Pstat 131 Project: Election Analysis

Echo Zheng Candice Yu Anthony Yang

Introduction

Making predictions of the 2016 voting results is difficult. Variation between polling results from a number of factors, and those variation makes it hard to predict voters' behavior. First off, taking random samples leads to sampling error. For example, the sample that has more Obama supporters than that is represented in the general population. Second, some voters do not answer truthfully as they may lie about who they will vote for or lie about whether they will vote. Also, there are non-response bias when the non-responders differ in a meaningful way to the responders.

Although there are numerous uncertainties, we can still employing various modeling methods to find out the underlying logics and patterns of the poll in order to make our prediction of election as accurate as possible. There are modeling methods like logistic regression, decision tree, and random forest, and each with its own advantages and disadvantages which we are going to explore in this report. Besides learning the patterns of the polls and making predictions, it is also worth exploring what factors are significant in driving the variations of the poll. What are the features of the county population and how or to what extent are these features related to the voting results?

Therefore, in this project, there are 2 main tasks. The first one is predicting the winning candidate between Hillary Clinton and Donald Trump in each county. We will basically use three models–logistic regression model, decision tree, and random forest–to make simulations and generate prediction maps for county winner. Then we find that random forest might be the most powerful model based on its prediction accuracy.

The second task is evaluating the modeling methods above through their misclassification errors. We will also analyzing the advantages and disadvantages among each model. After that, we will focus on finding out variables that are predictive of the election outcome based on the random forest model and logistic regression model. Then,we will compute k-means clusters and the plot of centroid coordinates against variable to get a direct impressions about the relationship between these geographic heterogeneity and the voting results. The variables we get are mostly related to racial issues and transit, so at last we will provide our own interpretations and insights about that.

Materials and methods

Datasets/Material:

The dataset used for this analysis is formed from a combination of the tract-level 2010 census data containing regional census informations and the county-level voting tallies from the 2016 election containing regional voting results. Both raw datasets needed to be cleaned first; we firstly removed duplications in the election data and extracted only county-level tallies. Since the raw census data contains informations more fine-grained than county-level, we had to aggregate the data to county-level. To do this, we cleaned up the data and removed variables that are highly correlated; then, we weighted the variables by population, and lastly aggregated the data to county-level.

After we prepared these two datasets, we kept the top two candidates by county in the election dataset, unifies the text-transform, and finally merged these two datasets together through left_joining by state and county. In this way, it combines the vote information for the winning candidate and runner-up in each county within the census data. This merged county-level dataset is named "merged_data" and this is the main dataset we are woring with for this analysis. It includes informations about regions, top two candidates, poll result, racial composition of the county population, poverty levels, and income levels, etc. The table below demonstrates the first few rows and columns of the dataset "merged_data":

county	fips	candidate	state	votes	total	pct	CensusTract
autauga	1001	Donald Trump	alabama	18172	24759	0.734	35312420
autauga	1001	Hillary Clinton	alabama	5936	24759	0.2398	35312420
baldwin	1003	Donald Trump	alabama	72883	94261	0.7732	19682790
baldwin	1003	Hillary Clinton	alabama	18458	94261	0.1958	19682790

Methods:

Task 1: Predicting the Winner of Each County

We are going to use three different models for prediction–logistic regression, decision tree, and randomforest. We will prepare the data by first partitioning it into a training set(80%) and a test set(20%). Then we are going to use the existing variable "pct" to create a new factor variable named "winnerThing" which consists of the levels "TrumpWin" and "ClintonWin" to represent the voting results. Specifically, model inputs include "winnerThing" and the census information for corresponding counties. We will firstly construct a decision tree with pruning by cost complexity on the training set and choose a optimal tree using the optimal tuning parameter. We would observe that the resulting tree has number of terminal nodes of 11 and variables actually used in tree construction are "White","Professional","Transit", etc. We will next find out the optimal threshold by constructing a ROC curve, and finally use it to estimate class probabilities and misclassification error rates on the test partition. Then, we will try a different model– logistic regression to model the probabilities that either major candidate win a county as follows:

$$\text{logit}\left[P\left(\text{winnerThing}\right)_i\right] = \beta_0 + \beta_1 \text{Women}_i + \cdots + \beta_{10} \text{White}_i \ , \quad \text{county } i = 1, \ldots, 2148$$

We can then use it to compute predicted probabilities on test partition. After that, we will try the random forest model and construct: randomForest(formula = winnerThing ~ ., data = train, ntree = 100, mtry = 5). Through calculate predictions on the test partition and cross-classify with the true labels, we can learn the accuracy of the model. When we finished building each models, we are going to make a map for each one to show the winner of each county through merging the winner to each county so that we can learn how these predictions work finally.

Task 2: Evaluating the Modeling Methods and Identify Variables that Were Predictive of the Election Outcome

Based on task 1, we are going to further evaluate those three models by their misclassification error, comparing with the real voting result, and discover their advantages and disadvantages from aspects of their interpretability and efficiency. Then we will determine which model fits the best in this case, which would be random forest. In addition, we are going to find significant variables that are predictive of the outcome based on the variable importance measures from the random forest model. Specifically, the variables would be arranged based on their mean decrease accuracy and the top three would be chosen as the most significant indicators of the election. Also, there would be compare and contrast between other important variables chosen from logistic regression model and decision tree. In these ways, we are able to visualize these factors and exploring their level of significance. In addition, we will futher understand the geographic heterogeneity of these variables among counties by K-means clustering–choosing K, ploting centroid coordinates against variables, and projecting k-means clusters onto the first two principal components for the continued analysis.

Procedure

Task 1: Predict the winner of each county

Step 1: Add the indicator variable

We will add the indicator variable named "winnerThing" indicating whether "TrumpWin" or "ClintonWin" to the dataset. The table below shows the major columns of the datasets to verify whether the indicator variable works properly.

candidate	pct	winnerThing
Donald Trump	0.734	TrumpWin
Hillary Clinton	0.2398	TrumpWin

candidate	pct	winnerThing
Donald Trump	0.7732	TrumpWin
Hillary Clinton	0.1958	TrumpWin

Step 2: Fit a decision tree to predict "winnerThing".

