2016 Election Prediction Project

Arthur Hla PSTAT 131

Background

The presidential election in 2012 did not come as a surprise. Some correctly 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 to two paragraphs for each.

1. What makes voter behavior prediction (and thus election forecasting) a hard problem? Voter behavior is hard to predict because there is no accurate model and the best we can do to determine voter behavior is by taking demographics of the area. Even this fails to predict human behavior because even the slight factor can change one's decision totally. Demographics leaves a lot of variance unexplained. For example, cues we get from social interactions, impaired judgments, and anything from facial expression to an accidental statement in speech can overturn the decision making which ultimately affects the outcomes.

2. What was unique to Nate Silver's approach in 2012 that allowed him to achieve good predictions? The uniqueness of Nate Silver's prediction is that he was able to minimize the variance of predicting voter's behavior using a different approach. He first created what he calls a nowcast which is a mathematical model of how people in the US states will vote if the election were held on any particular day. After that, he starts a timer where voting behavior will change with respect to time. Since as time goes on the there is more room for variance so he treats all of its uncertainties as a random component. Then he classifies that random component by classifying above 50% is the outcome that Obama would win. This is a good way to describe voters but he still has to connect it to polls and the election.

3. What went wrong in 2016? What do you think should be done to make future predictions better? In 2016, the final prediction was that Clinton had a 71% chance of winning but there was an error in the polls. We get our data from polls and make a prediction, but if the polls are missed then the predictions become skewed and bias. This is what happened in Tuesday's poll, where there was a large error than the expected 2 to 3 percent. The polls were underestimating Trump's winnings. To make prediction better, we need to better understand the poll's uncertainties and take them into account when doing our modeling, since if the data is wrong then our modeling is wrong.

Data

```
election.raw = read.csv("final-project/data/election/election.csv") %>% as.tbl
census_meta = read.csv("final-project/data/census/metadata.csv", sep = ";") %>%
    as.tbl
census = read.csv("final-project/data/census/census.csv") %>% as.tbl
census$CensusTract = as.factor(census$CensusTract)
```

Election data

Following is the first few rows of the `election.raw` data:

county	fips	candidate	state	votes
NA	US	Donald Trump	US	62984825
NA	US	Hillary Clinton	US	65853516
NA	US	Gary Johnson	US	4489221
NA	US	Jill Stein	US	1429596
NA	US	Evan McMullin	US	510002
NA	US	Darrell Castle	US	186545

In our dataset, `fips` values denote the area (US, state, or county) that each row of data represents: i.e., some rows in `election.raw` are summary rows. 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 state as `fips` value.

Census data

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

CensusTract	State	County	TotalPop	Men	Women	Hispanic	White	Black	Native	Asian	Pacific
1001020100	Alabama	Autauga	1948	940	1008	0.9	87.4	7.7	0.3	0.6	0.0
1001020200	Alabama	Autauga	2156	1059	1097	0.8	40.4	53.3	0.0	2.3	0.0
1001020300	Alabama	Autauga	2968	1364	1604	0.0	74.5	18.6	0.5	1.4	0.3
1001020400	Alabama	Autauga	4423	2172	2251	10.5	82.8	3.7	1.6	0.0	0.0
1001020500	Alabama	Autauga	10763	4922	5841	0.7	68.5	24.8	0.0	3.8	0.0
1001020600	Alabama	Autauga	3851	1787	2064	13.1	72.9	11.9	0.0	0.0	0.0

Citizen	Income	IncomeErr	IncomePerCap	IncomePerCapErr	Poverty	ChildPoverty	Professional	Service
1503	61838	11900	25713	4548	8.1	8.4	34.7	17.0
1662	32303	13538	18021	2474	25.5	40.3	22.3	24.7
2335	44922	5629	20689	2817	12.7	19.7	31.4	24.9
3306	54329	7003	24125	2870	2.1	1.6	27.0	20.8
7666	51965	6935	27526	2813	11.4	17.5	49.6	14.2
2642	63092	9585	30480	7550	14.4	21.9	24.2	17.5

Office	Construction	Production	Drive	Carpool	Transit	Walk	OtherTransp	WorkAtHome
21.3	11.9	15.2	90.2	4.8	0	0.5	2.3	2.1
21.5	9.4	22.0	86.3	13.1	0	0.0	0.7	0.0
22.1	9.2	12.4	94.8	2.8	0	0.0	0.0	2.5
27.0	8.7	16.4	86.6	9.1	0	0.0	2.6	1.6
18.2	2.1	15.8	88.0	10.5	0	0.0	0.6	0.9
35.4	7.9	14.9	82.7	6.9	0	0.0	6.0	4.5

MeanCommute	Employed	PrivateWork	PublicWork	SelfEmployed	FamilyWork	Unemployment
25.0	943	77.1	18.3	4.6	0	5.4
23.4	753	77.0	16.9	6.1	0	13.3

MeanCommute	Employed	PrivateWork	PublicWork	SelfEmployed	FamilyWork	Unemployment
19.6	1373	64.1	23.6	12.3	0	6.2
25.3	1782	75.7	21.2	3.1	0	10.8
24.8	5037	67.1	27.6	5.3	0	4.2
19.8	1560	79.4	14.7	5.8	0	10.9

Census data: column metadata

Column information is given in `metadata`.

CensusTract	Census.tract.ID	numeric
State	State, DC, or Puerto Rico	string
County	County or county equivalent	string
TotalPop	Total population	numeric
Men	Number of men	numeric
Women	Number of women	numeric
Hispanic	% of population that is Hispanic/Latino	numeric
White	% of population that is white	numeric
Black	% of population that is black	numeric
Native	% of population that is Native American or Native Alaskan	numeric
Asian	% of population that is Asian	numeric
Pacific	% of population that is Native Hawaiian or Pacific Islander	numeric
Citizen	Number of citizens	numeric
Income	Median household income ($)	numeric
IncomeErr	Median household income error ($)	numeric
IncomePerCap	Income per capita ($)	numeric
IncomePerCapErr	Income per capita error ($)	numeric
Poverty	% under poverty level	numeric
ChildPoverty	% of children under poverty level	numeric
Professional	% employed in management, business, science, and arts	numeric
Service	% employed in service jobs	numeric
Office	% employed in sales and office jobs	numeric
Construction	% employed in natural resources, construction, and maintenance	numeric
Production	% employed in production, transportation, and material movement	numeric
Drive	% commuting alone in a car, van, or truck	numeric
Carpool	% carpooling in a car, van, or truck	numeric
Transit	% commuting on public transportation	numeric
Walk	% walking to work	numeric
OtherTransp	% commuting via other means	numeric
WorkAtHome	% working at home	numeric
MeanCommute	Mean commute time (minutes)	numeric
Employed	% employed (16+)	numeric
PrivateWork	% employed in private industry	numeric
PublicWork	% employed in public jobs	numeric
SelfEmployed	% self-employed	numeric
FamilyWork	% in unpaid family work	numeric
Unemployment	% unemployed	numeric

Data wrangling

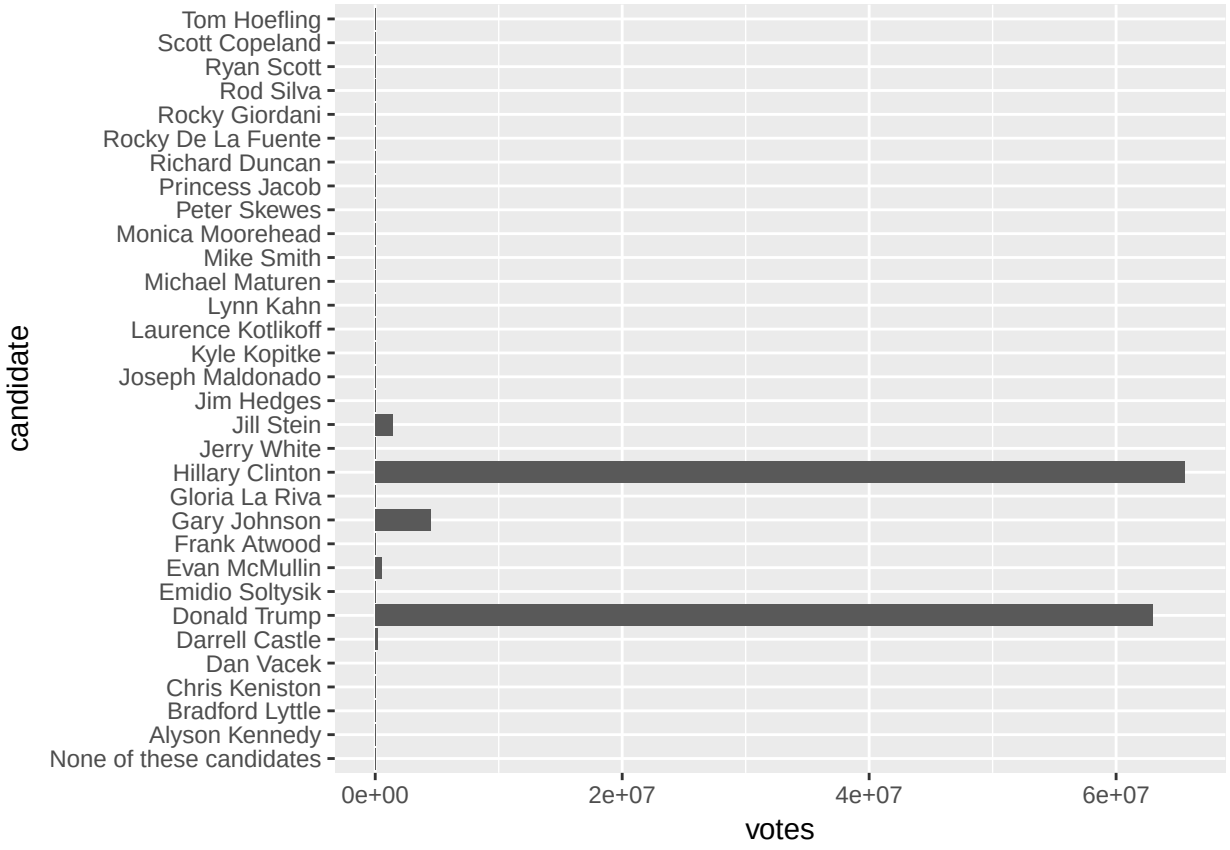
4. Remove summary rows from `election.raw` data: i.e.,
    - Federal-level summary into a `election_federal`.
    - State-level summary into a `election_state`.
    - Only county-level data is to be in `election`.

```
election <- election.raw[!is.na(as.numeric(election.raw$county)), ]

election_federal <- election.raw[election.raw$fips == "US", ]

election_state <- election.raw[election.raw$fips != "US", ]
election_state <- election_state[is.na(election_state$county), ]
# We have duplicated data, AZ, fips 46102 has unknown county, and DC is not
# a state.
election_state <- election_state[election_state$fips != "2000", ]
election_state <- election_state[election_state$state != "DC", ]
election_state <- election_state[election_state$fips != "46102", ]
```

5. How many named presidential candidates were there in the 2016 election? Draw a bar chart of all votes received by each candidate

There are 31 named presidential candidates in the 2016 election.

```
graph.votes <- election_state[, c(3, 5)]
graph.votes <- graph.votes %>% group_by(candidate) %>% summarise_at(vars(votes),
    funs(sum(votes)))
ggplot(graph.votes, aes(x = candidate, y = votes)) + geom_bar(stat = "identity") +
    coord_flip()
```



6. 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).

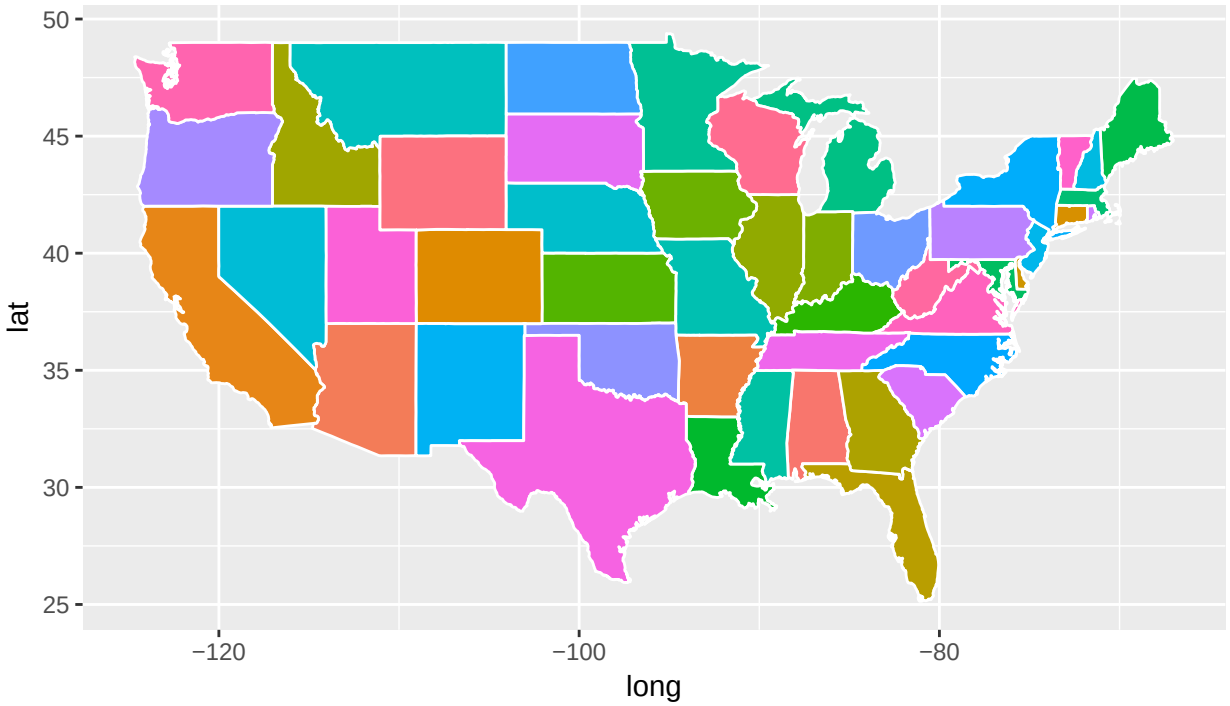
Visualization

```
county_votes <- election %>% group_by(fips) %>% add_tally(votes) %>% group_by(fips,
    candidate) %>% summarise_at(vars(pct = votes), funs(./n))
county_winner <- merge(election, county_votes, by = c("fips", "candidate"))
county_winner <- county_winner %>% group_by(fips) %>% top_n(1, pct)

state_votes <- election_state %>% group_by(fips) %>% add_tally(votes) %>% group_by(fips,
    candidate) %>% summarise_at(vars(pct = votes), funs(./n))
state_winner <- merge(election_state, state_votes, by = c("fips", "candidate"))
state_winner <- state_winner %>% group_by(fips) %>% top_n(1, pct)
```

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.

