castillo_anthony_proj_131

Anthony Castillo

12/5/2019

1) Voter data can be hard to predict because data gathered for measuring how voters view candidates and ballot measures is usually deeply inaccurate. Between faulty polling methodologies (ie - pollsters contacting voters using only landlines, and thus contacting only older voters), terrible wording of questions, and polling agencies with a political agendy in mind, it's a wonder that people take polls seriously. In addition, people may lie to a pollster to evade facing judgement, or may even experience a change of heart in political opinion altogether from between the polling and actual voting times. Thus, it becomes really difficult to predict voter data.

2) Part of what went right for Nate Silver in 2012 was actually the Republican Party and the US economy going through some structural changes at the time. Namely, the GOP was in the middle of the Tea Party movement, and thus was moving to the right on many issues. Inconveniently for the Tea Party, Romney, one of the last few moderates in the GOP (he was actually pro-choice in his Senate run back in the 90s) got the nomination after Speaker Gingrich, Senator Santorum, and Congressman Paul all were defeated in the 2012 GOP primary. The GOP base (the Tea Party) was disgruntled by this even though the economy had not recovered from the '08 recession. President Obama had 4 years to produce change and benefits for the American people who were effected by the recession and failed in the eyes of many, and thus many within the media establishment weren't sure on Obama winning re-eleciton. Thus, Romney blew millions of dollars surrounding himself with focus groups and media consultants who reassured him the presidency was his, and there wasn't much he had to do about it. Little did Romney know that national polling means little in a presidential election, and that because of the structure of the Electoral College, state polling is largely what counts.

By relating voter opinion to time on a state-by-state basis, Nate Silver was able to predict every single state in the 2012 election. Mixing Bayes' Theorem, hierarchical modelling, and graph theory, Nate Silver was able to successfully predict the 2012 election with ease. It largely revolved around centering his predictions on how Ohio was going to vote. In political terms, the Democrats hadn't yet lost the support of the white working class that voted Trump in 2016, and this demographic can be seen primarily in the Rust Belt. Romney, being a white-collar financier, wouldn't have the appeal unionists in the then-relevant Obama coalition had to those voters. Thus, by Obama winning Ohio, he likely would have also made the appeal to win the swing states necessary for 270 votes in the electoral college.

3) Actually, nothing went wrong in 2016. The Obama coalition no longer exists, the white working class in the Rust Belt felt cheated by the neoliberal globalism pushed by 2nd-term Obama, and then-candidate Trump simply capitalized on Obama's failures as president and Secretary Clinton's failures as America's top diplomat. Simply put, Trump saw an opportunity to push his more paleoconservative/nationalist platform through appealing to the American heartland while maintaining appeal to the religious right, the remnants of the Tea Party, and any Senator Sanders supporters who were disillusioned by the 2016 Democratic primary. I would go as far as say that Trump may have even won in 2012 had he run then, and my evidence is the fact that Obama won 2012 by only 64 electoral votes scattered across four swing states: Florida, Ohio, Virginia, and New Hampshire. I researched the actual voting margins and found Romney only needed 429,464 votes across these four states to win the 270 electoral votes for the presidency. Trump added 2 million votes to the GOP column with Florida and the Rust Belt making up the Electoral College difference. Moreover, I am one of the few people you will ever meet that called the 2016 election well before it happened.

Polling can be made better by divorcing itself from all biases and policy agendas and instead diversifying outreach to voters. This means issuing polls by email or cell phone call. I do not think this will happen, and thus the actual predictions themselves can be made better if we focus more on state-wide sentiments as it pertains to major policy issues. I do see the Sun Belt coming into play in the near future, and I think we can experiment there by testing analytics methods for unstructured data likely voters may yield while on the Internet (ie - scraping keystroke and page visit time data from whatever API is relevant).

```r
library(dplyr)
```

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

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

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

```r
library(readr)
library(knitr)
library(kableExtra)
```

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

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

```r
## read data and convert candidate from string to factor
election.raw <- read_delim("election.csv", delim = ",") %>%
  mutate(candidate=as.factor(candidate))
```

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

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

```
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_character(),
##   X3 = col_character()
## )
```

```r
census <- read_delim("census.csv", delim = ",")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   State = col_character(),
```

```
##   County = col_character()
## )
## See spec(...) for full column specifications.
```

```r
kable(election.raw %>% filter(county == "Los Angeles County"))  %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
  full_width=FALSE)
```

county	fips	candidate	state	votes
Los Angeles County	6037	Hillary Clinton	CA	2464364
Los Angeles County	6037	Donald Trump	CA	769743
Los Angeles County	6037	Gary Johnson	CA	88968
Los Angeles County	6037	Jill Stein	CA	76465
Los Angeles County	6037	Gloria La Riva	CA	21993

4) We exclude fips = 2000 because it is a duplicate for Arkansas, and thus are unncecessary. We are then left with the following dimensions for election.raw.

```r
# 4
election.raw <- filter(election.raw, fips != 2000)
dim(election.raw)
```

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

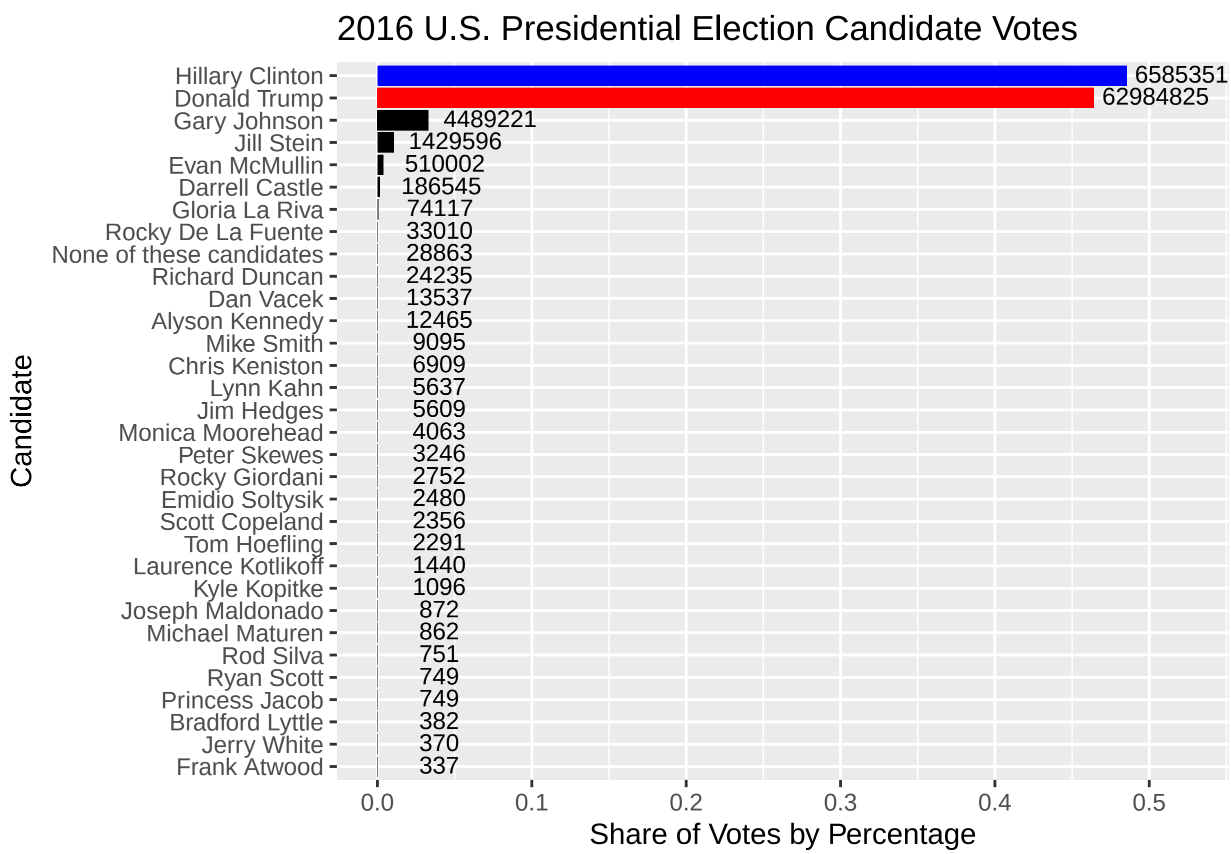
```r
# 18345 observations, 5 columns
```

