2016-2012 Election Modeling

This is a project to explore which features of counties made them more sucespitable to voting for trump. First, let's scrape results. Our 2016 results come courtesy of the AP, with some help from Python, and our 2012 results come courtesy of the Huffington Post, with bit of help from Unix.

```
import json
import requests
import time
import threading
import pandas as pd

# Scraping the AP's live feed to get county level 2016 results

with open('./Election-Model/president_metadata.json') as pres_meta:
    pres_meta = json.load(pres_meta)
with open('./Election-Model/president.json') as pres:
    pres = json.load(pres)
def getResults():
    t = requests.get("http://interactives.ap.org/interactives/2016/general-election/live-data/productio
    pres = t.json()
    return pres
"http://interactives.ap.org/interactives/2016/general-election/live-data/production/2016-11-08/presiden
pres['results']

def make_cand_dict():
    cand_dict = {}
    with open('./Election-Model/president_metadata.json') as pres_meta:
        pres_meta = json.load(pres_meta)
    for id,meta in pres_meta['cands'].items():
        cand_dict[id] = meta['fn']
    return cand_dict

def parse_results(pres, cand_dict):
    parsed_results = {}
    for state in pres['results'][1:]:
        p_res = list(zip(state['cand'], state['vote']))
        parsed_results[state['st'] + state['id']] = {'county':state['id'],'state':state['st'], 'PR': sta
    return parsed_results

def called_races(parsed_results):
    cl_races = dict()
    last_updated = time.time()
    print("Results as of {}".format(str(time.ctime(int(last_updated)))))
    for state,race in parsed_results.items():
        if race['Status'] is not None:
            cl_races[state] = race#(race,state)
    return cl_races
def parsePres():
    pres = getResults()
    cand_dict = make_cand_dict()
    parsed_results = parse_results(pres, cand_dict)
    return called_races(parsed_results)
```

```
parsePres()

getResults()

cl_races = parsePres()


results = []
for state,race in parsed_results.items():
    for candidate in race['results']:
         if candidate[0] in ['Hillary Clinton', 'Donald Trump']:
                results.append([state,race['PR'], candidate[0],candidate[1]])



pd.DataFrame(results).to_csv('./results_16.csv')
```

Now that we've extracted everything from the AP's live feed, we should still probably clean everything and also grab our 2012 results.

```
library(readr)
results_16 <- read_csv("~/results_16.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   X1 = col_integer(),
##   `0` = col_character(),
##   `1` = col_double(),
##   `2` = col_character(),
##   `3` = col_integer()
## )
```

```
results_16<- results_16[,2:ncol(results_16)]
colnames(results_16)<-c("State_FIPS", "pct_reporting", "Candidate", "Votes")
require(reshape2)
```

```
## Loading required package: reshape2
```

```
# The results we scrape from the AP are in long format, we want them in wide format
results_tmp<-dcast(results_16, State_FIPS ~ Candidate, value.var = "Votes")
results_tmp$FIPS<-substring(results_tmp$State_FIPS, 3)
results_tmp$vote_share_dem_16<-results_tmp$`Hillary Clinton`/(results_tmp$`Donald Trump`+results_tmp$`
```

So now that we've cleaned our 2016 results, we need to get our 2012 results. These are scraped from the Huffington posts's github-but they come in state-by-state CSVs- just takes a bit of command-line trickery to concatenate them.

```
system("cd ..; cd Downloads; cd election-2012-results-master; cd data; cat *.csv > results_12_merged.csv
results_12<-read_csv("~/Downloads/election-2012-results-master/data/results_12_merged.csv")
```

```
## Parsed with column specification:
## cols(
##   fips = col_character(),
##   county = col_character(),
##   candidate = col_character(),
```

```
##   votes = col_character()
## )
```

```
results_12<-results_12[,c(1,3,4)]
results_12<-results_12[results_12$fips!="fips", ]
results_12$votes<-as.numeric(results_12$votes)
results_12<-dcast(results_12, fips ~ candidate, value.var = "votes", fun.aggregate = sum)
results_12$R_Totals<-rowSums(results_12[,169:180]) + rowSums(results_12[,219:231])
results_12$D_Totals<-rowSums(results_12[,17:28]) + rowSums(results_12[,184:196])
results_12$vote_share_dem_12<-results_12$D_Totals/(results_12$D_Totals + results_12$R_Totals)
results_12<-as.data.frame(cbind(results_12$fips, results_12$vote_share_dem_12))
colnames(results_12)<-c("fips", "vote_share_dem_12" )
require(sqldf)
```

```
## Loading required package: sqldf
```

```
## Loading required package: gsubfn
```

```
## Loading required package: proto
```

