PSTAT231-final (231 Level)

Xianjun Yang and Selin Karabulut

March 18, 2021

Background The presidential election in 2012 did not come as a surprise. Some predicted the outcome of the election correctly including Nate Silver, and many speculated his approach.

Despite the success in 2012, the 2016 presidential election came as a big surprise to many, and it was a clear example that even the current state-of-the-art technology can surprise us.

Answer the following questions in one paragraph for each.

1. What makes voter behavior prediction (and thus election forecasting) a hard problem?

   Data collection: there are always wrong data in data collection process; also the collection can not cover the majority of people from all counties; and the data collection process might ignore some population and focus too much on the others; Data integrity: there are too many factors that could influence the election results, and it is difficult to know all the influencing factors; Noise: there are too much noise for forecasting that we include in our data; Randomness of voters: some voter just make votes ramdomly on election day, which is independent of factors.

2. What was unique to Nate Silver's approach in 2012 that allowed him to achieve good predictions?

   Compared with Political pundits who are paid to spread opinions, Silver scientifically collect data and build the model all on the mathitical base rather than personal preference. The size of the database of Silver is very large, not only including the president election data, but also many other votes data; so he already simulated the prediction modle on different voting events and thus gained valuable experience; Silver learned from these experience, and adjust his model timely according to the latest changes.

3. What went wrong in 2016? What do you think should be done to make future predictions better?

   In 2016, prediction was wrong for many reasons, for example: Data bias: the data was not evenly collected from all counties all over US, and in the future data collection should be made more evenly so that no certain population will be ignored; failure for the polls: it has been verified that the polls are wrong because of some systemic bias on polls. In the furure, more accuate polls should be designed for prediction Ignorance of cettain factors: for example, the voting behavior of the minority decreases and was not considered; Voters behavior: some people believe Clinton was going to win and thus did not vote, while more Trump supporters are encouraged to vote.

Data

```
## set the working directory as the file location
setwd(getwd())
## put the data folder and this handout file together.
## read data and convert candidate from string to factor
election.raw <- read_delim("data/election/election.csv", delim = ",") %>% mutate(candidate=as.factor(can

census_meta <- read_delim("data/census/metadata.csv", delim = ";", col_names = FALSE)
census <- read_delim("data/census/census.csv", delim = ",")
```

Election data The meaning of each column in election.raw is clear except fips. The accronym is short for Federal Information Processing Standard.

In our dataset, fips values denote the area (US, state, or county) that each row of data represent. For example, fips value of 6037 denotes Los Angeles County.

```r
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```r
kable(election.raw %>% filter(county == "Los Angeles County"))  %>% kable_styling(bootstrap_options = c
```

```
\begin{table}
\centering
\begin{tabular}{l|l|l|l|r}
\hline
county & fips & candidate & state & votes\\
\hline
Los Angeles County & 6037 & Hillary Clinton & CA & 2464364\\
\hline
Los Angeles County & 6037 & Donald Trump & CA & 769743\\
\hline
Los Angeles County & 6037 & Gary Johnson & CA & 88968\\
\hline
Los Angeles County & 6037 & Jill Stein & CA & 76465\\
\hline
Los Angeles County & 6037 & Gloria La Riva & CA & 21993\\
\hline
\end{tabular}
\end{table}
```

Some rows in election.raw are summary rows and these rows have county value of NA. There are two kinds of summary rows:

Federal-level summary rows have fips value of US. State-level summary rows have names of each states as fips value. **4** Report the dimension of election.raw after removing rows with fips=2000. Provide a reason for excluding them. Please make sure to use the same name election.raw before and after removing those observations.

```r
kable(election.raw %>% filter(fips == 2000))  %>% kable_styling(bootstrap_options = c("striped", "hover
```

```
\begin{table}
\centering
\begin{tabular}{l|l|l|l|r}
\hline
county & fips & candidate & state & votes\\
\hline
NA & 2000 & Donald Trump & AK & 163387\\
\hline
NA & 2000 & Hillary Clinton & AK & 116454\\
\hline
NA & 2000 & Gary Johnson & AK & 18725\\
\hline
NA & 2000 & Jill Stein & AK & 5735\\
\hline
NA & 2000 & Darrell Castle & AK & 3866\\
\hline
NA & 2000 & Rocky De La Fuente & AK & 1240\\
\hline
\end{tabular}
\end{table}
```

```r
election.raw <- election.raw%>%filter(fips!="2000")
dim(election.raw)
```

```
## [1] 18345     5
```

Reason: there are no corresponding county records when flips=2000, so these rows are considered missing

Census data Following is the first few rows of the census data:

```r
head(census, n=6)
```

```
## # A tibble: 6 x 36
##   State County TotalPop   Men Women Hispanic White Black Native Asian Pacific
##   <chr> <chr>     <dbl> <dbl> <dbl>    <dbl> <dbl> <dbl>  <dbl> <dbl>   <dbl>
## 1 Alab~ Autau~     1948   940  1008      0.9  87.4   7.7    0.3   0.6     0
## 2 Alab~ Autau~     2156  1059  1097      0.8  40.4  53.3    0     2.3     0
## 3 Alab~ Autau~     2968  1364  1604      0    74.5  18.6    0.5   1.4     0.3
## 4 Alab~ Autau~     4423  2172  2251     10.5  82.8   3.7    1.6   0       0
## 5 Alab~ Autau~    10763  4922  5841      0.7  68.5  24.8    0     3.8     0
## 6 Alab~ Autau~     3851  1787  2064     13.1  72.9  11.9    0     0       0
## # ... with 25 more variables: Citizen <dbl>, Income <dbl>, IncomeErr <dbl>,
## #   IncomePerCap <dbl>, IncomePerCapErr <dbl>, Poverty <dbl>,
## #   ChildPoverty <dbl>, Professional <dbl>, Service <dbl>, Office <dbl>,
## #   Construction <dbl>, Production <dbl>, Drive <dbl>, Carpool <dbl>,
## #   Transit <dbl>, Walk <dbl>, OtherTransp <dbl>, WorkAtHome <dbl>,
## #   MeanCommute <dbl>, Employed <dbl>, PrivateWork <dbl>, PublicWork <dbl>,
```

```
## #   SelfEmployed <dbl>, FamilyWork <dbl>, Unemployment <dbl>
```

Census data: column metadata Column information is given in metadata. **Data wrangling** 5. Remove summary rows from election.raw data: i.e., * Federal-level summary into a `election_federal`.

- State-level summary into a `election_state`.
- Only county-level data is to be in `election`.

```r
election <- election.raw %>% filter(fips!="US")
temp = is.na(as.numeric(election$fips))
```

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

```r
election_state<- election[temp, ]
election<- election[!temp, ]
election_federal <- election.raw %>% filter(fips =="US")
```

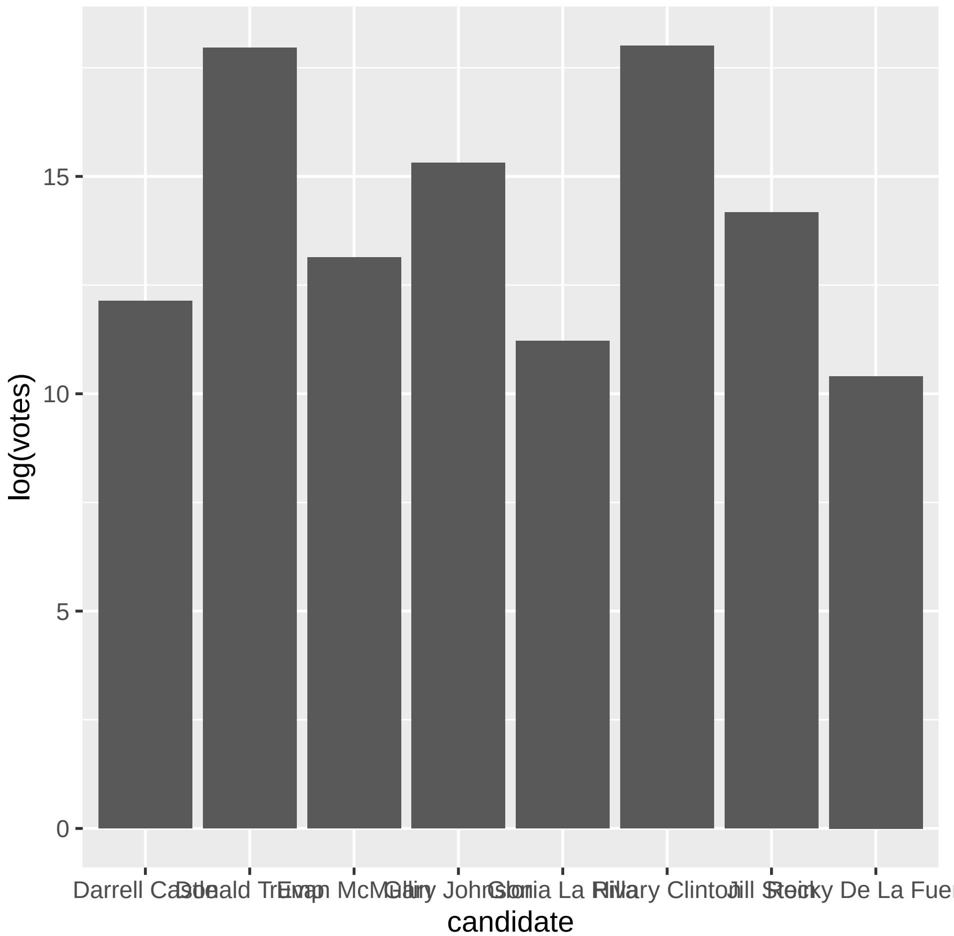
6. How many named presidential candidates were there in the 2016 election? Draw a bar chart of all votes received by each candidate. You can split this into multiple plots or may prefer to plot the results on a log scale. Either way, the results should be clear and legible!

```r
dim(election_federal)[1]
```

