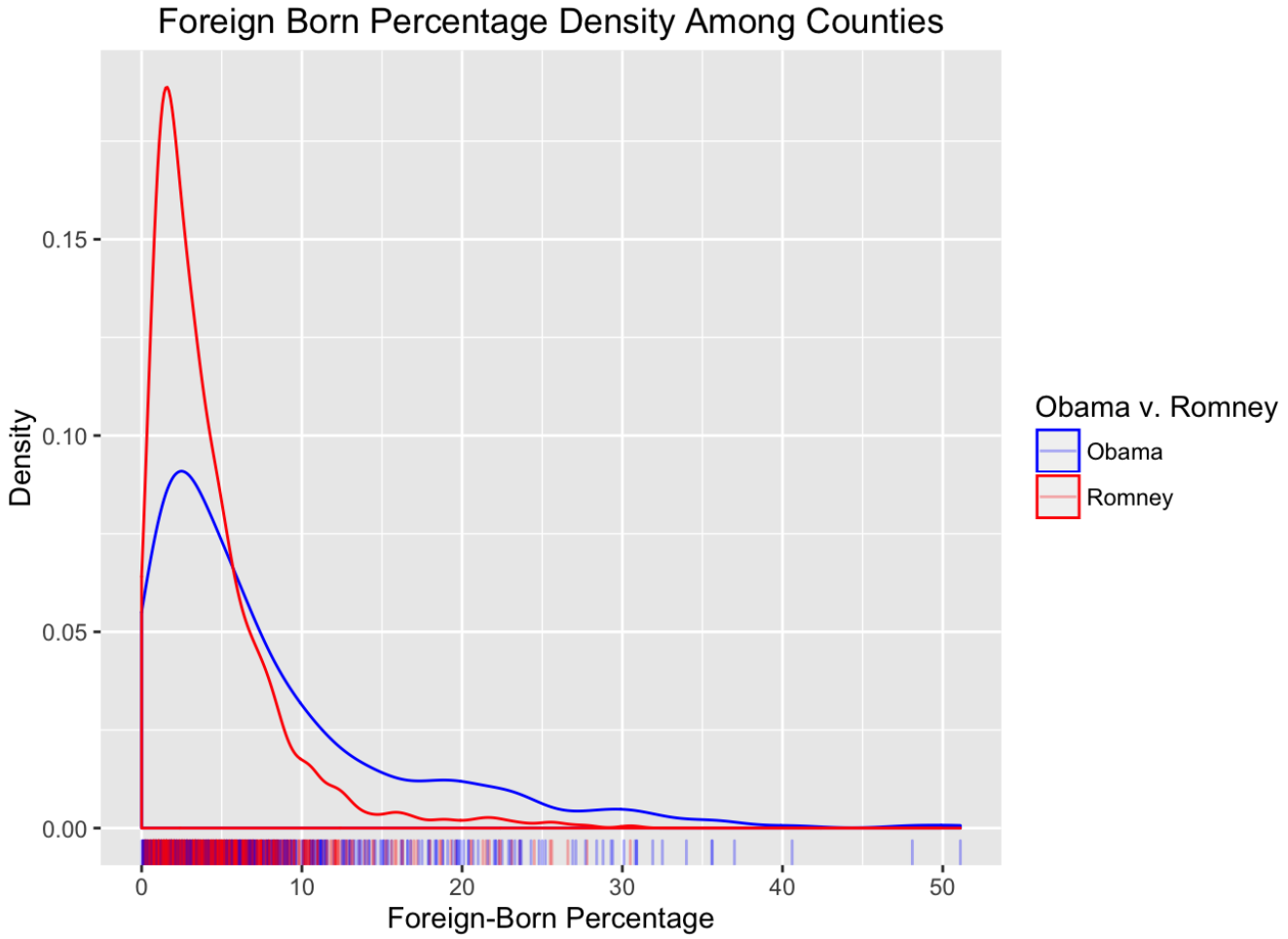


What we can see from this is that, for the counties that had a majority of voters side with Obama, there was on average a larger percentage of African Americans. Conversely, those counties that had a majority of Romney supporters tended to have a much higher percentage of White voters. Interestingly, the boxplots show that the spread for Obama-supporting counties is much higher, indicating that Obama as a candidate may have appealed to a more diversified range of counties.

Looking at Foreigner Status and Voting Preferences

Following this, we then created a density plot that looked at the percentage of unemployment in each county, and how that correlated to whether or not the county voted for Obama or Romney.

```
obama_romney_unemployment = ggplot(data = na.omit(final), mapping = aes(x = `Perce
nt; PLACE OF BIRTH - Foreign born`, color = `Obama v. Romney`)) +
  geom_density() +
  geom_rug(alpha = 0.3) +
  ggtitle("Foreign Born Percentage Density Among Counties") +
  scale_x_continuous(name = "Foreign-Born Percentage") +
  scale_y_continuous(name = "Density") +
  scale_color_manual(values = c("blue", "red"),
                     name = "Obama v. Romney",
                     labels = c("Obama", "Romney"))

obama_romney_unemployment
```

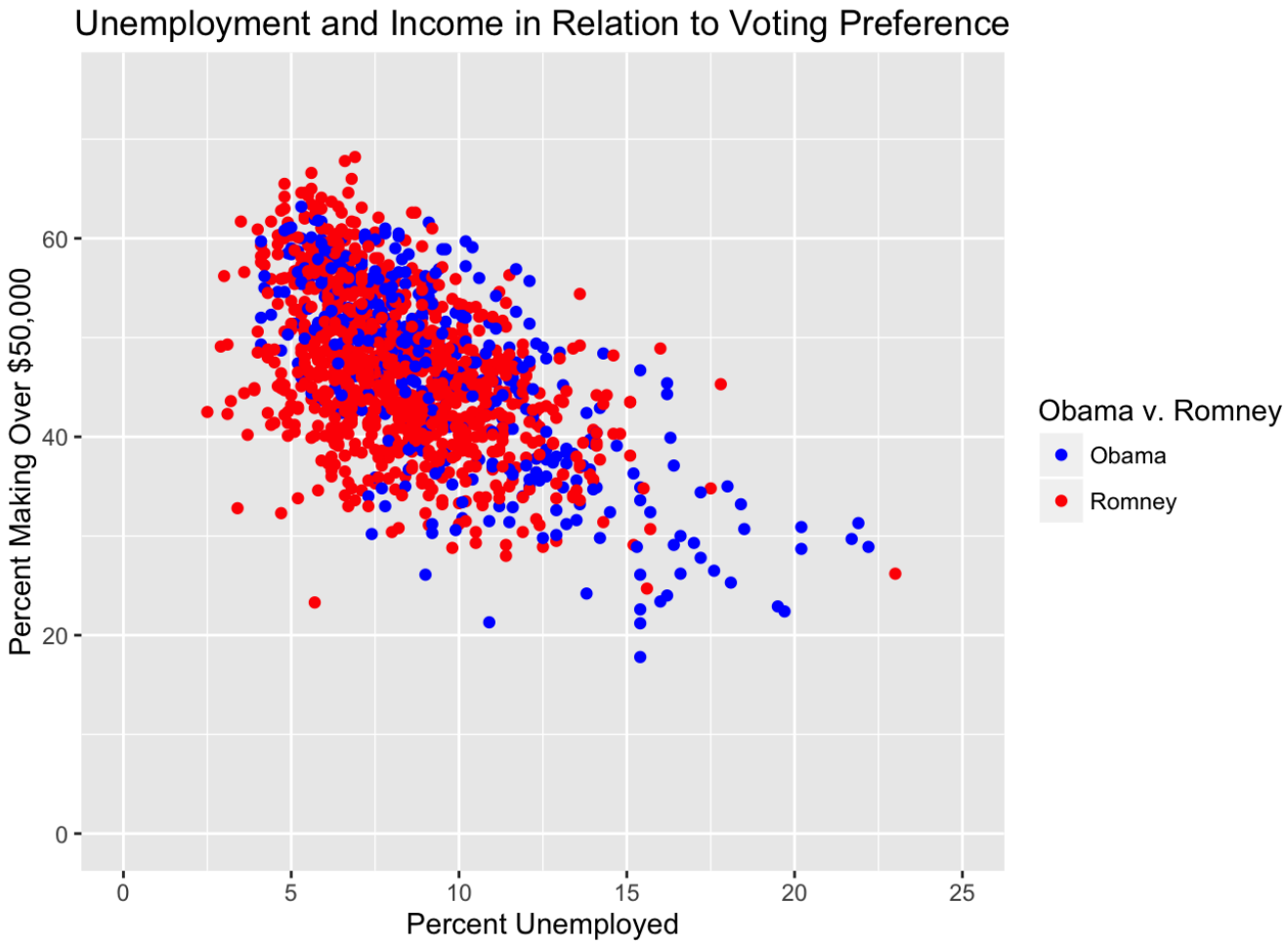


Interestingly, what seems to be shown here is that counties that voted for Romney tended to generally have much lower levels of foreign born citizens, while Obama tended to bring in higher numbers of votes from foreign-born residents.

Considering Income and Employment in Relation to Voting Preferences

Finally, we can consider an analysis of the relationship between the general age and employment percentage within each county, and whether there's any correlation to how those counties voted in the 2012 election:

```
ggplot(data = na.omit(final), mapping = aes(x = `Percent; EMPLOYMENT STATUS - Perc
ent Unemployed`, y = `% Over $50,000`, color = `Obama v. Romney`)) +
  geom_point() +
  scale_x_continuous(name = "Percent Unemployed", limits = c(0, 25)) +
  scale_y_continuous(name = "Percent Making Over $50,000", limits = c(0, 75)) +
  ggtitle("Unemployment and Income in Relation to Voting Preference") +
  scale_color_manual(values = c("blue", "red"),
                     name = "Obama v. Romney",
                     labels = c("Obama", "Romney"))
```



Although the correlation isn't exceptionally strong, we see that the counties who voted for Romney typically tend to be clustered around lower amounts of unemployment and higher percentages of income (over $50,000). Obama, on the other hand, has a larger spread of counties he appealed to, including many amongst the less financially well-off counties with higher unemployment.

NOTE: On this plot, the x and y-axis are not at an equal scale. Originally, the scale was set to 0 to 100% for both the x and y-axis, but this shrunk the points to a small cluster that aesthetically was very difficult to see. Hence, we zoomed in, but the general trend observed still holds regardless of how "close up" or "far out" one is.

Map/Data Visualization

Now, we will construct a map to help visualize the general voting patterns and trends that occurred in the most recent 2016 election.