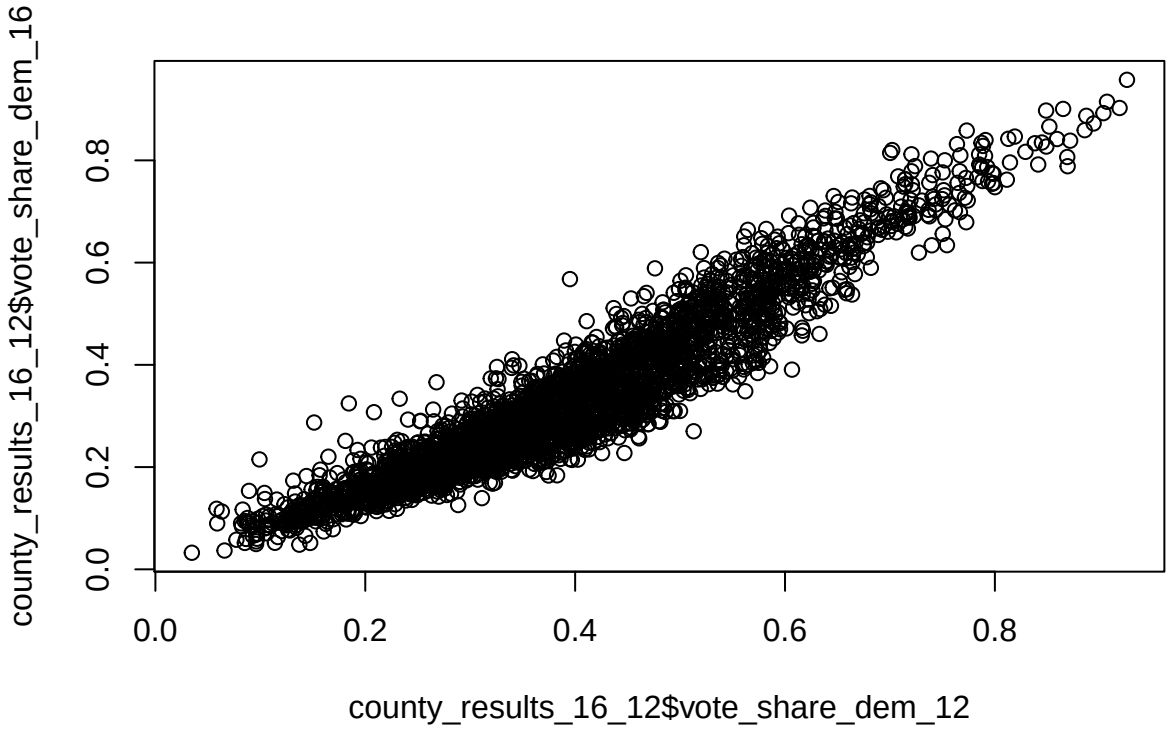
```
## Loading required package: RSQLite
```

```
## Loading required package: DBI
```

```
# joining our results by fips.
county_results_16_12<-sqldf("select a.fips, a.vote_share_dem_12, b.vote_share_dem_16 from results_tmp b
```

```
## Loading required package: tcltk
```

```
county_results_16_12<-county_results_16_12[!is.na(county_results_16_12$fips),]
plot(county_results_16_12$vote_share_dem_12, county_results_16_12$vote_share_dem_16)
```



