PSTAT 131 Final Project

Marissa Santiago: 6220214 and Leticia Gonzalez: 9823535

June 11, 2021

```
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
```

```
library(ggplot2)
library(readr)
library(tidyr)
library(ISLR)
library(knitr)
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──
```

```
## ✓ tibble  3.1.0     ✓ stringr 1.4.0
## ✓ purrr   0.3.4     ✓ forcats 0.5.1
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(maps)
```

```
##
## Attaching package: 'maps'
```

```
## The following object is masked from 'package:purrr':
##
##     map
```

```
library(tree)
```

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

```
library(maptree)
```

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

```
##
## Attaching package: 'cluster'
```

```
## The following object is masked from 'package:maps':
##
##     votes.repub
```

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

```
library(class)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
library(ROCR)
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
```

```
library(dendextend)
```

```
##
## ---------------------
## Welcome to dendextend version 1.15.1
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendex
tend/issues
## Or contact: <tal.galili@gmail.com>
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------
```

```
##
## Attaching package: 'dendextend'
```

```
## The following object is masked from 'package:rpart':
##
##     prune
```

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

```
library(kableExtra)
```

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

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

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

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

```
library(Rtsne)
library(NbClust)
library(gbm)
```

```
## Loaded gbm 2.1.8
```

Background

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

There can be a couple reasons why it is hard to predict voter behavior for elections. Voter prediction is difficult because data is conducted primarily through polls. This can be a very problematic source of data as strong supportors may not answer polls or surveys nor does it follow the assumption that the polls matchup exactly with the voter turnout. This demonstrates a bias in the data collection process (sample being truly representative of voting population on election day). This bias can be a no response bias or it can be a result of people responding to polls but not actually vote on election day. Second, people may provide false data to polls if they feel like their decision may be judged. Third, since polls are taken months in advance, voter opinion can change from the time of the poll to election day. Lastly, decisions on variables used for analysis is crucial for forecasting and presents another difficulty for forecasters.

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

Nate Silver's approach in 2012 was unique as it includes the unsupervised learning method of hierarchical modelling which allows for information to be moved around the movel. He considered multiple stages in the process of sampling and acknowledges the dynamics in behavior over time. He considers creating a time series graph to allow him to capture the percentage of variation in voting intention and the extent of its effect as the levels of uncertainty often rises closer to election date. Nate Silver also considers the distribution and statistics of the model he obtained to generate different results per state. He arrived at the formula:actual percentage + house effects + sampling variation. Instead of only looking at the maximum probability, Silver's approach utilizes

Bayes' theorem and takes in a full range of probabilities when creating the model that predicts change in support for candidates. Finally, he references previous election polling results and actual results to estimate possible bias and extent of deviation of support.

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

The 2016 polls had a high bias towards Hillary Clinton winning the elections. It is speculated that polls did not account for many of the Trump voters in their polls (either through people giving false information or being a shy Trump supporter) as well as the overwhelmingly large number of people who would vote on election day for Trump. The collection of data was conducted using phone polls in which voters would receive calls from a recorded voice instead of a live person which may be a source leading to changes in behavior and biases. Some Trump voters were distrustful of institutions and poll calls which led to the inaccuracy of polling results. The polling results also often does not capture the late supporters and those who were supporting Gary Johnson.

To make future predictions more accurate, voter demographic information should be taken into account at a federal, state, and county level and supervised learning models should be applied to better predict which factors are most influential in voter choice, and categorize voters into candidate groups. Polling organizations may also consider to use a variety of polling methods to encourage participation which includes email, text, web surveys, and not only using phone calls. Future predicitons should also consider the uncertainty to a greater extent.

Data

```
## set the working directory as the file location
setwd("/Users/marissa/Desktop")
## 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(candid
ate=as.factor(candidate))
```

```
##
## ── Column specification ────────────────────────────────────────────────────────
## cols(
##   county = col_character(),
##   fips = col_character(),
##   candidate = col_character(),
##   state = col_character(),
##   votes = col_double()
## )
```

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

```
##
## ── Column specification ────────────────────────────────────────────────────────
## cols(
##   X1 = col_character(),
##   X2 = col_character(),
##   X3 = col_character()
## )
```

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

```
##
## ── Column specification ────────────────────────────────────────────────────────
## cols(
##   .default = col_double(),
##   State = col_character(),
##   County = col_character()
## )
## ℹ Use `spec()` for the full column specifications.
```

Election Data

In our dataset, fips values denote the area (US, state, or county) that each row of data represent.

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.

```
kable(election.raw %>% filter(county == "Los Angeles County"))  %>% kable_styling(boots
trap_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. 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.

```
dim(election.raw)
```

```
## [1] 18351     5
```

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

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

```
dim(na.omit(election.raw))
```

```
## [1] 18011     5
```

There are now 18345 observations for 5 variables. 6 observations were removed. This is because they were outliers, displaying a very low fips value which could skew our data.

Census Data

```
kable(census %>% head, "html")  %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
  full_width=FALSE) %>%
  scroll_box(width = "100%")
```

State	County	TotalPop	Men	Women	Hispanic	White	Black	Native	Asian	Pacific	Citizen	Inc
Alabama	Autauga	1948	940	1008	0.9	87.4	7.7	0.3	0.6	0.0	1503	6
Alabama	Autauga	2156	1059	1097	0.8	40.4	53.3	0.0	2.3	0.0	1662	3
Alabama	Autauga	2968	1364	1604	0.0	74.5	18.6	0.5	1.4	0.3	2335	4
Alabama	Autauga	4423	2172	2251	10.5	82.8	3.7	1.6	0.0	0.0	3306	5
Alabama	Autauga	10763	4922	5841	0.7	68.5	24.8	0.0	3.8	0.0	7666	5
Alabama	Autauga	3851	1787	2064	13.1	72.9	11.9	0.0	0.0	0.0	2642	6

Data Wrangling

5.

```
# county can be seperate by its unique aspect of having the county defined
election <- filter(election.raw, !is.na(county))

#seperate election federal for its unique factor of having fips=US
election_federal <- filter(election.raw, fips == "US")

# state does not have the above unique attributes
election_state <- filter(election.raw, fips != "US" & is.na(county) & fips != "DC" &
  as.character(election.raw$fips) == as.character(election.raw$state))

dim(election_federal)
```

```
## [1] 32  5
```

```
dim(election_state)
```

```
## [1] 298   5
```

```
dim(election)
```

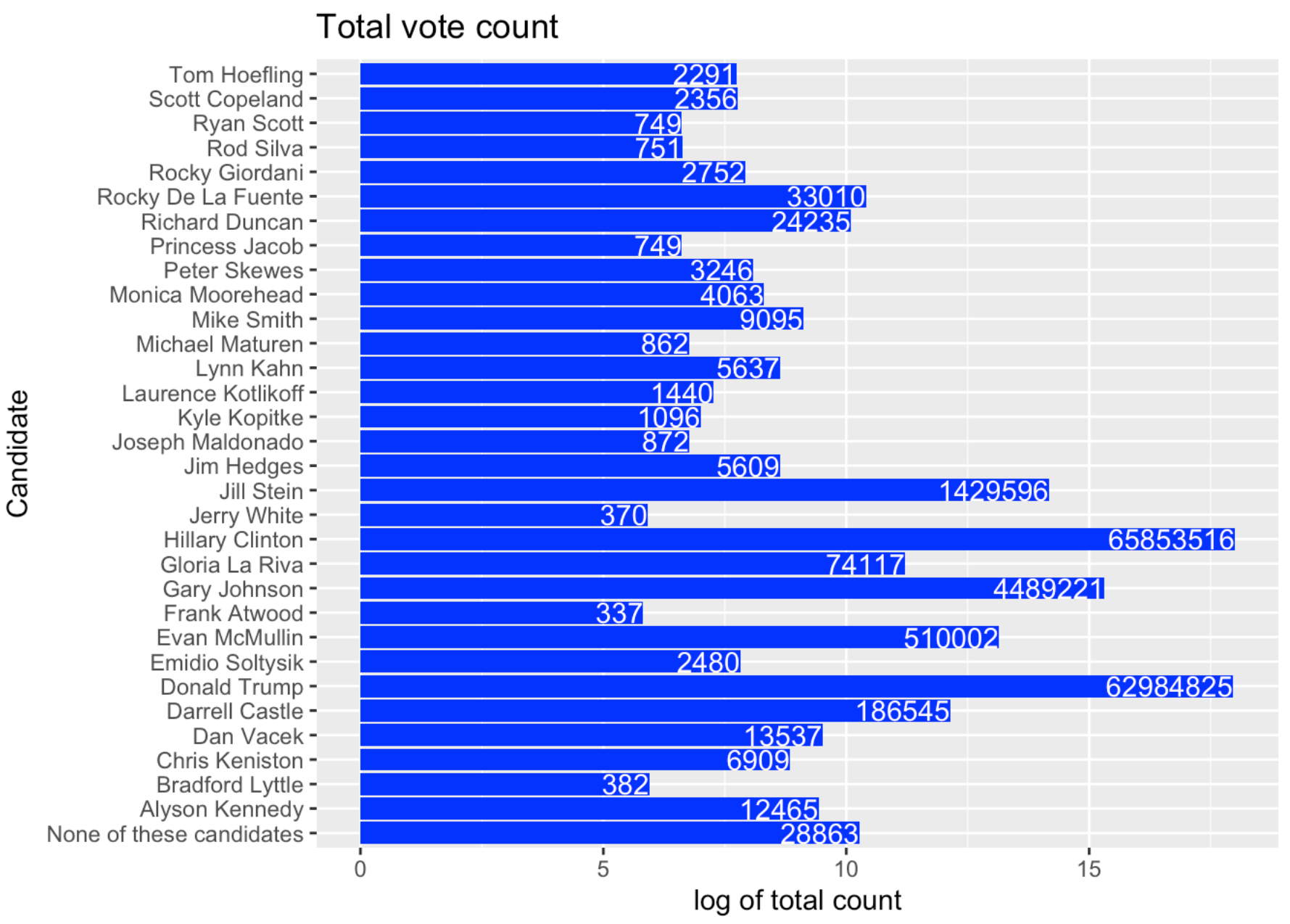
```
## [1] 18011     5
```

6.

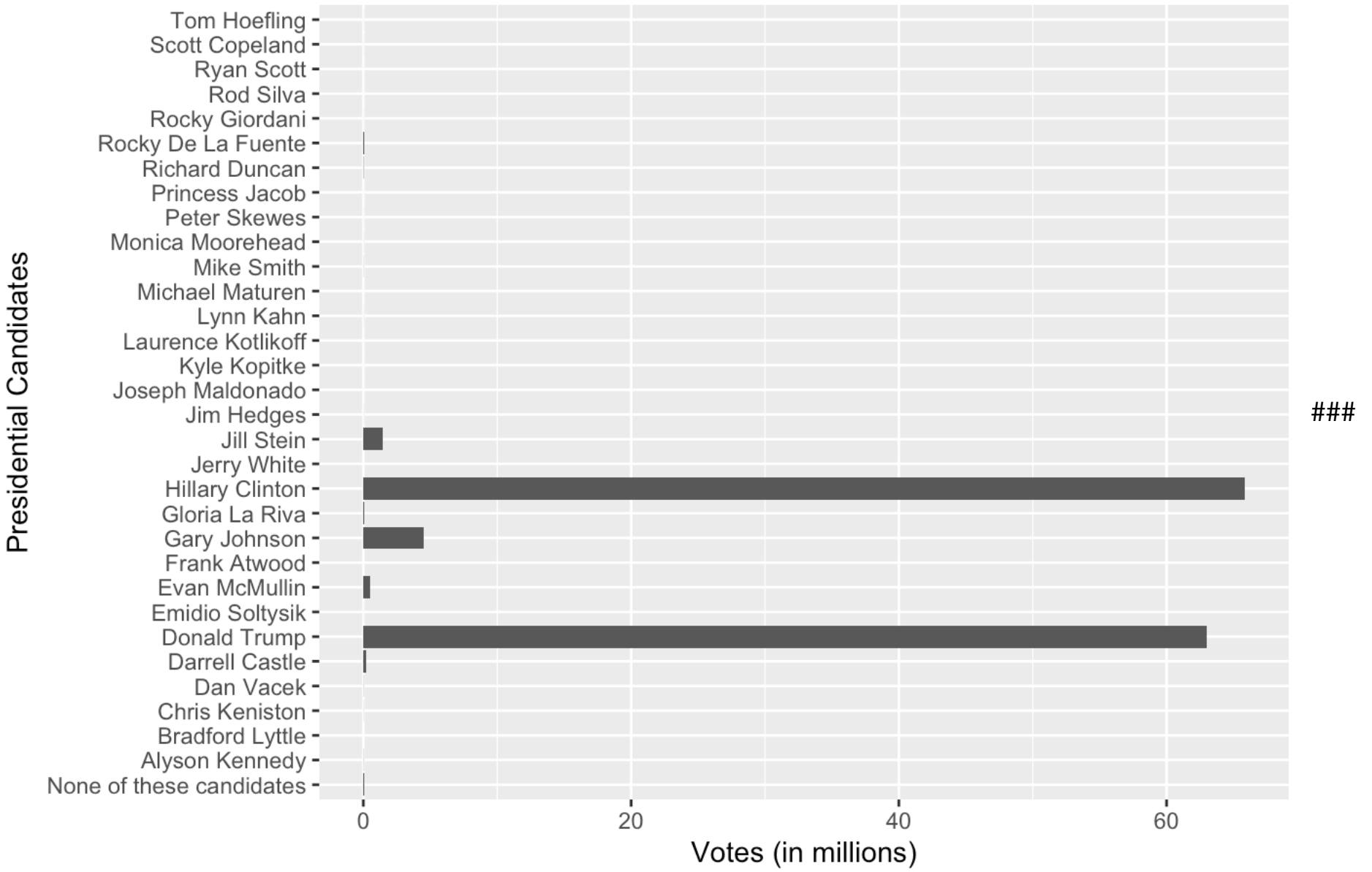
```
cat("There were",length(unique(election_federal$candidate)), "presidential candidates i
n the 2016 elections.")
```