We first begin with basic setup of a map:

```
library(mapdata)
```

```
## Loading required package: maps
```

```
library(ggplot2)
library(maps)

# Creating a column to indicate if that county had a Clinton or Trump majority.
final$`Clinton v. Trump` =
  final$`Clinton (Democrat) 2016 Votes` < final$`Trump (Republican) 2016 Votes`
final$`Clinton v. Trump` = factor(final$`Clinton v. Trump`, levels = c(TRUE, FALSE)
, labels = c('Trump', 'Clinton'))

# Load U.S. map data
all_counties = map_data("county")

# Plot which counties favored which candidate.
map_plot = ggplot() +
  geom_polygon(data=all_counties, aes(x=long, y=lat, group = group), colour="light
gray", fill="white" ) + scale_x_continuous(name = "Longitude") + scale_y_continuous
(name = "Latitude") + ggtitle("Basic Map of U.S. Counties")
```

Mapping Which Counties Favored Which Candidate

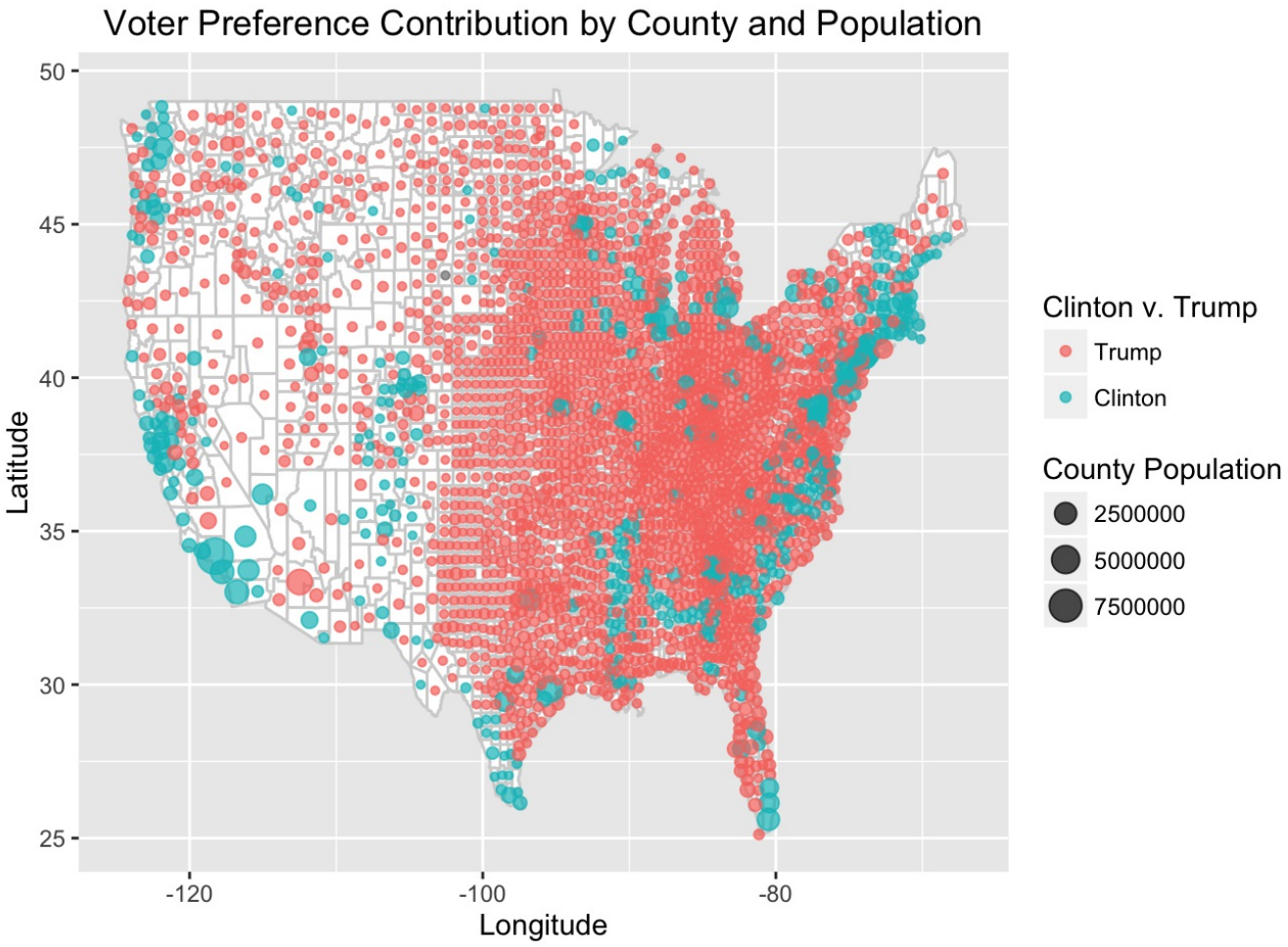
Here, we plot points on each county. The color of the dot indicates which candidate (Clinton or Trump) won that county, while the size of the dot indicates how large the population was in that county.

```
map_plot +
  geom_polygon(data=all_counties, aes(x=long, y=lat, group = group), colour="light
gray", fill="white" ) +
  geom_point(data = final, aes(x = Longitude, y = Latitude, color = `Clinton v. Tru
mp`, size = `Total Pop`), alpha = 0.7) +
  scale_x_continuous(name = "Longitude") +
  scale_y_continuous(name = "Latitude") +
  ggtitle("Voter Preference Contribution by County and Population") +
  labs(size = "County Population")
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which
## will replace the existing scale.
```

```
## Scale for 'y' is already present. Adding another scale for 'y', which
## will replace the existing scale.
```

```
## Warning: Removed 8 rows containing missing values (geom_point).
```



Predicting the 2016 Results by KNN

We'll focus on building our predictor of the 2016 Presidential Election Results based on chosen data from the 2010 Census data. To begin building our predictor, we'll first split the data into a test set and a training set. Before we do so, we'll drop all the columns that are not from our census data except for Winner2016, and then we'll check to see how many NAs each variable in our data frame contains. We may remove rows that contain many NA values. We'll also drop GEO.id and GEO.id2 since those columns only serve to identify the unique county and will not help us in predicting whether a county voted Democrat or Republican.

```
#Drop columns that we don't want
finalUpdated = final[, c(-1:-17, -21, -22, -54:-58)]

#Check for number of NAs in each column
sapply(finalUpdated, function(x) sum(is.na(x)))
```

```
##
Winner2016
##
4
##
Longitude
##
5
##
Latitude
##
5
```

```
##
Total Pop
##
7
##
White Pop
##
10
##
Black Pop
##
1536
##                                Percent; HOUSEHOLDS BY TYPE - Family households (
families)
##
7
##                                     Percent; HOUSEHOLDS BY TYPE - Nonfamily h
ouseholds
##
7
##                             Percent; EDUCATIONAL ATTAINMENT - Population 25 years
and over
##
7
##                                   Percent; EDUCATIONAL ATTAINMENT - Less than
9th grade
##
7
##                            Percent; EDUCATIONAL ATTAINMENT - 9th to 12th grade, n
o diploma
##
7
##            Percent; EDUCATIONAL ATTAINMENT - High school graduate (includes equ
ivalency)
##
7
##                                 Percent; EDUCATIONAL ATTAINMENT - Some college,
no degree
##
7
##                                      Percent; EDUCATIONAL ATTAINMENT - Associate
's degree
##
7
##                                       Percent; EDUCATIONAL ATTAINMENT - Bachelor
's degree
##
7
##                         Percent; EDUCATIONAL ATTAINMENT - Graduate or profession
al degree
##
7
##                                                          Percent; PLACE OF BIRT
H - Native
##
7
##                                                   Percent; PLACE OF BIRTH - For
```

```
eign born
##
7
##                                Percent; U.S. CITIZENSHIP STATUS - Naturalized U.S
. citizen
##
25
##                                     Percent; U.S. CITIZENSHIP STATUS - Not a U.S
. citizen
##
25
##                                           Percent; LANGUAGE SPOKEN AT HOME - Eng
lish only
##
7
##                               Percent; LANGUAGE SPOKEN AT HOME - Language other tha
n English
##
7
##                                               Percent; EMPLOYMENT STATUS - In la
bor force
##
7
##                                           Percent; EMPLOYMENT STATUS - Percent U
nemployed
##
7
##    Percent; INCOME AND BENEFITS (IN 2010 INFLATION-ADJUSTED DOLLARS) - Less than
$10,000
##
7
##   Percent; INCOME AND BENEFITS (IN 2010 INFLATION-ADJUSTED DOLLARS) - $10,000 to
$14,999
##
7
##   Percent; INCOME AND BENEFITS (IN 2010 INFLATION-ADJUSTED DOLLARS) - $15,000 to
$24,999
##
7
##   Percent; INCOME AND BENEFITS (IN 2010 INFLATION-ADJUSTED DOLLARS) - $25,000 to
$34,999
##
7
##   Percent; INCOME AND BENEFITS (IN 2010 INFLATION-ADJUSTED DOLLARS) - $35,000 to
$49,999
##
7
##   Percent; INCOME AND BENEFITS (IN 2010 INFLATION-ADJUSTED DOLLARS) - $50,000 to
$74,999
##
7
##   Percent; INCOME AND BENEFITS (IN 2010 INFLATION-ADJUSTED DOLLARS) - $75,000 to
$99,999
##
7
## Percent; INCOME AND BENEFITS (IN 2010 INFLATION-ADJUSTED DOLLARS) - $100,000 to
$149,999
```

```
##
7
##
% White
##
1537
##
% Black
##
1537
```

We'll drop the Black Pop, %White, and %Black columns since they have too many NAs. In addition, we note that most of the variables from the 2010 census data in the data frame contain 7 NAs and some columns have at most 10 NAs. We'll drop the rows that contain these NAs since we cannot build a predictor with NA values; dropping these values will not affect the success of our predictor significantly.

```
#White Pop has most NAs after dropping Black Pop column; lon and lat has 1 NA value
after White pop NAs are eliminated
#Also drop %White and % Black due to large number of NAs
finalUpdated = finalUpdated[!is.na(finalUpdated$`White Pop`) & !is.na(finalUpdated
$Longitude), c(-6, -33, -34)]

#Check number of NA values again
#Mostly 0 NA values in columns from census data
#except for Citizenship Status which still contains 17 NAs
sapply(finalUpdated, function(x) sum(is.na(x)))
```

```
##
Winner2016
##
0
##
Longitude
##
0
##
Latitude
##
0
##
Total Pop
##
0
##
White Pop
##
0
##                                Percent; HOUSEHOLDS BY TYPE - Family households (
families)
##
0
##                                        Percent; HOUSEHOLDS BY TYPE - Nonfamily h
ouseholds
##
0
```

```
##                            Percent; EDUCATIONAL ATTAINMENT - Population 25 years
and over
##
0
##                                    Percent; EDUCATIONAL ATTAINMENT - Less than
9th grade
##
0
##                           Percent; EDUCATIONAL ATTAINMENT - 9th to 12th grade, n
o diploma
##
0
##             Percent; EDUCATIONAL ATTAINMENT - High school graduate (includes equ
ivalency)
##
0
##                                 Percent; EDUCATIONAL ATTAINMENT - Some college,
no degree
##
0
##                                      Percent; EDUCATIONAL ATTAINMENT - Associate
's degree
##
0
##                                       Percent; EDUCATIONAL ATTAINMENT - Bachelor
's degree
##
0
##                         Percent; EDUCATIONAL ATTAINMENT - Graduate or profession
al degree
##
0
##                                                          Percent; PLACE OF BIRT
H - Native
##
0
##                                                   Percent; PLACE OF BIRTH - For
eign born
##
0
##                               Percent; U.S. CITIZENSHIP STATUS - Naturalized U.S
. citizen
##
17
##                                      Percent; U.S. CITIZENSHIP STATUS - Not a U.S
. citizen
##
17
##                                           Percent; LANGUAGE SPOKEN AT HOME - Eng
lish only
##
0
##                              Percent; LANGUAGE SPOKEN AT HOME - Language other tha
n English
##
0
##                                               Percent; EMPLOYMENT STATUS - In la
```

```
bor force
##
0
##                                          Percent; EMPLOYMENT STATUS - Percent U
nemployed
##
0
##    Percent; INCOME AND BENEFITS (IN 2010 INFLATION-ADJUSTED DOLLARS) - Less than
$10,000
##
0
##   Percent; INCOME AND BENEFITS (IN 2010 INFLATION-ADJUSTED DOLLARS) - $10,000 to
$14,999
##
0
##   Percent; INCOME AND BENEFITS (IN 2010 INFLATION-ADJUSTED DOLLARS) - $15,000 to
$24,999
##
0
##   Percent; INCOME AND BENEFITS (IN 2010 INFLATION-ADJUSTED DOLLARS) - $25,000 to
$34,999
##
0
##   Percent; INCOME AND BENEFITS (IN 2010 INFLATION-ADJUSTED DOLLARS) - $35,000 to
$49,999
##
0
##   Percent; INCOME AND BENEFITS (IN 2010 INFLATION-ADJUSTED DOLLARS) - $50,000 to
$74,999
##
0
##   Percent; INCOME AND BENEFITS (IN 2010 INFLATION-ADJUSTED DOLLARS) - $75,000 to
$99,999
##
0
## Percent; INCOME AND BENEFITS (IN 2010 INFLATION-ADJUSTED DOLLARS) - $100,000 to
$149,999
##
0
```

```
#Check percent of foreign born for each row that has NAs for Citizenship Status
finalUpdated[is.na(finalUpdated$`Percent; U.S. CITIZENSHIP STATUS - Not a U.S. citi
zen`), c(17:19)]
```

```
##      Percent; PLACE OF BIRTH - Foreign born
## 115                                       0
## 204                                       0
## 211                                       0
## 538                                       0
## 552                                       0
## 861                                       0
## 1349                                      0
## 1373                                      0
## 1485                                      0
## 1689                                      0
## 1756                                      0
## 1874                                      0
## 2196                                      0
## 2219                                      0
## 2560                                      0
## 2596                                      0
## 3008                                      0
##      Percent; U.S. CITIZENSHIP STATUS - Naturalized U.S. citizen
## 115                                                          NA
## 204                                                          NA
## 211                                                          NA
## 538                                                          NA
## 552                                                          NA
## 861                                                          NA
## 1349                                                         NA
## 1373                                                         NA
## 1485                                                         NA
## 1689                                                         NA
## 1756                                                         NA
## 1874                                                         NA
## 2196                                                         NA
## 2219                                                         NA
## 2560                                                         NA
## 2596                                                         NA
## 3008                                                         NA
##      Percent; U.S. CITIZENSHIP STATUS - Not a U.S. citizen
## 115                                                    NA
## 204                                                    NA
## 211                                                    NA
## 538                                                    NA
## 552                                                    NA
## 861                                                    NA
## 1349                                                   NA
## 1373                                                   NA
## 1485                                                   NA
## 1689                                                   NA
## 1756                                                   NA
## 1874                                                   NA
## 2196                                                   NA
## 2219                                                   NA
## 2560                                                   NA
## 2596                                                   NA
## 3008                                                   NA
```