This gives us a good indication of the correlation between 2012 and 2016 vote share- Let's append some census data onto everything, and see how it goes.

```
require(acs)
```

```
## Loading required package: acs
```

```
## Loading required package: stringr
```

```
## Loading required package: plyr
```

```
## Loading required package: XML
```

```
##
## Attaching package: 'acs'
```

```
## The following object is masked from 'package:base':
##
##     apply
```

```
require(sqldf)
#api.key.install(key="3fdf164b3e49c898d30e959ed1b18194fe3b4dfa")
all_counties<-geo.make(state="*",county="*")
race_table<-acs.fetch(endyear = 2015, span=1, geography = all_counties, table.number = "B02001", col.nam
```

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

```
hispanic_table<-acs.fetch(endyear = 2015, span=1, geography = all_counties, table.number = "B03003", col
income_table<-acs.fetch(endyear = 2015, span = 1, geography = all_counties, table.number = "B19001", col
education_table<-acs.fetch(endyear = 2015, span=1, geography = all_counties, table.number = "B15003", co
```

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

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

```
employment_table<-acs.fetch(endyear=2015, span = 1, geography = all_counties, table.number = "B23025", c
gender_age_table<-acs.fetch(endyear=2015, span = 1, geography = all_counties, table.number = "B01001", c
marital_table<-acs.fetch(endyear=2015, span = 1, geography = all_counties, table.number = "B12001", col

# function for taking an ACS table, turning total counts into proportions, and turning it into a data f
Clean_ACS<-function(table_name){
  estimate_mat<-estimate(table_name)
  estimate_table<-as.data.frame(prop.table(estimate_mat[,2:ncol(estimate_mat)],1))
  estimate_table$id<-rownames(estimate_table)
  geo_table<-as.data.frame(geography(table_name))
  joined_table<-sqldf("Select a.*, b.* from geo_table a left join estimate_table b on a.NAME=b.id")
  joined_table<-joined_table[,1:(ncol(joined_table)-1)]
  joined_table$fips<-paste0(str_pad(joined_table$state, width=2, side="left", pad="0"), joined_table$cou
  return(joined_table)
}

race_df<-Clean_ACS(race_table)
hispanic_df<-Clean_ACS(hispanic_table)
income_df<-Clean_ACS(income_table)
education_df<-Clean_ACS(education_table)
emp_df<-Clean_ACS(employment_table)
gender_age_df<-Clean_ACS(gender_age_table)
marital_df<-Clean_ACS(marital_table)

all_acs_data<-cbind(race_df, hispanic_df[4:(ncol(hispanic_df)-1)], income_df[4:(ncol(income_df)-1)], edu
                    marital_df[4:(ncol(marital_df)-1)] )
```

```
require(dplyr)
```

```
## Loading required package: dplyr
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:acs':
##
##     combine
```

```
## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
acs_results<-dplyr::left_join(all_acs_data, county_results_16_12, by=c('fips'='fips'))
```