```
## There were 32 presidential candidates in the 2016 elections.
```

```
#log bar plot
ggplot(data = election_federal,aes(x=candidate,y=log(votes)))+geom_bar(stat='identity',
fill='blue')+geom_text(aes(label=votes,hjust=1),color='white')+coord_flip()+ggtitle('To
tal vote count')+labs(x='Candidate',y='log of total count')
```




```
#bar plot
ggplot(data = election_federal, aes(x = candidate,y = votes/1000000)) +
  ggtitle("Votes recieved by each Candidate in the 2016 Presidential Elections") +
  xlab("Presidential Candidates") +
  ylab("Votes (in millions)") +
  geom_bar(stat="identity") +
  coord_flip()
```



7. Create variables county_winner and state_winner by taking the candidate with the highest proportion of votes.

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

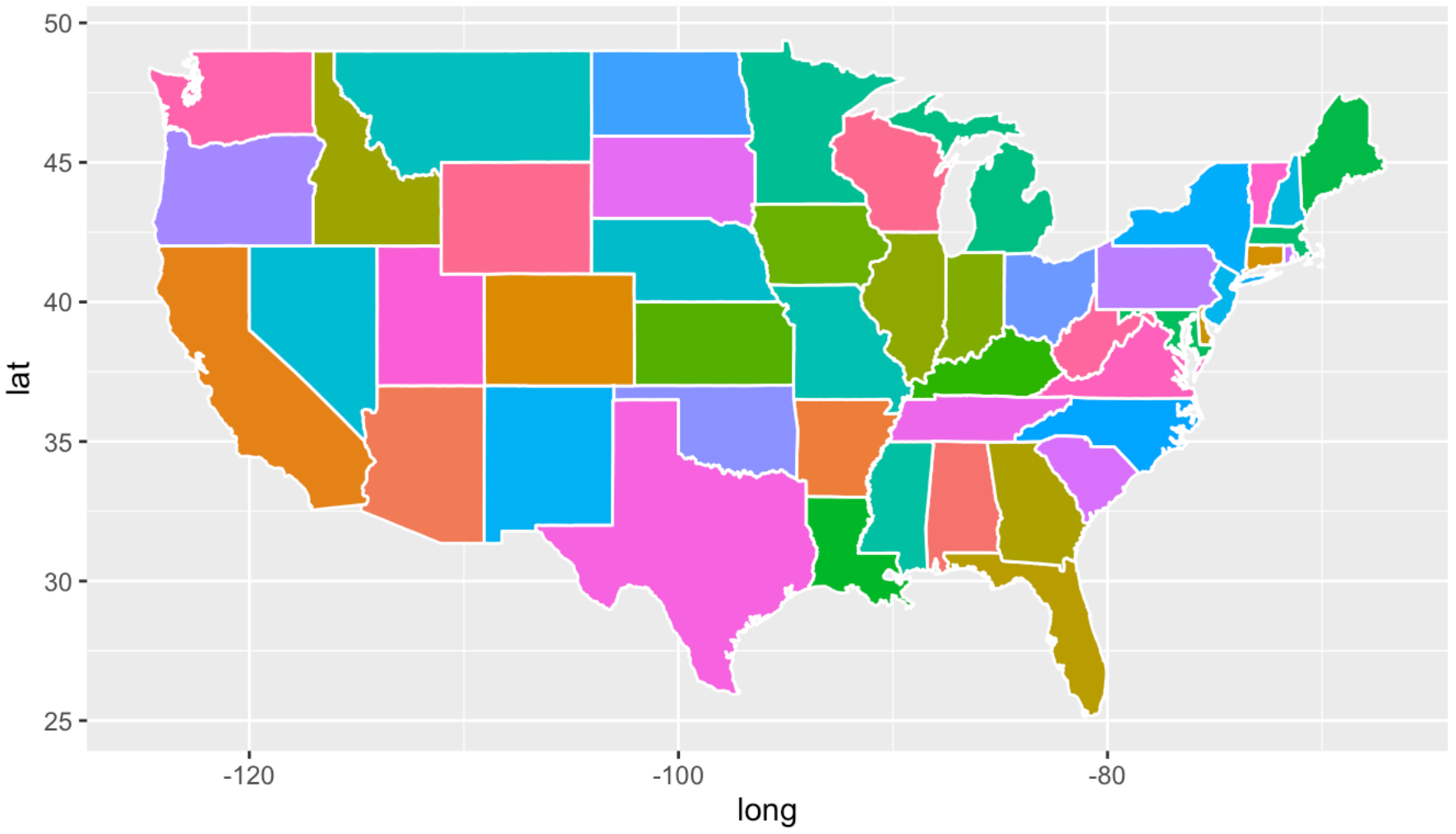
```
## Selecting by pct
```

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

```
## Selecting by pct
```

Visualization

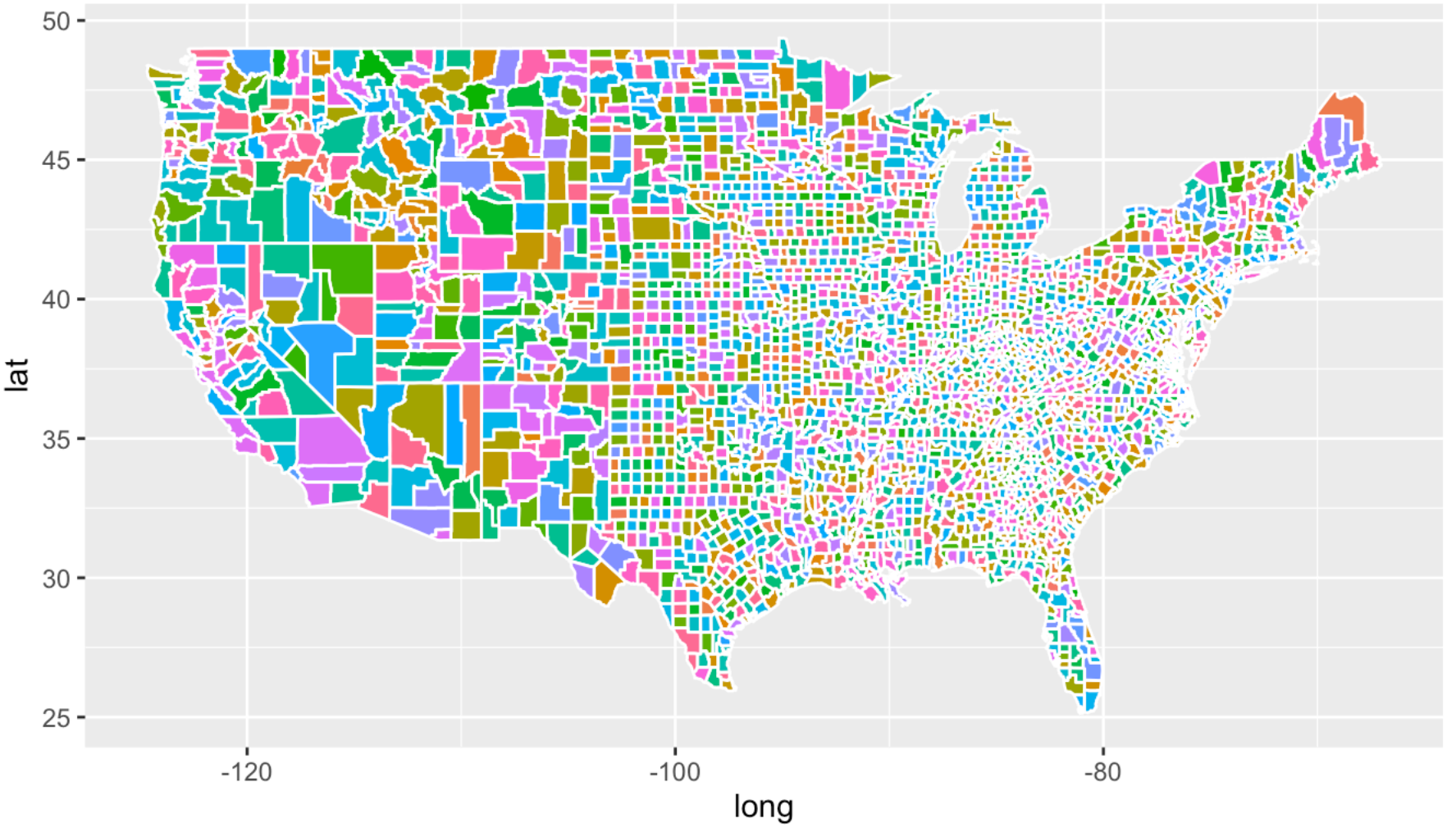
```
states <- map_data("state")

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. Draw a county-level map, Color by county.

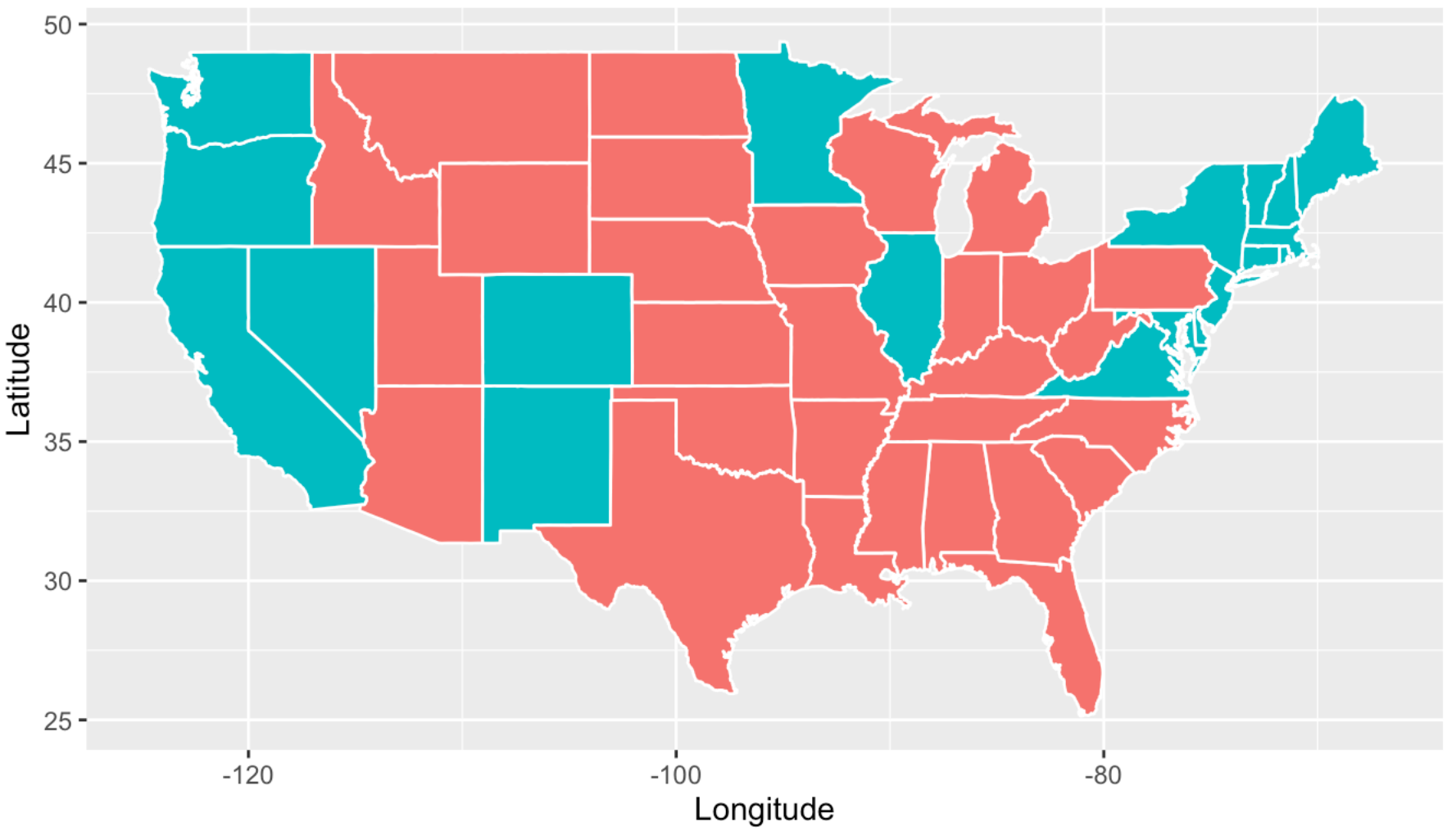
```
county <- map_data("county")
ggplot(data = county) +
  geom_polygon(aes(x = long, y = lat, fill = subregion, group = group), color = "white"
) +
  coord_fixed(1.3) +
  guides(fill= FALSE)
```