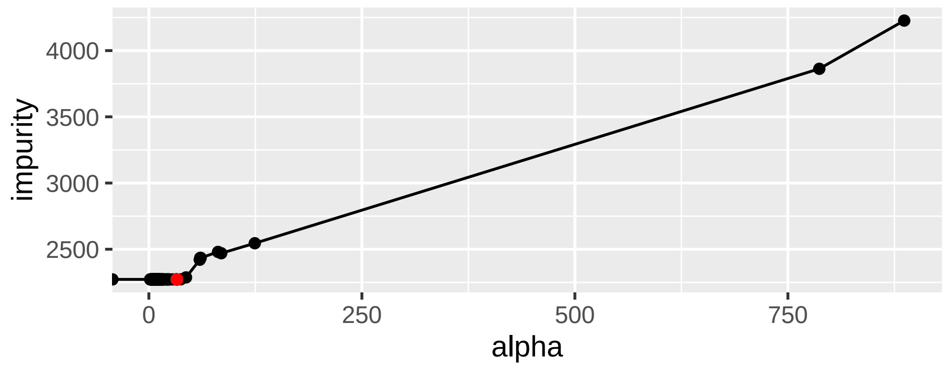
A large decision tree is grown and the summary of the model is presented.

- **call**: `tree(formula = winnerThing ~ ., data = train, na.action = na.pass,      control = tree_opts, split = "deviance")`
- **type**:

  Classification tree:
- **size**: *188*
- **df**: *4726*
- **dev**: *371.3*
- **misclass**: *125* and *4914*

Step 3: Cost-complexity pruning.

Select the optimal tuning parameter via 8-fold cross validation based on the graph below; Prune using the optimal tuning parameter. The best alpha we choose is 32.79772. The summary and a table of the number of splits on each variable is presented. The outputted tree will be displayed in the results section



- **call**: `snip.tree(tree = t_0, nodes = c(36L, 19L, 37L, 8L, 26L, 31L,  12L, 5L, 14L, 30L, 27L))`
- **type**:

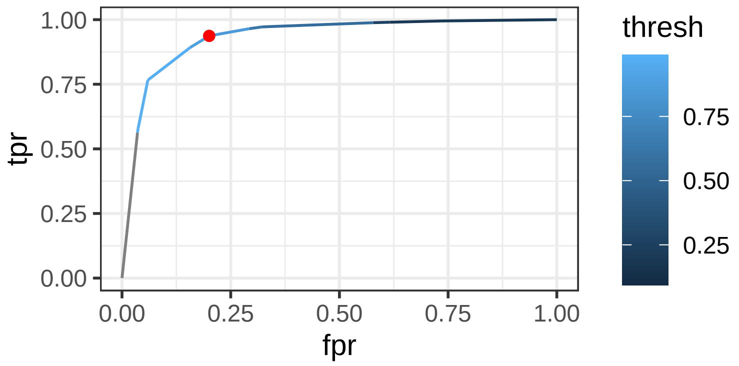
  Classification tree:
- **used**: *3*, *2*, *17*, *9*, *14* and *8*
- **size**: *11*
- **df**: *4903*
- **dev**: *2001*
- **misclass**: *361* and *4914*

var	n
	11
White	3
Professional	2
Transit	2
Women	1

Step 4: Roc curve.

From the estimated probabilities at each leaf node, we'll construct an ROC curve and determine whether the probability threshold should be adjusted away from 0.5. Below are the result of the optimal threshold and the plot of TPR against FPR. As the result shows, the optimal threshold should be large higher than 0.5.

fpr	tpr	thresh	youden
0.2005	0.9372	0.8493	0.7367



Step 5: Estimate class probabilities on the test partition

We then calculate the Misclassification errors; the outputs will be shown in the results section.

Step 6: Map the prediction results

We then plot a map of the election winner by county based on decision tree prediction model, the outputted map will be presented in the results section.

Step 7: Fit a logistic regression model to the data

We then fit the logistic regression model to our data and calculate the Misclassification errors; the outputs will be shown in the results section.

Step 8: Map the prediction results

We then plot a map of the election winner by county based on logistic regression model, the outputted map will be presented in the results section.

Step 9: Random Forest

We then fit a randomforest model to our dataset, the summary of the randomforest model will be presented in the results section.

Step 10: Calculate predictions on the test partition and cross-classify with the true labels

We then calculate the Misclassification errors of random forest model; the outputs will be shown in the results section.

Step 11: Map the prediction results

A map of the election winner by county based on random forest model We then plot a map of the election winner by county based on random forest model, the outputted map will be presented in the results section.

Task 2: Evaluating the Modeling Methods and Identify Variables that Were Predictive of the Election Outcome

Step 1: Compute variable importance scores

We will firstly compute MeanDecreaseAccuracy and constructing a plot of importance measures. The outputted plot of importance measures will be presented in the results section.

	MeanDecreaseAccuracy
White	0.0595
Minority	0.04007
Transit	0.03898
Professional	0.02315
Employed	0.02033
Unemployment	0.0188

Step 2: Find out level of correlation of indicators to response variables based on the coefficient of logistic regression model.

We then calculated coefficients of logistic regression model. The output will be presented in the results section

Step 3: Clustering with k-means:

We then Compute k-means clusters with $K = 2$ and plot centroid coordinates against variable. The outputted plot of centroid coordinates against variable will be presented in the results section.

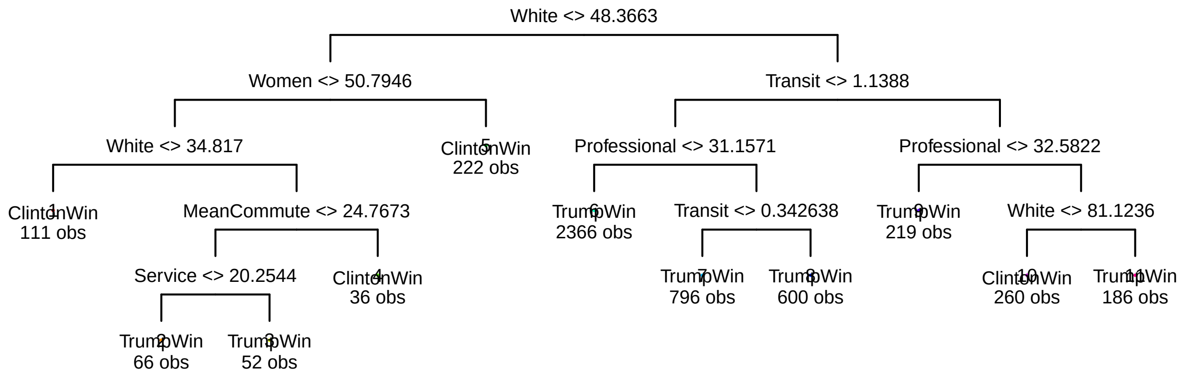
We then project the k-means cluster onto the first two principal components and plot the data together with the cluster centers. We display the winning result using the color aesthetic, and the shape aesthetic shows the cluster assignment. The plot will be presented in the results section

Results

Task 1: Predict the Winner of Each County

For decision tree model:

The decision tree has 11 leaf nodes; Variable "White", "Professional", and "Transit" are split on most often throughout the tree.



