PSTAT 131 final Project

Nathan Lai, Jinxiang Ma

March 12, 2022

Census Data

We essentially start with the 2017 United States county-level census data, which is available here. This dataset contains many demographic variables for each county in the U.S.

We load in and clean the census dataset by transforming the full state names to abbreviations (to match the education dataset in later steps). Specifically, R contains default global variables state.name and state.abb that store the full names and the associated abbreviations of the 50 states. However, it does not contain District of Columbia (and the associated DC). We added it back manually since census contains information in DC. We further remove data from Purto Rico to ease the visualization in later steps.

```
state.name <- c(state.name, "District of Columbia")
state.abb <- c(state.abb, "DC")
## read in census data
census <- read_csv("./acs2017_county_data.csv") %>% select(-CountyId, -ChildPoverty, -Income, -IncomeEr
  mutate(State = state.abb[match(`State`, state.name)]) %>%
  filter(State != "PR")
```

```
## Rows: 3220 Columns: 37
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): State, County
## dbl (35): CountyId, TotalPop, Men, Women, Hispanic, White, Black, Native, As...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(census)
```

```
## # A tibble: 6 x 31
##   State County   TotalPop   Men  Women Hispanic White Black Native Asian Pacific
##   <chr> <chr>       <dbl> <dbl>  <dbl>    <dbl> <dbl> <dbl>  <dbl> <dbl>   <dbl>
## 1 AL    Autauga~    55036 26899  28137      2.7  75.4  18.9    0.3   0.9       0
## 2 AL    Baldwin~   203360 99527 103833      4.4  83.1   9.5    0.8   0.7       0
## 3 AL    Barbour~    26201 13976  12225      4.2  45.7  47.8    0.2   0.6       0
## 4 AL    Bibb Co~    22580 12251  10329      2.4  74.6  22      0.4   0         0
## 5 AL    Blount ~    57667 28490  29177      9    87.4   1.5    0.3   0.1       0
## 6 AL    Bullock~    10478  5616   4862      0.3  21.6  75.6    1     0.7       0
## # ... with 20 more variables: VotingAgeCitizen <dbl>, Poverty <dbl>,
## #   Professional <dbl>, Service <dbl>, Office <dbl>, Construction <dbl>,
## #   Production <dbl>, Drive <dbl>, Carpool <dbl>, Transit <dbl>, Walk <dbl>,
## #   OtherTransp <dbl>, WorkAtHome <dbl>, MeanCommute <dbl>, Employed <dbl>,
## #   PrivateWork <dbl>, PublicWork <dbl>, SelfEmployed <dbl>, FamilyWork <dbl>,
## #   Unemployment <dbl>
```

Education data

We also include the education dataset, available at Economic Research Service at USDA. The dataset contains county-level educational attainment for adults age 25 and older in 1970-2019. We specifically use educational attainment information for the time period of 2015-2019.

To clean the data, we remove uninformative columns (as in FIPS Code, 2003 Rural-urban Continuum Code, 2003 Urban Influence Code, 2013 Rural-urban Continuum Code, and 2013 Urban Influence Code). To be consistent with census data, we exclude data from Purto Rico and we rename Area name to County in order to match that in the census dataset.

```
## read in education data
education <- read_csv("./Education.csv") %>%
  filter(!is.na(`2003 Rural-urban Continuum Code`)) %>%
  filter(State != "PR") %>%
  select(-`FIPS Code`,
         -`2003 Rural-urban Continuum Code`,
         -`2003 Urban Influence Code`,
         -`2013 Rural-urban Continuum Code`,
         -`2013 Urban Influence Code`) %>%
  rename(County = `Area name`)
```

```
## Rows: 3283 Columns: 47
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (3): FIPS Code, State, Area name
## dbl (24): 2003 Rural-urban Continuum Code, 2003 Urban Influence Code, 2013 R...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Preliminary data analysis

1. (1 pts) Report the dimension of census. (1 pts) Are there missing values in the data set? (1 pts) Compute the total number of distinct values in State in census to verify that the data contains all states and a federal district.

```
dim(census)
```

```
## [1] 3142   31
```

```
mean(is.na(census))>0
```

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

```
nrow(distinct(census, State))
```

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

Answer: The census dataset has 3142 rows and 31 columns. There is no missing values in this data set. There are 51 distinct state in the census data set.

2. (1 pts) Report the dimension of education. (1 pts) How many distinct counties contain missing values in the data set? (1 pts) Compute the total number of distinct values in County in education. (1 pts) Compare the values of total number of distinct county in education with that in census. (1 pts) Comment on your findings.

```
dim(education)
```

```
## [1] 3143   42
```

```r
nrow(distinct(education, County))
```

```
## [1] 1877
```

```r
nrow(distinct(census, County))
```

```
## [1] 1877
```

Data Wrangling

3. (2 pts) Remove all NA values in education, if there is any.

```r
education = na.omit(education)
mean(is.na(education))>0
```

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

4. (2 pts) In education, in addition to State and County, we will start only on the following 4 features: Less than a high school diploma, 2015-19, High school diploma only, 2015-19, Some college or associate's degree, 2015-19, and Bachelor's degree or higher, 2015-19. Mutate the education dataset by selecting these 6 features only, and create a new feature which is the total population of that county.

```r
education = education %>%
  select(c("State","County","Less than a high school diploma, 2015-19",
           "High school diploma only, 2015-19",
           "Some college or associate's degree, 2015-19",
           "Bachelor's degree or higher, 2015-19")) %>%
  mutate(total_population = rowSums(.[3:6]))
```

5. (3 pts) Construct aggregated data sets from education data: i.e., create a state-level summary into a data set named education.state

```r
education.state = aggregate(education[,3:7], by = list(education$State),FUN = sum)
head(education.state)
```

```
##   Group.1 Less than a high school diploma, 2015-19
## 1      AK                                    32338
## 2      AL                                   458922
## 3      AR                                   270168
## 4      AZ                                   604935
## 5      CA                                  4418675
## 6      CO                                   314312
##   High school diploma only, 2015-19 Some college or associate's degree, 2015-19
## 1                            126881                                      162816
## 2                           1022839                                      993344
## 3                            684659                                      593576
## 4                           1124129                                     1594817
## 5                           5423462                                     7648680
## 6                            810659                                     1114680
##   Bachelor's degree or higher, 2015-19 total_population
## 1                               137666           459701
## 2                               845772          3320877
## 3                               463236          2011639
## 4                              1392598          4716479
## 5                              8980726         26471543
## 6                              1538936          3778587
```

6. (4 pts) Create a data set named state.level on the basis of education.state, where you create a new feature which is the name of the education degree level with the largest population in that state.

```
level = education.state %>%
        select(c(2,3,4,5))%>%
        mutate(Largest_Level = names(.)[max.col(.)])
state.level = left_join(education.state,level)
```

```
## Joining, by = c("Less than a high school diploma, 2015-19", "High school diploma
## only, 2015-19", "Some college or associate's degree, 2015-19", "Bachelor's
## degree or higher, 2015-19")
```

```
head(state.level)
```

```
##   Group.1 Less than a high school diploma, 2015-19
## 1      AK                                    32338
## 2      AL                                   458922
## 3      AR                                   270168
## 4      AZ                                   604935
## 5      CA                                  4418675
## 6      CO                                   314312
##   High school diploma only, 2015-19 Some college or associate's degree, 2015-19
## 1                            126881                                      162816
## 2                           1022839                                      993344
## 3                            684659                                      593576
## 4                           1124129                                     1594817
## 5                           5423462                                     7648680
## 6                            810659                                     1114680
##   Bachelor's degree or higher, 2015-19 total_population
## 1                               137666           459701
## 2                               845772          3320877
## 3                               463236          2011639
## 4                              1392598          4716479
## 5                              8980726         26471543
## 6                              1538936          3778587
##                                  Largest_Level
## 1 Some college or associate's degree, 2015-19
## 2           High school diploma only, 2015-19
## 3           High school diploma only, 2015-19
## 4 Some college or associate's degree, 2015-19
## 5        Bachelor's degree or higher, 2015-19
## 6        Bachelor's degree or higher, 2015-19
```

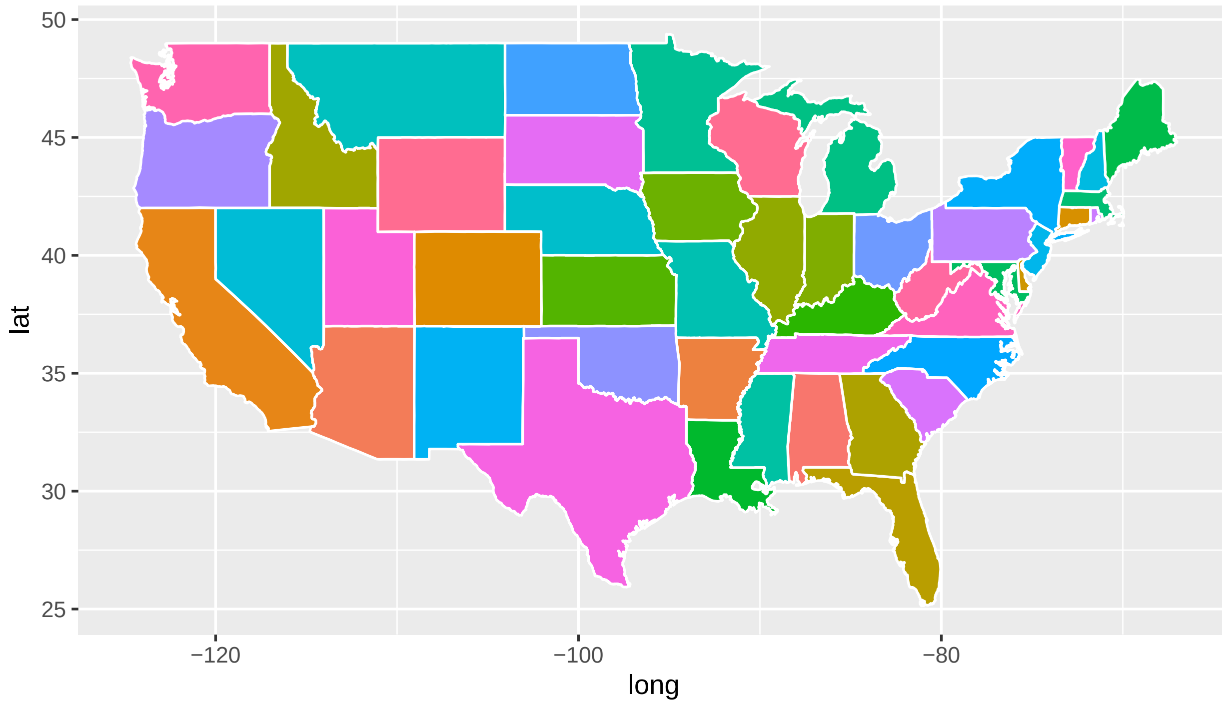
Visualization

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

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