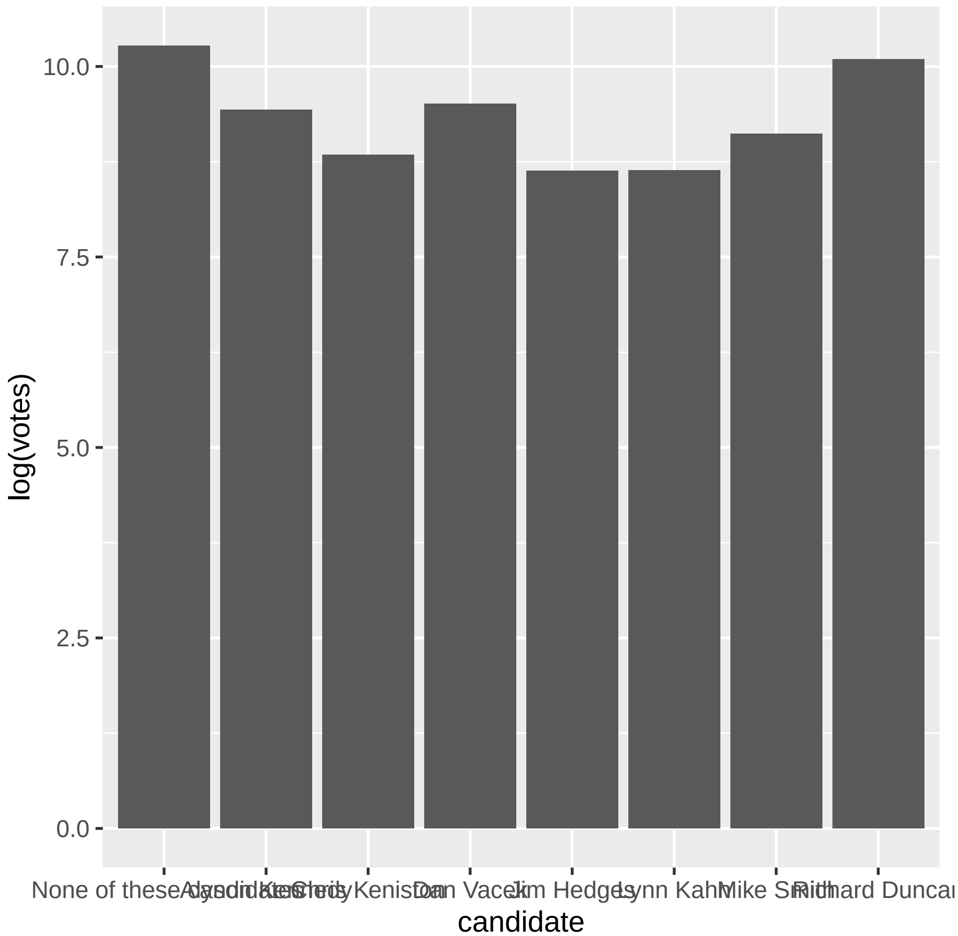
```
## [1] 32
```

There are 32 presidential candidates in the 2016 election

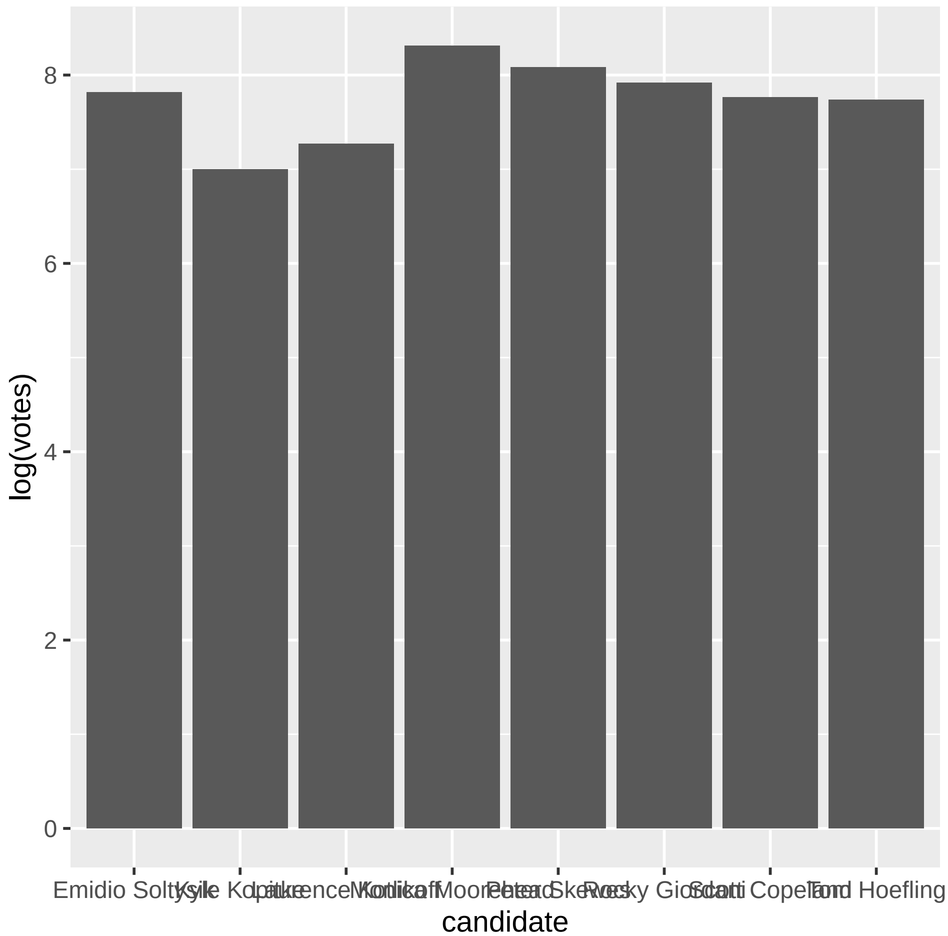
```r
data_candidates = data.frame( candidate = election_federal[1:8, 3],  votes = log(election_federal[1
ggplot(data = data_candidates, aes(x= candidate, y = votes))+
  geom_bar(stat='identity')+
  labs("candidates~votes", y = "log(votes)", x = "candidate")
```