Since these counties contain no populations that are foreign born, we can set each of the U.S. Citizenship

Status values to 0.

```
#Change Citizenship NAs to 0
finalUpdated[is.na(finalUpdated$`Percent; U.S. CITIZENSHIP STATUS - Not a U.S. citi
zen`), c(18:19)] = 0

#Check if the change was correct
#TRUE
all(sapply(finalUpdated, function(x) sum(is.na(x))) == 0)
```

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

```
#Change all columns except Winner2016 to numerics
finalUpdated[, -1] = sapply(finalUpdated[,-1], as.numeric)
```

Now we'll split the data into a test set and a training set. We'll set our test data to have a size of 701, leaving us with 2400 observations to train our predictor using 3-fold cross validation.

```
set.seed(9125021)
testSize = 701
testIndices = sample(testSize)
testSet = finalUpdated[testIndices, ]
testSetActual = testSet[, 1]
trainSet = finalUpdated[-testIndices, ]
trainSetActual = trainSet[, 1]

set.seed(5278143)
foldNum = 3
folds = matrix(sample(nrow(finalUpdated) - testSize), ncol = foldNum)
```

Now we can begin creating our predictor using 3-fold cross validation on k-Nearest Neighbor Classification. We will need to import the class package. First we'll first determine what k value to use for our predictor. For each fold, we run the knn algorithm.

```
kVals = c(1:20)
predsK = matrix(nrow = nrow(trainSet), ncol = length(kVals))
require(class)
```

```
## Loading required package: class
```

```
for(i in seq(foldNum)){
  trainFold = trainSet[as.integer(folds[, -i]),]
  trainFoldActual = trainFold[, 1]
  testFold = trainSet[folds[, i], ]
  testFoldActual = testFold[, 1]
  for(j in kVals) {
    predsK[folds[, i], j] = knn(train = trainFold[, -1], test = testFold[, -1], cl
= trainFoldActual, k = j)
  }
}
```

We calculate the proportion of correct predictions for each value of k and determine the k with the highest

success rate

```
#Change preds to factor
predsK = apply(predsK, 2, factor, levels = c(1, 2), labels = c("D", "R"))

#Calculate success rate
kCorrectRates = apply(predsK, 2, function(x) {
  sum(x == trainSet[, 1]) / nrow(trainSet)
})

data.frame(kVals, kCorrectRates)
```

```
##    kVals kCorrectRates
## 1      1     0.8525000
## 2      2     0.8570833
## 3      3     0.8895833
## 4      4     0.8870833
## 5      5     0.8954167
## 6      6     0.8945833
## 7      7     0.8962500
## 8      8     0.8941667
## 9      9     0.8920833
## 10    10     0.8916667
## 11    11     0.8908333
## 12    12     0.8916667
## 13    13     0.8891667
## 14    14     0.8879167
## 15    15     0.8875000
## 16    16     0.8866667
## 17    17     0.8850000
## 18    18     0.8837500
## 19    19     0.8825000
## 20    20     0.8804167
```

```
#Return 0.89625
max(kCorrectRates)
```

```
## [1] 0.89625
```

```
#Return k=7
k = max(which(kCorrectRates == max(kCorrectRates)))
print(paste("Best value of k is ", k, ".", sep =""))
```

```
## [1] "Best value of k is 7."
```

```
#Function for success rates for future use
success = function(preds, actual) {
  if (!is.factor(preds)) {
    preds = apply(preds, 2, factor, levels = c(1, 2), labels = c("D", "R"))
  }
  if(is.matrix(preds)) {
    return(apply(preds, 2, function(x) sum(x == actual[, 1]) / nrow(actual)))
  }
  return(sum(preds == actual[, 1]) / nrow(actual))
}
```

Let's see how well our predictor does with our testSet

```
#knn predictor on testSet
testPreds = knn(train = trainSet[, -1], test = testSet[, -1], cl = trainSet[, 1],
k = k)

#Determine success rate: 0.9044223
sum(testPreds == testSet[, 1]) / nrow(testSet)
```

```
## [1] 0.9044223
```

Now we'll adjust our predictor by determining which variables will provide us with a higher success rate and using only those variables in our predictor. To do this, for each variable v, we exclude the columns corresponding to v from our data frames and run our predictor before calculating the success rates. Lower success rates with variable j excluded will suggest that variable j is important to include in our final predictor.

```
#Column/variable 1 is the actual classification, so we subtract ncol(trainSet) by 1
varNum = seq(ncol(trainSet) -1)
predsVars = matrix(nrow = nrow(trainSet), ncol = length(varNum))

#For each variable/column j, perform knn with column j excluded
for (i in seq(foldNum)) {
  trainFold = trainSet[as.integer(folds[, -i]),]
  trainFoldActual = trainFold[, 1]
  testFold = trainSet[folds[, i], ]
  testFoldActual = testFold[, 1]
  #Because we don't want to include column 1, column needed from the folds is j + 1
  for (j in varNum) {
    var = j+1
    predsVars[folds[, i], j] = knn(train = trainFold[, c(-1, -var)], test = testFo
ld[, c(-1, -var)] , cl = trainFoldActual, k = k)
  }
}
varCorrectRates = success(predsVars, trainSet)
varCorrectRates
```

```
##  [1] 0.8962500 0.8962500 0.8920833 0.8612500 0.8962500 0.8962500 0.8991667
##  [8] 0.8962500 0.8962500 0.8962500 0.8962500 0.8962500 0.8962500 0.8962500
## [15] 0.8962500 0.8962500 0.8962500 0.8962500 0.8962500 0.8962500 0.8962500
## [22] 0.8962500 0.8962500 0.8962500 0.8962500 0.8962500 0.8962500 0.8962500
## [29] 0.8962500 0.8962500
```

```
which(varCorrectRates == max(varCorrectRates))
```

```
## [1] 7
```

The above contains our success rates from our aforementioned testing method. If we look at the success rates for each variable that was excluded, we see that the success rate seems to jump when the 7th variable is excluded, which corresponds to the 8th column in our finalUpdated data frame. We update data set to not include the 7th census variable before proceeding.

```
trainSet = trainSet[, -8]
testSet = testSet[, -8]
finalUpdated = finalUpdated[, -8]
```

we'll now continue to determine which variables we should include in our predictor. We'll run through the previous code again, this time with the 6th variable excluded.

```
#Column/variable 1 is the actual classification, so we subtract ncol(trainSet) by 1
varNum = seq(ncol(trainSet) -1)
predsVars = matrix(nrow = nrow(trainSet), ncol = length(varNum))

#For each variable/column j, perform knn with column j excluded
for (i in seq(foldNum)) {
  trainFold = trainSet[as.integer(folds[, -i]),]
  trainFoldActual = trainFold[, 1]
  testFold = trainSet[folds[, i], ]
  testFoldActual = testFold[, 1]
  #Because we don't want to include column 1, column needed from the folds is j + 1
  for (j in varNum) {
    var = j+1
    predsVars[folds[, i], j] = knn(train = trainFold[, c(-1, -var)],
                                   test = testFold[, c(-1, -var)],
                                   cl = trainFoldActual, k = k)
  }
}
varCorrectRates = success(predsVars, trainSet)
#Order variable numbers by increasing success rate i.e. by decreasing importance of
the variable
data.frame(varNumber = order(varCorrectRates) + 1, varCorrectRates = varCorrectRate
s[order(varCorrectRates)])
```

```
##    varNumber varCorrectRates
## 1          4       0.8541667
## 2          5       0.8620833
## 3         17       0.8987500
## 4         18       0.8987500
## 5          2       0.8991667
## 6          3       0.8991667
## 7          6       0.8991667
## 8          7       0.8991667
## 9          8       0.8991667
## 10         9       0.8991667
## 11        10       0.8991667
## 12        11       0.8991667
## 13        12       0.8991667
## 14        13       0.8991667
## 15        14       0.8991667
## 16        15       0.8991667
## 17        16       0.8991667
## 18        19       0.8991667
## 19        20       0.8991667
## 20        21       0.8991667
## 21        22       0.8991667
## 22        23       0.8991667
## 23        24       0.8991667
## 24        25       0.8991667
## 25        26       0.8991667
## 26        27       0.8991667
## 27        28       0.8991667
## 28        29       0.8991667
## 29        30       0.8991667
```