```
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 for this example and takes too long
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```




```
head(states)
```

```
##      long   lat group order  region subregion
## 1 -87.46 30.39     1     1 alabama      <NA>
## 2 -87.48 30.37     1     2 alabama      <NA>
## 3 -87.53 30.37     1     3 alabama      <NA>
## 4 -87.53 30.33     1     4 alabama      <NA>
## 5 -87.57 30.33     1     5 alabama      <NA>
## 6 -87.59 30.33     1     6 alabama      <NA>
```

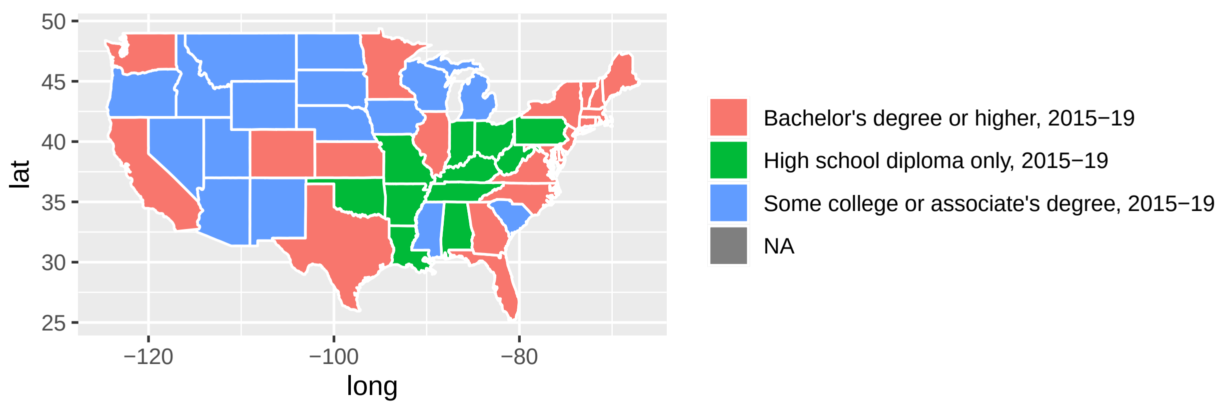
The variable states contain information to draw white polygons, and fill-colors are determined by region.

7. Now color the map (on the state level) by the education level with highest population for each state. Show the plot legend. First, combine states variable and state.level we created earlier using left_join(). Note that left_join() needs to match up values of states to join the tables. A call to left_join() takes all the values from the first table and looks for matches in the second table. If it finds a match, it adds the data from the second table; if not, it adds missing values: Here, we'll be combing the two data sets based on state name. However, the state names in states and state.level can be in different formats: check them! Before using left_join(), use certain transform to make sure the state names in the two data sets: states (for map drawing) and state.level (for coloring) are in the same formats. Then left_join().

```
states = states %>%
  select(c(-5))%>%
  mutate(Group.1 = state.abb[match(str_to_title(states$region), state.name)])
```

```r
states = left_join(states,state.level, by = "Group.1")
```

```r
ggplot(data = states) +
  geom_polygon(aes(x = long, y = lat , fill = Largest_Level, group = group),
               color = "white")+
  theme(legend.title = element_blank())+
  coord_fixed(1.3)
```



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

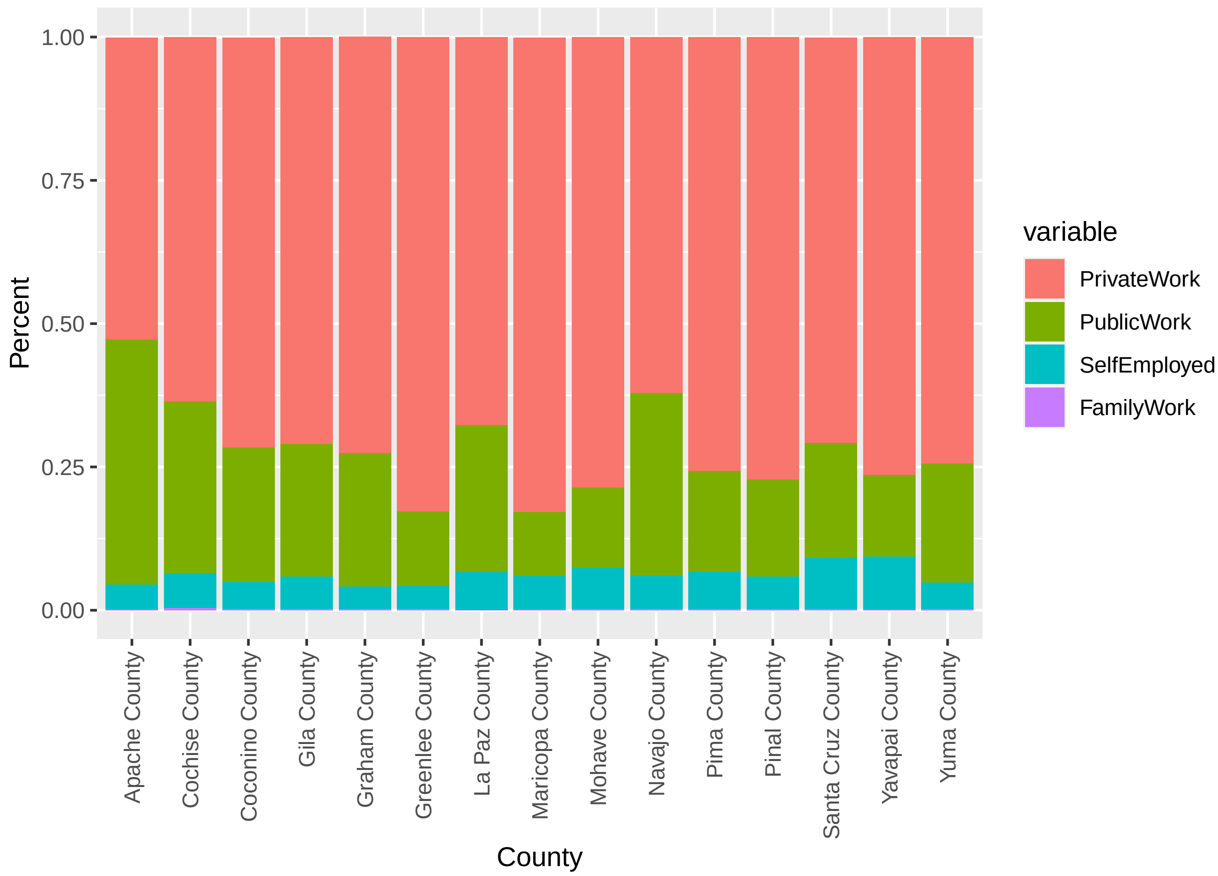
```r
#install.packages("RColorBrewer")
library(RColorBrewer)
#install.packages("reshape2")
library(reshape2)
```

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

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

```r
coul <- brewer.pal(3, "Pastel2")
census_employed = census %>%
    select(c(1,2,27,28,29,30)) %>%
    filter(State == "AZ")
```

```r
census_employed = melt(census_employed, na.rm = T, id.vars = c("State", "County"))
```

```r
ggplot(census_employed, aes(fill = variable, y = value/100, x = County))+
  ylab("Percent")+

  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))+
  geom_bar(position = "stack", stat = "identity")
```




```r
detach("package:reshape2", unload=TRUE)
```

