Statistics 133 - Final Project - "Team Feelin' the Bern"

Aniket Ketkar, Tanay Lathia, Eric Priest, Steve Wang

Contents

1 Abstract

This project involves the 2016 United States presidential primary races and the role of data in the campaign process. The goal is to show what ways campaign teams can use election and demographic data to help them properly allocate campaign resources. We predict that through the use of linear regression, we will be able to isolate the main variables that influence voting results and use those variables to create a prediction model for future states. We built a vectorized K Nearest Neighbors (machine learning) model from scratch, running it on the chosen features in the demographic data. We conclude that regression features with KNN produce adequate results for primary voting prediction, with surprising results in homogeneous counties such as Georgia.

2 Background

For the past year, the US Presidential election has consumed the attention of the American people. While many of the publications deal with who said what, the role of data cannot be understated, as we often see polls extrapolated to predict primary results or we see actual election data made into consumer friendly visualizations. With this in mind, our group wanted to look into the different ways data could be used to predict election results and how campaign team behind the scenes could use data. Through Kaggle, we found a data set involving peoples voting patterns in the primaries by county. More specifically, for each county in each state and for each major political party, the data told us the percentage of votes and also the demographic of the people within the county, including features like percentage of each race, average number of households, and percentage of each age range, among many more. The data was sourced from `https://www.kaggle.com/benhamner/2016-us-election`, compiled by Ben Hamner. See more cleaning details in the Data Analysis Methods section, and `util.R`.

3 Questions for Analysis

Using this data, we set out to answer these two primary questions:

- How can we predict the results for a certain county?
- Candidates want to canvass counties that are swing counties, or borderline in terms of votes. Using the data, can we predict which counties would be swing counties based on the features of their demographics?

We chose this idea over others because we wanted to see how we could apply data and the skills we learned in class to modern day relevant issues. Obviously, no one can exactly foresee the outcomes election, but we found the idea of seeing how close we could get to be enticing.

4 Data Analysis Methods

We obtained the data from Kaggle. The main data files that we used were the primary_results.csv and the county_facts.csv files. The primary_results.csv file gave us data as to how each county voted, for both parties and the county_facts.csv file gave us data as to the demographics of each county. The first step we needed to take was to wrangle the data. The data came in the form of three .csv files, one that had the election results for each county, one that had the demographic data for each county with the variables in codes, and one that had the key for the variables. We decided to inner join the county demographic data frame with the election results data frame to create a large data table that had all of the relevant information we wanted to work with. We then created separate data frames for each candidate to get isolated results for each by creating subsets of the large data frame. This separation was helpful later when we ran linear regressions for the candidates.

In order to find the answers to these questions, we needed to slim down which features of the demographics we wanted to analyze. Itd take far too much time to process each of these features so the most efficient method is to find the ones that impacted voting patterns the most. We ran linear regressions between each demographic variable and votes to see which variables correlated the most, and with this analysis we determined that racial variables and the average number of household members were the most influential factors. Once we found the main factors, we were able to make plots that depicted just how much influence those variables had.

We used K Nearest Neighbors in order to generate predictions based off a feature. KNN is explained more in depth in the next section, but its main purpose is as a classifier. For each state, our training data was every county in the United States minus that state, and we our testing data was every county within the state.

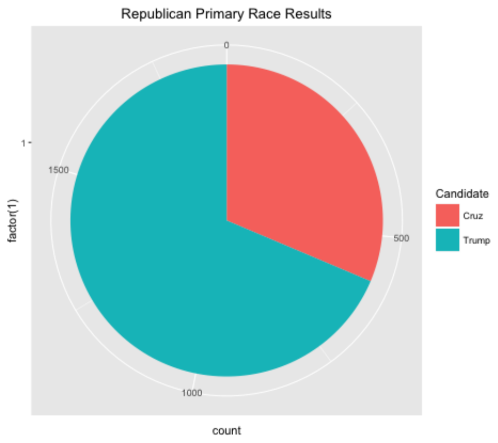
5 Technical Methods

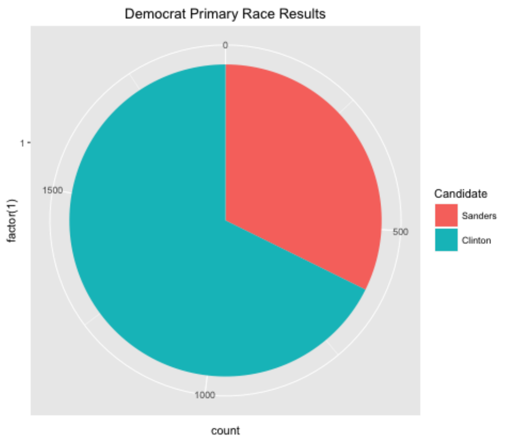
Technologies used:

1. `R` scripts
2. `choroplethr`, `ggplot2`, `DataComputing` libraries
3. Github and git, see `https://github.com/ketkar/election` for commit history
4. LaTeXfor final writeup (see Github for code)
5. Object oriented programming/abstraction.
6. Vectorized functions (apply)
7. Caching results using `.rds` files

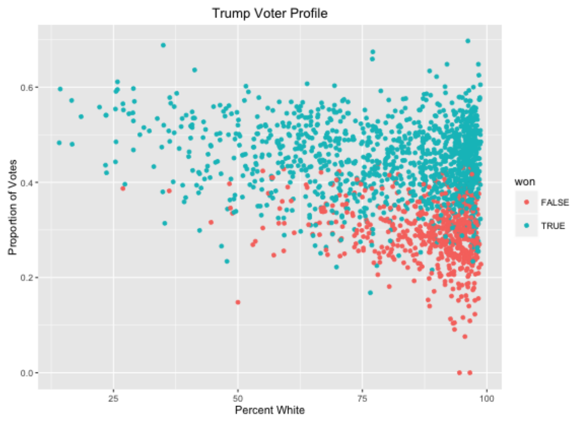
6 Exploratory Analysis

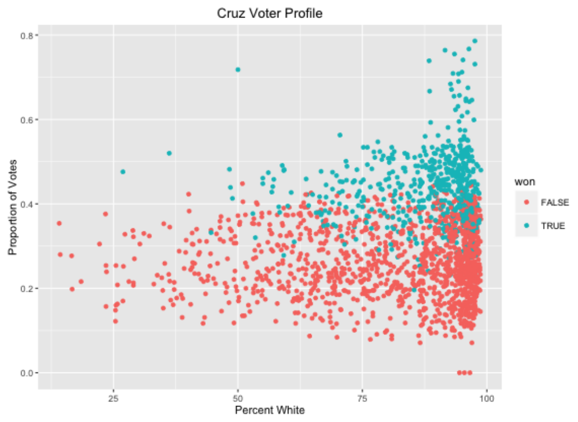
Before we jumped into performing machine learning to predict county results, we created a few visualizations to see any obvious trends in the data. The first is just a very simple pie-chart of how the race was stacking up at the time the data were collected.

