Final Project

Nikash Narula (Perm# 3877636) and Jagdeep Chahal (Perm# 7246374)

December 10, 2021

Loading in Census/Education Data

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.0.2
```

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

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

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

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

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

```
## See spec(...) for full column specifications.
```

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

```
## Parsed with column specification:
```

```
## cols(
##   .default = col_double(),
##   `FIPS Code` = col_character(),
##   State = col_character(),
##   `Area name` = col_character(),
##   `Less than a high school diploma, 1970` = col_number(),
##   `High school diploma only, 1970` = col_number(),
##   `Some college (1-3 years), 1970` = col_number(),
##   `Four years of college or higher, 1970` = col_number(),
##   `Less than a high school diploma, 1980` = col_number(),
##   `High school diploma only, 1980` = col_number(),
##   `Some college (1-3 years), 1980` = col_number(),
##   `Four years of college or higher, 1980` = col_number(),
##   `Less than a high school diploma, 1990` = col_number(),
##   `High school diploma only, 1990` = col_number(),
##   `Some college or associate's degree, 1990` = col_number(),
##   `Bachelor's degree or higher, 1990` = col_number(),
##   `Less than a high school diploma, 2000` = col_number(),
##   `High school diploma only, 2000` = col_number(),
##   `Some college or associate's degree, 2000` = col_number(),
##   `Bachelor's degree or higher, 2000` = col_number(),
##   `Less than a high school diploma, 2015-19` = col_number()
##   # ... with 3 more columns
## )
## See spec(...) for full column specifications.
```

```r
census = as.data.frame(census)
education = as.data.frame(education)
```

Preliminary Data Analysis

1. The dimension of census is 3142 x 31. There are 0 missing values. There are 51 distinct values in State in census which indicates all 50 states and federal district are contained in the data.
2. The dimension of education is 3143 x 42. 18 distinct counties contain missing values. There are 1877 distinct values in County in education. There are also 1877 distinct values in County in census. One can assume each dataset has the same counties.

```r
dim(census)
```

```
## [1] 3142   36
```

```r
sum(is.na(census))
```

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

```r
length(unique(census$State))
```

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

```r
dim(education)
```

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

```r
length(which(apply(education, 1, function(X) any(is.na(X)))))
```

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

```r
length(unique(education$County))
```

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

```r
length(unique(census$County))
```

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

Data Wrangling

3. Removed NA values from education.
4. Mutate to include 6 features + TotalPop.
5. State-level summary into a dataset named education.state.
6. state.level created with variable of highest degree of education in that state, LargestEduLevel.

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

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
edu_levels = c("Less than a high school diploma, 2015-19", "High school diploma only, 2015-19", "Some co
state.level = education.state %>% mutate(LargestEduLevel = edu_levels[max.col(education.state[2:5])])
```

Visualization

7. Color map by education level with highest population. Show legend.
8. Visualiztion for census data
9. Clean and aggregate census data. From the correlation matrix of census.clean, it's clear that Women is colinear with Total Pop, and White is colinear with Minority. Thus, one column of each pair should be deleted. Columns Women and White are deleted.
10. Print first 5 rows of census.clean

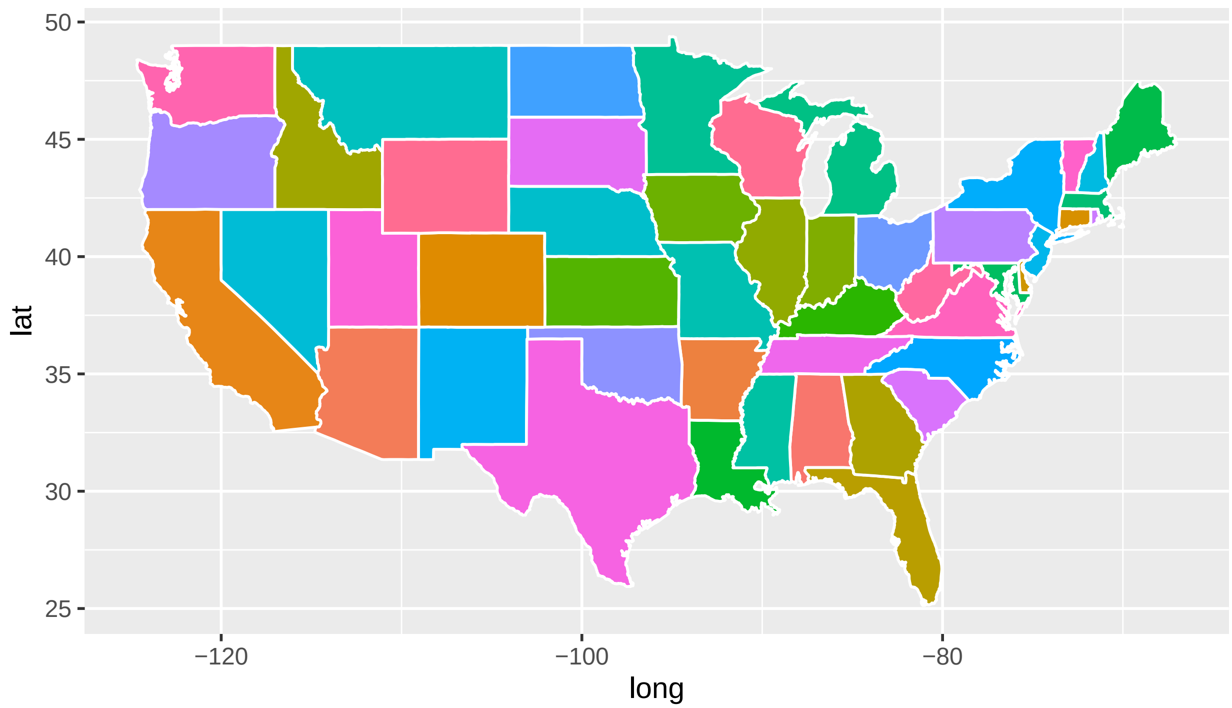
```r
install.packages("maps")
```

```
## Updating HTML index of packages in '.Library'

## Making 'packages.html' ... done
```

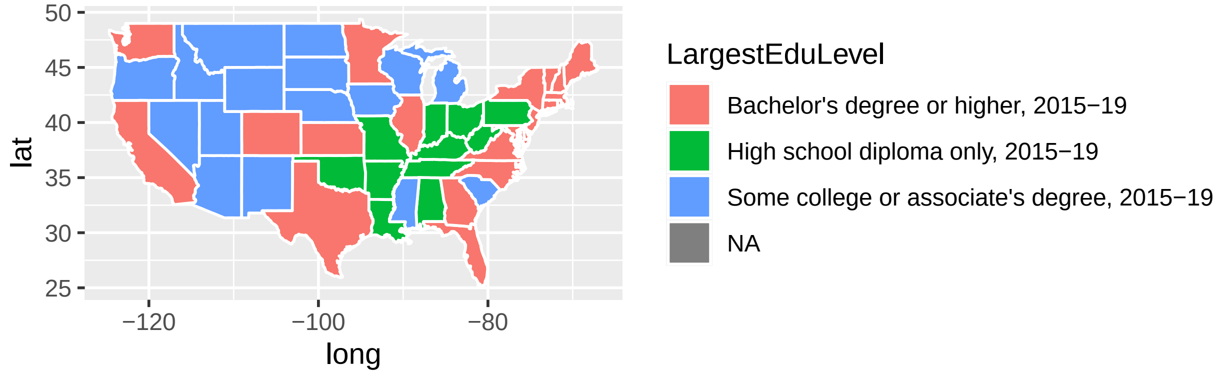
```r
library(maps)
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.0.1
```

```r
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
```