For logistic regression model:

The summary of logistic regression model is show below.

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	13.06	5.766	2.265	0.02349
Women	-0.0135	0.03251	-0.4154	0.6779
White	0.1677	0.04358	3.849	0.0001188

	Estimate	Std. Error	z value	Pr(>|z|)
Citizen	-0.1192	0.01565	-7.614	2.649e-14
IncomePerCap	-3.051e-05	2.482e-05	-1.229	0.2189
Poverty	-0.01606	0.02636	-0.6091	0.5425
ChildPoverty	0.009518	0.01699	0.5603	0.5753
Professional	-0.2386	0.02413	-9.886	4.765e-23
Service	-0.3281	0.03214	-10.21	1.81e-24
Office	-0.09556	0.0296	-3.228	0.001246
Production	-0.1576	0.02766	-5.696	1.229e-08
Drive	0.2166	0.03102	6.983	2.886e-12
Carpool	0.2197	0.04068	5.401	6.638e-08
Transit	-0.0114	0.06114	-0.1865	0.8521
OtherTransp	0.07423	0.06286	1.181	0.2376
WorkAtHome	0.1271	0.04854	2.618	0.008853
MeanCommute	-0.008805	0.01515	-0.5811	0.5612
Employed	-0.1606	0.0213	-7.539	4.726e-14
PrivateWork	-0.09523	0.01416	-6.725	1.757e-11
SelfEmployed	-0.00351	0.03207	-0.1095	0.9128
FamilyWork	0.2784	0.2214	1.258	0.2085
Unemployment	-0.172	0.02639	-6.516	7.215e-11
Minority	0.04453	0.04195	1.061	0.2885

(Dispersion parameter for binomial family taken to be 1)

Null deviance:	4226 on 4913 degrees of freedom
Residual deviance:	1879 on 4891 degrees of freedom

For random forest model:

The summary of random forest model is show below.

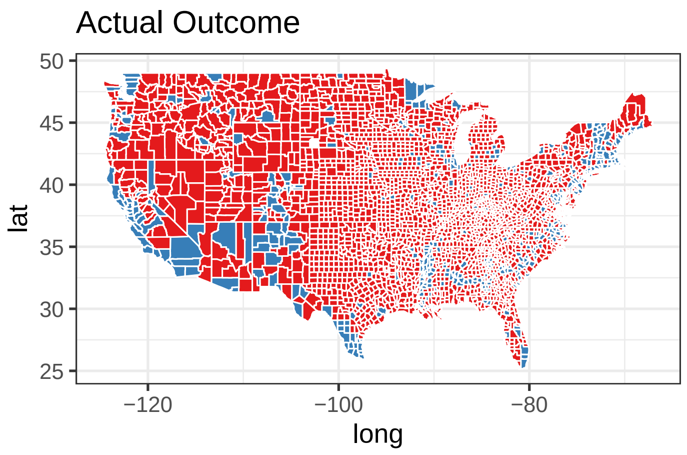
	Length	Class	Mode
call	6	-none-	call
type	1	-none-	character
predicted	4914	factor	numeric
err.rate	300	-none-	numeric
confusion	6	-none-	numeric
votes	9828	matrix	numeric
oob.times	4914	-none-	numeric
classes	2	-none-	character
importance	88	-none-	numeric
importanceSD	66	-none-	numeric
localImportance	0	-none-	NULL
proximity	0	-none-	NULL
ntree	1	-none-	numeric
mtry	1	-none-	numeric
forest	14	-none-	list
y	4914	factor	numeric
test	0	-none-	NULL
inbag	0	-none-	NULL
terms	3	terms	call

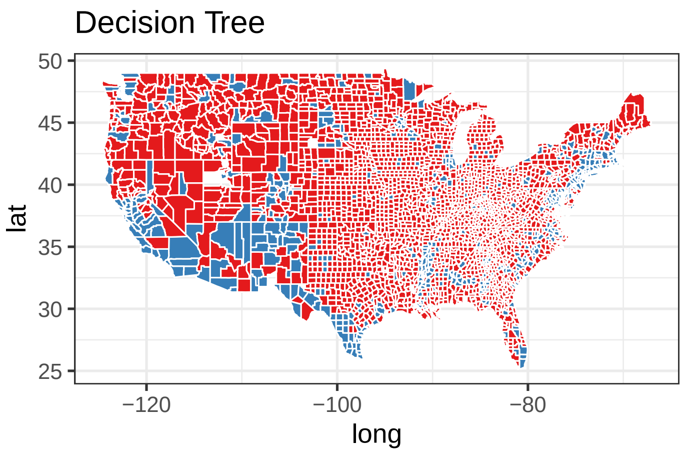
The figures below are the maps drawn based on the predictions of each models. The figure on the top left is the actual exit poll result.

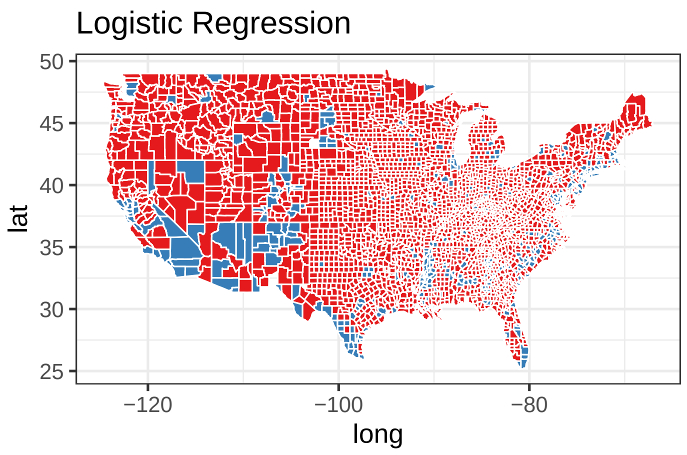
Based on the map, we can see that for the decision tree model, the prediction result that Clinton won more counties all across the map compared to what actually happened.