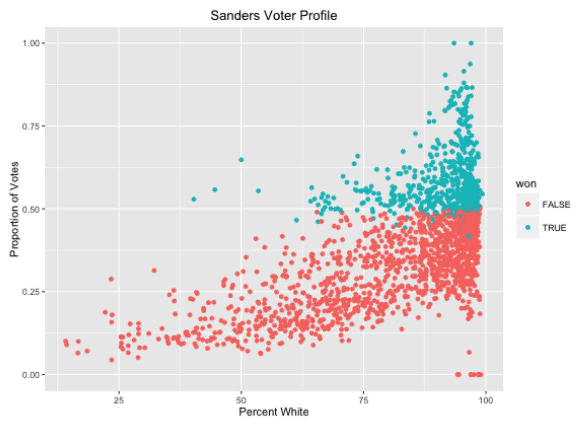


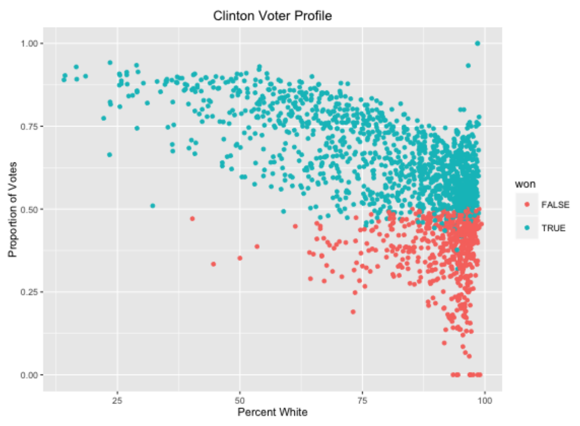


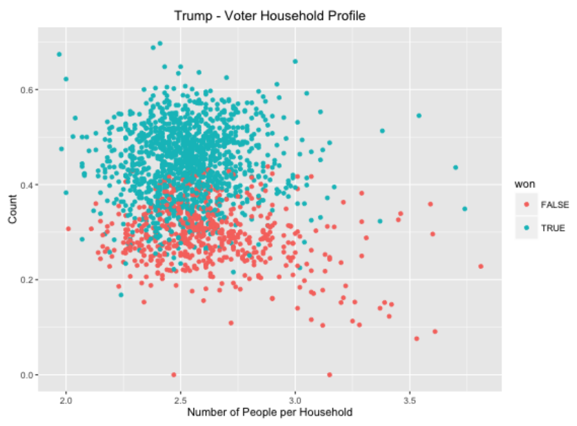
These graphs show the percentage of counties won by each candidate in their respective party. These plots are a great spot to begin our exploratory analysis because firstly they provide a motivation for the analysis: Sanders and Cruz are both on the backfoot in their party races. They need to determine which voter groups they are not hitting and refocus their efforts in order to stand a chance at claiming the nominations. Furthermore, it is important to consider these charts because the skewed nature of the current state of the primaries will bias any results we generate - especially when using K Nearest Neighbors, an unsupervised learning technique. The nature of the this tool, which we will describe in greater detail in the next section, lends itself to following the bias of the data that has already been collected; so, it is important to be fully aware of the biases before extrapolating results for predictions.

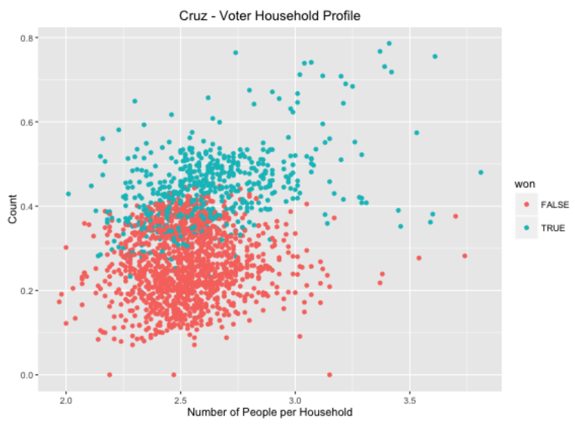


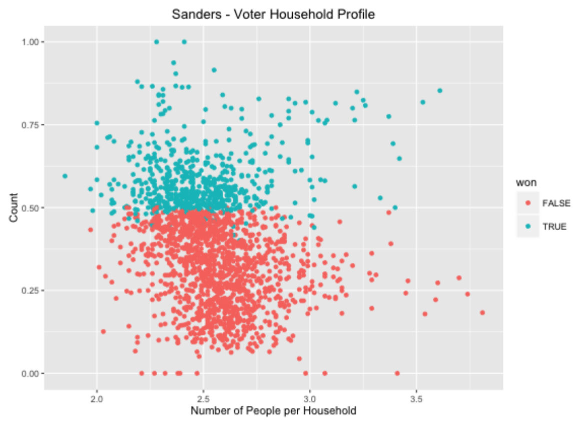


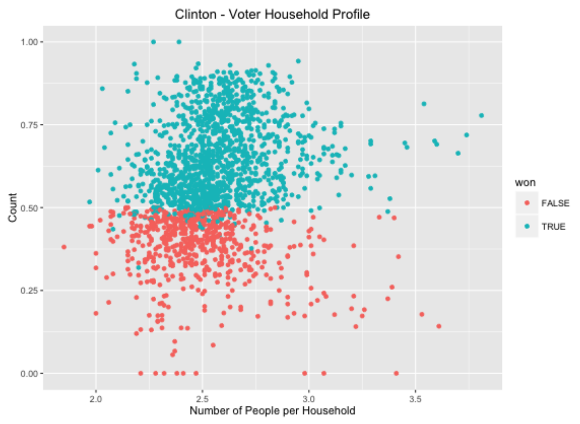

Sanders Voter Profile
Proportion of Votes
1.00
0.75
0.50
0.25
0.00
25
50
75
100
Percent White
won
FALSE
TRUE

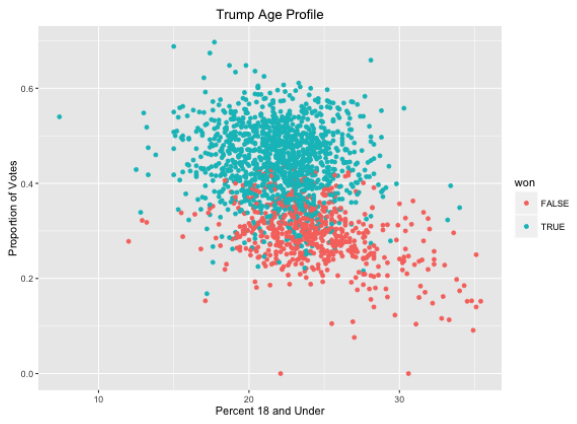

Clinton Voter Profile
Proportion of Votes
1.00
0.75
0.50
0.25
0.00
25
50
75
100
Percent White
won
FALSE
TRUE