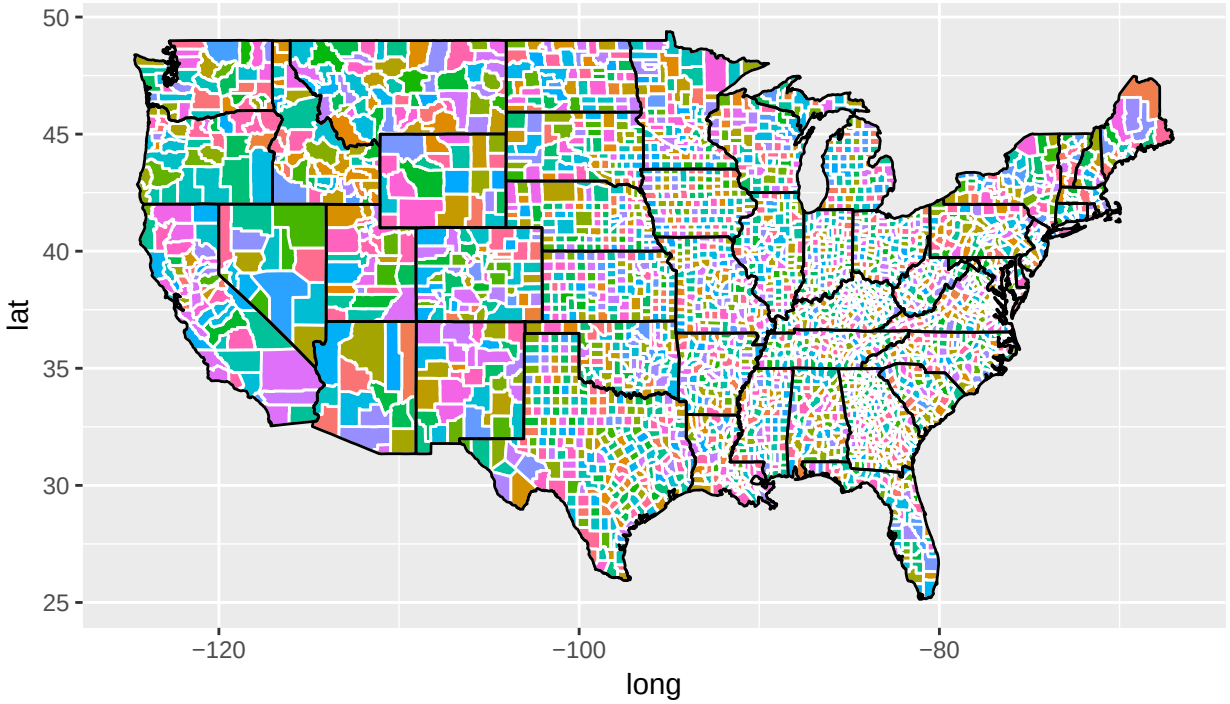
```
states = map_data("state")

ggplot(data = states) + geom_polygon(aes(x = long, y = lat, fill = region, group = group),
    color = "white") + coord_fixed(1.3) + guides(fill = FALSE)
```



The variable `states` contain information to draw white polygons, and fill-colors are determined by `region`.

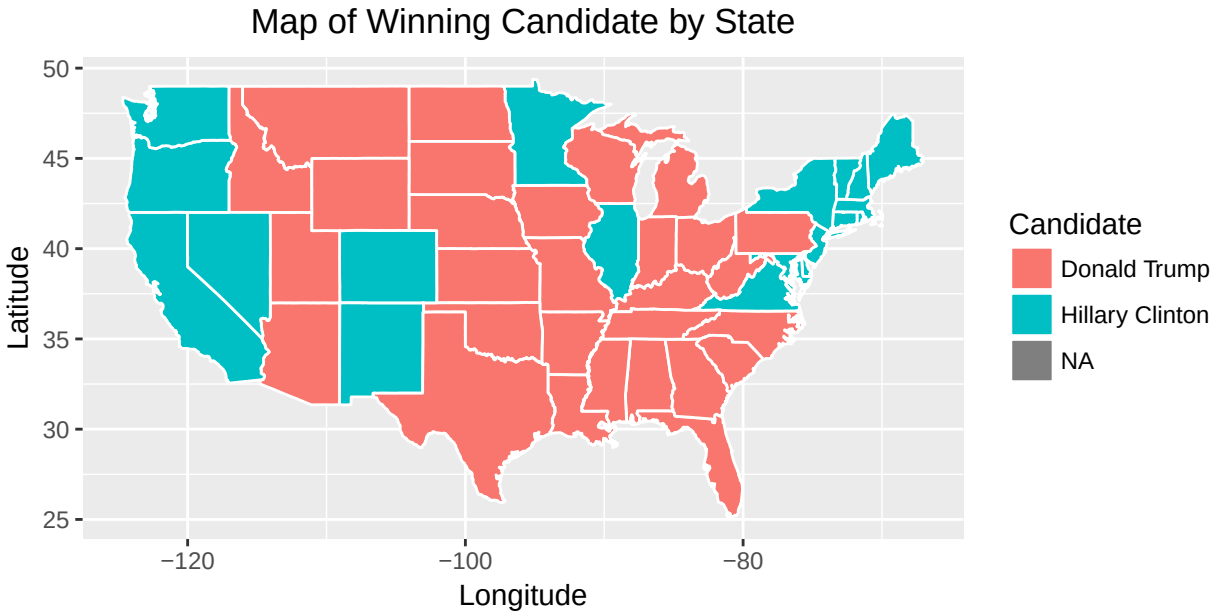
7. Draw county-level map by creating `counties = map_data("county")`. Color by county

```
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) +
    geom_path(aes(x = states$long, y = states$lat, group = group), data = states,
        colour = "black")
```



8. 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; however, they are in different formats: 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.

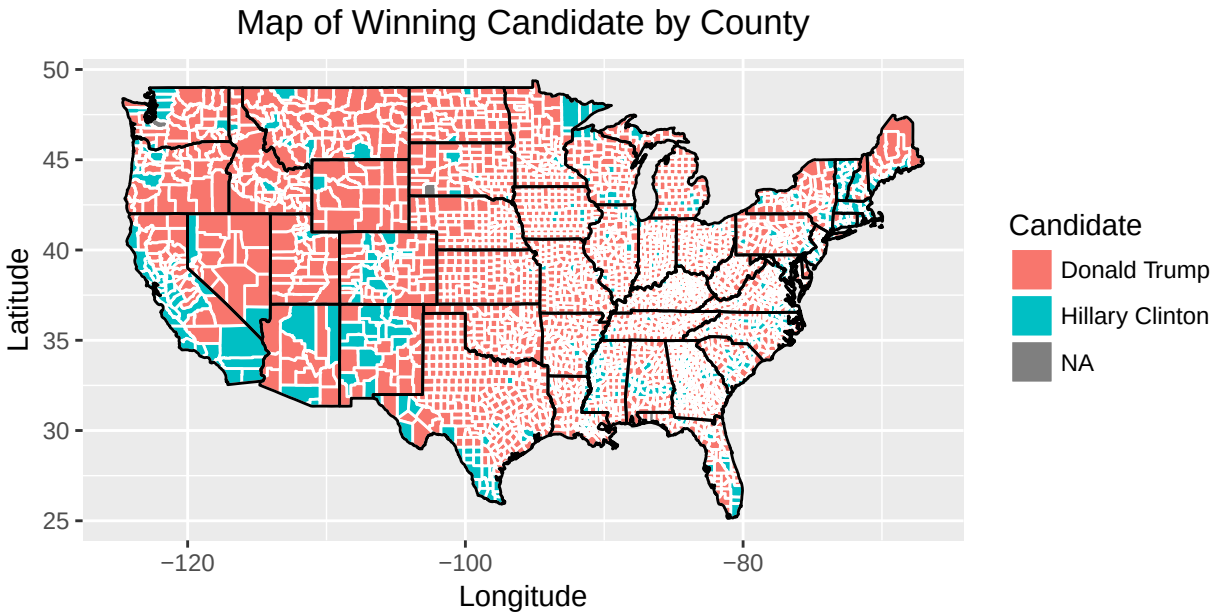
```
states = map_data("state")
fips = state.abb[match(states$region, casefold(state.name))]
states$region <- fips
new <- left_join(states, state_winner, by = c(region = "fips"))

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))
```



9. 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.

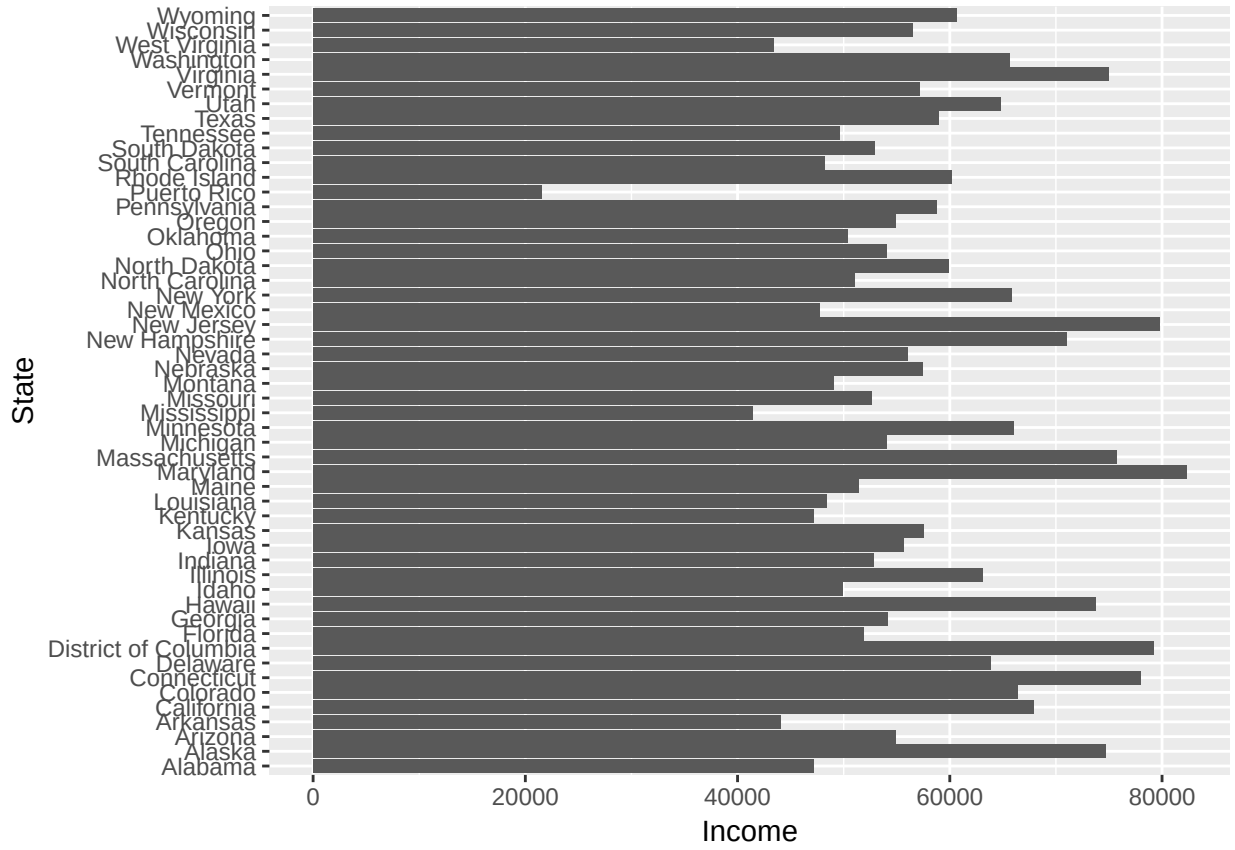
```
county = map_data("county")
county.str <- maps::county.fips
y <- unlist(strsplit(county.str$polyname, ","))

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

```
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")
```



10. 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.

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




```
# From this graph we can see that some states are making way more income
# than others.
```

11. 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 a percentages (meta data seems to be inaccurate), compute `Minority` attribute by combining {Hispanic, Black, Native, Asian, Pacific}, remove {`Walk`, `PublicWork`, `Construction`}.
      Many columns seem to be related, and, if a set that adds up to 100%, one column will be deleted.

