Pstat131 Project

By : Minh Pham (#3609831) and Eva Koujikov (#9775404)

```
library(scales)
library(dplyr)
library(ISLR)
library(tidyverse)
library(ROCR)
library(ggridges)
library(dendextend)
library(maps)
library(tree)
library(tibble)
library(maptree)
library(glmnet)
library(randomForest)
library(class)
library(FNN)
library(reshape2)
library(ggplot2)
```

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.

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.

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.

```r
# check dimensions of census
dim(census)
```

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

```r
# check if theres missing values
any(is.na(census))
```

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

Census is a dataset containing 3142 rows and 31 columns

There is no missing values in census

```r
# print all unique values of State in census
unique(census$State)
```

```
##  [1] "AL" "AK" "AZ" "AR" "CA" "CO" "CT" "DE" "DC" "FL" "GA" "HI" "ID" "IL" "IN"
## [16] "IA" "KS" "KY" "LA" "ME" "MD" "MA" "MI" "MN" "MS" "MO" "MT" "NE" "NV" "NH"
## [31] "NJ" "NM" "NY" "NC" "ND" "OH" "OK" "OR" "PA" "RI" "SC" "SD" "TN" "TX" "UT"
## [46] "VT" "VA" "WA" "WV" "WI" "WY"
```

```r
 # print the total # of ^
length(unique(census$State))
```

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

There are 51 unique values in state, 50 of them being all of the different states and 1 of them being the federal district

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.

```r
# check dimensions of education
dim(education)
```

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

```r
# check if there are any missing values in the County column of education
n_distinct(education[rowSums(is.na(education)) > 0,]$County)
```

```
## [1] 18
```

Education is a dataset containing 3143 rows and 42 columns

There is 18 distinct counties that contain missing values

```r
# print the total # of County in education
length(unique(education$County))
```

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

```r
# print the total # of County in census
length(unique(census$County))
```

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

```r
# Check their differences
length(unique(education$County)) - length(unique(census$County))
```

```
## [1] 0
```

There are 1877 distinct values in County in the education dataset. County in the education dataset has the same number of distinct values as county in the census dataset.

Data Wrangling

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

```r
# There are NA value
any(is.na(education))
```

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

```r
# Removed 18 rows
education = na.omit(education)
```

There were 18 rows that contained NA values so we removed them.

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
#list of the features
features = 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")

# prune education data by only selection the features + State and County
education = education %>% select(State, County, features)
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(features)` instead of `features` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```
# create total population by adding up all the features
education = education %>% mutate(TotalPop = rowSums(education[3:6]))
```

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

```
# aggregate education by state but dont include State and County, using sum function
education.state = aggregate(education[-c(1,2)], education["State"], sum)
```

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.

```
index = 0
new_col = 0
for(i in seq(1:51)){
  index[i] = which.max(education.state[i,2:5]) + 1
  new_col[i] = colnames(education.state)[index[i]]
}
state.level <- education.state %>% mutate(education_level = new_col)
state.level
```

```
##    State 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
## 7     CT                                   232663
## 8     DC                                    44850
## 9     DE                                    66816
## 10    FL                                  1767583
## 11    GA                                   885498
## 12    HI                                    79715
## 13    IA                                   165514
## 14    ID                                   102929
## 15    IL                                   937042
## 16    IN                                   495390
## 17    KS                                   172126
## 18    KY                                   414784
## 19    LA                                   461706
## 20    MA                                   441944
## 21    MD                                   405463
## 22    ME                                    71985
## 23    MI                                   626190
## 24    MN                                   258391
## 25    MO                                   418266
```

```
## 26    MS                                              306105
## 27    MT                                               46650
## 28    NC                                              853396
## 29    ND                                               36394
## 30    NE                                              107535
## 31    NH                                               66206
## 32    NJ                                              625931
## 33    NM                                              199172
## 34    NV                                              272132
## 35    NY                                             1796594
## 36    OH                                              767378
## 37    OK                                              310453
## 38    OR                                              269250
## 39    PA                                              848910
## 40    RI                                               82467
## 41    SC                                              430491
## 42    SD                                               47241
## 43    TN                                              575128
## 44    TX                                             2957959
## 45    UT                                              140800
## 46    VA                                              588440
## 47    VT                                               32276
## 48    WA                                              442449
## 49    WI                                              308216
## 50    WV                                              168624
## 51    WY                                               26688
##    High school diploma only, 2015-19
## 1                             126881
## 2                            1022839
## 3                             684659
## 4                            1124129
## 5                            5423462
## 6                             810659
## 7                             666828
## 8                              83185
## 9                             209449
## 10                           4276237
## 11                           1909067
## 12                            271631
## 13                            648398
## 14                            305181
## 15                           2254524
## 16                           1480084
## 17                            492807
## 18                            993098
## 19                           1061388
## 20                           1148525
## 21                           1018653
## 22                            306589
## 23                           1967316
## 24                            928450
## 25                           1270622
## 26                            601355
## 27                            208541
```

```
## 28                          1791532
## 29                           130842
## 30                           326594
## 31                           263321
## 32                          1670942
## 33                           365499
## 34                           574254
## 35                          3541274
## 36                          2634997
## 37                           812102
## 38                           659085
## 39                          3106571
## 40                           208387
## 41                          1003177
## 42                           173079
## 43                          1472003
## 44                          4525099
## 45                           416545
## 46                          1371838
## 47                           126832
## 48                          1122330
## 49                          1211981
## 50                           519091
## 51                           113535
##    Some college or associate's degree, 2015-19
## 1                                       162816
## 2                                       993344
## 3                                       593576
## 4                                      1594817
## 5                                      7648680
## 6                                      1114680
## 7                                       608139
## 8                                        76822
## 9                                       178917
## 10                                     4450224
## 11                                     1936098
## 12                                      314146
## 13                                      681042
## 14                                      399909
## 15                                     2484708
## 16                                     1282863
## 17                                      602276
## 18                                      880129
## 19                                      848474
## 20                                     1102149
## 21                                     1052168
## 22                                      285655
## 23                                     2234804
## 24                                     1220892
## 25                                     1248599
## 26                                      633057
## 27                                      236435
## 28                                     2156078
## 29                                      179075
```

```
## 30                                          417672
## 31                                          275073
## 32                                         1408555
## 33                                          439621
## 34                                          692618
## 35                                         3308262
## 36                                         2318076
## 37                                          808024
## 38                                          994695
## 39                                         2184466
## 40                                          194228
## 41                                         1044024
## 42                                          186993
## 43                                         1286117
## 44                                         5227820
## 45                                          646202
## 46                                         1544770
## 47                                          113869
## 48                                         1699233
## 49                                         1244179
## 50                                          334314
## 51                                          143438
##    Bachelor's degree or higher, 2015-19 TotalPop
## 1                                137666   459701
## 2                                845772  3320877
## 3                                463236  2011639
## 4                               1392598  4716479
## 5                               8980726 26471543
## 6                               1538936  3778587
## 7                                975465  2483095
## 8                                289259   494116
## 9                                214138   669320
## 10                              4471701 14965745
## 11                              2157616  6888279
## 12                               327451   992943
## 13                               597831  2092785
## 14                               307537  1115556
## 15                              3010025  8686299
## 16                              1172156  4430493
## 17                               635070  1902279
## 18                               731082  3019093
## 19                               753585  3125153
## 20                              2089065  4781683
## 21                              1662724  4139008
## 22                               309888   974117
## 23                              1985170  6813480
## 24                              1359167  3766900
## 25                              1212562  4150049
## 26                               435153  1975670
## 27                               231669   723295
## 28                              2182853  6983859
## 29                               148757   495068
## 30                               399225  1251026
## 31                               355737   960337
```

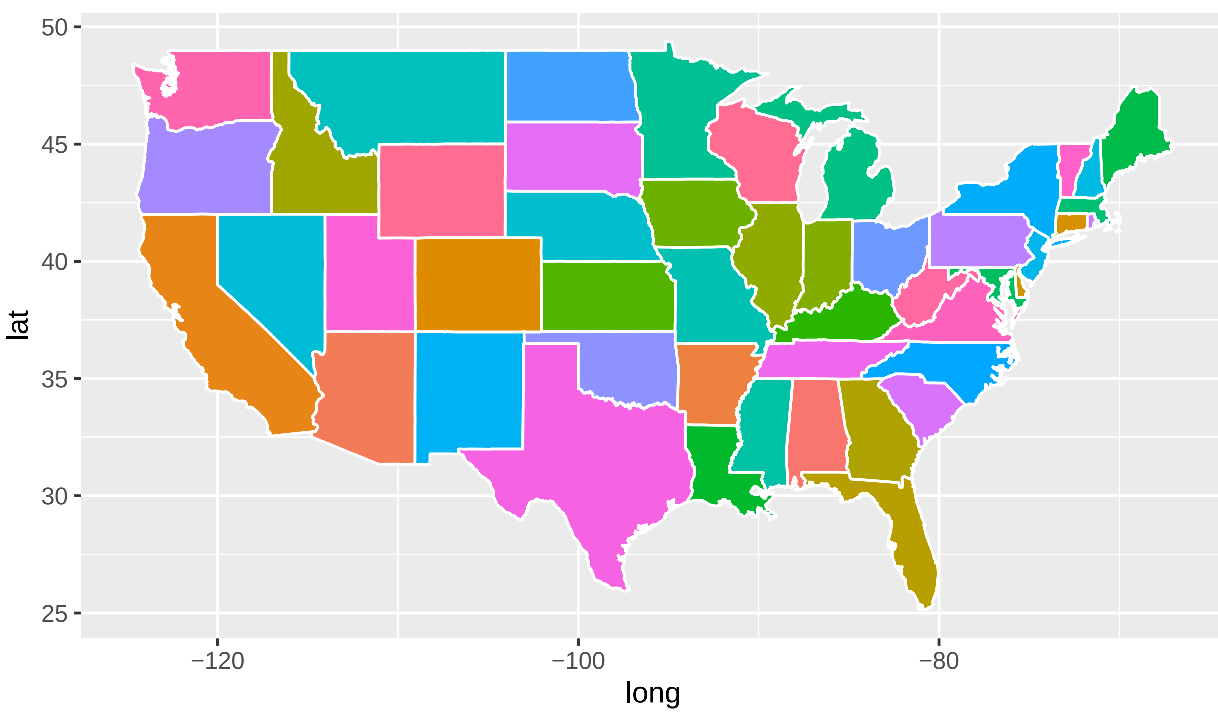
```
## 32                              2440951  6146379
## 33                               380945  1385237
## 34                               505691  2044695
## 35                              4985807 13631937
## 36                              2255326  7975777
## 37                               661509  2592088
## 38                               975920  2898950
## 39                              2814285  8954232
## 40                               252118   737200
## 41                               969279  3446971
## 42                               165080   572393
## 43                              1254145  4587393
## 44                              5420676 18131554
## 45                               620505  1824052
## 46                              2225876  5730924
## 47                               167483   440460
## 48                              1837612  5101624
## 49                              1191329  3955705
## 50                               265398  1287427
## 51                               106855   390516
##                                education_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
## 7         Bachelor's degree or higher, 2015-19
## 8         Bachelor's degree or higher, 2015-19
## 9         Bachelor's degree or higher, 2015-19
## 10        Bachelor's degree or higher, 2015-19
## 11        Bachelor's degree or higher, 2015-19
## 12        Bachelor's degree or higher, 2015-19
## 13 Some college or associate's degree, 2015-19
## 14 Some college or associate's degree, 2015-19
## 15        Bachelor's degree or higher, 2015-19
## 16           High school diploma only, 2015-19
## 17        Bachelor's degree or higher, 2015-19
## 18           High school diploma only, 2015-19
## 19           High school diploma only, 2015-19
## 20        Bachelor's degree or higher, 2015-19
## 21        Bachelor's degree or higher, 2015-19
## 22        Bachelor's degree or higher, 2015-19
## 23 Some college or associate's degree, 2015-19
## 24        Bachelor's degree or higher, 2015-19
## 25           High school diploma only, 2015-19
## 26 Some college or associate's degree, 2015-19
## 27 Some college or associate's degree, 2015-19
## 28        Bachelor's degree or higher, 2015-19
## 29 Some college or associate's degree, 2015-19
## 30 Some college or associate's degree, 2015-19
## 31        Bachelor's degree or higher, 2015-19
## 32        Bachelor's degree or higher, 2015-19
## 33 Some college or associate's degree, 2015-19
```

