Pstat 131 Project

Kalvin Goode (9454554), Patrick Chen (970890)

2019/5/26

1. First, before election voting day,voters’ behavior can be easily changed after the candidate express opinions or actions that voters may or may not strongly agree with. For example, President Trump had proposed and insisted political views on the neccessity of building wall between US and Mexico. This view was strongly viewed as racist to many voters. However. as he continued to take series of action to make “America Great Again”, people have shifted more faith toward President Trump. Another reason that makes prediction hard is the voters’ honesty. Many voters who supported candidates may be hesitate to tell surveyors their opinion even if the survey is anonymous since some voters feel that he or she may be in threat to report such opinion.
2. Silver used Bayesian statistic approach and time series to calculate the probability for a candidate to win in one state. In addition, Silver collect variables that can affect voters’ behavior if occured. For example, with prior probability and past data, the model simulated forward to election day to estimate the winning chance of each candidate for state and national. To account for possible bias errors, Silver use previous election polls and result to estimate possible bias for the current polls to adjust deviates in the model. With these techniques and scrutiny, Silver was having a better prediction model than others in 2012 election.
3. All the prediction model had overestimated Chlinton’s votes and underestimated Trump’s votes.

#election.raw = read.csv("data/election/election.csv") %>% as.tbl  
#census\_meta = read.csv("data/census/metadata.csv", sep = ";") %>% as.tbl  
#census = read.csv("data/census/census.csv") %>% as.tbl  
#census$CensusTract = as.factor(census$CensusTract)  
  
election.raw = read.csv("D:/downloads/temp/Pstat 131/Project/data/election/election.csv") %>% as.tbl  
census\_meta = read.csv("D:/downloads/temp/Pstat 131/Project/data/census/metadata.csv", sep = ";") %>% as.tbl  
census = read.csv("D:/downloads/temp/Pstat 131/Project/data/census/census.csv") %>% as.tbl  
census$CensusTract = as.factor(census$CensusTract)

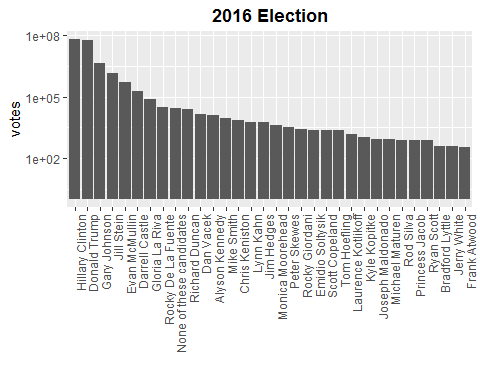
election.raw

## # A tibble: 18,351 x 5  
## county fips candidate state votes  
## <fct> <fct> <fct> <fct> <int>  
## 1 <NA> US Donald Trump US 62984825  
## 2 <NA> US Hillary Clinton US 65853516  
## 3 <NA> US Gary Johnson US 4489221  
## 4 <NA> US Jill Stein US 1429596  
## 5 <NA> US Evan McMullin US 510002  
## 6 <NA> US Darrell Castle US 186545  
## 7 <NA> US Gloria La Riva US 74117  
## 8 <NA> US Rocky De La Fuente US 33010  
## 9 <NA> US " None of these candidates" US 28863  
## 10 <NA> US Richard Duncan US 24235  
## # ... with 18,341 more rows

election\_federal=election.raw%>%  
 filter(fips=="US")  
election\_state=election.raw%>%  
 filter(fips!="US",!fips%in%c(1:99999))  
election=election.raw%>%  
 filter(fips%in%c(1:99999))

1. We see that there are 31 presidential candidates in 2016 election.

his=election\_federal%>%  
 ggplot(data=.,aes(x=reorder(candidate,-votes),y=votes))+geom\_col()+scale\_y\_log10()+xlab("")+ggtitle("2016 Election")+theme(axis.text.x=element\_text(angle = 90, hjust = 1),plot.title=element\_text(hjust=0.5,face="bold"))  
his