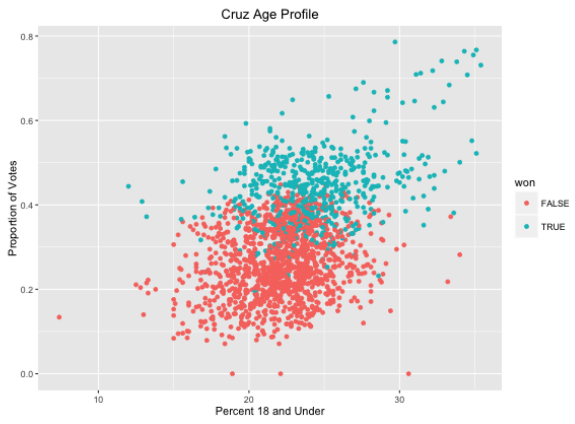

Trump - Voter Household Profile
Count
0.6
0.4
0.2
0.0
2.0
2.5
3.0
3.5
Number of People per Household
won
FALSE
TRUE

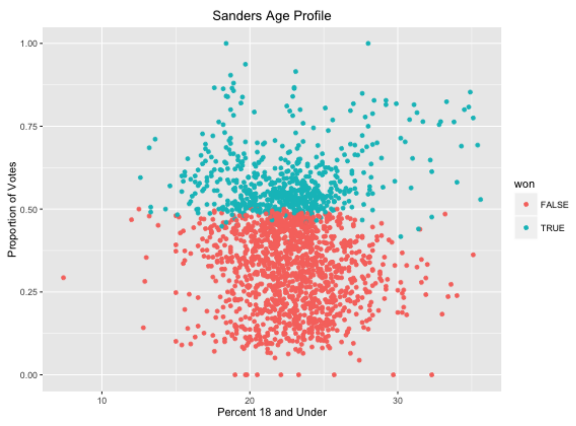

Cruz - Voter Household Profile
Count
0.8
0.6
0.4
0.2
0.0
2.0
2.5
3.0
3.5
Number of People per Household
won
FALSE
TRUE

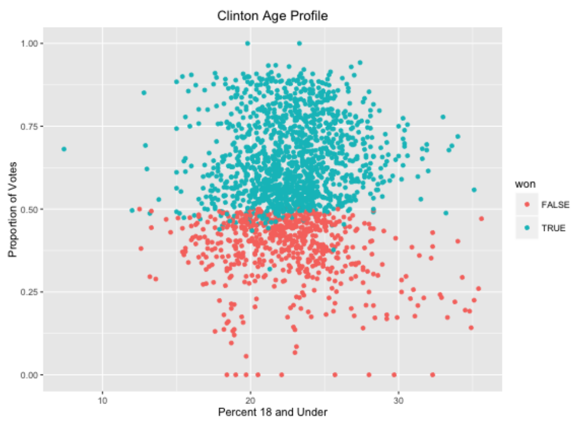

Sanders - Voter Household Profile
Count
1.00
0.75
0.50
0.25
0.00
2.0
2.5
3.0
3.5
Number of People per Household
won
FALSE
TRUE


Clinton - Voter Household Profile
Count
1.00
0.75
0.50
0.25
0.00
2.0
2.5
3.0
3.5
Number of People per Household
won
FALSE
TRUE


Trump Age Profile
Proportion of Votes
0.6
0.4
0.2
0.0
10
20
30
Percent 18 and Under
won
FALSE
TRUE


Cruz Age Profile
Proportion of Votes
0.8
0.6
0.4
0.2
0.0
10
20
30
Percent 18 and Under
won
FALSE
TRUE





Despite the shortcomings of KNN, our full exploratory analysis suggested that it made the most sense for us to use in our attempt at classification. Above are a selection of the plots we generated while reviewing the dataset. In every plot, each data point represents a county. The y-axis is always the proportion of votes that the plot's candidate received. The point of interest is always the x-axis which is the feature we were examining. In order from top to bottom, the plots examine how well each candidate does with respect to: proportion of white voters, average number of people per household, and proportion of young voters. Furthermore the plots are organized such that Cruz and Trump are next to each other and Clinton and Sanders are also next to each other in each row. This is so that inter-party comparisons (which are the only valid ones we can make based on this data) are easier. Also, the proportions of voters for Cruz plus Trump add up to 1 (approximately) and the same goes for Sanders and Clinton. This is because only these four candidates got any real traction in the election and this makes our assumption that there are only four candidates in the race.

The most compelling plot is the first one - how the proportion of votes received varies with proportion with white voters. We can see that for both Trump and Clinton if there are more than 50% non-white voters it is very, very likely that they win the county. The county is more of a tossup for counties with more white voters, but seeing this first plot made us hopeful that we could generate a linear classifier. Using just one feature, we can predict very accuracy counties with less than 50% white voters. However, as we explored more of the features in the data set, we found that no other demographic indicator was nearly as powerful as amount of whites. For example, with number of people per household Cruz seems to have more appeal than Trump for families however this doesn't seem to impact the Democratic race. There is also a similar trend that favors Cruz for younger voters - however it is not nearly as significant as the white voters feature.

Because of the limited data separation that we could get from adding more features, we determined that a linear classifier would not do well. The features we had access did not provide enough resolution between certain counties. Because of these limitations, we decided that KNN would be an appropriate model that would be the easiest for us to tune to get accurate results.

7 Machine Learning - K Nearest Neighbors

7.1 KNN Equations and Algorithm

K Nearest neighbors is a nonparametric machine learning technique to find similar points in a dataset. It makes the assumption that similar points will have similar results, so point's label is predicted as the average label of its neighbors. Check out `knn.R` for the team's implementation of KNN. KNN does not make any assumptions about the structure of the dataset, underlying trends, or high level ideas. It just finds the nearest neighbors. For large datasets, like the one in this paper, KNN must run through all of the training points before it can make a prediction. Performance is guaranteed to be linear ($O(nf)$) with the size of the training data n and number of features f. The more features, the farther points will be in space due to the curse of dimensionality, so we had to cut down on the featureset to make predictions better.

Let a single data point be $\mathbf{x_i} = (f_1, f_2, f_3 ...)$, in entire dataset $\mathbf{X}$. Then for training data point $\mathbf{x_j}$ for $i \neq j$, define the L2 distance metric for a single training example as $t_j = \sqrt{\Sigma(\mathbf{x_i} - \mathbf{x_j})^2}$. The training point $\mathbf{x_j}$ with minimum t_j is the closest example to the desired prediction point.