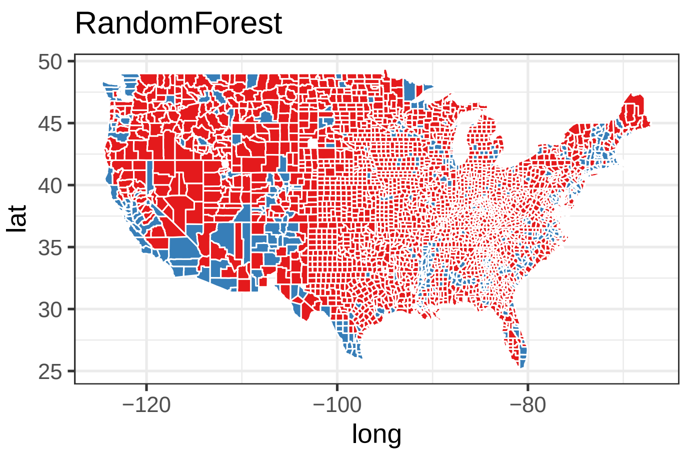
The map for the logistic regression models tell us that most counties of the whole U.S. vote for Trump except some of the southwest regions. Such prediction result is largely in consistency of the real voting result, which suggests the success of the model.

We can see from the map constructed from the randomforest model that there is a slight difference between the prediction results between logistic regression model and random forest model– a small number of counties in the northeast conner vote for Clinton instead of Trump, which better corresponds to the real situation. Thus, we would say that random forest model is the most successful prediction model among the three in this case based on the accuracy of the results.









Task 2: Evaluating the modeling methods and identify variables that were predictive of the election outcome

Evaluating the modeling methods

The statistics below shows the misclassification error for each of the three models. It is obvious that random forest model has the smallest misclassification error (0.05374593) and decision tree has the largest (0.08876221) among the three models. It offers an improvement over a single classification tree that there is a decrease in FPR of about 8.4% and a decrease in FNR of about 2.6%. Comparing with logistic regression model, it has only an increase in FPR about 0.8%, but a decrease in FNR about 13.2%. Therefore, connecting with the results obtained in task 1, we can conclude that the prediction generated by random forest model is most convincing and credible.

	ClintonWin	TrumpWin
ClintonWin	0.6947	0.3053
TrumpWin	0.02408	0.9759

Logistic Regression Misclassification error rate: 0.06758958

	ClintonWin	TrumpWin
ClintonWin	0.7421	0.2579
TrumpWin	0.0578	0.9422

Decision tree Misclassification error rate: 0.08876221

	ClintonWin	TrumpWin
ClintonWin	0.8263	0.1737
TrumpWin	0.03179	0.9682

Randomforest Misclassification error rate: 0.05374593

Predictive variables of election outcome

The first graph below presents the mean decrease in misclassification rate across trees for the variables generated from the random forest model. It is obvious that variable "White", "Minority", and "Transit" are the three most important variables as they have higher than average MeanDecreaseAccuracy values which indicates variable importance measures. It is not exactly the same comparing with the set of variables that figured most often in splitting the data for the decision tree, which are "White", "Professional", and "Transit". Also, if we look at the coefficients of the variables from the logistic regression model, we'll notice that "FamilyWork", "Drive", and "Carpool" are the variables with highest coefficients, and thus the strongest corrrelation with the response variable. Considering the conclusion from task 1 that random forest model best fit in this case, we think that the three most significant variables are "White", "Minority", and "Transit".

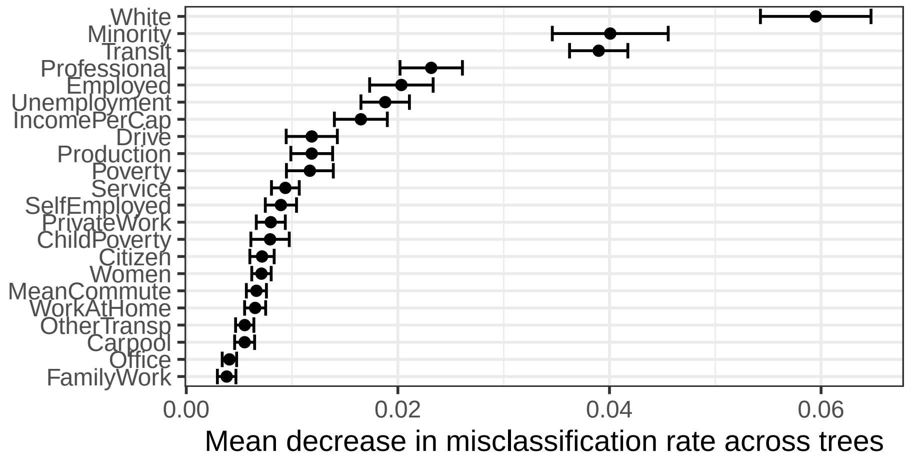


Table 12: Table continues below

(Intercept)	Women	White	Citizen	IncomePerCap	Poverty	ChildPoverty
470968	0.987	1.183	0.888	1	0.984	1.01

Table 13: Table continues below

Professional	Service	Office	Production	Drive	Carpool	Transit
0.788	0.72	0.909	0.854	1.242	1.246	0.989

Table 14: Table continues below

OtherTransp	WorkAtHome	MeanCommute	Employed	PrivateWork	SelfEmployed
1.077	1.135	0.991	0.852	0.909	0.996

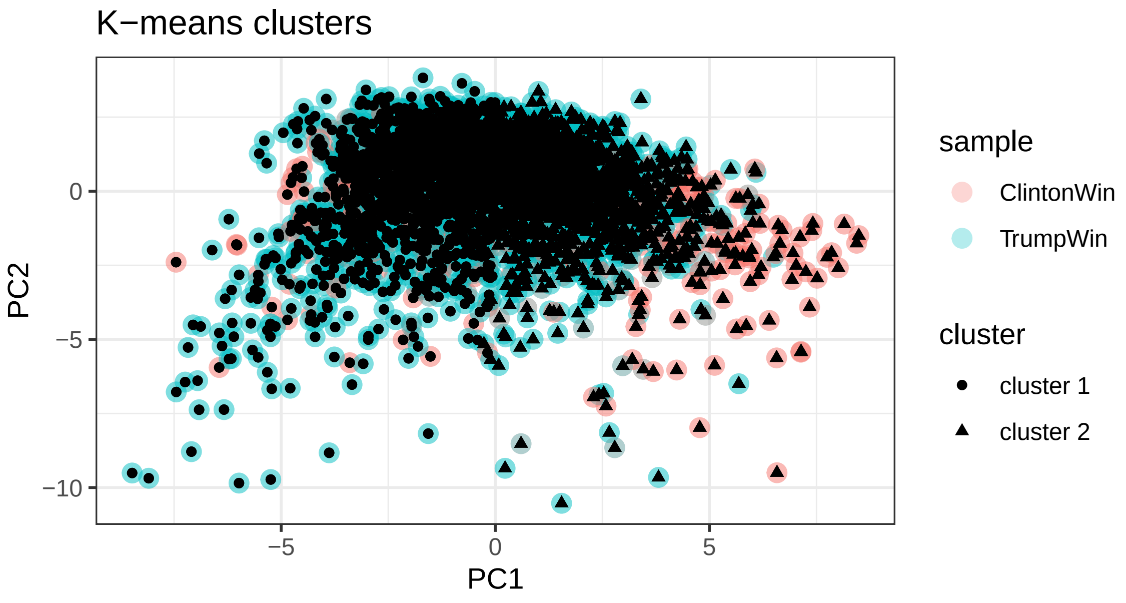
FamilyWork	Unemployment	Minority
1.321	0.842	1.046