```
## 34 Some college or associate's degree, 2015-19
## 35         Bachelor's degree or higher, 2015-19
## 36            High school diploma only, 2015-19
## 37            High school diploma only, 2015-19
## 38 Some college or associate's degree, 2015-19
## 39            High school diploma only, 2015-19
## 40         Bachelor's degree or higher, 2015-19
## 41 Some college or associate's degree, 2015-19
## 42 Some college or associate's degree, 2015-19
## 43            High school diploma only, 2015-19
## 44         Bachelor's degree or higher, 2015-19
## 45 Some college or associate's degree, 2015-19
## 46         Bachelor's degree or higher, 2015-19
## 47         Bachelor's degree or higher, 2015-19
## 48         Bachelor's degree or higher, 2015-19
## 49 Some college or associate's degree, 2015-19
## 50            High school diploma only, 2015-19
## 51 Some college or associate's degree, 2015-19
```

Visualization

```r
library(ggplot2)
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
```



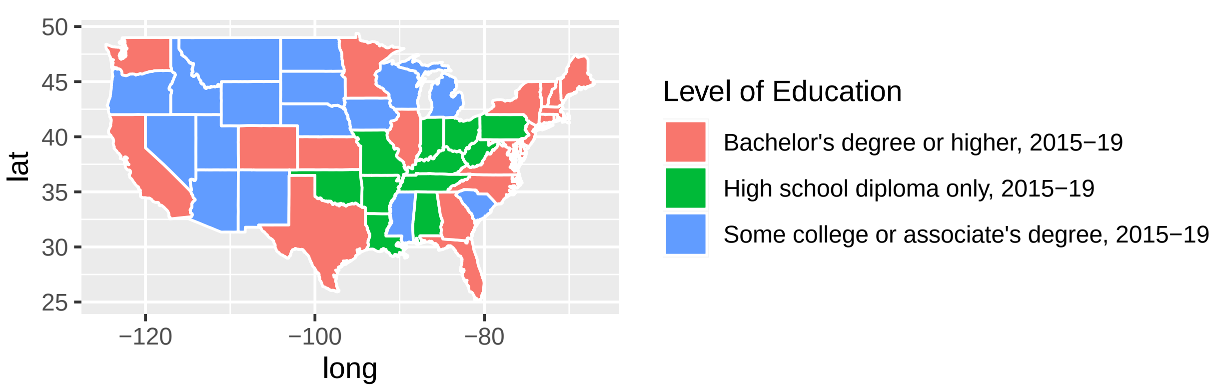
7. (6 pts) Now color the map (on the state level) by the education level with highest population for each state. Show the plot legend.

```
# convert regopm column from the full name of the States to the abbreviations
states$region <- state.abb[match(states$region,tolower(state.name))]
# convert column name region to States
names(states)[5] <- 'State'
```

```
# use left join of states and state.level to create stateseducation
stateseducation <- left_join(states, state.level)
```

```
## Joining, by = "State"
```

```
# plot states map by education level
ggplot(data = stateseducation) +
  geom_polygon(aes(x = long, y = lat, fill = education_level, group = group),
               color = "white") +
  coord_fixed(1.3) +
  guides(fill=guide_legend(title = 'Level of Education'))
```



8. (6 pts) (Open-ended) Create a visualization of your choice using census data. Use this R graph gallery for ideas and inspiration.

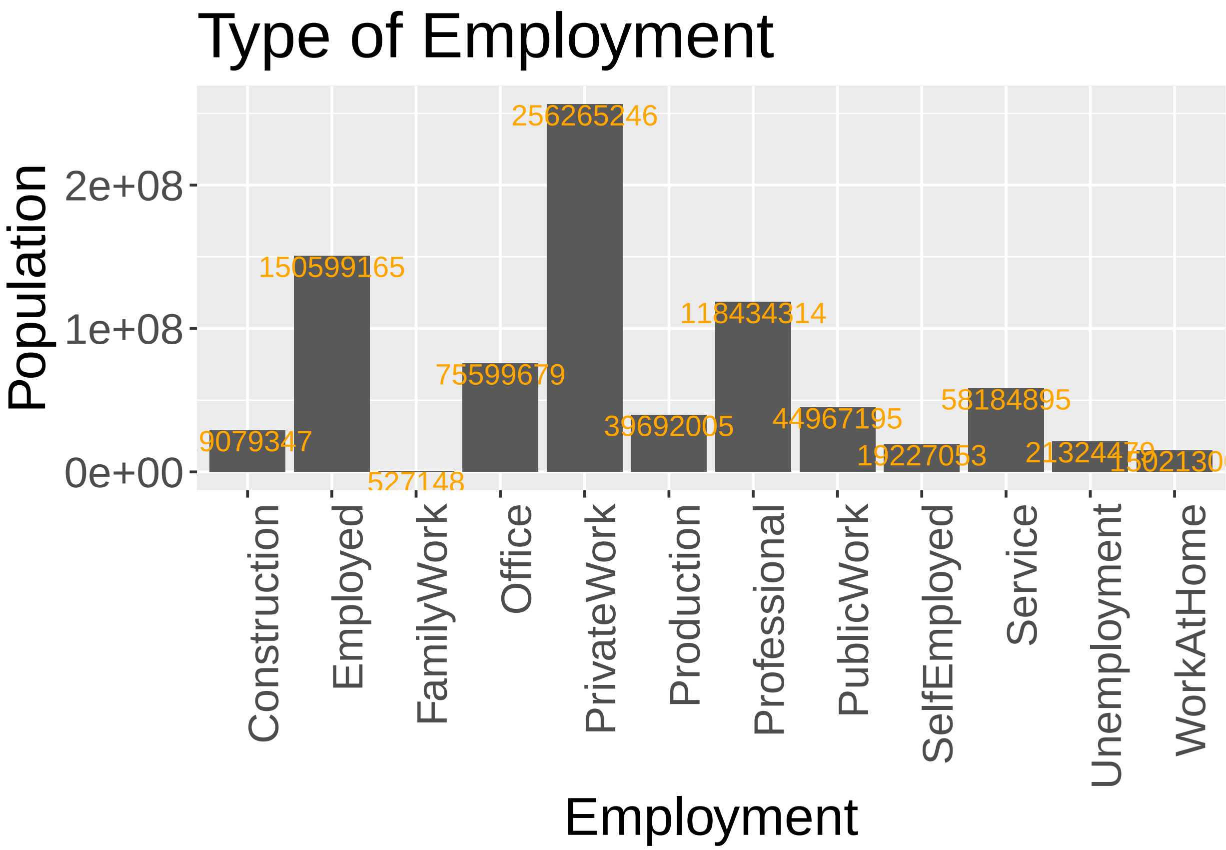
```
census.narm <- na.omit(census)
employment <- c("Professional",
                "Service",
                "Office",
                "Construction",
                "Production",
                "WorkAtHome",
                "Employed",
                "PrivateWork",
                "PublicWork",
                "SelfEmployed",
                "FamilyWork",
                "Unemployment")

census.prune = census.narm %>% select(c(State,
                                        TotalPop,
```

```
                                        Professional,
                                        Service,
                                        Office,
                                        Construction,
                                        Production,
                                        WorkAtHome,
                                        Employed,
                                        PrivateWork,
                                        PublicWork,
                                        SelfEmployed,
                                        FamilyWork,
                                        Unemployment)) %>%
  mutate(Professional = floor(TotalPop * Professional / 100.0)) %>%
  mutate(Service = floor(TotalPop * Service / 100.0)) %>%
  mutate(Office = floor(TotalPop * Office / 100.0)) %>%
  mutate(Construction = floor(TotalPop * Construction / 100.0)) %>%
  mutate(Production = floor(TotalPop * Production / 100.0)) %>%
  mutate(WorkAtHome = floor(TotalPop * WorkAtHome / 100.0)) %>%
  mutate(PrivateWork = floor(TotalPop * PrivateWork / 100.0)) %>%
  mutate(PublicWork = floor(TotalPop * PublicWork / 100.0)) %>%
  mutate(SelfEmployed = floor(TotalPop * SelfEmployed / 100.0)) %>%
  mutate(FamilyWork = floor(TotalPop * FamilyWork / 100.0)) %>%
  mutate(Unemployment = floor(TotalPop * Unemployment/ 100.0)) %>%
  select(-c(TotalPop))

census.prune = aggregate(. ~ State, data = census.prune, FUN = sum)
```

```
work = colSums(census.prune %>% select(-c(State)))
ggplot(data.frame(employment, work), aes(x = employment, y = work)) +
  geom_col() +
  geom_text(aes(label = work), vjust = 1, colour = "orange") +
  labs(x = "Employment", y = "Population", title = "Type of Employment") +
  theme(text = element_text(size=20),
        axis.text.x = element_text(angle=90, hjust=1))
```



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.)

```
set.seed(123)
# remove missing values within census data
census = na.omit(census)

# convert Men, Employed, and VotingAgeCitizen to percentage
census$Men <- (census$Men / census$TotalPop) * 100
census$Employed <- (census$Employed / census$TotalPop) * 100
census$VotingAgeCitizen <- (census$VotingAgeCitizen / census$TotalPop) * 100

#Create Minorty column by adding Hispanic, Black, Native,
# Asian, and Pacific columns then removing them
census$Minority <- census$Hispanic +
  census$Black +
  census$Native +
  census$Asian +
  census$Pacific
census.clean <- census %>% select(-c(Hispanic,
                                     Black,
                                     Native,
```

```
                                    Asian,
                                    Pacific,
                                    Walk,
                                    PublicWork,
                                    Construction,
                                    Unemployment))
```

10. (1 pts) Print the first 5 rows of census.clean

```r
# check first 5 rows of cleaned census data
head(census.clean,5)
```

```
## # A tibble: 5 x 23
##   State County TotalPop   Men  Women White VotingAgeCitizen Poverty Professional
##   <chr> <chr>     <dbl> <dbl>  <dbl> <dbl>            <dbl>   <dbl>        <dbl>
## 1 AL    Autau~    55036  48.9  28137  75.4             74.5    13.7         35.3
## 2 AL    Baldw~   203360  48.9 103833  83.1             76.4    11.8         35.7
## 3 AL    Barbo~    26201  53.3  12225  45.7             77.4    27.2         25
## 4 AL    Bibb ~    22580  54.3  10329  74.6             78.2    15.2         24.4
## 5 AL    Bloun~    57667  49.4  29177  87.4             73.7    15.6         28.5
## # ... 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>
```

Dimensionality Reduction

11. Run PCA for the cleaned county level census data (with State and County excluded). (2 pts) Save the first two principle components PC1 and PC2 into a two-column data frame, call it pc.county. (2 pts) Discuss whether you chose to center and scale the features before running PCA and the reasons for your choice. (2 pts) What are the three features with the largest absolute values of the first principal component? (2 pts) Which features have opposite signs and what does that mean about the correlation between these features?

```r
# check variance of data
apply(census.clean, 2, var)
```

```
##            State           County         TotalPop              Men
##               NA               NA     1.077758e+11     5.862892e+00
##            Women            White VotingAgeCitizen          Poverty
##     2.793755e+10     4.050465e+02     2.817957e+01     4.300811e+01
##     Professional          Service           Office       Production
##     4.277364e+01     1.358740e+01     9.297472e+00     3.380418e+01
##            Drive          Carpool          Transit      OtherTransp
##     5.884476e+01     8.485439e+00     9.616366e+00     2.828719e+00
##       WorkAtHome      MeanCommute         Employed      PrivateWork
##     9.462904e+00     3.179096e+01     4.269568e+01     5.706079e+01
##     SelfEmployed       FamilyWork         Minority
##     1.495226e+01     2.040316e-01     3.977816e+02
```

```r
set.seed(123)
# do pr composition of data only without
# variables State and County (it will be scaled and centered)
pr.out = census.clean %>% select(-c("State", "County"))
pr.out = prcomp(pr.out, scale = TRUE)

# create dataframe of pc1 and pc2 called pc.county
pc.county <- data.frame(pr.out$x[,1],pr.out$x[,2])
```

I chose to scale and center before performing PCA because the variables have vastly different variances. There are columns of different units and I wanted to ensure each column had an equal weight of importance. For example, some variables were converted to percentage such as Men while others remained as a count such as Women.