9. The census data contains county-level census information. In this problem, we clean and aggregate the information as follows. (4 pts) Start with census, filter out any rows with missing values, convert {Men, Employed, VotingAgeCitizen} attributes to percentages, compute Minority attribute by combining {Hispanic, Black, Native, Asian, Pacific}, remove these variables after creating Minority, remove {Walk, PublicWork, Construction, Unemployment}. (Note that many columns are perfectly collineared, in which case one column should be deleted.)

```r
census = drop_na(census)
census %>%
  select(Men, Women, Employed, Unemployment, TotalPop, VotingAgeCitizen)
```

```
## # A tibble: 3,142 x 6
##      Men  Women Employed Unemployment TotalPop VotingAgeCitizen
##    <dbl>  <dbl>    <dbl>        <dbl>    <dbl>            <dbl>
##  1 26899  28137    24112          5.2    55036            41016
##  2 99527 103833    89527          5.5   203360           155376
```

```
##   3 13976  12225     8878         12.4    26201            20269
##   4 12251  10329     8171          8.2    22580            17662
##   5 28490  29177    21380          4.9    57667            42513
##   6  5616   4862     4290         12.1    10478             8212
##   7  9416  10710     7727          7.6    20126            15459
##   8 55593  59934    47392         10.1   115527            88383
##   9 16320  17575    14527          6.4    33895            26259
##  10 12862  12993     9879          5.3    25855            20620
## # ... with 3,132 more rows
```

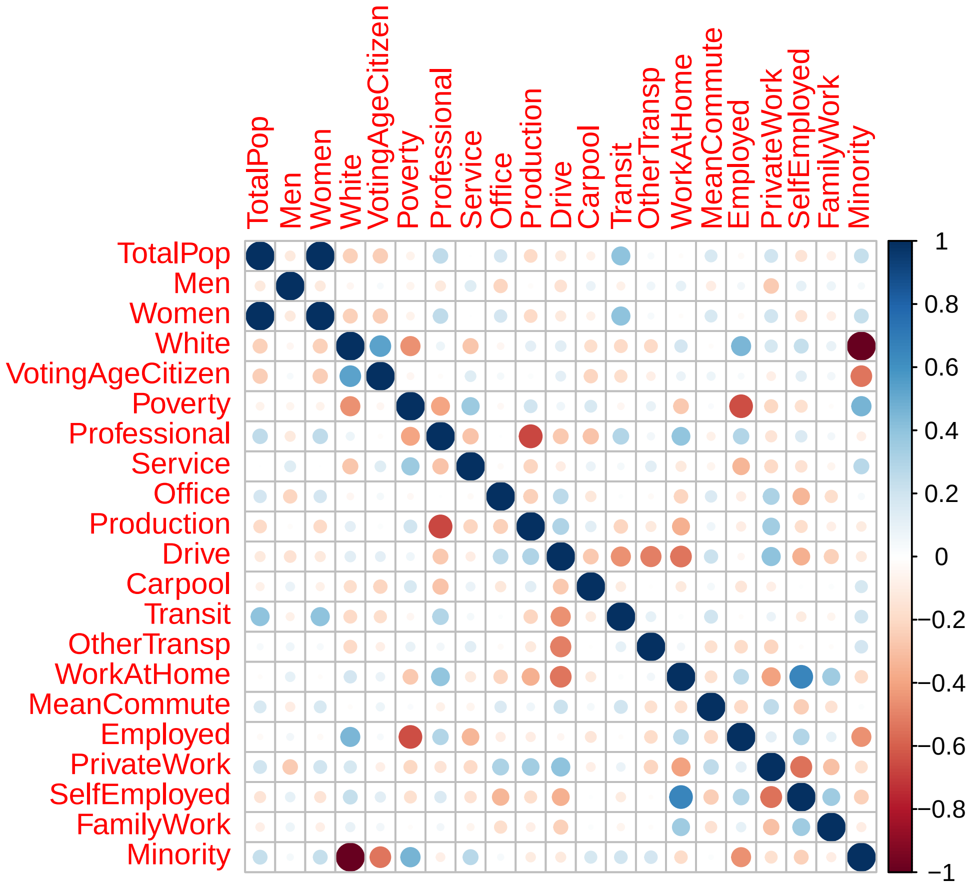
```
census.clean  = census%>%
  mutate(Men = Men/(Men + Women)) %>%
  mutate(Employed = 1 - (Unemployment/100)) %>%
  mutate(VotingAgeCitizen = VotingAgeCitizen / TotalPop) %>%
  mutate(Minority = rowSums(
    census[, c('Hispanic', 'Black', 'Native', 'Asian', 'Pacific')])) %>%
  select(-c('Hispanic', 'Black', 'Native', 'Asian', 'Pacific', 'Walk',
            'PublicWork', 'Construction', 'Unemployment'))
head(census.clean)
```

```
## # A tibble: 6 x 23
##   State County TotalPop   Men  Women White VotingAgeCitizen Poverty Professional
##   <chr> <chr>     <dbl> <dbl>  <dbl> <dbl>            <dbl>   <dbl>        <dbl>
## 1 AL    Autau~    55036 0.489  28137  75.4            0.745    13.7         35.3
## 2 AL    Baldw~   203360 0.489 103833  83.1            0.764    11.8         35.7
## 3 AL    Barbo~    26201 0.533  12225  45.7            0.774    27.2         25
## 4 AL    Bibb ~    22580 0.543  10329  74.6            0.782    15.2         24.4
## 5 AL    Bloun~    57667 0.494  29177  87.4            0.737    15.6         28.5
## 6 AL    Bullo~    10478 0.536   4862  21.6            0.784    28.5         19.7
## # ... with 14 more variables: Service <dbl>, Office <dbl>, Production <dbl>,
## #   Drive <dbl>, Carpool <dbl>, Transit <dbl>, OtherTransp <dbl>,
## #   WorkAtHome <dbl>, MeanCommute <dbl>, Employed <dbl>, PrivateWork <dbl>,
## #   SelfEmployed <dbl>, FamilyWork <dbl>, Minority <dbl>
```

```
#install.packages("corrplot")
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
census_selected = census.clean %>%
  select(-c("State", "County"))
corrplot::corrplot(cor(census_selected))
```



Upon above, Women and Total population, minority and white are perfectly collineared. Therefore, we decided to drop Women, and white.

```
census.clean = census.clean %>%
  select(-c("White","Women"))
```

10. Print the first 5 rows of census.clean

```
head(census.clean, 5)
```

```
## # A tibble: 5 x 21
##   State County      TotalPop   Men VotingAgeCitizen Poverty Professional Service
##   <chr> <chr>          <dbl> <dbl>            <dbl>   <dbl>        <dbl>   <dbl>
## 1 AL    Autauga Co~    55036 0.489            0.745    13.7         35.3    18
## 2 AL    Baldwin Co~   203360 0.489            0.764    11.8         35.7    18.2
## 3 AL    Barbour Co~    26201 0.533            0.774    27.2         25      16.8
## 4 AL    Bibb County    22580 0.543            0.782    15.2         24.4    17.6
## 5 AL    Blount Cou~    57667 0.494            0.737    15.6         28.5    12.9
## # ... with 13 more variables: Office <dbl>, Production <dbl>, Drive <dbl>,
## #   Carpool <dbl>, Transit <dbl>, OtherTransp <dbl>, WorkAtHome <dbl>,
## #   MeanCommute <dbl>, Employed <dbl>, PrivateWork <dbl>, SelfEmployed <dbl>,
## #   FamilyWork <dbl>, Minority <dbl>
```

Dimensionality reduction

Dimensionality reduction

11. Run PCA for the cleaned county level census data (with State and County excluded).

```
census.clean.pr = prcomp(census.clean[, c(-1, -2)], scale = TRUE, center = TRUE)
```

Save the first two principle components PC1 and PC2 into a two-column data frame, call it pc.county .

```
pc.county.x = tibble(PC1 = census.clean.pr$x[, 1], PC2 = census.clean.pr$x[, 2])
head(pc.county.x)
```

```
## # A tibble: 6 x 2
##     PC1     PC2
##   <dbl>   <dbl>
## 1 0.764  0.805
## 2 0.279  1.50
## 3 2.40  -1.90
## 4 1.58   0.0467
## 5 2.10   2.01
## 6 3.11  -2.95
```

```
pc.county.rotated = tibble(PC1 = census.clean.pr$rotation[, 1], PC2 = census.clean.pr$rotation[, 2])
pc.county.rotated
```

```
## # A tibble: 19 x 2
##        PC1      PC2
##      <dbl>    <dbl>
##  1  0.0203 -0.0475
##  2 -0.104  -0.126
##  3 -0.0545  0.185
##  4  0.228  -0.383
##  5 -0.305   0.109
##  6  0.0867 -0.327
##  7  0.175   0.119
##  8  0.265   0.0949
##  9  0.337   0.305
## 10  0.0557 -0.219
## 11 -0.0612 -0.122
## 12 -0.0921 -0.285
## 13 -0.433  -0.0114
## 14  0.188   0.0751
## 15 -0.249   0.365
## 16  0.297   0.315
## 17 -0.394  -0.00715
## 18 -0.238  -0.0535
## 19  0.142  -0.425
```

Discuss whether you chose to center and scale the features before running PCA and the reasons for your choice. We need to scale the features before running because some features are recorded on different scales. i.e. TotalPop is recorded in numbers but Men is recorded in poportion.

What are the three features with the largest absolute values of the first principal component?

```
head(sort(abs(census.clean.pr$rotation[, 1]), decreasing = TRUE), 3)
```

```
##   WorkAtHome SelfEmployed        Drive
##       0.4327       0.3935       0.3366
```

A: Minority, White, SelfEmployed has the largest absolute values of the first principal component.