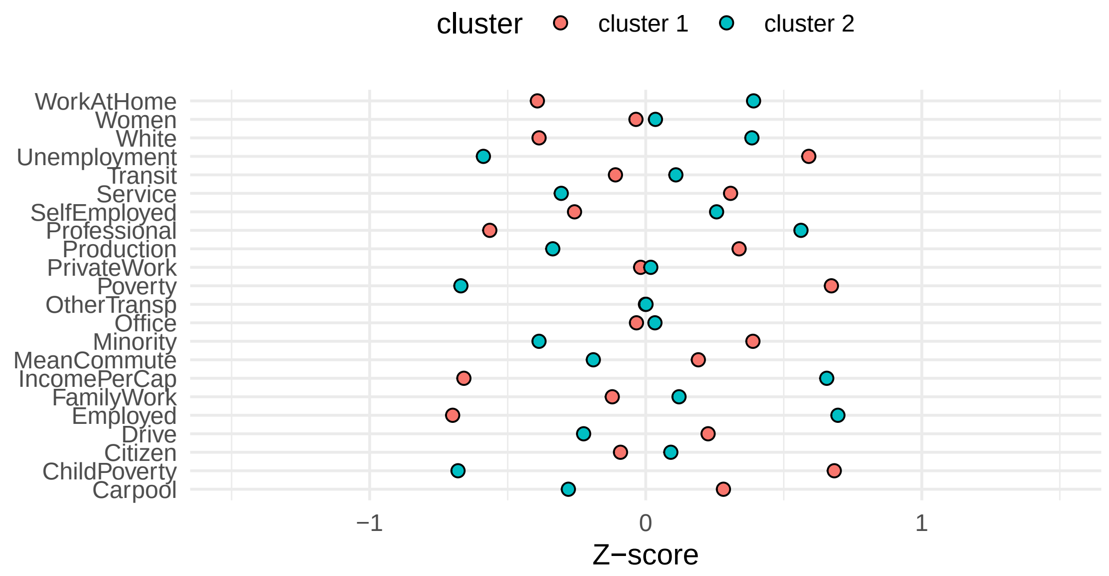
The figures below are obtained from k-means clustering, including the plot of centroid coordinates against variable and the projection of k-means clusters onto the first two principal components.

From the first plot, we learn that cluster 2 has the strongest association with the response "TrumpWin", and cluster 1 is mostly associated with "ClintonWin". Then we can look for some patterns in the plot of centroid finding out which variables are most associated with the voting results. As we can see, variable "WorkAtHome", "Women", "White", "PrivateWork", "SelfEmployed", "Transit", "Professional", "PrivateWork", "Office", "IncomePerCap", "FamilyWork", "Employed", and "citizen" are positively related to TrumpWin, while variable "Unemployment", "Service", "Production", "Poverty", "Minority", "MeanCommute", "Drive", "ChildPoverty", and "Carpool" are positively related to CLintonWin.

Generally, we can find the pattern that the population of the counties which voted for Trump consists of mostly White females, people who are self-employed or employed in private industries with offices, people who are professionals, and have higher than average income per capita with high employment rate. It seems that in general, the voters who voted for Clinton are mostly minorities who work more in service jobs, those commutes to work mostly via means such as carpool, and those in poverty. I would interpret this result as that wealthier and more well-off people, especially White, generally vote for Trump, and that people, especially Minorities, who are not as well-off or who works in lower end jobs such as service jobs generally vote for Clinton.

Therefore, this intepretation confirms the three influencial variables who chose previously, which are "White", "Minority", and "Transit". It makes sense that race and people's quality of living are the main contributing factors to the election results.





Discussion

We respectively used decision tree, logistic regression model, and random forest to build predictions on the voting results between Trump and Clinton based on the dataset "merged_data". According to model prediction accuracy, we have concluded that the randomforest model fits the best with the lowest misclassification error in this case compared to logistic regression model and decision tree.Also, based on the map visualization of county winner according to the prediction of randomforest model, we find that it strongly comforms to the actual voting results; so our model is considered to be successful. However, there are also disadvantages of it and advantages on other models. Although randomforest offers the highest accuracy, it can not plot the actual tree for us to examine which variables are used in splits. In comparison, decision tree model is easy to visualize and interpretable although it is sensitive to small changes in the data. Logistic regression also has high interpretability that it can explain association between predictors and the response variable and identify important varibales directly, while decision tree model is weaker in such aspects.

Based on the importance measures from the randomforest model, we also find out that the variables "White", "Minority", and "Transit" are the three most significant variables that influence the voting results. Moreover, the plot of centroid coordinates against variables and k-means clustering suggests that people with high social status, generally White, mostly vote for Trump while people in lower social strata mostly vote for Clinton, which mostly corresponds to what actually happened. Trump's agendas in 2016 mostly appealed to those who are generally more higher up, such as cutting taxes at all levels and strengthening immigration laws. The agendas of Clinton in 2016 mostly appealed to those who are not as well off and most minorities, such as supporting immigration reforms, and increase taxes for the most well off. One unexpected findings from our result is that our model suggests that female voters are more likely to vote for Trump instead of Clinton, while what actually happened was that more female voters voted for Clinton. It might be due to some sampling errors occured in our datasets or modeling bias, so we may continue to improve our models or looking for other prediction moethod that can lower the influence of sampling bias as much as possible.