Great, now we have it all appended. Let's do some descriptive statistics

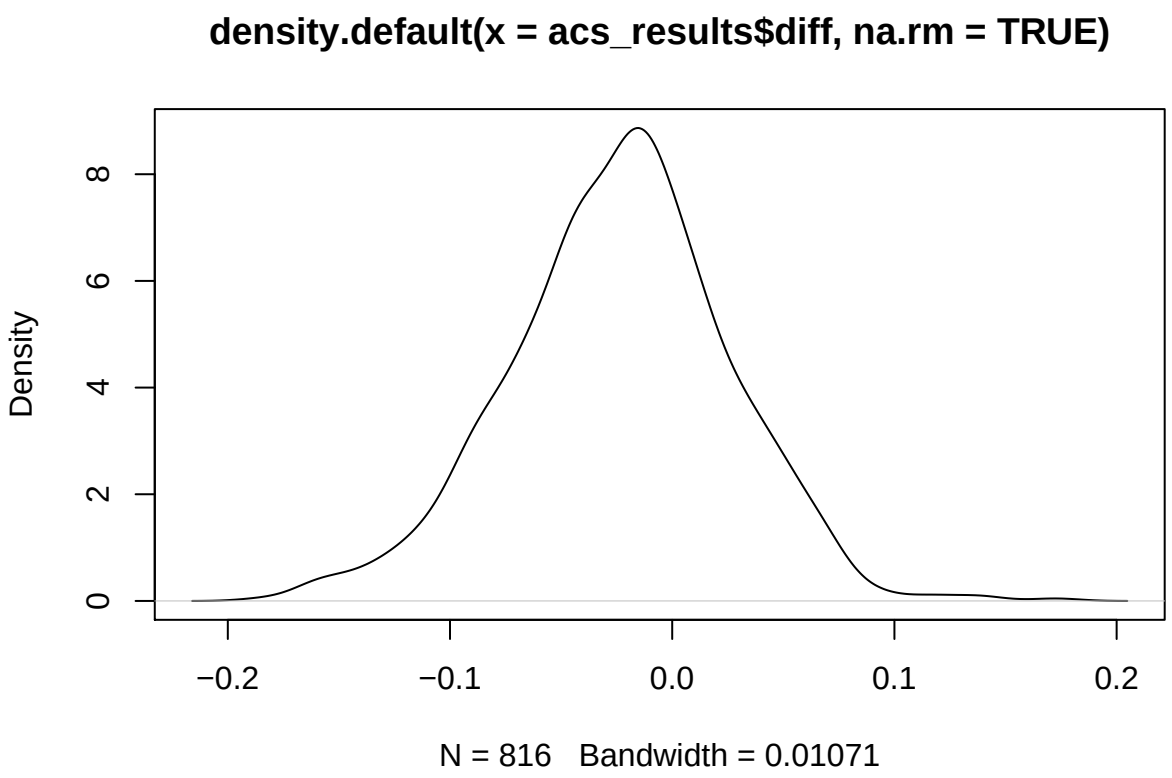
```
acs_results$vote_share_dem_12<-as.numeric(acs_results$vote_share_dem_12)
acs_results$diff<-acs_results$vote_share_dem_16-acs_results$vote_share_dem_12
mean(acs_results$diff, na.rm = TRUE)
```

```
## [1] -0.02389214
```

```
sd(acs_results$diff, na.rm = TRUE)
```

```
## [1] 0.04923842
```