Which features have opposite signs and what does that mean about the correlation between these features?

```
which(census.clean.pr$rotation[, 1] < 0)
```

```
##              Men VotingAgeCitizen     Professional          Transit
##                2                3                5               11
##      OtherTransp        WorkAtHome         Employed      SelfEmployed
##               12               13               15               17
##       FamilyWork
##               18
```

```
# Q11 Jason
filtered_data = census.clean %>%
        select(c(-1,-2))
pr.out = prcomp(filtered_data, scale = TRUE, center = TRUE)
pc.county = pr.out$x[,c(1,2)]
```

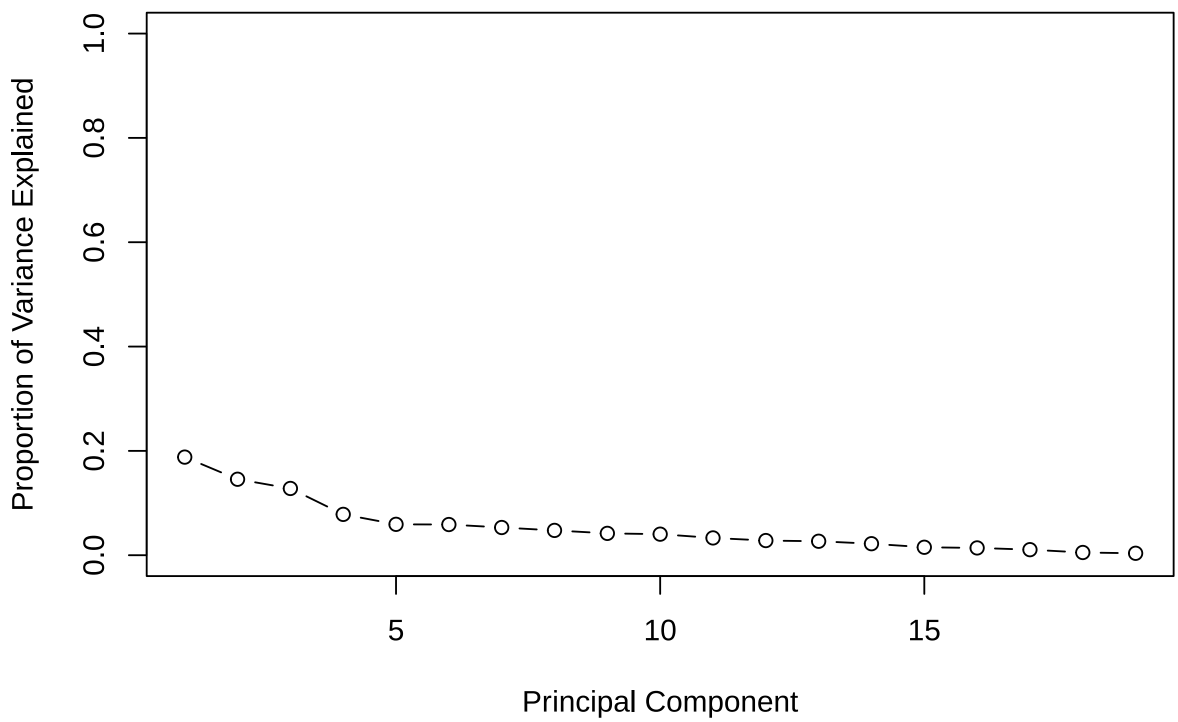
Answer: By setting the option scale = TRUE and center = TRUE, we scale the variables to have mean 0 and variance 1.

Determine the number of minimum number of PCs needed to capture 90% of the variance for the analysis. (2 pts) Plot proportion of variance explained (PVE) and cumulative PVE.

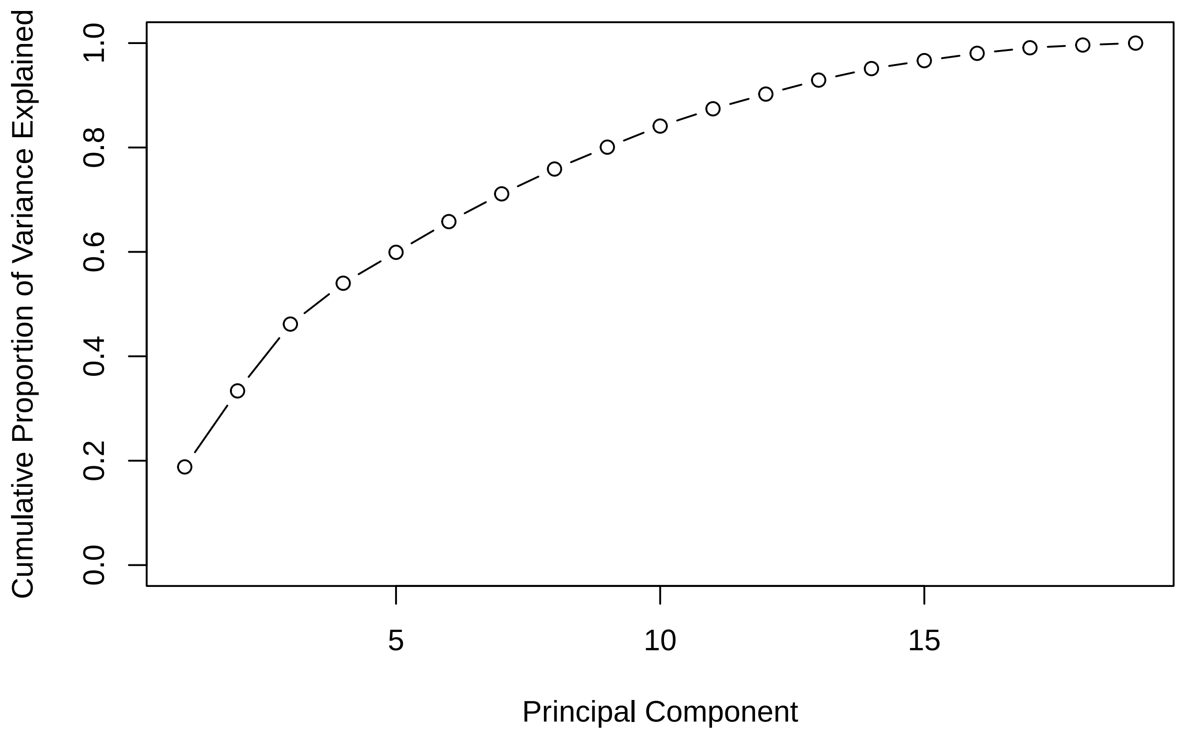
```
pr.var = pr.out$sdev^2

pve = pr.var/sum(pr.var)
table(cumsum(pve)<=0.9)["TRUE"]
```

```
## TRUE
##   11
```

```
plot(pve, xlab = "Principal Component",ylab = "Proportion of Variance Explained", ylim = c(0,1), type =
```




```
plot(cumsum(pve),xlab = "Principal Component",ylab = "Cumulative Proportion of Variance Explained", ylim
```



Clustering

With census.clean (with State and County excluded), perform hierarchical clustering with complete linkage. (2 pts) Cut the tree to partition the observations into 10 clusters.

```
census_scale = scale(filtered_data, center = TRUE, scale = TRUE)
distance = dist(census_scale, method = "euclidean")
set.seed(123)
census.hclust = hclust(distance, method = 'complete')
clus = cutree(census.hclust, 10)
table(clus)
```

```
## clus
##    1    2    3    4    5    6    7    8    9   10
## 2959    7    6    7  118   10    1   24    6    4
```

Re-run the hierarchical clustering algorithm using the first 2 principal components from pc.county as inputs instead of the original features.

```
pc_scale = scale(pc.county.x, center = TRUE, scale = TRUE)
pc_dist = dist(pc_scale, method = "euclidean")
set.seed(123)
pc.hclust = hclust(pc_dist, method = 'complete')
```

Compare the results and comment on your observations. For both approaches investigate the cluster that contains Santa Barbara County.

```r
clus1 = cutree(census.hclust, 10)
table(clus1)
```

```
## clus1
##    1    2    3    4    5    6    7    8    9   10
## 2959    7    6    7  118   10    1   24    6    4
```

```r
which(census.clean$County == "Santa Barbara County")
```

```
## [1] 228
```

```r
clus1[228]
```

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

```r
clus2 = cutree(pc.hclust,10)
table(clus2)
```

```
## clus2
##    1    2    3    4    5    6    7    8    9   10
## 1926   37  548  209    7   20  349   11    1   34
```

```r
which(census.clean$County == "Santa Barbara County")
```

```
## [1] 228
```

```r
clus2[228]
```

```
## [1] 4
```

Which approach seemed to put Santa Barbara County in a more appropriate clusters? Comment on what you observe and discuss possible explanations for these observations.

The second approach seemed to put Santa Barbara County in a more appropriate clusters. For the first hierarchical clustering, the Santa Barbara County is in the first cluster with 2959 counties. After we run hierarchical clustering on the PC1 and PC2, the Santa Barbara County is located in a smaller cluster(4th cluster, with only 209 counties), which means the data is more separable in each cluster.