5) This is just us filtering our data

```r
# 5
election <- filter(election.raw, !is.na(county))
election_federal <- filter(election.raw, fips == "US")
election_state <- filter(election.raw, fips != "US" & is.na(county))
election <- rbind(election, election_state[309:312,])
```

6) Here, we have total population vote count for all candidates.

```r
# 6
Candidate_Votes <- (election_federal %>% select(candidate, votes))
Candidate_Votes <- Candidate_Votes[order(Candidate_Votes$votes),]
candidate.ordered <-  factor(Candidate_Votes$candidate, levels =
                             as.vector(Candidate_Votes$candidate))
library(ggplot2)
Candidate_Votes <- Candidate_Votes %>% mutate(percentage = votes/sum(votes),
                                              candidate = candidate.ordered)
ggplot(Candidate_Votes, aes(candidate, percentage)) +
  geom_col(fill = c(rep("black", times = nrow(Candidate_Votes) - 2), "red", "blue")) +
  coord_flip() + labs(title = "2016 U.S. Presidential Election Candidate Votes",
                      x = "Candidate", y = "Share of Votes by Percentage") +
  geom_text(aes(label=votes), size = 3, nudge_y = 0.04, nudge_x = 0.08) +
  guides("Legend", nrow = 3, ncol = 2 )
```



7) Here, we create our county_winner and state_winner objects (no output).

```
# 7
county.group <- group_by(election, fips)
total.group <- dplyr::summarize(county.group, total = sum(votes))
count.group <- left_join(county.group, total.group, by = "fips")
county.pct <- mutate(count.group, pct = votes/total)
county_winner <- top_n(county.pct, n =1)
```

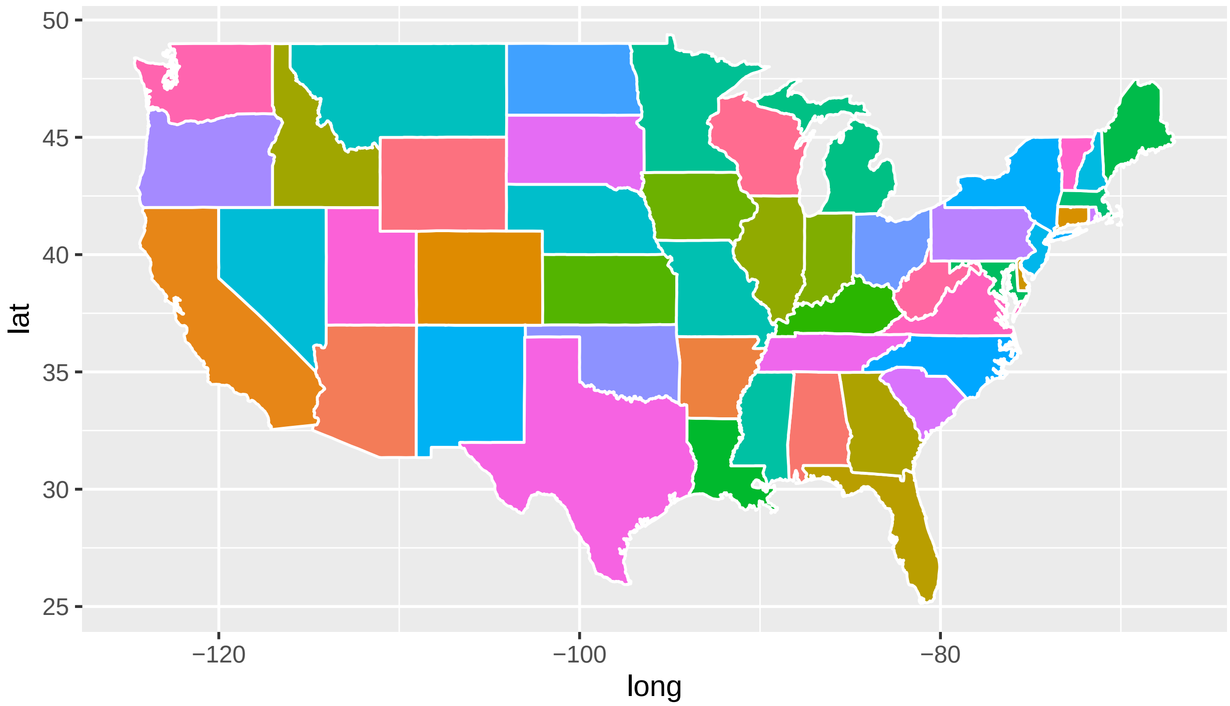
```
## Selecting by pct
```

```
state.group <- group_by(election_state, state)
total.stqte <- dplyr::summarize(state.group, total = sum(votes))
join.state <- left_join(state.group, total.stqte, by = "state")
state.pct <- mutate(join.state, pct = votes/total)
state_winner <- top_n(state.pct, n= 1)
```

```
## Selecting by pct
```

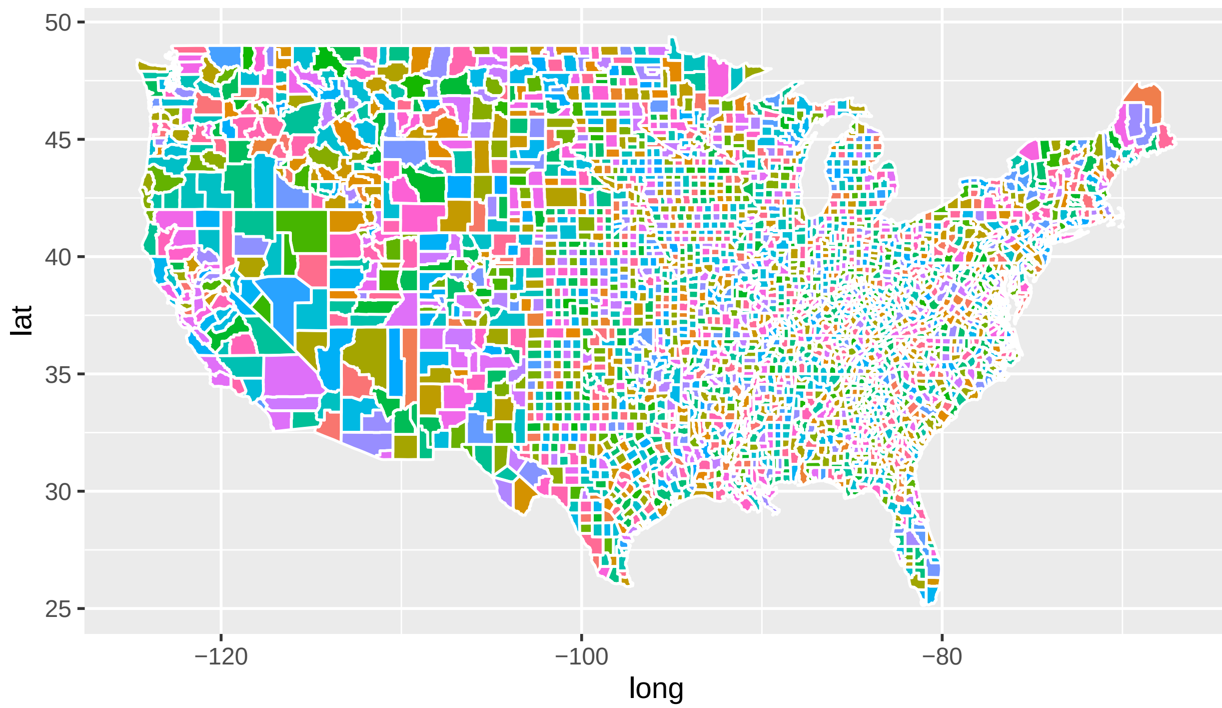
This is our state map.

```
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
```