The algorithm is:

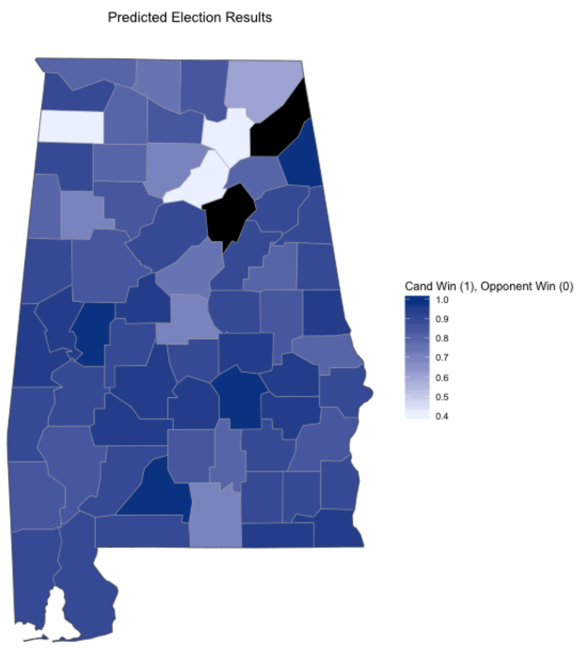
1. For all $j \neq i$, compute t_j
2. Find k smallest t_j
3. Return average of these k values.

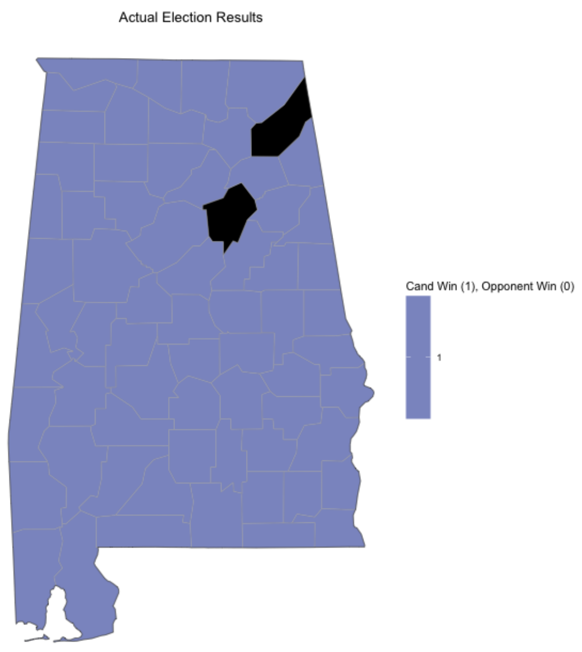
As you can see, this involves the entire training set. For state level predictions, we removed the state we are predicting on, leaving approximately 1600 counties to predict on. Each state prediction takes approximately 4 minutes (approximately 20 individual predictions).

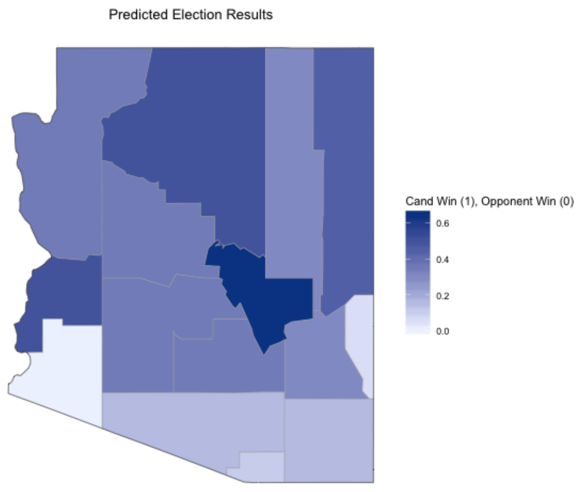
Despite the downsides, KNN performed much better than chance. Assuming two people races, a uniform assumption has expected win rate of 0.5, but every state had at least 70% success, with some over 90

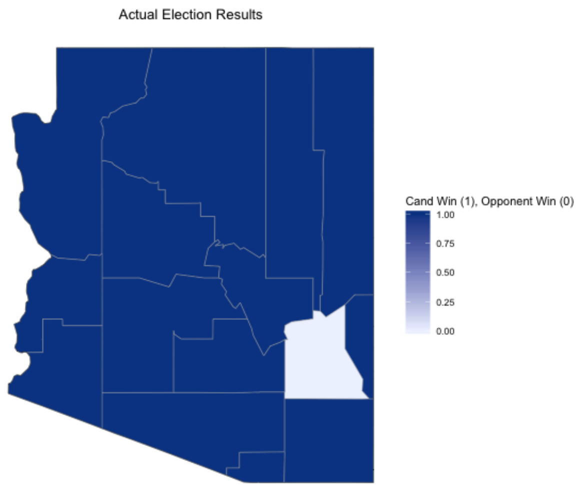
7.2 Trump v Cruz Plots

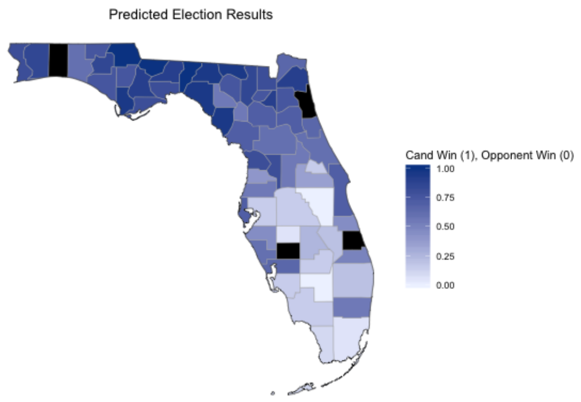
Trump's data worked mostly well with KNN. Below are some examples of states. The predictions are on the left, the actual results are on the right. Darker blue areas correspond to his predicted wins. See the Analysis section for breakdowns.