```r
pr.out$rotation[,1]
```

```
##         TotalPop              Men            Women            White
##      0.084209128     -0.035162988      0.084294222     -0.322449822
## VotingAgeCitizen          Poverty     Professional          Service
##     -0.159465979      0.322372526     -0.282769854      0.173800329
##           Office       Production            Drive          Carpool
##      0.125842149      0.176167904      0.202193080      0.115910771
##          Transit      OtherTransp        WorkAtHome      MeanCommute
##      0.008546658     -0.004590811     -0.376893965      0.149501935
##         Employed      PrivateWork     SelfEmployed       FamilyWork
##     -0.295453817      0.156212383     -0.345203844     -0.195438701
##         Minority
##      0.326089271
```

three features with the largest absolute values of the first principal component :

SelfEmployed

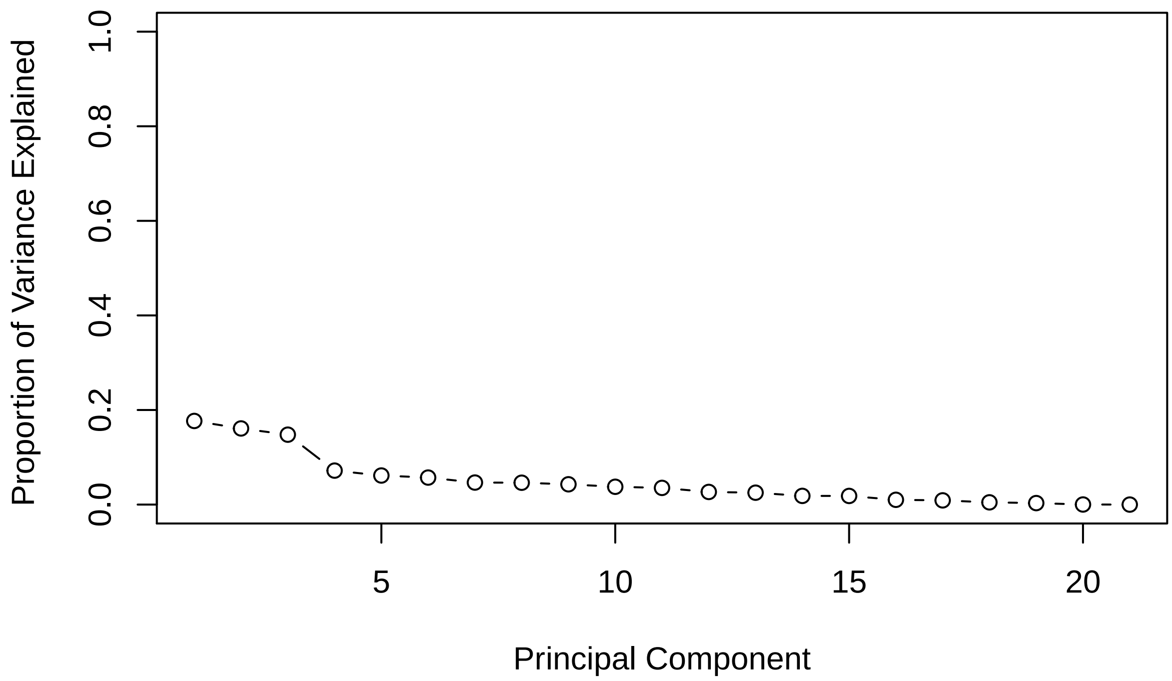
WorkAtHome

Minority

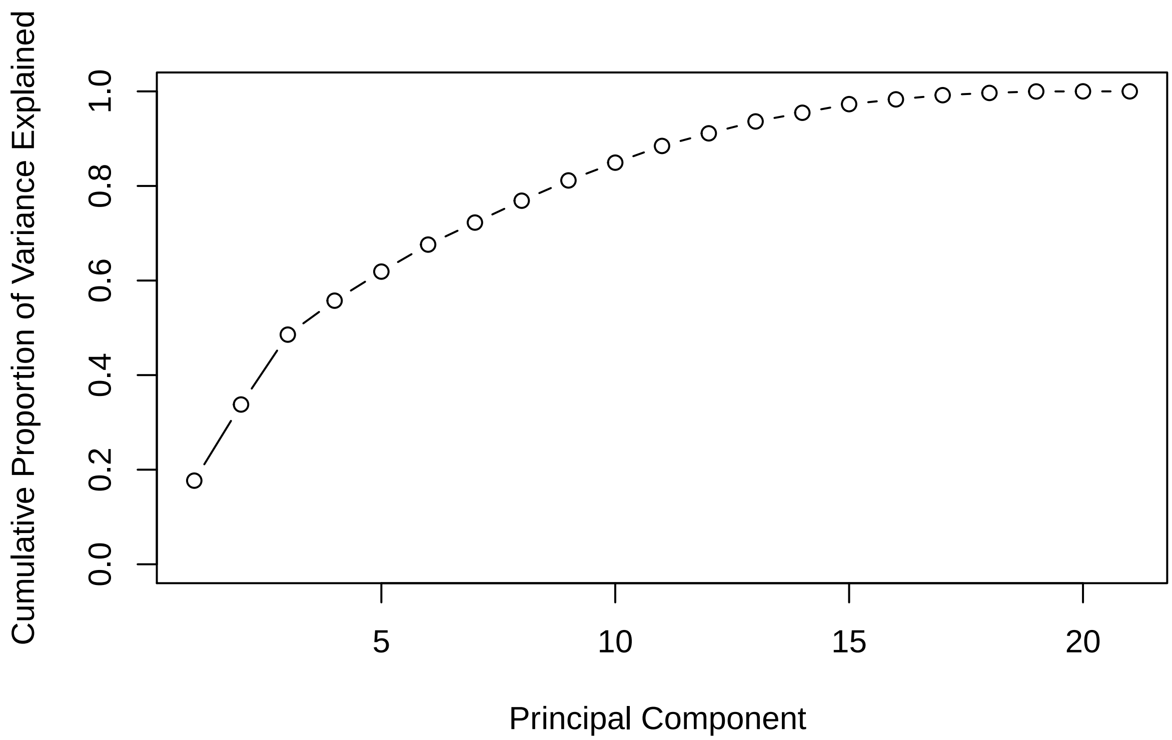
The features Men, White, VotingAgeCitizen, Professional, OtherTransp, WorkAtHome, Employed, SelfEmployed, and FamilyWork have approximately opposite signs to Poverty which implies that those features are negatively linearly correlated with Poverty. So as those features increase, poverty decreases.

12. (2 pts) 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.

```r
# create variance
pr.var = pr.out$sdev^2
# create pve and the cumulative sum of pve + plot
pve = pr.var/sum(pr.var)
cumsum_pve = cumsum(pve)
plot(pve, xlab="Principal Component",
ylab="Proportion of Variance Explained ", ylim=c(0,1),type='b')
```




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




```
# print out the number of pcs needed to cover 90% oc the variance
(length(cumsum_pve[cumsum_pve < 0.90])+1)
```

```
## [1] 12
```

The number of minimum number of PCs needed to capture 90% of the variance is 12

Clustering

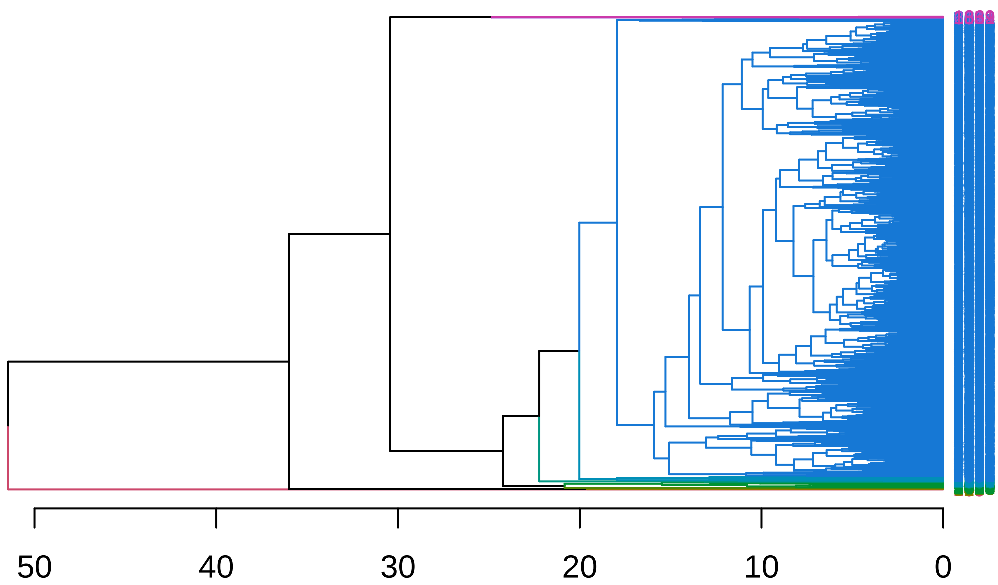
13. (2 pts) With census.clean (with State and County excluded), perform hierarchical clustering with complete linkage. (2 pts) Cut the tree to partition the observations into 10 clusters. (2 pts) Re-run the hierarchical clustering algorithm using the first 2 principal components from pc.county as inputs instead of the original features. (2 pts) Compare the results and comment on your observations. For both approaches investigate the cluster that contains Santa Barbara County. (2 pts) 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.

```r
library(cluster)
library(tidyverse)
set.seed(1)
# With census.clean (with State and County excluded),
# perform hierarchical clustering with complete linkage.

# scale data and exclude State and County
scensus = scale(census.clean[, -c(1,2)], center=TRUE, scale=TRUE)
# gets distance matrix
census.dist = dist(scensus)
# by default hclust uses complete linkage
census.hclust = hclust(census.dist)
# Cut the tree to partition the observations into 10 clusters
clus.census = cutree(census.hclust, 10)

dg_1 <- as.dendrogram(census.hclust)
dg_1 <- color_branches(dg_1, k=10)
dg_1 <- color_labels(dg_1, k=10)
dg_1 <- set(dg_1, "labels_cex", 0.6)
plot(dg_1, horiz=T, main = "Dendrogram colored by 10 clusters for census")
```

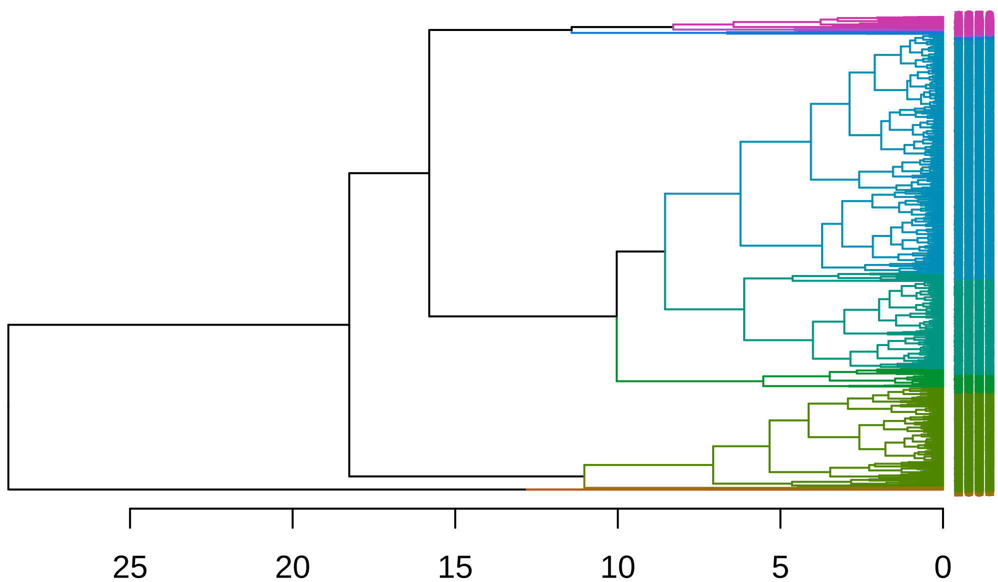
Dendrogram colored by 10 clusters for census




```r
# Re-run the hierarchical clustering algorithm using the
# first 2 principal components from pc.county as inputs
# instead of the original features
# gets distance matrix
county.pc.dist = dist(pc.county)
# by default hclust uses complete linkage
county.pc.hclust = hclust(county.pc.dist)
# Cut the tree to partition the observations into 10 clusters
clus.county = cutree(county.pc.hclust, 10)

dg.1 <- as.dendrogram(county.pc.hclust)
dg.1 <- color_branches(dg.1, k=10)
dg.1 <- color_labels(dg.1, k=10)
dg.1 <- set(dg.1, "labels_cex", 0.6)
plot(dg.1, horiz=T, main = "Dendrogram colored by 10 clusters for pc.county")
```

Dendrogram colored by 10 clusters for pc.county




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

# gives index of SB
which(census.clean$County == "Santa Barbara County")
```

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

```
clus.census[228]
```

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

```
table(clus.census)
```

```
## clus.census
##    1    2    3    4    5    6    7    8    9   10
## 3044    6    1    3   33    1    6   38    6    4
```

```
clus.county[228]
```

```
## [1] 6
```

```
table(clus.county)
```

```
## clus.county
##    1    2    3    4    5    6    7    8    9   10
##  631 1590  123  658   24   76   13    1    5   21
```

Clus.county seems to put Santa Barbara County in a more appropriate cluster because there is a smaller size group in the cluster making it more informative.