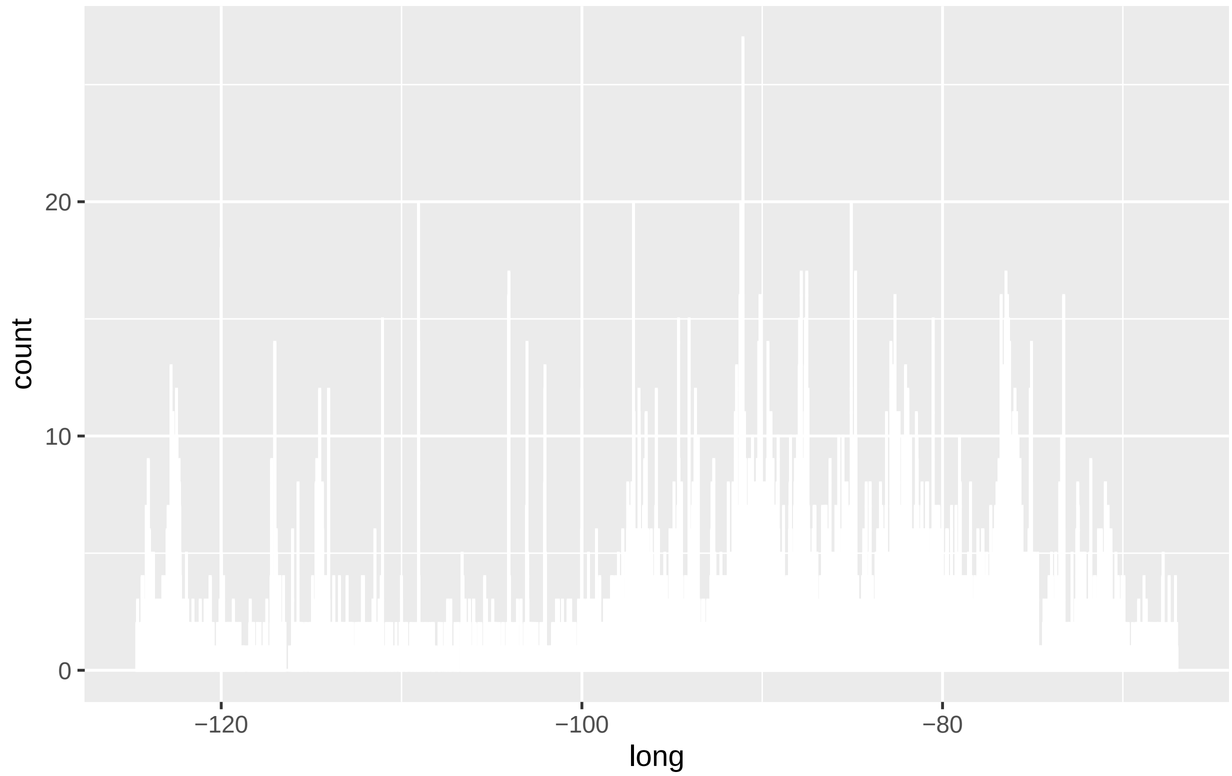



```
library(stringr)
states.rename = states %>% mutate(State = state.abb[match(str_to_title(region), state.name)])
states.join = left_join(x=states.rename, y=state.level, by="State", all.x=TRUE)
ggplot(data = states.join) +
  geom_polygon(aes(x = long, y = lat, fill = `LargestEduLevel`, group = group),
               color = "white") +
  coord_fixed(1.3) + scale_color_manual(name='Largest Education Level Per State',
                     breaks=edu_levels,
                     values=c('Less than a high school diploma, 2015-19'='grey', 'High school diploma or
```




```
ggplot(data = states.join) +
    geom_bar(aes(x = long, fill = `LargestEduLevel`, group = group),
                color = "white") +
    coord_fixed(1.3) + guides(fill=FALSE)
```

```
## Warning: position_stack requires non-overlapping x intervals
```




```
census.clean = census[complete.cases(census), ]
census.clean = census.clean %>% mutate(Men = Men/TotalPop, Employed = Employed/TotalPop, VotingAgeCitiz
cor_census = cor(census.clean[3:23])
census.clean = census.clean %>% select(-c(Women, White))
head(census.clean, 5)
```

```
##   State         County TotalPop       Men VotingAgeCitizen Income IncomeErr
## 1    AL Autauga County    55036 0.4887528        0.7452576  55317      2838
## 2    AL Baldwin County   203360 0.4894129        0.7640441  52562      1348
## 3    AL Barbour County    26201 0.5334148        0.7735964  33368      2551
## 4    AL    Bibb County    22580 0.5425598        0.7821966  43404      3431
## 5    AL  Blount County    57667 0.4940434        0.7372154  47412      2630
##   IncomePerCap IncomePerCapErr Poverty ChildPoverty Professional Service Office
## 1        27824            2024    13.7         20.1         35.3    18.0   23.2
## 2        29364             735    11.8         16.1         35.7    18.2   25.6
## 3        17561             798    27.2         44.9         25.0    16.8   22.6
## 4        20911            1889    15.2         26.6         24.4    17.6   19.7
## 5        22021             850    15.6         25.4         28.5    12.9   23.3
##   Production Drive Carpool Transit OtherTransp WorkAtHome MeanCommute  Employed
## 1       15.4  86.0     9.6     0.1         1.3        2.5        25.8 0.4381132
## 2       10.8  84.7     7.6     0.1         1.1        5.6        27.0 0.4402390
## 3       24.1  83.4    11.1     0.3         1.7        1.3        23.4 0.3388420
## 4       22.4  86.4     9.5     0.7         1.7        1.5        30.0 0.3618689
## 5       19.5  86.8    10.2     0.1         0.4        2.1        35.0 0.3707493
##   PrivateWork SelfEmployed FamilyWork Minority
## 1        74.1          5.6        0.1     22.8
```

```
## 2          80.7          6.3          0.1       15.4
## 3          74.1          6.5          0.3       52.8
## 4          76.0          6.3          0.3       24.8
## 5          83.9          4.0          0.1       10.9
```