###

9. Now color the map by the winning candidate for each state.

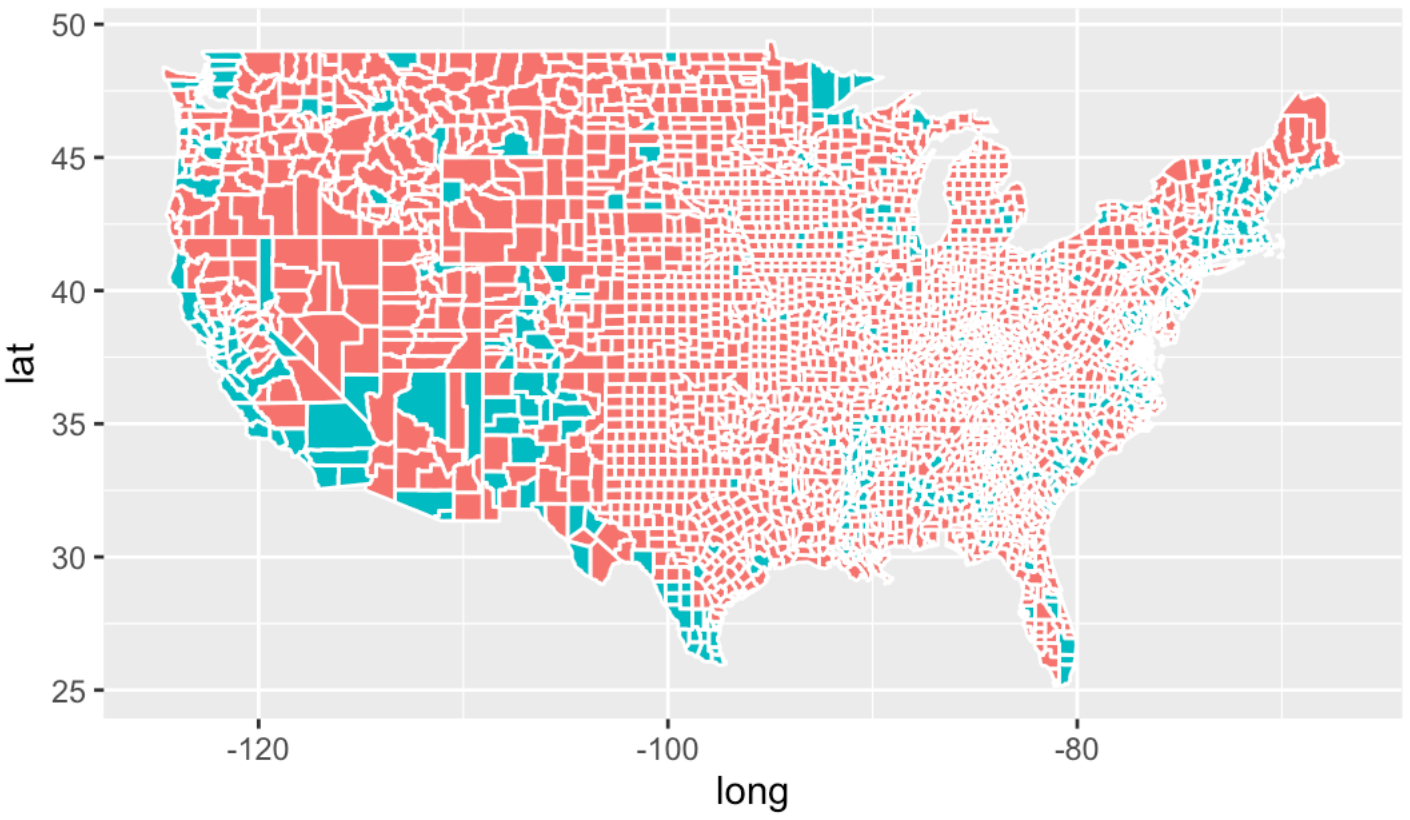
```
states <- map_data("state")

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

winner.states <- left_join(states, state_winner, by = c("fips" = "state"))

ggplot(data = winner.states) + geom_polygon(aes(x = long, y = lat, fill = candidate, gr
oup = group), color = "white") + coord_fixed(1.3) + xlab("Longitude") + ylab("Latitude"
) + guides(fill= FALSE)+ ggtitle("Winning candidate for each state")
```



###

10.

```
county.fips <- maps::county.fips %>% separate(polyname, c("region","subregion"), sep=",
")

counties <- left_join(county.fips,county,by=c("region", "subregion"))
county_winner$fips <- as.integer(county_winner$fips)

counties <- left_join(counties,county_winner,by=c('fips'))

ggplot(data=counties)+geom_polygon(aes(x=long,y=lat,fill=candidate,group=group),color='
white')+coord_fixed(1.3)
```



###

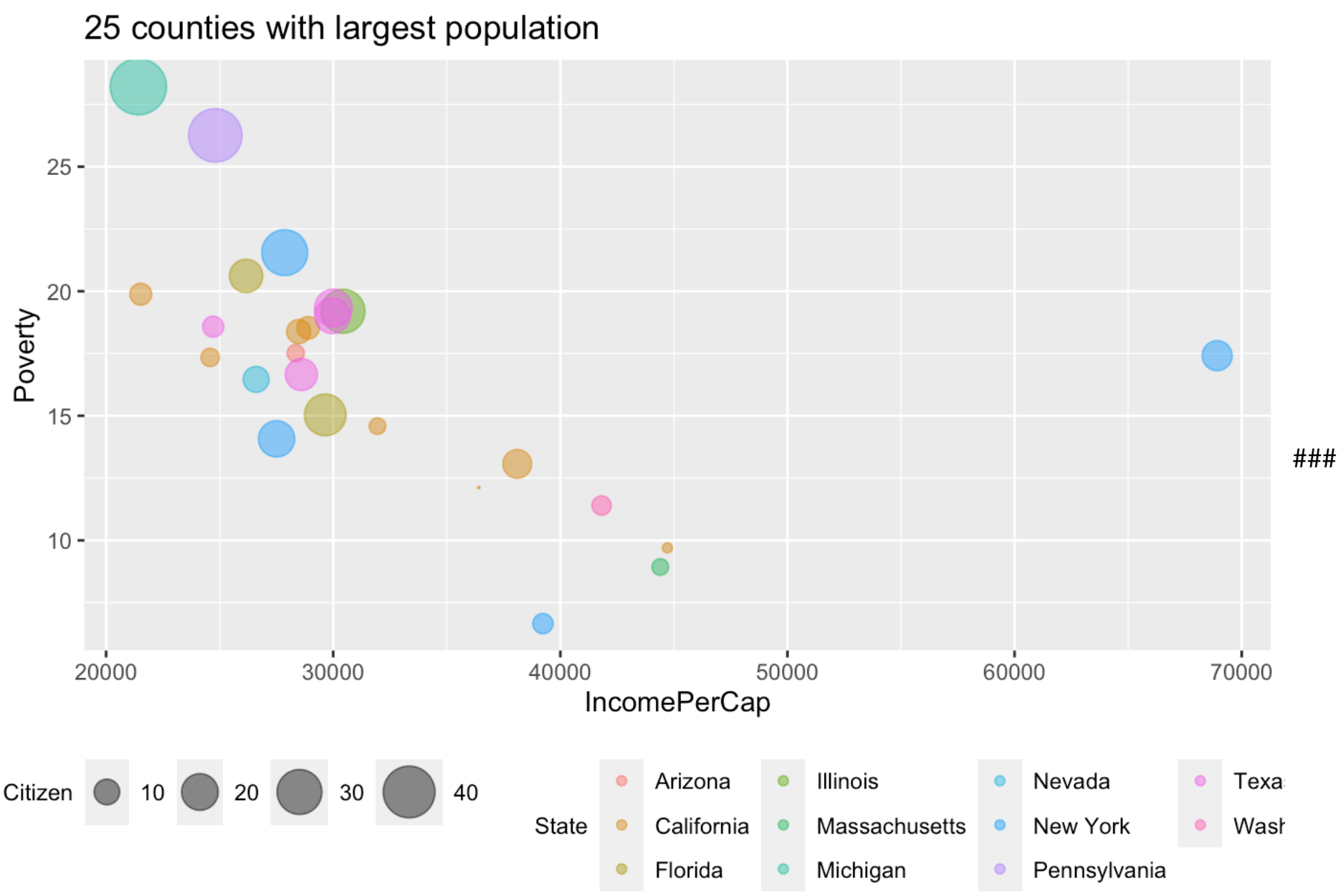
11. Create a visualization of your choice using census data.

```
census.visual <- na.omit(census)%>%group_by(State,County)%>%mutate(TotalPop=sum(TotalPo
p))%>%summarise_each(funs(mean),TotalPop:PrivateWork)
```

```
## Warning: `summarise_each_()` was deprecated in dplyr 0.7.0.
## Please use `across()` instead.
```

```
## Warning: `funs()` was deprecated in 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))
```

```
top25 <- census.visual[order(-census.visual$TotalPop),][1:25,]

ggplot(data=top25,aes(x=IncomePerCap,y=Poverty,size=Black,color=State))+geom_point(alph
a=0.5)+scale_size(range=c(.1,10),name='Citizen')+theme(legend.position = 'bottom',legen
d.title=element_text(size=9))+ggtitle('25 counties with largest population')
```



###

12.

```
census.del <- na.omit(census)  %>%
  mutate(Men=Men/TotalPop*100,
         Employed=Employed/TotalPop*100,
         Citizen=Citizen/TotalPop*100,
         Minority=(Hispanic+Black+Native+Asian+Pacific)) %>%
  select(-c(Women,Hispanic, Black, Native, Asian, Pacific, Walk, PublicWork, Constructi
on))
dim(census.del)
```

```
## [1] 72727    28
```

```
census.subct <- census.del %>%
  group_by(State,County) %>%
  add_tally(TotalPop, name="CountyTotal") %>%
  mutate( Weight=TotalPop/CountyTotal)
dim(census.subct)
```

```
## [1] 72727    30
```

```
#county
census.ct <- census.subct %>%
  group_by(State,County) %>%
  summarise_at(vars(Men:CountyTotal), funs(weighted.mean))
dim(census.ct)
```

```
## [1] 3218   28
```

```
kable(head(census.ct[,1:10]),caption ="Some observations from Election Data Frame") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"), f
ull_width=FALSE)
```

Some observations from Election Data Frame

State	County	Men	White	Citizen	Income	IncomeErr	IncomePerCap	IncomePerCapErr	Poverty
Alabama	Autauga	48.37	73.15	74.98	49985	8036	24387	3704	14.07
Alabama	Baldwin	48.83	83.45	76.37	48673	9268	26843	4008	14.36
Alabama	Barbour	52.22	46.61	76.31	32368	5891	17105	2523	26.79
Alabama	Bibb	53.25	77.50	76.83	40212	6143	18807	3340	15.60
Alabama	Blount	49.53	87.80	73.54	45101	8920	20171	2053	17.30
Alabama	Bullock	51.78	22.00	76.05	33445	8511	17735	3368	25.57

Dimensionality Reduction

13. Run PCA for both county & sub-county level data.

```
# perform PCA on subcounty data
subct.pca <- prcomp(census.subct[,-c(1,2)], scale = TRUE)
subct.pc <- as.data.frame(subct.pca$rotation[,1:2])


# perform PCA on county data
ct.pca <- prcomp(census.ct[,-c(1,2)], scale = TRUE)
ct.pc <- as.data.frame(ct.pca$rotation[,1:2])

#largest abs for ct
topct <- order(abs(ct.pc$PC1), decreasing = TRUE)[1:3]
kable(ct.pc[topct,],caption ="3 features with largest absolute values of PC1 for county
") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"), f
ull_width=FALSE)
```

3 features with largest absolute values of PC1 for county

	PC1	PC2
IncomePerCap	0.3463	0.1544
ChildPoverty	-0.3403	0.0597
Poverty	-0.3383	0.0854

```
#largest abs for subct
topsubct <- order(abs(subct.pc$PC1), decreasing = TRUE)[1:3]
kable(subct.pc[topsubct,],caption ="3 features with largest absolute values of PC1 for
sub-county") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"), f
ull_width=FALSE)
```

3 features with largest absolute values of PC1 for sub-county

	PC1	PC2
IncomePerCap	-0.3181	0.1679
Professional	-0.3064	0.1416
Poverty	0.3047	0.0514

We chose to center and scale the features before running PCA because otherwise, most of the principal components that we observed would be driven by a weighted variable that has the largest mean and variance. By doing so, we would remove biases in the original variables, now all the variables have the same standard

deviation and same weight. This process is known as standardization and is important because it puts an emphasis on variables with higher variances than those with low variances to help with identifying the right principal components.

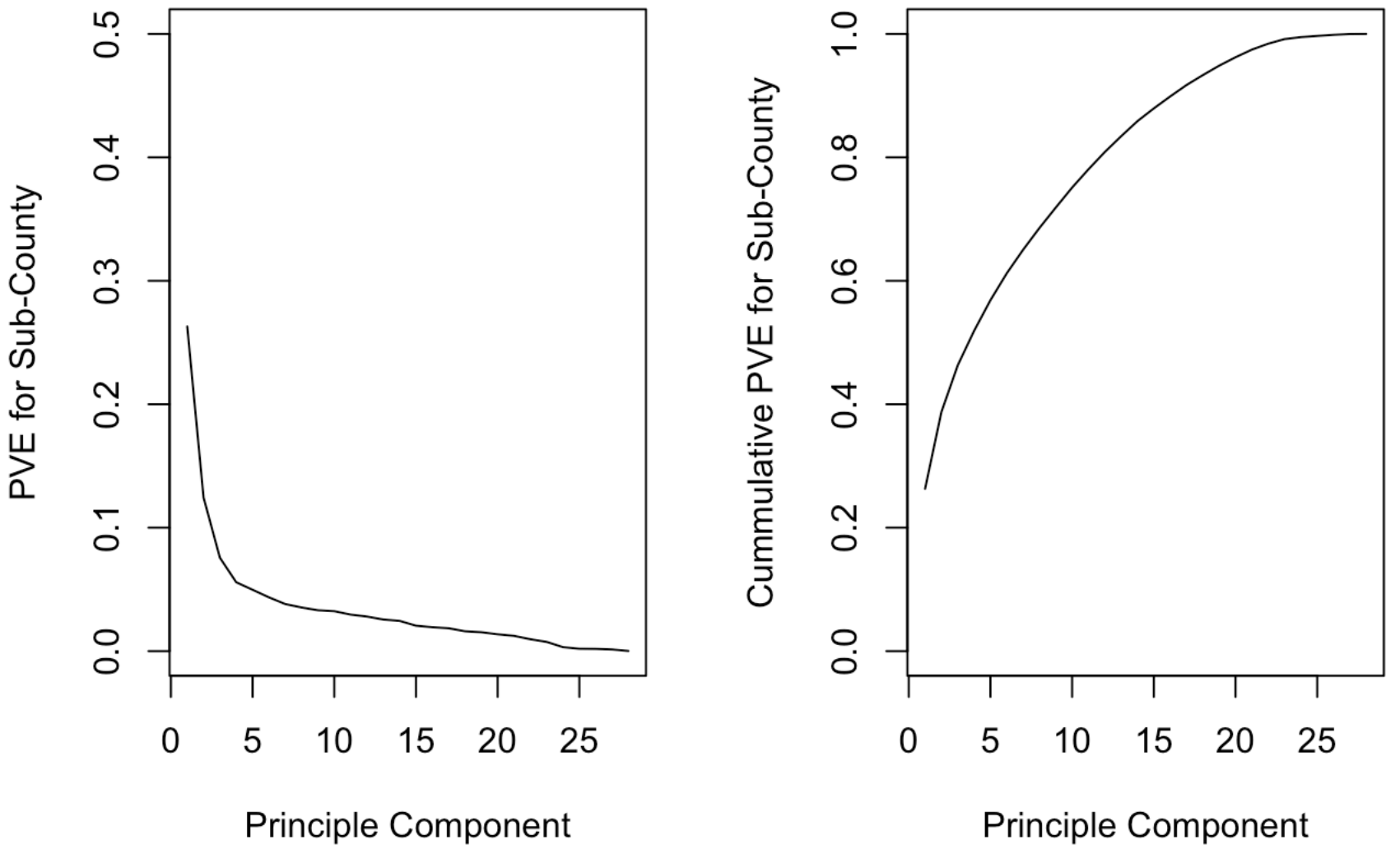
The three features with the largest absolute value of PC1 for sub-county are IncomePerCapita, Professionaland Poverty while the three largest absolute values of PC1 for county are IncomePerCaptia, ChildPoverty and Poverty.