```
census.del <- na.omit(census)
# New Col Minority
Minority <- rowSums(census.del[, c(7, 9, 10, 11, 12)])
# names(census.del[,c(6,7,9,10,11,12,23,28,34)]) Cleaning Data
census.del <- census.del[, -c(6, 7, 9, 10, 11, 12, 23, 28, 34)]
# Converting to % and rounding to be consistant with data
# paste(round((census.del$Citizen/census.del$TotalPop)*100,2), '%')
census.del$Men <- round((census.del$Men/census.del$TotalPop) * 100, 1)
census.del$Employed <- round((census.del$Employed/census.del$TotalPop) * 100,
    1)
census.del$Citizen <- round((census.del$Citizen/census.del$TotalPop) * 100,
    1)

census.del <- cbind(census.del, Minority)
```

* _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`.

```
census.subct <- census.del %>% group_by(State, County) %>% add_tally(TotalPop)
census.subct <- cbind(census.subct, Weight = census.subct$TotalPop/census.subct$n)
```

* _County census data, `census.ct`_:
  start with `census.subct`, use `summarize_at()` to compute weighted sum

```
census.ct <- census.subct %>% group_by(State, County) %>% summarize_at(vars(TotalPop:Minority),
    funs(sum(. * Weight)))
census.ct <- as.data.frame(census.ct)
# removing tally and weight columns from census.subct because they are not
# meaningful attributes.
census.subct <- census.subct[, -c(30, 31)]
```

* _Print few rows of `census.ct`_:

```
head(census.ct)
```

```
##     State  County TotalPop      Men    White  Citizen   Income IncomeErr
## 1 Alabama Autauga 6486.404 48.43782 75.78823 73.75219 51696.29  7771.009
## 2 Alabama Baldwin 7698.957 48.84047 83.10262 75.69931 51074.36  8745.050
## 3 Alabama Barbour 3325.195 53.81905 46.23159 76.90508 32959.30  6031.065
## 4 Alabama    Bibb 6380.718 53.41468 74.49989 77.40621 38886.63  5662.358
## 5 Alabama  Blount 7018.573 49.39945 87.85385 73.38814 46237.97  8695.786
## 6 Alabama Bullock 4263.211 53.02620 22.19918 75.45868 33292.69  9000.345
##   IncomePerCap IncomePerCapErr  Poverty ChildPoverty Professional  Service
## 1     24974.50        3433.674 12.91231     18.70758     32.79097 17.17044
## 2     27316.84        3803.718 13.42423     19.48431     32.72994 17.95092
## 3     16824.22        2430.189 26.50563     43.55962     26.12404 16.46343
## 4     18430.99        3073.599 16.60375     27.19708     21.59010 17.95545
## 5     20532.27        2052.055 16.72152     26.85738     28.52930 13.94252
## 6     17579.57        3110.645 24.50260     37.29116     19.55253 14.92420
##     Office Production    Drive   Carpool    Transit OtherTransp WorkAtHome
## 1 24.28243   17.15713 87.50624  8.781235 0.09525905   1.3059687  1.8356531
## 2 27.10439   11.32186 84.59861  8.959078 0.12662092   1.4438000  3.8504774
## 3 23.27878   23.31741 83.33021 11.056609 0.49540324   1.6217251  1.5019456
## 4 17.46731   23.74415 83.43488 13.153641 0.50313661   1.5620952  0.7314679
## 5 23.83692   20.10413 84.85031 11.279222 0.36263213   0.4199411  2.2654133
## 6 20.17051   25.73547 74.77277 14.839127 0.77321596   1.8238247  3.0998783
##   MeanCommute Employed PrivateWork SelfEmployed FamilyWork Unemployment
## 1    26.50016 43.42316    73.73649     5.433254 0.00000000     7.733726
## 2    26.32218 44.05189    81.28266     5.909353 0.36332686     7.589820
## 3    24.51828 31.91015    71.59426     7.149837 0.08977425    17.525557
## 4    28.71439 36.65799    76.74385     6.637936 0.39415148     8.163104
## 5    34.84489 38.44758    81.82671     4.228716 0.35649281     7.699640
## 6    28.63106 36.18313    79.09065     5.273684 0.00000000    17.890026
##   Minority
## 1 22.53687
## 2 15.21426
## 3 51.94382
## 4 24.16597
## 5 10.59474
## 6 76.53587
```

Dimensionality reduction

12. Run PCA for both county & sub-county level data. Save the principal components data frames, call them ct.pc and subct.pc, respectively. What are the most prominent loadings of the first two principal components PC1 and PC2?

For ct.pc, PCA2 is more prominent since it has higher coefficients and for subct.pc, PCA1 is more prominent.

```
# SCALE DATA BEFORE PCA!
(ct.pc <- prcomp(census.ct[, -c(1, 2)], scale. = T, center = T))
```

```
## Standard deviations (1, .., p=26):
##  [1] 2.58355904 1.92479972 1.81666225 1.30616034 1.09793759 1.07424778
##  [7] 1.07055791 0.99209144 0.93456328 0.88405386 0.86653210 0.78066511
## [13] 0.70710923 0.69384016 0.62993578 0.59577074 0.56021594 0.54533400
## [19] 0.46452533 0.42229382 0.36777968 0.31066952 0.24868336 0.21151315
## [25] 0.17220602 0.04920957
##
## Rotation (n x k) = (26 x 26):
##                            PC1          PC2          PC3          PC4
## TotalPop         0.024364637 -0.342737315  0.041606794 -0.040089074
## Men             -0.007529529  0.150615592  0.034587267 -0.202318156
## White           -0.224167784  0.126069831 -0.358245565  0.075981348
## Citizen         -0.006440785  0.163054281 -0.210462832  0.464768012
## Income          -0.318098032 -0.206936996  0.097098565 -0.077012536
## IncomeErr       -0.167873737 -0.255878145  0.205783257  0.003658551
## IncomePerCap    -0.350610145 -0.121097834  0.092926954  0.042724095
## IncomePerCapErr -0.193721318 -0.130664789  0.198685635  0.093887480
## Poverty          0.342070387  0.026735443  0.128279507  0.123752023
## ChildPoverty     0.343253028  0.023850148  0.082309966  0.096003486
## Professional    -0.249663701 -0.012481654  0.242142483  0.258111081
## Service          0.181690835  0.026058869  0.155733672  0.118958236
## Office           0.017407191 -0.283812534 -0.002520126  0.275770873
## Production       0.118326633 -0.009205083 -0.358015309 -0.348598161
## Drive            0.095570526 -0.216809527 -0.339420540  0.260486717
## Carpool          0.076566384  0.048841175  0.028221803 -0.457727241
## Transit         -0.069782922 -0.141803318  0.267846320 -0.025001239
## OtherTransp      0.009674174  0.042620812  0.227387700 -0.221488110
## WorkAtHome      -0.177044971  0.316928895  0.156309555  0.097117188
## MeanCommute      0.060612301 -0.234824421 -0.033516801  0.145249068
## Employed        -0.327662885 -0.031449001 -0.033610528 -0.115852074
## PrivateWork     -0.053964893 -0.343670084 -0.273242216 -0.120977653
## SelfEmployed    -0.100846134  0.392834900  0.077568308  0.091550234
## FamilyWork      -0.050551289  0.264290298  0.055396945  0.047539900
## Unemployment     0.291460759 -0.093682655  0.129884996  0.121220390
## Minority         0.227646705 -0.125029172  0.351427941 -0.072439994
##                         PC5          PC6          PC7          PC8
## TotalPop         0.05687943 -0.13560398 -0.13847004  0.47330548
## Men              0.47150549 -0.43792315 -0.24003279 -0.01277195
## White            0.22551854  0.12916907  0.02476681  0.07945801
## Citizen          0.37124341  0.19816036 -0.00269513 -0.04077813
## Income           0.01581716 -0.08942733 -0.02133061  0.06985384
## IncomeErr        0.11225855 -0.20325078  0.09484741 -0.17883055
## IncomePerCap     0.02630150  0.03004899  0.04614187 -0.08650604
## IncomePerCapErr  0.10136914 -0.10078716  0.14016440 -0.35661391
```

```
## Poverty         -0.09622637  0.07112665  0.07214563 -0.04474680
## ChildPoverty    -0.08236976  0.05546985  0.11366079 -0.07785299
## Professional    -0.11815630 -0.01640508 -0.03003989  0.09250419
## Service          0.39331434 -0.01827375 -0.23008027 -0.23726238
## Office          -0.08169349  0.13131716 -0.22139021  0.34167680
## Production      -0.07441325  0.14551012  0.19497968 -0.16688241
## Drive           -0.18550502 -0.34644818 -0.09396269 -0.12570639
## Carpool          0.19329899 -0.13550662  0.23859159  0.25144416
## Transit          0.16899788  0.36174327  0.35511269 -0.20568541
## OtherTransp      0.13945873  0.49692744 -0.37960077  0.17605734
## WorkAtHome      -0.05454235 -0.01775991  0.12106938  0.21619041
## MeanCommute      0.29484337 -0.06694240  0.54571365  0.32847187
## Employed        -0.19058644  0.14368918 -0.06992823 -0.06564259
## PrivateWork     -0.03635884  0.19926004  0.05647481 -0.06150341
## SelfEmployed    -0.15360352 -0.09416145  0.18077279  0.08565300
## FamilyWork      -0.17533365 -0.06531116  0.18049230  0.19870590
## Unemployment     0.09696692  0.12007830  0.10175587  0.09909515
## Minority        -0.23228878 -0.13219379 -0.01618406 -0.08441643
##                         PC9          PC10         PC11         PC12
## TotalPop        -0.107170512 -0.1302219557  0.087525900  0.239669122
## Men             -0.423094541  0.0916203057  0.074114778 -0.350721913
## White            0.063384643  0.0035819734  0.017775394  0.110623042
## Citizen          0.039604813  0.0988327768 -0.093671779 -0.004785864
## Income          -0.107487511 -0.0269422997 -0.002664957  0.081246774
## IncomeErr        0.108224353  0.3633962912  0.038142572  0.046980382
## IncomePerCap     0.029389199  0.0274984389 -0.046468068  0.065275709
## IncomePerCapErr  0.274588818  0.3937223315 -0.120219631 -0.038109468
## Poverty         -0.032377186 -0.0022224658 -0.053387025  0.047353304
## ChildPoverty    -0.007743176  0.0354083460 -0.064823447  0.025016821
## Professional    -0.160431346 -0.1626478912 -0.199973033  0.278243543
## Service          0.312063476 -0.3238233895  0.332525469  0.192343557
## Office           0.279148066  0.1193679484  0.116246119 -0.654990970
## Production      -0.194511827  0.2112094584 -0.031374731 -0.036401649
## Drive           -0.020429956  0.0650142570 -0.009543079  0.112453762
## Carpool          0.591727371 -0.1093370865 -0.142627720  0.027431406
## Transit         -0.244886405 -0.3060682023  0.262767485 -0.253390463
## OtherTransp     -0.081774047  0.4292660081 -0.145194865  0.169090523
## WorkAtHome      -0.036439732 -0.0506065293 -0.051276536 -0.113487564
## MeanCommute     -0.169892365  0.0589141090 -0.107158409  0.039191988
## Employed         0.067527763 -0.1793409030  0.058172544 -0.020926787
## PrivateWork      0.028873481 -0.0260712133  0.168725990 -0.076812669
## SelfEmployed     0.071691278  0.0056560901 -0.174631118 -0.228703267
## FamilyWork       0.017019703  0.3564623812  0.772393938  0.187384103
## Unemployment    -0.043201978  0.1574086389 -0.060012524  0.131227803
## Minority        -0.065622229  0.0003209897 -0.018787952 -0.116448115
##                        PC13         PC14        PC15        PC16
## TotalPop        -0.429483913 -0.387222804  0.11064619  0.19404578
## Men              0.062825420 -0.184574051 -0.13981310 -0.01481080
## White           -0.020535145 -0.169103151  0.26327165 -0.22268140
## Citizen          0.131008660 -0.089258552 -0.31998126  0.24325306
## Income           0.093828250  0.178191167 -0.17903060 -0.07318382
## IncomeErr        0.021321075  0.105659753  0.33990219 -0.44140340
## IncomePerCap     0.007112754  0.022372083 -0.21925306  0.09894192
## IncomePerCapErr -0.288862960 -0.337108844 -0.09924380  0.26421476
```