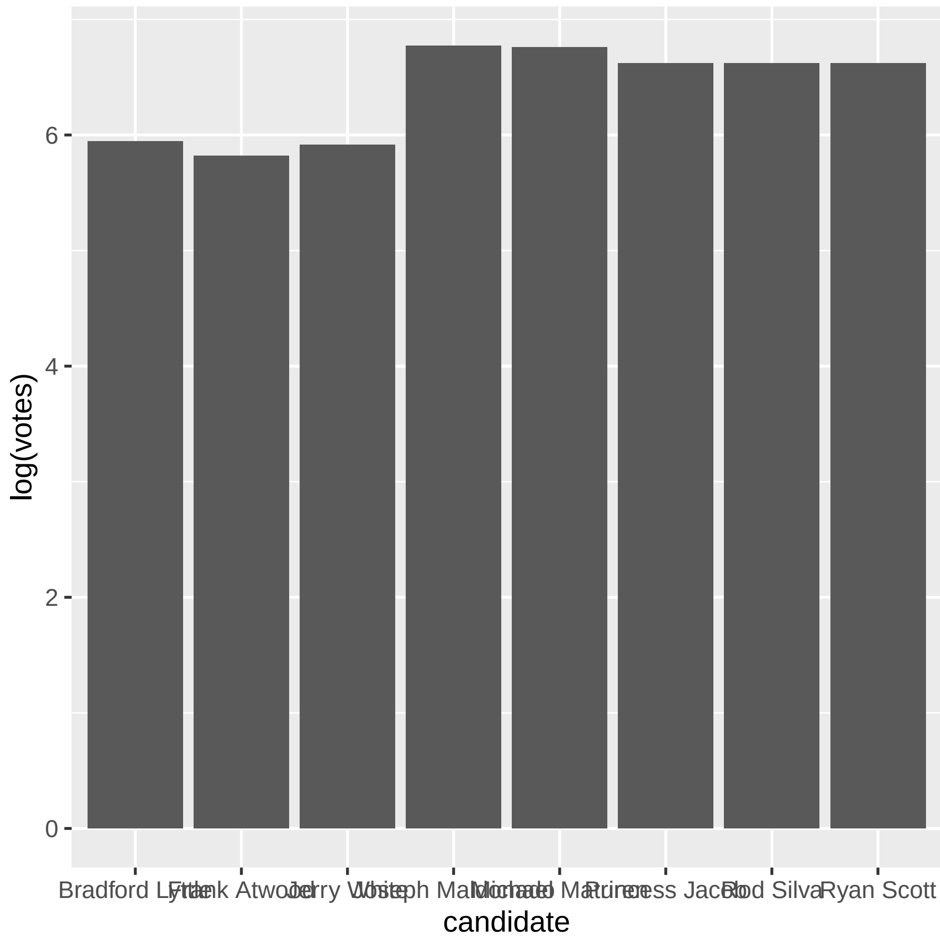

```
data_candidates = data.frame( candidate = election_federal[9:16, 3],  votes = log(election_federal[
ggplot(data = data_candidates, aes(x= candidate, y = votes))+
  geom_bar(stat='identity')+
  labs("candidates~votes", y = "log(votes)", x = "candidate")
```




```
data_candidates = data.frame( candidate = election_federal[17:24, 3],  votes = log(election_federal
ggplot(data = data_candidates, aes(x= candidate, y = votes))+
  geom_bar(stat='identity')+
  labs("candidates~votes", y = "log(votes)", x = "candidate")
```




```
data_candidates = data.frame( candidate = election_federal[25:32, 3],  votes = log(election_federal
ggplot(data = data_candidates, aes(x= candidate, y = votes))+
  geom_bar(stat='identity')+
  labs("candidates~votes", y = "log(votes)", x = "candidate")
```



7. Create variables county_winner and state_winner by taking the candidate with the highest proportion of votes. Hint: to create county_winner, start with election, group by fips, compute total votes, and pct = votes/total. Then choose the highest row using top_n (variable state_winner is similar).

```r
county_winner <- election %>%
  group_by(fips) %>%
  mutate(total=sum(votes), pct=votes/total) %>%
  top_n(1, pct)

state_winner <- election_state %>%
  group_by(fips) %>%
  mutate(total=sum(votes), pct=votes/total) %>%
  top_n(1, pct)
```

Visualization Visualization is crucial for gaining insight and intuition during data mining. We will map our data onto maps.

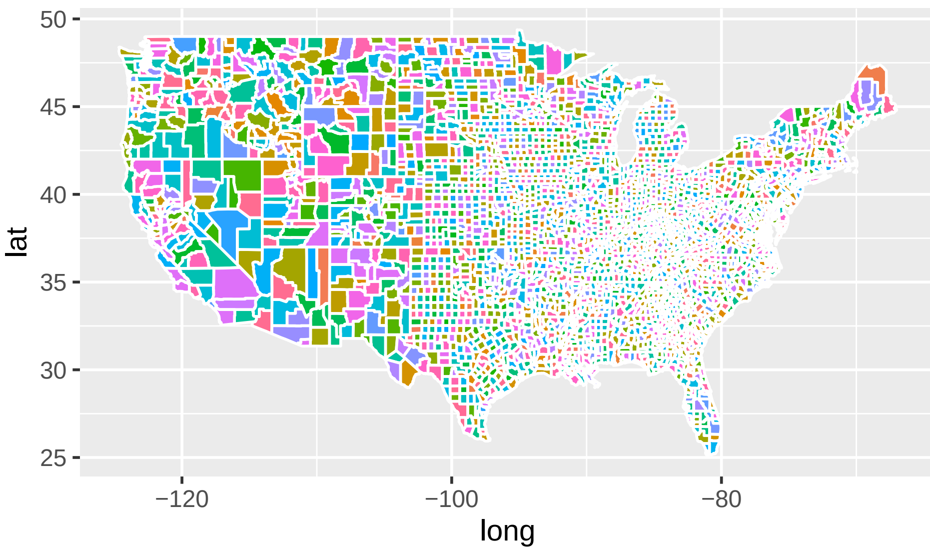
The R package ggplot2 can be used to draw maps. Consider the following code.

````
```r
states <- map_data("state")
ggplot(data = states) +
  geom_polygon(aes(x = long, y = lat, fill = region, group = group), color = "white") +
  coord_fixed(1.3) +
  guides(fill=FALSE)  # color legend is unnecessary and takes too long
```
````

\begin{center}\includegraphics{final_files/figure-latex/Visualization-1} \end{center}

8. Draw county-level map by creating counties = map_data("county"). Color by county

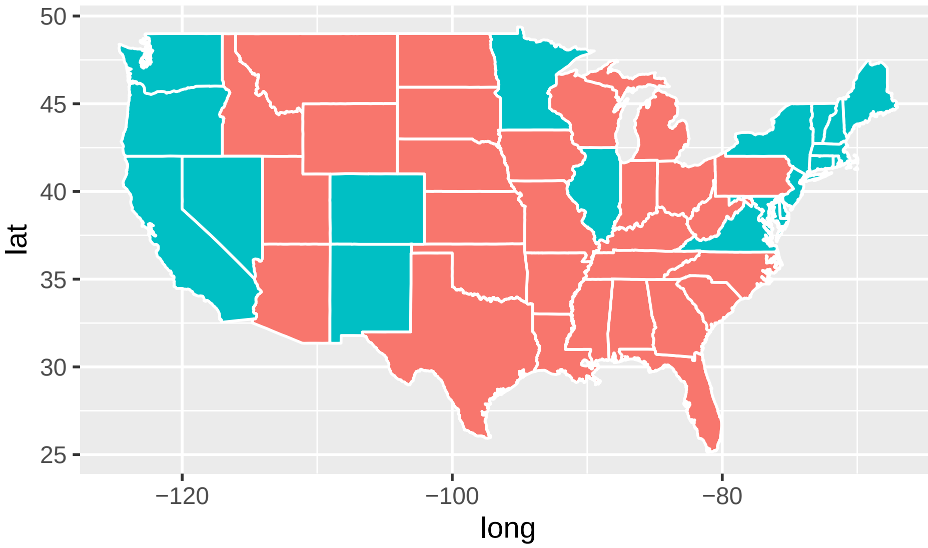
```
counties <- map_data("county")
ggplot(data = counties) +
  geom_polygon(aes(x = long, y = lat, fill = subregion, group = group), color = "white") +
  coord_fixed(1.3) +
  guides(fill=FALSE)  # color legend is unnecessary and takes too long
```



9. Now color the map by the winning candidate for each state. First, combine states variable and state_winner we created earlier using left_join(). Note that left_join() needs to match up values of states to join the tables. A call to left_join() takes all the values from the first table and looks for matches in the second table. If it finds a match, it adds the data from the second table; if not, it adds missing values: Here, we'll be combing the two datasets based on state name. However, the state names are in different formats in the two tables: e.g. AZ vs. arizona. Before using left_join(), create a common column by creating a new column for states named fips = state.abb[match(some_column, some_function(state.name))]. Replace some_column and some_function to complete creation of this new column. Then left_join(). Your figure will look similar to state_level New York Times map.

```
fips = state.abb[match(states$region, tolower(state.name))]
states$fips = fips
```

```
states_combined = left_join(states, state_winner, by = "fips")
ggplot(data = states_combined) +
  geom_polygon(aes(x = long, y = lat, fill = candidate, group = group), color = "white") +
  coord_fixed(1.3) +
  guides(fill=FALSE)  # color legend is unnecessary and takes too long
```