Modeling

We start considering supervised learning tasks now. The most interesting/important question to ask is: can we use census information as well as the education information in a county to predict the level of poverty in that county? For simplicity, we are interested in a binary classification problem. Specifically, we will transform Poverty into a binary categorical variable: high and low, and conduct its classification. In order to build classification models, we first need to combine education and census.clean data (and removing all NAs), which can be achieved using the following code.

```r
# we join the two datasets
all <- census.clean %>%
  left_join(education, by = c("State"="State", "County"="County")) %>%
  na.omit
head(all)
```

```
## # A tibble: 6 x 26
##   State County      TotalPop   Men VotingAgeCitizen Poverty Professional Service
##   <chr> <chr>          <dbl> <dbl>            <dbl>   <dbl>        <dbl>   <dbl>
## 1 AL    Autauga Co~    55036 0.489            0.745    13.7         35.3    18
## 2 AL    Baldwin Co~   203360 0.489            0.764    11.8         35.7    18.2
## 3 AL    Barbour Co~    26201 0.533            0.774    27.2         25      16.8
## 4 AL    Bibb County    22580 0.543            0.782    15.2         24.4    17.6
```

```
## 5 AL    Blount Cou~    57667 0.494           0.737    15.6         28.5    12.9
## 6 AL    Bullock Co~    10478 0.536           0.784    28.5         19.7    17.1
## # ... with 18 more variables: Office <dbl>, Production <dbl>, Drive <dbl>,
## #   Carpool <dbl>, Transit <dbl>, OtherTransp <dbl>, WorkAtHome <dbl>,
## #   MeanCommute <dbl>, Employed <dbl>, PrivateWork <dbl>, SelfEmployed <dbl>,
## #   FamilyWork <dbl>, Minority <dbl>,
## #   `Less than a high school diploma, 2015-19` <dbl>,
## #   `High school diploma only, 2015-19` <dbl>,
## #   `Some college or associate's degree, 2015-19` <dbl>, ...
```

Transform the variable Poverty into a binary categorical variable with two levels: 1 if Poverty is greater than 20, and 0 if Poverty is smaller than or equal to 20. Remove features that you think are uninformative in classfication tasks. Partition the dataset into 80% training and 20% test data. Make sure to set.seed before the partition.

```
all = all %>%
  mutate(Poverty = as.factor(ifelse(all$Poverty > 20, 1, 0)))
```

```
colnames(all)[which(names(all) == "Less than a high school diploma, 2015-19")] <- "LessThanAHighSchoolDi
colnames(all)[which(names(all) == "High school diploma only, 2015-19")] <- "HighSchoolDiplomaOnly2015to
colnames(all)[which(names(all) == "Some college or associate's degree, 2015-19")] <- "SomeCollegeOrAsso
colnames(all)[which(names(all) == "Bachelor's degree or higher, 2015-19")] <- "BachelorsDegreeOrHigher2
```

```
all.selected.tree = all %>%
  mutate(LessThanAHighSchoolDiploma2015to19 = LessThanAHighSchoolDiploma2015to19 / TotalPop) %>%
  mutate(HighSchoolDiplomaOnly2015to19 = HighSchoolDiplomaOnly2015to19 / TotalPop) %>%
  mutate(SomeCollegeOrAssociatesDegree2015to19 = SomeCollegeOrAssociatesDegree2015to19 / TotalPop) %>%
  mutate(BachelorsDegreeOrHigher2015to19 = BachelorsDegreeOrHigher2015to19 / TotalPop) %>%
  select(c( "Poverty", "Professional", "Service", "Office", "Production", "Employed",
           "PrivateWork", "SelfEmployed", "FamilyWork", "Minority", "LessThanAHighSchoolDiploma2015to19
           "HighSchoolDiplomaOnly2015to19", "SomeCollegeOrAssociatesDegree2015to19",
           "BachelorsDegreeOrHigher2015to19"))
set.seed(123)
n <- nrow(all.selected.tree)
idx.tr <- sample.int(n, 0.8*n)
all.train <- all.selected.tree[idx.tr, ]
all.test <- all.selected.tree[-idx.tr, ]
```

Use the following code to define 10 cross-validation folds:

```
set.seed(123)
nfold <- 10
folds <- sample(cut(1:nrow(all.train), breaks=nfold, labels=FALSE))
```

Using the following error rate function. And the object records is used to record the classification performance of each method in the subsequent problems.

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

Classification

15. Decision tree:

train a decision tree by cv.tree().

```
tree.all = tree(Poverty~., data=all.train)
```

Prune tree to minimize misclassification error. Be sure to use the folds from above for cross-validation.

```
set.seed(123)
cv = cv.tree(tree.all, FUN=prune.misclass, k = folds)
```

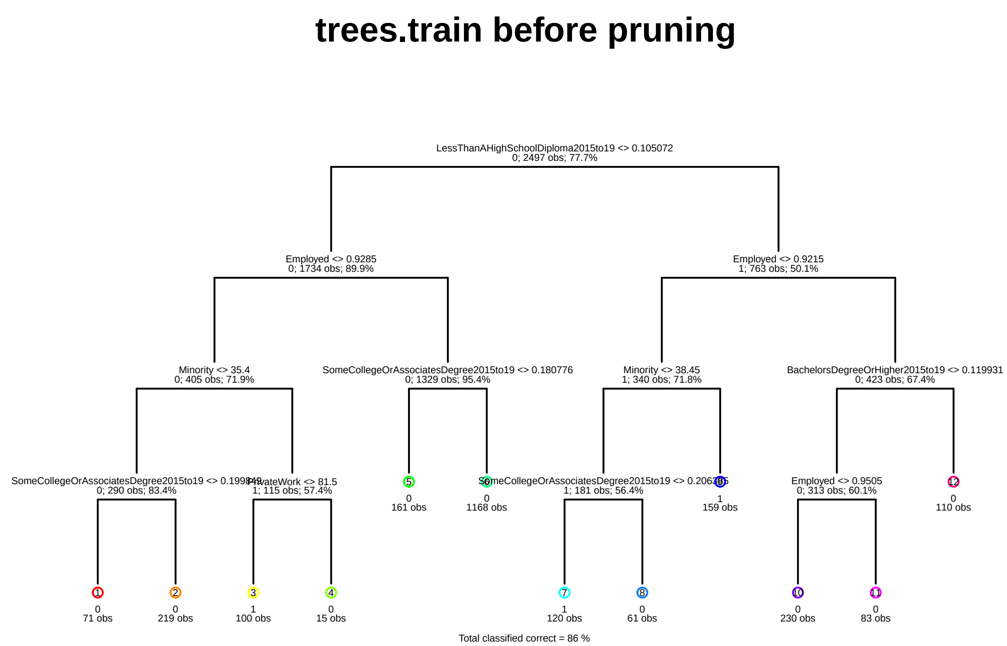
```
best.cv = min(cv$size[cv$dev == min(cv$dev)])
best.cv
```

```
## [1] 8
```

```
pt.cv = prune.misclass(tree.all, best = best.cv)
```

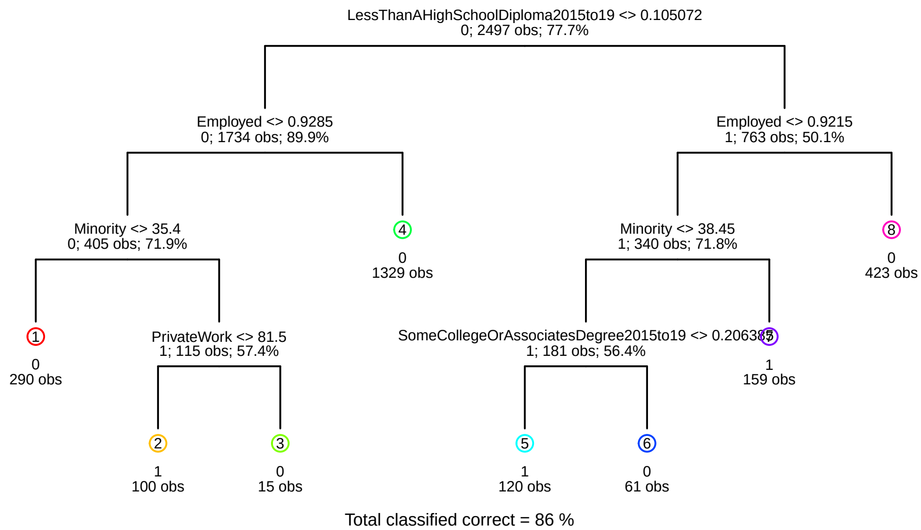
Visualize the trees before and after pruning.

```
draw.tree(tree.all, nodeinfo = TRUE, cex = 0.3)
title("trees.train before pruning")
```




```
draw.tree(pt.cv, nodeinfo = TRUE, cex = 0.5)
title("trees.train after pruning")
```

trees.train after pruning



Save training and test errors to records object.

```
Poverty.test = all.test$Poverty
tree.pred = predict(tree.all, all.test, type = "class")
tree_error.test = table(tree.pred, Poverty.test)
tree_error.test
```

```
##          Poverty.test
## tree.pred   0   1
##         0 460  71
##         1  20  74
```

```
#Test acccuracy rate
accu_test = sum(diag(tree_error.test))/sum(tree_error.test)
#Test error rate
tree_test_error = 1-accu_test

tree_test_error
```

```
## [1] 0.1456
```

```
Poverty.train = all.train$Poverty
tree.pred2 = predict(tree.all, all.train, type = "class")
tree_error.train = table(tree.pred2, Poverty.train)
tree_error.train
```

```
##           Poverty.train
## tree.pred2    0    1
```

```
##          0 1854  264
##          1   86  293
```

```r
#Test acccuracy rate
accu_train = sum(diag(tree_error.train))/sum(tree_error.train)
#Test error rate
tree_train_error = 1-accu_train

tree_train_error
```

```
## [1] 0.1402
```

```r
records[1,1] = tree_train_error
records[1,2] = tree_test_error
records
```

```
##          train.error test.error
## tree          0.1402     0.1456
## logistic          NA         NA
## lasso             NA         NA
```

Interpret and discuss the results of the decision tree analysis.