For sub-county PC1, IncomePerCap and Professional have negative PC1 values while Poverty has a positive PC1 value. This indicates that Poverty and PC1 are positively correlated where the increase in one variable corresponds to an increase in the other. The positive sign also indicates the direction Poverty is going in the single dimension vector. The negative values for IncomePerCap and Professional indicates that these values are negatively correlated with PC1 where the increase in one corresponds to a decrease in the other. The sign also indicates the negative direction IncomePerCap and Professional is going in the single dimension vector.

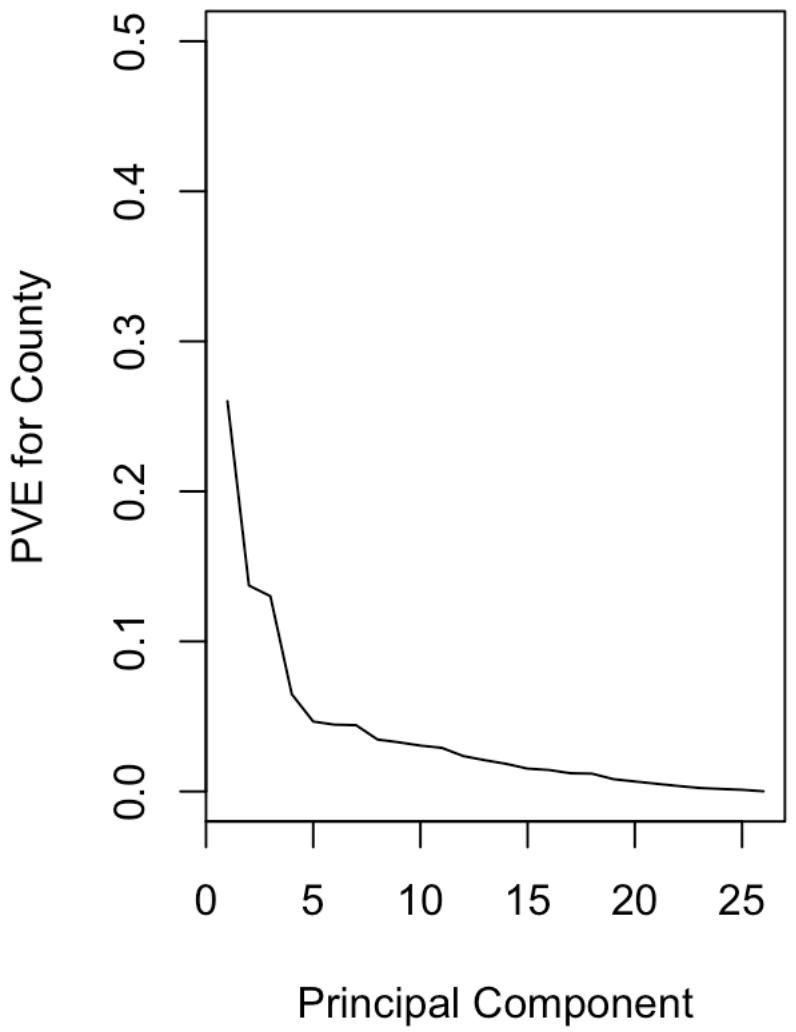
For county PC1, IncomePerCap has a negative PC1 while ChildPoverty and Povery have positive values for PC1. This indicates that Child Poverty and Poverty are positively correlated with PC1 where the increase in one corresponds to an increase in the other. The sign also indicates a positive direction that ChildProverty and Poverty are going in the single dimension vector. On the other hand, IncomePerCap is negatively correlated with PC1 and the increase in one corresponds to a decrease in the other. The sign for IncomePerCap indicates a negative direction that it is going in the single dimension vector.

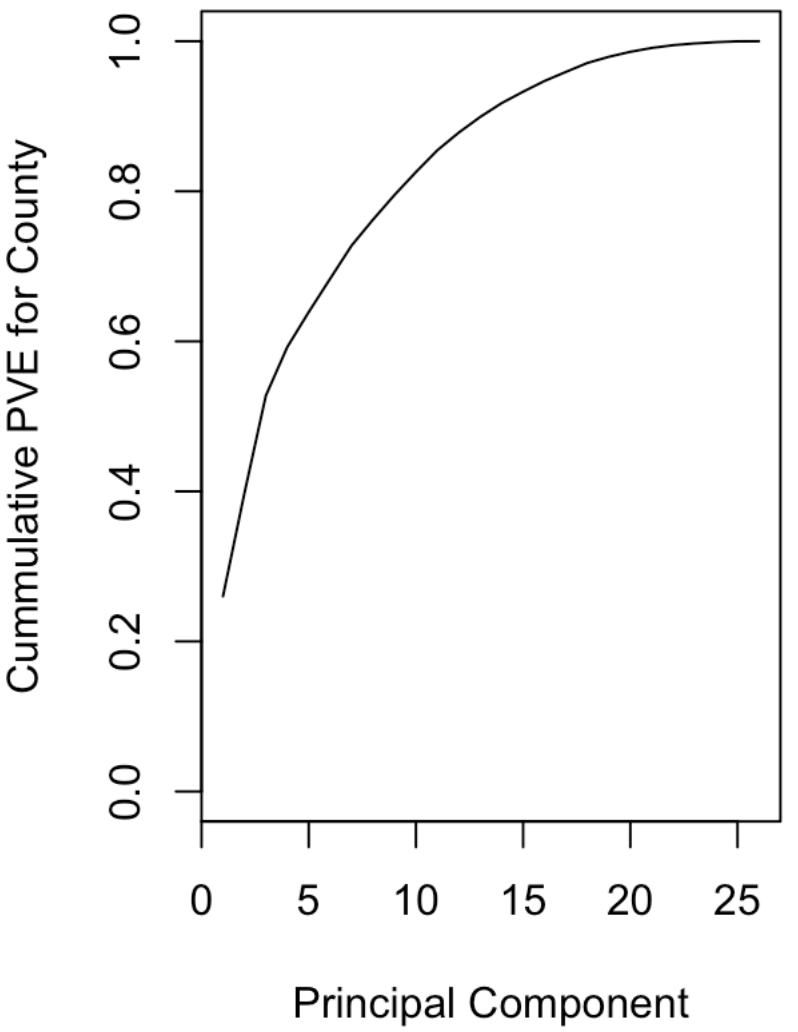
14. Determine the minimum number of PCs needed to capture 90% of the variance for both the county and sub-county analyses.

```
# calculate pve for subct
pr.subct.var <- subct.pca$sdev^2
pve.subct <- pr.subct.var/sum(pr.subct.var)

# the number of PCs needed to explain at least 90% of total variation for subcounty
min.subct.pc <- min(which(cumsum(pve.subct)>=0.9))
#min.subct.pc #17

### Plot PVE and Cumulative PVE
par(mfrow=c(1,2))
plot(pve.subct,xlab='Principle Component',ylab='PVE for Sub-County',type='l',ylim=c(0,0
.5))
plot(cumsum(pve.subct),xlab='Principle Component',ylab='Cummulative PVE for Sub-County'
,ylim=c(0,1),type='l')
```




```
# calculate pve for ct
pr.ct.var <- ct.pca$sdev^2
pve.ct <- pr.ct.var/sum(pr.ct.var)
min.ct.pc <- min(which(cumsum(pve.ct)>=0.9))
#min.ct.pc #14

par(mfrow=c(1, 2))
plot(pve.ct, xlab = "Principal Component", ylab = "PVE for County", type = "l", ylim =
c(0,0.5))
plot(cumsum(pve.ct), xlab = "Principal Component", ylab = "Cummulative PVE for County",
ylim = c(0,1), type = "l")
```

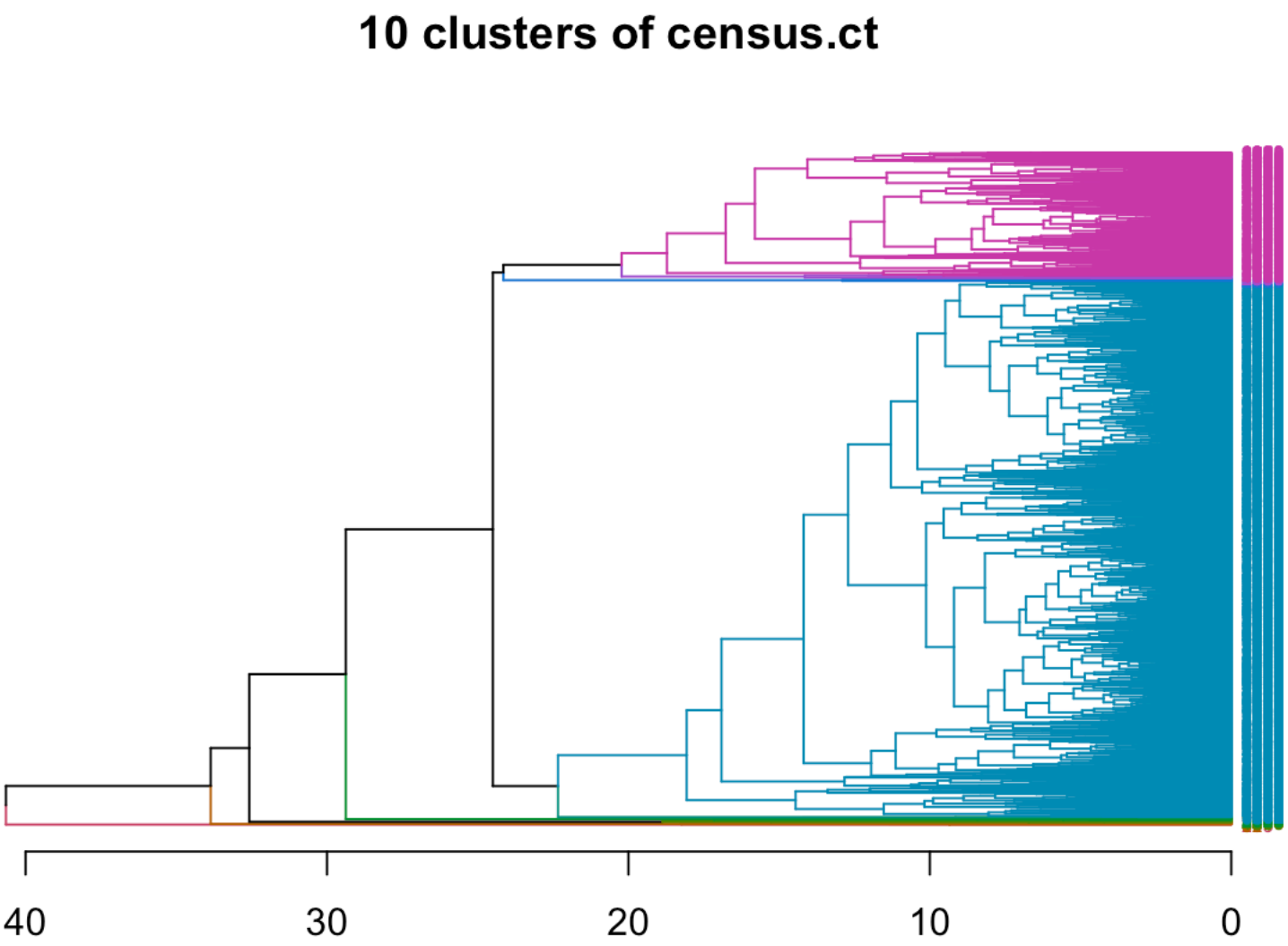


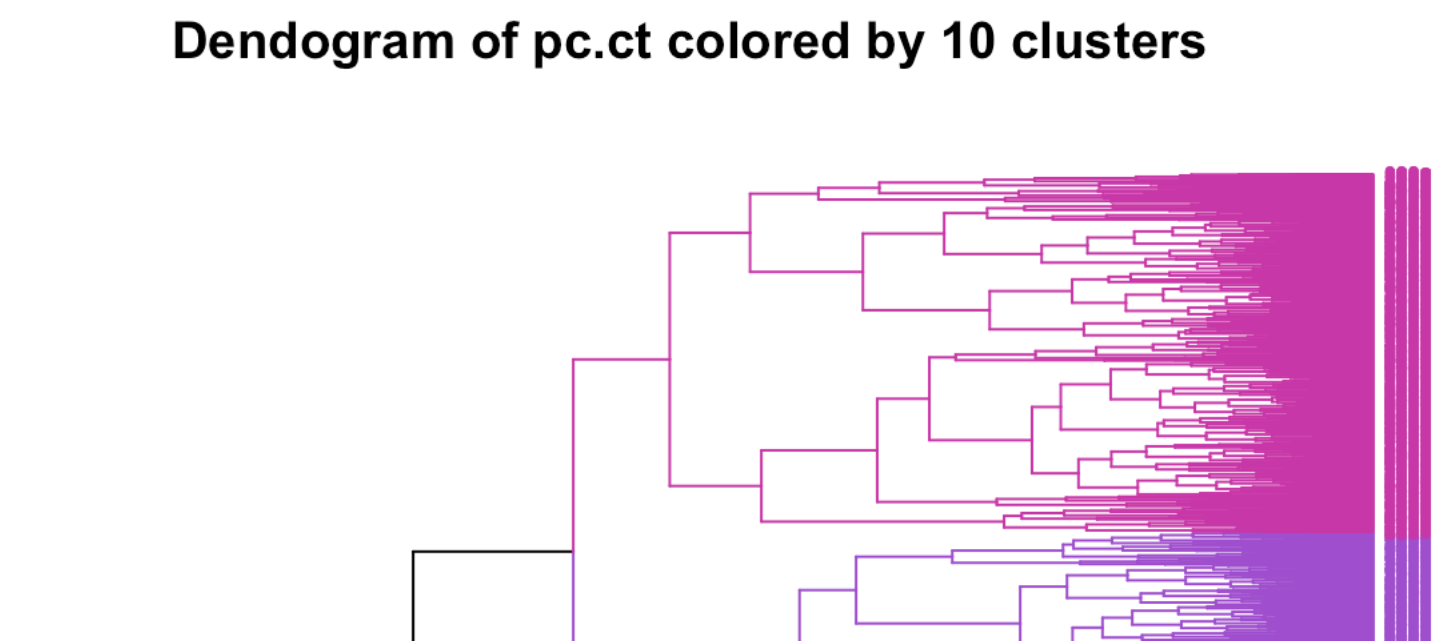


The minimum number of PCs needed to capture 90% of the variance for county is 14 and the minimum number if PCs needed to capture 90% of the variance for sub county is 17.