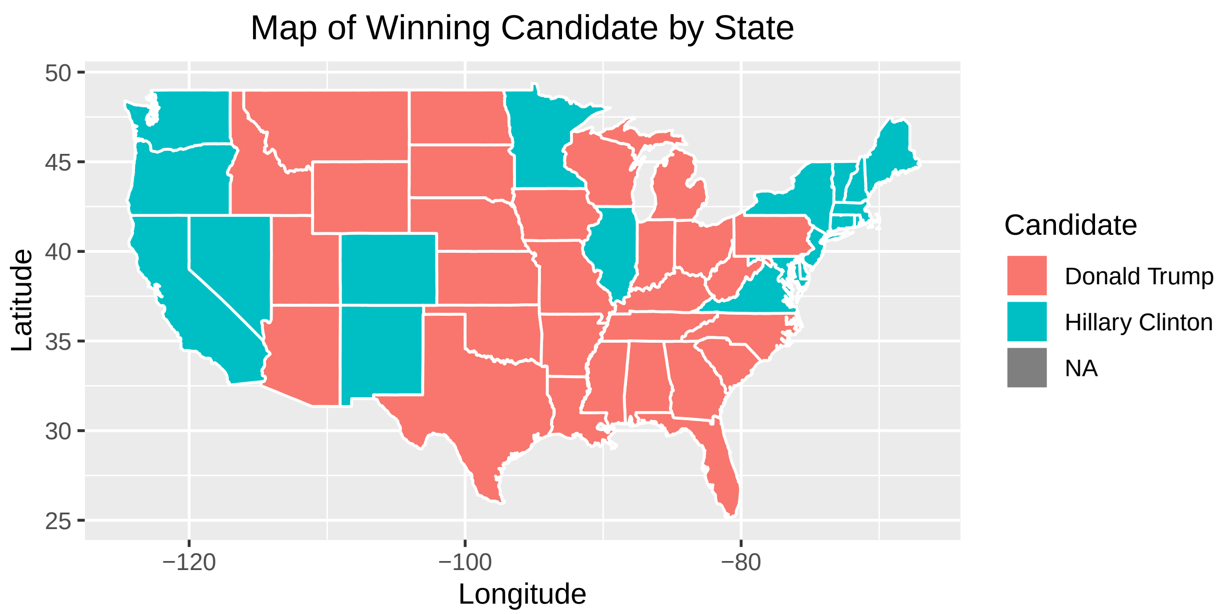
8) This is our county map.

```
# 8
county = map_data("county")
ggplot(data = county) +
  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) This is the state map in accordance with the candidate that won each state in the 2016 presidential election.

```
# 9
states = map_data("state")
fips = state.abb[match(states$region, casefold(state.name))]
states$region <- fips
new <- left_join(states, state_winner, by = c("region" = "fips" ))
ggplot(data = new) +
  geom_polygon(aes(x = long, y = lat, fill = candidate, group = group), color =
                "white") +
  coord_fixed(1.3)+
  ggtitle("Map of Winning Candidate by State")+
  labs(y="Latitude", x = "Longitude")+
  guides(fill=guide_legend(title="Candidate"))+
  theme(plot.title = element_text(hjust = 0.5))
```



10) Here, we create our fips field for county. I then threw in a plot showing county- by-county results for the election on a visual basis.

```
# 10
county = map_data("county")
county.str <- maps::county.fips
y <- unlist(strsplit(county.str$polyname, ","))
region <- NULL
subregion <-  NULL
for(i in seq(1,length(y), by = 2)){
  region <- c(region, y[i])}
for(i in seq(2,length(y), by = 2)){
  subregion <- c(subregion, y[i])}
county.str <- cbind(county.str, region)
county.str <- cbind(county.str, subregion)
county.str <- county.str[,c(1,3,4)]
county <- left_join(county, county.str, by = c("region","subregion"))
```

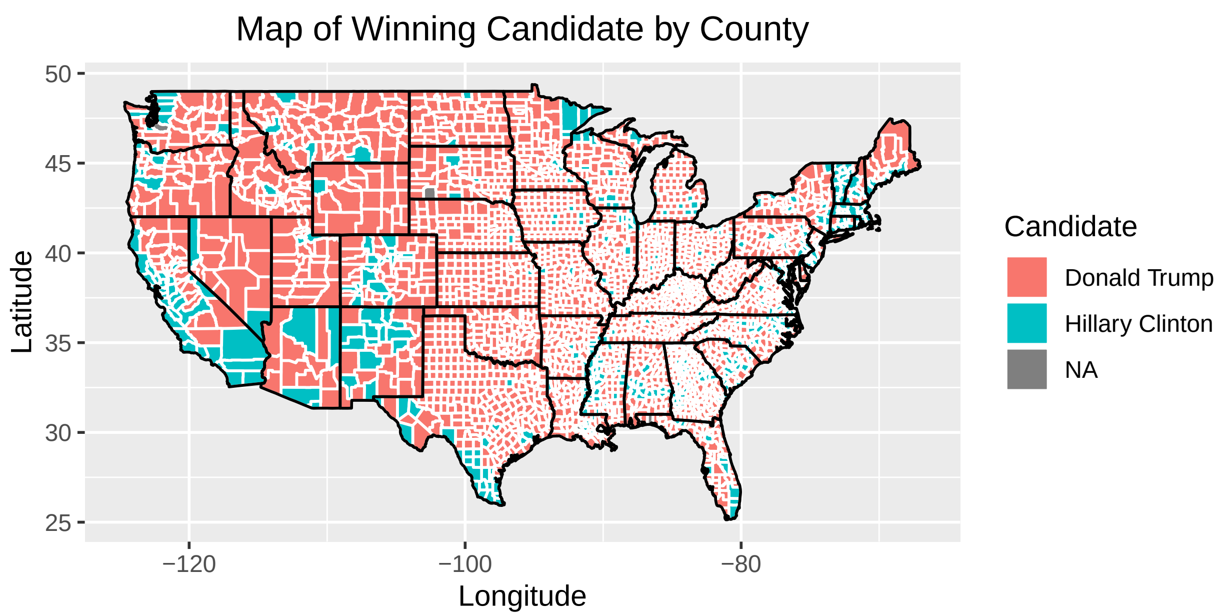
```
## Warning: Column `region` joining character vector and factor, coercing into
## character vector

## Warning: Column `subregion` joining character vector and factor, coercing into
## character vector
```

```
county$fips <- as.factor(county$fips)
county <- left_join(county, county_winner, by = "fips")
```

```
## Warning: Column `fips` joining factor and character vector, coercing into
```