```
## Poverty         -0.032888418 -0.261094176  0.03117250 -0.07689606
## ChildPoverty    -0.076796103 -0.221219533  0.04975950 -0.06820849
## Professional     0.277532375 -0.282491109 -0.09831998 -0.04620655
## Service         -0.287678653  0.284797806 -0.02994289  0.01199600
## Office           0.099159343 -0.011909225  0.01376475 -0.05637987
## Production      -0.168609864 -0.019678001 -0.15659876  0.01906680
## Drive            0.034329022  0.100089408  0.10354780  0.10592630
## Carpool          0.276839483 -0.170719628 -0.11757464  0.03798363
## Transit          0.107094492 -0.179384296  0.25111846  0.06317391
## OtherTransp     -0.047735067  0.165025800  0.15456002  0.17535115
## WorkAtHome      -0.483729003  0.075319904 -0.27436132 -0.38649031
## MeanCommute     -0.045538025  0.421571933  0.03828346  0.22351327
## Employed        -0.028841491  0.102810473 -0.17554867  0.09458037
## PrivateWork     -0.250042898 -0.111303509 -0.27095502 -0.12569398
## SelfEmployed    -0.283141185  0.074111186  0.26396010  0.19183251
## FamilyWork       0.144634237 -0.069112134 -0.09471834  0.12505319
## Unemployment     0.063664784  0.008225793 -0.33476562 -0.42605018
## Minority         0.010396133  0.168980142 -0.26286086  0.22749334
##                        PC17         PC18         PC19         PC20
## TotalPop         0.03206089 -0.349685846 -0.046773450 -0.004916628
## Men              0.07064033  0.121304307  0.207484974 -0.060395406
## White           -0.10824735  0.112625848  0.064401444 -0.161112104
## Citizen          0.34372349 -0.353498632 -0.065267742  0.169707359
## Income          -0.05375338 -0.027292310  0.138975789 -0.303033212
## IncomeErr        0.43008282 -0.298200860  0.002689688  0.140596072
## IncomePerCap    -0.05884885 -0.045454499  0.187204455 -0.226519381
## IncomePerCapErr -0.27743339  0.187074247 -0.128160147 -0.081438519
## Poverty          0.27389819  0.129796530  0.111793423 -0.247975889
## ChildPoverty     0.30735761  0.208157776  0.205362450 -0.389181712
## Professional     0.05970672  0.127817122  0.052244377  0.130215361
## Service         -0.02412281  0.002066238  0.089210146 -0.114631096
## Office          -0.04363766 -0.009546510 -0.045517103 -0.194185060
## Production       0.00333451 -0.343036061 -0.142110059 -0.232090921
## Drive           -0.11212530  0.042849542  0.127280094  0.023742205
## Carpool          0.07545738 -0.013740019  0.051236763  0.050004278
## Transit         -0.13629147 -0.119789867 -0.024952015  0.070499253
## OtherTransp      0.03676589  0.155695744  0.104255448  0.060893976
## WorkAtHome       0.08631637  0.113874863 -0.330743488  0.031138853
## MeanCommute      0.09058508  0.267087561 -0.008354436 -0.101532984
## Employed         0.31559630 -0.129293498  0.226904543 -0.338734252
## PrivateWork      0.18516468  0.342152637  0.360790058  0.494203778
## SelfEmployed    -0.06721740 -0.237568708  0.586191770  0.149885888
## FamilyWork       0.01439632  0.039526728  0.046608732 -0.010253865
## Unemployment    -0.46329222 -0.277390912  0.342347870 -0.006189924
## Minority         0.11477470 -0.103409071 -0.059225132  0.181101218
##                        PC21         PC22         PC23          PC24
## TotalPop        -0.01934684  0.092427096 -0.022442506 -0.0301029375
## Men             -0.09642399 -0.075965620 -0.067510422  0.0154061174
## White           -0.04529677 -0.006339989  0.020931582  0.0207760279
## Citizen          0.08817659  0.123863497 -0.040877865  0.0523361307
## Income           0.41362698  0.116241992  0.214398660  0.1822377280
## IncomeErr       -0.01856813 -0.050115665 -0.004186289 -0.0229986402
## IncomePerCap     0.32858033  0.217041668 -0.081674929  0.0656859335
## IncomePerCapErr -0.18673107 -0.041937676  0.041263135 -0.0167243140
```

```
## Poverty         -0.08615645 -0.018987489  0.338452326  0.6687233933
## ChildPoverty     0.26903035  0.142814437 -0.306245433 -0.4832097523
## Professional     0.03898057 -0.615181403  0.048587330 -0.1652131121
## Service          0.10058851 -0.330724486  0.073460343 -0.0377999489
## Office           0.07075410 -0.227571727  0.049935348 -0.0398886217
## Production       0.19363054 -0.513498941  0.082846967 -0.0389667657
## Drive            0.08499220 -0.167551385 -0.574968604  0.3600461950
## Carpool          0.09270912 -0.101655625 -0.219160659  0.1497851336
## Transit          0.07082687 -0.030152549 -0.307876027  0.1729556501
## OtherTransp      0.05824485 -0.105137974 -0.231877038  0.1467519447
## WorkAtHome       0.08754564 -0.049015086 -0.328622623  0.1909331107
## MeanCommute     -0.17973404 -0.104992391  0.065837669 -0.0576688246
## Employed        -0.61940629 -0.047166179 -0.206551666 -0.0063645816
## PrivateWork      0.07876572 -0.001044910  0.093448729 -0.0101953210
## SelfEmployed     0.11298824 -0.109836466  0.113446308  0.0035874516
## FamilyWork       0.01857095 -0.037178091 -0.005582665  0.0005380865
## Unemployment    -0.25411701  0.043228887 -0.093276577 -0.0362393653
## Minority         0.02340751  0.004894302 -0.022823478 -0.0080516952
##                        PC25          PC26
## TotalPop        -0.009445087 -0.0027795161
## Men             -0.057484139  0.0015342674
## White           -0.001089469 -0.7094376931
## Citizen          0.154238292  0.0001480271
## Income           0.592808372 -0.0149033541
## IncomeErr       -0.051007702  0.0009484470
## IncomePerCap    -0.718548890 -0.0056126221
## IncomePerCapErr  0.182031033  0.0046977298
## Poverty         -0.106082954  0.0077225770
## ChildPoverty     0.130816945 -0.0084160937
## Professional    -0.033489526 -0.0026475358
## Service         -0.029699842 -0.0056714790
## Office          -0.051964131 -0.0045403690
## Production      -0.062208452 -0.0034552365
## Drive            0.066869739  0.0069090882
## Carpool          0.030155470 -0.0029136330
## Transit          0.089892838 -0.0003756108
## OtherTransp      0.033896433 -0.0014334897
## WorkAtHome       0.025359438  0.0031713718
## MeanCommute     -0.065829165  0.0060513525
## Employed         0.089724974  0.0055079225
## PrivateWork      0.026922397  0.0121581987
## SelfEmployed     0.061138509  0.0057514032
## FamilyWork       0.005085304  0.0003906839
## Unemployment     0.015968750 -0.0012835997
## Minority        -0.004750162 -0.7041953888
```

```
ct.pc.loads <- ct.pc$rotation
ct.pc <- ct.pc$x
# colSums(abs(ct.pc.loads[,1:2]))

(subct.pc <- prcomp(census.subct[, -c(1, 2, 3)], scale. = T, center = T))
```

```
## Standard deviations (1, .., p=26):
##  [1] 2.71295609 1.77647436 1.40595348 1.21940619 1.17401658 1.08513608
##  [7] 1.03287839 0.97710695 0.94979365 0.91596424 0.90248330 0.83898379
```

```
## [13] 0.83123220 0.75068101 0.74456214 0.69053113 0.66556733 0.61919103
## [19] 0.59537799 0.52053678 0.45800399 0.29947325 0.23165859 0.22814878
## [25] 0.19371283 0.05737688
##
## Rotation (n x k) = (26 x 26):
##                          PC1          PC2         PC3         PC4
## TotalPop        -0.03256684  0.004344250 -0.29697273  0.32154900
## Men             -0.01743589  0.041606903  0.11085459  0.34680478
## White           -0.23959268  0.308393237  0.12631369 -0.11144817
## Citizen         -0.16002274  0.216552445  0.25164043 -0.40051951
## Income          -0.30293368 -0.149905348 -0.13438314  0.14951311
## IncomeErr       -0.19945679 -0.233293926 -0.02355034  0.03646279
## IncomePerCap    -0.31850052 -0.178379135  0.01387095 -0.04395881
## IncomePerCapErr -0.21271747 -0.219215205  0.11680000 -0.12926544
## Poverty          0.30480599 -0.076770996  0.18260401 -0.14252293
## ChildPoverty     0.29804641 -0.047665008  0.15437100 -0.11202684
## Professional    -0.30673797 -0.154124324  0.04199482 -0.05829816
## Service          0.26897064 -0.072202450  0.07483368 -0.06180141
## Office           0.01400718  0.084393296 -0.21590437 -0.31080600
## Production       0.20706801  0.207951388 -0.05370154  0.15823973
## Drive           -0.07825004  0.437134970 -0.12524117  0.07035294
## Carpool          0.16261643  0.042546594 -0.03889044  0.36909356
## Transit          0.05666141 -0.437840435 -0.04718134 -0.18469865
## OtherTransp      0.04496477 -0.163029055  0.18915005 -0.07615588
## WorkAtHome      -0.17345248 -0.119674666  0.32313372  0.10114071
## MeanCommute     -0.01082383 -0.285060827 -0.17255011  0.12586872
## Employed        -0.22140128 -0.039288173 -0.21866181 -0.02076874
## PrivateWork      0.04224415 -0.003005965 -0.41227321 -0.18371252
## SelfEmployed    -0.07054355 -0.043211529  0.43963259  0.30448759
## FamilyWork      -0.01531123  0.027203434  0.20686114  0.17512444
## Unemployment     0.25288245 -0.088888407  0.12043056 -0.14857045
## Minority         0.24118322 -0.305212518 -0.12653399  0.11246354
##                          PC5          PC6          PC7          PC8
## TotalPop        -0.185050210  0.01594234 -0.403309224 -0.046089645
## Men              0.313617009 -0.07726130  0.050606508 -0.187034680
## White            0.168884771 -0.19927493  0.094173606  0.012110229
## Citizen         -0.046066891 -0.15713208  0.035634890 -0.088648579
## Income          -0.070934509  0.10408913  0.099215106 -0.023306523
## IncomeErr       -0.049287340  0.26082887  0.274761027  0.010279823
## IncomePerCap     0.032064596  0.18667037  0.118809468  0.024813330
## IncomePerCapErr  0.057020812  0.33443561  0.222441799  0.069659793
## Poverty          0.021937212  0.17352940 -0.003129824 -0.003729132
## ChildPoverty     0.002064742  0.18580652  0.060001841  0.010794767
## Professional    -0.026931708  0.12148430 -0.095636435 -0.105935146
## Service          0.074500639 -0.05988021 -0.093617329 -0.030666692
## Office          -0.391048565 -0.01543774 -0.204764952  0.310370631
## Production       0.216715912 -0.08345306  0.364868162  0.020818586
## Drive           -0.243286750  0.25492508  0.072439162 -0.092015051
## Carpool          0.130814537  0.16164688  0.100083405  0.143805508
## Transit          0.098533523 -0.42439340  0.041875045 -0.009732675
## OtherTransp      0.383790136  0.15486390 -0.456274588  0.077025703
## WorkAtHome      -0.088798035  0.04489517 -0.026453029  0.112747429
## MeanCommute     -0.230787131 -0.45262009  0.289777746 -0.069900452
## Employed         0.294554059 -0.11282815 -0.251223885  0.110982301
```

```
## PrivateWork      0.363403231  0.14286897  0.226845761  0.266183885
## SelfEmployed    -0.152944030 -0.10077736 -0.057531463 -0.021209227
## FamilyWork      -0.087871002 -0.12680073  0.058769358  0.834568131
## Unemployment    -0.210768740  0.11428242  0.201082753 -0.081793124
## Minority        -0.171921850  0.19775402 -0.086517986 -0.011297545
##                         PC9         PC10         PC11         PC12
## TotalPop        -0.26467126  0.182207586  0.524673858 -0.08485538
## Men             -0.66853180  0.087320965 -0.417961197  0.22515702
## White           -0.08005675 -0.068641315  0.218319672 -0.13283783
## Citizen         -0.10941335  0.091488711 -0.073954997 -0.20894859
## Income          -0.04327402  0.050318229 -0.008342747 -0.05758554
## IncomeErr       -0.10800935  0.002436109 -0.137666858 -0.09334428
## IncomePerCap     0.01548251  0.010583465  0.048784899 -0.02446834
## IncomePerCapErr -0.01772901 -0.050263947  0.018375923  0.00898823
## Poverty         -0.07598696  0.053510930  0.131306439  0.05939300
## ChildPoverty    -0.05138366  0.021068685  0.149031596  0.09037031
## Professional     0.06931105  0.221187109  0.037560790 -0.06163119
## Service          0.11445328  0.039425709 -0.064659958  0.21075751
## Office          -0.36614313 -0.470339120 -0.255666380 -0.03053069
## Production       0.02286509 -0.037864013  0.205598707 -0.04507811
## Drive            0.04442285  0.154953067 -0.063247490  0.14299979
## Carpool          0.16031759 -0.365670979 -0.148036230 -0.54455579
## Transit          0.01762012  0.042189603  0.022371091  0.04262339
## OtherTransp     -0.18930533  0.002330899  0.060718843 -0.40486137
## WorkAtHome      -0.18318175 -0.307724631  0.327481968  0.27874447
## MeanCommute     -0.12514295 -0.045644979  0.081446243 -0.21833419
## Employed         0.27872937 -0.088158237 -0.172602883  0.18986913
## PrivateWork     -0.16566009 -0.176395678  0.286649160  0.26014917
## SelfEmployed     0.16790976 -0.399352124  0.088827596  0.21306441
## FamilyWork      -0.01324451  0.438828394 -0.070274641  0.01056130
## Unemployment    -0.19135158  0.112174197  0.094577005 -0.16890842
## Minority         0.08052057  0.068318581 -0.211566988  0.13914197
##                         PC13         PC14         PC15         PC16
## TotalPop        -0.201646863  0.294240954  0.18528664 -0.025707128
## Men             -0.109843485 -0.013019471  0.13440244  0.084483452
## White           -0.111814211  0.171126495 -0.03688911 -0.081368402
## Citizen         -0.143771032  0.029229439  0.05163410  0.169549095
## Income           0.036427684 -0.039759263 -0.14453756  0.004964813
## IncomeErr        0.089636415  0.378432788 -0.28623896 -0.530702973
## IncomePerCap    -0.020518485  0.066039562  0.13635593  0.158878246
## IncomePerCapErr -0.035209481  0.298979088  0.35386139  0.292762000
## Poverty         -0.104518994  0.001147538  0.20554927 -0.095843956
## ChildPoverty    -0.064733917  0.040625650  0.26143679 -0.074859486
## Professional    -0.127330451 -0.311584185  0.11497806 -0.005573461
## Service         -0.247019777  0.504983447 -0.47633508  0.233114272
## Office           0.066474163  0.061340903  0.13587897 -0.101687067
## Production       0.380305934 -0.048942972  0.17916174 -0.202569128
## Drive            0.244809261  0.080342107 -0.06554553  0.224109955
## Carpool         -0.439939900 -0.060667465  0.02873581  0.099034805
## Transit         -0.082158902  0.057533733  0.18181040 -0.216556770
## OtherTransp      0.514139486  0.029212162 -0.18234584  0.120876941
## WorkAtHome      -0.141321968 -0.400037838 -0.36005669 -0.026736058
## MeanCommute      0.203493720  0.018555343 -0.07798159  0.382891029
## Employed         0.010756069  0.007491659  0.13855997  0.163756628
```