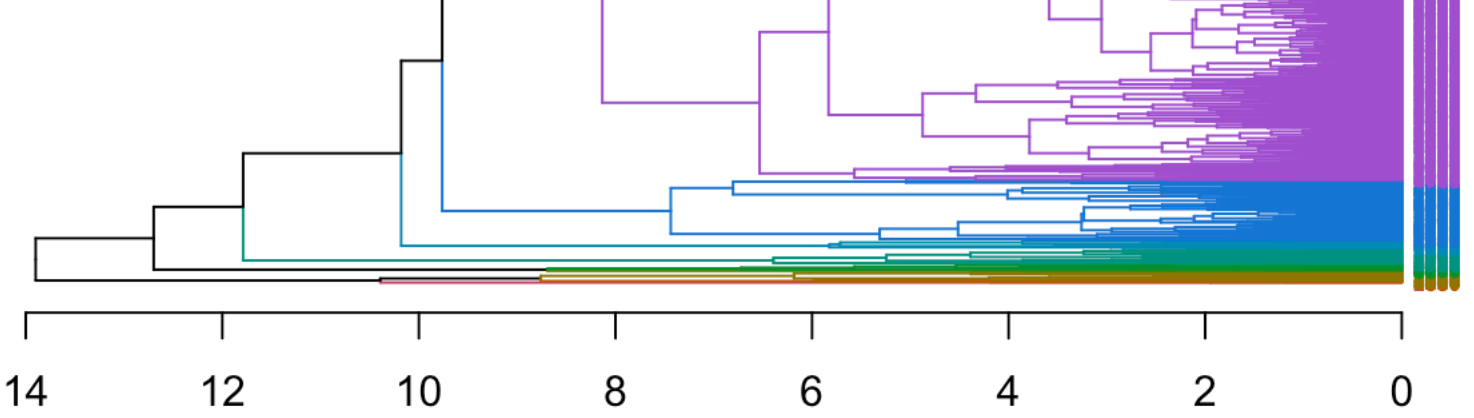
Clustering

###15. With census.ct, perform hierarchical clustering with complete linkage.

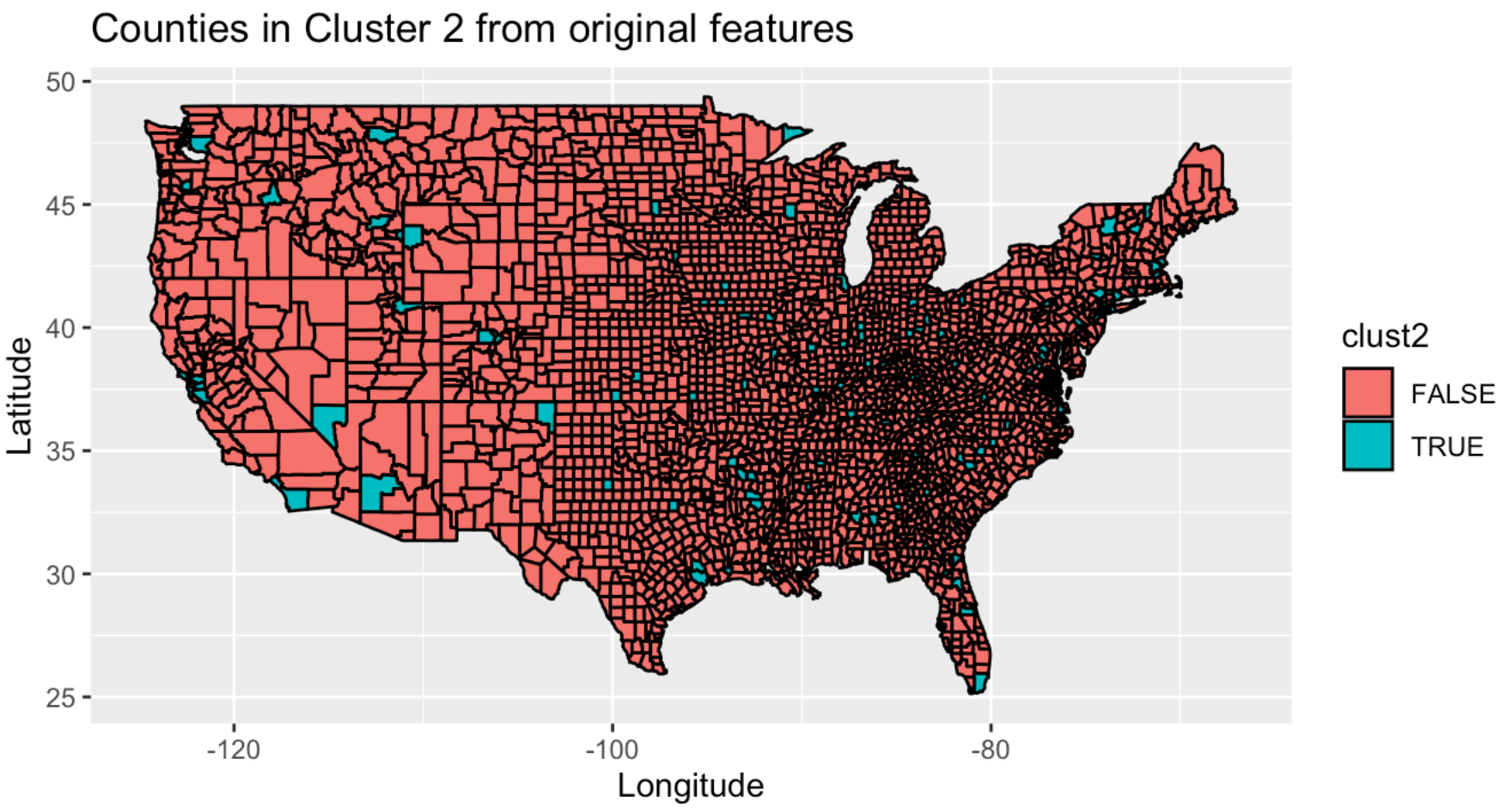


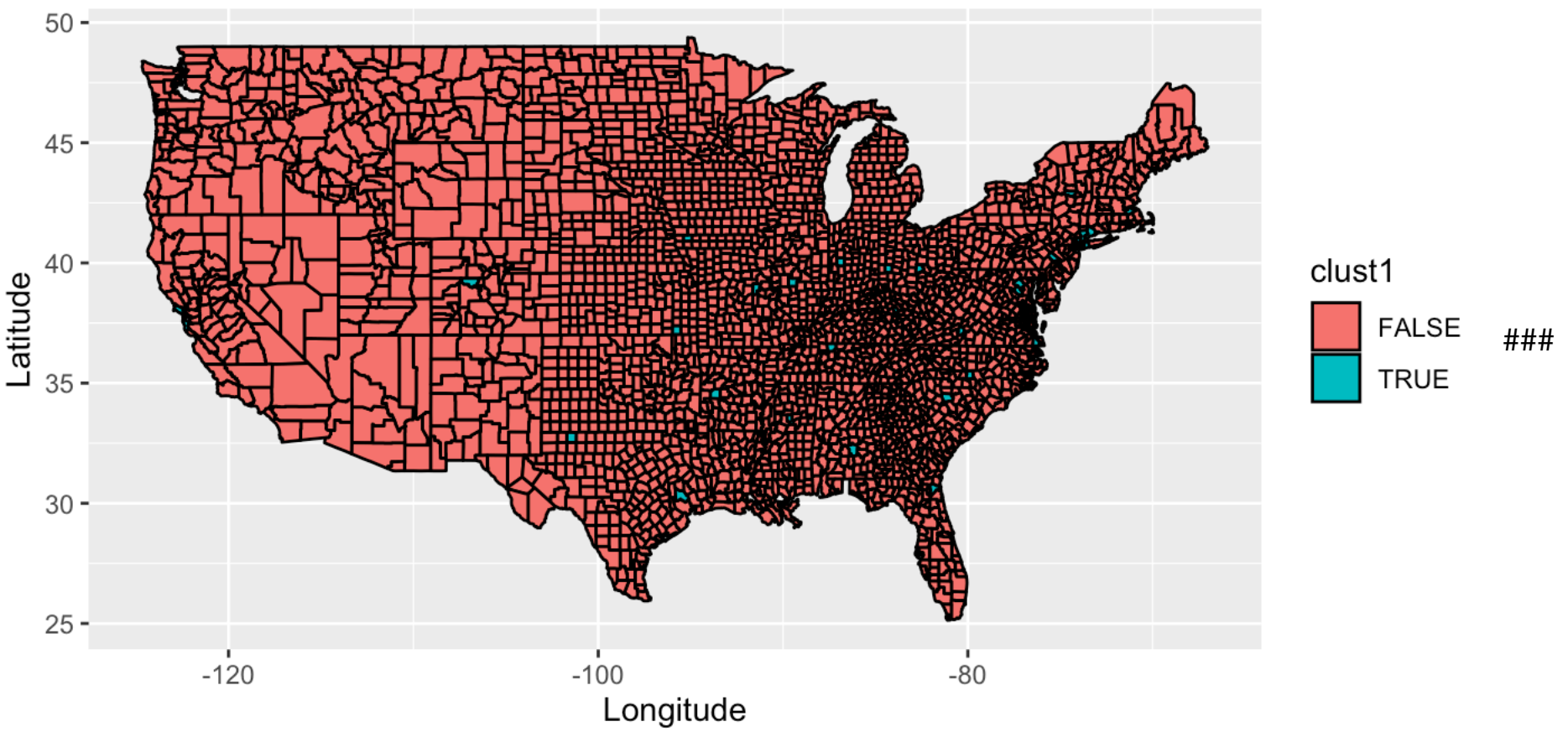


###


14
12
10
8
6
4
2
0

Cluster Map


Counties in Cluster 2 from original features
Latitude
50
45
40
35
30
25
-120
-100
-80
Longitude
clust2
FALSE
TRUE



Classification

```
tmpwinner <- county_winner %>% ungroup %>%
  mutate(state = state.name[match(state, state.abb)]) %>%                ## state abbrev
iations
  mutate_at(vars(state, county), tolower) %>%                            ## to all lower
case
  mutate(county = gsub(" county| columbia| city| parish", "", county))  ## remove suffi
xes
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))
```