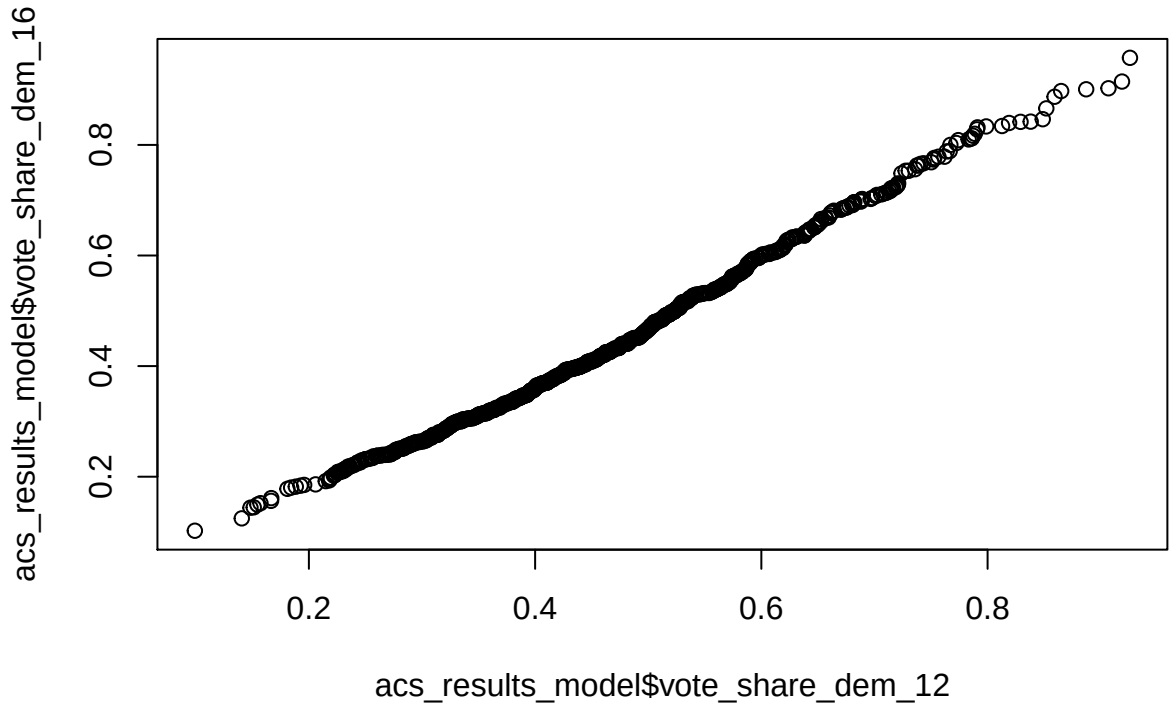
```
plot(density(acs_results$diff, na.rm=TRUE))
```




```
acs_results_model<-acs_results[!is.na(acs_results$diff),]
```

Ok, this shows what we've already known- in these 816 counties, she did worse- it doesn't take a day of appending data to figure that one out. Now, a qq plot. Because I love my qqplots.

```
qqplot(acs_results_model$vote_share_dem_12, acs_results_model$vote_share_dem_16)
```



Let's try to model this a few ways- first let's just run a glmnet to predict what caused a county to switch to

republican in 2016- a simple y/n.

```
require(caret)
```

```
## Loading required package: caret
```

```
## Warning: package 'caret' was built under R version 3.3.2
```

```
## Loading required package: lattice
```

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

```
require(glmnet)
```

```
## Loading required package: glmnet
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-5
```

```
require(pROC)
```

```
## Loading required package: pROC
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following object is masked from 'package:glmnet':
##
##     auc
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
require(Matrix)
require(ROCR)
```

```
## Loading required package: ROCR
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
set.seed(7)

for(i in 1:ncol(acs_results_model)){
  acs_results_model[is.na(acs_results_model[,i]), i] <- mean(acs_results_model[,i], na.rm = TRUE)
}
```

```
## Warning in mean.default(acs_results_model[, i], na.rm = TRUE): argument is
## not numeric or logical: returning NA
```