The above is a list of variable numbers (i.e. column indices of trainSet and testSet) in the order of importance to include in our predictor. We see that the 4th, 5th, and 17th columns in the trainSet and testSet are respectively the most, second most, and third most important variables to include in our predictor. We'll now determine how many variables we should exclude. To do this, starting with a training set that has the first two of the "most important" variables, we'll run the predictor after each time we add another variable (by decreasing importance) and determine the success rates each time we add another variable.

```
varImportanceOrder = order(varCorrectRates) + 1
predsInclude = matrix(nrow = nrow(trainSet), ncol = length(varImportanceOrder) - 2
)
for (i in seq(foldNum)) {
  trainFold = trainSet[as.integer(folds[, -i]),]
  trainFoldActual = trainFold[, 1]
  testFold = trainSet[folds[, i], ]
  testFoldActual = testFold[, 1]

  #Start with first two important variables in train set
  for (j in seq(length(varImportanceOrder) - 2)) {
    varPos = j + 2
    predsInclude[folds[, i], j] = knn(train = trainFold[, varImportanceOrder[seq(1
:varPos)]],
                                      test = testFold[, varImportanceOrder[seq(1:v
arPos)]],
                                      cl = trainFoldActual)
  }
}
includeCR = success(predsInclude, trainSet)
data.frame(`Number of Variables` = seq(3, length(varImportanceOrder)), `Success Rat
e` = includeCR)
```

```
##    Number.of.Variables Success.Rate
## 1                    3    0.8504167
## 2                    4    0.8500000
## 3                    5    0.8504167
## 4                    6    0.8504167
## 5                    7    0.8508333
## 6                    8    0.8508333
## 7                    9    0.8508333
## 8                   10    0.8508333
## 9                   11    0.8512500
## 10                  12    0.8512500
## 11                  13    0.8512500
## 12                  14    0.8512500
## 13                  15    0.8512500
## 14                  16    0.8512500
## 15                  17    0.8512500
## 16                  18    0.8512500
## 17                  19    0.8512500
## 18                  20    0.8512500
## 19                  21    0.8512500
## 20                  22    0.8512500
## 21                  23    0.8512500
## 22                  24    0.8512500
## 23                  25    0.8512500
## 24                  26    0.8512500
## 25                  27    0.8512500
## 26                  28    0.8512500
## 27                  29    0.8512500
```

According to the data frame the more variables we include, the more successful our predictor is. So we'll include all the variables in our predictor. We test our predictor on the test data and determine who would win the popular vote out of the counties in our test set.

```
predsOnTest = knn(train = trainSet[, -1], test = testSet[, -1], cl = trainSet[, 1]
, k=k)
print(paste("Success rate is ", success(predsOnTest, testSet) * 100, "%.", sep = "
"))
```

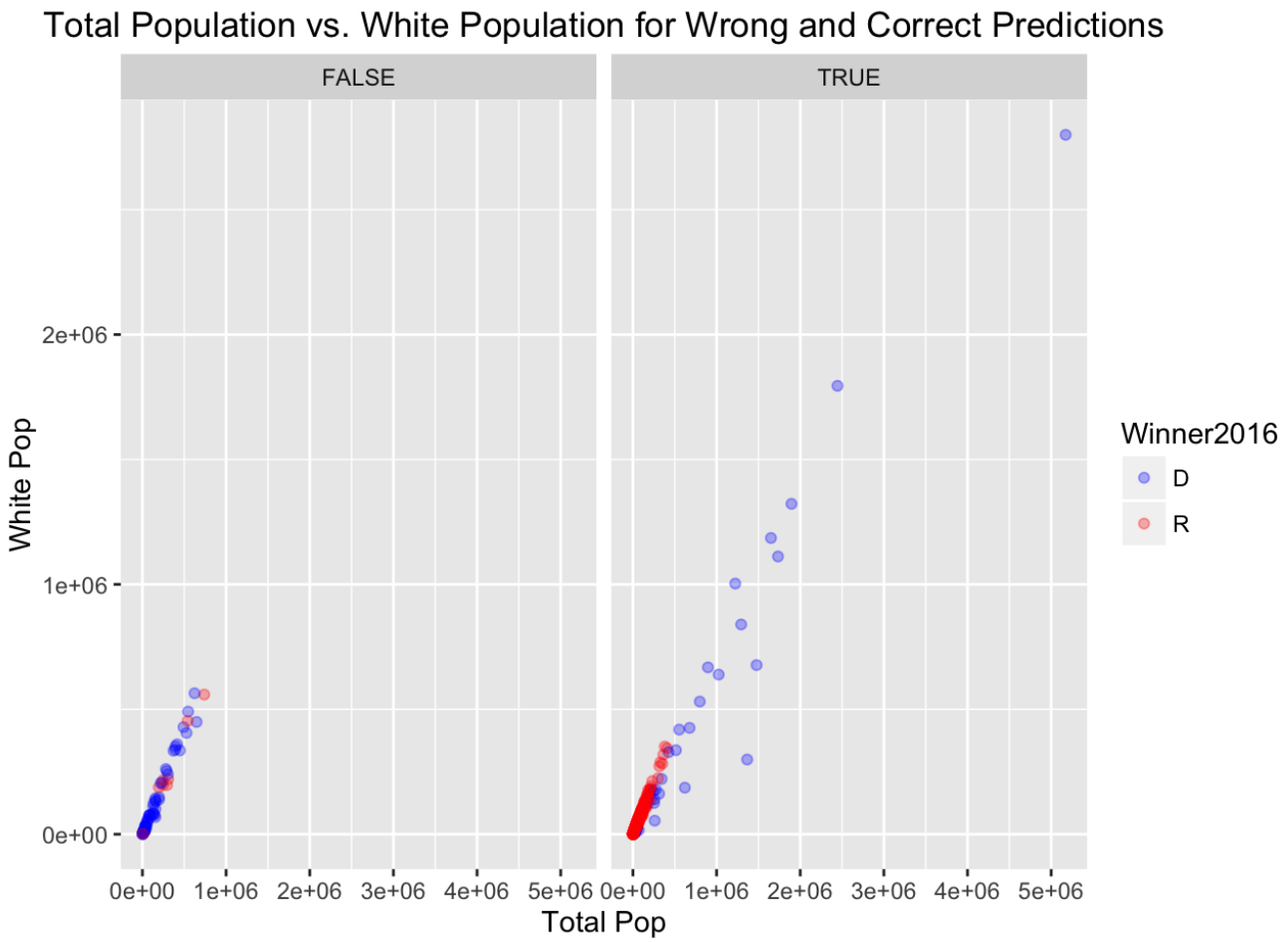
```
## [1] "Success rate is 90.0142653352354%."
```

```
#Check who wins most counties out of testSet
demPop = sum(testSet[testSet$Winner2016 == "D", "Total Pop"])
repPop = sum(testSet[testSet$Winner2016 == "R", "Total Pop"])
if(demPop > repPop) {
  print("2016 Predictor: Hillary Wins Popular Vote (Among Test Set)")
} else {
  print("2016 Predictor: Trump Wins Popular Vote (Among Test Set)")
}
```

```
## [1] "2016 Predictor: Hillary Wins Popular Vote (Among Test Set)"
```

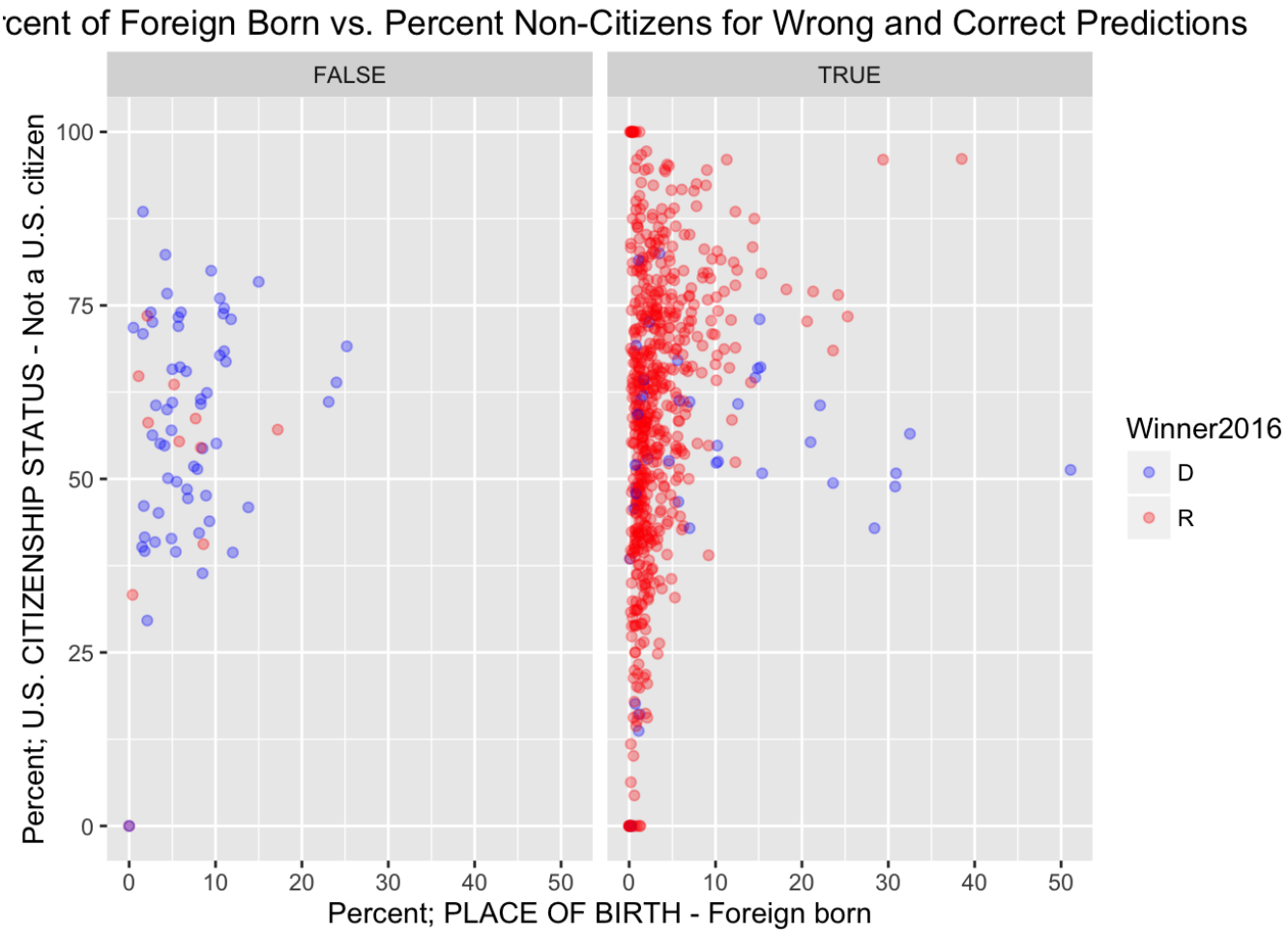
We now examine where the predictor went wrong. To do this, we'll create a copy of the testSet data frame with a logical vector that is TRUE if our prediction was correct and FALSE otherwise and continue our analysis on this data frame, particularly on the rows where our predictor was wrong. Note that in the following graphs, the color represents the actual majority party of the county. We'll first examine the relationship between total county populations and county white populations among counties where the predictor was wrong and where the predictor was right.

```
testCompare = cbind(correctPrediction = predsOnTest == testSet[, 1], testSet)

ggplot(data= testCompare, mapping = aes(x = `Total Pop`, y = `White Pop`, color =
Winner2016)) +
  geom_point(alpha = 0.3) +
  facet_grid(. ~correctPrediction) +
  scale_color_manual(values = c("blue", "red")) +
  ggtitle("Total Population vs. White Population for Wrong and Correct Predictions"
)
```



We notice that some smaller counties with large white populations were mostly predicted as Republican when they actually voted Democrat.

```
ggplot(data = testCompare, mapping = aes(x = `Percent; PLACE OF BIRTH - Foreign bor
n`,
                                         y = `Percent; U.S. CITIZENSHIP STATUS - N
ot a U.S. citizen`,
                                         color = Winner2016)) +
  facet_grid(. ~ correctPrediction) +
  geom_point(alpha = 0.3)+
  scale_color_manual(values = c("blue", "red")) +
  ggtitle("Percent of Foreign Born vs. Percent Non-Citizens for Wrong and Correct P
redictions")
```



We notice that there are counties with low percentages of foreign born populations that were predicted to be Republican when they voted Democrat.