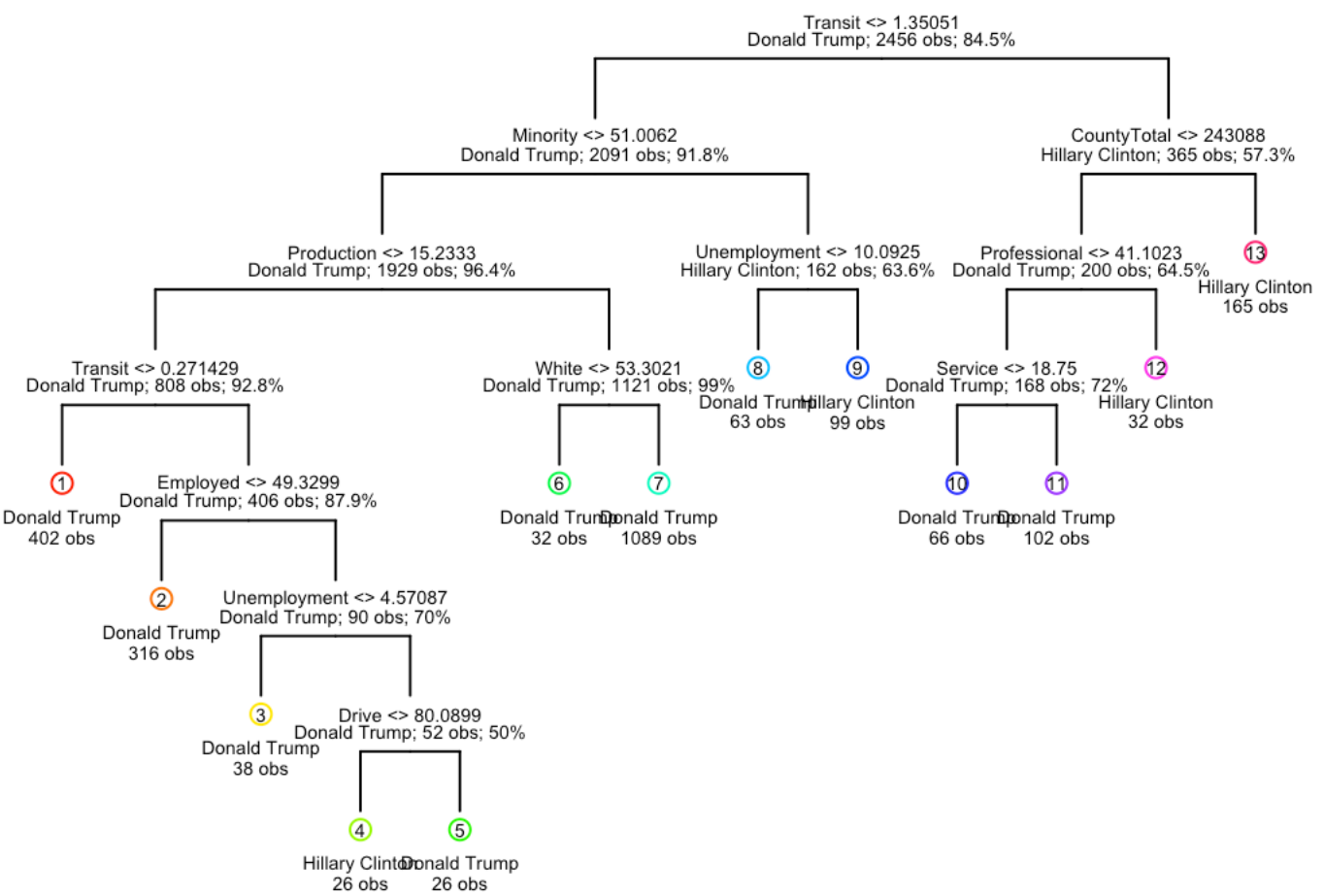
```
#Using the following code, partition data into 80% training and 20% testing:
set.seed(10)
n <- nrow(election.cl)
in.trn <- sample.int(n, 0.8*n)
trn.cl <- election.cl[ in.trn,]
tst.cl <- election.cl[-in.trn,]
```

```
#define a 10 cross-validation folds:

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))
}
records = matrix(NA, nrow=3, ncol=2)
colnames(records) = c("train.error","test.error")
rownames(records) = c("tree","logistic","lasso")
```

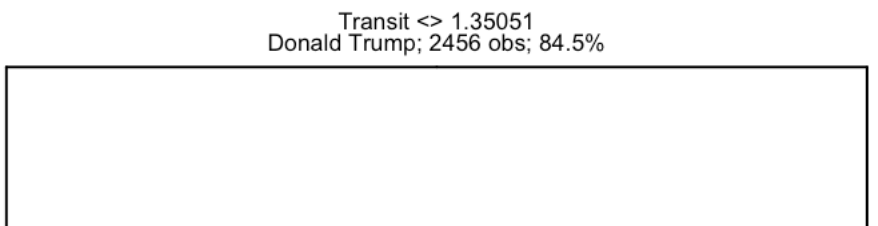
16. Decision tree: train a decision tree

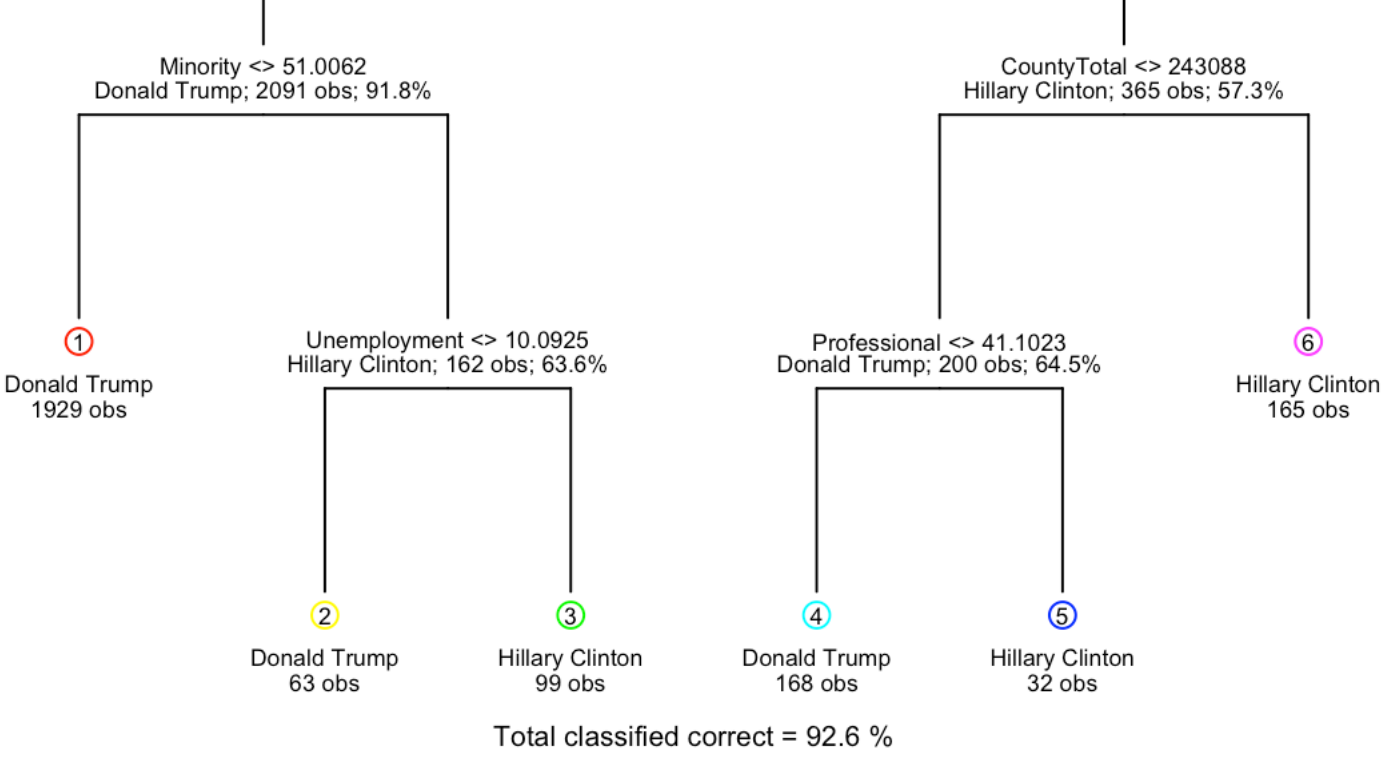
Classification tree for election Built on Training Set



Total classified correct = 93.4 %

Pruned Election Tree





Intepret and discuss the results.

We can see that the pruned tree has an overall lower error rate as compared to the pruned one but the difference is so small. In the end, it is still better for us to use the pruned tree since it's less complex and requires less work. The pruned decision tree overall provides a more clear visualization since almost all of pur previous observations from the unpruned tree still applies. In addition, it also helps us easily observe the different factor that can affect a voter's decision.

If the transit rate is less than 1.35 percent, and if the percentage of minorites is less than 51%, then it is 91.8% likely that Trump will win. If the percentage of minorities is greater than 51 percent, and if the unemployment rate is greater than 10.09%, it is 63.6% likely that Clinton will Win.

If the transit rate is greater than 1.35%, and if the County total is greater than 243,088, Hilary Clinton is 57.3% likely to win. If the County total is less than 243,088 and if the percentage of professionals in the county is greater than 41.1%, then it is 64.5% likely that Donald Trump will win in that county.

```
##           train.error test.error
## unpruned      0.06637    0.07317
## pruned        0.07370    0.07154
```

	train.error	**test.error**
tree	0.0737	0.0715
logistic	NA	NA
lasso	NA	NA

Using the new pruned tree with best size from cross validation we are able to find our test and train error. Although the unpruned tree has an overall low error rate, the difference between the values are very small. Given this factor we would choose the pruned tree as it has a smaller size and is less complex.

17. Run a logistic regression to predict the winning candidate in each county.

```
# we cannot do logistic regression on non numeric values
trn.clN <- trn.cl %>% select(-candidate)
trn.clY <- trn.cl$candidate
tst.clN <- tst.cl %>% select(-candidate)
tst.clY <- tst.cl$candidate

# logistic regression model on election train data
glm.election <- glm(candidate~., data =  trn.cl, family = "binomial")
```

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

```
election.fitted.train <- predict(glm.election, trn.clN, type = "response")
glm.pred.train <- rep("Donald Trump", length(trn.clY))
glm.pred.train[election.fitted.train>0.5] = "Hillary Clinton"

# logistic regression model on election test data
election.fitted.test <- predict(glm.election, tst.clN, type = "response")
glm.pred.test <- rep("Donald Trump", length(tst.clY))
glm.pred.test[election.fitted.test>0.5] = "Hillary Clinton"
records[2,1] <- calc_error_rate(glm.pred.train,trn.clY)
records[2,2] <- calc_error_rate(glm.pred.test,tst.clY)
kable(records[c(1,2),c(1,2)])
```

	train.error	test.error
tree	0.0737	0.0715
logistic	0.0741	0.0715

```
# summary
summary(glm.election)
```

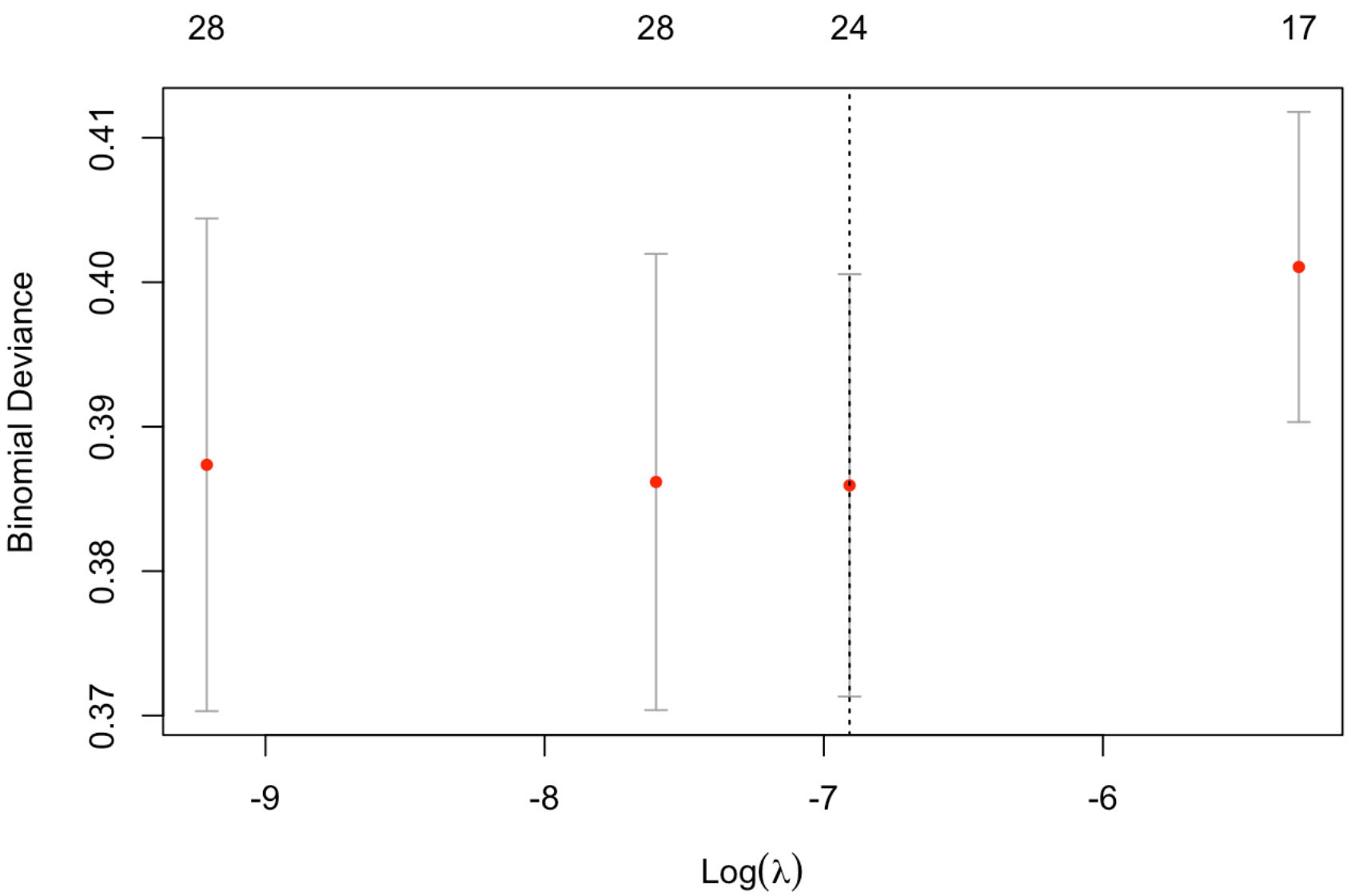
```
##
## Call:
## glm(formula = candidate ~ ., family = "binomial", data = trn.cl)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -3.547  -0.267  -0.119  -0.045   3.465
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -2.57e+01   9.94e+00   -2.58  0.00983 **
## Men              1.02e-01   5.28e-02    1.93  0.05421 .
## White           -1.10e-01   7.23e-02   -1.51  0.12985
## Citizen          1.41e-01   2.86e-02    4.95  7.6e-07 ***
## Income          -7.46e-05   2.82e-05   -2.64  0.00822 **
## IncomeErr       -7.82e-06   6.36e-05   -0.12  0.90209
## IncomePerCap     2.48e-04   6.69e-05    3.71  0.00021 ***
## IncomePerCapErr -3.08e-04   1.67e-04   -1.84  0.06511 .
## Poverty          6.02e-02   4.22e-02    1.43  0.15354
## ChildPoverty    -2.14e-02   2.52e-02   -0.85  0.39614
## Professional     2.42e-01   3.80e-02    6.38  1.7e-10 ***
## Service          2.77e-01   4.49e-02    6.18  6.5e-10 ***
## Office           3.97e-02   4.33e-02    0.92  0.35896
## Production       1.40e-01   4.00e-02    3.50  0.00047 ***
## Drive           -2.27e-01   4.71e-02   -4.82  1.4e-06 ***
## Carpool         -1.73e-01   6.03e-02   -2.87  0.00408 **
## Transit          6.62e-02   9.34e-02    0.71  0.47855
## OtherTransp     -8.02e-02   9.67e-02   -0.83  0.40694
## WorkAtHome      -1.78e-01   7.20e-02   -2.47  0.01348 *
## MeanCommute      6.74e-02   2.40e-02    2.81  0.00488 **
## Employed         1.98e-01   3.30e-02    6.01  1.8e-09 ***
## PrivateWork      8.41e-02   2.11e-02    3.99  6.5e-05 ***
## SelfEmployed     4.27e-02   4.81e-02    0.89  0.37465
## FamilyWork      -6.31e-01   3.82e-01   -1.65  0.09807 .
## Unemployment     1.98e-01   3.96e-02    5.00  5.7e-07 ***
## Minority         2.43e-02   7.01e-02    0.35  0.72947
## CountyTotal      4.86e-07   3.94e-07    1.23  0.21772
## Cluster         -1.03e-01   1.53e-01   -0.68  0.49860
## Cluster_PC      -2.30e-01   9.67e-02   -2.38  0.01735 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2119.6  on 2455  degrees of freedom
## Residual deviance:  871.4  on 2427  degrees of freedom
## AIC: 929.4
##
## Number of Fisher Scoring iterations: 7
```