```
## character vector
```

```r
ggplot(data = county) +
  geom_polygon(aes(x = long, y = lat, fill = candidate, group = group), color =
                 "white") +
  coord_fixed(1.3)+
  ggtitle("Map of Winning Candidate by County")+
  labs(y="Latitude", x = "Longitude")+
  guides(fill=guide_legend(title="Candidate"))+
  theme(plot.title = element_text(hjust = 0.5))+
  geom_path(aes(x = states$long, y = states$lat, group = group), data = states ,
            colour = "black")
```



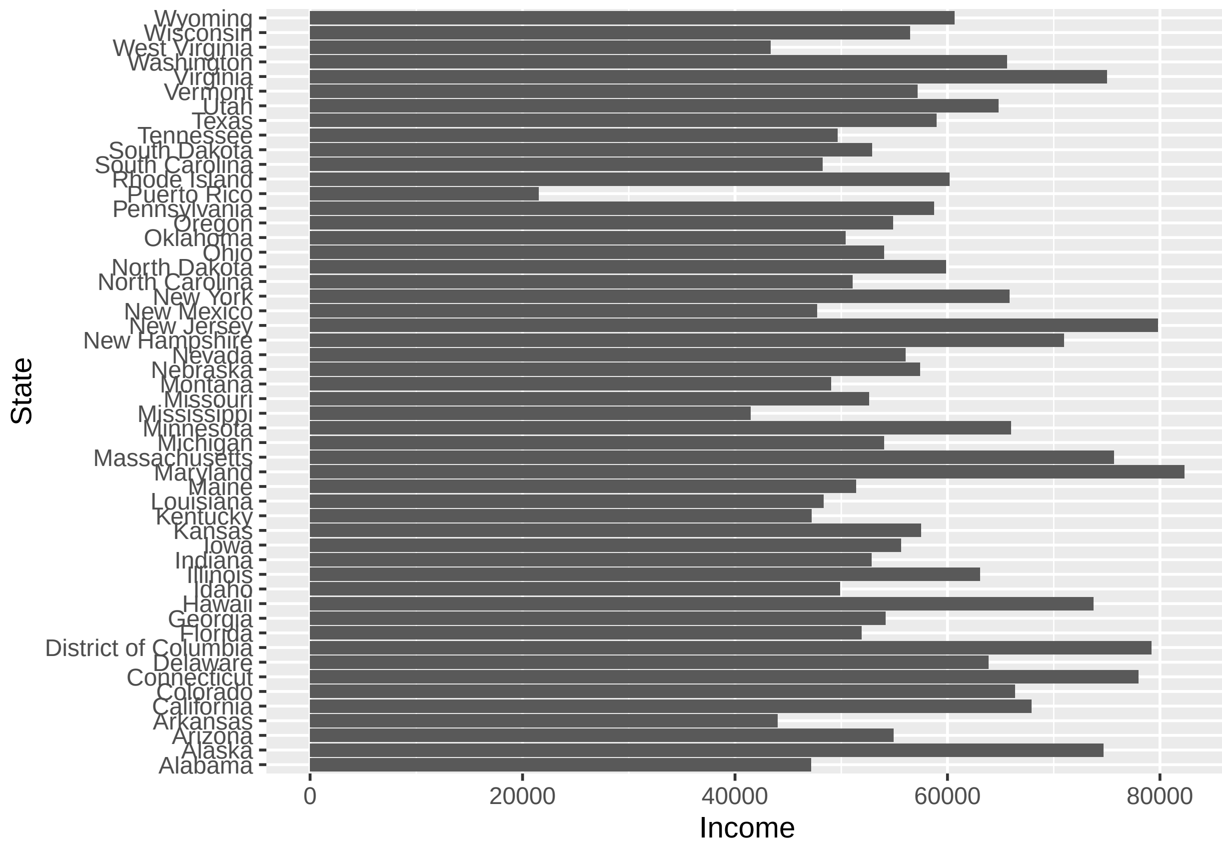
11) This visual is actually really simple. All it shows is average income by state.

```r
# 11
plot.10 <- na.omit(census)
plot.10 <- plot.10 %>% group_by(State) %>% add_tally(TotalPop)
plot.10 <- cbind(plot.10, Weight = plot.10$TotalPop/plot.10$n )
plot.10 <- plot.10 %>% group_by(State) %>% summarise_at(vars(Income), funs(sum(. * Weight)))
```

```
## Warning: funs() is soft 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 per session.
```

```r
ggplot(plot.10, aes(x=State, y=Income)) + geom_bar(stat = "identity") + coord_flip()
```



12) Here we cleaned our data and created census.del, census.subct, and census.ct.

```r
# 12
census.del <- census
census.del <- census.del[complete.cases(census.del),]
census.del <- census.del %>%
  mutate(Men = 100*Men/TotalPop,
         Employed = 100*Employed/TotalPop,
         Citizen = 100*Citizen/TotalPop)
census.del <- census.del %>% mutate(Minority =
              Hispanic + Black + Native + Asian +  Pacific) %>%
              select(-Hispanic, -Black, -Native, -Asian, -Pacific)
census.del <- census.del[c(1:7, ncol(census.del), 8:(ncol(census.del)-1))]
census.del <- select(census.del, -Walk, -PublicWork, -Construction)
census.del <- census.del %>% select(-Women,-White)
head(census.del)
```

```
## # A tibble: 6 x 28
##   CensusTract State County TotalPop   Men Minority Citizen Income IncomeErr
##         <dbl> <chr> <chr>     <dbl> <dbl>    <dbl>   <dbl>  <dbl>     <dbl>
```

```
## 1  1001020100 Alab… Autau…     1948  48.3      9.5     77.2  61838     11900
## 2  1001020200 Alab… Autau…     2156  49.1     56.4     77.1  32303     13538
## 3  1001020300 Alab… Autau…     2968  46.0     20.8     78.7  44922      5629
## 4  1001020400 Alab… Autau…     4423  49.1     15.8     74.7  54329      7003
## 5  1001020500 Alab… Autau…    10763  45.7     29.3     71.2  51965      6935
## 6  1001020600 Alab… Autau…     3851  46.4     25       68.6  63092      9585
## # … with 19 more variables: IncomePerCap <dbl>, 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>
```

```r
census.subct <- group_by(census.del, State, County)
census.subct <- add_tally(census.subct)
names(census.subct)[ncol(census.subct)] <- "CountyTotal"
census.subct <- mutate(census.subct, CountyWeight = TotalPop/CountyTotal)
head(census.subct)
```

```
## # A tibble: 6 x 30
## # Groups:   State, County [1]
##   CensusTract State County TotalPop   Men Minority Citizen Income IncomeErr
##         <dbl> <chr> <chr>     <dbl> <dbl>    <dbl>   <dbl>  <dbl>     <dbl>
## 1  1001020100 Alab… Autau…     1948  48.3      9.5     77.2  61838     11900
## 2  1001020200 Alab… Autau…     2156  49.1     56.4     77.1  32303     13538
## 3  1001020300 Alab… Autau…     2968  46.0     20.8     78.7  44922      5629
## 4  1001020400 Alab… Autau…     4423  49.1     15.8     74.7  54329      7003
## 5  1001020500 Alab… Autau…    10763  45.7     29.3     71.2  51965      6935
## 6  1001020600 Alab… Autau…     3851  46.4     25       68.6  63092      9585
## # … with 21 more variables: IncomePerCap <dbl>, 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>, CountyTotal <int>, CountyWeight <dbl>
```

```r
census.ct <- census.subct
CountyWeightSum <- summarise_at(census.ct, .funs = funs(sum), .vars =
                                   vars("CountyWeight"))
names(CountyWeightSum)[ncol(CountyWeightSum)] <- "CountyWeightSum"
census.ct <- left_join(census.ct,CountyWeightSum , by = c("State", "County"))
census.ct <- mutate(census.ct, CountyWeight = CountyWeight/CountyWeightSum)
census.ct <- select(census.ct, -CountyWeightSum, - CountyTotal)
census.ct[5:28] <- census.ct[5:28]*census.ct$CountyWeight
census.ct <- census.ct %>% summarise_at(vars(TotalPop:Unemployment), funs(sum))
census.ct <- ungroup(census.ct)
head(census.ct)
```