Codes

```r
knitr::opts_chunk$set(echo = F,
                      message = F,
                      warning = F,
                      fig.align = 'center',
                      fig.height = 4,
                      fig.width = 4)

getwd()
library(pander)
library(tidyverse)
library(ggmap)
library(tree)
library(maptree)
#install.packages('randomForest')
library(randomForest)
library(modelr)
#install.packages('ROCR')
library(ROCR)
library(tidyverse)
#install.packages('gbm')
library(gbm)
library(pander)
library(dplyr)
#install.packages('NbClust')
library(NbClust)
load('data/project_data.RData')
set.seed(123)
# filter out fips == 2000
election_raw <- election_raw %>%
  filter(fips != 2000)
# create one dataframe per observational unit
election_federal <- filter(election_raw, fips == 'US')

election_state <- filter(election_raw, is.na(county), fips != 'US', fips != 46102)

electionbackup <- election_raw
electionbackup$fips <- as.numeric(electionbackup$fips)
election <- filter(electionbackup, fips >= 0)
# clean census data
census_clean <- census %>%
  drop_na() %>%
  mutate(Men = Men/TotalPop*100, Women = Women/TotalPop*100, Employed = Employed/TotalPop*100,
         Citizen = Citizen/TotalPop*100) %>%
  select(-Men) %>%
  group_by(CensusTract) %>%
  mutate(Minority = sum(Hispanic, Black, Native, Asian, Pacific)) %>%
  select(-c(Hispanic, Black, Native, Asian, Pacific)) %>%
  select(-c(Income, Walk, PublicWork, Construction)) %>%
  select(-ends_with('Err'))

# compute population-weighted quantitative variables
# census_clean_weighted <- census_clean %>%
#   group_by(State, County) %>%
#   add_tally() %>%
#   mutate(CountyPop = sum(TotalPop)) %>%
#   mutate(pop_wt = TotalPop/CountyPop) %>%
#   mutate(across(c(TotalPop:CountyPop, -n), ~ .x*pop_wt)) %>%
#   ungroup() %>%
#   select(-c(n, CountyPop, pop_wt, TotalPop))

census_clean_weighted <- census_clean %>%
  group_by(State, County) %>%
  add_tally(TotalPop, name = 'CountyPop') %>%
  mutate(pop_wt = TotalPop/CountyPop) %>%
  mutate(across(where(is.numeric), ~ .x*pop_wt)) %>%
  ungroup() %>%
  select(-c(TotalPop, CountyPop, pop_wt))

# aggregate to county level
census_tidy <- census_clean_weighted %>%
  #select(-CensusTract) %>%
  group_by(State, County) %>%
  mutate(across(c(Women:Minority), sum)) %>%
  slice(1) %>%
  ungroup()

# clean up environment
rm(list = setdiff(ls(), c('election_federal', 'election_state', 'election', 'census_tidy')))
counties <- map_data("county")
fips <- maps::county.fips %>%
  separate(polyname, c('region', 'subregion'), sep = ',')
counties <- counties %>% left_join(fips)

# electionchange <- election
#
# electionchange$county <- substr(electionchange$county, start = 1,
#                                 stop = nchar(electionchange$county) - 7)
#
# name2abbcounty <- function(countyname){
#   ix <- match(countyname, tolower(electionchange$county))
#   out <- electionchange$fips[ix]
#   return(out)
# }
# county <- county %>%
#   mutate(fips = name2abbcounty(subregion))

# who won each county?
county_winner <- election %>% # this line depends on your results above!
  group_by(fips) %>%
  mutate(total = sum(votes),
         pct = votes/total) %>%
  slice_max(pct)

# merge winner with plotting boundaries and make map
map_true <- left_join(counties, county_winner) %>%
  ggplot() +
  geom_polygon(aes(x = long,
                   y = lat,
                   fill = candidate,
                   group = group),
               color = "white",
               size=0.3) +
  coord_fixed(1.3) +
  guides(fill=FALSE) +
  scale_fill_brewer(palette="Set1") +
  ggtitle("Actual Outcome") +
  theme(plot.margin=unit(c(1,1,1,1),"pt")) +
  theme_bw()
# define function to coerce state abbreviations to names
abb2name <- function(stateabb){
  ix <- match(stateabb, state.abb)
  out <- tolower(state.name[ix])
  return(out)
}

# top two candidates by county
toptwo <- election %>%
  group_by(fips) %>%
  mutate(total = sum(votes),
         pct = votes/total) %>%
  slice_max(pct, n = 2)

# create temporary dataframes with matching state/county information
tmpelection <- toptwo %>%
  ungroup %>%
  # coerce names to abbreviations
  mutate(state = abb2name(state)) %>%
  # everything lower case
  mutate(across(c(state, county), tolower)) %>%
  # remove county suffixes
  mutate(county = gsub(" county| columbia| city| parish",
                       "",
                       county))
tmpcensus <- census_tidy %>%
  # coerce state and county to lowercase
  mutate(across(c(State, County), tolower))

# merge
merged_data <- tmpelection %>%
  left_join(tmpcensus,
            by = c("state"="State", "county"="County")) %>%
  na.omit()

# clear temporary dataframes from environment
rm(list = c('tmpwinner', 'tmpcensus'))


# print first few rows
merged_data[1:4, 1:8] %>% pander()
# creating the indicator variable
merged_data <- merged_data %>%
  mutate(winnerThing = ifelse(candidate == 'Donald Trump' & pct > 0.5, 'TrumpWin',
                              ifelse(candidate == 'Donald Trump' & pct < 0.5, 'ClintonWin',
                                     ifelse(candidate == 'Hillary Clinton' & pct > 0.5,'ClintonWin',
                                            'TrumpWin'))))
merged_data$winnerThing <- as.factor(merged_data$winnerThing)


# check first few rows
merged_data %>% select(candidate, pct, winnerThing) %>% head(4) %>% pander()
```

```r
# Exclude the other substance
new_merged_data <- merged_data %>% select(-1:-8)

# partition the rest of data into 80% training set and 20% test set
new_merged_data <- resample_partition(new_merged_data, c(test = 0.2, train = 0.8))
train <- as_tibble(new_merged_data$train)
test <- as_tibble(new_merged_data$test)
# grow a large decision tree
nmin <- 2
tree_opts <- tree.control(nobs = nrow(train),
                          minsize = nmin,
                          mindev = exp(-8))