```
## PrivateWork     -0.015069348 -0.043826571 -0.10675819  0.219509021
## SelfEmployed     0.246855085  0.261287309  0.21432971  0.115346516
## FamilyWork      -0.004415835  0.018365362  0.01258643  0.034851494
## Unemployment    -0.012138479 -0.101955109 -0.10033892  0.280204744
## Minority         0.119512679 -0.170340322  0.04192665  0.082327233
##                        PC17         PC18         PC19         PC20
## TotalPop        -0.17465897 -0.161916284  0.029224937  0.043321591
## Men              0.02876603 -0.006358349  0.033690405 -0.012382207
## White            0.20305020  0.147087234  0.051066384 -0.196706562
## Citizen         -0.24441820 -0.493516962 -0.053387833  0.363031202
## Income           0.07645057  0.107733380  0.118447973 -0.262109001
## IncomeErr        0.08648952 -0.379769656  0.064236709  0.088590496
## IncomePerCap    -0.07009196  0.076596450  0.027080240 -0.120231501
## IncomePerCapErr -0.29182393  0.207985113 -0.277273242 -0.055454026
## Poverty          0.30346618 -0.152905669 -0.110985843 -0.084349979
## ChildPoverty     0.39840600 -0.186629773 -0.213189336 -0.202201083
## Professional     0.21114651 -0.025306633  0.121673761  0.159483817
## Service         -0.10763488  0.040067189 -0.081458509 -0.089517302
## Office          -0.02831918  0.068148842  0.005625307 -0.126961735
## Production      -0.42787378 -0.154350316 -0.064218111 -0.148370724
## Drive            0.14699867 -0.046568214 -0.075196159 -0.024277302
## Carpool          0.03134151 -0.075332543  0.010363771  0.053238690
## Transit         -0.09599301  0.130179488  0.151005596  0.014086516
## OtherTransp      0.04793798  0.036809123 -0.049053037  0.017078667
## WorkAtHome      -0.20245720 -0.154194636 -0.253355143 -0.133076473
## MeanCommute      0.30400941 -0.155234795 -0.339425356 -0.002201638
## Employed         0.02409955 -0.549990147  0.076327074 -0.441860369
## PrivateWork      0.16383220 -0.040641032  0.229837053  0.404574047
## SelfEmployed     0.14282575 -0.060659066  0.377776789  0.247355980
## FamilyWork       0.02644212 -0.012566445  0.031042378  0.019141430
## Unemployment    -0.12999130 -0.107982598  0.630616180 -0.358598525
## Minority        -0.20124768 -0.144949222 -0.055171096  0.207677078
##                        PC21         PC22          PC23          PC24
## TotalPop         0.013434902 -0.016886193  0.0043330282  0.0105187966
## Men              0.004021268 -0.013523445 -0.0486853301 -0.0361763967
## White            0.097174816  0.009238881 -0.0104886901 -0.0212779767
## Citizen         -0.296565436 -0.005574182  0.1187907985  0.0546484414
## Income          -0.650290974  0.165715361  0.4766981617 -0.0548942574
## IncomeErr        0.192666050 -0.036740211 -0.0388031296  0.0051832397
## IncomePerCap    -0.321269089  0.010840593 -0.7198287884  0.3182317417
## IncomePerCapErr  0.287634716  0.005155090  0.3045751618 -0.1248433936
## Poverty         -0.022601142  0.724990018 -0.0458530416  0.0855710545
## ChildPoverty    -0.269363862 -0.581266988  0.0691175035 -0.0598326622
## Professional     0.072173530  0.037881074 -0.1928085408 -0.7228139041
## Service         -0.190156668  0.065402212 -0.1625112288 -0.3699151894
## Office          -0.082895117  0.054435342 -0.1109698493 -0.2344348885
## Production      -0.178349990  0.105781126 -0.1417666168 -0.3660906941
## Drive            0.027923047 -0.142658252 -0.1460908975 -0.0630694880
## Carpool         -0.040500402 -0.059620178 -0.0602816554 -0.0574546140
## Transit         -0.070546592 -0.207153484 -0.0636674951 -0.0221750765
## OtherTransp     -0.027901086 -0.062958360 -0.0170462243 -0.0231330363
## WorkAtHome       0.067630575 -0.072333819 -0.0347630255 -0.0002114244
## MeanCommute      0.109587815  0.071794437 -0.0543411123 -0.0404060611
## Employed         0.143020865  0.028756999  0.0546930386  0.0102378273
```

```
## PrivateWork     -0.041393648 -0.007119736  0.0526919696 -0.0008741115
## SelfEmployed    -0.083755948  0.029755398  0.0354596253 -0.0563083827
## FamilyWork      -0.002836646  0.001256445 -0.0009733497 -0.0089880658
## Unemployment     0.203081907 -0.064059106  0.0181354253  0.0131540027
## Minority        -0.090300572 -0.008708551  0.0030892882  0.0509295839
##                         PC25          PC26
## TotalPop         0.006346871  0.0001940407
## Men              0.010613474 -0.0020328713
## White            0.013202475 -0.7079803432
## Citizen          0.048878331 -0.0002286694
## Income           0.081355665 -0.0091807823
## IncomeErr        0.001468988  0.0000861598
## IncomePerCap    -0.115222650  0.0076766691
## IncomePerCapErr  0.051092085 -0.0008037159
## Poverty          0.233863429  0.0036983422
## ChildPoverty    -0.149433814 -0.0025737388
## Professional    -0.060774458 -0.0148904446
## Service         -0.044930546 -0.0110033517
## Office          -0.027238170 -0.0063792592
## Production      -0.043253196 -0.0039045095
## Drive            0.629621717  0.0053602117
## Carpool          0.229307505 -0.0044199084
## Transit          0.617300325  0.0014662726
## OtherTransp      0.126502916 -0.0028040255
## WorkAtHome       0.190804080  0.0006021152
## MeanCommute     -0.077713867  0.0032669735
## Employed         0.015480829 -0.0050797830
## PrivateWork      0.002362305  0.0047283455
## SelfEmployed    -0.011373624  0.0030436941
## FamilyWork       0.001758535  0.0008056995
## Unemployment    -0.021915772 -0.0032355888
## Minority        -0.007533234 -0.7057328385
```

```
subct.pc.loads <- subct.pc$rotation
subct.pc <- subct.pc$x
# colSums(abs(subct.pc.loads[,1:2]))
```

Clustering

13. With `census.ct`, perform hierarchical clustering using a Euclidean distance metric complete linkage to find 10 clusters. Repeat clustering process with the first 5 principal components of `ct.pc`. Compare and contrast clusters containing San Mateo County. Can you hypothesize why this would be the case?

From the tables below, the average Income, IncomeErr, IncomePerCap, IncomePerCapErr, Professional, Transit, and Employed for the cluster with San Mateo are higher than average which casues them to be in the same cluster. The Cluster also has less average Poverty, ChildPoverty, Production, and Unemployment. From this analysis we can hypothesis that people living in these areas are higher class since they have more income and employed as management jobs.

```
# Hierarchical clustering with all principle component
ct.pc.all <- prcomp(census.ct[, c(-1, -2)], scale. = T, center = T)
dist.ct.pc.all <- dist(ct.pc.all$x, method = "euclidean")
hc.c.all <- hclust(dist.ct.pc.all, method = "complete")
partition.c.all <- cutree(hc.c.all, k = 10)
```

```r
# Hierarchical clustering with 5 principle component
ct.pc.5 <- prcomp(census.ct[, c(-1, -2)], scale. = T, center = T)
dist.ct.pc.5 <- dist(ct.pc.5$x[, c(1:5)], method = "euclidean")
hc.c.5 <- hclust(dist.ct.pc.5, method = "complete")
partition.c.5 <- cutree(hc.c.5, k = 10)


pca.san.mateo.all <- (ct.pc.all$x %*% ct.pc.all$rotation) %*% t(ct.pc.all$rotation)
pca.san.mateo.all <- as.data.frame(cbind(pca.san.mateo.all, partition.c.all))
average.comparison.all <- as.matrix(rbind(colMeans(census.ct[which(pca.san.mateo.all$partition.c.all ==
    partition.c.all[which(census.ct[, 2] == "San Mateo")]), ][-c(1, 2)]), colMeans(census.ct[-c(1,
    2)])))
Average <- c("San Mateo Cluster", " Total")
average.comparison.all <- as.data.frame(cbind(Average, average.comparison.all))



pca.san.mateo.5 <- (ct.pc.5$x %*% ct.pc.5$rotation) %*% t(ct.pc.5$rotation)
pca.san.mateo.5 <- as.data.frame(cbind(pca.san.mateo.5, partition.c.5))
average.comparison.5 <- as.matrix(rbind(colMeans(census.ct[which(pca.san.mateo.5$partition.c.5 ==
    partition.c.5[which(census.ct[, 2] == "San Mateo")]), ][-c(1, 2)]), colMeans(census.ct[-c(1,
    2)])))
average.comparison.5 <- as.data.frame(cbind(Average, average.comparison.5))

average.comparison.all
```

```
##             Average         TotalPop              Men            White
## 1 San Mateo Cluster 4352.79546039273 50.0837837249323 79.7170710880501
## 2             Total  4450.7736550832 49.9538302856847 75.4799036821512
##            Citizen           Income        IncomeErr     IncomePerCap
## 1 74.1454707412872 57805.0876074773 8241.53340200291 29161.5831975049
## 2 74.6747678532428 47221.7002616651 7081.75813224634 24000.7488885814
##    IncomePerCapErr          Poverty     ChildPoverty     Professional
## 1 3682.10425553508 12.4696793048259 16.2172768210449 36.1042172086041
## 2 3120.46035313414 17.5706570229159 23.8336932248276 30.8004060227024
##            Service           Office       Production            Drive
## 1 17.3920360089644 21.8605015566203 11.8875634634102 75.9243459222557
## 2  18.472858282297 22.1830928074174 15.8060173865796 79.1462674727629
##            Carpool           Transit      OtherTransp       WorkAtHome
## 1 9.98097962928566   1.4142498754102 1.77443452911037 6.44282587307868
## 2 10.3194871134777 0.989733401044941 1.62516443074591 4.60507443307083
##        MeanCommute         Employed      PrivateWork     SelfEmployed
## 1 22.4251308005902 47.9014982407108 72.1320400006316 9.61840510595394
## 2 23.3134361924172 43.0247851947141  74.183032238375  7.9164536646503
##          FamilyWork     Unemployment         Minority
## 1   0.3347519127609  5.9457426331919 18.1055085230767
## 2 0.287330219645564 8.15706443334943 22.6687728423035
```

```r
average.comparison.5
```

```
##             Average         TotalPop              Men            White
## 1 San Mateo Cluster 5644.24608220864 49.2428069818042 74.0433363891041
## 2             Total  4450.7736550832 49.9538302856847 75.4799036821512
```

```
##            Citizen           Income        IncomeErr     IncomePerCap
## 1  72.473407561732 64318.4826079609 9483.57275694574 30436.1103863885
## 2 74.6747678532428 47221.7002616651 7081.75813224634 24000.7488885814
##    IncomePerCapErr          Poverty     ChildPoverty     Professional
## 1 3883.73686256545 12.1336613787863 15.9876976267022 36.5985854151773
## 2 3120.46035313414 17.5706570229159 23.8336932248276 30.8004060227024
##            Service           Office       Production            Drive
## 1 17.1713670250135 24.3376162617077 11.8520330907688 80.6102881218389
## 2  18.472858282297 22.1830928074174 15.8060173865796 79.1462674727629
##            Carpool           Transit      OtherTransp       WorkAtHome
## 1 9.33538641199219  1.93789176423845 1.53064958383191 4.42219723022139
## 2 10.3194871134777 0.989733401044941 1.62516443074591 4.60507443307083
##        MeanCommute         Employed     PrivateWork     SelfEmployed
## 1 25.6847835208181 47.8783922884466 78.818074412551 5.86966568097656
## 2 23.3134361924172 43.0247851947141 74.183032238375  7.9164536646503
##          FamilyWork     Unemployment         Minority
## 1 0.149931514966753 6.99596586038594 23.7330168928279
## 2 0.287330219645564 8.15706443334943 22.6687728423035
```

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.