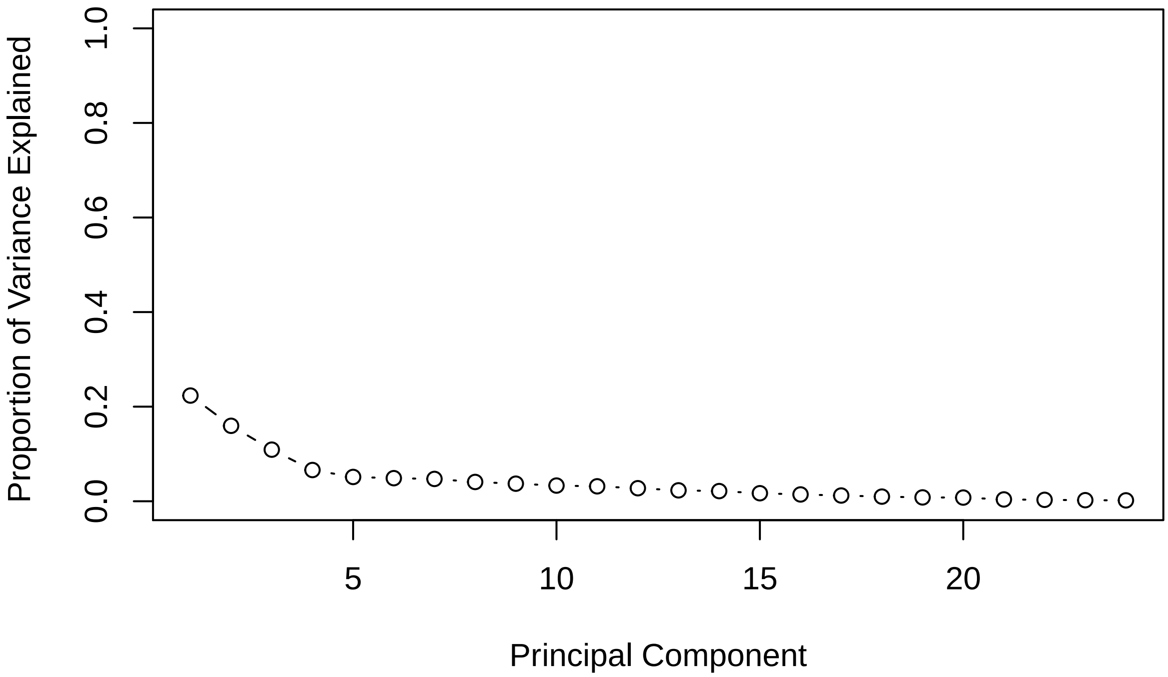
Dimensionality Reduction

11. Run PCA for the cleaned county level census data (with State and County excluded). We chose to center (by default) and scale the features before running PCA because many of the variables has vastly different means and variances. If we failed to scale these vairables prior to PCA, then most of the principal components would be driven largely by the variables with the largest mean and variance, like TotalPop. The three features with largest asbolute values for first principal component are WorkAtHome, SelfEmployed and Drive, respectively. The features with positive signs for the first principal component are TotalPop, Men, VotingAgeCitizen, Professional, Transit, OtherTransp, WorkAtHome, Employed, SelfEmployed and FamilyWork. The features with negative signs are Poverty, Service, Office, Production, Drive, Carpool, MeanCommute, PrivateWork and Minority. This means that these features with opposing signs are negatively correlated.
12. Determine the number of minimum number of PCs needed to capture 90% of the variance for the analysis. We need 12 PCs to explain 90% of total variation in the data as the cumulative proportion of variance explained at 12 Pcs is 0.9075578.

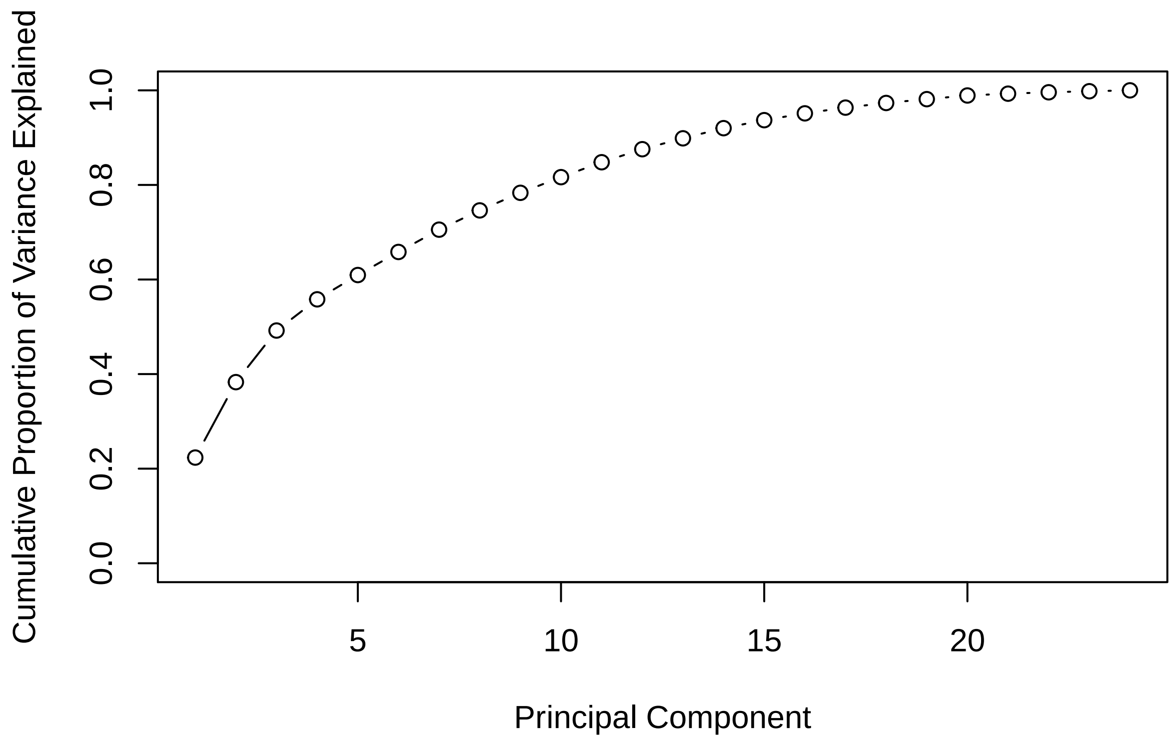
```r
census.pr = census.clean %>% select(-c(State, County))
pr.out = prcomp(census.pr, scale=TRUE, center=TRUE )
PC1 = pr.out$rotation[,1]
PC2 = pr.out$rotation[,2]
pc.county = data.frame(PC1,PC2)
sort(abs(PC1))
```

```
##           Office      PrivateWork VotingAgeCitizen      OtherTransp
##      0.002907516      0.005937895      0.006132599      0.010641829
##              Men      MeanCommute       FamilyWork        IncomeErr
##      0.015947571      0.039032888      0.050835270      0.053038045
##  IncomePerCapErr         TotalPop          Carpool          Transit
##      0.072502808      0.088857836      0.115579495      0.120120968
##     SelfEmployed            Drive         Minority          Service
##      0.131881803      0.150276607      0.152370315      0.174015125
##       Production       WorkAtHome     Professional         Employed
##      0.205268871      0.220633028      0.323761366      0.347204440
##          Poverty     ChildPoverty           Income     IncomePerCap
##      0.358959867      0.362446835      0.363167830      0.387193804
```

```r
pr.var = pr.out$sdev^2
pve = pr.var/sum(pr.var)
plot(pve, xlab="Principal Component", ylab="Proportion of Variance Explained ", ylim=c(0,1), type='b')
```