Predicting 2016 Results by Linear Regression

```
changeovertimeDF = final[ ,c("County", "State", "State Abbreviation", "Kerry (Democrat) 2004 Votes", "Bush (Republican) 2004 Votes", "Obama (Democrat) 2008 Votes", "McCain (Republican) 2008 Votes", "Obama (Democrat) 2012 Votes", "Romney (Republican) 2012 Votes", "Clinton (Democrat) 2016 Votes", "Trump (Republican) 2016 Votes")]
```

Create a column of total votes per county for each election (# dem votes + # rep votes).

```
changeovertimeDF$totalvotes2004 = changeovertimeDF$`Kerry (Democrat) 2004 Votes` + changeovertimeDF$`Bush (Republican) 2004 Votes`
changeovertimeDF$totalvotes2008 = changeovertimeDF$`Obama (Democrat) 2008 Votes` + changeovertimeDF$`McCain (Republican) 2008 Votes`
changeovertimeDF$totalvotes2012 = changeovertimeDF$`Obama (Democrat) 2012 Votes` + changeovertimeDF$`Romney (Republican) 2012 Votes`
changeovertimeDF$totalvotes2016 = changeovertimeDF$`Clinton (Democrat) 2016 Votes` + changeovertimeDF$`Trump (Republican) 2016 Votes`
```

Create a column of percentage of democratic voters per county for each election (# dem votes / # total votes).

```
changeovertimeDF$percentDem2004 = changeovertimeDF$`Kerry (Democrat) 2004 Votes` /
changeovertimeDF$totalvotes2004
changeovertimeDF$percentDem2008 = changeovertimeDF$`Obama (Democrat) 2008 Votes` /
changeovertimeDF$totalvotes2008
changeovertimeDF$percentDem2012 = changeovertimeDF$`Obama (Democrat) 2012 Votes` /
changeovertimeDF$totalvotes2012
changeovertimeDF$percentDem2016 = changeovertimeDF$`Clinton (Democrat) 2016 Votes`
/
changeovertimeDF$totalvotes2016
```

Calculate percent change (+/-) of democratic voters per county for each election (1 - (% dem year / % dem next year)).

```
changeovertimeDF$change04_08 = 1 - changeovertimeDF$percentDem2008 / changeovertime
DF$percentDem2004
changeovertimeDF$change08_12 = 1 - changeovertimeDF$percentDem2012 / changeovertime
DF$percentDem2008
changeovertimeDF$change12_16 = 1 - changeovertimeDF$percentDem2016 / changeovertime
DF$percentDem2012
```

Save file.

```
save(changeovertimeDF, file = "changeovertimeDF.rda")
```

Create a new dataframe with relevant columns and then transpose and reformat data so each column is a unique county. Add the proper headers to each column with County-State identifier.

```
#Pull out a subset of the dataframe containing only the variables we will use.
regressionDF = data.frame(changeovertimeDF$County,
                          changeovertimeDF$percentDem2004,
                          changeovertimeDF$percentDem2008,
                          changeovertimeDF$percentDem2012)

regressionDF$changeovertimeDF.County = paste(changeovertimeDF$County, changeovertim
eDF$State, sep = "-")

#Transpose the dataframe so each column contains 1 county's voting history. Name ea
ch column the County-State pair.
transpRegressionDF = t(regressionDF)
colnames(transpRegressionDF) = transpRegressionDF[1, ]
transpRegressionDF = transpRegressionDF[-1, ]

#Vector of years to be used as the time component in our linear model.
aaatime = c(2004, 2008, 2012)

#Create a matrix version and a dataframe version of the data.
transpRegressionMatr = transpRegressionDF
transpRegressionDF = as.data.frame(transpRegressionDF)
#Reformat the dataframe so the values are all numerics.
transpRegressionDF = lapply(transpRegressionDF, function(x) {as.numeric(as.characte
r(x))})
transpRegressionDF = as.data.frame(transpRegressionDF)
```

Remove columns with too many NAs. If a data set has too many NAs, the predictor is unable to create a

linear model. In order to deal with this issue we ignore all columns with this error.

Removed values: kings new york oglala richmond 1495 2044 2093 2397

```
missingstates = which(colMeans(is.na(transpRegressionMatr)) > .5)
transpRegressionMatr = transpRegressionMatr[, -missingstates]
```

Predictor function. Using simple linear regression, given a column of data with each county's % Dem votes for each election, create a linear model where X (the % of voters) is explained by Y (Year). Then predict the YEAR (default 2016) with the equation aX + b where a is the estimated rate and b is the intercept.

```
approx = function(x, year = 2016) {
  var = lm(x ~ aaatime, na.action = na.omit)
  coef = coefficients(var)
  return(coef[2] * year + coef[1])
}
```

Apply predictor to all columns of the matrix containing each county's voting history.

```
predictions = apply(transpRegressionMatr, 2, FUN = approx)
```

Merge predictions back to orginal DF by County and State names.

```
#Use regex to separate the "county-state" names back to "county" "state" columns.

County = gsub("-.+", "", names(predictions))
State = gsub(".+-", "", names(predictions))

#Merge back into one dataframe.
predictionsdf = cbind(County, State, predictions)
colnames(predictionsdf) = c("County", "State", "Prediction2016")
predictionsdf = as.data.frame(predictionsdf)

#Convert data types to the correct ones.
predictionsdf$County = lapply(predictionsdf$County, as.character)
predictionsdf$State = lapply(predictionsdf$State, as.character)
predictionsdf$Prediction2016 = lapply(predictionsdf$Prediction2016, function(x) {as
.numeric(as.character(x))})

#Merge the two dataframes by County and State.
changeovertimeDF = merge(changeovertimeDF, predictionsdf, by = c("County", "State"
), all = T)
changeovertimeDF$Prediction2016 =  as.numeric(as.character(changeovertimeDF$Predict
ion2016))
```

```
## Warning: NAs introduced by coercion
```

Calculate the winner of the election by popular vote.

```
#Calc number of votes per county
demvotes = changeovertimeDF$Prediction2016 * changeovertimeDF$totalvotes2016
repubvotes = (1 - changeovertimeDF$Prediction2016) * changeovertimeDF$totalvotes201
6

#Predicted Num of dem votes: 63526190
PopularDemVotes = sum(demvotes, na.rm = T)
#Predicted Num of rep votes: 55680829
PopularRepVotes = sum(repubvotes, na.rm = T)

#PREDICTED WINNER: HILLARY
if(PopularRepVotes > PopularDemVotes) {
  print("2012 to 2016 Predictor: Trump Wins Popular Vote")
} else {
  print("2012 to 2016 Predictor: Hillary Wins Popular Vote")
}
```

```
## [1] "2012 to 2016 Predictor: Hillary Wins Popular Vote"
```

Add two columns. Percent Error calculates how far off our prediction was. Predict Correct is a boolean vector which shows if the Prediction was correct.

```
changeovertimeDF$percentError = changeovertimeDF$Prediction2016 - changeovertimeDF$
percentDem2016
changeovertimeDF$predictCorrect = (changeovertimeDF$percentDem2016 > .5 &
                                     changeovertimeDF$Prediction2016 > .5) |
                                   (changeovertimeDF$percentDem2016 < .5 &
                                     changeovertimeDF$Prediction2016 < .5)
```

Calculate accuracy (% correctly predicted / % incorrectly predicted).

```
#Correct: 2767 counties correctly guessed
correct = length(which(changeovertimeDF$predictCorrect))
correct
```

```
## [1] 2767
```

```
#Wrong: 337 counties incorrectly guessed
wrong = length(which(!changeovertimeDF$predictCorrect))
wrong
```

```
## [1] 337
```

```
#Accuracy: 88.91% accurate
accuracy = correct / length(changeovertimeDF$predictCorrect)
```

Calculate predicted margin of victory (Prediction - 0.5).

```
MarginOfVictory = changeovertimeDF$Prediction2016 - 0.5

#Average predicted margin of victory for Democrat and Republican states.
#Average Predicted Democratic Margin of Victory: 11.93%
AvgDemMargin = mean(MarginOfVictory[MarginOfVictory > 0], na.rm = T)
#Average Predicted Republican Margin of Victory: 17.85%
AvgRepMargin = -mean(MarginOfVictory[MarginOfVictory < 0], na.rm = T)
TrumpWin = MarginOfVictory < 0
```

Save Margin data in a dataframe for further analysis.

```
MarginOfVictoryDF = as.data.frame(cbind(changeovertimeDF$County, changeovertimeDF$S
tate, abs(MarginOfVictory), TrumpWin))
colnames(MarginOfVictoryDF) =  c("County", "State", "MarginOfVictory", "Republican
Win")

save(MarginOfVictoryDF, file = "MarginOfVictory.rda")
```

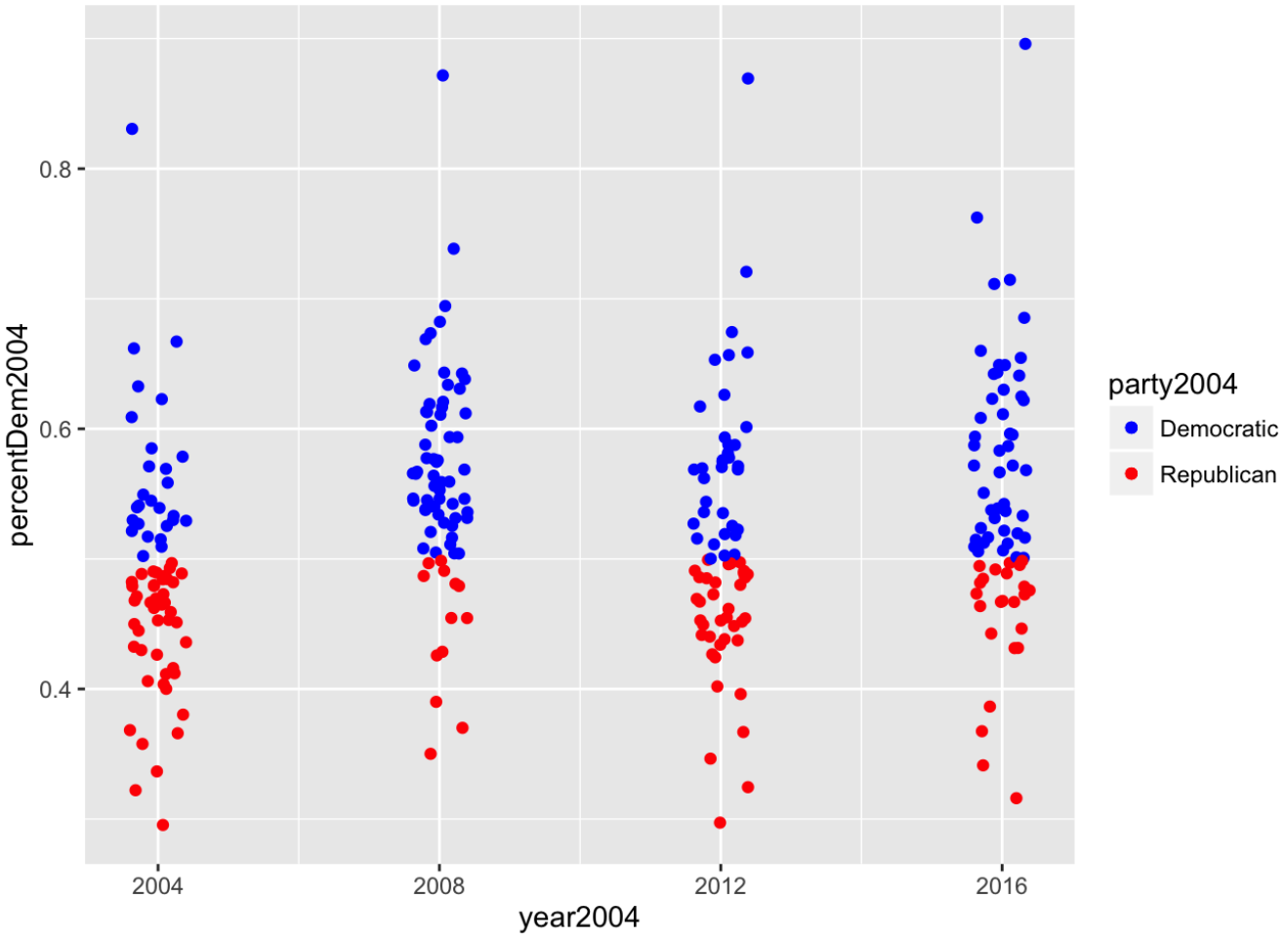
Save final file.

```
save(changeovertimeDF, file = "Predictions2012to2016.rda")
```