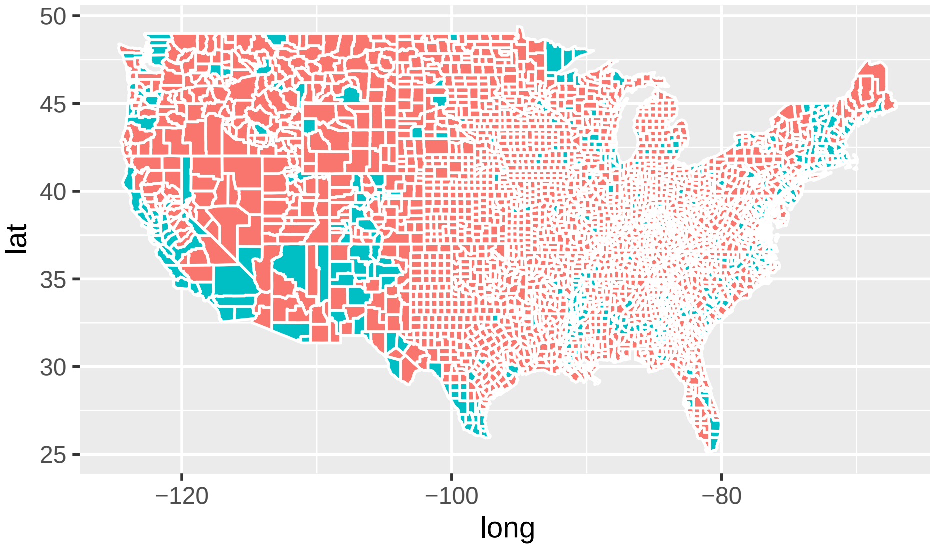
10. The variable county does not have fips column. So we will create one by pooling information from maps::county.fips. Split the polyname column to region and subregion. Use left_join() combine county.fips into county. Also, left_join() previously created variable county_winner. Your figure will look similar to county-level New York Times map.

```
countyinfo = maps::county.fips
regiontotal = unlist( strsplit(countyinfo$polyname, ',') )
index = seq(1,6170,2)
region = regiontotal[index]
subregion = regiontotal[-index]

countyinfo$region = region
countyinfo$subregion = subregion
county_combined = left_join(counties, countyinfo)
```

```
## Joining, by = c("region", "subregion")
```

```
county_combined$fips = as.character(county_combined$fips)
county_final = left_join(county_winner, county_combined, by = "fips")
```

```
ggplot(data = county_final) +
  geom_polygon(aes(x = long, y = lat, fill = candidate, group = group), color = "white") +
  coord_fixed(1.3) +
  guides(fill=FALSE)  # color legend is unnecessary and takes too long
```



11. Create a visualization of your choice using census data. Many exit polls noted that demographics played a big role in the election. Use this Washington Post article and this R graph gallery for ideas and inspiration.

12. The census data contains high resolution information (more fine-grained than county-level). In this problem, we aggregate the information into county-level data by computing TotalPop-weighted average of each attributes for each county. Create the following variables:

Clean census data census.del: start with census, filter out any rows with missing values, convert {Men, Employed, Citizen} attributes to percentages (meta data seems to be inaccurate), compute Minority attribute by combining {Hispanic, Black, Native, Asian, Pacific}, remove these variables after creating Minority, remove {Walk, PublicWork, Construction}. Many columns seem to be related, and, if a set that adds up to 100%, one column will be deleted.

````
```r
census.del <- na.omit(census)
#convert {`Men`, `Employed`, `Citizen`} attributes to a percentages
census.del$Men <- census.del$Men/census.del$TotalPop
census.del$Employed <- census.del$Employed/census.del$TotalPop
census.del$Citizen <- census.del$Citizen/census.del$TotalPop
#combining {Hispanic, Black, Native, Asian, Pacific}
census.del$Minority <- census.del$Hispanic+census.del$Black+census.del$Native+census.del$Asian+census.de
#remove {`Walk`, `PublicWork`, `Construction`}
census.del <- select(census.del, -Walk,-PublicWork,-Construction)
#Remove colunms that are unneccessary
census.del <- select(census.del, -Women)
census.del<- select(census.del,-Hispanic,-Black,-Native,-Asian,-Pacific)
census.del
```
````

```
## # A tibble: 72,727 x 28
##    State County TotalPop   Men White Citizen Income IncomeErr IncomePerCap
##    <chr> <chr>     <dbl> <dbl> <dbl>   <dbl>  <dbl>     <dbl>        <dbl>
##  1 Alab~ Autau~     1948 0.483  87.4   0.772  61838     11900        25713
##  2 Alab~ Autau~     2156 0.491  40.4   0.771  32303     13538        18021
##  3 Alab~ Autau~     2968 0.460  74.5   0.787  44922      5629        20689
##  4 Alab~ Autau~     4423 0.491  82.8   0.747  54329      7003        24125
```

```
##  5 Alab~ Autau~    10763 0.457  68.5   0.712  51965      6935        27526
##  6 Alab~ Autau~     3851 0.464  72.9   0.686  63092      9585        30480
##  7 Alab~ Autau~     2761 0.438  74.5   0.746  34821      7867        20442
##  8 Alab~ Autau~     3187 0.471  84     0.750  73728      2447        32813
##  9 Alab~ Autau~    10915 0.503  89.5   0.713  60063      8602        24028
## 10 Alab~ Autau~     5668 0.511  85.5   0.744  41287      7857        24710
## # ... with 72,717 more rows, and 19 more variables: IncomePerCapErr <dbl>,
## #   Poverty <dbl>, ChildPoverty <dbl>, Professional <dbl>, Service <dbl>,
## #   Office <dbl>, Production <dbl>, Drive <dbl>, Carpool <dbl>, Transit <dbl>,
## #   OtherTransp <dbl>, WorkAtHome <dbl>, MeanCommute <dbl>, Employed <dbl>,
## #   PrivateWork <dbl>, SelfEmployed <dbl>, FamilyWork <dbl>,
## #   Unemployment <dbl>, Minority <dbl>
```

Sub-county census data, census.subct: start with census.del from above, group_by() two attributes {State, County}, use add_tally() to compute CountyTotal. Also, compute the weight by TotalPop/CountyTotal.

```r
#group by {`State`, `County`}
census.subct <- census.del %>%
group_by(State,County)
#compute `CountyTotal`
census.subct <- add_tally(census.subct,TotalPop,sort=FALSE)
colnames(census.subct)[29] <- "CountyTotal"
#compute the weight by `TotalPop/CountyTotal`
census.subct$Weight <- census.subct$TotalPop/census.subct$CountyTotal
census.subct
```

```
## # A tibble: 72,727 x 30
## # Groups:   State, County [3,218]
##    State County TotalPop   Men White Citizen Income IncomeErr IncomePerCap
##    <chr> <chr>     <dbl> <dbl> <dbl>   <dbl>  <dbl>     <dbl>        <dbl>
##  1 Alab~ Autau~     1948 0.483  87.4   0.772  61838     11900        25713
##  2 Alab~ Autau~     2156 0.491  40.4   0.771  32303     13538        18021
##  3 Alab~ Autau~     2968 0.460  74.5   0.787  44922      5629        20689
##  4 Alab~ Autau~     4423 0.491  82.8   0.747  54329      7003        24125
##  5 Alab~ Autau~    10763 0.457  68.5   0.712  51965      6935        27526
##  6 Alab~ Autau~     3851 0.464  72.9   0.686  63092      9585        30480
##  7 Alab~ Autau~     2761 0.438  74.5   0.746  34821      7867        20442
##  8 Alab~ Autau~     3187 0.471  84     0.750  73728      2447        32813
##  9 Alab~ Autau~    10915 0.503  89.5   0.713  60063      8602        24028
## 10 Alab~ Autau~     5668 0.511  85.5   0.744  41287      7857        24710
## # ... with 72,717 more rows, and 21 more variables: IncomePerCapErr <dbl>,
## #   Poverty <dbl>, ChildPoverty <dbl>, Professional <dbl>, Service <dbl>,
## #   Office <dbl>, Production <dbl>, Drive <dbl>, Carpool <dbl>, Transit <dbl>,
## #   OtherTransp <dbl>, WorkAtHome <dbl>, MeanCommute <dbl>, Employed <dbl>,
## #   PrivateWork <dbl>, SelfEmployed <dbl>, FamilyWork <dbl>,
## #   Unemployment <dbl>, Minority <dbl>, CountyTotal <dbl>, Weight <dbl>
```