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



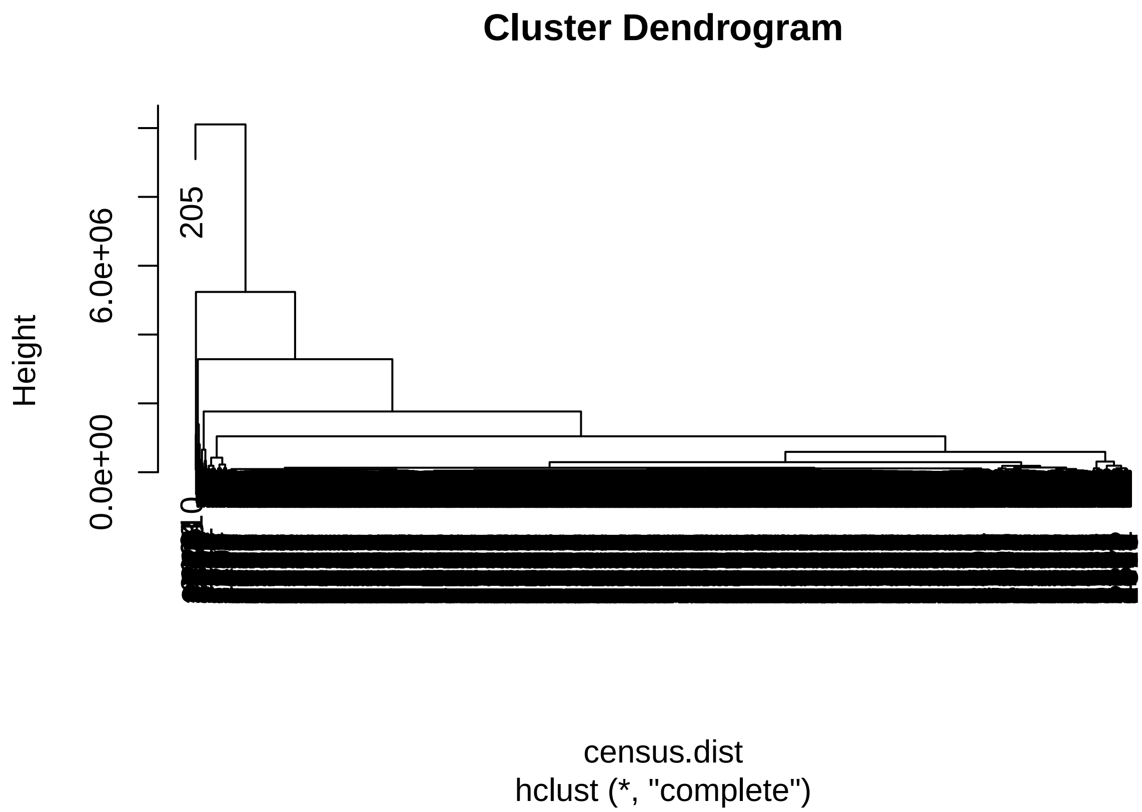

```
cumsum(pve)
```

```
##  [1] 0.2234663 0.3830035 0.4920980 0.5581063 0.6094430 0.6582416 0.7054074
##  [8] 0.7461646 0.7833175 0.8164488 0.8480002 0.8755389 0.8985869 0.9200365
## [15] 0.9369918 0.9513764 0.9634717 0.9732903 0.9813963 0.9891477 0.9929490
## [22] 0.9959209 0.9981316 1.0000000
```

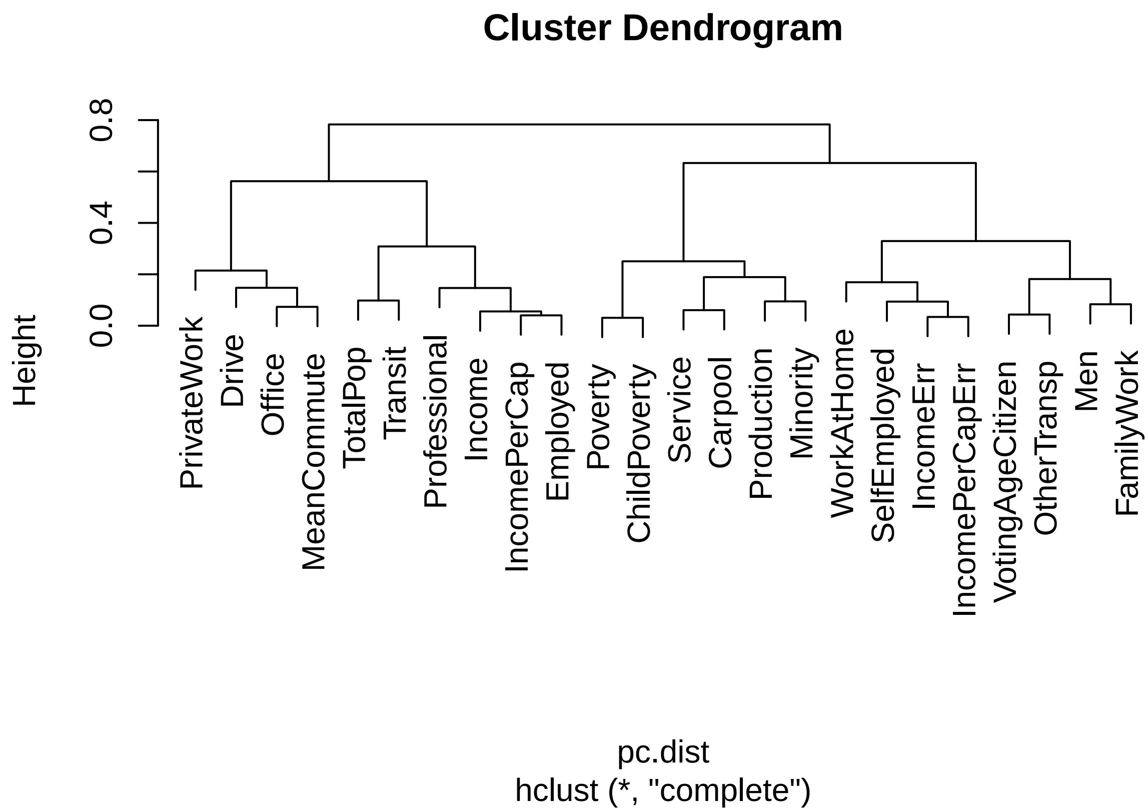
Clustering

13. Perform hierarchical clustering with complete linkage on census.clean (with State and County excluded).

```
# cleaned census \data
census.hc = census.pr
census.dist = dist(census.hc)
set.seed(1)
census.hclust = hclust(census.dist)
plot(census.hclust)
```



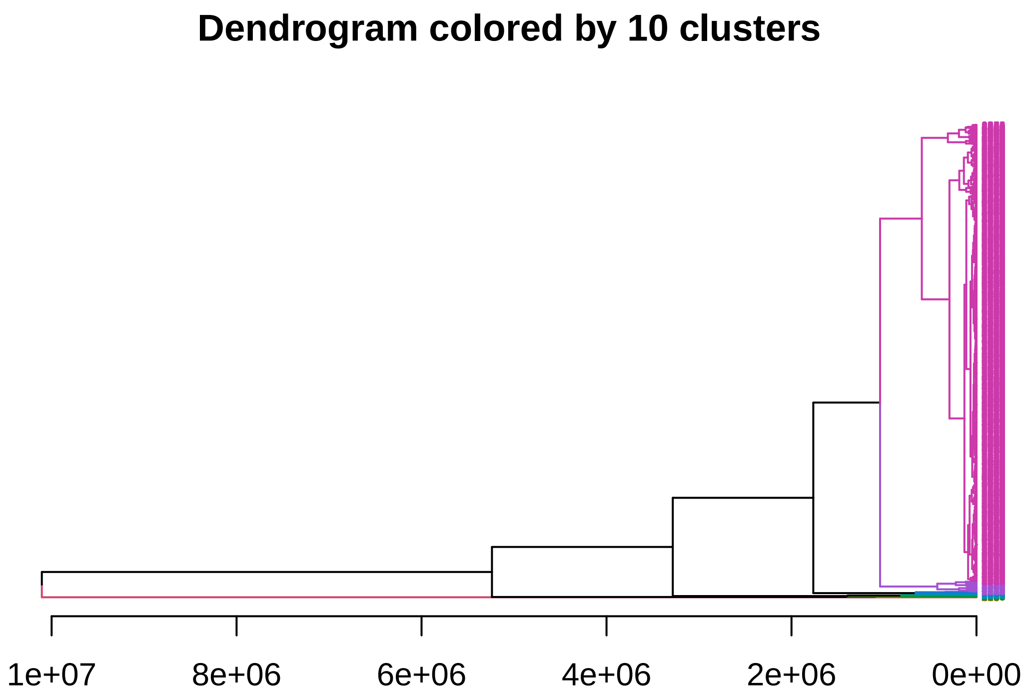

```
census.clus = cutree(census.hclust, k=10)
census.hc.ct <- mutate(census.hc, cluster = census.clus)

# PC1,PC2 data
census.pc = data.frame(PC1,PC2)
pc.dist = dist(census.pc)
set.seed(1)
pc.hclust = hclust(pc.dist)
plot(pc.hclust)
```