For the selected decision tree, The True Positive rate(TPR) is TP/(TP+FN) = 293/(293+264) = 0.526 The False Positive rate(FPR) is FP/(FN+TR) = 264/(264+1854) = 0.1246. When TPR is high, FPR is low, it implies that mis-classifications are low.

Use this plot to tell a story about Poverty.

By observing the prune tree, it is reasonable to conclude that Minority, Private Work and Education level are key factor that affects the poverty rate. For minority people working in the private sector, it is likely that these people get lower salary because of racism.

Run a logistic regression to predict Poverty in each county.

```r
#set.seed(123)
#n <- model.matrix(Poverty~., all.selected.glm)
#idx.tr2 <- sample.int(n, 0.8*n)



all.selected.glm = all %>%
  select(c( "Men", "Poverty", "Professional", "Service", "Office", "Production", "Employed",
           "PrivateWork", "SelfEmployed", "FamilyWork", "Minority",    "LessThanAHighSchoolDiploma2015to
           "HighSchoolDiplomaOnly2015to19", "SomeCollegeOrAssociatesDegree2015to19",
           "BachelorsDegreeOrHigher2015to19"))
```

```r
set.seed(123)
n <- nrow(all)
idx.tr <- sample.int(n, 0.8*n)
all.tr <- all.selected.glm[idx.tr, ]
all.te <- all.selected.glm[-idx.tr, ]

glm.poverty = glm(Poverty~., data = all.selected.glm, family = binomial)
```

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

```r
#summary(glm.poverty)
pred.training = predict(glm.poverty,all.tr, type ="response")
pred.test = predict(glm.poverty,all.te, type = "response")
all.tr = all.tr %>%
```

```
  mutate(predPoverty=as.factor(ifelse(pred.training<=0.5, 0, 1)))
all.te = all.te %>%
  mutate(predPoverty = as.factor(ifelse(pred.test <= 0.5, 0, 1 )))
logi_test_error = calc_error_rate(all.te$predPoverty,all.te$Poverty)
print(logi_test_error)
```

```
## [1] 0.144
```

```
logi_train_error = calc_error_rate(all.tr$predPoverty, all.tr$Poverty)
print(logi_train_error)
```

```
## [1] 0.1442
```

Save training and test errors to records variable.

```
records[2,1] = logi_train_error
records[2,2] = logi_test_error
records
```

```
##          train.error test.error
## tree          0.1402     0.1456
## logistic      0.1442     0.1440
## lasso             NA         NA
```

What are the significant variables?

```
which(summary(glm.poverty)$coeff[-1,4] < 0.05)
```

```
##                                  Men                              Service
##                                    1                                    3
##                           Production                             Employed
##                                    5                                    6
##                          PrivateWork                          SelfEmployed
##                                    7                                    8
##                             Minority    LessThanAHighSchoolDiploma2015to19
##                                   10                                   11
## SomeCollegeOrAssociatesDegree2015to19
##                                   13
```

```
summary(glm.poverty)
```

```
##
## Call:
## glm(formula = Poverty ~ ., family = binomial, data = all.selected.glm)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -3.385  -0.532  -0.293  -0.043   4.540
##
## Coefficients:
##                                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)                           4.45e+01   3.30e+00   13.48  < 2e-16 ***
## Men                                  -1.77e+01   2.50e+00   -7.08  1.5e-12 ***
## Professional                         -1.08e-02   1.90e-02   -0.57  0.57205
## Service                               7.73e-02   2.12e-02    3.64  0.00027 ***
## Office                                1.15e-03   2.36e-02    0.05  0.96124
## Production                            6.51e-02   1.84e-02    3.53  0.00042 ***
## Employed                             -3.47e+01   2.54e+00  -13.67  < 2e-16 ***
```

```
## PrivateWork                           -8.63e-02   1.29e-02   -6.68  2.4e-11 ***
## SelfEmployed                          -1.02e-01   2.42e-02   -4.22  2.4e-05 ***
## FamilyWork                            -8.41e-02   1.53e-01   -0.55  0.58248
## Minority                               3.18e-02   3.18e-03    9.99  < 2e-16 ***
## LessThanAHighSchoolDiploma2015to19     7.43e-05   1.36e-05    5.48  4.2e-08 ***
## HighSchoolDiplomaOnly2015to19          1.06e-06   8.50e-06    0.12  0.90072
## SomeCollegeOrAssociatesDegree2015to19 -6.14e-05   1.15e-05   -5.34  9.3e-08 ***
## BachelorsDegreeOrHigher2015to19       -1.07e-06   3.85e-06   -0.28  0.78185
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3328.0  on 3121  degrees of freedom
## Residual deviance: 2090.1  on 3107  degrees of freedom
## AIC: 2120
##
## Number of Fisher Scoring iterations: 7
```

Are they consistent with what you saw in decision tree analysis? Yes. Employed, Minority and Private Work are important features in decision tree analysis.

Interpret the meaning of a couple of the significant coefficients in terms of a unit change in the variables.

Let take Employed and Minority as examples. A one unit increase in Minority will result in 0.00318 increase in Poverty. A one unit increase in Employed will result in 0.347 decrease in Poverty.

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

Use the cv.glmnet function from the glmnet library to run a 10-fold cross validation and select the best regularization parameter for the logistic regression with LASSO penalty. Set lambda = seq(1, 20) * 1e-5 in cv.glmnet() function to set pre-defined candidate values for the tuning parameter .

```
set.seed(123)
x = model.matrix(Poverty~.,all.selected.glm)[,-1]
y = all.selected.glm$Poverty
lasso.x.train = x[idx.tr,]
lasso.y.train = y[idx.tr]
lasso.x.test = x[-idx.tr,]
lasso.y.test = y[-idx.tr]
```

```
set.seed(123)
lambda.val = seq(1, 20) * 1e-5
cv.out.lasso=cv.glmnet(lasso.x.train, lasso.y.train, nfolds=10, lambda = lambda.val, alpha = 1, family =
bestlam.l = cv.out.lasso$lambda.min
bestlam.l
```

```
## [1] 0.00019
```

```
lasso.all = glmnet(lasso.x.train, lasso.y.train, alpha = 1, family = "binomial")
predict(lasso.all, type="coefficients", s=bestlam.l)
```

```
## 15 x 1 sparse Matrix of class "dgCMatrix"
##                                              s1
## (Intercept)                           4.293e+01
```

```
## Men                                    -1.907e+01
## Professional                             .
## Service                                 7.783e-02
## Office                                  7.257e-03
## Production                              6.816e-02
## Employed                               -3.187e+01
## PrivateWork                            -9.390e-02
## SelfEmployed                           -1.276e-01
## FamilyWork                             -1.579e-01
## Minority                                3.027e-02
## LessThanAHighSchoolDiploma2015to19      9.267e-05
## HighSchoolDiplomaOnly2015to19          -8.609e-06
## SomeCollegeOrAssociatesDegree2015to19 -7.147e-05
## BachelorsDegreeOrHigher2015to19          .
```

What is the optimal value of in cross validation? A: best λ from cross validation is 0.00019.

What are the non-zero coefficients in the LASSO regression for the optimal value of ?

A: Most coefficient in the Lasso regression are non-zero except Professional and BachelorsDegreeOrHigher2015to19

How do they compare to the unpenalized logistic regression?

Lasso shrinks the coefficients of the less contributive variables toward zero. Compare to the unpenalized logistic model, it is helpful to avoid overfitting.

Comment on the comparison.

For Lasso regression, Men, Employed and Family work are important features, whereas in unpenalized logistic regression, all of these variables are significant.

Save training and test errors to the records variable.

```r
lasso.pred.train = predict(lasso.all,s = bestlam.l, newx = lasso.x.train)
lasso.pred.train_mod = as.factor(ifelse(lasso.pred.train > 0.5, 1, 0))
lasso_train.error=calc_error_rate(lasso.pred.train_mod,lasso.y.train)
print(lasso_train.error)
```

```
## [1] 0.1498
```

```r
lasso.pred.test = predict(lasso.all, s = bestlam.l, newx = lasso.x.test)
lasso.pred.test_mod = as.factor(ifelse(lasso.pred.test > 0.5, 1, 0))
lasso_test.error = calc_error_rate(lasso.pred.test_mod,lasso.y.test)
print(lasso_test.error)
```

```
## [1] 0.1696
```

```r
records[3,1] = lasso_train.error
records[3,2] = lasso_test.error
records
```

```
##          train.error test.error
## tree          0.1402     0.1456
## logistic      0.1442     0.1440
## lasso         0.1498     0.1696
```

```r
#Load package for ROC curve
library(ROCR)
```