```
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

## saves meta information to attributes
attr(election.cl, "location") = election.cl %>% dplyr::select(c(county, fips, state, votes, pct))
election.cl = election.cl %>% dplyr::select(-c(county, fips, state, votes, pct))
```

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, ]
```

Using the following code, define 10 cross-validation folds:

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

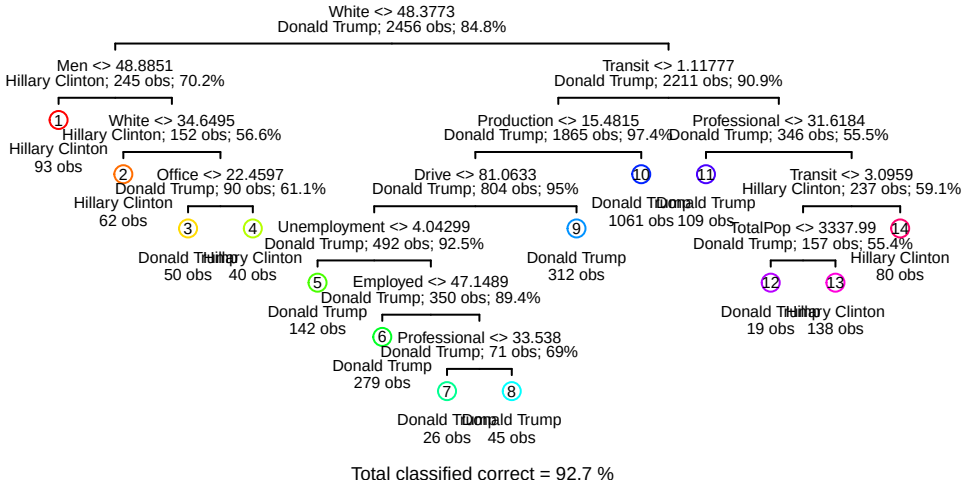
Using the following error rate function:

```
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", "knn", "lda")
```

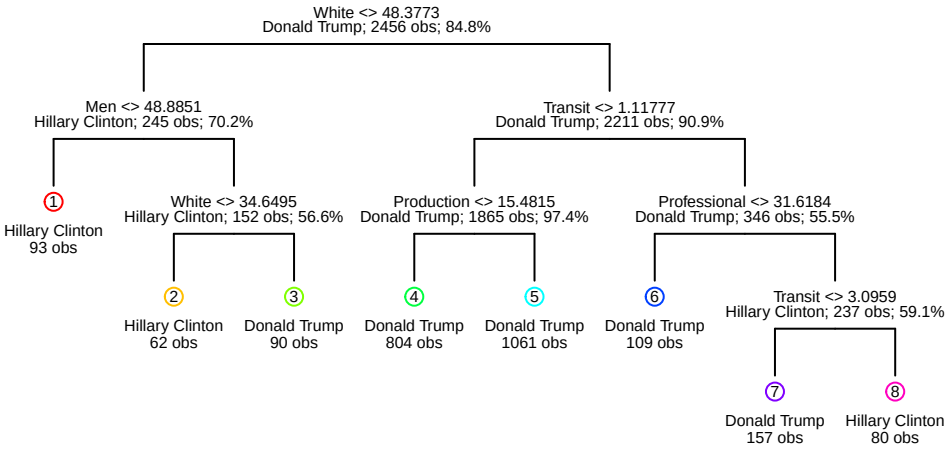
Classification: native attributes

13. Decision tree: train a decision tree by `cv.tree()`. Prune tree to minimize misclassification. 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.

```
trn.cl.tree <- tree(candidate ~ ., data = trn.cl, method = "class")
cv.trn.cl.tree <- cv.tree(trn.cl.tree, rand = folds, method = "misclass", K = nfold)
draw.tree(trn.cl.tree, nodeinfo = T, cex = 0.5)
```




```
best.size <- min(cv.trn.cl.tree$size[cv.trn.cl.tree$dev == min(cv.trn.cl.tree$dev)])
trn.cl.tree.pruned <- prune.tree(trn.cl.tree, best = best.size)
draw.tree(trn.cl.tree.pruned, nodeinfo = T, cex = 0.5)
```

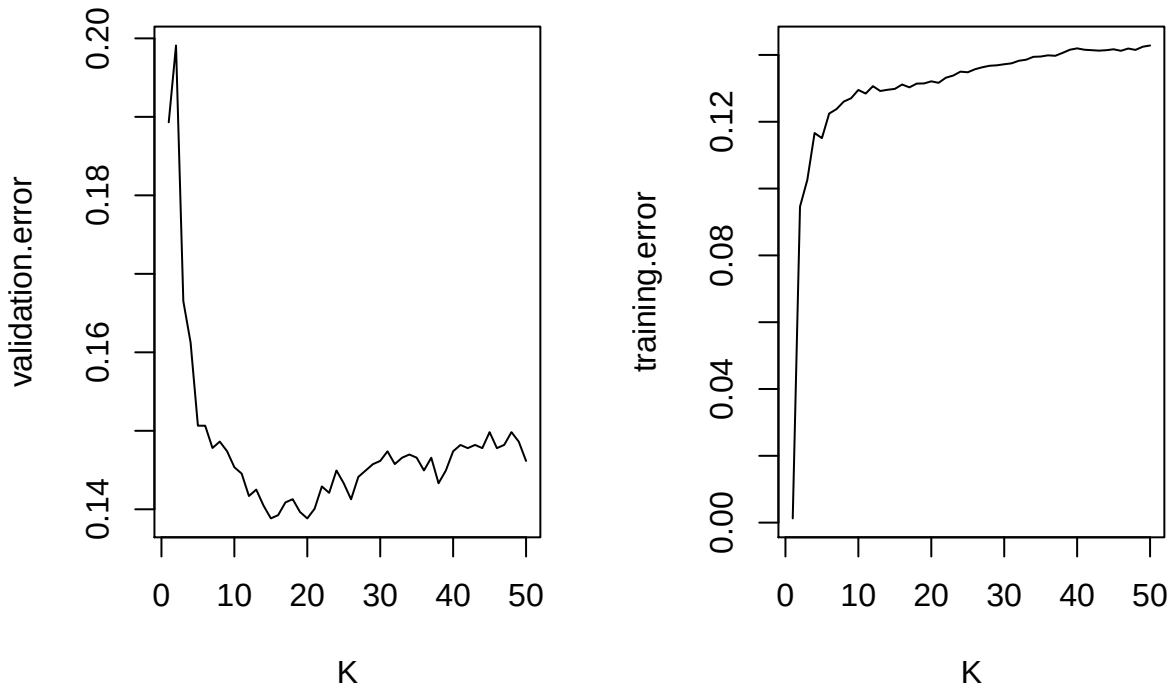



```
Predict.tree.train <- predict(trn.cl.tree.pruned, trn.cl, type = "class")
Predict.tree.test <- predict(trn.cl.tree.pruned, tst.cl, type = "class")

records[1, 1] <- calc_error_rate(Predict.tree.train, trn.cl$candidate)
records[1, 2] <- calc_error_rate(Predict.tree.test, tst.cl$candidate)
```

14. K-nearest neighbor: train a KNN model for classification. Use cross-validation to determine the best number of neighbors, and plot number of neighbors vs. resulting training and validation errors. Compute test error and save to `records`.

```
trn.x <- trn.cl %>% dplyr::select(-candidate)
trn.y <- trn.cl$candidate
kvec <- 1:50

set.seed(2)

do.chunk <- function(chunkid, folddef, Xdat, Ydat, k) {
    train = (folddef != chunkid)

    X.Train = Xdat[train, ]
    X.Validation = Xdat[!train, ]
    Y.Train = Ydat[train]
    Y.Validation = Ydat[!train]

    Predict.Y.Train = knn(train = X.Train, test = X.Train, cl = Y.Train, k = k)
    Predict.Y.Validation = knn(train = X.Train, test = X.Validation, cl = Y.Train,
        k = k)
```

```
    data <- data.frame(train.error = calc_error_rate(Predict.Y.Train, Y.Train),
        val.error = calc_error_rate(Predict.Y.Validation, Y.Validation))
}

validation.error = NULL
training.error = NULL
for (i in kvec) {
    k <- plyr::ldply(1:nfold, do.chunk, folddef = folds, Xdat = trn.x, Ydat = trn.y,
        k = i)
    validation.error <- c(validation.error, mean(k$val.error))
    training.error <- c(training.error, mean(k$train.error))
}

par(mfrow = c(1, 2))
plot(validation.error, xlab = "K", type = "l")
plot(training.error, xlab = "K", type = "l")
```




```
# The graph shows that the best k is within the range of 1:50 because the
# validation error starts to increase as k increases.

best.kfold = max(kvec[validation.error == min(validation.error)])

knn.Predict.Train = knn(train = trn.x, test = trn.x, cl = trn.y, k = best.kfold)
knn.Predict.Test = knn(train = trn.x, test = tst.cl %>% dplyr::select(-candidate),
    cl = trn.y, k = best.kfold)
```

```
knn.training.error <- calc_error_rate(knn.Predict.Train, trn.cl$candidate)
knn.test.error <- calc_error_rate(knn.Predict.Test, tst.cl$candidate)
records[2, 1] <- knn.training.error
records[2, 2] <- knn.test.error
```

Classification: principal components

Instead of using the native attributes, we can use principal components in order to train our classification models. After this section, a comparison will be made between classification model performance between using native attributes and principal components.

```
pca.records = matrix(NA, nrow = 3, ncol = 2)
colnames(pca.records) = c("train.error", "test.error")
rownames(pca.records) = c("tree", "knn", "lda")
```