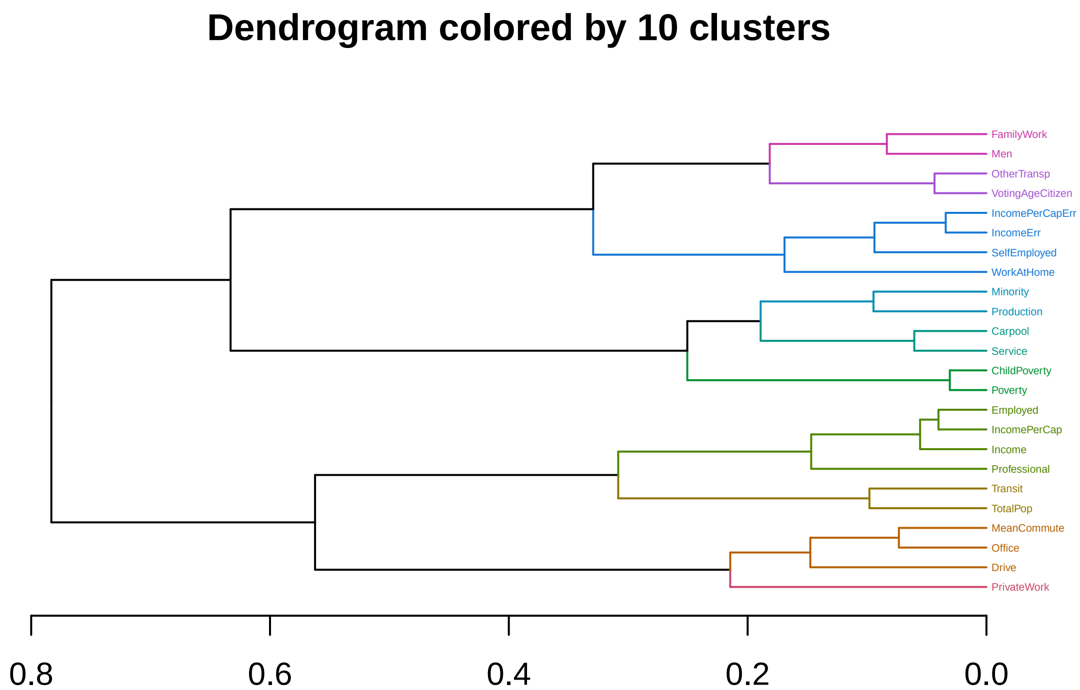

```r
pc.clus = cutree(pc.hclust, k=10)
pc.hc.ct <- mutate(census.pc, cluster = pc.clus)

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

```
##
## ---------------------
## Welcome to dendextend version 1.15.2
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## You may ask questions at stackoverflow, use the r and dendextend tags:
##   https://stackoverflow.com/questions/tagged/dendextend
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------

##
## Attaching package: 'dendextend'

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

```r
# dendrogram: branches colored by 10 groups
dend_census = as.dendrogram(census.hclust)
# color branches and labels by 10 clusters
dend_census = color_branches(dend_census, k=10)
dend_census = color_labels(dend_census, k=10)
# change label size
dend_census = set(dend_census, "labels_cex", 0.3)
# plot the dendrogram
plot(dend_census, horiz=T, main = "Dendrogram colored by 10 clusters")
```




```r
# dendrogram: branches colored by 10 groups
dend_pc = as.dendrogram(pc.hclust)
# color branches and labels by 10 clusters
dend_pc = color_branches(dend_pc, k=10)
dend_pc = color_labels(dend_pc, k=10)
# change label size
dend_pc = set(dend_pc, "labels_cex", 0.3)
# plot the dendrogram
plot(dend_pc, horiz=T, main = "Dendrogram colored by 10 clusters")
```



Modeling

14. Transform 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. In this case, we have removed VotingAgeCitizen, Drive, Carpool, Transit, OtherTransp as they don't seem to have any effect on the classification of Poverty form looking at the data.

```
# we join the two datasets
all <- census.clean %>%
  left_join(education, by = c("State"="State", "County"="County")) %>% na.omit
all = all %>% mutate(Poverty=factor(ifelse(Poverty >= 20,1,0))) %>% select(-c(VotingAgeCitizen, Drive, C

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

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

calc_error_rate = function(predicted.value, true.value){
  return(mean(true.value!=predicted.value))
}
records = matrix(NA, nrow=3, ncol=2)
```

```r
colnames(records) = c("train.error","test.error")
rownames(records) = c("tree","logistic","lasso")
```

Classification

15. Train a decision tree by cv.tree(). Looking at the decision trees, both pre-pruned and pruned trees split on Employed first, followed by Minority. Intuitively, this makes sense as much of the poverty in America is unfortunately associated with minorities (Black, Asian, Native, Latino, etc.). Historically, these groups have faced much more hardship and typically had far less opportunities to make their way up the social ladder. Finally, the training error obtained was 0.1561874 and test error rate obtained was 0.152.
16. Run Logistic Regression to predict poverty in each county. The statistically significant variables at the 0.05 level are TotalPop, Men, Professional, Service, Production, WorkAtHome, Employed, PrivateWork, Minority, and the 4 variables representing education level. This is consistent with what we saw in the tree decision analyis, where Employed, Minority, Service and Men were the most significant. The variable Employed has a coefficient -30.07. For a one unit increase in EMployed, the log odds of being in poverty decreases by 30.07, holding other variables fixed.
17. Logistic Regression with Lasso.
18. ROC curves.

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------------------------------- t

## v tibble  3.0.3     v purrr   0.3.4
## v tidyr   1.1.1     v forcats 0.5.0

## Warning: package 'tibble' was built under R version 4.0.2

## Warning: package 'tidyr' was built under R version 4.0.2

## -- Conflicts ------------------------------------------------------------------------------ tidyve
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::map()    masks maps::map()
```

```r
library(ISLR)
library(glmnet)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.0-2
```

```r
library(tree)
```

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

```r
library(maptree)
```

```
## Loading required package: cluster

##
```

```
## Attaching package: 'cluster'

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

## Loading required package: rpart

##
## Attaching package: 'rpart'

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

```
library(randomForest)
```

```
## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

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

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