Modeling

We are interested in binary classification. 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.

```
# join the two data set and remove na
all <- census.clean %>%
  left_join(education, by = c("State"="State", "County"="County")) %>%
  na.omit
```

14. (4 pts) 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.

We wanted to remove as many features as possible in order to simplify our model so we assessed which features did not necessarilyimpact Poverty. We removed the features Drive, Carpool, Transit, OtherTransp, State, County, MeanCommute, Men, Women, and VotingAgeCitizen because we believe these features are uninformative in regards to questions surrounding the causes of Poverty. For example, the total amount of Men and Women in a county does not give us any specific information about poverty. The gender of a person does not necessarily impact whether or not one experiences poverty.

```
set.seed(123)
all = all %>%
  mutate(Poverty = as.factor(ifelse(Poverty > 20, "1", "0"))) %>%
  select(-Drive,
         -Carpool,
         -Transit,
         -OtherTransp,
         -State,
         -County,
         -MeanCommute,
         -Men,
         -Women,
         - VotingAgeCitizen)

# Partition the dataset into 80% training and 20% test data.
# Make sure to set.seed before the partition.
colnames(all) <- make.names(colnames(all)) # fix colnames for modeling
n <- nrow(all)
idx.tr <- sample.int(n, 0.8*n)
all.tr <- all[idx.tr, ]
all.te <- all[-idx.tr, ]
```

```
# define 10 cross-validation folds:
set.seed(123)
nfold <- 10
folds <- sample(cut(1:nrow(all.tr), breaks=nfold, labels=FALSE))
```

```
# error rate function, the object records is used to record
# the classification performance of each method in the subsequent problems.
```

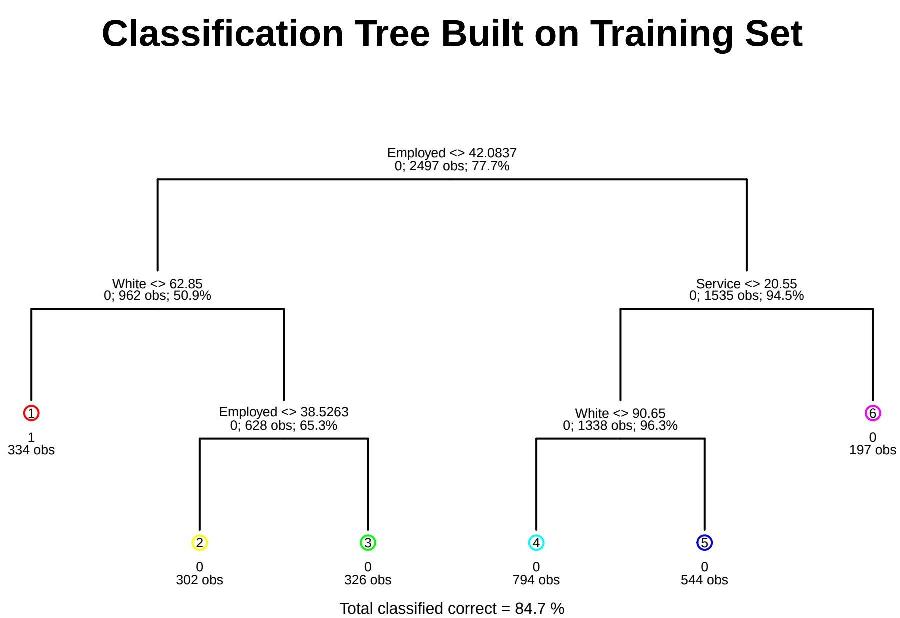
```r
calc_error_rate = function(predicted.value, true.value){
  return(mean(true.value!=predicted.value))
}
records = matrix(NA, nrow=3, ncol=2)
colnames(records) = c("train.error","test.error")
rownames(records) = c("tree","logistic","lasso")
```

Classification

15. Decision tree: (2 pts) train a decision tree by cv.tree(). (2 pts) Prune tree to minimize misclassification error. Be sure to use the folds from above for cross-validation. (2 pts) Visualize the trees before and after pruning. (1 pts) Save training and test errors to records object. (2 pts) Interpret and discuss the results of the decision tree analysis. (2 pts) Use this plot to tell a story about Poverty.

```r
set.seed(123)
#fit model on training set
tree.all = tree(Poverty~., data = all.tr)

#plot tree
draw.tree(tree.all, nodeinfo=TRUE, cex = 0.4)
title("Classification Tree Built on Training Set")
```




```r
# true label of test cases
Poverty.test = all.te$Poverty
# Predict on test set
tree.pred = predict(tree.all, all.te, type="class")
# Test error rate
calc_error_rate(tree.pred, Poverty.test)
```

```
## [1] 0.1648
```

```
# Predict on train set
tree.pred = predict(tree.all, all.tr, type="class")
# Train error rate
calc_error_rate(tree.pred, all.tr$Poverty)
```

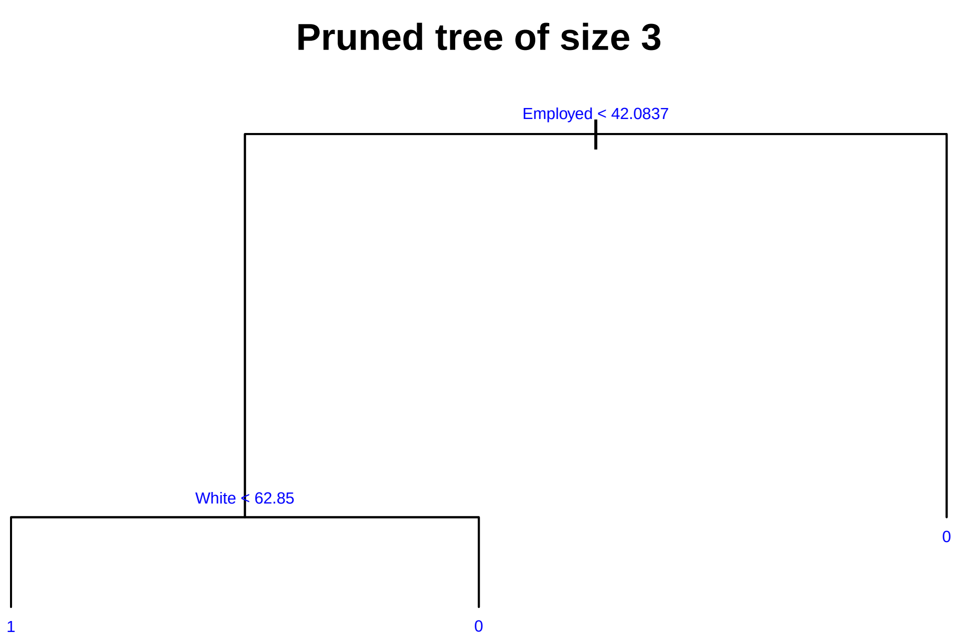
```
## [1] 0.1533841
```

Prune tree

```
set.seed(123)
# do cross validation
cv = cv.tree(tree.all, FUN= prune.misclass, K = folds)
# determine best size for tree
best.cv = min(cv$size[cv$dev == min(cv$dev)])
best.cv
```

```
## [1] 3
```

```
# prune tree
pt.cv = prune.misclass(tree.all, best=best.cv)
# plot the pruned tree
plot(pt.cv)
text(pt.cv, pretty=0, col = "blue", cex = .5)
title("Pruned tree of size 3")
```




```
# Predict on test set
pred.pt.cv.t = predict(pt.cv, all.te, type="class")
#test error rate
(records[1,2] = calc_error_rate(pred.pt.cv.t, all.te$Poverty))
```

```
## [1] 0.1648
```

```
# Predict on train set
pred.pt.cv = predict(pt.cv, all.tr, type="class")
# train error rate
(records[1,1] = calc_error_rate(pred.pt.cv, all.tr$Poverty))
```

```
## [1] 0.1533841
```

The test result for the pruned tree is the same as the unpruned tree. Therefore, we would use the pruned tree since it is simpler without any cost of prediction error rate.

From the graph of the pruned tree, one can see that Poverty is determined mainly by the features employed and white. If a the percentage of employment is greater than 42%, we classify there to be no poverty. However, if it is less than 42% we will look at the white feature. If the white feature is less than 63%, we predict that there will be poverty and vice-versa.

16. (2 pts) Run a logistic regression to predict Poverty in each county. (1 pts) Save training and test errors to records variable. (1 pts) What are the significant variables? (1 pts) Are they consistent with what you saw in decision tree analysis? (2 pts) Interpret the meaning of a couple of the significant coefficients in terms of a unit change in the variables.

```
set.seed(123)
# fit logistic regression model on training set
glm.fit = glm(Poverty ~.,
data=all.tr, family=binomial)
```

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

```
# Specify type="response" to get the estimated probabilities
prob.training = predict(glm.fit, all.tr, type="response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
predPoverty=as.factor(ifelse(prob.training<=0.5, "0", "1"))
# Confusion matrix (training error/accuracy)
accuracy = table(predPoverty, all.tr$Poverty)
(records[2,1] = calc_error_rate(predPoverty, all.tr$Poverty))
```

```
## [1] 0.1357629
```

```
prob.test = predict(glm.fit, all.te, type="response")
predPoverty1=as.factor(ifelse(prob.test<=0.5, "0", "1"))
# Confusion matrix (test error/accuracy)
accuracy1 = table(predPoverty1, all.te$Poverty)
(records[2,2] = calc_error_rate(predPoverty1, all.te$Poverty))
```

```
## [1] 0.1392
```

warning glm.fit: fitted probabilities numerically 0 or 1 occurred 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.

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.

(3 pts) 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 .

(1 pts) What is the optimal value of in cross validation? (1 pts) What are the non-zero coefficients in the LASSO regression for the optimal value of ? (1 pts) How do they compare to the unpenalized logistic regression? (1 pts) Comment on the comparison. (1 pts) Save training and test errors to the records variable.

```
# we must pass in an x (as predictors matrix) as well as a y (response vector), and we do not use the y
x.train = model.matrix(Poverty~., all.tr)[,-1]
y.train = all.tr$Poverty
x.test = model.matrix(Poverty~., all.te)[,-1]
y.test = all.te$Poverty
```

```
set.seed(123)
# cross validation to find best value of lambda
cv.out.lasso <- cv.glmnet(x.train, y.train, alpha = 1, lambda =  seq(1,20) * 1e-5, nfolds = nfold, famil
str_interp('The value for lambda is equal to : ${cv.out.lasso$lambda.min}')
```

```
## [1] "The value for lambda is equal to : 2e-05"
```

```
#create model with best lambda
lasso.model <- glmnet(x.train, y.train, alpha = 1, family = "binomial",
                      lambda = cv.out.lasso$lambda.min)

predict(lasso.model,type="coefficients",s=cv.out.lasso$lambda.min)
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                                                1
## (Intercept)                         1.3694413529
## TotalPop.x                          0.0001235873
## White                               .
## Professional                        0.0943502241
## Service                             0.1155302531
## Office                              0.0617500694
## Production                          0.1097472492
## WorkAtHome                         -0.0806686067
## Employed                           -0.2267838648
## PrivateWork                        -0.0155850478
## SelfEmployed                       -0.0233921317
## FamilyWork                         -0.1050032562
## Minority                            0.0333462850
```

```
## Less.than.a.high.school.diploma..2015.19    -0.0001443340
## High.school.diploma.only..2015.19           -0.0001494181
## Some.college.or.associate.s.degree..2015.19 -0.0003022226
## Bachelor.s.degree.or.higher..2015.19        -0.0001558204
## TotalPop.y                                   .
```