County census data, census.ct: start with census.subct, use summarize_at() to compute weighted sum Print few rows of census.ct:

```r
```

```
census.ct<-census.subct %>%
summarise_at(vars(Men:CountyTotal), funs(weighted.mean(., Weight)))
```

```
## Warning: `funs()` is deprecated as of dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```r
census.ct <- data.frame(census.ct)
head(census.ct)
```

```
##     State  County       Men    White   Citizen   Income IncomeErr IncomePerCap
## 1 Alabama Autauga 0.4843266 75.78823 0.7374912 51696.29  7771.009     24974.50
## 2 Alabama Baldwin 0.4884866 83.10262 0.7569406 51074.36  8745.050     27316.84
## 3 Alabama Barbour 0.5382816 46.23159 0.7691222 32959.30  6031.065     16824.22
## 4 Alabama    Bibb 0.5341090 74.49989 0.7739781 38886.63  5662.358     18430.99
## 5 Alabama  Blount 0.4940565 87.85385 0.7337550 46237.97  8695.786     20532.27
## 6 Alabama Bullock 0.5300618 22.19918 0.7545420 33292.69  9000.345     17579.57
##   IncomePerCapErr  Poverty ChildPoverty Professional  Service   Office
## 1        3433.674 12.91231     18.70758     32.79097 17.17044 24.28243
## 2        3803.718 13.42423     19.48431     32.72994 17.95092 27.10439
## 3        2430.189 26.50563     43.55962     26.12404 16.46343 23.27878
## 4        3073.599 16.60375     27.19708     21.59010 17.95545 17.46731
## 5        2052.055 16.72152     26.85738     28.52930 13.94252 23.83692
## 6        3110.645 24.50260     37.29116     19.55253 14.92420 20.17051
##   Production    Drive   Carpool    Transit OtherTransp WorkAtHome MeanCommute
## 1   17.15713 87.50624  8.781235 0.09525905   1.3059687  1.8356531    26.50016
## 2   11.32186 84.59861  8.959078 0.12662092   1.4438000  3.8504774    26.32218
## 3   23.31741 83.33021 11.056609 0.49540324   1.6217251  1.5019456    24.51828
## 4   23.74415 83.43488 13.153641 0.50313661   1.5620952  0.7314679    28.71439
## 5   20.10413 84.85031 11.279222 0.36263213   0.4199411  2.2654133    34.84489
## 6   25.73547 74.77277 14.839127 0.77321596   1.8238247  3.0998783    28.63106
##    Employed PrivateWork SelfEmployed FamilyWork Unemployment Minority
## 1 0.4343637    73.73649     5.433254 0.00000000     7.733726 22.53687
## 2 0.4405113    81.28266     5.909353 0.36332686     7.589820 15.21426
## 3 0.3192113    71.59426     7.149837 0.08977425    17.525557 51.94382
## 4 0.3669262    76.74385     6.637936 0.39415148     8.163104 24.16597
## 5 0.3844914    81.82671     4.228716 0.35649281     7.699640 10.59474
## 6 0.3619592    79.09065     5.273684 0.00000000    17.890026 76.53587
```

~~~
##   CountyTotal
## 1       55221
## 2      195121
## 3       26932
## 4       22604
## 5       57710
## 6       10678
```
~~~

13. Run PCA for both county & sub-county level data. Save the first two principle components PC1 and PC2 into a two-column data frame, call it ct.pc and subct.pc, respectively. Discuss whether you chose to center and scale the features before running PCA and the reasons for your choice. What are the three features with the largest absolute values of the first principal component? Which features have opposite signs and what does that mean about the correaltion between these features?

creating pca objects

```
ct.pca <- prcomp(census.ct[3:28], center = TRUE, scale=TRUE)
subct.pca <- prcomp(census.subct[4:30], center = TRUE, scale=TRUE)
```

getting the principal components

```
ct.pc <- data.frame(ct.pca$rotation[,1:2])
subct.pc <- data.frame(subct.pca$rotation[,1:2])
```

```
top_n(abs(ct.pc[1]), 3)
```

```
## Selecting by PC1
```

```
##                   PC1
## IncomePerCap 0.3530767
## Poverty      0.3405832
## ChildPoverty 0.3421530
```

```
top_n(abs(subct.pc[1]), 3)
```

```
## Selecting by PC1
```

```
##                   PC1
## IncomePerCap 0.3184551
## Poverty      0.3043313
## Professional 0.3065537
```

```
Scele and center is used, because the columns values have a large difference and also have different mea
The top 3 prominent loadings at the county level of PC1 are IncomePerCap, Poverty and ChildPoverty for c
The top 3 prominent loadings at the county level of PC1 are IncomePerCap, Poverty and Professional for s
```

14. Determine the number of minimum number of PCs needed to capture 90% of the variance for both the county and sub-county analyses. Plot proportion of variance explained (PVE) and cumulative PVE for both county and sub-county analyses.

```
pr.var = ct.pca$sdev ^2
pve.ct = pr.var/sum(pr.var)
cumulative_pve.ct <- cumsum(pve.ct)
```

```
# This will put the next two plots side by side
par(mfrow=c(1, 2))
# Plot proportion of variance explained
plot(pve.ct, type="l", lwd=3, xlab="Principal Component",
     ylab="PVE of county", ylim =c(0,1))
```

```r
plot(cumulative_pve.ct, type="l", lwd=3, xlab="Principal Component ",
     ylab=" Cumulative PVE of county ", ylim=c(0,1))
```

```r
round(pve.ct[1:25], 2)
sum(pve.ct[1:13])
sum(pve.ct[1:14])
```

So the minimum number of PCs needed to capture 90% of the variance for the county is 14.

```r
pr.varsubct = subct.pca$sdev ^2
pve.subct = pr.varsubct/sum(pr.varsubct)
cumulative_pve.subct <- cumsum(pve.subct)
```

```r
# This will put the next two plots side by side
par(mfrow=c(1, 2))
# Plot proportion of variance explained
plot(pve.subct, type="l", lwd=3, xlab="Principal Component",
     ylab="PVE of subcounty", ylim =c(0,1))
plot(cumulative_pve.subct, type="l", lwd=3, xlab="Principal Component ",
     ylab=" Cumulative PVE of subcounty ", ylim=c(0,1))
```

```r
round(pve.subct[1:25], 2)
sum(pve.subct[1:15])
sum(pve.subct[1:16])
```

So the minimum number of PCs needed to capture 90% of the variance for the county is 16.

Clustering

15. With census.ct, perform hierarchical clustering with complete linkage. Cut the tree to partition the observations into 10 clusters. Re-run the hierarchical clustering algorithm using the first 5 principal components of ct.pc as inputs instead of the originald features. Compare and contrast the results. For both approaches investigate the cluster that contains San Mateo County. Which approach seemed to put San Mateo County in a more appropriate clusters? Comment on what you observe and discuss possible explanations for these observations.

```r
Scensus.ct <- scale(census.ct[,-c(1,2)],center =T,scale = T)
dist_census.ct <- dist(Scensus.ct)
hc.census.ct <- hclust(dist_census.ct)
hc.census.ct <- cutree(hc.census.ct, k = 10)
table(hc.census.ct)
```

```
## hc.census.ct
##    1    2    3    4    5    6    7    8    9   10
## 2632  501    6    7    5    1   11   13   38    4
```

```r
hc.census.pc <- cutree(hclust(dist(scale(data.frame(ct.pca$x[,1:5])))),k=10)
table(hc.census.pc)
```

```
## hc.census.pc
##    1    2    3    4    5    6    7    8    9   10
## 2441  525   97    6    8   31    5   18    7   80
```

```r
census.ct[227,]
```

```
##          State    County       Men    White  Citizen   Income IncomeErr
## 227 California San Mateo 0.4919773 40.63851 0.642005 100369.9  16123.02
##     IncomePerCap IncomePerCapErr  Poverty ChildPoverty Professional  Service
```

```
## 227     47881.29        6115.552 8.011122     9.705514     45.73565 18.28979
##     Office Production    Drive  Carpool  Transit OtherTransp WorkAtHome
## 227 22.304    7.34329 69.92713 10.68144 9.257082    2.598808   5.077957
##     MeanCommute  Employed PrivateWork SelfEmployed FamilyWork Unemployment
## 227    26.82681 0.5172497    79.76635     8.367532  0.1716192     6.689483
##     Minority CountyTotal
## 227 55.53405      746069
```

```
hc.census.ct[227]
```

```
## [1] 2
```

```
hc.census.pc[227]
```

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

```
hc.census.ct.df<-as.data.frame(hc.census.ct) ##change into data frame
hc.census.pc.df<-as.data.frame(hc.census.pc)
sanmateo.ct<- data.frame(hc.census.ct.df,census.ct)## combine into 1 file
sanmateo.ct <- sanmateo.ct %>%
group_by(hc.census.ct) ## group the census.ct data according to cluster id
head(sanmateo.ct)
```

```
## # A tibble: 6 x 29
## # Groups:   hc.census.ct [1]
##   hc.census.ct State County   Men White Citizen Income IncomeErr IncomePerCap
##          <int> <chr> <chr>  <dbl> <dbl>   <dbl>  <dbl>     <dbl>        <dbl>
## 1            1 Alab~ Autau~ 0.484  75.8   0.737 51696.     7771.       24974.
## 2            1 Alab~ Baldw~ 0.488  83.1   0.757 51074.     8745.       27317.
## 3            1 Alab~ Barbo~ 0.538  46.2   0.769 32959.     6031.       16824.
## 4            1 Alab~ Bibb   0.534  74.5   0.774 38887.     5662.       18431.
## 5            1 Alab~ Blount 0.494  87.9   0.734 46238.     8696.       20532.
## 6            1 Alab~ Bullo~ 0.530  22.2   0.755 33293.     9000.       17580.
## # ... with 20 more variables: IncomePerCapErr <dbl>, Poverty <dbl>,
## #   ChildPoverty <dbl>, Professional <dbl>, Service <dbl>, Office <dbl>,
## #   Production <dbl>, Drive <dbl>, Carpool <dbl>, Transit <dbl>,
## #   OtherTransp <dbl>, WorkAtHome <dbl>, MeanCommute <dbl>, Employed <dbl>,
## #   PrivateWork <dbl>, SelfEmployed <dbl>, FamilyWork <dbl>,
## #   Unemployment <dbl>, Minority <dbl>, CountyTotal <dbl>
```