```
library(gbm)
```

```
## Loaded gbm 2.1.8
```

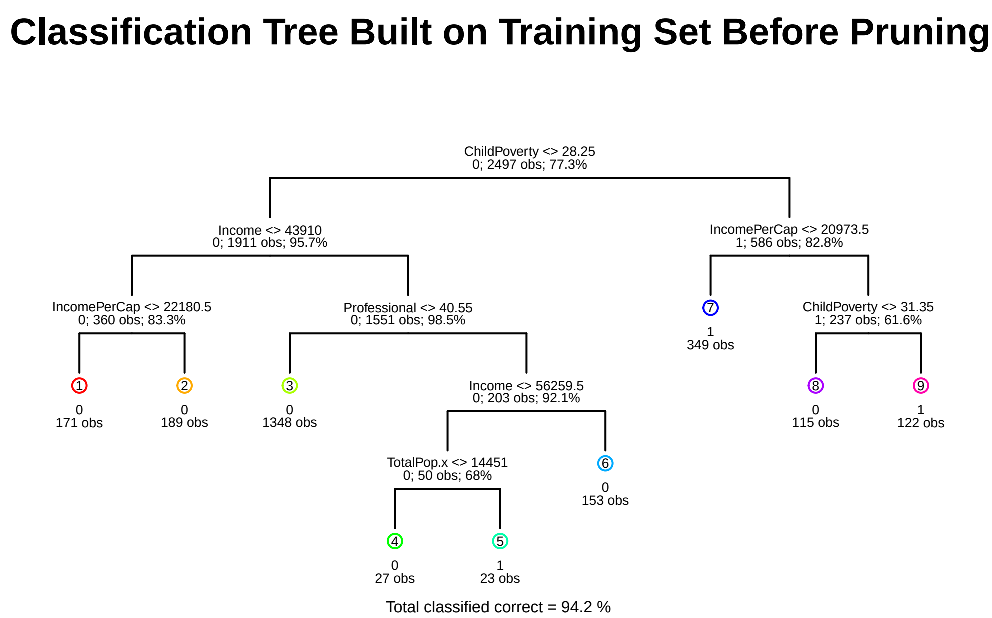
```
library(ROCR)
tree.poverty = tree(Poverty ~ ., data = all.tr)
```

```
## Warning in tree(Poverty ~ ., data = all.tr): NAs introduced by coercion
```

```
summary(tree.poverty)
```

```
##
## Classification tree:
## tree(formula = Poverty ~ ., data = all.tr)
## Variables actually used in tree construction:
## [1] "ChildPoverty" "Income"       "IncomePerCap" "Professional" "TotalPop.x"
## Number of terminal nodes:  9
## Residual mean deviance:  0.3037 = 755.5 / 2488
## Misclassification error rate: 0.05767 = 144 / 2497
```

```
draw.tree(tree.poverty, nodeinfo=TRUE, cex = 0.4)
title("Classification Tree Built on Training Set Before Pruning")
```




```r
# prune the data
set.seed(3)
cv = cv.tree(tree.poverty, FUN=prune.misclass, K=10, rand=folds) # Print out cv
```

```
## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion
```

```
## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion
```

```
cv$size
```

```
## [1] 9 8 4 2 1
```

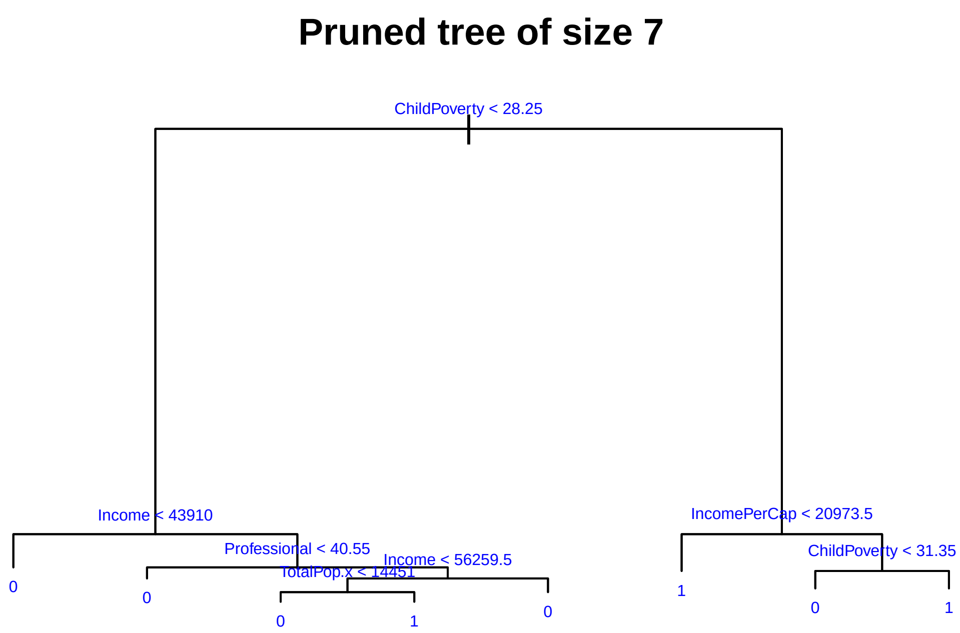
```
cv$dev
```

```
## [1] 157 157 162 184 568
```

```
best.cv = min(cv$size[cv$dev == min(cv$dev)])
best.cv # 7 is best cv
```

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

```
pt.cv = prune.misclass(tree.poverty, best=best.cv)
# Plot pruned tree
plot(pt.cv)
text(pt.cv, pretty=0, col = "blue", cex = .5)
title("Pruned tree of size 7")
```




```
tree.poverty.pred.tr = predict(pt.cv, all.tr, type="class")
```

```
## Warning in pred1.tree(object, tree.matrix(newdata)): NAs introduced by coercion
```

```
tree.poverty.pred.te = predict(pt.cv, all.te, type="class")
```

```
## Warning in pred1.tree(object, tree.matrix(newdata)): NAs introduced by coercion
```

```
records[1,1] = calc_error_rate(tree.poverty.pred.tr, all.tr$Poverty)
records[1,2] = calc_error_rate(tree.poverty.pred.te, all.te$Poverty)

all.train = all.tr[,3:21]
all.test = all.te[,3:21]
glm.fit.tr = glm(Poverty~., data = all.train, family=binomial)
```

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

```
glm.fit.te = glm(Poverty~., data = all.test, family=binomial)
```

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

```
summary(glm.fit.tr)
```

```
##
## Call:
## glm(formula = Poverty ~ ., family = binomial, data = all.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
```

```
## -3.1010  -0.1282  -0.0150  -0.0001   3.5778
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      9.489e+00  4.651e+00   2.040 0.041326 *
## TotalPop.x       3.849e-07  4.318e-07   0.891 0.372739
## Men             -1.107e+01  4.709e+00  -2.350 0.018790 *
## Income          -1.604e-04  3.212e-05  -4.995 5.88e-07 ***
## IncomeErr       -1.316e-04  8.491e-05  -1.550 0.121131
## IncomePerCap    -5.617e-04  7.421e-05  -7.569 3.76e-14 ***
## IncomePerCapErr  2.909e-04  1.619e-04   1.797 0.072299 .
## ChildPoverty     3.880e-01  2.918e-02  13.297  < 2e-16 ***
## Professional     2.586e-01  3.896e-02   6.637 3.20e-11 ***
## Service          3.358e-02  4.228e-02   0.794 0.427070
## Office          -3.999e-02  4.441e-02  -0.901 0.367798
## Production      -3.928e-02  3.403e-02  -1.154 0.248399
## WorkAtHome       1.172e-02  5.111e-02   0.229 0.818689
## MeanCommute      1.189e-02  2.368e-02   0.502 0.615564
## Employed         1.429e+01  3.502e+00   4.079 4.51e-05 ***
## PrivateWork     -8.870e-02  2.563e-02  -3.461 0.000539 ***
## SelfEmployed    -2.637e-01  4.998e-02  -5.277 1.31e-07 ***
## FamilyWork      -6.504e-01  2.566e-01  -2.535 0.011249 *
## Minority         2.088e-03  6.572e-03   0.318 0.750759
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2677.81  on 2496  degrees of freedom
## Residual deviance:  671.85  on 2478  degrees of freedom
## AIC: 709.85
##
## Number of Fisher Scoring iterations: 8
```