t_0 <- tree(winnerThing ~ ., data = train, control = tree_opts, split = 'deviance', na.action = na.pass)
#draw.tree(t_0, cex = 0.3, size = 0.25, digits = 2)

summary(t_0) %>% pander()

# examine nodes
#t_0$frame %>% select(1:4) %>% head(5)
# cost-complexity pruning
nfolds <- 8
cv_out <- cv.tree(t_0, K = nfolds)

# convert to tibble
cv_df <- tibble(alpha = cv_out$k,
                impurity = cv_out$dev,
                size = cv_out$size)

# choose optimal alpha
best_alpha <- slice_min(cv_df, impurity) %>%
  slice_min(size)

# plot impurity against tuning parameter
cv_df %>%
  ggplot(aes(alpha,impurity)) +
  geom_point() +
  geom_line() +
  geom_point(aes(best_alpha$alpha,best_alpha$impurity),col="red")

# select final tree
t_opt <- prune.tree(t_0, k = best_alpha$alpha)
#draw.tree(t_opt, cex = 0.4, size = 0.1, digits = 2)
summary(t_opt) %>% pander()
t_opt $frame %>% select(var) %>% group_by(var) %>% count() %>% arrange(desc(n)) %>% head(5) %>% pander()
# compute estimated probabilities for each observation in the training set
probs_train <- predict(t_opt, newdata = merged_data$train, type = 'vector')
train_label <- train %>% pull(winnerThing)

# create prediction object (ROCR)
topt_prediction = prediction(predictions = probs_train[, 2],
                             labels = train_label )

# create performance object (ROCR)
topt_perf <- performance(topt_prediction, 'tpr', 'fpr')

# find TPR, FPR
rate_topt <- tibble(fpr = topt_perf@x.values,
                   tpr = topt_perf@y.values,
                   thresh = topt_perf@alpha.values) %>%
          unnest(everything())

# Optimal probability threshold
rate_topt <-rate_topt %>% mutate(youden = tpr - fpr)
opt_thresh <- rate_topt %>%
  slice_max(youden)
opt_thresh %>%
  pander()

# plot the ROC curve
rate_topt %>%
  ggplot(aes(x = fpr, y = tpr)) +
  geom_path(aes(color = thresh), size = 0.5) +
  geom_point(data = opt_thresh, color = 'red') +
  theme_bw()

# class probabilities on test partition
pred_test <- predict(t_opt, newdata = test, type = 'vector')

# predicted class labels
y_hat <- factor(pred_test[ ,2] >= opt_thresh$thresh, ordered = F,
                labels = c('ClintonWin', 'TrumpWin'))

# misclassification error rates
test_class <- test %>% pull(winnerThing)
error_table <- table(class = test_class, pred = y_hat)
table_topt <- error_table/rowSums(error_table)
#table_topt %>%
  #pander()
#cat("Misclassification error rate:", mean(y_hat != test_class)) %>%
  #pander()
topt_mse <- mean(y_hat != test_class)

test_with_prediction = test %>%
  mutate(predicted = y_hat)

merged_data_for_pred <- merged_data %>% select(-1:-8)
pred_decision <- predict(t_opt, newdata = merged_data_for_pred, type = 'vector')
y_hat <- factor(pred_decision[ ,2] >= opt_thresh$thresh, ordered = F,
                labels = c('ClintonWin', 'TrumpWin'))

merged_data_with_pred_decision = merged_data %>%
  mutate(predicted = y_hat)

# plotting boundaries for US counties
library(ggmap)
counties <- map_data("county")
fips <- maps::county.fips %>%
  separate(polyname, c('region', 'subregion'), sep = ',')
counties <- counties %>% left_join(fips)

# who won each county?
county_winner <- merged_data_with_pred_decision %>% # this line depends on your results above!
  group_by(fips) %>%
  #mutate(checking = ifelse(candidate == 'Donald Trump' & predicted == 'ClintonWin')) %>%
  filter((candidate == 'Donald Trump' & predicted != 'ClintonWin') | (candidate == 'Hillary Clinton' & predi
  #filter(-(candidate == 'Hillary Clinton' & predicted == 'TrumpWin')) %>%
  #slice_max(pct)

# merge winner with plotting boundaries and make map
map_t <- left_join(counties, county_winner) %>%
  ggplot() +
  geom_polygon(aes(x = long,
                   y = lat,
                   fill = candidate,
                   group = group),
               color = "white",
               size=0.3) +
  coord_fixed(1.3) +
  guides(fill=FALSE) +
  scale_fill_brewer(palette="Set1") +
  ggtitle("Decision Tree") +
  theme(plot.margin=unit(c(1,1,1,1),"cm")) +
  theme_bw()
#map_t
# fit logistic regression model on training partition
fit_glm <- glm(winnerThing ~., family = 'binomial', data = train)
#coef(fit_glm) %>% exp() %>% round(3)

# compute predicted probabilities on test partition
p_hat_glm<- predict(fit_glm, test, type = 'response')
# convert to classes using probability threshold 0.5
y_hat_glm <- factor(p_hat_glm > 0.5, labels = c('ClintonWin', 'TrumpWin'))

# cross-tabulate with true labels
errors_glm <- table(class = test$winnerThing,
                    pred = y_hat_glm)

table_lg <- errors_glm / rowSums(errors_glm)
lg_mse <-mean(y_hat_glm != test$winnerThing)
#table_lg %>% pander()
#mean(y_hat_glm != test$winnerThing) %>%
  #pander()

# compute predicted probabilities on test partition
p_hat_glm<- predict(fit_glm, merged_data_for_pred, type = 'response')
# convert to classes using probability threshold 0.5
y_hat_glm <- factor(p_hat_glm > 0.5, labels = c('ClintonWin', 'TrumpWin'))

merged_data_with_pred_glm = merged_data %>%
  mutate(predicted = y_hat_glm)

# plotting boundaries for US counties
library(ggmap)
counties <- map_data("county")
fips <- maps::county.fips %>%
  separate(polyname, c('region', 'subregion'), sep = ',')
counties <- counties %>% left_join(fips)

# who won each county?
county_winner <- merged_data_with_pred_glm %>% # this line depends on your results above!
  group_by(fips) %>%
  #mutate(checking = ifelse(candidate == 'Donald Trump' & predicted == 'ClintonWin')) %>%
  filter((candidate == 'Donald Trump' & predicted != 'ClintonWin') | (candidate == 'Hillary Clinton' & predi

# merge winner with plotting boundaries and make map
map_lg <- left_join(counties, county_winner) %>%
  ggplot() +
  geom_polygon(aes(x = long,
                   y = lat,
                   fill = candidate,
                   group = group),