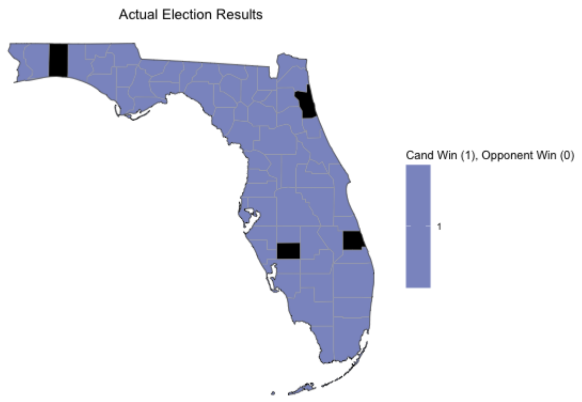

Predicted Election Results
Cand Win (1), Opponent Win (0)
1.0
0.9
0.8
0.7
0.6
0.5
0.4

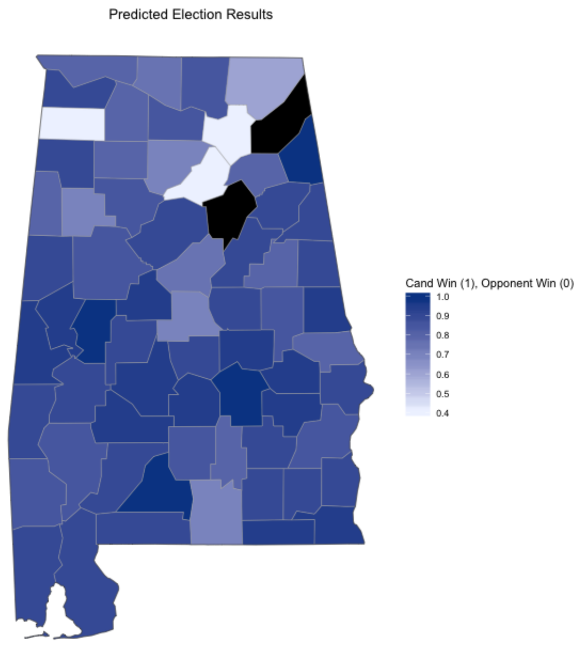

Actual Election Results
Cand Win (1), Opponent Win (0)
1

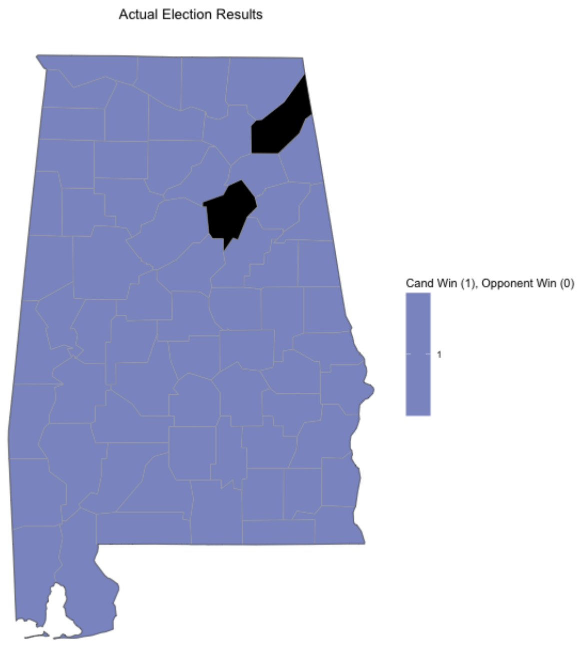

Predicted Election Results
Cand Win (1), Opponent Win (0)
0.6
0.4
0.2
0.0

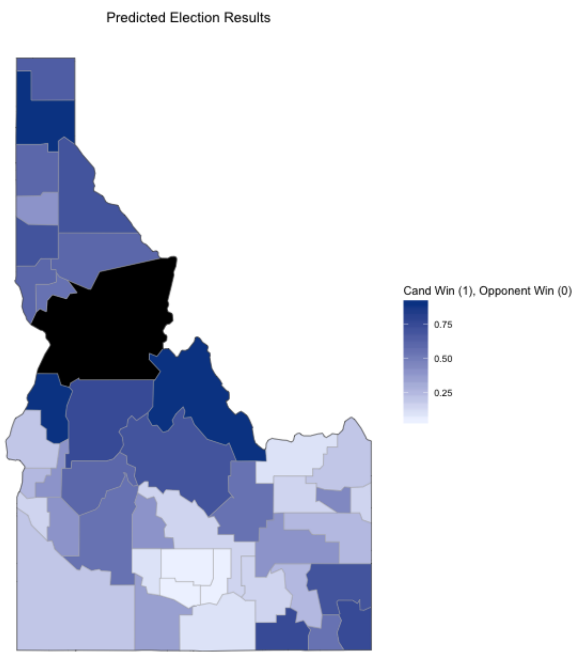

Actual Election Results
Cand Win (1), Opponent Win (0)
1.00
0.75
0.50
0.25
0.00

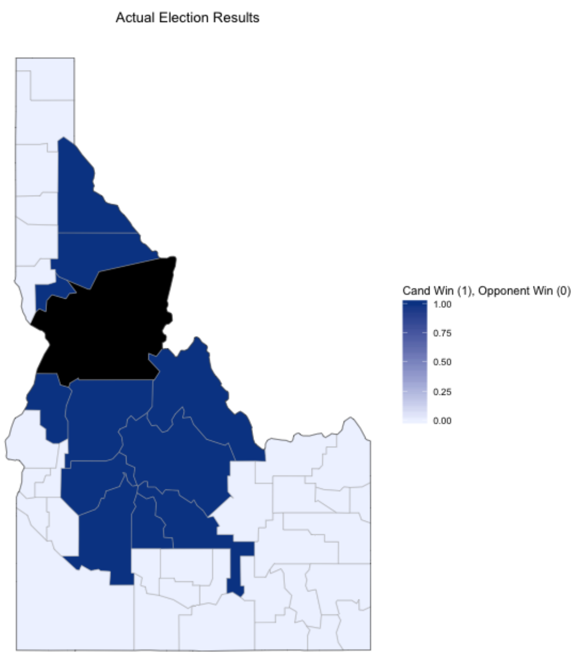

Predicted Election Results
Cand Win (1), Opponent Win (0)
1.00
0.75
0.50
0.25
0.00


Actual Election Results
Cand Win (1), Opponent Win (0)
1


Predicted Election Results
Cand Win (1), Opponent Win (0)
1.0
0.9
0.8
0.7
0.6
0.5
0.4