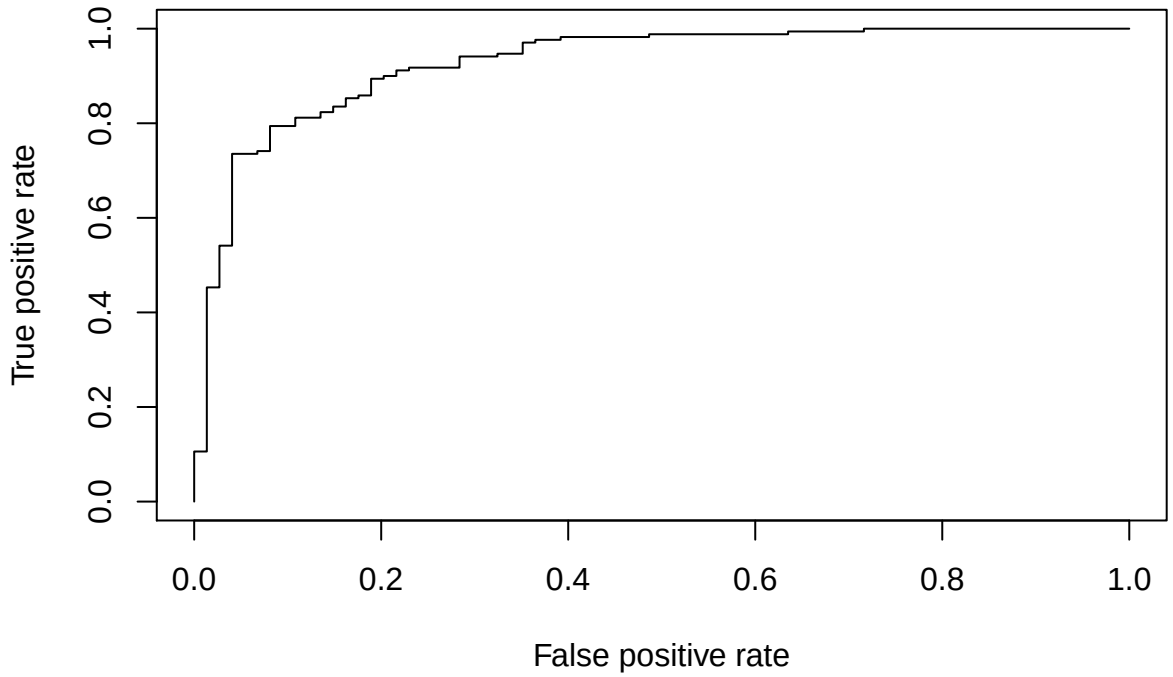
```
## Warning in mean.default(acs_results_model[, i], na.rm = TRUE): argument is
## not numeric or logical: returning NA
```

```
## Warning in mean.default(acs_results_model[, i], na.rm = TRUE): argument is
## not numeric or logical: returning NA
```

```
acs_df<-as.data.frame(acs_results_model)
acs_df$flipped<-ifelse(acs_df$diff<0, "y", "n")
exclude.vars<-c("NAME", "state", "county", "vote_share_dem_12", "vote_share_dem_16", "diff", "fips")
acs_df = acs_df[,!(names(acs_df) %in% exclude.vars)]
reg.var<-"flipped"
partition_rows<-createDataPartition(acs_df[, reg.var], p=.7, list=FALSE, times=1)
acs_train<-acs_df[partition_rows,]
acs_test<-acs_df[-partition_rows,]
flip_model = cv.glmnet(as.matrix(acs_train[,1:ncol(acs_train)-1]), acs_train$flipped, family="binomial"
acs_test$fitted<-predict(flip_model, newx = as.matrix(acs_test[,1:ncol(acs_test)-1]), type="response", s
pred<-prediction(acs_test$fitted, acs_test$flipped)
perf <- performance(pred,"tpr","fpr")
performance(pred,"auc")
```

```
## An object of class "performance"
## Slot "x.name":
## [1] "None"
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
## Slot "alpha.name":
## [1] "none"
##
## Slot "x.values":
## list()
##
## Slot "y.values":
## [[1]]
## [1] 0.927345
##
##
## Slot "alpha.values":
## list()
```