```
## # A tibble: 6 x 27
##   State County TotalPop   Men Minority Citizen Income IncomeErr IncomePerCap
##   <chr> <chr>     <dbl> <dbl>    <dbl>   <dbl>  <dbl>     <dbl>        <dbl>
## 1 Alab… Autau…    55221  48.4     22.5    73.7 51696.     7771.       24974.
## 2 Alab… Baldw…   195121  48.8     15.2    75.7 51074.     8745.       27317.
## 3 Alab… Barbo…    26932  53.8     51.9    76.9 32959.     6031.       16824.
## 4 Alab… Bibb      22604  53.4     24.2    77.4 38887.     5662.       18431.
## 5 Alab… Blount    57710  49.4     10.6    73.4 46238.     8696.       20532.
## 6 Alab… Bullo…    10678  53.0     76.5    75.5 33293.     9000.       17580.
```

```
## # … with 18 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>
```

13) From the first principle component, the three features that have the largest absolute values for principle component are: IncomePerCap, Income, and ChildPoverty (for county); and the same for sub-county. Minority, Poverty, ChildPoverty, Professional, Service, Drive, OtherTransp, WorkAtHome, MeanCommute, Employed, SelfEmployed, FamilyWork, and Unemployment all have opposite signs.

```
# 13
numericcensus.ct=select(ungroup(census.ct), -State, -County)
ct.pc=prcomp(scale(numericcensus.ct))
ct.pc2=ct.pc$rotation[,c(1,2)]
ct.pc2
```

```
##                          PC1          PC2
## TotalPop         0.082956993 -0.191690230
## Men              0.002298384  0.178560146
## Minority        -0.187712295 -0.074083164
## Citizen         -0.025016336  0.115116080
## Income           0.340954419 -0.162191758
## IncomeErr        0.197996494 -0.212660726
## IncomePerCap     0.367811444 -0.086233331
## IncomePerCapErr  0.216814369 -0.093489509
## Poverty         -0.336766254  0.023301745
## ChildPoverty    -0.341416433  0.008132606
## Professional     0.271781737  0.053960768
## Service         -0.175454443  0.039398090
## Office          -0.003999597 -0.286046143
## Production      -0.144894212 -0.092577866
## Drive           -0.111256279 -0.284989906
## Carpool         -0.078333721  0.060208311
## Transit          0.099178191 -0.116944274
## OtherTransp      0.003309337  0.092775111
## WorkAtHome       0.175133040  0.366615567
## MeanCommute     -0.053947922 -0.248831313
## Employed         0.332432299 -0.018674046
## PrivateWork      0.051782145 -0.403078434
## SelfEmployed     0.088996938  0.416458494
## FamilyWork       0.042840553  0.284131135
## Unemployment    -0.281027637 -0.094958643
```

```
numericcensus.subct=select(ungroup(census.subct), -County , -State)
subct.pc=prcomp(scale(numericcensus.subct))
subct.pc2=ct.pc$rotation[,c(1,2)]
subct.pc2
```

```
##                          PC1          PC2
## TotalPop         0.082956993 -0.191690230
## Men              0.002298384  0.178560146
## Minority        -0.187712295 -0.074083164
## Citizen         -0.025016336  0.115116080
## Income           0.340954419 -0.162191758
## IncomeErr        0.197996494 -0.212660726
```

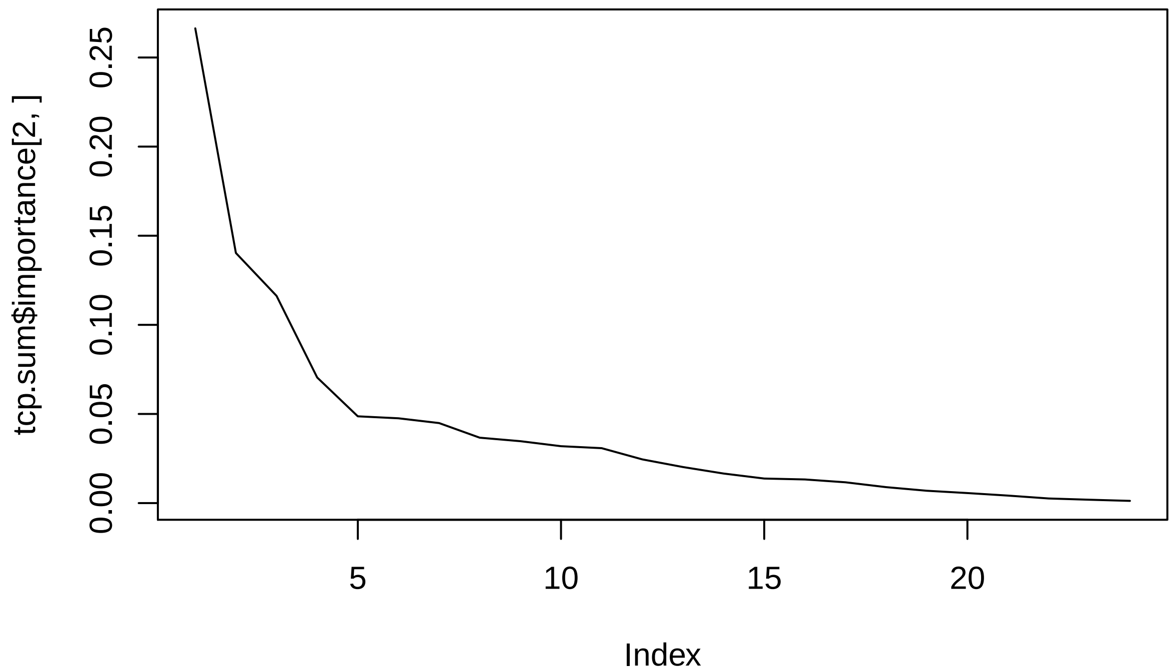
```
## IncomePerCap     0.367811444 -0.086233331
## IncomePerCapErr  0.216814369 -0.093489509
## Poverty         -0.336766254  0.023301745
## ChildPoverty    -0.341416433  0.008132606
## Professional     0.271781737  0.053960768
## Service         -0.175454443  0.039398090
## Office          -0.003999597 -0.286046143
## Production      -0.144894212 -0.092577866
## Drive           -0.111256279 -0.284989906
## Carpool         -0.078333721  0.060208311
## Transit          0.099178191 -0.116944274
## OtherTransp      0.003309337  0.092775111
## WorkAtHome       0.175133040  0.366615567
## MeanCommute     -0.053947922 -0.248831313
## Employed         0.332432299 -0.018674046
## PrivateWork      0.051782145 -0.403078434
## SelfEmployed     0.088996938  0.416458494
## FamilyWork       0.042840553  0.284131135
## Unemployment    -0.281027637 -0.094958643
```

14) I count 12 FALSE and 13 TRUE for numericccensus.ct and 15 FALSE and 11 TRUE for numericcensus.subct, and that should tell us how many principal components are needed to capture 90% of the variance.