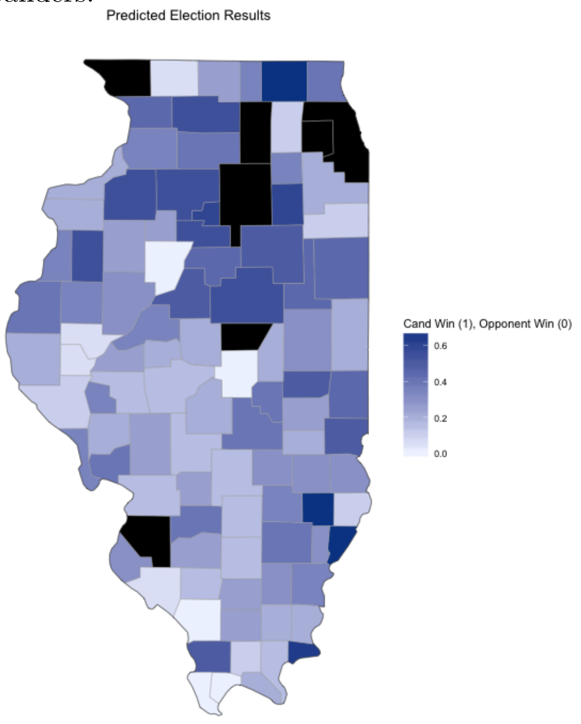

Actual Election Results
Cand Win (1), Opponent Win (0)
1

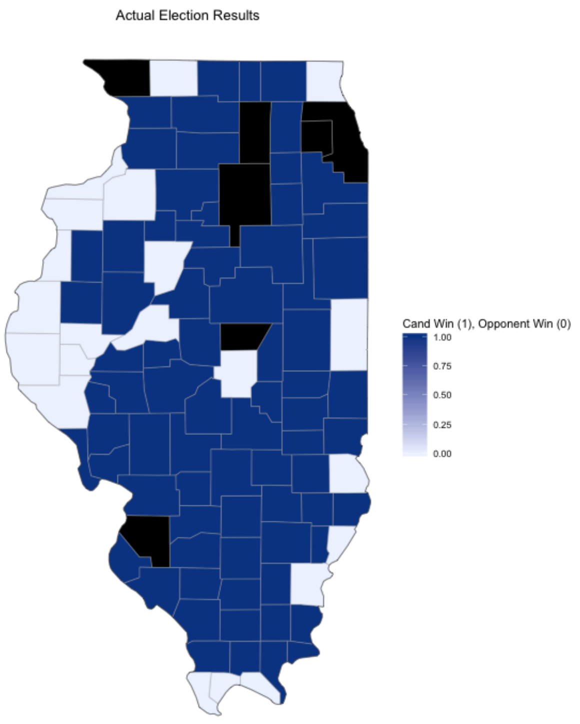


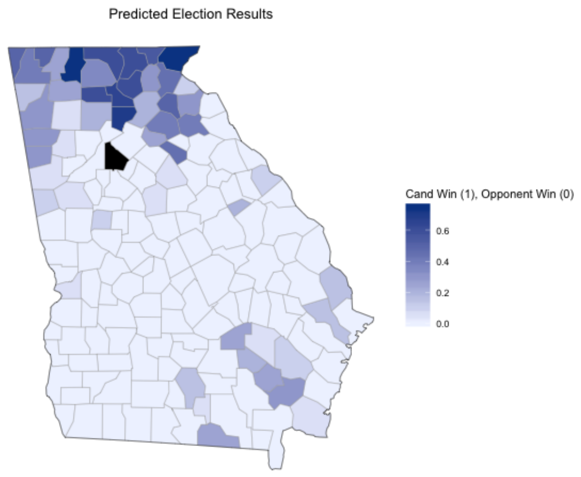


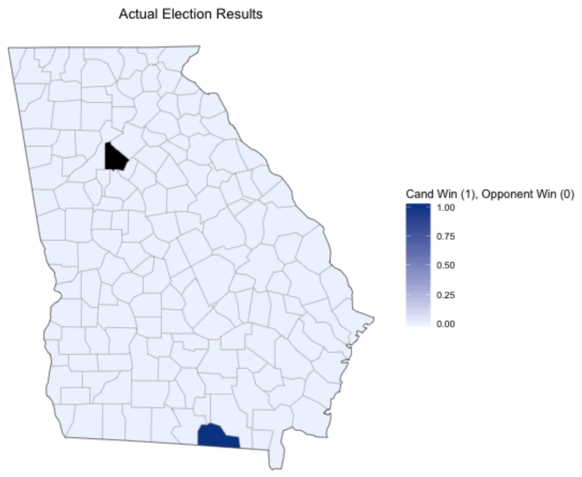
7.3 Sanders v Clinton Plots

The model somewhat works for Democratic candidates. Here are the predictions for Sanders:









7.4 KNN Analysis

The model works on a macro scale for each state, correctly predicting the majority of counties. It does well for homogeneous states, or states with clear demographic boundaries. A homogenous example is Sander's overwhelming loss in Georgia, since the counties are small and similar to each other and other Southern Clinton wins. Illinois has high variance between Chicago and other counties, and Northern states split more evenly between Sanders and Clinton, so the neighbors were split as well, leading to poorer performance.

Some states, like Arizona, Alabama, and Florida, are complete Trump wins. The model does somewhat account for Trump's wins, but it doesn't capture the totality of his wins. The 2016 election is unusual with a rogue candidate, so traditional metrics like demography do not perform as expected in these overwhelming cases. States have many different styles of counties, so it's hard for a model to predict a 100% Trump win.

However, KNN performs better in nuanced and battleground states – in particular in Idaho. Idaho's central regions voted for Trump and the periphery voted for Cruz, and the KNN model accurately captures this pattern. This is a big success for the model – we predicted the central areas to have Trump tendencies, and they did. The model does a good job because the state had well-known demographic demarcations and was not as affected by Trump hysteria.

Overall, the model works well when the training data is relatively homogenous. If a candidate has clearly won similar states, the model correctly predicts that they will win future similar states. In addition, if there are clear boundaries within a state, the model will pick up on that fact. However, it does poorly with overwhelming and unusual results, which unfortunately are common in this election.

The results confirm that a candidate can expect demographic data to be a good proxy for prediction, but the data do not make perfect predictions. The best performance is for historically strongly voting counties. Novel counties do not perform well with KNN analysis. A candidate should run KNN and canvas in doubtful areas, but not rely completely on the model. We imagine that a candidate will have a good idea of his or her performance, but wants to fine tune their strategy in battleground states like Idaho. The regression-tuned KNN model will be useful in these scenarios.

8 Conclusion - Answering the Questions

We concluded that using even one feature could give us somewhat accurate results. We made predictions solely based on the race and household population demographics of a county. As seen in these side-by-side comparisons of our predicted election results based on a sole feature, we could obtain results that looked fairly close to the actual results. There were some counties that were vastly different, but that is expected in using only a single feature. To answer the question we initially asked, as a whole, **yes, we can conclude that race and demographic data can be used as a predictor of voting patterns.** In addition, we can use this data to predict which counties are swing counties and if we were a team working to support a specific candidate, we would invest more time and money into these counties to swing things in our favor. Although the model performs poorly with unusual scenarios (Trump), it can be useful in hard fought states.