Citizen, IncomePerCap, Professional, Service, Production, Drive, Carpool, Employed, PrivateWork, and Unemployment are important predictors as they have a significance level between 0 and 0.001. This means that the p-value for these variables are significantly smaller than alpha values which rejects the null hypothesis that each variable has a coefficient of 0. To conclude, we are 99.9% confident that all the predictors listed are important predictors for the logistic model. At a 99% confidence level, Intercept, Carpool, and Income are also considered important predictors including the listed predictors. At a 95% confidence level, Men, White, IncomePerCapErr, WorkAtHome, MeanComute, and FamilyWork are added to the list of significant predictors.

This is not consistent with the decision tree analysis. The largest split on the decision tree is on the Transit variable followed by White and CountyTotal which are not considered significant variables in the logistic regression, except White is significant at a 95% confidence level.

18.

```
# code categorical predictor variables
x <- model.matrix(candidate~., trn.cl)[,-1]
# Convert the outcome (class) to a numerical variable
y <- ifelse(trn.cl$candidate == "Hillary Clinton", 1, 0)

set.seed(1)
# control overfitting in logistic regression is through regularization
cv.lasso <- cv.glmnet(x=x,y=y,family='binomial',alpha=1,lambda=c(1,5,10,50)*1e-4)
plot(cv.lasso)
```



```
# optimal lambda
bestlambda <- cv.lasso$lambda.min #0.001
# Fit the final model on the training data
log.lasso <- glmnet(x=x,y=y, alpha = 1, family = "binomial",
                lambda = cv.lasso$lambda.min)
coef(log.lasso)
```

```
## 29 x 1 sparse Matrix of class "dgCMatrix"
##                          s0
## (Intercept)     -2.428e+01
## Men              4.488e-02
## White           -8.875e-02
## Citizen          1.501e-01
## Income          -2.918e-05
## IncomeErr       -2.793e-05
## IncomePerCap     1.305e-04
## IncomePerCapErr -1.137e-04
## Poverty          3.913e-02
## ChildPoverty     .
## Professional     1.973e-01
## Service          2.264e-01
## Office           .
## Production       8.575e-02
## Drive           -1.669e-01
## Carpool         -1.160e-01
## Transit          1.002e-01
## OtherTransp      .
## WorkAtHome      -9.068e-02
## MeanCommute      3.655e-02
## Employed         1.811e-01
## PrivateWork      7.395e-02
## SelfEmployed     .
## FamilyWork      -4.649e-01
## Unemployment     1.785e-01
## Minority         3.072e-02
## CountyTotal      4.825e-07
## Cluster         -3.481e-02
## Cluster_PC      -1.864e-01
```

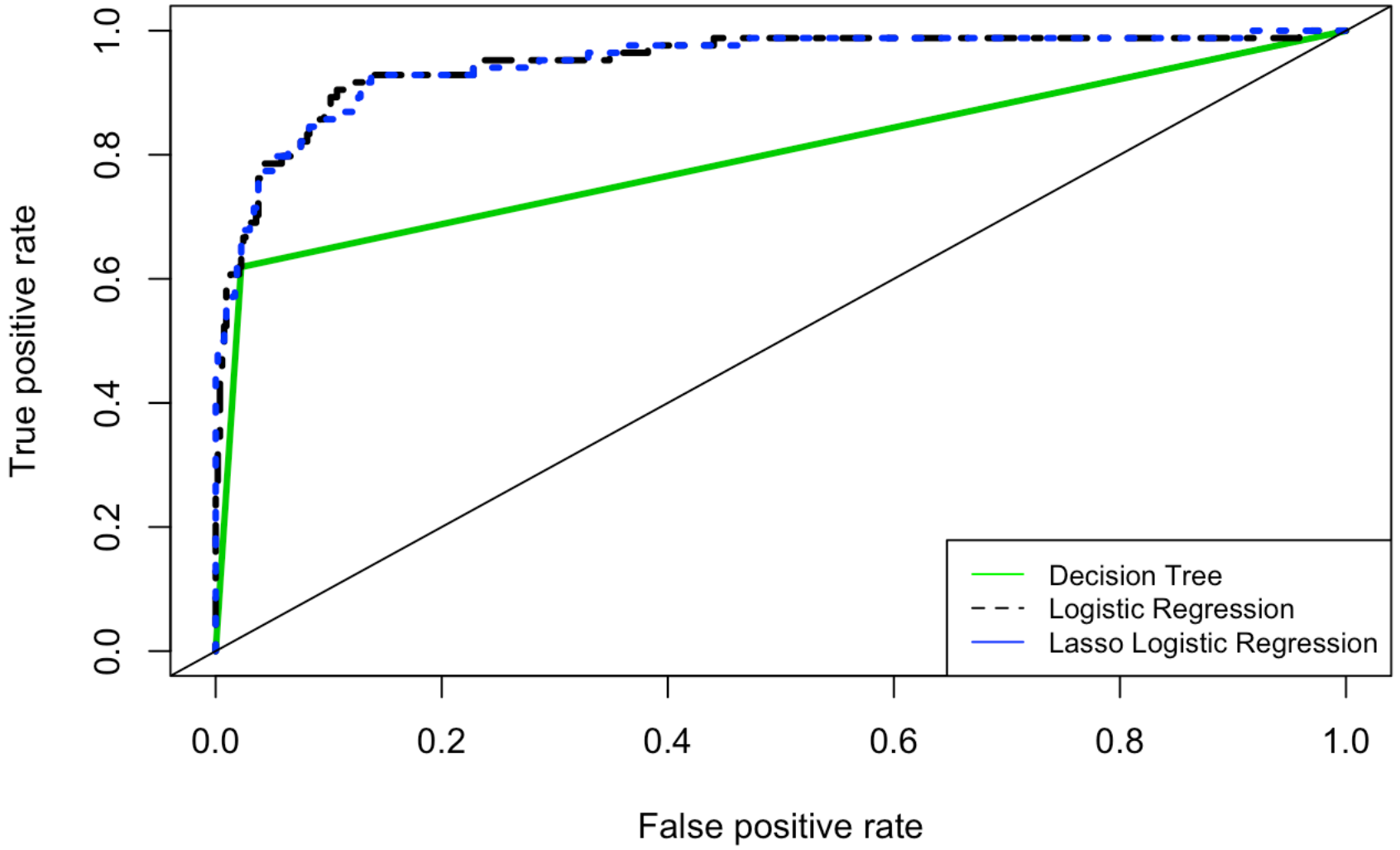
```
# Make predictions on the train data
lasso.train.prob <- predict(log.lasso,x,type='response')
pred.train.class <- ifelse(lasso.train.prob>0.5,'Hillary Clinton','Donald Trump')

# Make predictions on the test data
x2 <- model.matrix(candidate~.,tst.cl)[,-1]
lasso.test.prob <- predict(log.lasso,x2,type = 'response')
pred.test.class <- ifelse(lasso.test.prob>0.5,'Hillary Clinton','Donald Trump')

# Model accuracy
lasso.test.error <- calc_error_rate(pred.test.class,tst.cl$candidate)
lasso.train.error <- calc_error_rate(pred.train.class,trn.cl$candidate)
records[3,2] <- lasso.test.error
records[3,1] <- lasso.train.error
kable(records)
```

	train.error	test.error
tree	0.0737	0.0715
logistic	0.0741	0.0715
lasso	0.0733	0.0683

The optimal λ value in cross validation is 0.001. The non-zero coefficients in the LASSO regression for the optimal value of λ were all of the variables excluding Transit, Self-Employed, and Minority because many of the variables in our dataset affect the outcome. The LASSO regression is used for data sets with not enough data, which has high variance estimates; it enables us to use the shinkrage method and this lowers our variance. This is in contrast to logistic regression, which is better for big data. Here, men, Office, MeanCommute, and PrivateWork are the largest non-zero coefficients and these also have the highest estimates from logistic regression. Let's remember that logistic regression is unpenalized, and on the other hand the LASSO regression has less variables to work with because some of the variables are equal to 0. In conclusion, the LASSO and logistic regression fits are similar. Since the errors are so close to each other; then we can concoude that the LASSO regression does not provide any extra insight.

19. Compute ROC curves for the decision tree, logistic regression and LASSO logistic regression using predictions on the test data.

```
pruned.pred.tree <- predict(pruned.election.tree, tst.clN, type = "class")
# make prediction on candidate: tree (as numeric)
pred.tree <- prediction(as.numeric(pruned.pred.tree), as.numeric(tst.clY))
# make prediction on candidate: logistic (as numeric)
pred.logis <- prediction(as.numeric(election.fitted.test), as.numeric(tst.clY))
# make prediction on candidate: lasso (as numeric)
pred.lasso <- prediction(lasso.test.prob, as.numeric(tst.clY))
# calculate the performance for each of the processes
tree.perf <- performance(pred.tree, measure = "tpr", x.measure = "fpr")
logis.perf <- performance(pred.logis, measure = "tpr", x.measure = "fpr")
lasso.perf <- performance(pred.lasso, measure = "tpr", x.measure = "fpr")
# plotting each of the ROC curves
plot(tree.perf, col = 3, lwd = 3, main = "All ROC Curves")
plot(logis.perf, col = 1, lty= 4,  lwd = 3, main = "All ROC Curves", add = TRUE)
plot(lasso.perf, col = 4, lty= 3, lwd = 3, main = "All ROC Curves", add = TRUE)
legend("bottomright" ,legend=c("Decision Tree", "Logistic Regression", "Lasso Logistic
Regression"),
       col=c("green", "black","blue"), lty=1:2, cex=0.8)
abline(0,1)
```



```
# calculate AUC
auc_tree = performance(pred.tree,"auc")@y.values
auc_logis = performance(pred.logis,"auc")@y.values
auc_lasso = performance(pred.lasso,"auc")@y.values
# creating a matrix to store it
auc.records = matrix(NA, nrow=1, ncol=3)
colnames(auc.records) <- c("Decision Tree", "Logistic Regression", "Lasso Logistic Regr
ession")
rownames(auc.records) <- "Area Under the Curve"
auc.records[1,1] =  auc_tree[[1]][1]
auc.records[1,2] = auc_logis[[1]][1]
auc.records[1,3] = auc_lasso [[1]][1]
auc.records
```

```
##                       Decision Tree Logistic Regression
## Area Under the Curve        0.7982              0.9483
##                       Lasso Logistic Regression
## Area Under the Curve                     0.948
```

Decision trees are very simple to use but they do not have the best accuracy. Since they also have high variance and tend to overfit, any small changes can lead to a completely different tree. This form of classification will only work well if the data can easily be split into rectangular regions. Logistic regression is good for classifying between two different values. In this class, we are classifying the election result for each county (either Hillary Clinton or Donald Trump). However, if the data is linear or has complete separation, it will be harder to classify. Lasso Regression is most useful when some predictors are redundant and can be removed. Much like all regularization methods as well as logistic regression, Lasso Regression tends to have a lower variance and does not overfit as much. Since it ignores non significant variables, that may be problematic because we'll never know how interesting or uninteresting they are.

Based on the result from the AUC calculation, decision trees perform poorly while logistic and lasso regression perform pretty much the same with values.