15. Compute principal components from the independent variables in training data. Then, determine the number of minimum number of PCs needed to capture 90% of the variance. Plot proportion of variance explained. We need at least 14 principle component to explain at least 90% of the variance.

```
(trn.cl.pc <- prcomp(trn.cl[-1], center = T, scale. = T))
```

```
## Standard deviations (1, .., p=26):
##  [1] 2.51369425 1.99846712 1.81593090 1.25662740 1.14948913 1.09663077
##  [7] 1.03986079 0.99282121 0.93322545 0.89250002 0.85763435 0.78094514
## [13] 0.72331853 0.69615730 0.65332252 0.60985334 0.57684635 0.53946319
## [19] 0.47580886 0.42226257 0.38079565 0.31382084 0.23303978 0.19648699
## [25] 0.17947520 0.05729932
##
## Rotation (n x k) = (26 x 26):
##                            PC1          PC2         PC3          PC4
## TotalPop         0.010774489 -0.327035056  0.04552977 -0.012372167
## Men              0.020040441  0.121529912 -0.08452903 -0.008791092
## White           -0.170499590  0.245483564  0.33736787  0.142893412
## Citizen         -0.013777253  0.233057137  0.19141662  0.512031179
## Income          -0.321516834 -0.230006650 -0.01011250 -0.064762609
## IncomeErr       -0.153585952 -0.298740075 -0.14851183  0.085503207
## IncomePerCap    -0.358876784 -0.135167661 -0.03527283  0.050306122
## IncomePerCapErr -0.180734056 -0.165573787 -0.17415619  0.155407371
## Poverty          0.340035371 -0.001121678 -0.14593127  0.094673415
## ChildPoverty     0.342204713 -0.001795486 -0.10154788  0.080890693
## Professional    -0.291131478 -0.047295189 -0.16919441  0.173892367
## Service          0.174999543 -0.012170035 -0.19706061  0.285224852
## Office          -0.025922369 -0.247580925  0.09467807  0.280938211
## Production       0.175106429  0.050363427  0.32130417 -0.347162736
## Drive            0.105014258 -0.134307848  0.37791639  0.157814230
## Carpool          0.136574208  0.012798824 -0.10466025 -0.390060255
## Transit         -0.078364822 -0.174490022 -0.21101186  0.001970866
## OtherTransp      0.001132933 -0.013589628 -0.20732429 -0.024694963
## WorkAtHome      -0.183079975  0.269913866 -0.22998837  0.034450816
## MeanCommute      0.039364469 -0.206701607  0.08780448  0.162808080
## Employed        -0.326336813 -0.023469314  0.06413715 -0.212678809
## PrivateWork     -0.018422957 -0.278075599  0.32578730 -0.160249894
## SelfEmployed    -0.117424863  0.362073345 -0.17214242 -0.034677360
## FamilyWork      -0.050513326  0.242749679 -0.12686555 -0.030425910
```

```
## Unemployment     0.279884307 -0.134363296 -0.07965447  0.227624291
## Minority         0.173081946 -0.243468890 -0.33333496 -0.149093591
##                         PC5           PC6          PC7           PC8
## TotalPop         0.006743224 -0.1629863829  0.259301455  0.4102316127
## Men              0.132840968 -0.6890876513  0.270615190 -0.2164833237
## White           -0.231375843 -0.0255930994  0.087844770  0.0207447972
## Citizen         -0.190897692 -0.0416446280  0.028839976 -0.1038220275
## Income           0.021005492 -0.1443582769  0.003152886  0.0502677242
## IncomeErr        0.054621528 -0.1263943019 -0.171181551 -0.0912396662
## IncomePerCap    -0.038968445  0.0075588746 -0.072320493 -0.0707750990
## IncomePerCapErr  0.021404538  0.0004580678 -0.228351252 -0.2457171930
## Poverty         -0.032545926  0.1769059851 -0.103659938  0.0057507922
## ChildPoverty    -0.034651686  0.1591058088 -0.156214577 -0.0015373409
## Professional     0.086369911  0.0606798081 -0.078979004  0.1193214267
## Service         -0.002543965 -0.0760242452  0.321543816 -0.2890549192
## Office           0.049032286  0.2433372639  0.257895365  0.3421388390
## Production      -0.158055076  0.0199124883 -0.178642264 -0.1731503112
## Drive            0.450041339 -0.0276652949 -0.114525073  0.0005507058
## Carpool         -0.233402131 -0.2275107611  0.114732198  0.2564265572
## Transit         -0.483271834  0.0779636863 -0.126368450 -0.3051729530
## OtherTransp     -0.241010540  0.2516389177  0.546168005 -0.0122088034
## WorkAtHome      -0.052033227 -0.0048949404 -0.051226219  0.2305044446
## MeanCommute     -0.431659298 -0.3624112375 -0.304832310  0.2944293593
## Employed         0.014287940  0.2136529816  0.091569569 -0.0892032144
## PrivateWork     -0.182420101  0.1564336621  0.071112275 -0.1152307574
## SelfEmployed     0.018358979  0.0330429819 -0.161389094  0.1453680938
## FamilyWork       0.043358899  0.0911336653 -0.160718062  0.3079858497
## Unemployment    -0.151036484 -0.0124556492 -0.103460430  0.1254439817
## Minority         0.231557469  0.0261967557 -0.094747070 -0.0267982363
##                         PC9          PC10         PC11          PC12
## TotalPop        -0.175683328  0.080688486  0.085430165  0.236262425
## Men             -0.127443502  0.292965988 -0.115397666 -0.263755141
## White            0.074588016 -0.021043802  0.001801871  0.105673885
## Citizen          0.123320640 -0.007637376  0.005876476  0.007172115
## Income          -0.069193431  0.022070399  0.028584272  0.094213165
## IncomeErr        0.304976783  0.138187532 -0.156174191  0.020108255
## IncomePerCap     0.084957073 -0.014049435  0.018377980  0.095407502
## IncomePerCapErr  0.536750341  0.082901934 -0.106756431 -0.034222364
## Poverty         -0.007785932  0.060214726  0.109612894  0.051194019
## ChildPoverty     0.028471922  0.054431019  0.086923345  0.033706379
## Professional    -0.162729447  0.002731809  0.363190732  0.204055893
## Service         -0.043644891 -0.393111063 -0.242070567  0.361142339
## Office           0.051502698 -0.118387884 -0.316683096 -0.613535458
## Production       0.044880532  0.259847751 -0.020379255 -0.064947901
## Drive            0.051947959  0.098398545  0.034367128  0.111139413
## Carpool          0.452033388 -0.431671209 -0.046598183  0.077392713
## Transit         -0.437519761 -0.041895120 -0.155637659 -0.170753664
## OtherTransp      0.267108796  0.558170605  0.181257138  0.086134035
## WorkAtHome      -0.075152976  0.039596025  0.088550740 -0.156225824
## MeanCommute     -0.005669280  0.076926110  0.066880798  0.016511728
## Employed        -0.071323074 -0.124844804 -0.027730058  0.094064033
## PrivateWork     -0.006834349  0.033149217 -0.163753750  0.018107224
## SelfEmployed     0.105703941 -0.046822837  0.077720876 -0.229938616
## FamilyWork      -0.093357415  0.306794805 -0.722443397  0.356923030
```

```
## Unemployment    -0.006315345  0.085492990  0.057505193  0.076602404
## Minority        -0.067343292  0.026650876 -0.005025717 -0.111672896
##                        PC13         PC14        PC15         PC16
## TotalPop        -0.20057735 -0.505052404  0.25758167  0.185582555
## Men             -0.21704631 -0.059527248 -0.13840790 -0.038285539
## White           -0.20482740 -0.046567163  0.17197039 -0.251948497
## Citizen         -0.03632494  0.051280487 -0.26146584  0.304567019
## Income           0.11986165  0.089455387 -0.15607134 -0.091363623
## IncomeErr       -0.02924771  0.163806856  0.30165928 -0.563603941
## IncomePerCap     0.04205523 -0.054356461 -0.18873923  0.117589560
## IncomePerCapErr -0.16727988 -0.408219619 -0.02337765  0.287450246
## Poverty         -0.28185778 -0.095561339  0.06594630 -0.121760959
## ChildPoverty    -0.21081631 -0.119512019  0.08374926 -0.075995523
## Professional    -0.33490964  0.132527190 -0.21750096  0.014902228
## Service          0.34778039 -0.171164735  0.11490270 -0.051695664
## Office          -0.05931614  0.060313849 -0.03908834 -0.002152557
## Production       0.10844032 -0.171921796 -0.09901176  0.042371670
## Drive            0.10157941  0.081562320  0.10871955  0.103797210
## Carpool         -0.22756481  0.150208881 -0.17525449  0.056940423
## Transit         -0.20934572  0.059094336  0.17091850  0.117683977
## OtherTransp      0.17396796  0.186894796  0.08174302  0.086360069
## WorkAtHome       0.28243646 -0.446850360 -0.18636448 -0.302140637
## MeanCommute      0.36700766  0.153317008  0.18230098  0.178990601
## Employed         0.08580911 -0.007203118 -0.03901678  0.013850621
## PrivateWork      0.03003766 -0.309117367 -0.23014598 -0.154823106
## SelfEmployed     0.17448091 -0.178249537  0.31722673  0.092715573
## FamilyWork      -0.09102745  0.086348355 -0.08579420  0.100944999
## Unemployment     0.10057083 -0.037397544 -0.49346553 -0.308365356
## Minority         0.21062430  0.040318181 -0.17158700  0.251832177
##                        PC17         PC18         PC19         PC20
## TotalPop         0.12213372 -0.327471188 -0.023038039 -0.001405249
## Men             -0.17076243  0.072397575  0.222576997 -0.032679106
## White            0.11480923  0.161334382  0.044770737 -0.115623115
## Citizen         -0.23094477 -0.521161839 -0.072601271  0.156874933
## Income          -0.03923475  0.077671880  0.195515551 -0.266764264
## IncomeErr       -0.09661275 -0.430865663 -0.080092427  0.134283125
## IncomePerCap    -0.05300692  0.008783455  0.244671393 -0.160279534
## IncomePerCapErr  0.15880824  0.244375024 -0.098387284 -0.112831103
## Poverty         -0.30249624  0.027124375  0.090280625 -0.164745143
## ChildPoverty    -0.40974741  0.098335113  0.256687245 -0.199473426
## Professional    -0.07614121  0.042256369  0.006620492  0.127742142
## Service         -0.03483643  0.041651219  0.070562352 -0.111961638
## Office          -0.03709821  0.019899725  0.047965842 -0.207770493
## Production       0.07555571 -0.315706826 -0.054169478 -0.355095597
## Drive            0.03897151  0.096178206  0.152346091 -0.004138134
## Carpool         -0.04217733 -0.049666413  0.043912707  0.018405336
## Transit          0.19457317 -0.043523538  0.027345013  0.020916568
## OtherTransp      0.00878962  0.076349030  0.044897506  0.020602601
## WorkAtHome      -0.18271862  0.035470083 -0.352671897 -0.069606412
## MeanCommute     -0.26478702  0.219067525 -0.002279662 -0.047015097
## Employed        -0.30788810 -0.248529352  0.273048208 -0.313047868
## PrivateWork     -0.27448369  0.191442281  0.155439804  0.605573963
## SelfEmployed     0.11939342 -0.140990786  0.614024096  0.272577401
## FamilyWork      -0.02895638  0.035248753  0.025630302  0.018250111
```

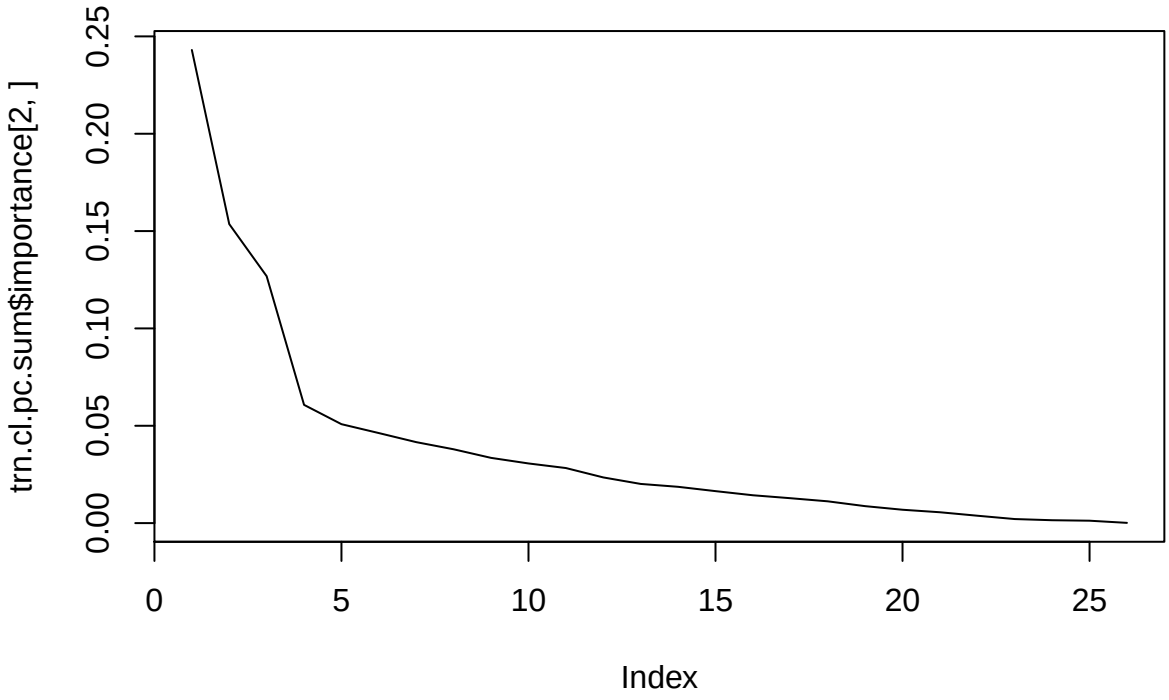
```
## Unemployment     0.48255449 -0.093465975  0.336365993 -0.052452098
## Minority        -0.11912340 -0.165688395 -0.048434793  0.133849758
##                        PC21         PC22          PC23          PC24
## TotalPop        -0.008216474  0.092100414 -0.0148141415 -0.0198403137
## Men              0.116478631 -0.062968447 -0.0330176797  0.0381806244
## White            0.020189849  0.003483895 -0.0136440802  0.0006274353
## Citizen         -0.089830808  0.122929870  0.0259277641  0.0157276932
## Income          -0.410415288  0.095002844  0.2480485399 -0.1567789506
## IncomeErr       -0.023184101 -0.051327425 -0.0469398468  0.0151121075
## IncomePerCap    -0.326150053  0.242687233  0.1125312973  0.2538496733
## IncomePerCapErr  0.200911582 -0.053756586 -0.0239201068 -0.0855685961
## Poverty          0.121907096 -0.048171131  0.6694761310  0.2987124652
## ChildPoverty    -0.278099584  0.184856368 -0.5293889087 -0.2362429892
## Professional    -0.008398882 -0.626989645 -0.1130541589 -0.0997050594
## Service         -0.103912518 -0.330919727  0.0090930379 -0.0371832977
## Office          -0.053575298 -0.215301894  0.0093429698 -0.0433260991
## Production      -0.211836980 -0.482539022  0.0288980267 -0.0796641028
## Drive           -0.047259885 -0.125253597 -0.2388537904  0.6147696157
## Carpool         -0.081647887 -0.096927858 -0.1182988994  0.2849295963
## Transit         -0.079783786 -0.024022304 -0.1743515823  0.3583070523
## OtherTransp     -0.034608352 -0.047514552 -0.0677551915  0.1240711437
## WorkAtHome      -0.065707037 -0.018954425 -0.1715441857  0.3574771039
## MeanCommute      0.217019850 -0.107756098  0.0175587535 -0.0565275891
## Employed         0.614603647  0.033925041 -0.1583890893  0.0395756095
## PrivateWork     -0.018825635 -0.075079987  0.0479280148 -0.0145957613
## SelfEmployed    -0.082568214 -0.159449906  0.0925746718 -0.0290630148
## FamilyWork      -0.009800134 -0.044282164 -0.0008780233  0.0042055023
## Unemployment     0.241789898  0.080840081 -0.0830354884  0.0033583609
## Minority         0.003184753  0.008344023  0.0026834293 -0.0142057534
##                        PC25          PC26
## TotalPop        -0.011112122 -0.0018591007
## Men             -0.060795999  0.0021961430
## White           -0.016252491 -0.7101342675
## Citizen          0.193745236 -0.0006528128
## Income           0.603761495 -0.0224435172
## IncomeErr       -0.054018653  0.0052339218
## IncomePerCap    -0.659924959  0.0018380246
## IncomePerCapErr  0.169131299  0.0035276773
## Poverty          0.051760139 -0.0094871933
## ChildPoverty     0.017852109  0.0012510195
## Professional    -0.072996638 -0.0070387133
## Service         -0.049354787 -0.0074254978
## Office          -0.066519017 -0.0053324797
## Production      -0.096833898 -0.0018464548
## Drive            0.185264907 -0.0087490427
## Carpool          0.089394919 -0.0064808398
## Transit          0.162737826 -0.0081800274
## OtherTransp      0.045389042 -0.0028592961
## WorkAtHome       0.101786744 -0.0027900984
## MeanCommute     -0.086650231  0.0043587096
## Employed         0.085405627  0.0064826786
## PrivateWork      0.035980202  0.0090921668
## SelfEmployed     0.069527377  0.0005445098
## FamilyWork       0.005709311 -0.0008951011
```

```
## Unemployment     0.004116618  0.0018400621
## Minority        -0.007968846 -0.7032609726
```

```
trn.cl.pc.sum <- summary(trn.cl.pc)
trn.cl.pc.sum$importance[3, ] >= 0.9
```