Lasso regression removes totalpop.y and white compared to logistic regression in which it keeps all the features.

```r
# Make prediction on train data
probabilities <- lasso.model %>% predict(newx = x.train, s = cv.out.lasso$lambda.min, type = "response")
predicted.classes <- ifelse(probabilities <= 0.5, "0", "1")
# Model train error
(records[3,1] <- calc_error_rate(predicted.classes, y.train))
```

```
## [1] 0.1361634
```

```r
# Make prediction on train data
probabilities <- lasso.model %>% predict(newx = x.test, s = cv.out.lasso$lambda.min, type = "response")
predicted.classes.t <- ifelse(probabilities <= 0.5, "0", "1")
# Model test error
(records[3,2] <- calc_error_rate(predicted.classes.t, y.test))
```

```
## [1] 0.1408
```

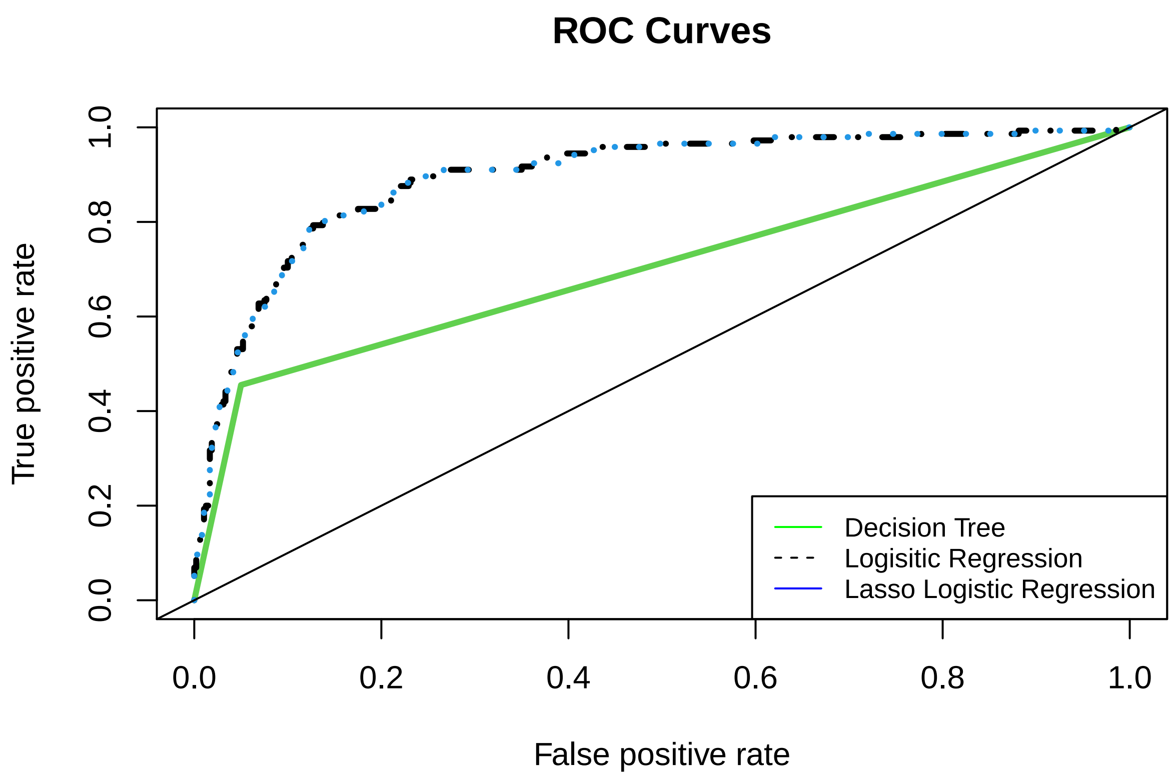
18. (6 pts) Compute ROC curves for the decision tree, logistic regression and LASSO logistic regression using predictions on the test data. Display them on the same plot. (2 pts) Based on your classification results, discuss the pros and cons of the various methods. (2 pts) Are the different classifiers more appropriate for answering different kinds of questions about Poverty?

```r
# Decision Tree ROC
tree.pred.cv = predict(pt.cv, all.te, type="class")
pred = prediction(as.numeric(tree.pred.cv) ,as.numeric(y.test))
tree.perf = performance(pred, measure="tpr", x.measure="fpr")

# Logistic
logistic.pred = predict(glm.fit, all.te, type="response")
pred1 = prediction(as.numeric(logistic.pred), as.numeric(y.test))
log.perf = performance(pred1, measure="tpr", x.measure="fpr")

# Lasso
lasso.pred = predict(lasso.model, x.test, s=cv.out.lasso$lambda.min, type = "response")
pred2 = prediction(as.numeric(lasso.pred), as.numeric(y.test))
lasso.perf = performance(pred2, measure="tpr", x.measure="fpr")
```

```r
plot(tree.perf, col = 3, lwd = 3, main = "ROC Curves")
plot(log.perf, col = 1,lty = 4, lwd = 3, add = TRUE )
plot(lasso.perf, col = 4, lty= 3, lwd = 3, add = TRUE)
legend("bottomright", legend = c("Decision Tree", "Logisitic Regression", "Lasso Logistic Regression"),
abline(0,1)
```




```
records
```

```
##            train.error test.error
## tree         0.1533841     0.1648
## logistic     0.1357629     0.1392
## lasso        0.1361634     0.1408
```

Lasso creates a simpler model by removing features but it has a higher test error than logistic regression. By removing redundant variables, new observations that include those features may possibly contain information that could more accurately predict questions regarding poverty.

Logistic regression has the lowest test error rate, however it assumes linearity between the dependent and independent variables. Linearly separable data is rarely found within the real world.

The tree model has better interpretability than the other models, but it is limited in its variable selection and prediction accuracy.

The tree model would not be useful in answering questions about new data regarding poverty since a small change in the data can cause a big change in the structure of the tree.

19. (9 pts) 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 :

```
rf.model = randomForest(Poverty ~ ., data=all.tr, mtry=3, importance=TRUE)
yhat.rf = predict(rf.model, newdata = all.te)
test.rf.err = mean(yhat.rf != all.te$Poverty)
str_interp("the test error for randomforest is : ${test.rf.err}")
```

```
## [1] "the test error for randomforest is : 0.1344"
```

kNN :

```
YTrain = all.tr$Poverty
XTrain = all.tr %>% select(-Poverty) %>% scale(center = TRUE, scale = TRUE)

YTest = all.te$Poverty
XTest = all.te %>% select(-Poverty) %>% scale(center = TRUE, scale = TRUE)
```

```
# do.chunk() for k-fold Cross-validation
do.chunk <- function(chunkid, folddef, Xdat, Ydat, ...){
  # Get training index
  train = (folddef!=chunkid)

  # Get training set by the above index
  Xtr = Xdat[train,]
  # Get responses in training set
  Ytr = Ydat[train]

  # Get validation set
  Xvl = Xdat[!train,]
  # Get responses in validation set
  Yvl = Ydat[!train]

  # Predict training labels
  predYtr = knn(train=Xtr, test=Xtr, cl=Ytr, ...)
  # Predict validation labels
  predYvl = knn(train=Xtr, test=Xvl, cl=Ytr, ...)

  data.frame(fold = chunkid,
             train.error = mean(predYtr != Ytr), # Training error for each fold
             val.error = mean(predYvl != Yvl)) # Validation error for each fold

}
```

```
# Specify we want a 5-fold CV
nfold = 5

# cut: divides all training observations into 3 intervals;
#      labels = FALSE instructs R to use integers to code different intervals
set.seed(66)
folds = cut(1:nrow(XTrain), breaks=nfold, labels=FALSE) %>% sample()
```

```
# Set error.folds (a vector) to save validation errors in future
error.folds = NULL

# Give possible number of nearest neighbours to be considered
allK = 1:50

# Set seed since do.chunk() contains a random component induced by knn()
set.seed(888)
```

```r
# Loop through different number of neighbors
for (k in allK){
  # Loop through different chunk id
  for (j in seq(5)){
    tmp = do.chunk(chunkid=j, folddef=folds, Xdat=XTrain, Ydat=YTrain, k=k)

    tmp$neighbors = k # Record the last number of neighbor

    error.folds = rbind(error.folds, tmp) # combine results
  }
}
```

```r
# Transform the format of error.folds for further convenience
errors = melt(error.folds, id.vars=c('fold', 'neighbors'), value.name='error')

# Choose the number of neighbors which minimizes validation error
val.error.means = errors %>%
    # Select all rows of validation errors
    filter(variable=='val.error') %>%
    # Group the selected data frame by neighbors
    group_by(neighbors, variable) %>%
    # Calculate CV error rate for each k
    summarise_each(funs(mean), error) %>%
    # Remove existing group
    ungroup() %>%
    filter(error==min(error))
```

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

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
```

```r
# Best number of neighbors
#     if there is a tie, pick larger number of neighbors for simpler model
numneighbor = max(val.error.means$neighbors)
```

```r
set.seed(99)
pred.YTest = knn(train=XTrain, test=XTest, cl=YTrain, k=numneighbor)

# Confusion matrix
conf.matrix = table(predicted=pred.YTest, true=YTest)
```

```r
# Test error rate
test.knn.err <- 1 - sum(diag(conf.matrix)/sum(conf.matrix))

str_interp("the test error for knn is : ${test.knn.err}")
```

```
## [1] "the test error for knn is : 0.1536"
```

AUC

```r
# Compare AUC
# random forest
yhat.rf = predict(rf.model, all.te)
pred3 = prediction(as.numeric(yhat.rf), as.numeric(all.te$Poverty))
rf.perf = performance(pred3, "auc")
auc.rf <- as.numeric(rf.perf@y.values)

# knn
Ytest.pred = knn(train=XTrain, test=XTest, cl=YTrain, k=numneighbor)
pred4 = prediction(as.numeric(Ytest.pred), as.numeric(all.te$Poverty))
knn.perf = performance(pred4, "auc")
auc.knn <- as.numeric(knn.perf@y.values)

# Decision Tree
tree.pred.cv = predict(pt.cv, all.te, type="class")
pred = prediction(as.numeric(tree.pred.cv) ,as.numeric(y.test))
tree.perf = performance(pred, "auc")
auc.tree <- as.numeric(tree.perf@y.values)

# Logistic
logistic.pred = predict(glm.fit, all.te, type="response")
pred1 = prediction(as.numeric(logistic.pred), as.numeric(y.test))
log.perf = performance(pred1, "auc")
auc.log <- as.numeric(log.perf@y.values)

# Lasso
lasso.pred = predict(lasso.model, x.test, s=cv.out.lasso$lambda.min, type = "response")
pred2 = prediction(as.numeric(lasso.pred), as.numeric(y.test))
lasso.perf = performance(pred2, "auc")
auc.lasso <- as.numeric(lasso.perf@y.values)

str_interp('AUC of rf : ${auc.rf}')
```

```
## [1] "AUC of rf : 0.772916666666667"
```

```r
str_interp('AUC of knn : ${auc.knn}')
```

```
## [1] "AUC of knn : 0.714691091954023"
```

```
str_interp('AUC of tree : ${auc.tree}')
```

```
## [1] "AUC of tree : 0.702586206896552"
```

```
str_interp('AUC of logistic : ${auc.log}')
```

```
## [1] "AUC of logistic : 0.895086206896553"
```

```
str_interp('AUC of lasso : ${auc.lasso}')
```

```
## [1] "AUC of lasso : 0.895244252873566"
```

When we are comparing the area under the curves we see that the random forest model and knn is better than the tree model however, it is worse than logistic and lasso.

20. 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?

Linear Regression

```
all1 <- census.clean %>%
  left_join(education, by = c("State"="State", "County"="County")) %>%
  na.omit
all1 <- all1 %>% select(-Drive,
                        -Carpool,
                        -Transit,
                        -OtherTransp,
                        -State,
                        -County,
                        -MeanCommute,
                        -Men,
                        -Women,
                        -VotingAgeCitizen)
```

```
set.seed(123)
colnames(all) <- make.names(colnames(all)) # fix colnames for modeling
n <- nrow(all)
idx.tr <- sample.int(n, 0.8*n)
all.tr1 <- all1[idx.tr, ]
all.te1 <- all1[-idx.tr, ]

YTrain1 = all.tr1$Poverty
XTrain1 = all.tr1 %>% select(-Poverty) %>% scale(center = TRUE, scale = TRUE)

YTest1 = all.te1$Poverty
XTest1 = all.te1 %>% select(-Poverty) %>% scale(center = TRUE, scale = TRUE)
```

We can apply stepwise selection to see which features are significant to our linear regression model.