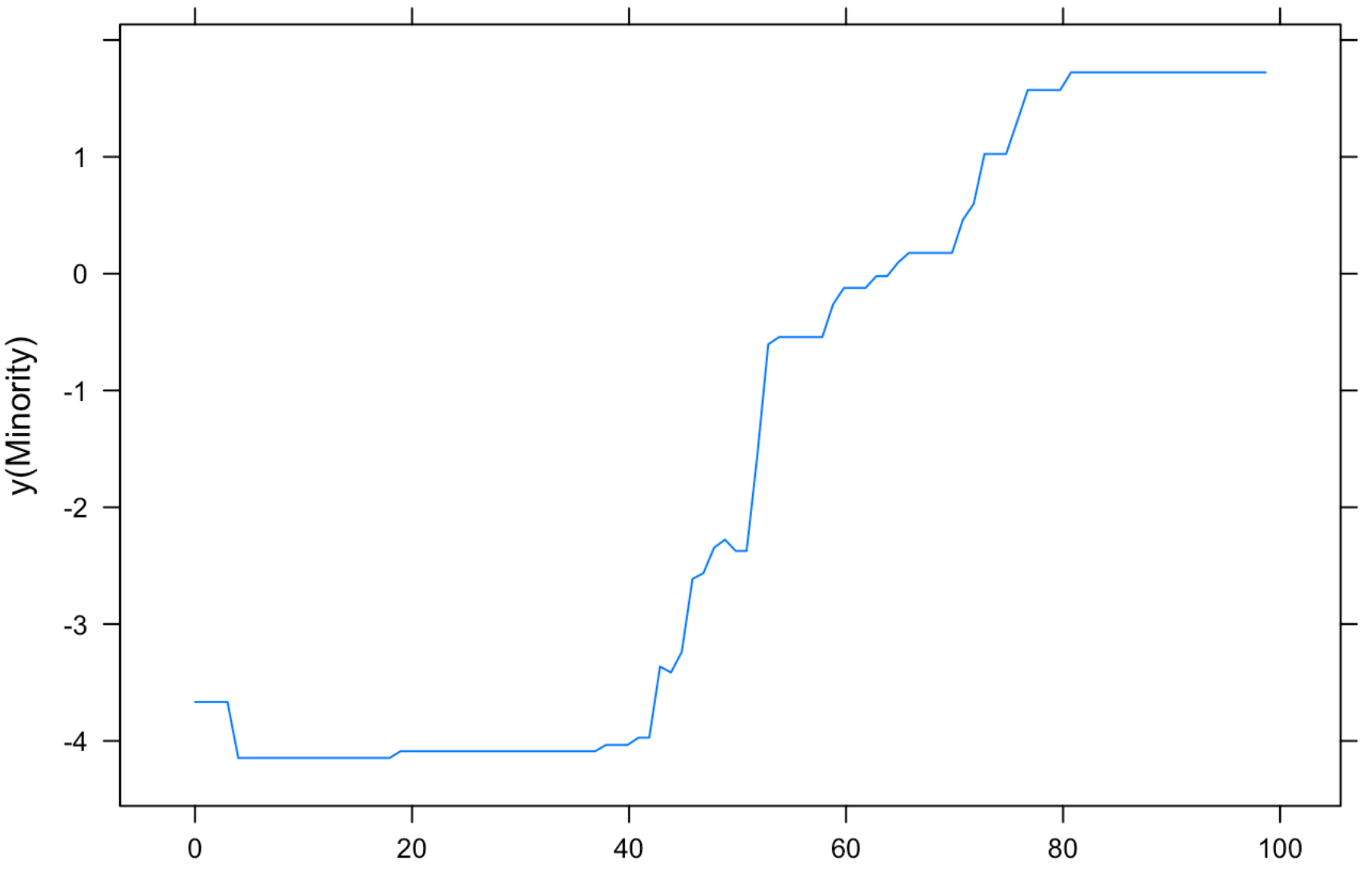
We would choose the model which has an AUC value closer to 1. Since the election data couldn't easily fit in a rectangular region, using a decision tree classifier wasn't the best for classifying election results.

20.

We found that some key factors that may have an impact on the election may be transit, county total, white, and unemployment from the decision tree model. Other important predictors identified from the logistic regression model were citizen, income per cap, professional, service, production, drive, employed, and private work. Amongst these variables, service and professional have a greater impact on the probability of a candidate winning the election.

For this question we decided to choose the section of "Exploring additional classification methods" and chose: boosting; random forests and KNN. We will fit our given data to these 3 models and see which one is the best out of the 3. Finally we will compare them against logistic regression and tree models. Thus we conlcude that the latter two methods are more appropriate. These methods are able to give us better insight on the voter behavior. The LASSO helps us with the large data set and the logistic regression helps us identify the best candidate in the election for each variable group through separation.

Boosting

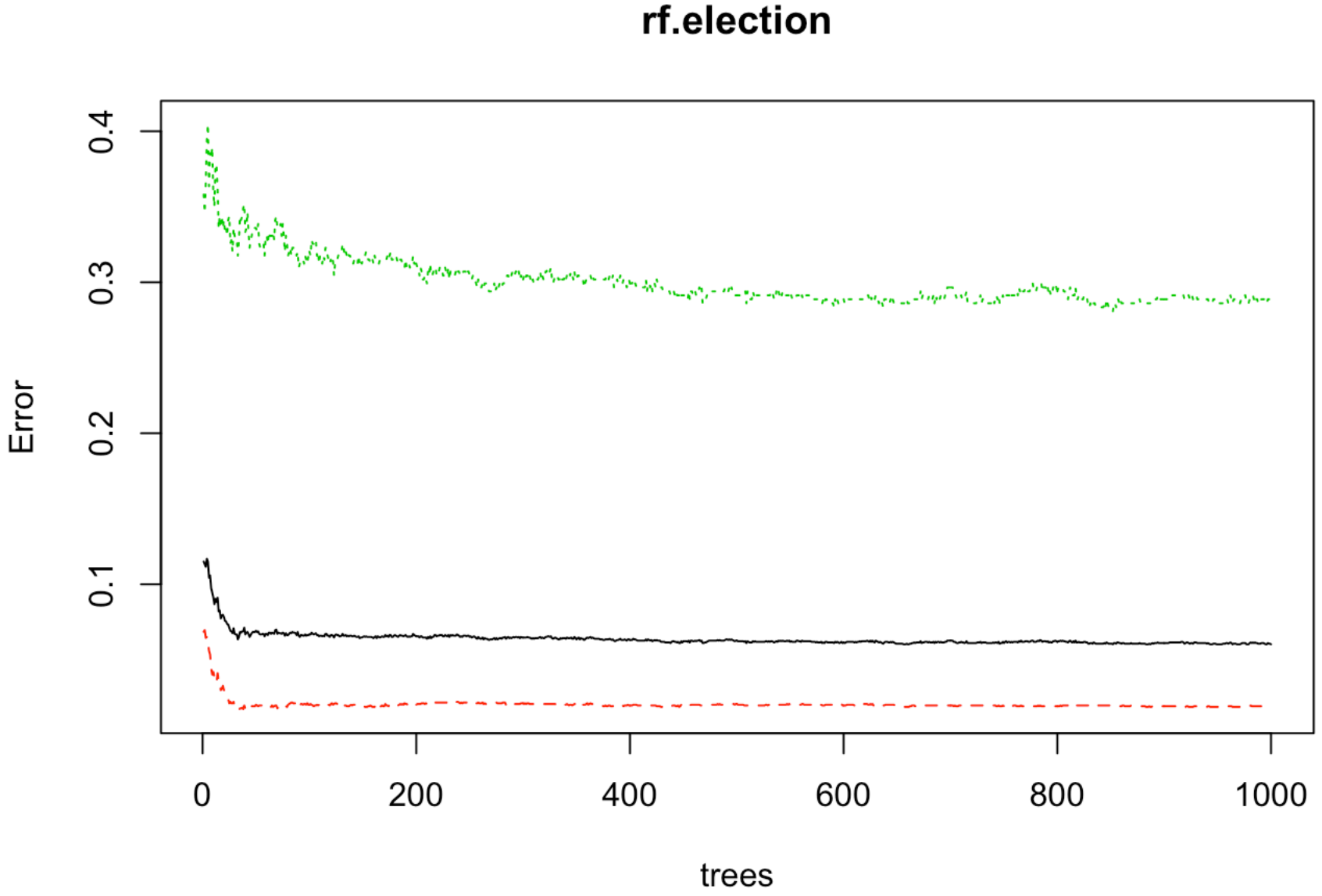



```
## [1] 0.9984
```

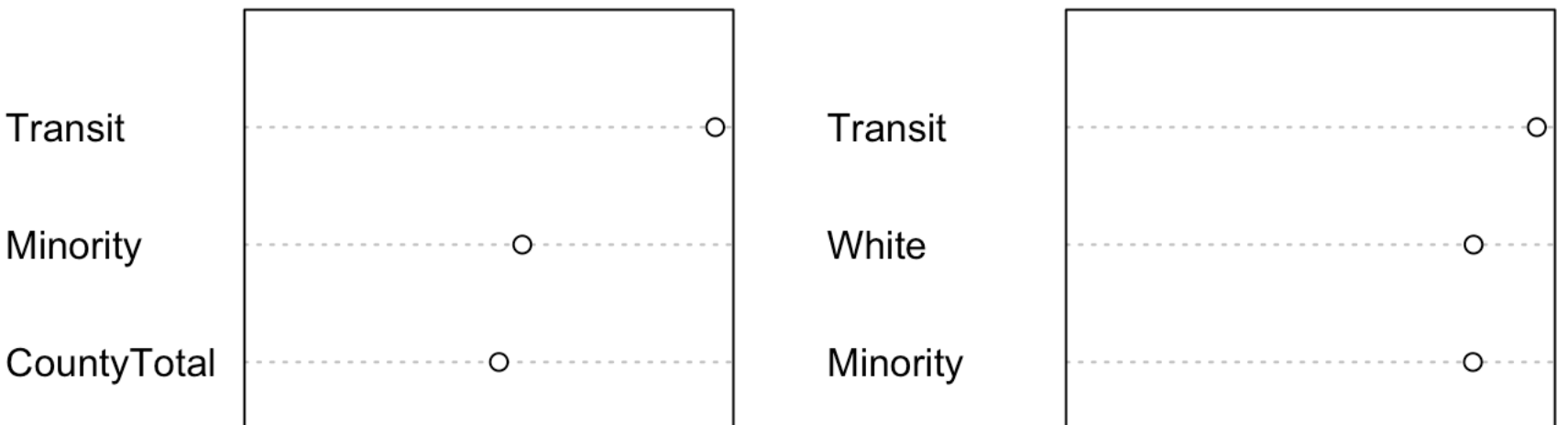
We first explored the boosting method. This gave us an error of 0.9984, which is very high. We explain this by recalling that boosting works best for smaller data sets and decision trees. Overall, since we have a large data set compared; the boosting method fails to perform accurately as compared to logistic regression and tree methods. Boosting gives us plots and graphs that do not really help relate variables correctly to data nor voter behavior.

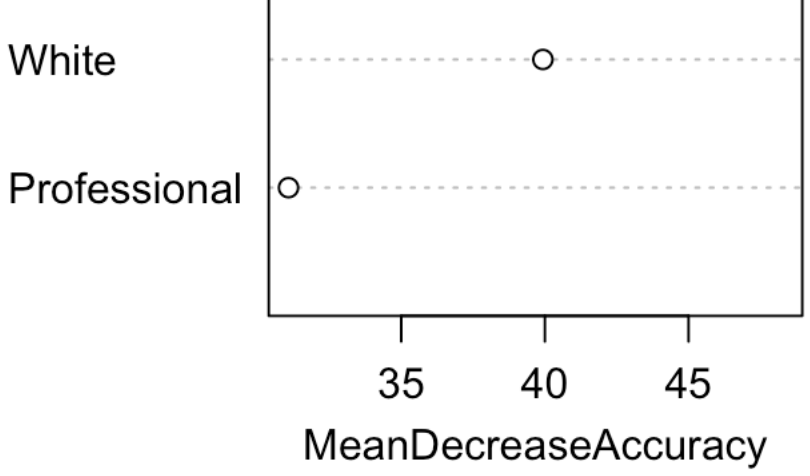
Random Forest

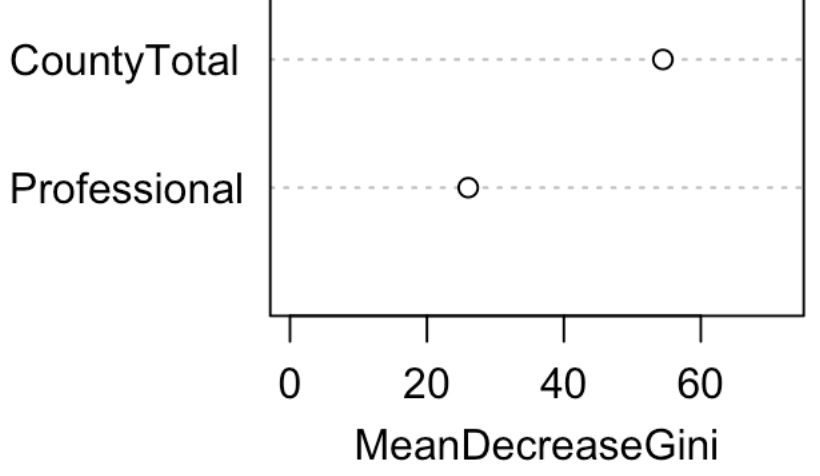
Next, we look at the method of Random Forest. For this; we get an error of 0.05366, which is a lot smaller than what we got with boosting. This is small but it's still a good result. Looking at the Variance Importance charts; we notice that the variables Transit, White and Minority are the most important for minimizing Gini impurity. This data resembles what we got with our decision trees in the sense that the top of the trees has the most important variables and the less important ones are placed down on the tree branches. We know that voter's demographic and their social-economic status play a big role in their voting behavior. For instance; minorities will tend to vote for Clinton while whites will tend to vote for Trump. Thus; we can conclude that even though the Random Forest Method is more prone to over-fitting; it gives more information that helps us see which predictors are more important.



Variable Importance for Random Forest Election








```
## [1] 0.05366
```

KNN

Lastly, we used knn.cv to do a KNN classification on a training set using LOOCV. After determining the best k value is 16, the test error rate comes to 0.1138 and the training error comes to 0.112. This is a much larger error than the other methods explored. This may be because our data is more linear and since KNN classification is nonparametic, it is subject to overfitting due to its high variance.

```
## [1] 16
```

```
## [1] 0.1138
```

```
## [1] 0.112
```

This last question helps us analyze 3 other differet methods that were not included in the project itself. This allowed us to see that between Boosting, Random Forest and KNN, the Random Forest method is best. This is becaue random forest yeilds a much lower error that is more similar to the logisitc regression and the tree methods which are the other meothds that we worked with previously.