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

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



16. Create a new training data by taking class labels and principal components. Call this variable `tr.pca`. Create the test data based on principal component loadings: i.e., transforming independent variables in test data to principal components space. Call this variable `test.pca`.

```
# PCA training set
tr.pca <- as.data.frame(cbind(V = trn.cl$candidate, trn.cl.pc$x[, 1:14]))
# levels(trn.cl$candidate)[c(7,13)]
tr.pca$V[tr.pca$V == 7] <- 0  #'Donald Trump'
tr.pca$V[tr.pca$V == 13] <- 1  #'Hillary Clinton'
tr.pca$V <- as.factor(tr.pca$V)

# PCA test set test.pca <- predict(trn.cl.pc, newdata = tst.cl) #Does the
# same thing below:
```

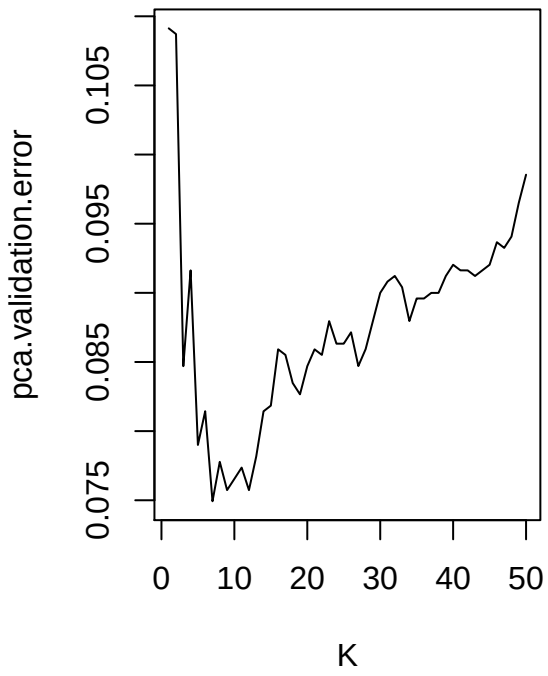
```
test.pca <- scale(tst.cl[-1], center = attr(scale(trn.cl[-1]), "scaled:center"),
    scale = attr(scale(trn.cl[-1]), "scaled:scale"))
test.pca <- test.pca %*% trn.cl.pc$rotation
#-----------------------------------------------------------
test.pca <- predict(trn.cl.pc, newdata = tst.cl)
test.pca <- as.data.frame(cbind(V = tst.cl$candidate, test.pca[, 1:14]))
# levels(tst.cl$candidate)[c(7,13)]
test.pca$V[test.pca$V == 7] <- 0  #'Donald Trump'
test.pca$V[test.pca$V == 13] <- 1  #'Hillary Clinton'
test.pca$V <- as.factor(test.pca$V)
```

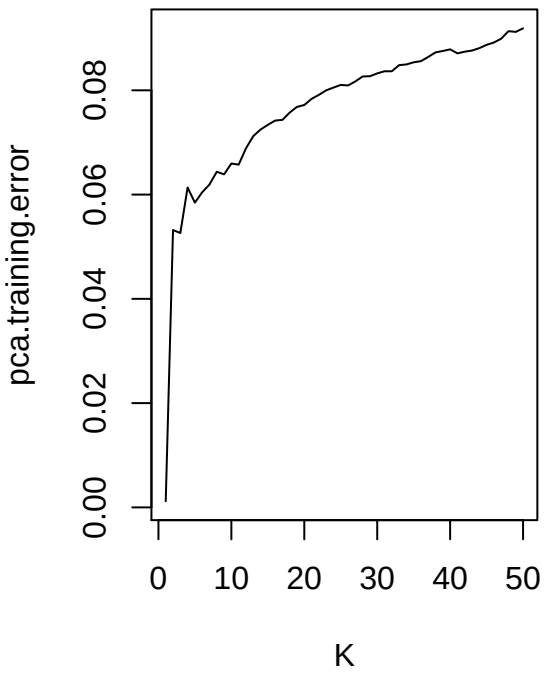
17. Decision tree: repeat training of decision tree models using principal components as independent variables. Record resulting errors.

```
tr.pca.tree <- tree(V ~ ., data = tr.pca, method = "class")
cv.tr.pca.tree <- cv.tree(tr.pca.tree, rand = folds, method = "misclass", K = nfold)
pca.best.size <- min(cv.tr.pca.tree$size[cv.tr.pca.tree$dev == min(cv.tr.pca.tree$dev)])
tr.pca.tree.pruned <- prune.tree(tr.pca.tree, best = pca.best.size)
Predict.pca.tree.train <- predict(tr.pca.tree.pruned, tr.pca, type = "class")
Predict.pca.tree.test <- predict(tr.pca.tree.pruned, test.pca, type = "class")

pca.records[1, 1] <- calc_error_rate(Predict.pca.tree.train, tr.pca$V)
pca.records[1, 2] <- calc_error_rate(Predict.pca.tree.test, test.pca$V)
```

18. K-nearest neighbor: repeat training of KNN classifier using principal components as independent variables. Record resulting errors.

```
pca.trn.x <- tr.pca %>% dplyr::select(-V)
pca.trn.y <- tr.pca$V

set.seed(2)

pca.validation.error = NULL
pca.training.error = NULL
for (i in kvec) {
    pca.k <- plyr::ldply(1:nfold, do.chunk, folddef = folds, Xdat = pca.trn.x,
        Ydat = pca.trn.y, k = i)
    pca.validation.error <- c(pca.validation.error, mean(pca.k$val.error))
    pca.training.error <- c(pca.training.error, mean(pca.k$train.error))
}

par(mfrow = c(1, 2))
plot(pca.validation.error, xlab = "K", type = "l")
plot(pca.training.error, xlab = "K", type = "l")
```






```
# The graph shows that the best k is within the range of 1:50 because the
# validation error is increasing as k increases.

pca.best.kfold = min(kvec[pca.validation.error == min(pca.validation.error)])


knn.Predict.Train.pca = knn(train = pca.trn.x, test = pca.trn.x, cl = pca.trn.y,
    k = pca.best.kfold)

knn.Predict.Test.pca = knn(train = pca.trn.x, test = test.pca %>% dplyr::select(-V),
    cl = pca.trn.y, k = pca.best.kfold)


pca.records[2, 1] <- calc_error_rate(knn.Predict.Train.pca, tr.pca$V)
pca.records[2, 2] <- calc_error_rate(knn.Predict.Test.pca, test.pca$V)
```

Interpretation & Discussion

19. This is an open question. Interpret and discuss any insights gained 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)

The dataset: 1. There was a duplicate observation for the state AZ and SD for state level so it was removed. 2. Although DC is specified to be part of the state, DC was removed since it does not belong in our data

which is supposed to be a state level.

How these errors were found is fromt he codes below:

```
election_state <- election.raw[election.raw$fips != "US", ]
election_state <- election_state[is.na(election_state), ]
state_votes <- election_state %>% group_by(fips) %>% add_tally(votes) %>% group_by(fips,
    candidate) %>% summarise_at(vars(pct = votes), funs(./n))
state_winner <- merge(election_state, state_votes, by = c("fips", "candidate"))
state_winner <- state_winner %>% group_by(fips) %>% top_n(1, pct)
# Grouping candidate winners by state, without removing any data.

length(factor(state_winner$state))
```

```
## [1] 53
```

```
# We have 53 levels instead of the 50 levels since there are 50 states.

state_winner$state[which(duplicated(state_winner$state))]
```

```
## [1] AK SD
## 52 Levels: AK AL AR AZ CA CO CT DC DE FL GA HI IA ID IL IN KS KY LA ... WY
```

```
# By this command we found that there were 2 duplicates AK and SD

election_state <- election_state[election_state$fips != "2000", ]
election_state <- election_state[election_state$fips != "46102", ]
state_votes <- election_state %>% group_by(fips) %>% add_tally(votes) %>% group_by(fips,
    candidate) %>% summarise_at(vars(pct = votes), funs(./n))
state_winner <- merge(election_state, state_votes, by = c("fips", "candidate"))
state_winner <- state_winner %>% group_by(fips) %>% top_n(1, pct)
# The states were AK and SD so they were moved from the original data.

length(factor(state_winner$state))
```

```
## [1] 51
```

```
# We did one more check to make sure we have 50 states, but we have 51
# levels still.

state_winner$state[c(which(state_winner$state %in% state.abb == F))]
```

```
## [1] DC
## 52 Levels: AK AL AR AZ CA CO CT DC DE FL GA HI IA ID IL IN KS KY LA ... WY
```

```
# We found that DC is not a state

election_state <- election_state[election_state$state != "DC", ]
state_votes <- election_state %>% group_by(fips) %>% add_tally(votes) %>% group_by(fips,
    candidate) %>% summarise_at(vars(pct = votes), funs(./n))
state_winner <- merge(election_state, state_votes, by = c("fips", "candidate"))
state_winner <- state_winner %>% group_by(fips) %>% top_n(1, pct)
# The state DC was removed from the original data.

length(factor(state_winner$state))
```

```
## [1] 50
```

```
# Now we have 50 states!
```

Visualization: 1. #5 is an inacurate visualization of total votes recieved by each candidate because our data set is incomplete. 2. Map of candidate won by state is a complete visualization 3. Map of candidate won by county has a few grey coloring because our data is incomplete for those areas. We would need to collect data specific to those counties to complete the map.

```
county.str <- maps::county.fips
length(factor(unique(county.str$fips)))
```

```
## [1] 3075
```

```
length(factor(unique(election$fips)))
```

```
## [1] 3111
```

```
county.str$fips[c(which(unique(election$fips) %in% unique(county.str$fips) ==
    F))]
```

```
##  [1]  5033  8099 10003 13061 13141 13307 17063 17153 18053 19163 20195
## [12] 21237 22101 26159 27155 29053 29099 29187 29510 30073 34027 35009
## [23] 36083 37137 37181 38089 39017 40107 41001 42129 47159 49015 49039
## [34] 49047 49049 49053 53065
```

```
# Missing counties by fip number, thus we have an incomplete data
```

Prediction model: 1. The knn for PCA resulted in the best model. We can reduce our dimensions to 14 PC and get accurate predictions.

```
records
```

```
##      train.error test.error
## tree  0.07939739 0.07980456
## knn   0.13029316 0.14169381
## lda           NA         NA
```

```
pca.records
```

```
##      train.error test.error
## tree  0.11400651 0.11237785
## knn   0.06270358 0.06840391
## lda           NA         NA
```

2. The lowest test error is the knn for PCA within our analysis methods, so this would be the best method out of knn, tree, and tree PCA.
3. LDA was not computed yet check #20 for further analysis.

Taking it further

20. Propose and tackle at least one interesting question. Be creative! Some possibilities are:

```
trn.cl.lda <- MASS::lda(candidate ~ ., data = trn.cl)

ypred.train.lda <- predict(trn.cl.lda, trn.cl)$class
ypred.test.lda <- predict(trn.cl.lda, tst.cl)$class

records[3, 1] <- calc_error_rate(ypred.train.lda, trn.cl$candidate)
records[3, 2] <- calc_error_rate(ypred.test.lda, tst.cl$candidate)
```

```r
# The LDA gives a low test error but it is not pratical, so we are going to
# perfrom PCA to see if we can reduce the dimensions with at least 90% of
# the variance explained


trn.cl.pca.lda <- MASS::lda(V ~ ., data = tr.pca)

ypred.train.pca.lda <- predict(trn.cl.pca.lda, tr.pca)$class
ypred.test.pca.lda <- predict(trn.cl.pca.lda, test.pca)$class

pca.records[3, 1] <- calc_error_rate(ypred.train.pca.lda, tr.pca$V)
pca.records[3, 2] <- calc_error_rate(ypred.test.pca.lda, test.pca$V)

records
```

```
##      train.error test.error
## tree  0.07939739 0.07980456
## knn   0.13029316 0.14169381
## lda   0.06962541 0.06351792
```

```r
pca.records
```

```
##      train.error test.error
## tree  0.11400651 0.11237785
## knn   0.06270358 0.06840391
## lda   0.09242671 0.08631922
```

Our best model is using LDA since it resulted in a lower test error, but it is not too practical since it has too many dimensions. The difference in prediction from LDA and knn for PCA is around .5% which is not a significant difference. Our final model should be knn for PCA rather than LDA since it is more practical.