```r
int_only1 = lm(Poverty ~ 1, data = all.tr1)
tot_1 = lm(Poverty ~ . , data = all.tr1)
step(int_only1, direction = 'both', scope = formula(tot_1))
```

```
## Start:  AIC=9361.3
## Poverty ~ 1
##
##                                                Df Sum of Sq    RSS    AIC
## + Employed                                      1     53364  52622 7615.0
## + Minority                                      1     22757  83229 8759.8
## + White                                         1     22101  83885 8779.4
## + Professional                                  1     16446  89540 8942.2
## + Service                                       1     13143  92843 9032.7
## + WorkAtHome                                    1      6440  99546 9206.8
## + PrivateWork                                   1      5124 100862 9239.6
## + Production                                    1      4079 101907 9265.3
## + SelfEmployed                                  1      2491 103495 9303.9
## + `Bachelor's degree or higher, 2015-19`        1      1104 104882 9337.2
## + `Some college or associate's degree, 2015-19` 1       683 105303 9347.1
## + TotalPop.y                                    1       572 105414 9349.8
## + TotalPop.x                                    1       490 105496 9351.7
## + `High school diploma only, 2015-19`           1       445 105541 9352.8
## + Office                                        1       207 105779 9358.4
## <none>                                                      105986 9361.3
## + FamilyWork                                    1        77 105908 9361.5
## + `Less than a high school diploma, 2015-19`    1         0 105986 9363.3
##
## Step:  AIC=7614.96
## Poverty ~ Employed
##
##                                                Df Sum of Sq    RSS    AIC
## + Minority                                      1      6705  45917 7276.6
## + White                                         1      6515  46107 7286.9
## + Service                                       1      1346  51276 7552.3
## + SelfEmployed                                  1       575  52047 7589.5
## + WorkAtHome                                    1       505  52117 7592.9
## + Production                                    1       333  52289 7601.1
## + `Less than a high school diploma, 2015-19`    1       282  52340 7603.5
## + Professional                                  1       189  52433 7608.0
## + TotalPop.x                                    1       188  52434 7608.0
## + Office                                        1       171  52451 7608.8
## + TotalPop.y                                    1       155  52467 7609.6
## + PrivateWork                                   1       147  52475 7610.0
## + `High school diploma only, 2015-19`           1       146  52476 7610.0
## + `Bachelor's degree or higher, 2015-19`        1       136  52486 7610.5
## + `Some college or associate's degree, 2015-19` 1        96  52526 7612.4
## <none>                                                       52622 7615.0
## + FamilyWork                                    1         5  52617 7616.7
## - Employed                                      1     53364 105986 9361.3
##
## Step:  AIC=7276.65
## Poverty ~ Employed + Minority
##
```

```
##                                               Df Sum of Sq   RSS    AIC
## + Production                                   1      1204 44714 7212.3
## + Service                                      1       517 45400 7250.4
## + Professional                                 1       496 45421 7251.5
## + Office                                       1       345 45573 7259.8
## + `Some college or associate's degree, 2015-19`  1       233 45685 7266.0
## + `Bachelor's degree or higher, 2015-19`       1       189 45728 7268.3
## + WorkAtHome                                   1       175 45743 7269.1
## + TotalPop.y                                   1       165 45753 7269.7
## + `High school diploma only, 2015-19`          1       155 45762 7270.2
## + TotalPop.x                                   1       140 45777 7271.0
## + White                                        1        83 45835 7274.1
## + SelfEmployed                                 1        50 45867 7275.9
## + FamilyWork                                   1        48 45869 7276.0
## <none>                                                     45917 7276.6
## + `Less than a high school diploma, 2015-19`   1        31 45886 7276.9
## + PrivateWork                                  1        31 45886 7277.0
## - Minority                                     1      6705 52622 7615.0
## - Employed                                     1     37312 83229 8759.8
##
## Step:  AIC=7212.31
## Poverty ~ Employed + Minority + Production
##
##                                               Df Sum of Sq   RSS    AIC
## + Service                                      1      1213 43500 7145.6
## + PrivateWork                                  1       494 44220 7186.6
## + Office                                       1       133 44580 7206.8
## + `Some college or associate's degree, 2015-19`  1       129 44584 7207.1
## + FamilyWork                                   1       125 44588 7207.3
## + `High school diploma only, 2015-19`          1        87 44627 7209.5
## + TotalPop.y                                   1        82 44632 7209.7
## + `Bachelor's degree or higher, 2015-19`       1        80 44634 7209.9
## + TotalPop.x                                   1        67 44646 7210.6
## <none>                                                     44714 7212.3
## + `Less than a high school diploma, 2015-19`   1        16 44698 7213.4
## + White                                        1        11 44702 7213.7
## + WorkAtHome                                   1         1 44713 7214.3
## + Professional                                 1         0 44713 7214.3
## + SelfEmployed                                 1         0 44714 7214.3
## - Production                                   1      1204 45917 7276.6
## - Minority                                     1      7575 52289 7601.1
## - Employed                                     1     31799 76513 8551.7
##
## Step:  AIC=7145.61
## Poverty ~ Employed + Minority + Production + Service
##
##                                               Df Sum of Sq   RSS    AIC
## + PrivateWork                                  1     557.1 42943 7115.4
## + Professional                                 1     394.8 43105 7124.8
## + FamilyWork                                   1     175.5 43325 7137.5
## + `Some college or associate's degree, 2015-19`  1     112.7 43387 7141.1
## + `High school diploma only, 2015-19`          1      84.9 43415 7142.7
## + Office                                       1      65.4 43435 7143.9
## + TotalPop.y                                   1      64.3 43436 7143.9
```

```
## + TotalPop.x                                  1      51.4 43449 7144.7
## + `Bachelor's degree or higher, 2015-19`      1      44.6 43456 7145.1
## <none>                                                     43500 7145.6
## + SelfEmployed                                1      31.8 43468 7145.8
## + White                                       1      21.5 43479 7146.4
## + WorkAtHome                                  1      19.4 43481 7146.5
## + `Less than a high school diploma, 2015-19`  1      16.5 43484 7146.7
## - Service                                     1    1213.4 44714 7212.3
## - Production                                  1    1899.9 45400 7250.4
## - Minority                                    1    6757.8 50258 7504.2
## - Employed                                    1   23697.1 67197 8229.5
##
## Step:  AIC=7115.43
## Poverty ~ Employed + Minority + Production + Service + PrivateWork
##
##                                              Df Sum of Sq   RSS    AIC
## + Professional                                1     306.2 42637 7099.6
## + SelfEmployed                                1     151.8 42791 7108.6
## + WorkAtHome                                  1      47.7 42895 7114.7
## + FamilyWork                                  1      37.3 42906 7115.3
## <none>                                                     42943 7115.4
## + White                                       1      31.5 42911 7115.6
## + `Some college or associate's degree, 2015-19` 1    16.7 42926 7116.5
## + Office                                      1      13.6 42929 7116.6
## + `High school diploma only, 2015-19`         1       4.7 42938 7117.2
## + TotalPop.y                                  1       2.9 42940 7117.3
## + TotalPop.x                                  1       0.8 42942 7117.4
## + `Bachelor's degree or higher, 2015-19`      1       0.3 42943 7117.4
## + `Less than a high school diploma, 2015-19`  1       0.1 42943 7117.4
## - PrivateWork                                 1     557.1 43500 7145.6
## - Service                                     1    1276.7 44220 7186.6
## - Production                                  1    2445.0 45388 7251.7
## - Minority                                    1    6748.9 49692 7477.9
## - Employed                                    1   18876.3 61819 8023.2
##
## Step:  AIC=7099.56
## Poverty ~ Employed + Minority + Production + Service + PrivateWork +
##     Professional
##
##                                              Df Sum of Sq   RSS    AIC
## + Office                                      1      90.5 42546 7096.2
## + SelfEmployed                                1      88.6 42548 7096.4
## + WorkAtHome                                  1      64.4 42572 7097.8
## + FamilyWork                                  1      54.3 42583 7098.4
## + `Some college or associate's degree, 2015-19` 1    51.8 42585 7098.5
## + White                                       1      48.6 42588 7098.7
## <none>                                                     42637 7099.6
## + `Bachelor's degree or higher, 2015-19`      1      27.4 42609 7100.0
## + TotalPop.y                                  1      26.4 42610 7100.0
## + `High school diploma only, 2015-19`         1      25.4 42611 7100.1
## + TotalPop.x                                  1      18.2 42619 7100.5
## + `Less than a high school diploma, 2015-19`  1       1.7 42635 7101.5
## - Professional                                1     306.2 42943 7115.4
## - PrivateWork                                 1     468.6 43105 7124.8
```

```
## - Service                                      1    1578.7 44216 7188.3
## - Production                                   1    2276.1 44913 7227.4
## - Minority                                     1    6653.6 49290 7459.6
## - Employed                                     1   18062.3 60699 7979.5
##
## Step:  AIC=7096.25
## Poverty ~ Employed + Minority + Production + Service + PrivateWork +
##     Professional + Office
##
##                                               Df Sum of Sq   RSS    AIC
## + FamilyWork                                   1      70.8 42476 7094.1
## + White                                        1      67.5 42479 7094.3
## + `Some college or associate's degree, 2015-19`  1      66.9 42479 7094.3
## + SelfEmployed                                 1      56.3 42490 7094.9
## + WorkAtHome                                   1      43.1 42503 7095.7
## + `High school diploma only, 2015-19`          1      37.0 42509 7096.1
## + TotalPop.y                                   1      34.3 42512 7096.2
## <none>                                                      42546 7096.2
## + `Bachelor's degree or higher, 2015-19`       1      32.0 42514 7096.4
## + TotalPop.x                                   1      24.8 42521 7096.8
## + `Less than a high school diploma, 2015-19`   1       2.8 42543 7098.1
## - Office                                       1      90.5 42637 7099.6
## - Professional                                 1     383.2 42929 7116.6
## - PrivateWork                                  1     555.6 43102 7126.6
## - Service                                      1    1666.6 44213 7190.2
## - Production                                   1    2120.1 44666 7215.7
## - Minority                                     1    6545.0 49091 7451.5
## - Employed                                     1   17464.3 60011 7953.0
##
## Step:  AIC=7094.09
## Poverty ~ Employed + Minority + Production + Service + PrivateWork +
##     Professional + Office + FamilyWork
##
##                                               Df Sum of Sq   RSS    AIC
## + WorkAtHome                                   1      79.0 42397 7091.4
## + SelfEmployed                                 1      77.0 42399 7091.6
## + `Some college or associate's degree, 2015-19`  1      72.2 42403 7091.8
## + White                                        1      63.9 42412 7092.3
## + `High school diploma only, 2015-19`          1      41.3 42434 7093.7
## + TotalPop.y                                   1      38.0 42438 7093.9
## + `Bachelor's degree or higher, 2015-19`       1      35.1 42440 7094.0
## <none>                                                      42476 7094.1
## + TotalPop.x                                   1      27.9 42448 7094.4
## + `Less than a high school diploma, 2015-19`   1       3.9 42472 7095.9
## - FamilyWork                                   1      70.8 42546 7096.2
## - Office                                       1     107.0 42583 7098.4
## - Professional                                 1     412.8 42888 7116.2
## - PrivateWork                                  1     433.4 42909 7117.4
## - Service                                      1    1719.9 44195 7191.2
## - Production                                   1    2172.8 44648 7216.7
## - Minority                                     1    6612.3 49088 7453.4
## - Employed                                     1   17535.0 60011 7955.0
##
## Step:  AIC=7091.45
```