```
# 14
tcp <- prcomp(numericcensus.ct[-1], center = T , scale. = T)
tcp.sum <- summary(tcp)
tcp.sum$importance[3,] >= .9
```

```
##   PC1   PC2   PC3   PC4   PC5   PC6   PC7   PC8   PC9  PC10  PC11  PC12  PC13
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
##  PC14  PC15  PC16  PC17  PC18  PC19  PC20  PC21  PC22  PC23  PC24
##  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

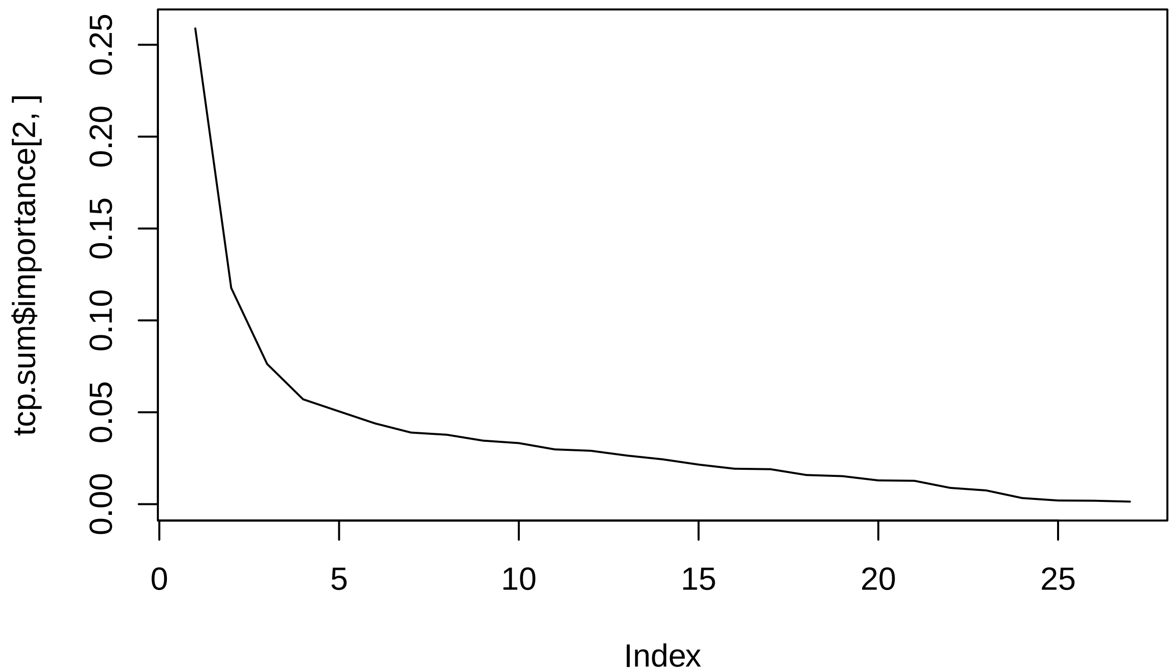
```
plot(tcp.sum$importance[2,], type="l")
```




```
tcp <- prcomp(numericcensus.subct[-1], center = T , scale. = T)
tcp.sum <- summary(tcp)
tcp.sum$importance[3,] >= .9
```

```
##   PC1   PC2   PC3   PC4   PC5   PC6   PC7   PC8   PC9  PC10  PC11  PC12  PC13
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  PC14  PC15  PC16  PC17  PC18  PC19  PC20  PC21  PC22  PC23  PC24  PC25  PC26
## FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  PC27
##  TRUE
```

```
plot(tcp.sum$importance[2,], type="l")
```



15) Here is the model for San Mateo

```r
# 15
cpa <- prcomp(census.ct[,c(-1,-2)], scale. = T, center = T)
dist.cpa <-  dist(cpa$x, method = "euclidean")
hc.c.all <- hclust(dist.cpa, method = "complete")
pca <- cutree(hc.c.all, k = 10)
cp5 <- prcomp(census.ct[,c(-1,-2)], scale. = T, center = T)
dist.cp5 <-  dist(cp5$x[,c(1:2)], method = "euclidean")
hc.c.5 <- hclust(dist.cp5, method = "complete")
partition.c.5 <- cutree(hc.c.5, k = 10)

psm.all <-  (cpa$x %*% cpa$rotation) %*% t(cpa$rotation)
psm.all <- as.data.frame(cbind(psm.all,pca))
acall <-
  as.matrix(rbind(colMeans(census.ct[which(psm.all$pca ==
  pca[which(census.ct[,2] == "San Mateo")]),][-c(1,2)]),
  colMeans(census.ct[-c(1,2)])))
Average <- c("San Mateo Cluster", " Total")
acall <- as.data.frame(cbind(Average , acall))
psm.5 <- (cp5$x %*% cp5$rotation) %*% t(cp5$rotation)
psm.5 <- as.data.frame(cbind(psm.5, partition.c.5))
ac5 <-
  as.matrix(rbind(colMeans(census.ct[which(psm.5$partition.c.5 ==
  partition.c.5[which(census.ct[,2] == "San Mateo")]),][-c(1,2)]),
  colMeans(census.ct[-c(1,2)])))
ac5 <- as.data.frame(cbind(Average , ac5))
```

```r
acall
```

```
##              Average         TotalPop              Men         Minority
## 1 San Mateo Cluster 644084.844444444 48.9612773922446 38.0631892110413
## 2             Total 99068.5966438782 49.9537311831876 22.6687728423035
##             Citizen           Income         IncomeErr      IncomePerCap
## 1 68.3341259824253 94010.9499792061 13783.9593490511 44077.2477186096
## 2  74.675072588377 47221.7002616651 7081.75813224634 24000.7488885814
##     IncomePerCapErr          Poverty      ChildPoverty     Professional
## 1 5600.71539384471 9.41529570040947 11.4339930459409 48.8651848382481
## 2 3120.46035313414 17.5706570229159 23.8336932248276 30.8004060227024
##             Service           Office        Production            Drive
## 1 16.0464520538116   22.09027495994  6.8603955618439 67.7241216419104
## 2  18.472858282297 22.1830928074174 15.8060173865796 79.1462674727629
##             Carpool            Transit       OtherTransp        WorkAtHome
## 1 8.55244245336479  12.8722225189201 1.87297397721839 5.44253614186442
## 2 10.3194871134777 0.989733401044941 1.62516443074591 4.60507443307083
##         MeanCommute         Employed       PrivateWork      SelfEmployed
## 1 31.4541257473991 51.0486964782138 77.1783898547935 5.81303108269043
## 2 23.3134361924172 43.0247144453537  74.183032238375  7.9164536646503
##           FamilyWork      Unemployment
## 1 0.128536340131728 6.89145909506166
## 2 0.287330219645564 8.15706443334943
```

```r
ac5
```

```
##              Average         TotalPop              Men         Minority
## 1 San Mateo Cluster 625849.078947368 49.1084530855665 30.8991961185339
## 2             Total 99068.5966438782 49.9537311831876 22.6687728423035
##             Citizen           Income         IncomeErr      IncomePerCap
## 1 68.7299570850578 98745.3032057856 14436.0581973632 47613.8830317222
## 2  74.675072588377 47221.7002616651 7081.75813224634 24000.7488885814
##     IncomePerCapErr          Poverty      ChildPoverty     Professional
## 1 6331.33075172364 7.54169673679544 8.71351745420431 51.4006161484117
## 2 3120.46035313414 17.5706570229159 23.8336932248276 30.8004060227024
##             Service           Office        Production            Drive
## 1 14.6141823950696 22.2957872852996 6.00811108009643 69.4862453375363
## 2  18.472858282297 22.1830928074174 15.8060173865796 79.1462674727629
##             Carpool            Transit       OtherTransp        WorkAtHome
## 1 7.65060024649833  11.0895485917158 1.93415349024991 6.38112481935979
## 2 10.3194871134777 0.989733401044941 1.62516443074591 4.60507443307083
##         MeanCommute         Employed       PrivateWork      SelfEmployed
## 1 29.7472034804768 52.5944666486074 78.8336795248032 6.27143036435642
## 2 23.3134361924172 43.0247144453537  74.183032238375  7.9164536646503
##           FamilyWork      Unemployment
## 1  0.13124711210741 5.93870316509058
## 2 0.287330219645564 8.15706443334943
```

```r
tmpwinner <- county_winner %>% ungroup %>%
  mutate(state = state.name[match(state, state.abb)]) %>%
  mutate_at(vars(state, county), tolower) %>%
  mutate(county = gsub(" county| columbia| city| parish", "", county))

tmpcensus <- census.ct %>% mutate_at(vars(State, County), tolower)
election.cl <- tmpwinner %>%
```

```r
  left_join(tmpcensus, by = c("state"="State", "county"="County")) %>%
  na.omit
election.meta <- election.cl %>% select(c(county, fips, state, votes, pct, total))
election.cl = election.cl %>% select(-c(county, fips, state, votes, pct, total))
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,]
set.seed(20)
nfold <- 10
folds <- sample(cut(1:nrow(trn.cl), breaks=nfold, labels=FALSE))
calc_error_rate = function(predicted.value, true.value){
  return(mean(true.value!=predicted.value))
}
# this adjustment is for later... just bear with me (it's number 20)
records = matrix(NA, nrow=5, ncol=2)
colnames(records) = c("train.error","test.error")
rownames(records) = c("tree","logistic","lasso","knn","lda")
```