This project could be expanded further by using multiple features, and seeing how well we could predict using only five or ten features. Due to limited time and resources, we werent able to do it here. Random forests (with decision stumps) is well suited to the data as well, but takes an even longer time to train. Future analysis could include a random forest algorithm running on a more powerful computer.

9 Code Appendix

Code Structure

We went with an abstraction approach, modularizing parts for easy reuse. `util.R` cleans and organizes the data by joining the county data and candidate data tables, then renaming columns for human-parsable form, and attaching additional labels. This script is imported in every other file, so it was easy to change data formats because of a singular dependency. `visualizations.R` used `util.R` to make the exploratory data analysis plots. We implemented from scratch KNN in `knn.R` and exposed `knn_pred()` with multiple arguments for other files to use. Finally, `map.R` runs the KNN in a mapping function, which can be easily switched to any state in the dataset. The flexibility in the code allows for quick changes to output without major refactoring, so this code is usable to other teams interested in the same topic. RDS files were used to avoid redundant computations.

9.1 util.R - Utility Functions and Data Cleaning

```
############
## Util ####
############

# Use source("util.R") to import these methods.

library("dplyr")

county_features <- read.csv("data/county_facts_dictionary.csv")
county_data <- read.csv("data/county_facts.csv")
results <- read.csv("data/primary_results.csv")

data_with_results <- left_join(results, county_data, by = c("fips" = "
    fips"))

trump <- filter(data_with_results, candidate == "Donald Trump")
sanders <- filter(data_with_results, candidate == "Bernie Sanders")
clinton <- filter(data_with_results, candidate == "Hillary Clinton")
cruz <- filter(data_with_results, candidate == "Ted Cruz")

trump_nums <- subset(trump, select = - c(state:candidate, fraction_
    votes:state_abbreviation.y))
sanders_nums <- subset(sanders, select = - c(state:candidate, fraction_
    votes:state_abbreviation.y))
clinton_nums <- subset(clinton, select = - c(state:candidate, fraction_
    votes:state_abbreviation.y))
cruz_nums <- subset(cruz, select = - c(state:candidate, fraction_votes:
    state_abbreviation.y))

i <- match(names(clinton_nums), county_features$column_name)
names(trump_nums) <- county_features$description[i]
names(trump_nums)[[1]] <- "votes"
names(clinton_nums) <- county_features$description[i]
names(clinton_nums)[[1]] <- "votes"
names(cruz_nums) <- county_features$description[i]
names(cruz_nums)[[1]] <- "votes"
names(sanders_nums) <- county_features$description[i]
names(sanders_nums)[[1]] <- "votes"

trump_model <- lm(votes ~ ., trump_nums)
sanders_model <- lm(votes ~ ., sanders_nums)
clinton_model <- lm(votes ~ ., clinton_nums)
cruz_model <- lm(votes ~ ., cruz_nums)

#Example: Top weighted features for Sanders:
head(sort(abs(sanders_model$coefficients), decreasing = T), 10)
#Least important features:
head(sort(abs(sanders_model$coefficients), decreasing = F), 10)
```

9.2 visualizations.R - Exploratory Data Analysis

```r
library(DataComputing)
source("util.R")
#create scatter plots of white voter percentage vs proportion of votes
trump_winloss <- trump_nums
trump_names <- names(trump_winloss)
trump_names[7] = "Young.Percentage"
trump_names[10] = "White.Percentage"
trump_names[22] = "College.Percentage"
trump_names[30] = "Persons.Per.Household"
names(trump_winloss) <- trump_names
trump_winloss$won <- as.logical(trump$votes >cruz$votes)
trump_winloss$fraction_votes <- trump$fraction_votes

Trump.White.Profile <- ggplot(trump_winloss, aes(x=White.Percentage, y=
    fraction_votes, col=won)) +
  geom_point() +
  labs(title = "Trump Voter Profile",
       x = "Percent White",
       y = "Proportion of Votes")

cruz_winloss <- cruz_nums
cruz_names <- names(cruz_winloss)
cruz_names[7] = "Young.Percentage"
cruz_names[10] = "White.Percentage"
cruz_names[22] = "College.Percentage"
cruz_names[30] = "Persons.Per.Household"
names(cruz_winloss) <- cruz_names
cruz_winloss$won <- as.logical(cruz$votes >trump$votes)
cruz_winloss$fraction_votes <- cruz$fraction_votes
Cruz.White.Profile <- ggplot(cruz_winloss, aes(x=White.Percentage, y=
    fraction_votes, col=won)) +
  geom_point() +
  labs(title = "Cruz Voter Profile",
       x = "Percent White",
       y = "Proportion of Votes")

sanders_winloss <- sanders_nums
sanders_names <- names(sanders_winloss)
sanders_names[7] = "Young.Percentage"
sanders_names[10] = "White.Percentage"
sanders_names[22] = "College.Percentage"
sanders_names[30] = "Persons.Per.Household"
names(sanders_winloss) <- sanders_names
sanders_winloss$won <- as.logical(sanders$votes > clinton$votes)
sanders_winloss$fraction_votes <- sanders$fraction_votes
```

```
Sanders.White.Profile <- ggplot(sanders_winloss, aes(x=White.Percentage
    , y=fraction_votes, col=won)) +
  geom_point() +
  labs(title = "Sanders Voter Profile",
       x = "Percent White",
       y = "Proportion of Votes")

clinton_winloss <- clinton_nums
clinton_names <- names(clinton_winloss)
clinton_names[7] = "Young.Percentage"
clinton_names[10] = "White.Percentage"
clinton_names[22] = "College.Percentage"
clinton_names[30] = "Persons.Per.Household"
names(clinton_winloss) <- clinton_names
clinton_winloss$won <- as.logical(clinton$votes > sanders$votes)
clinton_winloss$fraction_votes <- clinton$fraction_votes
Clinton.White.Profile <- ggplot(clinton_winloss, aes(x=White.Percentage
    , y=fraction_votes, col=won)) +
  geom_point() +
  labs(title = "Clinton Voter Profile",
       x = "Percent White",
       y = "Proportion of Votes")

Trump.White.Profile
Cruz.White.Profile
Clinton.White.Profile
Sanders.White.Profile

# histogram of people per household, facetted by win/loss
Trump.Num.People.Profile <- ggplot(trump_winloss, aes(x = Persons.Per.
    Household, y=fraction_votes, col=won)) +
  geom_point() +
  labs(title = "Trump - Voter Household Profile",
       x = "Number of People per Household",
       y = "Count")

Cruz.Num.People.Profile <- ggplot(cruz_winloss, aes(x = Persons.Per.
    Household, y=fraction_votes, col=won)) +
  geom_point() +
  labs(title = "Cruz - Voter Household Profile",
       x = "Number of People per Household",
       y = "Count")

Sanders.Num.People.Profile <- ggplot(sanders_winloss, aes(x = Persons.
    Per.Household,y=fraction_votes, col=won)) +
  geom_point() +
  labs(title = "Sanders - Voter Household Profile",
       x = "Number of People per Household",
```

```r
        y = "Count")

Clinton.Num.People.Profile <- ggplot(clinton_winloss, aes(x = Persons.
    Per.Household,y=fraction_votes, col=won)) +
  geom_point() +
  labs(title = "Clinton - Voter Household Profile",
        x = "Number of People per Household",
        y = "Count")

Trump.Num.People.Profile
Cruz.Num.People.Profile
Sanders.Num.People.Profile
Clinton.Num.People.Profile

# show initial county turnouts (who is winning)
Republican.Pie.Chart <- ggplot(trump_winloss, aes(x = factor(1), fill=
    won)) +
  geom_bar(width=1) +
  coord_polar(theta="y") +
  scale_fill_discrete(name="Candidate",
                          breaks=c("FALSE", "TRUE"),
                          labels=c("Cruz", "Trump")) +
  labs(title = "Republican Primary Race Results")

Democrat.Pie.Chart <- ggplot(clinton_winloss, aes(x = factor(1), fill=
    won)) +
  geom_bar(width=1) +
  coord_polar(theta="y") +
  scale_fill_discrete(name="Candidate",
                          breaks=c("FALSE", "TRUE"),
                          labels=c("Sanders", "Clinton")) +
  labs(title = "Democrat Primary Race Results")


Republican.Pie.Chart
Democrat.Pie.Chart

#Plot of age voter profile
Trump.Age.Profile <- ggplot(trump_winloss, aes(x=Young.Percentage, y=
    fraction_votes, col=won)) +
  geom_point() +
  labs(title = "Trump Age Profile",
        x = "Percent 18 and Under",
        y = "Proportion of Votes")
Trump.Age.Profile

Cruz.Age.Profile <- ggplot(cruz_winloss, aes(x=Young.Percentage, y=
    fraction_votes, col=won)) +
  geom_point() +
  labs(title = "Cruz Age Profile",
        x = "Percent 18 and Under",
        y = "Proportion of Votes")
Cruz.Age.Profile

Sanders.Age.Profile <- ggplot(sanders_winloss, aes(x=Young.Percentage,
    y=fraction_votes, col=won)) +
  geom_point() +
  labs(title = "Sanders Age Profile",
        x = "Percent 18 and Under",
        y = "Proportion of Votes")
Sanders.Age.Profile

Clinton.Age.Profile <- ggplot(clinton_winloss, aes(x=Young.Percentage,
    y=fraction_votes, col=won)) +
  geom_point() +
  labs(title = "Clinton Age Profile",
        x = "Percent 18 and Under",
        y = "Proportion of Votes")
Clinton.Age.Profile
```