```
sanmateo.pc<- data.frame(hc.census.pc.df,census.ct)## combine into 1 file
sanmateo.pc <- sanmateo.pc %>%
group_by(hc.census.pc) ## group the census.ct data according to cluster id
head(sanmateo.pc)
```

```
## # A tibble: 6 x 29
## # Groups:   hc.census.pc [2]
##   hc.census.pc State County   Men White Citizen Income IncomeErr IncomePerCap
##          <int> <chr> <chr>  <dbl> <dbl>   <dbl>  <dbl>     <dbl>        <dbl>
## 1            1 Alab~ Autau~ 0.484  75.8   0.737 51696.     7771.       24974.
## 2            1 Alab~ Baldw~ 0.488  83.1   0.757 51074.     8745.       27317.
## 3            2 Alab~ Barbo~ 0.538  46.2   0.769 32959.     6031.       16824.
## 4            2 Alab~ Bibb   0.534  74.5   0.774 38887.     5662.       18431.
## 5            1 Alab~ Blount 0.494  87.9   0.734 46238.     8696.       20532.
## 6            2 Alab~ Bullo~ 0.530  22.2   0.755 33293.     9000.       17580.
## # ... with 20 more variables: IncomePerCapErr <dbl>, Poverty <dbl>,
## #   ChildPoverty <dbl>, Professional <dbl>, Service <dbl>, Office <dbl>,
```

```
## #   Production <dbl>, Drive <dbl>, Carpool <dbl>, Transit <dbl>,
## #   OtherTransp <dbl>, WorkAtHome <dbl>, MeanCommute <dbl>, Employed <dbl>,
## #   PrivateWork <dbl>, SelfEmployed <dbl>, FamilyWork <dbl>,
## #   Unemployment <dbl>, Minority <dbl>, CountyTotal <dbl>
```

```
In census.ct, San Mateo has index at row no.227 so we look for no.227 in the cluster lists of census.ct

When using census.ct, the county San Mateo is placed into cluster 2. But when using the first five prin

When using clustering, we want the clusters that have been found to represent true subgroups in the dat

To look at the cluster as a whole to see the association with other county, i.e, which counties are gro
```

Classification

In order to train classification models, we need to combine county_winner and census.ct data. This seemingly straightforward task is harder than it sounds. Following code makes necessary changes to merge them into election.cl for classification.

```r
tmpwinner <- county_winner %>% ungroup %>%
  mutate(state = state.name[match(state, state.abb)]) %>%               ## state abbreviations
  mutate_at(vars(state, county), tolower) %>%                           ## to all lowercase
  mutate(county = gsub(" county| columbia| city| parish", "", county))  ## remove suffixes
tmpcensus <- census.ct %>% mutate_at(vars(State, County), tolower)

election.cl <- tmpwinner %>%
  left_join(tmpcensus, by = c("state"="State", "county"="County")) %>%
  na.omit

## save meta information
election.meta <- election.cl %>% select(c(county, fips, state, votes, pct, total))

## save predictors and class labels
election.cl = election.cl %>% select(-c(county, fips, state, votes, pct, total))
```

Using the following code, partition data into 80% training and 20% testing:

```r
set.seed(10)
n <- nrow(election.cl)
in.trn <- sample.int(n, 0.8*n)
trn.cl <- election.cl[ in.trn,]
tst.cl <- election.cl[-in.trn,]
```

```r
set.seed(20)
nfold <- 10
folds <- sample(cut(1:nrow(trn.cl), breaks=nfold, labels=FALSE))
```

```r
calc_error_rate = function(predicted.value, true.value){
  return(mean(true.value!=predicted.value))
}
records = matrix(NA, nrow=3, ncol=2)
colnames(records) = c("train.error","test.error")
rownames(records) = c("tree","logistic","lasso")
```

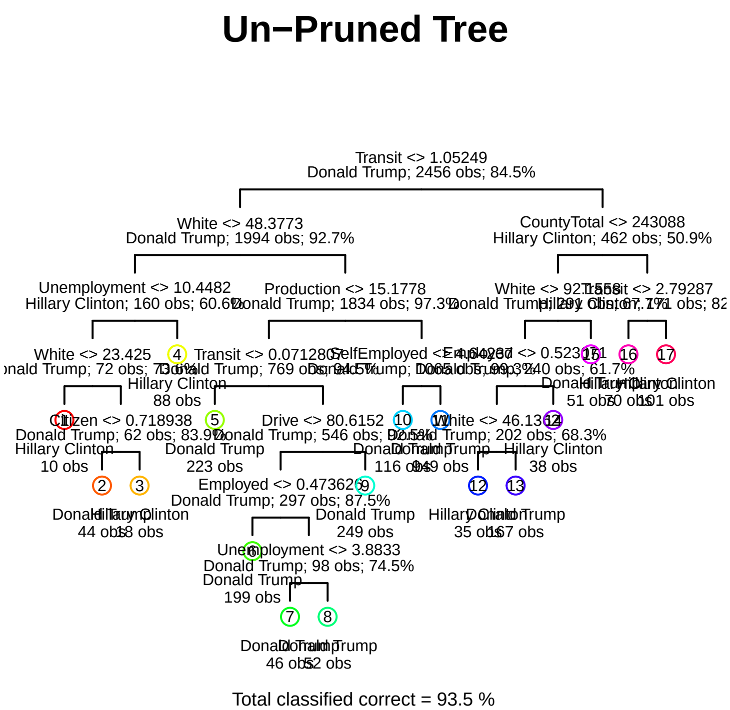
Classification

16. Decision tree: train a decision tree by cv.tree(). Prune tree to minimize misclassification error. Be sure to use the folds from above for cross-validation. Visualize the trees before and after pruning. Save training and test errors to records variable. Intepret and discuss the results of the decision tree analysis. Use this plot to tell a story about voting behavior in the US (remember the NYT infographic?)

```
cv.tree <- tree(candidate~., data = trn.cl)
summary(cv.tree)
```

```
##
## Classification tree:
## tree(formula = candidate ~ ., data = trn.cl)
## Variables actually used in tree construction:
## [1] "Transit"      "White"        "Unemployment" "Citizen"      "Production"
## [6] "Drive"        "Employed"     "SelfEmployed" "CountyTotal"
## Number of terminal nodes:  17
## Residual mean deviance:  0.325 = 792.6 / 2439
## Misclassification error rate: 0.06474 = 159 / 2456
```

```
draw.tree(cv.tree, cex=.5, nodeinfo=TRUE)
title("Un-Pruned Tree")
```



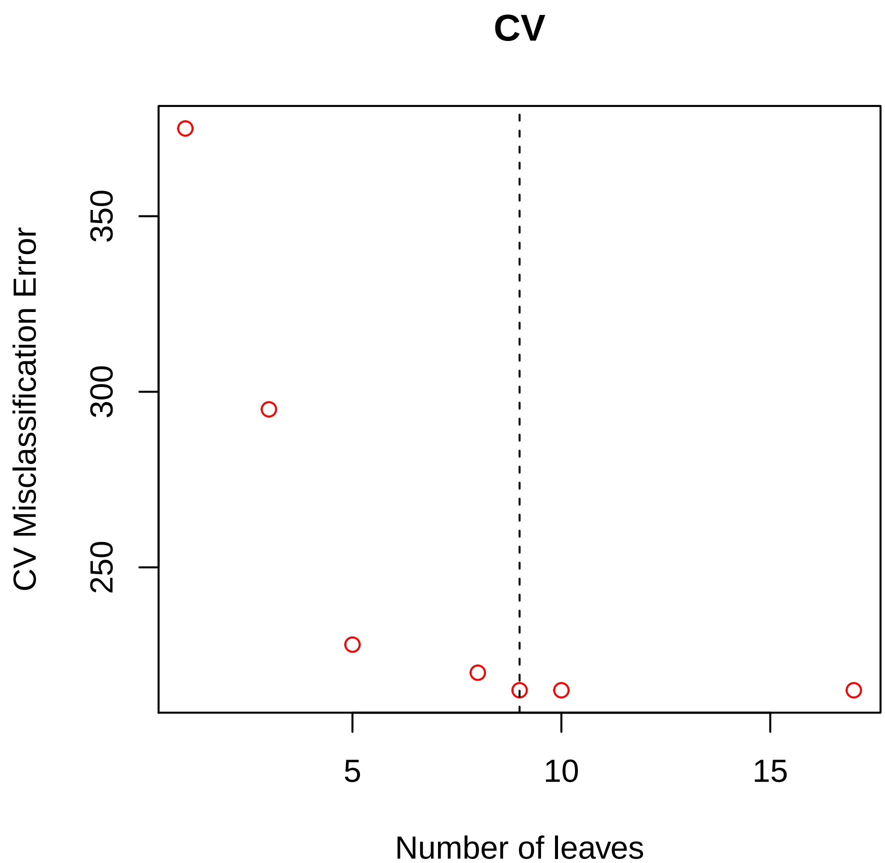
Prune