16) From what I can gather from this tree, Transit, Minority, TotalPop, Professional, and Income are the best predictors for this decision tree. As the story goes, while the professionals

```r
# 16
library(tree)
```

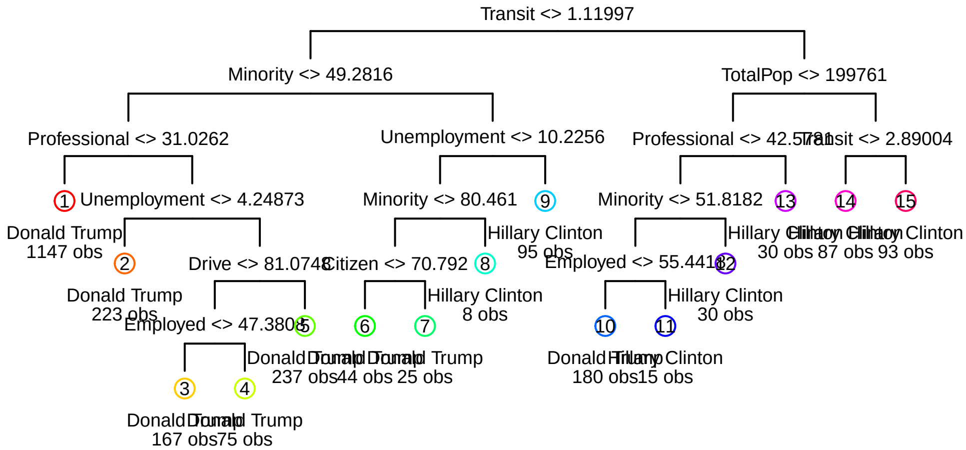
```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

```r
library(maptree)
```

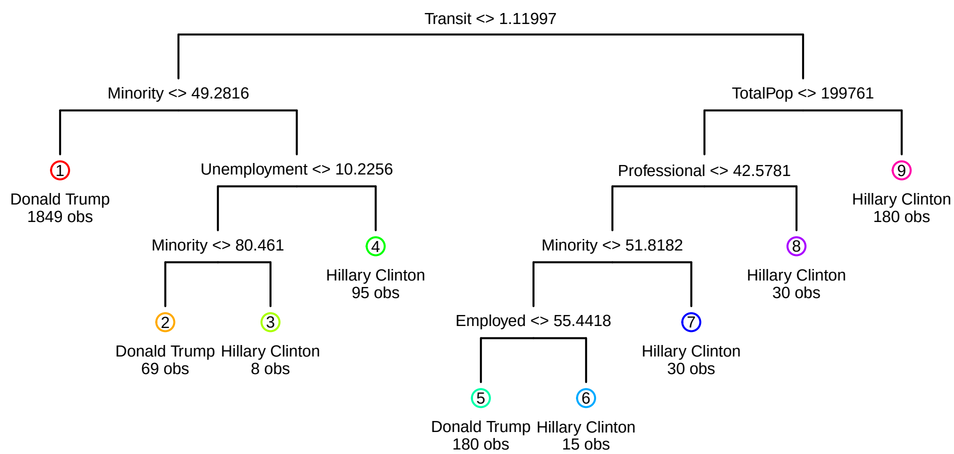
```
## Loading required package: cluster
```

```
## Loading required package: rpart
```

```r
candidate.tree <- tree(candidate ~ ., data = trn.cl)
cv <- cv.tree(candidate.tree, rand = folds, FUN = prune.misclass, K = nfold)
min.dev <- min(cv$dev)
best.size.cv <- cv$size[which(cv$dev == min.dev)]
draw.tree(candidate.tree, cex = 0.55)
```




```
tree.pruned <- prune.misclass(candidate.tree, best = best.size.cv)
draw.tree(tree.pruned, cex = 0.5)
```




```
tree.train <- predict(tree.pruned, trn.cl, type = "class")
tree.test <- predict(tree.pruned, tst.cl, type = "class")
records[1,1] <- calc_error_rate(tree.train, trn.cl$candidate)
records[1,2] <- calc_error_rate(tree.test, tst.cl$candidate)
records
```

```
##          train.error test.error
## tree      0.06107492 0.07166124
## logistic          NA         NA
## lasso             NA         NA
## knn               NA         NA
## lda               NA         NA
```

17) See below for the significant variables. No, this isn't entirely consistent with what we got out of the decision tree model. One of the variables that is rated differently is TotalPop, and I think it has to do with the fact that a decision tree might not be the best model in extrapolating insights for a variable concerning the entire populace and not just a segment of it.

```
# 17
logmodel <- glm(candidate ~ ., data = trn.cl, family = "binomial")
```

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

```
summary(logmodel)
```

```
##
## Call:
## glm(formula = candidate ~ ., family = "binomial", data = trn.cl)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.2217  -0.2540  -0.1072  -0.0373   3.5503
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -4.369e+01  7.271e+00  -6.009 1.86e-09 ***
## TotalPop         2.723e-07  4.170e-07   0.653 0.513760
## Men              6.488e-02  4.736e-02   1.370 0.170700
## Minority         1.363e-01  9.687e-03  14.066  < 2e-16 ***
## Citizen          1.356e-01  2.787e-02   4.866 1.14e-06 ***
## Income          -6.370e-05  2.598e-05  -2.452 0.014195 *
## IncomeErr       -1.231e-05  6.161e-05  -0.200 0.841683
## IncomePerCap     2.172e-04  6.286e-05   3.455 0.000551 ***
## IncomePerCapErr -2.817e-04  1.319e-04  -2.137 0.032612 *
## Poverty          4.061e-02  4.074e-02   0.997 0.318861
## ChildPoverty    -9.712e-03  2.461e-02  -0.395 0.693093
## Professional     2.887e-01  3.855e-02   7.490 6.87e-14 ***
## Service          3.311e-01  4.775e-02   6.935 4.07e-12 ***
## Office           8.651e-02  4.565e-02   1.895 0.058118 .
## Production       1.519e-01  4.109e-02   3.697 0.000218 ***
## Drive           -1.848e-01  4.653e-02  -3.971 7.14e-05 ***
## Carpool         -1.373e-01  5.811e-02  -2.363 0.018146 *
## Transit          1.330e-01  9.323e-02   1.426 0.153825
## OtherTransp     -4.851e-02  9.462e-02  -0.513 0.608154
## WorkAtHome      -1.705e-01  7.440e-02  -2.292 0.021885 *
## MeanCommute      3.966e-02  2.381e-02   1.665 0.095830 .
## Employed         1.952e-01  3.307e-02   5.903 3.57e-09 ***
## PrivateWork      1.133e-01  2.133e-02   5.313 1.08e-07 ***
## SelfEmployed     5.923e-02  4.635e-02   1.278 0.201314
## FamilyWork      -8.196e-01  3.877e-01  -2.114 0.034526 *
## Unemployment     2.187e-01  4.005e-02   5.461 4.73e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2122.94  on 2455  degrees of freedom
## Residual deviance:  847.56  on 2430  degrees of freedom
## AIC: 899.56
##
## Number of Fisher Scoring iterations: 7
```