9.3 knn.R - From scratch KNN Implementation (using apply!)

```
##########################
### KNN Implemntation ###
##########################

source("util.R")

#Cleaning some more, with labels for win/losses relative to each other.
#Assumes two people races
trump_wins <- as.numeric(trump_nums$votes - cruz_nums$votes > 0)
sanders_wins <- as.numeric(sanders_nums$votes - clinton_nums$votes > 0)

# DEPRECATED out <- knn(subset(train, select =-c(votes)), subset(test,
    select =-c(votes)), train$votes, k = 2)

#Rewriting K Nearest Neighbors
calculate_distance <- function(vec1, vec2){
  #Calculates Euclidian distance between two vectors
  return(sqrt(sum((vec1 - vec2)^2)))
}
knn <- function(cand, data, num = 5){
  #Returns num nearest neighbors to cand in dataset
  distances <- apply(data, 1, calculate_distance, cand)
  return(sort.int(distances, index.return = T)$ix[1:num])
}

#Packaging it up
knn_pred <- function(cand_nums, cand_wins, cand, state = "Idaho", k=20,
    features = c(30, 10, 13, 11, 16)){
  #Returns proportion of KNN with that label for that state (
      predictions)
  ##USAGE EXAMPLE## : test <- knn_pred(trump_nums, trump_wins, trump, k
      =20)
  cand_cut <- cand_nums[features]
  cand_idaho_train <- cand_cut[which(cand$state == state),]
  cand_idaho_labels <- cand_wins[which(cand$state == state)]
  cand_train <- cand_cut[which(cand$state != state), ]
  cand_train_labels <- cand_wins[which(cand$state != state)]

  cand_confidence <- numeric()

  for (i in 1:nrow(cand_idaho_train)){
    cand_confidence[[i]] <- mean(cand_train_labels[knn(cand_idaho_train
        [i, ], cand_train, k)[1:k]])
  }
  correct <- 1 - sum(abs(cand_idaho_labels - as.numeric(cand_confidence
      > 0.5)))/length(cand_confidence)
  if (correct < 0.5){
    correct <- 1 - correct # :) binary prediction shortcut
  }
  print(paste("Correct rate is ,", correct))
  df <- data.frame(indexes = which(cand$state == state)[1:length(cand_
      confidence)], confidence = cand_confidence)
  return(df)
}
```

9.4 map.R - Running KNN and Mapping Results

```
#install.packages("maps")
#install.packages("ggplot2")
library(ggplot2)
library(choroplethr)
library(mapproj)

source("knn.R")
map_data <- function(cand_nums, cand_wins, cand, opponent, k = 20,
    state = "idaho"){

  State <- paste(toupper(substr(state, 1, 1)), substr(state, 2, nchar(
     state)), sep = "")

  #cand_confidence <- knn_pred(cand_nums, cand_wins, cand, State, k=k)
  #saveRDS(cand_confidence, paste(state, ".rds", sep=""))
  cand_confidence <- readRDS(paste(state, ".rds", sep=""))

  cand_confidence$value = cand_confidence$confidence
  cand_confidence$region <- cand[which(cand$state == State),]$fips
  print(State)


  cand_idaho_labels <- as.numeric(cand$votes - opponent$votes > 0)[
     which(cand$state == State)]
  true_values <- data.frame(value = cand_idaho_labels, region = cand_
     confidence$region)
  a <- county_choropleth(cand_confidence,
                        state_zoom = state,
                        title      = "Predicted Election Results",
                        legend = c("Cand Win (1), Opponent Win (0)"),
                        num_colors = 1)
  print(true_values$value)
  b <- county_choropleth(true_values,
                        state_zoom = state,
                        title      = "Actual Election Results",
                        legend = c("Cand Win (1), Opponent Win (0)"),
                        num_colors = 1)

  plot(a)
  plot(b)
}
map_data(trump_nums, trump_wins, trump, cruz,  k = 20, state = "idaho")
```