Exploration of the Predictor

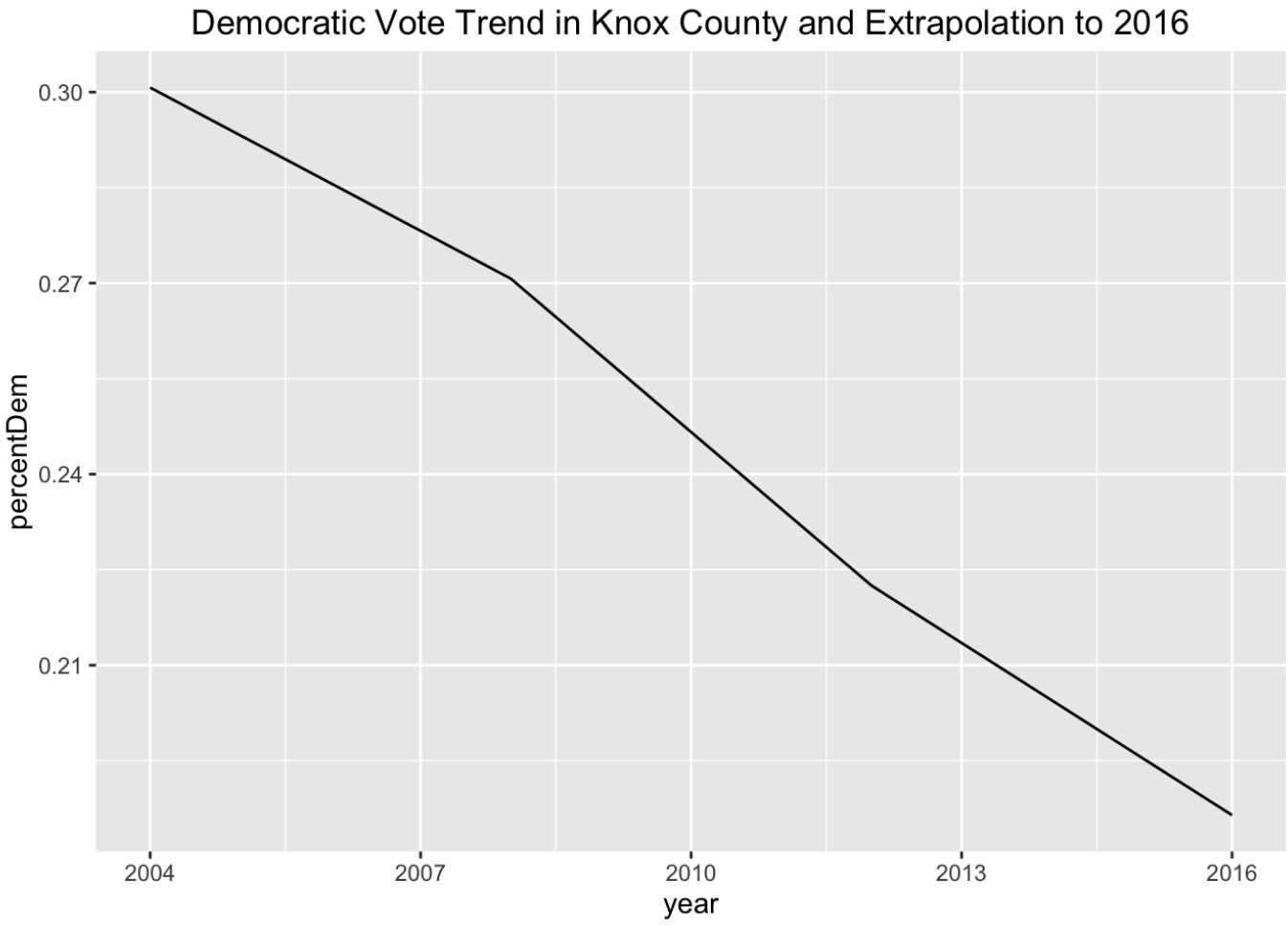
Let's take Wisconsin counties as an illustration.

```
Wisconsin_data = changeovertimeDF[changeovertimeDF$State == "wisconsin", c("County"
, "percentDem2004", "percentDem2008", "percentDem2012", "Prediction2016", "percentD
em2016")]
Wisconsin_data$year2004 = rep(2004, 72)
Wisconsin_data$year2008 = rep(2008, 72)
Wisconsin_data$year2012 = rep(2012, 72)
Wisconsin_data$year2016 = rep(2016, 72)

Func = function (x) {if (x > 0.5) "Democratic" else "Republican"}
Wisconsin_data$party2004 = sapply(Wisconsin_data$percentDem2004, Func)
Wisconsin_data$party2008 = sapply(Wisconsin_data$percentDem2008, Func)
Wisconsin_data$party2012 = sapply(Wisconsin_data$percentDem2012, Func)
Wisconsin_data$party2016 = sapply(Wisconsin_data$Prediction2016, Func)

ggplot(data = Wisconsin_data) + geom_jitter(aes(x = year2004, y = percentDem2004,
color = party2004)) + geom_jitter(aes(x = year2008, y = percentDem2008, color = pa
rty2008)) + geom_jitter(aes(x = year2012, y = percentDem2012, color = party2012))
+ geom_jitter(aes(x = year2016, y = Prediction2016, color = party2016)) + scale_co
lour_manual(values= c("blue", "red") )
```



Examine the predictor on the county that have min predicting error. Through this plot, we can see how the predictor uses linear regression to predict the results of 2016.

```
mostac_county = changeovertimeDF[which.min(abs(changeovertimeDF$percentError)), ]
new.df1 = data.frame(year = c(2004, 2008, 2012, 2016), percentDem = cbind(mostac_c
ounty$percentDem2004, mostac_county$percentDem2008, mostac_county$percentDem2012, m
ostac_county$Prediction2016)[1, ])
ggplot(data = new.df1) + geom_line(aes(x = year, y = percentDem)) + ggtitle("Democ
ratic Vote Trend in Knox County and Extrapolation to 2016")
```



Discussion

In pulling all our results and mappings together, we noticed some interesting results.

The 2012 to 2016 predictor performed better with a success rate of 88.91% compared to the 2016 predictor which had a success rate of 90.01%. The main difference between the two predictors was that the 2016 predictor was based on 2010 census data while the 2012 to 2016 predictor was based on voting trends.

As will be shown below, something interesting to note is that even our predictor with an 90.01% success rate was unable to predict the margins of victory for Trump. In our analysis, we created plots of all U.S. counties that graphically displayed the margins of victory in each county. Both our predictors came to two erroneous conclusions that were similar to what was predicted by analysts in the media: 1) They overestimated the number of counties that Clinton would win over. 2) They predicted smaller margins in many of the counties that Trump won, when in actuality his margin of victory was staggeringly large. Examples would be many of the swing states (e.g., Wisconsin) that went to Trump.

So although both predictors ultimately failed to predict a Trump victory, many of our conclusions drawn paralleled that of other data analysis reports leading up to the election. Given our success rate on both predictors, it seems that our predictors were successful in being able to predict the voting preferences of individual counties. However, what they failed to do was take into account a more holistic analysis and see the implications within each state Because ultimately the election winner is based upon electoral votes as opposed to sheer number of counties won or individual margins of victory, these may have been factors that skewed our results. Our analysis most likely fell to many of the same obstacles that other Clinton-predictors' did, and we have a far clearer understanding of how that could have come to pass.

Mapping Each Candidates Margin of Victory

Here, we create a plot that points on each county as well. The color of the dot now indicates the margin of victory, while the size of the dot is still county population.

```
# Create two new data frames based on whether Clinton or Trump won those counties.
trump_majority = final[final$`Clinton v. Trump` == "Trump", ]
clinton_majority = final[final$`Clinton v. Trump` == "Clinton", ]
```