```r
summary(glm.fit.te)
```

```
##
## Call:
## glm(formula = Poverty ~ ., family = binomial, data = all.test)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9045  -0.1079  -0.0085   0.0000   3.2324
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -2.630e+00  1.206e+01  -0.218  0.82739
## TotalPop.x       2.872e-07  1.270e-06   0.226  0.82114
## Men             -1.275e+01  1.114e+01  -1.144  0.25267
## Income          -1.464e-04  6.462e-05  -2.266  0.02345 *
## IncomeErr       -1.459e-04  1.901e-04  -0.768  0.44273
## IncomePerCap    -5.782e-04  1.481e-04  -3.904 9.44e-05 ***
## IncomePerCapErr  3.553e-04  3.559e-04   0.998  0.31824
## ChildPoverty     3.844e-01  5.666e-02   6.785 1.16e-11 ***
## Professional     1.977e-01  9.145e-02   2.162  0.03065 *
```

```
## Service          9.377e-02  8.795e-02   1.066  0.28634
## Office           8.966e-02  1.042e-01   0.860  0.38962
## Production       4.172e-02  8.372e-02   0.498  0.61826
## WorkAtHome       2.249e-01  1.068e-01   2.106  0.03520 *
## MeanCommute      2.342e-03  4.817e-02   0.049  0.96122
## Employed         1.006e+00  8.544e+00   0.118  0.90629
## PrivateWork      6.165e-02  6.415e-02   0.961  0.33654
## SelfEmployed    -5.265e-02  1.119e-01  -0.470  0.63811
## FamilyWork      -5.327e-01  3.947e-01  -1.349  0.17718
## Minority         3.581e-02  1.279e-02   2.799  0.00512 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 677.10  on 624  degrees of freedom
## Residual deviance: 162.83  on 606  degrees of freedom
## AIC: 200.83
##
## Number of Fisher Scoring iterations: 9
```

```
prob.training = predict(glm.fit.tr, type="response")
prob.test = predict(glm.fit.te, type="response")
records[2,1] = calc_error_rate(prob.training, all.tr$Poverty)
records[2,2] = calc_error_rate(prob.test, all.te$Poverty)

# NOTE: The Lasso code is not working - but the commented code is generally the approach one would take
#lambda.list.lasso = seq(1, 20) * 1e-5
#lasso_mod_train = glmnet(all.train, all.train$Poverty, alpha = 1, lambda = lambda.list.lasso) # fit la
# find optimal value
#set.seed(1)
#cv.out.lasso = cv.glmnet(all.train, all.train$Poverty, alpha = 1, lambda=lambda.list.lasso,
#                          nfolds=10)
#plot(cv.out.lasso)
#abline(v = log(cv.out.lasso$lambda.min), col="red", lwd=3, lty=2)
#bestlam_lasso = cv.out.lasso$lambda.min
#bestlam_lasso

#out = glmnet(x,y,alpha=1)
#predict(out,type="coefficients",s=bestlam_lasso)[1:11,] # display coefficient estimates using #optimal
#lasso.pred=predict(lasso_mod_train,s=bestlam_lasso,newx=all.train)
#records[3,1] = mean((lasso.pred-all.train$Poverty)^2)
#lasso.pred=predict(lasso_mod_train,s=bestlam_lasso,newx=all.test)
#records[3,2] = mean((lasso.pred-all.test$Poverty)^2)
```

Taking it Further

19. Additional Classification Methods: Random Forest Looking at the Random Forest method, it's clear that our conclusions are also consistent with that of logistic regression and the tree method conucted earlier. The graph of variable importance shows that Employed and Minority are the 2 most important variables in determining poverty level.
20. Considering Another Regression Problem Another regression we considered was linear regression. Doing so, we did not convert Poverty to a classification variable (1 and 0) before. We found very similar results in the data, specifically that Employed and minority were the 2 most significant factors. I prefer the classification method, however, because this is more intuitive and easily explainable to someone

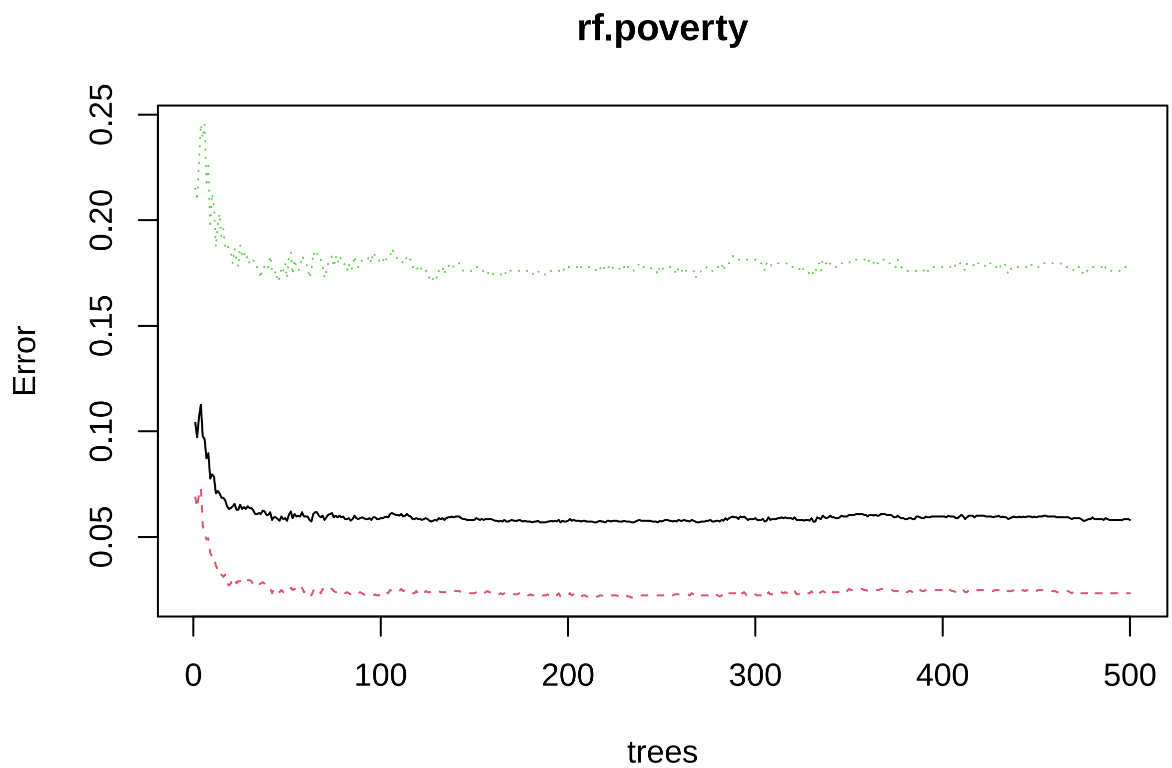
who would not otherwise understand the data.

21. Overall Insights There are many different things I learned from this project and one of the most important ideas is that most of my inferences prior to this project can now be seen as results and conclusions made by the data. A prime example is how I came to understand how all the possible different factors there are when taking into account Poverty. In my Random forest model, the exact level of importance can be seen and we can conclude that Employed and Minority are considerably more important than the other listed factors. The predictions are mainly influenced by Employed and Minority. We can see positive correlation how value of men increases as minority and employed are still increased. But a key thing to note from this deduction is that the training and test errors for Lasso Regression, Logistic Regression, and Decision Trees were all around 15%. Because variables and features were similar in their data set resulted in them nit having that much of an impact. We took voting, carpool, transit, drive and other transport out of the equation and the training and test errors were still around 15%. For the future, I would like to take other variables into account knowing now that so many factors come into play for Poverty, in this case specifically.