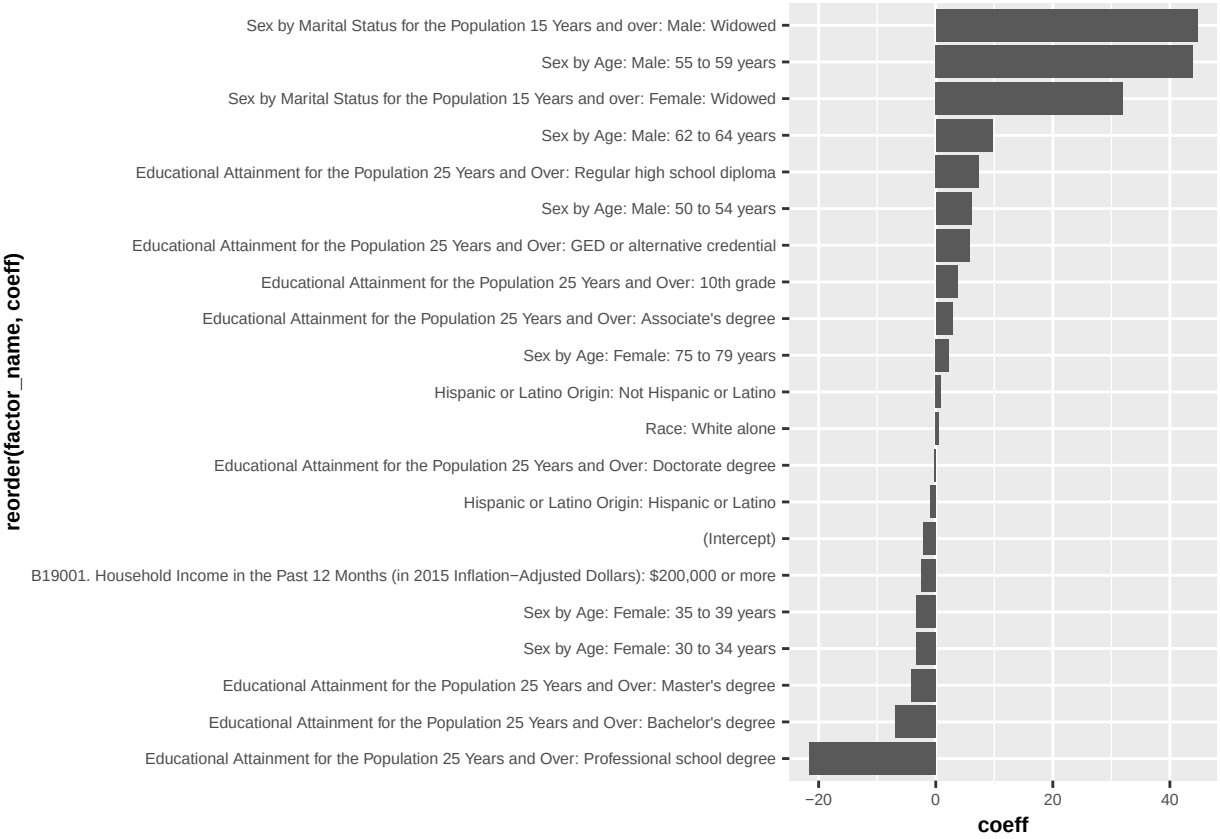
```
plot(perf,colorize=FALSE, col="black")
```



Ok, so we're either really overfit, or we're actually doing pretty well here- I'll leave that up to you. What's important to note isn't really the model- not too much point in trying to predict what already happened, but we can learn a lot from creating this model- specifically, what are the biggest influencers in a county switching from 2012-2016?

```
flip_coeffs<-coef.cv.glmnet(flip_model)
tmp<-as.data.frame(as.matrix(flip_coeffs))
tmp$factor<-rownames(tmp)
colnames(tmp)<-c("coeff", "factor_name")
tmp<-tmp[tmp$coeff!=0,]
imp_plot<-ggplot(data=tmp, aes(x=reorder(factor_name, coeff), y=coeff)) +
  geom_bar(stat="identity") + coord_flip()+theme(axis.text=element_text(size=6),
        axis.title=element_text(size=8,face="bold"))
imp_plot
```

```
## Warning: Stacking not well defined when ymin != 0
```



What stands out: Education, Age, and Gender seem to carry the most weight when determining which counties stayed democratic, and which ones did not. There is a marked split between people with 2 year college degrees and people with 4 year college degrees- which suggests that a 4 year college degree is really the tipping point at which education affects partisanship. Furthermore, I think that the high weights of both widowed men and women suggest serve as proxies for Age and Income- people whom are older are more likley to be widowed, and people who have lower income are more likely to have shorter lifespans, which would create relatively high amounts of widows in the population. To me, the most interesting part is that in an eleciton where a common theme was racial political polarization, race seemed to play a small role in determining which counties flipped republican, compared with overall education, race, and gender.