```
cv <- cv.tree(cv.tree, folds, method='misclass' )
# Best size
best.size.cv = min(cv$size[which(cv$dev == min(cv$dev))])
best.size.cv
```

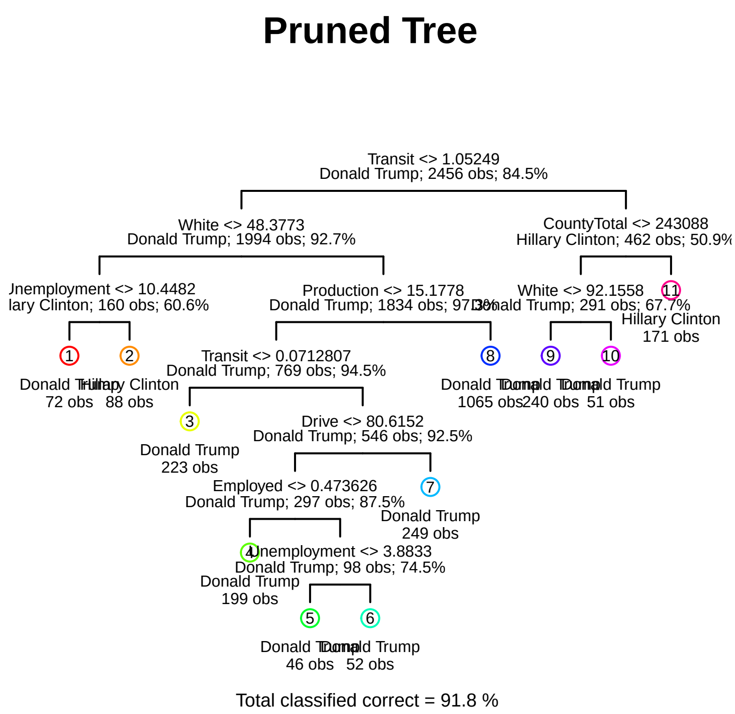
```
## [1] 9
```

```
# Plot size vs. cross-validation error rate
plot(cv$size , cv$dev,
xlab = "Number of leaves", ylab = "CV Misclassification Error", col = "red", main="CV")
```

```
abline(v=best.size.cv, lty=2)
```




```
pruned <- prune.tree(cv.tree, best=best.size.cv)
draw.tree(pruned, cex=.5, nodeinfo=TRUE)
title("Pruned Tree")
```




```
pred.test_tree = predict(pruned, tst.cl, type="class")
calc_error_rate(pred.test_tree, tst.cl$candidate)
```

```
## [1] 0.07804878
```

```
# Predict on training set
pred.train_tree = predict(pruned, trn.cl, type="class")
calc_error_rate(pred.train_tree, trn.cl$candidate)
```

```
## [1] 0.08224756
```

```
records[1,] = c(calc_error_rate(pred.train_tree,trn.cl$candidate),calc_error_rate(pred.test_tree, tst.cl
records
```

```
##          train.error test.error
## tree      0.08224756 0.07804878
## logistic          NA         NA
## lasso             NA         NA
```

```
From the result, we can see that:
Donald Trump won the vast maiority of commuting on public transportation or counties with more white pe
```

17. Run a logistic regression to predict the winning candidate in each county. Save training and test errors to records variable. What are the significant variables? Are the consistent with what you saw in decision tree analysis? Interpret the meaning of a couple of the significant coefficients in terms of a unit change in the variables.

```
logistic_log <- glm(candidate~., data = trn.cl, family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
pre.log_train = ifelse(predict(logistic_log, type = "response") > 0.5, "Hillary Clinton", "Donald Trump
pre.log_test = ifelse(predict(logistic_log, tst.cl, type = "response") > 0.5, "Hillary Clinton", "Donalc
coef_unpenalized = data.frame(logistic_log$coefficients)
coef_unpenalized
```

```
##                  logistic_log.coefficients
## (Intercept)                  -2.487078e+01
## Men                           9.589251e+00
## White                        -1.688549e-01
## Citizen                       1.302148e+01
## Income                       -8.708481e-05
## IncomeErr                    -3.325814e-06
## IncomePerCap                  2.670812e-04
## IncomePerCapErr              -3.604893e-04
## Poverty                       4.741402e-02
## ChildPoverty                 -1.582269e-02
## Professional                  2.802314e-01
## Service                       3.242049e-01
## Office                        7.590227e-02
## Production                    1.668326e-01
## Drive                        -2.096881e-01
## Carpool                      -1.735883e-01
## Transit                       7.577993e-02
## OtherTransp                  -6.257976e-02
## WorkAtHome                   -1.657167e-01
## MeanCommute                   5.615662e-02
## Employed                      2.056419e+01
## PrivateWork                   1.010086e-01
## SelfEmployed                  1.968066e-02
## FamilyWork                   -8.873246e-01
## Unemployment                  2.072704e-01
## Minority                     -3.053242e-02
## CountyTotal                   3.511207e-07
```

```
top_n(abs(coef_unpenalized), 5)
```

```
## Selecting by logistic_log.coefficients
```

```
##              logistic_log.coefficients
## (Intercept)                 24.8707796
## Men                          9.5892512
## Citizen                     13.0214787
## Employed                    20.5641902
## FamilyWork                   0.8873246
```

```
From the top 4 significant coefficients: Men, Citizen, Employed, FamilyWork plays a significant role in
Interpret the meaning of a couple of the significant coefficients in terms of a unit change in the vari
```

```
log_train_error = calc_error_rate(pre.log_train, trn.cl$candidate)
log_test_error = calc_error_rate(pre.log_test, tst.cl$candidate)
records[2,] = c(log_train_error, log_test_error)
records
```

```
##          train.error test.error
## tree      0.08224756 0.07804878
## logistic  0.07043974 0.06341463
## lasso             NA         NA
```

18. You may notice that you get a warning glm.fit: fitted probabilities numerically 0 or 1 occurred. As we discussed in class, this is an indication that we have perfect separation (some linear combination of variables perfectly predicts the winner). This is usually a sign that we are overfitting. One way to control overfitting in logistic regression is through regularization. Use the cv.glmnet function from

the glmnet library to run K-fold cross validation and select the best regularization parameter for the logistic regression with LASSO penalty. Reminder: set α=1 to run LASSO regression, set λ = c(1, 5, 10, 50) * 1e-4 in cv.glmnet() function to set pre-defined candidate values for the tuning parameter λ. This is because the default candidate values of λ in cv.glmnet() is relatively too large for our dataset thus we use pre-defined candidate values. What is the optimal value of λ in cross validation? What are the non-zero coefficients in the LASSO regression for the optimal value of λ ? How do they compare to the unpenalized logistic regression? Save training and test errors to the records variable.

```
library(glmnet)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-1
```

```
lasso_log = cv.glmnet(x = data.matrix(select(trn.cl, -candidate)), y = data.matrix( select(trn.cl, candi
lasso_log$lambda.min
```

```
## [1] 5e-04
```

```
So the optimal value of $\lambda$ in cross validation is 5e-04
```

```
coef_lasso = coef(lasso_log, lasso_log$lambda.min)
coef_lasso
```

```
## 27 x 1 sparse Matrix of class "dgCMatrix"
##                            1
## (Intercept)     -2.641036e+01
## Men              6.954729e+00
## White           -1.295344e-01
## Citizen          1.356411e+01
## Income          -6.089664e-05
## IncomeErr       -1.301761e-05
## IncomePerCap     2.036079e-04
## IncomePerCapErr -2.535308e-04
## Poverty          3.434226e-02
## ChildPoverty    -1.915472e-03
## Professional     2.528782e-01
## Service          2.924197e-01
## Office           5.282736e-02
## Production       1.356323e-01
## Drive           -1.799961e-01
## Carpool         -1.426987e-01
## Transit          9.330294e-02
## OtherTransp     -2.116718e-02
## WorkAtHome      -1.236069e-01
## MeanCommute      3.960605e-02
## Employed         1.943018e+01
## PrivateWork      9.292939e-02
## SelfEmployed     .
## FamilyWork      -7.510898e-01
## Unemployment     1.941987e-01
```