```r
logpred <- predict(logmodel, trn.cl, type = "response")
trn.cl <- trn.cl %>% mutate(candidate = as.factor(ifelse(candidate == "Donald Trump",
                          "Donald Trump", "Hillary Clinton")))
library(ROCR)
```

```
## Loading required package: gplots

##
## Attaching package: 'gplots'

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

```r
logprediction <- prediction(logpred, trn.cl$candidate)
fpr.train = performance(logprediction, "fpr")@y.values[[1]]
cutoff.train <- performance(logprediction, "fpr")@x.values[[1]]
fnr.train <- performance(logprediction, "fnr")@y.values[[1]]
train.rate <- as.data.frame(cbind(Cutoff = cutoff.train, FPR = fpr.train, FNR =
                                    fnr.train))
train.rate$distance <- sqrt((train.rate[,2]^2) + (train.rate[,3])^2)
index = which.min(train.rate$distance)
best = train.rate$Cutoff[index]
trn.cl.pred <- trn.cl %>% mutate(predCandidate =
as.factor(ifelse(logpred <= best, "Donald Trump", "Hillary Clinton")))
trainerror <- calc_error_rate(trn.cl.pred$candidate,
                                        trn.cl.pred$predCandidate)
logistic.test.predict <- predict(logmodel, tst.cl, type = "response")
tst.cl <- tst.cl %>% mutate(candidate = as.factor(ifelse(candidate == "Donald Trump",
                            "Donald Trump", "Hillary Clinton")))
tst.cl.pred <- tst.cl %>% mutate(predCandidate =
as.factor(ifelse(logistic.test.predict <= best, "Donald Trump", "Hillary Clinton")))
testerror <- calc_error_rate(tst.cl.pred$candidate,
                                       tst.cl.pred$predCandidate)
records[2,1] = trainerror
records[2,2] = testerror
records
```

```
##          train.error test.error
## tree      0.06107492 0.07166124
## logistic  0.10504886 0.10749186
## lasso             NA         NA
## knn               NA         NA
## lda               NA         NA
```

18) The optimal value of lambda is .001, and its non-zero coefficients are as listed below in the output.

```r
# 18
library(glmnet)
```

```
## Loading required package: Matrix
## Loaded glmnet 3.0-2
```

```r
trn.cl = na.omit(trn.cl)
x=model.matrix(candidate~., election.cl)[,-1]
y1=trn.cl$candidate
y2=tst.cl$candidate
ychar=as.character(election.cl$candidate)
grid=c(1,5,10,50) * 1e-4
cvlasso = cv.glmnet(x[in.trn,], ychar[in.trn], lambda=grid,
                    alpha=1, family='binomial', foldid=folds)
bestlambda = cvlasso$lambda.min
bestlambda
```

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

```r
model = glmnet(x[in.trn,], ychar[in.trn], alpha=1, family='binomial')
lassocoef = predict(model, type='coefficients', s=bestlambda)
lassocoef
```

```
## 26 x 1 sparse Matrix of class "dgCMatrix"
##                             1
## (Intercept)     -4.134967e+01
## TotalPop         3.181848e-07
## Men              4.218882e-02
## Minority         1.289709e-01
## Citizen          1.407467e-01
## Income          -4.301211e-05
## IncomeErr       -1.229650e-05
## IncomePerCap     1.649208e-04
## IncomePerCapErr -2.042494e-04
## Poverty          3.077325e-02
## ChildPoverty     .
## Professional     2.582219e-01
## Service          2.985173e-01
## Office           5.644868e-02
## Production       1.202155e-01
## Drive           -1.556694e-01
## Carpool         -1.115773e-01
## Transit          1.423687e-01
## OtherTransp     -4.859619e-03
## WorkAtHome      -1.229121e-01
## MeanCommute      2.551169e-02
## Employed         1.860462e-01
## PrivateWork      1.061805e-01
## SelfEmployed     3.466455e-02
## FamilyWork      -6.924729e-01
## Unemployment     2.057937e-01
```

```r
lassotrain = predict(model, s=bestlambda, newx=x[in.trn,], type='class')
lassotest = predict(model, s=bestlambda, newx=x[-in.trn,], type='class')
records[3,1] = calc_error_rate(lassotrain, y1)
records[3,2] = calc_error_rate(lassotest, y2)
records
```

```
##          train.error test.error
## tree      0.06107492 0.07166124
## logistic  0.10504886 0.10749186
## lasso     0.06718241 0.06840391
## knn               NA         NA
## lda               NA         NA
```

19) Here, I wanted to test errors for Kth nearnest neighbor and linear discrimination analysis. As you can see, LDA outperforms both KNN and logistic regression, and thus should be considered a top candidate for model selection. While it does have the worst test error, it does train better than every other model, which can actually be really useful depending on what you are looking for. Thus, this concludes my 2016 elections analysis, and I would like to thank the professor for providing the requesite materials necessary to complete this project.

```r
# 19
library(class)
k.test = c(1, seq(10, 50, length.out = 9))
do.chunk <- function(chunkid, folddef, Xdat, Ydat, k){
  train = (folddef!=chunkid)
  Xtr = Xdat[train,]
  Ytr = Ydat[train]
```

```r
  Xvl = Xdat[!train,]
  Yvl = Ydat[!train]
  predYtr = knn(train = Xtr, test = Xtr, cl = Ytr, k = k)
  predYvl = knn(train = Xtr, test = Xvl, cl = Ytr, k = k)
  data.frame(train.error = calc_error_rate(predYtr, Ytr),
  val.error = calc_error_rate(predYvl, Yvl))
}
K_Errors <- tibble("K" = k.test, "AveTrnError" = NA, "AveTstError" = NA)
predictors <- select(trn.cl, -candidate)
for(i in 1:10){
  temp <- plyr::ldply(1:10, do.chunk, folds,predictors, trn.cl$candidate,
                      K_Errors$K[i])
  K_Errors$AveTrnError[i] <- mean(temp[,1])
  K_Errors$AveTstError[i] <- mean(temp[,2])
}
pred.Train = knn(train=tst.cl[,2:26], test=tst.cl[,2:26],
                 cl=tst.cl$candidate, k=10)
erate.train <- calc_error_rate(pred.Train, trn.cl$candidate)
pred.Test = knn(train=trn.cl[,2:26], test=trn.cl[,2:26],
                cl=trn.cl$candidate, k=10)
erate.test <- calc_error_rate(pred.Test, tst.cl$candidate)
records[4,] <- c(erate.train, erate.test)
tcl <- MASS::lda(candidate ~ . , data = trn.cl)
trainlda <- predict(tcl, trn.cl)$class
testlda <- predict(tcl, tst.cl)$class
records[5,1] <- calc_error_rate(trainlda, trn.cl$candidate)
records[5,2] <- calc_error_rate(testlda, tst.cl$candidate)
records
```

```
##            train.error test.error
## tree        0.06107492 0.07166124
## logistic    0.10504886 0.10749186
## lasso       0.06718241 0.06840391
## knn         0.20114007 0.20399023
## lda         0.06718241 0.07003257
```