```
## Poverty ~ Employed + Minority + Production + Service + PrivateWork +
##     Professional + Office + FamilyWork + WorkAtHome
##
##                                                Df Sum of Sq   RSS    AIC
## + White                                         1      73.6 42323 7089.1
## + `Some college or associate's degree, 2015-19`  1      64.4 42332 7089.7
## + `High school diploma only, 2015-19`           1      36.0 42361 7091.3
## <none>                                                      42397 7091.4
## + TotalPop.y                                    1      33.1 42363 7091.5
## + `Bachelor's degree or higher, 2015-19`        1      31.2 42365 7091.6
## + SelfEmployed                                  1      25.7 42371 7091.9
## + TotalPop.x                                    1      24.0 42373 7092.0
## + `Less than a high school diploma, 2015-19`    1       2.5 42394 7093.3
## - WorkAtHome                                    1      79.0 42476 7094.1
## - Office                                        1      80.7 42477 7094.2
## - FamilyWork                                    1     106.7 42503 7095.7
## - Professional                                  1     425.3 42822 7114.4
## - PrivateWork                                   1     506.1 42903 7119.1
## - Service                                       1    1605.1 44002 7182.2
## - Production                                    1    1989.8 44386 7204.0
## - Minority                                      1    6204.3 48601 7430.5
## - Employed                                      1   16694.6 59091 7918.5
##
## Step:  AIC=7089.11
## Poverty ~ Employed + Minority + Production + Service + PrivateWork +
##     Professional + Office + FamilyWork + WorkAtHome + White
##
##                                                Df Sum of Sq   RSS    AIC
## + `Some college or associate's degree, 2015-19`  1      54.2 42269 7087.9
## + SelfEmployed                                  1      35.0 42288 7089.0
## <none>                                                      42323 7089.1
## + `High school diploma only, 2015-19`           1      29.8 42293 7089.3
## + TotalPop.y                                    1      27.2 42296 7089.5
## + `Bachelor's degree or higher, 2015-19`        1      25.5 42297 7089.6
## + TotalPop.x                                    1      19.1 42304 7090.0
## + `Less than a high school diploma, 2015-19`    1       1.8 42321 7091.0
## - White                                         1      73.6 42397 7091.4
## - WorkAtHome                                    1      88.7 42412 7092.3
## - Office                                        1      97.6 42420 7092.9
## - FamilyWork                                    1     104.9 42428 7093.3
## - Minority                                      1     252.4 42575 7102.0
## - Professional                                  1     459.8 42783 7114.1
## - PrivateWork                                   1     542.1 42865 7118.9
## - Service                                       1    1654.4 43977 7182.9
## - Production                                    1    2004.8 44328 7202.7
## - Employed                                      1   16612.8 58936 7913.9
##
## Step:  AIC=7087.91
## Poverty ~ Employed + Minority + Production + Service + PrivateWork +
##     Professional + Office + FamilyWork + WorkAtHome + White +
##     `Some college or associate's degree, 2015-19`
##
##                                                Df Sum of Sq   RSS    AIC
## + TotalPop.x                                    1     350.4 41918 7069.1
```

```
## + `Less than a high school diploma, 2015-19`     1     261.0 42008 7074.4
## + TotalPop.y                                   1     168.9 42100 7079.9
## + `High school diploma only, 2015-19`          1      62.8 42206 7086.2
## <none>                                                     42269 7087.9
## + SelfEmployed                                 1      27.7 42241 7088.3
## + `Bachelor's degree or higher, 2015-19`       1      26.5 42242 7088.3
## - `Some college or associate's degree, 2015-19`  1      54.2 42323 7089.1
## - White                                        1      63.4 42332 7089.7
## - WorkAtHome                                   1      80.3 42349 7090.6
## - FamilyWork                                   1     108.3 42377 7092.3
## - Office                                       1     111.7 42380 7092.5
## - Minority                                     1     239.8 42508 7100.0
## - PrivateWork                                  1     433.0 42702 7111.4
## - Professional                                 1     503.4 42772 7115.5
## - Service                                      1    1692.4 43961 7183.9
## - Production                                   1    2016.2 44285 7202.3
## - Employed                                     1   16553.6 58822 7911.1
##
## Step:  AIC=7069.12
## Poverty ~ Employed + Minority + Production + Service + PrivateWork +
##     Professional + Office + FamilyWork + WorkAtHome + White +
##     `Some college or associate's degree, 2015-19` + TotalPop.x
##
##                                                Df Sum of Sq   RSS    AIC
## + TotalPop.y                                   1     300.5 41618 7053.2
## + `Bachelor's degree or higher, 2015-19`       1     151.0 41767 7062.1
## + `Less than a high school diploma, 2015-19`     1      38.4 41880 7068.8
## + SelfEmployed                                 1      36.0 41882 7069.0
## <none>                                                     41918 7069.1
## - White                                        1      34.5 41953 7069.2
## - WorkAtHome                                   1      59.4 41978 7070.7
## + `High school diploma only, 2015-19`          1       0.0 41918 7071.1
## - FamilyWork                                   1     103.7 42022 7073.3
## - Minority                                     1     174.4 42093 7077.5
## - Office                                       1     177.3 42096 7077.7
## - TotalPop.x                                   1     350.4 42269 7087.9
## - `Some college or associate's degree, 2015-19`  1     385.5 42304 7090.0
## - PrivateWork                                  1     405.9 42324 7091.2
## - Professional                                 1     442.5 42361 7093.3
## - Service                                      1    1734.7 43653 7168.4
## - Production                                   1    1976.9 43895 7182.2
## - Employed                                     1   16549.4 58468 7898.0
##
## Step:  AIC=7053.15
## Poverty ~ Employed + Minority + Production + Service + PrivateWork +
##     Professional + Office + FamilyWork + WorkAtHome + White +
##     `Some college or associate's degree, 2015-19` + TotalPop.x +
##     TotalPop.y
##
##                                                Df Sum of Sq   RSS    AIC
## - White                                        1      31.4 41649 7053.0
## <none>                                                     41618 7053.2
## + SelfEmployed                                 1      24.8 41593 7053.7
## - WorkAtHome                                   1      45.8 41664 7053.9
```

```
## + `Bachelor's degree or higher, 2015-19`        1       0.9 41617 7055.1
## + `High school diploma only, 2015-19`          1       0.6 41617 7055.1
## + `Less than a high school diploma, 2015-19`   1       0.3 41618 7055.1
## - FamilyWork                                   1     102.6 41720 7057.3
## - Minority                                     1     163.8 41782 7061.0
## - Office                                       1     170.3 41788 7061.3
## - TotalPop.y                                   1     300.5 41918 7069.1
## - PrivateWork                                  1     325.5 41943 7070.6
## - `Some college or associate's degree, 2015-19` 1     328.1 41946 7070.8
## - TotalPop.x                                   1     482.0 42100 7079.9
## - Professional                                 1     504.8 42123 7081.3
## - Service                                      1    1784.9 43403 7156.0
## - Production                                   1    1945.5 43563 7165.2
## - Employed                                     1   16786.7 58404 7897.3
##
## Step:  AIC=7053.04
## Poverty ~ Employed + Minority + Production + Service + PrivateWork +
##     Professional + Office + FamilyWork + WorkAtHome + `Some college or associate's degree, 2015-19` +
##     TotalPop.x + TotalPop.y
##
##                                               Df Sum of Sq   RSS    AIC
## <none>                                                     41649 7053.0
## + White                                        1      31.4 41618 7053.2
## - WorkAtHome                                   1      40.4 41690 7053.5
## + SelfEmployed                                 1      19.3 41630 7053.9
## + `Bachelor's degree or higher, 2015-19`        1       1.9 41647 7054.9
## + `Less than a high school diploma, 2015-19`   1       1.1 41648 7055.0
## + `High school diploma only, 2015-19`          1       0.8 41648 7055.0
## - FamilyWork                                   1     103.8 41753 7057.3
## - Office                                       1     159.4 41809 7060.6
## - TotalPop.y                                   1     303.6 41953 7069.2
## - PrivateWork                                  1     305.4 41955 7069.3
## - `Some college or associate's degree, 2015-19` 1     357.6 42007 7072.4
## - Professional                                 1     485.1 42134 7079.9
## - TotalPop.x                                   1     494.5 42144 7080.5
## - Service                                      1    1758.7 43408 7154.3
## - Production                                   1    1935.7 43585 7164.5
## - Minority                                     1    5534.4 47184 7362.6
## - Employed                                     1   16842.3 58491 7899.0
```

```
##
## Call:
## lm(formula = Poverty ~ Employed + Minority + Production + Service +
##     PrivateWork + Professional + Office + FamilyWork + WorkAtHome +
##     `Some college or associate's degree, 2015-19` + TotalPop.x +
##     TotalPop.y, data = all.tr1)
##
## Coefficients:
##                                  (Intercept)
##                                    2.680e+01
##                                     Employed
##                                   -5.479e-01
##                                     Minority
##                                    8.691e-02
```

```
##                                     Production
##                                      2.949e-01
##                                        Service
##                                      3.115e-01
##                                    PrivateWork
##                                     -6.748e-02
##                                   Professional
##                                      1.279e-01
##                                         Office
##                                      1.065e-01
##                                     FamilyWork
##                                      4.981e-01
##                                      WorkAtHome
##                                     -5.408e-02
## `Some college or associate's degree, 2015-19`
##                                     -4.607e-05
##                                     TotalPop.x
##                                      3.421e-05
##                                     TotalPop.y
##                                     -3.857e-05
```

```
linear.model <- lm(formula = Poverty ~ Employed + Minority + Production + Service +
    PrivateWork + Professional + Office + FamilyWork + WorkAtHome +
    `Some college or associate's degree, 2015-19` + TotalPop.x +
    TotalPop.y, data = all.tr1)
sm <- summary(linear.model)
sm
```

```
##
## Call:
## lm(formula = Poverty ~ Employed + Minority + Production + Service +
##     PrivateWork + Professional + Office + FamilyWork + WorkAtHome +
##     `Some college or associate's degree, 2015-19` + TotalPop.x +
##     TotalPop.y, data = all.tr1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -23.8848  -2.3161  -0.1606   2.0793  20.3728
##
## Coefficients:
##                                                 Estimate Std. Error t value
## (Intercept)                                    2.680e+01  2.079e+00  12.891
## Employed                                      -5.479e-01  1.729e-02 -31.694
## Minority                                       8.691e-02  4.783e-03  18.168
## Production                                     2.949e-01  2.745e-02  10.745
## Service                                        3.115e-01  3.042e-02  10.242
## PrivateWork                                   -6.748e-02  1.581e-02  -4.268
## Professional                                   1.279e-01  2.377e-02   5.379
## Office                                         1.065e-01  3.453e-02   3.083
## FamilyWork                                     4.981e-01  2.002e-01   2.489
## WorkAtHome                                    -5.408e-02  3.483e-02  -1.552
## `Some college or associate's degree, 2015-19` -4.607e-05  9.975e-06  -4.618
## TotalPop.x                                     3.421e-05  6.300e-06   5.431
## TotalPop.y                                    -3.857e-05  9.064e-06  -4.255
```

```
##                                                Pr(>|t|)
## (Intercept)                                    < 2e-16 ***
## Employed                                       < 2e-16 ***
## Minority                                       < 2e-16 ***
## Production                                     < 2e-16 ***
## Service                                        < 2e-16 ***
## PrivateWork                                   2.05e-05 ***
## Professional                                  8.21e-08 ***
## Office                                         0.00207 **
## FamilyWork                                     0.01289 *
## WorkAtHome                                     0.12067
## `Some college or associate's degree, 2015-19` 4.07e-06 ***
## TotalPop.x                                    6.16e-08 ***
## TotalPop.y                                    2.16e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.095 on 2484 degrees of freedom
## Multiple R-squared:  0.607,  Adjusted R-squared:  0.6051
## F-statistic: 319.8 on 12 and 2484 DF,  p-value: < 2.2e-16
```

```r
lm.pred <- predict(linear.model, all.te1, type="response")
lm.pred
```

```
##         1         2         3         4         5         6         7         8
## 25.834127 24.318004 23.934099 16.390849 17.176013 16.159275 28.134310 33.042349
##         9        10        11        12        13        14        15        16
## 18.912792 21.664520 14.014736 20.869485 22.207713 28.567407 23.902209  2.975898
##        17        18        19        20        21        22        23        24
## 22.297075 26.539048 16.998687 17.270545 16.495343 14.650571 17.662655 13.095131
##        25        26        27        28        29        30        31        32
## 16.120310 16.010173 17.264215 18.521246 18.370855 19.099910 16.492523 18.002579
##        33        34        35        36        37        38        39        40
## 19.043728 16.759288 18.725060 10.503516 17.596473 13.153875 10.542541 13.785553
##        41        42        43        44        45        46        47        48
## 18.596120 18.619989 20.260479 21.822452  9.310991 23.202770  6.358799 12.799922
##        49        50        51        52        53        54        55        56
##  9.352899 16.091778 10.992468 18.229346 11.150555 12.874415 13.169749 12.349713
##        57        58        59        60        61        62        63        64
## 16.490958 13.570217 19.154254 22.167833 12.104495 19.098871 15.407674 22.241460
##        65        66        67        68        69        70        71        72
## 17.974414 17.132728 24.311547 19.215027 11.838468  6.665194 11.724338 16.831728
##        73        74        75        76        77        78        79        80
## 25.983830 30.740830 20.616938 15.797196 20.184992 18.711336 15.358165 18.551538
##        81        82        83        84        85        86        87        88
## 17.725990 23.386169 30.086197 22.060499  9.547886 18.239825 18.529028 20.226999
##        89        90        91        92        93        94        95        96
## 12.890775 15.563548 17.045970 16.271435 23.008058 15.791386 15.856515 13.933187
##        97        98        99       100       101       102       103       104
## 17.331515 14.563032 25.260417 19.662555 12.723193 18.917837 17.935059 22.921269
##       105       106       107       108       109       110       111       112
## 20.907862 24.422957 23.899467 29.102400 23.933748 19.149294 19.612414 15.633115
##       113       114       115       116       117       118       119       120
## 20.050711 16.188888 28.553285 18.973658 17.043110 15.566404 18.125807 14.603077
```