```
## Minority         .
## CountyTotal      4.056588e-07
```

Above is the non-zero coefficients in the LASSO regression for the optimal value except for SelfEmployed

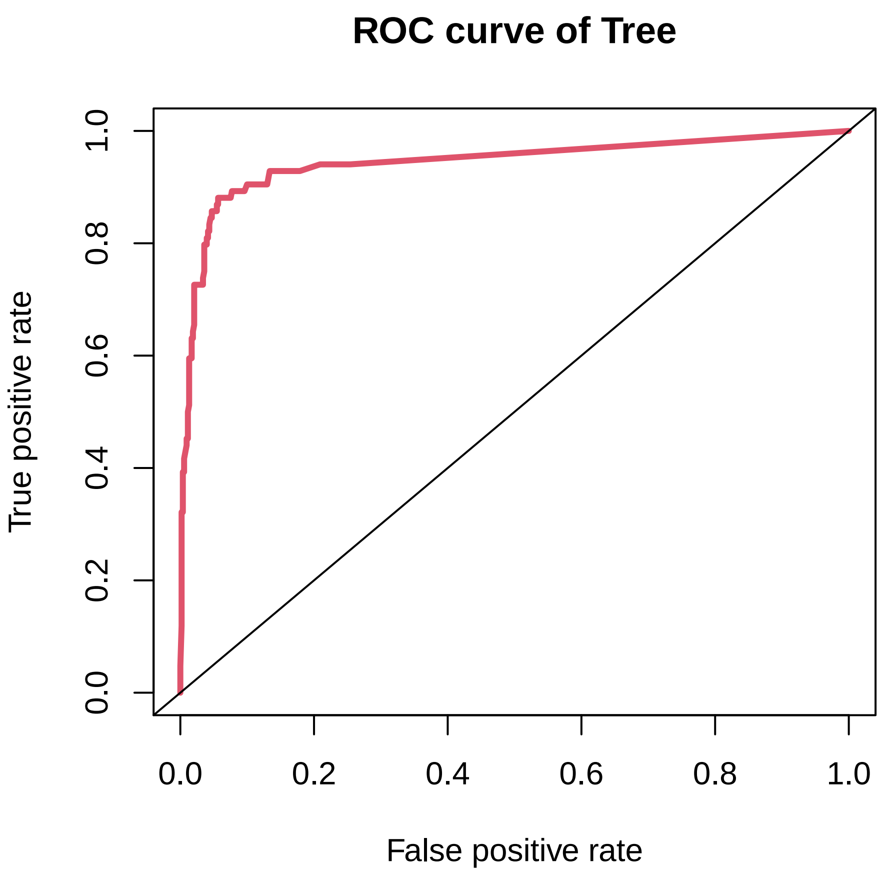
```
pre.lasso_train = ifelse(predict(lasso_log, s = "lambda.min", newx = data.matrix(select(trn.cl, -candida
pre.lasso_test = ifelse(predict(lasso_log, s = "lambda.min", newx = data.matrix(select(tst.cl, -candidat

log_train_error = calc_error_rate(pre.lasso_train, trn.cl$candidate)
log_test_error = calc_error_rate(pre.lasso_test, tst.cl$candidate)
records[3,] = c(log_train_error, log_test_error)
records
```

```
##          train.error test.error
## tree      0.08224756 0.07804878
## logistic  0.07043974 0.06341463
## lasso     0.06962541 0.06504065
```

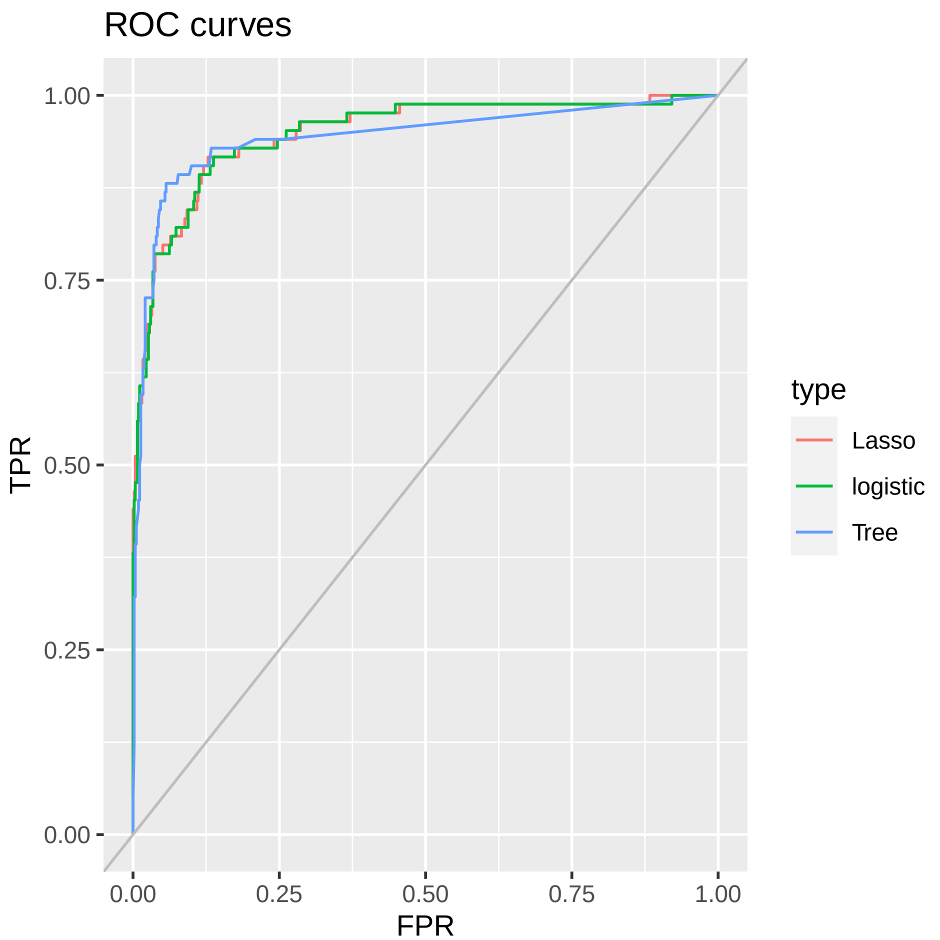
19. Compute ROC curves for the decision tree, logistic regression and LASSO logistic regression using predictions on the test data. Display them on the same plot. Based on your classification results, discuss the pros and cons of the various methods. Are the different classifiers more appropriate for answering different kinds of questions about the election?

Since decesion tree does not explictly output a probability for class labels, we use 200 bootstrap replicates of the testing data to predict the class label.

```
set.seed(8)
test2 <- tst.cl
train2 <- trn.cl

for(i in 1:200){
  train_index <- sample(nrow(trn.cl), replace = TRUE)
  nfold <- 10
  folds <- sample(cut(1:nrow(trn.cl[train_index, ]), breaks=nfold, labels=FALSE))

  cv.tree <- tree(candidate~., data = trn.cl[train_index, ])
  cv <- cv.tree(cv.tree, folds, method='misclass' )
  pruned <- prune.tree(cv.tree, best=best.size.cv)
  precandidate = as.numeric(predict(pruned, tst.cl, type='class')) -1
  test2 <- cbind(test2, precandidate)
}

test2 <- rowSums(test2[, 28:227])
probs <- test2/200

reals = ifelse(tst.cl$candidate == "Hillary Clinton", 1, 0)

predCandidate <- prediction(probs, reals)

perfCandidate = performance(predCandidate, measure="tpr", x.measure="fpr")

plot(perfCandidate, col=2, lwd=3, main="ROC curve of Tree")
abline(0,1)
```




```
test_label = ifelse(tst.cl$candidate == "Hillary Clinton", 1, 0)
pre.log_test = predict(logistic_log, type = "response", tst.cl)
pred_log = prediction(pre.log_test, test_label)
perf_log = performance(pred_log, measure = "tpr", x.measure = "fpr")
log_data = data.frame(x = perf_log@x.values[[1]], y = perf_log@y.values[[1]], type = "logistic")

pre.lasso_test = predict(lasso_log, s = "lambda.min", newx = data.matrix(select(tst.cl, -candidate)), t
pred_lasso = prediction(pre.lasso_test, test_label)
perf_lasso = performance(pred_lasso, measure = "tpr", x.measure = "fpr")
lasso_data = data.frame(x = perf_lasso@x.values[[1]], y = perf_lasso@y.values[[1]], type = "Lasso")

tree_data = data.frame(x = perfCandidate@x.values[[1]], y = perfCandidate@y.values[[1]], type = "Tree")
data_combined = rbind(log_data, lasso_data, tree_data)

#plot ROC
ggplot(data= data_combined, aes(x=x, y=y, color=type)) +
  geom_line() +
  geom_abline(intercept = 0, slope = 1, color = "grey", size = 0.5)+
  labs(title = "ROC curves", x="FPR", y="TPR")
```




```
tree_auc = performance(predCandidate, "auc")@y.values
logistic_auc = performance(pred_log, "auc")@y.values
lasso_auc = performance(pred_lasso, "auc")@y.values
data2 = data.frame(tree=tree_auc[[1]], logistic = logistic_auc[[1]], lasso=lasso_auc[[1]])
row.names(data2) = c("AUC")
data2
```

```
##          tree  logistic     lasso
## AUC 0.9432674 0.9482782 0.9488387
```

```
As the significant cofficients are quite different for different methods, different classifiers are more
From the AUC results, lasso has higher AUC value so lasso should be perferred;
```

Taking it further

20. This is an open question. Interpret and discuss any overall insights gained in this analysis and possible explanations. Use any tools at your disposal to make your case: visualize errors on the map, discuss what does/doesn't seems reasonable based on your understanding of these methods, propose possible directions (collecting additional data, domain knowledge, etc). In addition, propose and tackle at least one more interesting question.