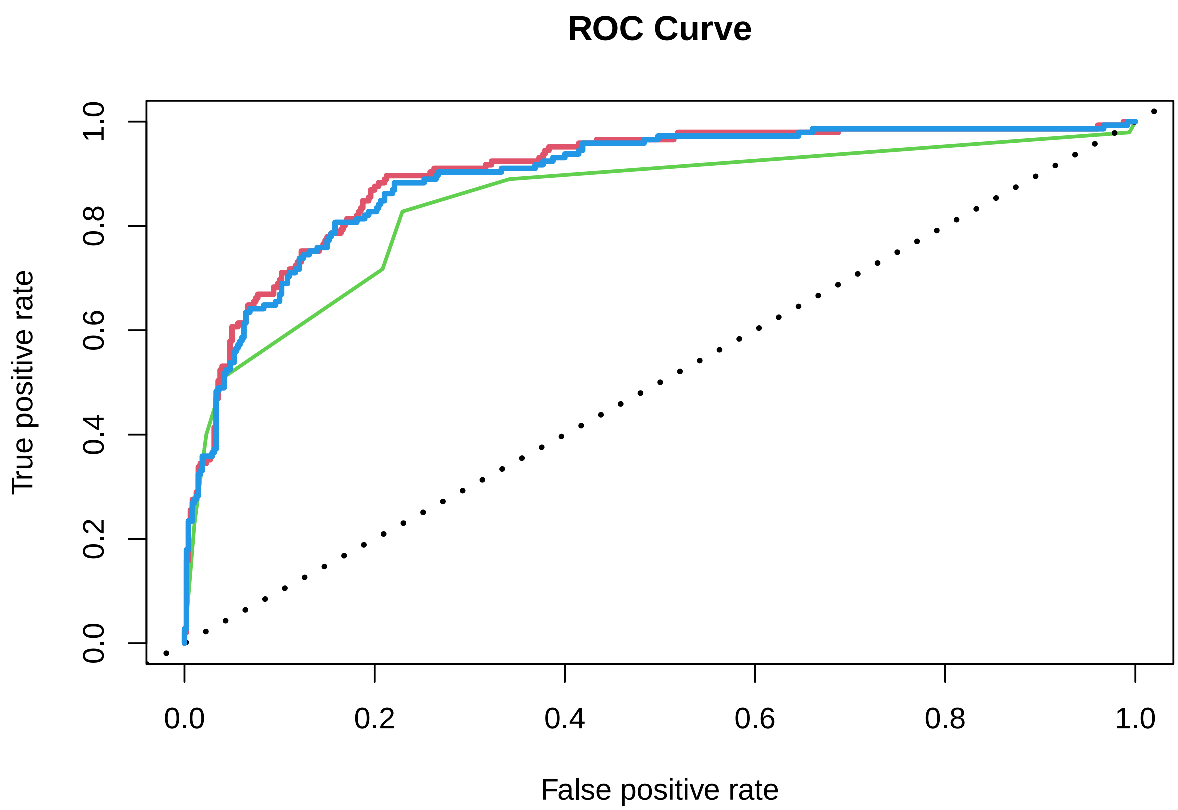
18. (6 pts) Compute ROC curves for the decision tree, logistic regression and LASSO logistic regression using predictions on the test data. Display them on the same plot.

```
# ROC on the decision tree test data
DTPrediction = predict(pt.cv, all.test, type = "vector")
Tree_Predict = prediction(DTPrediction[,2],all.test$Poverty)
Tree_Perf = performance(Tree_Predict,"tpr","fpr")
```

```
# ROC on the logistic regression test data
pred.logistic.test = prediction(pred.test, all.te$Poverty)
logistic_perf = performance(pred.logistic.test, measure = "tpr", x.measure = "fpr")
```

```
# ROC on the LASSO test data
#lasso.pred.test = predict(lasso.all, s = bestlam.l, newx = lasso.x.test, type = "prob")
pred.LASSO.test = prediction(lasso.pred.test , lasso.y.test)
lasso_perf = performance(pred.LASSO.test, measure = "tpr", x.measure = "fpr")
```

```
#Plot all ROC curve in one plot.
plot(Tree_Perf, main = "ROC Curve", col = 3, lwd = 2)
abline(0,1,lwd = 3, lty = 3, col = "black")
plot(logistic_perf, add = TRUE, col = 2, lwd = 3)
plot(lasso_perf, add = TRUE, col = 4, lwd = 3)
legend(2,4,legend = c("Decision Tree", "Logistic Regrssion", "Lasso"), fill = c("green","red","blue"))
```



Explore additional classification methods. Consider applying additional two classification methods from KNN, LDA, QDA, SVM, random forest, boosting, neural networks etc. (You may research and use methods beyond those covered in this course). How do these compare to the tree method, logistic regression, and the lasso logistic regression?

random forest

```
# setting training data & test data
set.seed(123)
n <- nrow(all.selected.tree)
idx.tr <- sample.int(n, 0.8*n)
all.train.rf <- all.selected.tree[idx.tr, ]
all.test.rf <- all.selected.tree[-idx.tr, ]
```

```
# apply randomforest
rf.all = randomForest(Poverty~. , data = all.train.rf, mtry = 13, importance = TRUE)
rf.all
```

```
##
## Call:
##  randomForest(formula = Poverty ~ ., data = all.train.rf, mtry = 13,      importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 13
##
##         OOB estimate of  error rate: 13.18%
## Confusion matrix:
##      0   1 class.error
## 0 1821 119     0.06134
## 1  210 347     0.37702
```

```
# bagging error test error
test.rf = predict(rf.all, newdata = all.test, type = "response")
test.rf.error = mean(test.rf != all.test$Poverty)
test.rf.error
```

```
## [1] 0.1248
```

LDA

```
library(MASS)
```

```
##
## Attaching package: 'MASS'

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

```
lda_fit = lda(Poverty ~., data = all.selected.glm)
lda_fit
```

```
## Call:
## lda(Poverty ~ ., data = all.selected.glm)
##
## Prior probabilities of groups:
##      0      1
## 0.7751 0.2249
##
## Group means:
```

```
##      Men Professional Service Office Production Employed PrivateWork
## 0 0.5012        32.43   17.62  21.88      15.58   0.9445       75.90
## 1 0.4987        28.50   19.79  21.48      17.16   0.9085       72.48
##   SelfEmployed FamilyWork Minority LessThanAHighSchoolDiploma2015to19
## 0        8.013     0.2905    16.86                              9082
## 1        6.895     0.2548    35.70                              6322
##   HighSchoolDiplomaOnly2015to19 SomeCollegeOrAssociatesDegree2015to19
## 0                         21360                                 23410
## 1                         10943                                  9962
##   BachelorsDegreeOrHigher2015to19
## 0                           26841
## 1                            8356
##
## Coefficients of linear discriminants:
##                                             LD1
## Men                                  -9.658e+00
## Professional                         -2.223e-02
## Service                               3.307e-02
## Office                               -1.661e-02
## Production                            3.345e-02
## Employed                             -2.066e+01
## PrivateWork                          -4.914e-02
## SelfEmployed                         -3.362e-02
## FamilyWork                           -2.549e-02
## Minority                              2.439e-02
## LessThanAHighSchoolDiploma2015to19    1.024e-05
## HighSchoolDiplomaOnly2015to19        -5.997e-06
## SomeCollegeOrAssociatesDegree2015to19 -7.988e-06
## BachelorsDegreeOrHigher2015to19       2.967e-06
```

```
lda_preds = predict(lda_fit,all.selected.glm)
str(lda_preds)
```

```
## List of 3
##  $ class    : Factor w/ 2 levels "0","1": 1 1 2 1 1 2 2 2 1 1 ...
##  $ posterior: num [1:3122, 1:2] 0.896 0.977 0.134 0.752 0.968 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:3122] "1" "2" "3" "4" ...
##   .. ..$ : chr [1:2] "0" "1"
##  $ x        : num [1:3122, 1] -0.0468 -0.9723 2.2622 0.5533 -0.7663 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:3122] "1" "2" "3" "4" ...
##   .. ..$ : chr "LD1"
```

```
error_lda = table(class = all.selected.glm$Poverty,pred = lda_preds$class)
error_lda/rowSums(error_lda)
```

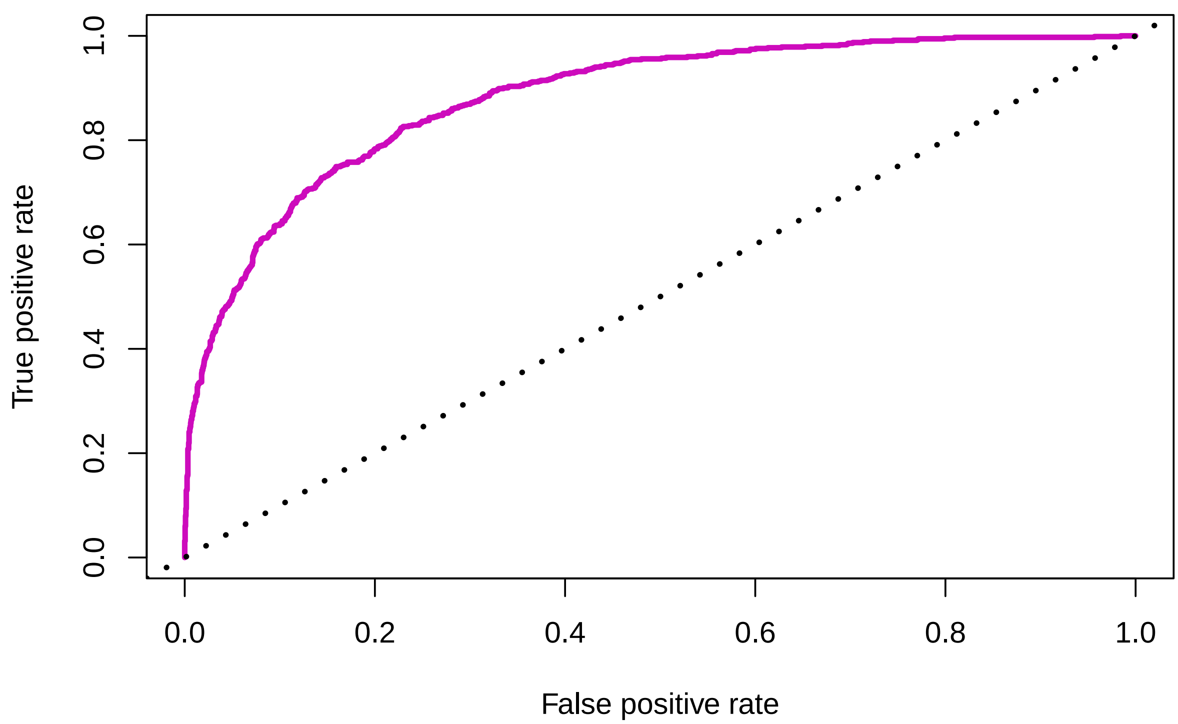
```
##      pred
## class       0       1
##     0 0.96074 0.03926
##     1 0.53561 0.46439
```