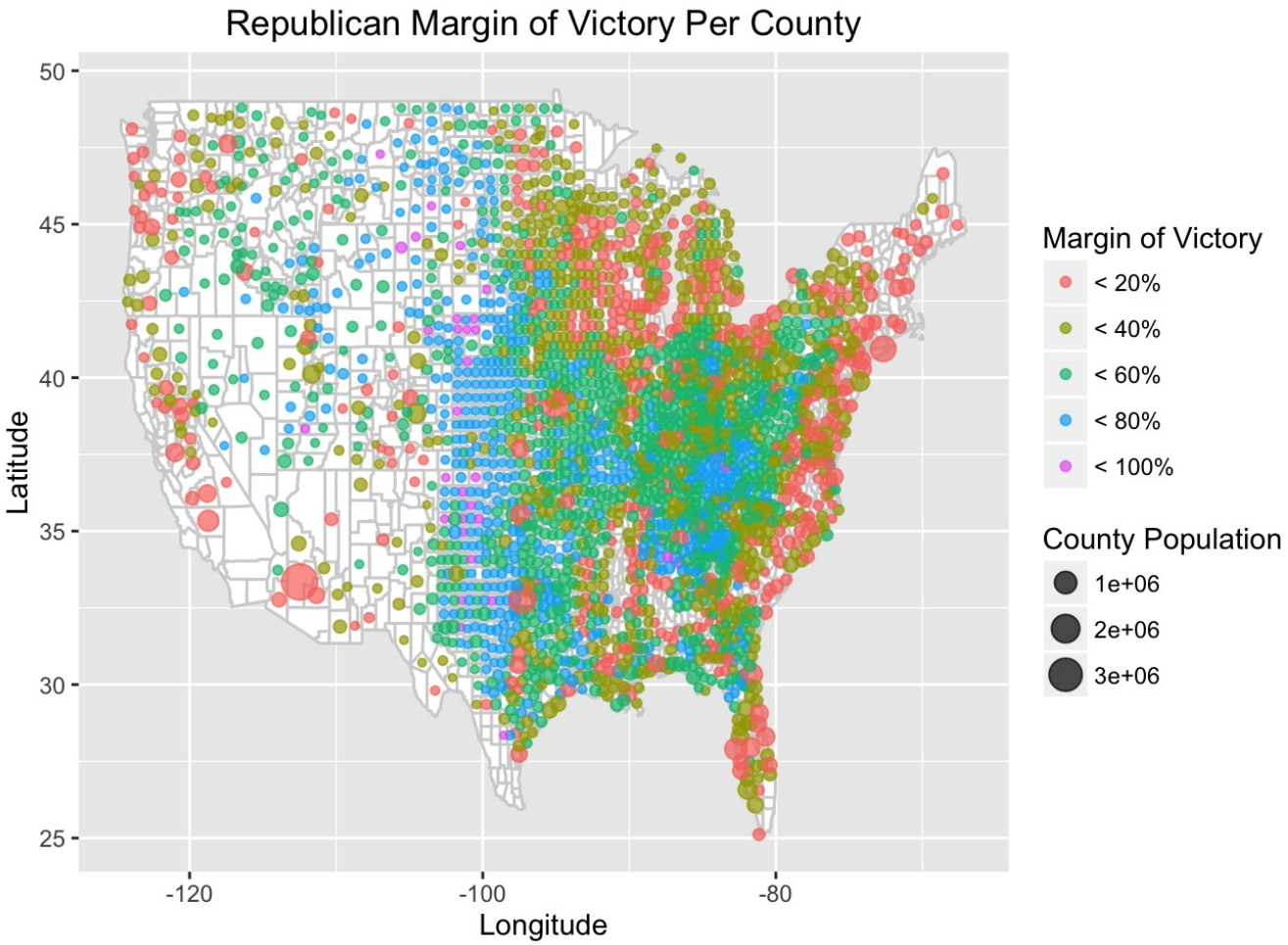
The Republican Margin of Victory

```
# Calculate the margin of victory in percentage.
trump_majority$margin_of_victory = (trump_majority$`Percent Trump 2016` - trump_maj
ority$`Percent Hillary 2016`) * 100

trump_majority$margin_of_victory = sapply(trump_majority$margin_of_victory, functio
n(x) {
  if (is.na(x)) {return(0)}
  if (x < 20) {return(0)}
  else if (x < 40) {return(1)}
  else if (x < 60) {return(2)}
  else if (x < 80) {return(3)}
  else {return(4)}
})

# Convert the margin of victory into 5 factor levels.
trump_majority$margin_of_victory = factor(trump_majority$margin_of_victory, levels
= c(0, 1, 2, 3, 4), labels = c("< 20%", "< 40%", "< 60%", "< 80%", "< 100%"))

pal = c("#f4564d","#db4d45","#c3443d","#aa3c35","#92332e")

# Plot
map_plot +
  geom_polygon(data=all_counties, aes(x=long, y=lat, group = group), colour="light
gray", fill="white" ) +
  geom_point(data = trump_majority, aes(x = Longitude, y = Latitude, color = margi
n_of_victory, size = `Total Pop`), alpha = 0.7) +
  scale_x_continuous(name = "Longitude") +
  scale_y_continuous(name = "Latitude") +
  ggtitle("Republican Margin of Victory Per County") +
  labs(size = "County Population", color = "Margin of Victory") +
  scale_fill_manual(values = pal)
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which
## will replace the existing scale.
```

```
## Scale for 'y' is already present. Adding another scale for 'y', which
## will replace the existing scale.
```

```
## Warning: Removed 6 rows containing missing values (geom_point).
```



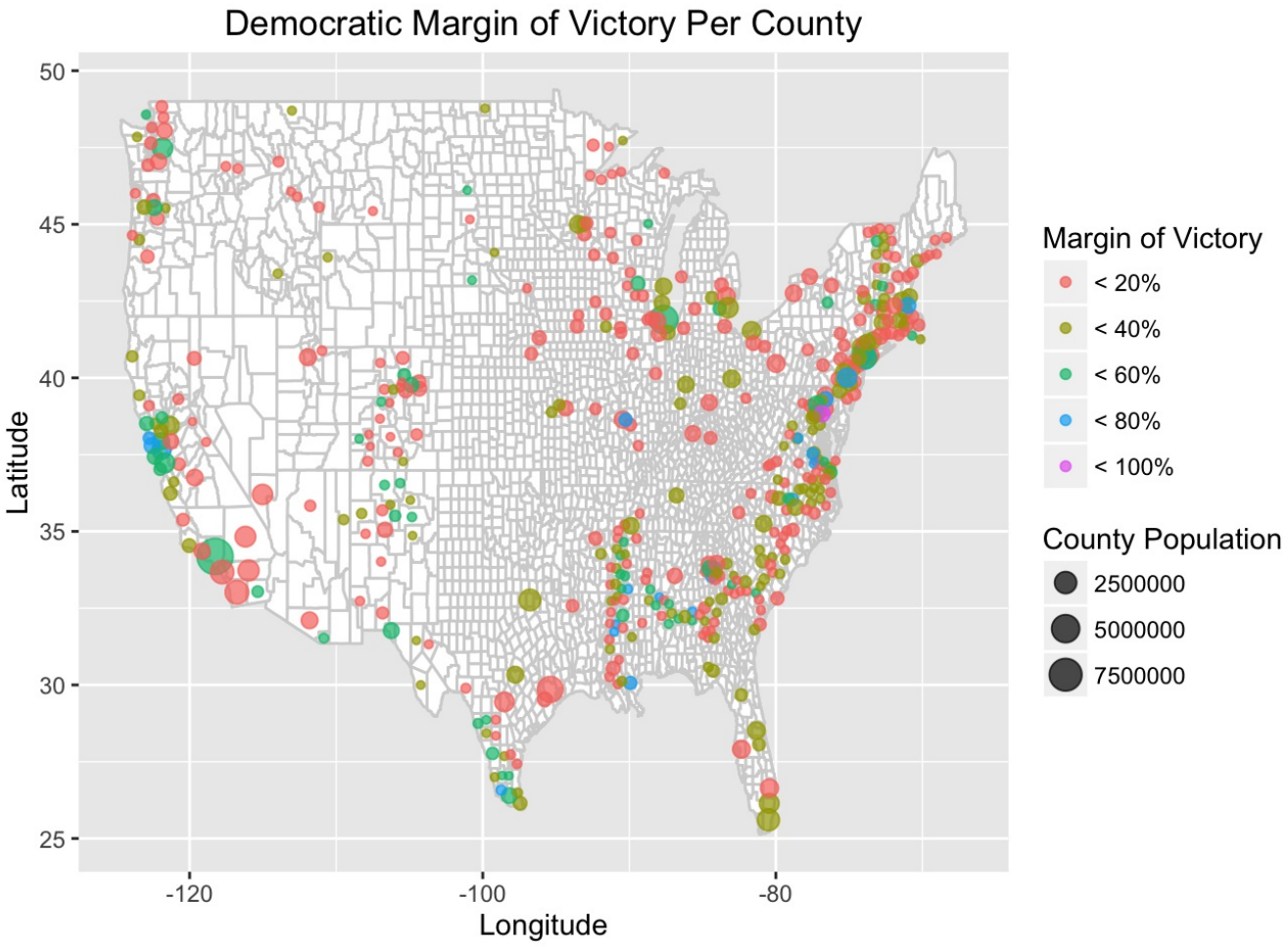
The Democratic Margin of Victory

```r
# Calculate the margin of victory in percentage.
clinton_majority$margin_of_victory = (clinton_majority$`Percent Hillary 2016` - cli
nton_majority$`Percent Trump 2016`) * 100

clinton_majority$margin_of_victory = sapply(clinton_majority$margin_of_victory, fun
ction(x) {
  if (is.na(x)) {return(0)}
  if (x < 20) {return(0)}
  else if (x < 40) {return(1)}
  else if (x < 60) {return(2)}
  else if (x < 80) {return(3)}
  else {return(4)}
})

# Convert the margin of victory into 5 factor levels.
clinton_majority$margin_of_victory = factor(clinton_majority$margin_of_victory, lev
els = c(0, 1, 2, 3, 4), labels = c("< 20%", "< 40%", "< 60%", "< 80%", "< 100%"))

# Plot
map_plot +
  geom_polygon(data=all_counties, aes(x=long, y=lat, group = group), colour="light
gray", fill="white" ) +
  geom_point(data = clinton_majority, aes(x = Longitude, y = Latitude, color = mar
gin_of_victory, size = `Total Pop`), alpha = 0.7) +
  scale_x_continuous(name = "Longitude") +
  scale_y_continuous(name = "Latitude") +
  ggtitle("Democratic Margin of Victory Per County") +
  labs(size = "County Population", color = "Margin of Victory")
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which
## will replace the existing scale.
```

```
## Scale for 'y' is already present. Adding another scale for 'y', which
## will replace the existing scale.
```

```
## Warning: Removed 7 rows containing missing values (geom_point).
```



Comparing with Predicted Margins of Victory

Now that we have plots of the margin of victory based on the actual data, we can make comparisons between these plots and the margins of victory created by our predictors.

```
load('MarginOfVictory.rda')

colnames(MarginOfVictoryDF) = c("County", "State", "Predicted 2016 Margin", "Predicted Clinton v. Trump")
final = merge(final, MarginOfVictoryDF, by = c("County", "State"))

pred_trump_majority = final[final$`Predicted Clinton v. Trump` == "TRUE", ]
pred_clinton_majority = final[final$`Predicted Clinton v. Trump` == "FALSE", ]
```