               color = "white",
               size=0.3) +
  coord_fixed(1.3) +
  guides(fill=FALSE) +
  scale_fill_brewer(palette="Set1") +
  ggtitle("Logistic Regression") +
  theme(plot.margin=unit(c(1,1,1,1),"cm")) +
  theme_bw()
#map_lg

# grow forest
train_rf <- randomForest(winnerThing ~., ntree = 100, mtry = 5,
                         data = train)

# show summary
#summary(train_rf) %>% pander()
# calculate predictions
pred_rf <- predict(train_rf, newdata = test, type = 'response')
test_class <- test %>% pull(winnerThing)

# compute misclassification errors
# print table
error_table_rf <- table(class = test_class, pred = pred_rf)
table_rf <- error_table_rf/rowSums(error_table_rf)
rf_mse <- mean(pred_rf != test_class)
#table_rf %>% pander()
#cat(mean(pred_rf != test_class))
# compute predicted probabilities on test partition
pred_rf <- predict(train_rf, newdata = merged_data_for_pred, type = 'response')

merged_data_with_pred_randomforest = merged_data %>%
  mutate(predicted = pred_rf)

# plotting boundaries for US counties
library(ggmap)
counties <- map_data("county")
fips <- maps::county.fips %>%
  separate(polyname, c('region', 'subregion'), sep = ',')
counties <- counties %>% left_join(fips)

# who won each county?
county_winner <- merged_data_with_pred_randomforest %>% # this line depends on your results above!
  group_by(fips) %>%
  #mutate(checking = ifelse(candidate == 'Donald Trump' & predicted == 'ClintonWin')) %>%
  filter((candidate == 'Donald Trump' & predicted != 'ClintonWin') | (candidate == 'Hillary Clinton' & predi
  #filter(-(candidate == 'Hillary Clinton' & predicted == 'TrumpWin')) %>%
  #slice_max(pct)
```

```
# merge winner with plotting boundaries and make map
map_rf <- left_join(counties, county_winner) %>%
  ggplot() +
  geom_polygon(aes(x = long,
                   y = lat,
                   fill = candidate,
                   group = group),
               color = "white",
               size=0.3) +
  coord_fixed(1.3) +
  guides(fill=FALSE) +
  scale_fill_brewer(palette="Set1") +
  ggtitle("RandomForest") +
  theme(plot.margin=unit(c(1,1,1,1),"cm")) +
  theme_bw()
#map_rf
# Construct randomforest with variable importance measures
train_rf <- randomForest(winnerThing ~., ntree = 100, mtry = 5,
                         data = train, importance = T)

# display the variable importance
train_rf$importance %>% as.data.frame() %>%
  select(MeanDecreaseAccuracy) %>%
  arrange(desc(MeanDecreaseAccuracy)) %>%
  head() %>%
  pander()

# compute importance measures
## construct plot: outputs variable importance measures in terms of classification accuracy (0-1 loss) and G
mda_plot <- train_rf$importance %>%
  as_tibble() %>%
  mutate(var = factor(rownames(train_rf$importance)),
         total_sd = train_rf$importanceSD[ , 3]) %>%
  rename(Total = MeanDecreaseAccuracy) %>%
  ggplot(aes(y = fct_reorder(var, Total), x = Total)) +
  geom_point() +
  geom_errorbarh(aes(xmin = Total - 2*total_sd, xmax = Total + 2*total_sd)) +
  theme_bw() +
  labs(x = 'Mean decrease in misclassification rate across trees', y = '')
#mda_plot
# coefficients of logistic regression model
# coef(fit_glm) %>% exp() %>% round(2) %>%
#   pander()
# center and scale
x_mx <- merged_data %>%
  select(-c('county', 'fips', 'candidate', 'state', 'votes', 'total', 'pct', 'CensusTract', 'winnerThing'))
  scale(center= T, scale = T)

# compute loadings for PC1 and PC2
x_svd <- svd(x_mx)
v_svd <- x_svd$v[,1:2]
z_mx <- x_mx %*% x_svd$v

# compute wss for several k and plot (The result is K=2)
#nb_out <- NbClust(x_mx, method = 'kmeans')
#k <- nb_out$Best.nc %>% t() %>% as_tibble() %>% count(Number_clusters)
```

```r
# compute cluters, obtain centriods
kmeans_out <- kmeans(x_mx, centers = 2, nstart = 5)
kmeans_best <- kmeans(x_mx, centers = 2, nstart = 25)
centers <- scale(kmeans_best$centers,
                 center = apply(x_mx, 2, mean),
                 scale = apply(x_mx, 2, sd)) %>% t() %>% as.data.frame() %>% mutate(variable = colnames(kmea
colnames(centers)[1:2] <- paste("cluster", 1:2)
centers <- gather(centers, key = 'cluster', value = 'center', 1:2)
clusters <- factor(kmeans_out$cluster,
                   labels = paste('cluster', 1:2))

# plot centroid coordinates against variable
distribution <- x_mx %>%
  as.data.frame() %>%
  mutate(across(everything(), scale),
         cluster = factor(kmeans_out$cluster, labels = paste('cluster', 1:2))) %>%
  gather(key = "variable", value = "value", -cluster) %>%
  ggplot(aes(x = value, y = variable)) +
  geom_point(aes(x = center, fill = cluster), shape = 21, size = 2, data = centers) +
  theme_minimal() +
  xlim(c(-1.5, 1.5)) +
  labs(x = 'Z-score', y = '') +
  theme(legend.position = 'top')
#distribution
sample <- merged_data$winnerThing
p <- z_mx[, 1:2] %>%
  as.data.frame() %>%
  rename(PC1 = V1, PC2 = V2) %>%
  mutate(cluster = clusters) %>%
  ggplot(aes(x = PC1, y = PC2)) +
  geom_point(aes(color = sample), alpha = 0.3, size = 3) +
  geom_point(aes(shape = cluster)) +
  ggtitle("K-means clusters") +
  theme_bw()

kmeans_ctrs <- kmeans_out$centers %*% v_svd %>%
  as.data.frame()
colnames(kmeans_ctrs) <- paste('PC', 1:2, sep = '')

kmean_pcPlot <- p + geom_point(data = kmeans_ctrs[, 1:2], size = 1)
#kmean_pcPlot
draw.tree(t_opt, cex = 0.6, size = 0.1, digits = 2)
summary(fit_glm) %>%
  pander()
summary(train_rf) %>%
  pander()
library(gridExtra)
grid.arrange(map_true, map_t, map_lg, map_rf, nrow = 2)
table_lg %>%
  pander()
cat("Logistic Regression Misclassification error rate:", lg_mse) %>%
  pander()
table_topt %>%
  pander()
cat("Decision tree Misclassification error rate:", topt_mse) %>%
  pander()
table_rf %>%
  pander()
cat("Randomforest Misclassification error rate:", rf_mse) %>%
  pander()
mda_plot
coef(fit_glm) %>% exp() %>% round(3) %>%
  pander()
kmean_pcPlot
distribution
```