```
lda_preds $ posterior %>% head()
```

```
##         0       1
## 1 0.89582 0.10418
```

```
## 2 0.97728 0.02272
## 3 0.13412 0.86588
## 4 0.75170 0.24830
## 5 0.96780 0.03220
## 6 0.04837 0.95163
```

```r
#ROC curve for LDA
prediction_lda = prediction(predictions = lda_preds$posterior[,2], labels = all.selected.glm$Poverty)
perf_lda = performance(prediction.obj = prediction_lda, 'tpr','fpr')

plot(perf_lda, col = 6, lwd = 3)
abline(0,1,lwd = 3, lty = 3, col = "black")
```




```r
# Finding AUC value for LDA
auc_lda = performance(prediction_lda,"auc")@y.values
auc_lda
```

```
## [[1]]
## [1] 0.8809
```

```r
detach("package:MASS", unload=TRUE)
```

How do these compare to the tree method, logistic regression, and the lasso logistic regression?

```r
auc_tree = performance(Tree_Predict,"auc")@y.values
auc_logi = performance(pred.logistic.test,"auc")@y.values
auc_lasso = performance(pred.LASSO.test,"auc")@y.values
```

```r
auc_tree
```

```
## [[1]]
## [1] 0.8442
```

```r
auc_logi
```

```
## [[1]]
## [1] 0.8996
```

```r
auc_lasso
```

```
## [[1]]
## [1] 0.8929
```

Comment: By the comparing the AUC rate of LDA method, tree method, logistic regression and lasso regression,all of these model are acceptable. The lasso regression has the high rate of AUC, which is 0.8929. It implies that the lasso regression has the best performance.

Tackle at least one more interesting question. Creative and thoughtful analysis will be rewarded! Some possibilities for further exploration are: Consider a regression problem! Use regression models to predict the actual value of Poverty (before we transformed Poverty to a binary variable) by county. Compare and contrast these results with the classification models. Which do you prefer and why? How might they complement one another?

```r
#cleaning data
all2 <- census.clean %>%
  left_join(education, by = c("State"="State", "County"="County")) %>%
  na.omit
colnames(all2)[which(names(all2) == "Less than a high school diploma, 2015-19")] <- "LessThanAHighSchoo
colnames(all2)[which(names(all2) == "High school diploma only, 2015-19")] <- "HighSchoolDiplomaOnly2015
colnames(all2)[which(names(all2) == "Some college or associate's degree, 2015-19")] <- "SomeCollegeOrAss
colnames(all2)[which(names(all2) == "Bachelor's degree or higher, 2015-19")] <- "BachelorsDegreeOrHighe
all.selected.lm = all2 %>%
  select(c( "Poverty", "Professional", "Service", "Office", "Production", "Employed",
           "PrivateWork", "SelfEmployed", "FamilyWork", "Minority", "LessThanAHighSchoolDiploma2015to19
           "HighSchoolDiplomaOnly2015to19", "SomeCollegeOrAssociatesDegree2015to19",
           "BachelorsDegreeOrHigher2015to19"))
set.seed(123)
n.lm <- nrow(all.selected.lm)
idx.tr.lm <- sample.int(n.lm, 0.8*n.lm)
all.train.lm <- all.selected.lm[idx.tr.lm, ]
all.test.lm <- all.selected.lm[-idx.tr.lm, ]
```

```r
lm.all = lm(Poverty~., data = all.train.lm)
summary(lm.all)
```

```
##
## Call:
## lm(formula = Poverty ~ ., data = all.train.lm)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -29.462  -2.668  -0.394   2.229  20.553
##
## Coefficients:
##                                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)                              1.03e+02   4.10e+00   25.14  < 2e-16 ***
```

```
## Professional                           -8.58e-02   2.58e-02   -3.32  0.00092 ***
## Service                                 2.32e-01   3.39e-02    6.85  9.1e-12 ***
## Office                                  6.44e-02   3.76e-02    1.71  0.08656 .
## Production                              2.27e-01   2.95e-02    7.70  1.9e-14 ***
## Employed                               -8.70e+01   3.56e+00  -24.42  < 2e-16 ***
## PrivateWork                            -1.73e-01   1.84e-02   -9.39  < 2e-16 ***
## SelfEmployed                           -8.29e-02   3.26e-02   -2.54  0.01110 *
## FamilyWork                              4.32e-01   2.08e-01    2.08  0.03789 *
## Minority                                7.29e-02   5.40e-03   13.51  < 2e-16 ***
## LessThanAHighSchoolDiploma2015to19      3.36e-05   7.18e-06    4.68  3.0e-06 ***
## HighSchoolDiplomaOnly2015to19           1.80e-06   9.20e-06    0.20  0.84494
## SomeCollegeOrAssociatesDegree2015to19 -3.63e-05   8.94e-06   -4.06  5.1e-05 ***
## BachelorsDegreeOrHigher2015to19         6.72e-06   3.38e-06    1.99  0.04649 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.34 on 2483 degrees of freedom
## Multiple R-squared:  0.558,  Adjusted R-squared:  0.556
## F-statistic:  241 on 13 and 2483 DF,  p-value: <2e-16
```

```
which(summary(lm.all)$coeff[-1,4] < 0.05)
```

```
##                          Professional                               Service
##                                     1                                     2
##                            Production                              Employed
##                                     4                                     5
##                           PrivateWork                          SelfEmployed
##                                     6                                     7
##                            FamilyWork                              Minority
##                                     8                                     9
##    LessThanAHighSchoolDiploma2015to19 SomeCollegeOrAssociatesDegree2015to19
##                                    10                                    12
##       BachelorsDegreeOrHigher2015to19
##                                    13
```

```
# prediction
pred.lm =predict(lm.all, all.test.lm)
```

```
# Compute errors: error
error.lm = pred.lm  - all.test.lm[["Poverty"]]

# Calculate RMSE
sqrt(mean(error.lm^2))
```

```
## [1] 4.331
```

```
pred.training.lm = predict(lm.all, all.train.lm, type ="response")
pred.test.lm = predict(lm.all,all.test.lm, type = "response")
all.train.lm = all.train.lm %>%
  mutate(predPoverty=as.factor(ifelse(pred.training.lm <= 20, 0, 1)))
all.train.lm = all.train.lm %>%
  mutate(Poverty=as.factor(ifelse(Poverty <= 20, 0, 1)))
all.test.lm = all.test.lm %>%
  mutate(predPoverty = as.factor(ifelse(pred.test.lm <= 20, 0, 1 )))
all.test.lm = all.test.lm %>%
  mutate(Poverty = as.factor(ifelse(Poverty <= 20, 0, 1 )))
```

```r
lm_test_error = calc_error_rate(all.test.lm$predPoverty, all.test.lm$Poverty)
print(lm_test_error)
```

```
## [1] 0.1376
```

```r
lm_train_error = calc_error_rate(all.train.lm$predPoverty, all.train.lm$Poverty)
print(lm_train_error)
```

```
## [1] 0.1626
```

We convert the predicted poverty and true poverty into dummy variables 1 and 0 after we fit the linear regression model. For linear regression, test error rate is 0.1376, and train error rate is 0.1626. Comparing to the logistic regression, test error rate is 0.1440, and train error rate is 0.1442. Also, for the linear regression we computed the RMSE = 4.331.We conclude that linear regression is better, since it has smaller test error rate.

Interpret and discuss any overall insights gained in this analysis and possible explanations. Use any tools at your disposal to make your case: visualize errors on the map, discuss what does/doesn't seems reasonable based on your understanding of these methods, propose possible directions (collecting additional data, domain knowledge, etc).

```r
# choose logistic regression to predict poverty and graph the map to see each error rate by state
all.selected.glm2 = all %>%
  select(c( "State","Men", "Poverty", "Professional", "Service", "Office", "Production", "Employed",
           "PrivateWork", "SelfEmployed", "FamilyWork", "Minority", "LessThanAHighSchoolDiploma2015to19
           "HighSchoolDiplomaOnly2015to19", "SomeCollegeOrAssociatesDegree2015to19",
           "BachelorsDegreeOrHigher2015to19"))

pred.logistic = predict(glm.poverty, all.selected.glm2[-c(1)], type = "response")

all.selected.glm2 = all.selected.glm2 %>%
  mutate(pred.logistic=as.factor(ifelse(pred.logistic<=0.5, 0, 1)))
```

For logistic regression the train.error is slightly larger than the test.error, which means the model, that means for the logistic model it may be overfitting.

For LASSO since it puts a constrain on the regression, it performs better than logistic regression.

For random forest, since we have too many data it is hard to use for real life prediction, because it takes too long to predict.

For decision tree, since it is unstatble, it is hard to compare to other decision predictors.

In this project, we think lasso did better job than other model, since it has the largest AUC value.