The Predicted Republican Margin of Victory

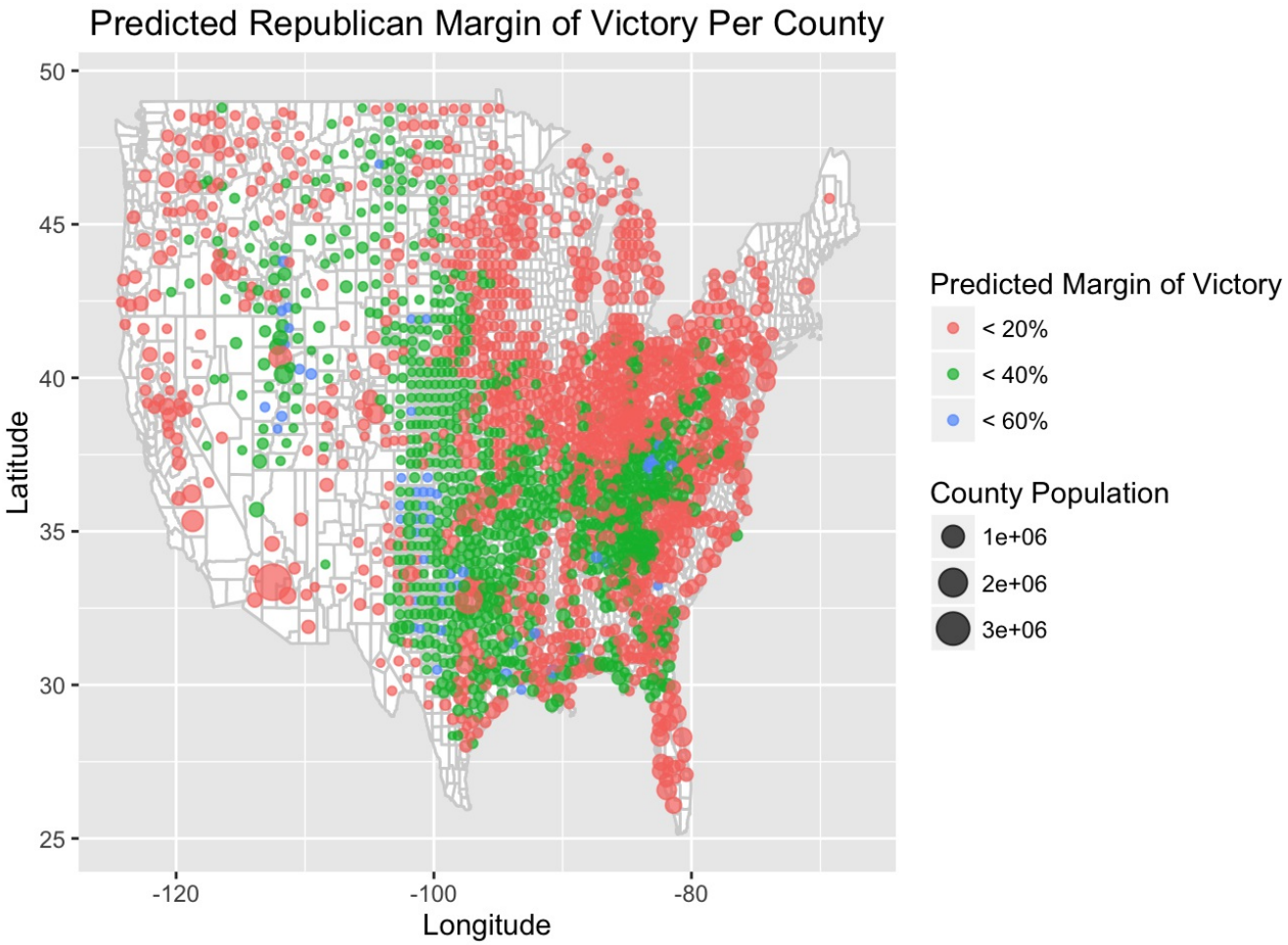
This map will plot the predicted Republican margin of victory in counties that Trump was predicted to have a majority in.

```r
# Calculate the margin of victory in percentage.
pred_trump_majority$`Predicted 2016 Margin` = as.numeric(as.character(pred_trump_majority$`Predicted 2016 Margin`)) * 100

pred_trump_majority$`Predicted 2016 Margin` = sapply(pred_trump_majority$`Predicted 2016 Margin`, function(x) {
  if (is.na(x)) {return(0)}
  if (x < 20) {return(0)}
  else if (x < 40) {return(1)}
  else if (x < 60) {return(2)}
  else if (x < 80) {return(3)}
  else {return(4)}
})

# Convert the margin of victory into 5 factor levels.
pred_trump_majority$`Predicted 2016 Margin` = factor(pred_trump_majority$`Predicted 2016 Margin`, levels = c(0, 1, 2, 3, 4), labels = c("< 20%", "< 40%", "< 60%", "< 80%", "< 100%"))

pal = c("#f4564d","#db4d45","#c3443d","#aa3c35","#92332e")

# Plot
map_plot +
  geom_polygon(data=all_counties, aes(x=long, y=lat, group = group), colour="light gray", fill="white" ) +
  geom_point(data = pred_trump_majority, aes(x = Longitude, y = Latitude, color = `Predicted 2016 Margin`, size = `Total Pop`), alpha = 0.7) +
  scale_x_continuous(name = "Longitude") +
  scale_y_continuous(name = "Latitude") +
  ggtitle("Predicted Republican Margin of Victory Per County") +
  labs(size = "County Population", color = "Predicted Margin of Victory") +
  scale_fill_manual(values = pal)
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which
## will replace the existing scale.
```

```
## Scale for 'y' is already present. Adding another scale for 'y', which
## will replace the existing scale.
```

```
## Warning: Removed 6 rows containing missing values (geom_point).
```



As shown by the plot, our predictor believed that the election itself would be far closer than it was. Graphically, the multitude of red dots indicated that our predictor believed that the margins of victory would be much smaller (and the presidential race would be much closer) than the actuality. Interesingly, this matches with the predictions of much of the media prior to the presidential election, who believed that Trump would have a far more difficult time winning over key regions than he actually did.

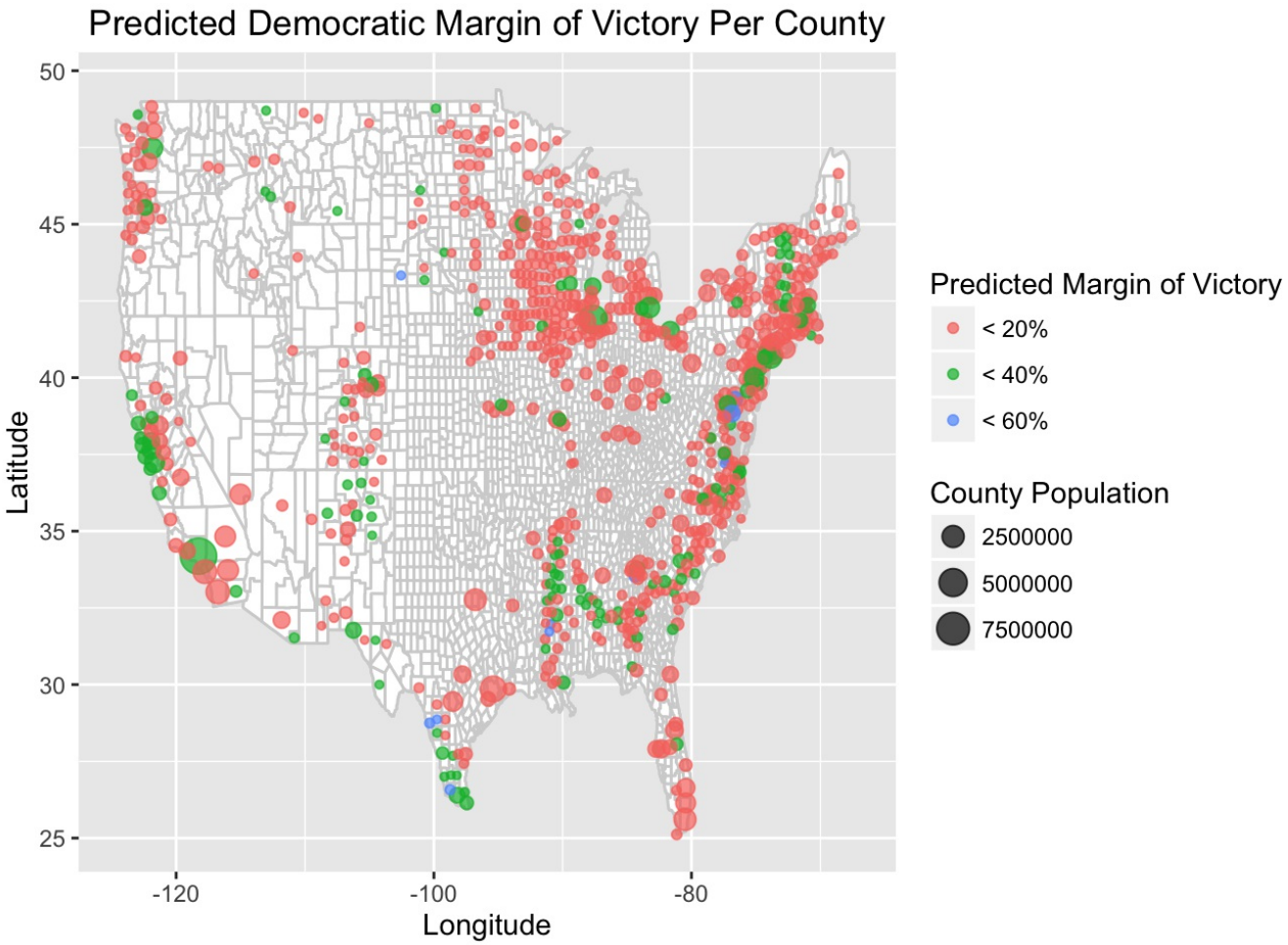
The Predicted Democratic Margin of Victory

```r
# Calculate the margin of victory in percentage.
pred_clinton_majority$`Predicted 2016 Margin` = as.numeric(as.character(pred_clinto
n_majority$`Predicted 2016 Margin`)) * 100

pred_clinton_majority$`Predicted 2016 Margin` = sapply(pred_clinton_majority$`Predi
cted 2016 Margin`, function(x) {
  if (is.na(x)) {return(0)}
  if (x < 20) {return(0)}
  else if (x < 40) {return(1)}
  else if (x < 60) {return(2)}
  else if (x < 80) {return(3)}
  else {return(4)}
})

# Convert the margin of victory into 5 factor levels.
pred_clinton_majority$`Predicted 2016 Margin` = factor(pred_clinton_majority$`Predi
cted 2016 Margin`, levels = c(0, 1, 2, 3, 4), labels = c("< 20%", "< 40%", "< 60%"
, "< 80%", "< 100%"))

pal = c("#f4564d","#db4d45","#c3443d","#aa3c35","#92332e")

# Plot
map_plot +
  geom_polygon(data=all_counties, aes(x=long, y=lat, group = group), colour="light
gray", fill="white" ) +
  geom_point(data = pred_clinton_majority, aes(x = Longitude, y = Latitude, color
= `Predicted 2016 Margin`, size = `Total Pop`), alpha = 0.7) +
  scale_x_continuous(name = "Longitude") +
  scale_y_continuous(name = "Latitude") +
  ggtitle("Predicted Democratic Margin of Victory Per County") +
  labs(size = "County Population", color = "Predicted Margin of Victory") +
  scale_fill_manual(values = pal)
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which
## will replace the existing scale.
```

```
## Scale for 'y' is already present. Adding another scale for 'y', which
## will replace the existing scale.
```

```
## Warning: Removed 9 rows containing missing values (geom_point).
```



Here, our predictor seemed to graphically match with the reality a bit better than the predicted Republican margin of victory. The main differing factor, though, is that our predictions assigned more counties to favoring Clinton than the reality did. Like with the predicted Republican margin of victory, our predictor came to similar conclusions as the predictions of the media prior to the election. For example, looking at the counties making up Wisconsin, our predictor predicts that Wisconsin will have a small margin of victory (as a swing state) but will ultimately go to Clinton. Reality, however, defied this as well as the predictions of the media.

References

2016 Presidential Election Results at County Level: http://www.stat.berkeley.edu/users/nolan/data/voteProject/2016_US_County_Level_Presidential_Results.csv

2012 Presidential Election Results at County Level: http://www.politico.com/2012-election/map/#/President/2012/

2008 Presidential Election Results at County Level: https://www.theguardian.com/news/datablog/2009/mar/02/us-elections-2008

2004 Presidential Election Results at County Level: http://www.stat.berkeley.edu/users/nolan/data/voteProject/countyVotes2004.txt

2010 Census Data: https://factfinder.census.gov/faces/nav/jsf/pages/searchresults.xhtml?refresh=t

GML County Location Data: http://www.stat.berkeley.edu/users/nolan/data/voteProject/counties.gml

Packages used:

```
sessionInfo()
```

```
## R version 3.3.1 (2016-06-21)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: OS X 10.11.6 (El Capitan)
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] class_7.3-14    mapdata_2.2-6   maps_3.1.1      gridExtra_2.2.1
## [5] ggplot2_2.1.0
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.6      digest_0.6.10    plyr_1.8.4       grid_3.3.1
##  [5] gtable_0.2.0     formatR_1.4      magrittr_1.5     evaluate_0.9
##  [9] scales_0.4.0     stringi_1.1.1    reshape2_1.4.1   rmarkdown_1.0
## [13] labeling_0.3     tools_3.3.1      stringr_1.1.0    munsell_0.4.3
## [17] yaml_2.1.13      colorspace_1.2-6 htmltools_0.3.5  knitr_1.14
```