```
rf.poverty = randomForest(Poverty~., data=all.tr, importance=TRUE)
rf.poverty
```

```
##
## Call:
##  randomForest(formula = Poverty ~ ., data = all.tr, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 5.81%
## Confusion matrix:
##      0   1 class.error
## 0 1884  45  0.02332815
## 1  100 468  0.17605634
```

```
plot(rf.poverty)
```




```
yhat.rf = predict(rf.poverty, newdata = all.tr)
train.rf.err = mean(yhat.rf != all.tr$Poverty)
train.rf.err
```

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

```
yhat.rf = predict(rf.poverty, newdata = all.te)
test.rf.err = mean(yhat.rf != all.te$Poverty)
test.rf.err
```

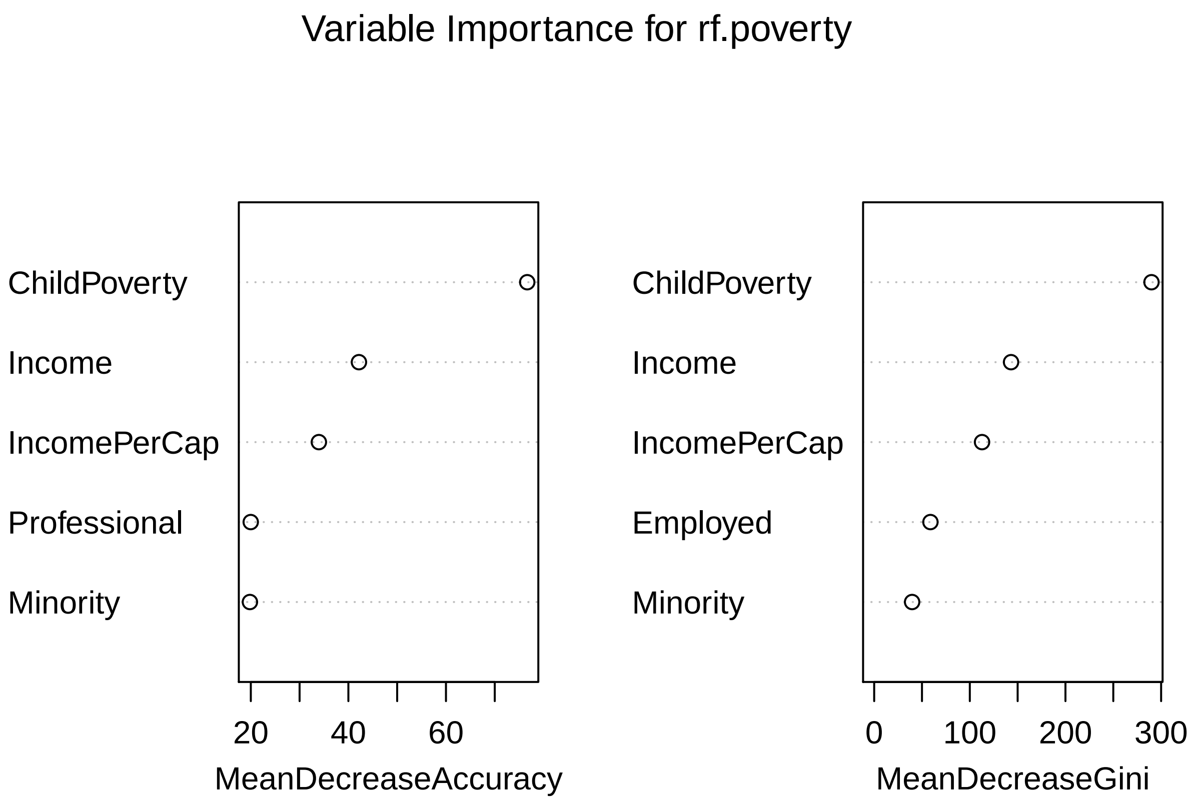
```
## [1] 0.048
```

```
importance(rf.poverty)
```

```
##                            0            1 MeanDecreaseAccuracy
## State               6.017071  3.275520687             6.640028
## County              1.455221 -0.005328497             1.198748
## TotalPop.x         10.748792  6.639675617            11.945050
## Men                10.794479  3.579330848            11.054682
## Income             28.934621 31.040536871            42.163170
## IncomeErr           9.989550  5.451054956            11.996741
## IncomePerCap       21.962234 29.012954752            33.948566
## IncomePerCapErr     8.597356  6.789623764            11.007727
## ChildPoverty       46.452185 71.159065164            76.657352
## Professional       20.735209  3.479909148            19.984089
## Service            12.171781  6.522422747            13.526208
## Office              9.083341  1.201356530             8.895583
## Production         12.524988  4.264858261            14.101506
```

```
## WorkAtHome                    7.375359  4.960548124              9.142917
## MeanCommute                   9.346696  6.921032903             11.376417
## Employed                     12.270815 13.981157203             18.878882
## PrivateWork                  15.416843  9.512702943             17.502361
## SelfEmployed                 12.057136  4.519498016             12.932863
## FamilyWork                    3.560107  0.323513282              3.529537
## Minority                     11.886064 16.032616668             19.814257
## LessThanHighSchoolDiploma     8.506231  5.253380320              9.642641
## HighSchoolDiplomaOnly         9.155929  4.505406861              9.776543
## SomeCollegeOrAssociateDegree  8.969776  6.350840953             10.682924
## BachelorDegreeOrHigher       14.817066  6.587602303             16.358752
## TotalPop.y                    8.038239  4.806475962              9.201823
##                              MeanDecreaseGini
## State                               10.026052
## County                               9.952581
## TotalPop.x                           9.402142
## Men                                 17.810077
## Income                             143.114255
## IncomeErr                           10.080247
## IncomePerCap                       112.782734
## IncomePerCapErr                      9.909433
## ChildPoverty                       289.942708
## Professional                        18.979937
## Service                             14.422030
## Office                               9.423659
## Production                          11.302579
## WorkAtHome                          15.934297
## MeanCommute                         14.067508
## Employed                            58.830238
## PrivateWork                         13.590369
## SelfEmployed                        13.562704
## FamilyWork                           5.411789
## Minority                            39.682949
## LessThanHighSchoolDiploma            9.254308
## HighSchoolDiplomaOnly                8.535179
## SomeCollegeOrAssociateDegree        10.228792
## BachelorDegreeOrHigher              12.594216
## TotalPop.y                           8.703660
```

```r
varImpPlot(rf.poverty, sort=T, main="Variable Importance for rf.poverty", n.var=5)
```


Variable Importance for rf.poverty
ChildPoverty
Income
IncomePerCap
Professional
Minority
20
40
60
MeanDecreaseAccuracy
ChildPoverty
Income
IncomePerCap
Employed
Minority
0
100
200
300
MeanDecreaseGini