```
##       121       122       123       124       125       126       127       128
## 17.258912 15.311007 12.011548 13.341703 17.795939 18.685070 16.746764 16.453991
##       129       130       131       132       133       134       135       136
## 15.884337 13.452439 11.001001 14.112681 12.995182 10.332980 13.477692 16.669066
##       137       138       139       140       141       142       143       144
## 12.899974 14.305076 19.565564 16.300808 10.484487 18.120381 13.528990 14.747298
##       145       146       147       148       149       150       151       152
## 24.976326 14.581253 19.217886 15.113093 13.432317 14.242168 13.373332 12.614812
##       153       154       155       156       157       158       159       160
##  9.642878 16.092005 12.285468 13.241269 14.270288 17.681375 11.223861 13.393081
##       161       162       163       164       165       166       167       168
## 17.814060 12.486872 17.696832 16.731110 12.108590  9.891877 11.936408 10.063373
##       169       170       171       172       173       174       175       176
##  9.230873 10.932760 12.933579  6.899176 13.342971 12.176307  9.685290  9.397587
##       177       178       179       180       181       182       183       184
## 15.094098 10.998667 12.957418 12.410436  9.862443 15.416402 14.039635  5.774241
##       185       186       187       188       189       190       191       192
## 15.656054 16.699890 17.149857 10.528777 14.024752 14.677380 10.364662 10.816519
##       193       194       195       196       197       198       199       200
## 15.741453 10.388350 15.607452 11.499004 12.311983 11.787584  9.770014 11.783116
##       201       202       203       204       205       206       207       208
## 17.449787 10.156141 10.110956 12.661364  4.875566 10.487854 13.306978 16.905831
##       209       210       211       212       213       214       215       216
## 17.193554 17.326208 13.967120 14.670227 11.833445 15.254043 16.846511 16.843210
##       217       218       219       220       221       222       223       224
## 20.669978 22.998762 14.352417 17.231251 23.029076 19.810592 18.176526 21.614118
##       225       226       227       228       229       230       231       232
## 22.413947 12.907938 18.178197 22.523796 23.323062 15.726942 20.619377 23.060363
##       233       234       235       236       237       238       239       240
## 33.206533 19.175935 28.894993 18.911079 20.015208 19.984963 16.649864 26.751958
##       241       242       243       244       245       246       247       248
## 17.508445 15.468661 11.445726  8.726225 13.994082 10.433883 11.108387 14.289669
##       249       250       251       252       253       254       255       256
##  9.986843 12.414476 14.962017  9.603074 16.105787 10.572448 21.931652 12.747044
##       257       258       259       260       261       262       263       264
## 16.685821 15.055212 12.429809 17.663041 12.667610 24.846422  8.331397 16.062043
##       265       266       267       268       269       270       271       272
## 21.369096 17.209387 15.723445 21.400385 17.621820 15.888199 11.169046  8.522081
##       273       274       275       276       277       278       279       280
##  6.073893 15.872791  8.137016  9.361831 10.725954 12.604583 14.189009 10.433425
##       281       282       283       284       285       286       287       288
## 22.426852 12.580871 11.019066 12.213240 11.004874 10.791481  9.701522 14.302655
##       289       290       291       292       293       294       295       296
## 11.662309 22.915260 23.421293 20.858406 19.587775 23.279022 22.398909 36.642860
##       297       298       299       300       301       302       303       304
## 36.668743 14.051469 19.164472 30.149974 20.097508 20.524578 29.050581 18.422728
##       305       306       307       308       309       310       311       312
## 30.553987 20.832357 17.027895 12.947491 19.751004 11.018007 15.080317 20.484060
##       313       314       315       316       317       318       319       320
## 13.002771 15.578177 15.120504 16.515841 16.211864 21.308007  8.455602 18.651049
##       321       322       323       324       325       326       327       328
## 10.677900 20.686604 13.331118 20.636591 16.355118 16.843223  7.905370 16.378255
##       329       330       331       332       333       334       335       336
##  7.294757 10.018025  9.618298 20.229506 15.753523 10.619405 12.041119 12.271288
```

```
##       337       338       339       340       341       342       343       344
##  9.448566  9.661231  9.619644 10.475546  8.096874  8.955060  7.559422 11.076192
##       345       346       347       348       349       350       351       352
## 13.664376 11.087846  9.178991 11.200089 15.778864  9.012046 13.496615  7.106652
##       353       354       355       356       357       358       359       360
##  8.471736 12.729446  7.904266 16.236209 10.318140 24.120249 28.159548 17.775297
##       361       362       363       364       365       366       367       368
## 20.589430 25.038153 11.760412 16.816562 16.459612 11.961495 12.195721 11.619375
##       369       370       371       372       373       374       375       376
## 13.955515 15.785528 10.580851 15.679638 16.057429 11.242305 12.834207 16.475876
##       377       378       379       380       381       382       383       384
## 16.664434 24.070651 10.863251 16.417810 14.779869 18.477784 22.975038  9.381670
##       385       386       387       388       389       390       391       392
## 20.050600 13.102516 19.612660 21.146593 11.477851 24.013554 12.227664 18.780812
##       393       394       395       396       397       398       399       400
## 16.998312 17.703533 19.219474 16.082906 24.988627 16.905990 16.819166  6.906715
##       401       402       403       404       405       406       407       408
## 10.396699  9.462380  9.493044  9.149200 10.085208  6.429664  9.491818  9.363607
##       409       410       411       412       413       414       415       416
## 33.146642  6.169156 10.656958 10.285020 19.576915 18.192365 15.408435 16.204585
##       417       418       419       420       421       422       423       424
## 13.064400 12.714223 15.563596 12.876688 12.130205 15.326713 16.177576 11.091930
##       425       426       427       428       429       430       431       432
## 15.283437  8.338918 17.503191 11.669787 17.055376 14.698770 13.615125 19.123404
##       433       434       435       436       437       438       439       440
## 15.159643 16.033951 11.642647 14.129690 14.576136 19.998945 14.430899 16.806174
##       441       442       443       444       445       446       447       448
## 13.377149 19.212642 12.908815 14.018180 20.900084 17.024085 20.079965 18.132712
##       449       450       451       452       453       454       455       456
## 14.391761 20.834553 15.099715 15.198709 14.433741 11.582575 14.095591 22.702300
##       457       458       459       460       461       462       463       464
## 13.388771 11.707389 13.543688 15.365354 13.493178 13.547113 14.305938 13.342376
##       465       466       467       468       469       470       471       472
## 13.458923  9.084261 15.368061 22.059713 17.305973 17.581352 10.057648 10.041485
##       473       474       475       476       477       478       479       480
## 20.898708 17.059756 23.309738 21.046771 24.404592 13.008926 16.680050 11.518757
##       481       482       483       484       485       486       487       488
## 25.077789 17.146997  9.056303 13.324424  9.876517  8.303610 10.066814 12.061805
##       489       490       491       492       493       494       495       496
##  8.594515 10.157628 10.343904  9.534507 10.327452 32.925938 11.949590 21.220726
##       497       498       499       500       501       502       503       504
## 12.299363 18.927096 18.306027 15.884824 18.037786 18.403341 25.590633 17.681345
##       505       506       507       508       509       510       511       512
## 18.483102 13.642850 18.828147 14.440057 12.481658 14.345340 16.194822 17.346915
##       513       514       515       516       517       518       519       520
## 19.840285 12.956320 21.512044 20.212748 15.514494 18.528028 17.295382 13.700144
##       521       522       523       524       525       526       527       528
## 19.312939 14.737653 18.436180 14.948619 17.565039 17.712870 20.278950 20.315937
##       529       530       531       532       533       534       535       536
## 25.302050 14.837909 26.371342 16.530842 24.622986 20.704320 15.951850 19.192762
##       537       538       539       540       541       542       543       544
## 16.228717 17.512718 21.491793 18.519270 12.598608 16.214791 25.911220 19.066693
##       545       546       547       548       549       550       551       552
## 18.075259 21.407068 14.786424 17.825973 24.708215 27.174259 16.056871 20.091823
```

```
##        553        554        555        556        557        558        559        560
## 27.484783 21.821444 14.638693 17.762649 16.306617 22.888718 27.028134  9.413878
##        561        562        563        564        565        566        567        568
## 22.164115 21.015542 25.364979 19.710411 17.550265 14.922030 16.153198 16.373225
##        569        570        571        572        573        574        575        576
## 14.245438 11.466673 15.443247 13.624969 19.784995 13.883348  7.412912 15.770698
##        577        578        579        580        581        582        583        584
## 12.597095 11.015669 23.071610  9.927913 11.856553 12.527406 13.201878 16.344480
##        585        586        587        588        589        590        591        592
## 21.527620 11.416886 13.627548 17.579039 17.426745 13.053766 19.866244 16.001600
##        593        594        595        596        597        598        599        600
## 19.494265 22.553699 19.995400 16.234468 18.305037 16.487056 17.323396 13.766241
##        601        602        603        604        605        606        607        608
## 15.795829 23.913832 16.763371 21.792922 16.642948 19.004877 17.345616 18.705703
##        609        610        611        612        613        614        615        616
## 19.243268 14.884045  9.114708  9.177053 12.892976  9.987665 12.357114 16.636808
##        617        618        619        620        621        622        623        624
##  7.867989 12.810111 16.829206 14.603966 14.243290 10.933023 15.333754  9.772141
##        625
##  8.347697
```

Based off of the results of the linear regression model, the features SelfEmployed and White have p-values above the threshold 0.05 and so they are not meaningful to the overall model. The adjusted coefficient of determination is 0.6051 and tells us that the model explains around 60% of the varability in the data. We could potentially remove SelfEmployed and White as features in order to improve the overall fit of our model and adjusted R^2.

The linear regression model is useful in that the predictions give us actual numeric ranking values for Poverty. In the classification methods, we classify Poverty as 1 if the numeric ranking is greater than 20 and 0 if it is less than. This is less informative than the linear regression model's predictions in depicting the intensity of perhaps a poverty ranking of 22 versus 38. By using the linear model we have more prediction accuracy regarding the scale of poverty between counties.

The model we prefer depends on the question we are asking regarding Poverty. If we want to focus on questions regarding only groups over the threshold of 20, we would use our classification methods to analyze the data. However, if we want more informative numeric data, perhaps the mean poverty ranking of all counties in one state, we would apply our regression model.

They compliment one another because our classification methods allow us to analyze our data through a broad scope while the regression model allows us to approach our questions through a more magnified lens.

21.(9 pts) (Open ended) 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).

Before beginning to perform our data analytics we believed the education level would have a heavy weight on the ranking of Poverty in a county. However, through the use of our lasso model's coefficients we discovered that the four education variables had a very low coefficient, around .0002, almost zero. This shows that they are not significant features. When we performed step selection on our linear model, three of the four education variables were completely removed from the model. Also, our pruned tree did not include these features.From this we can infer that education level does not necessarily contribute to poverty. However, if we were analyzing wealth disparity, we would expect to see education level variables play a more significant role in the model.

Our best performing model based off of test error was our logistic regression model. The logistic regression model performs well with binary classification. This model is more interpretable and gives us a broad overview when sorting counties into a 1 or 0 rank. Although this model fit our data the best, we believe that we could possibly be dismissing valuable differences between counties within the same ranked group that could address the degree of poverty.

Through all of our data analytics, Employment is consistently a significant feature. Employment was the first feature in our decision tree as well as the most significant feature when performing lasso. In our PC1 analyses, we observe that employment is negatively linearly correlated with Poverty. This means that as employment increases within a county, poverty decreases. This intuitively makes sense when thinking about the causes of Poverty and from these data tools we can infer that employment is a dominant factor when assessing poverty within a county.

If we wanted to further explore the causes of Poverty we could sample and add new features such as homelessness within a county and population by square foot. These features could shed light on poverty within cities where homelessness is high. We have shown that different models provide different answers that other models cannot and thus it is important to select your model based on what question you are asking.