# ggtitle("Chart of 2016 Election")+theme(plot.title = #element\_text(hjust = 0.5))  
#his

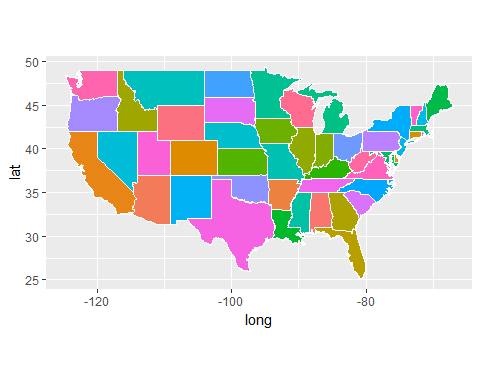
state\_winner=election\_state%>%  
 dplyr::group\_by(state)%>%  
 dplyr::mutate(total=sum(votes))%>%  
 dplyr::mutate(pct=votes/total)%>%  
 dplyr::top\_n(1,pct)  
   
county\_winner=election%>%  
 dplyr::group\_by(fips)%>%  
 dplyr::mutate(total=sum(votes))%>%  
 dplyr::mutate(pct=votes/total)%>%  
 dplyr::top\_n(1,pct)  
head(state\_winner)

## # A tibble: 6 x 7  
## # Groups: state [6]  
## county fips candidate state votes total pct  
## <fct> <fct> <fct> <fct> <int> <int> <dbl>  
## 1 <NA> CA Hillary Clinton CA 8753788 14060856 0.623  
## 2 <NA> FL Donald Trump FL 4617886 9419886 0.490  
## 3 <NA> TX Donald Trump TX 4685047 8917965 0.525  
## 4 <NA> NY Hillary Clinton NY 4556124 7660190 0.595  
## 5 <NA> PA Donald Trump PA 2970733 6115402 0.486  
## 6 <NA> IL Hillary Clinton IL 3090729 5523142 0.560

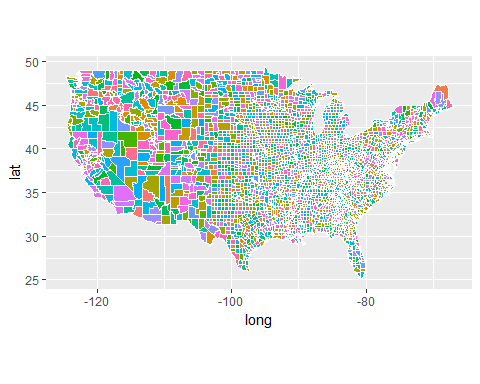
head(county\_winner)

## # A tibble: 6 x 7  
## # Groups: fips [6]  
## county fips candidate state votes total pct  
## <fct> <fct> <fct> <fct> <int> <int> <dbl>  
## 1 Los Angeles County 6037 Hillary Clinton CA 2464364 3421533 0.720  
## 2 Cook County 17031 Hillary Clinton IL 1611946 2156395 0.748  
## 3 Maricopa County 4013 Donald Trump AZ 747361 1536743 0.486  
## 4 Harris County 48201 Hillary Clinton TX 707914 1305434 0.542  
## 5 San Diego County 6073 Hillary Clinton CA 735476 1291078 0.570  
## 6 Orange County 6059 Hillary Clinton CA 609961 1186203 0.514

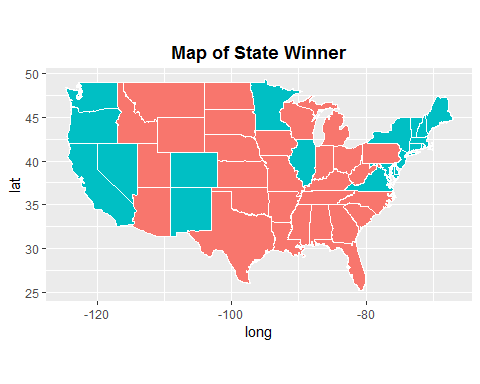
states = map\_data("state")  
ggplot(data = states) +  
geom\_polygon(aes(x = long, y = lat, fill = region, group = group), color = "white") +  
coord\_fixed(1.3) +  
guides(fill=FALSE) # color legend is unnecessary and takes too long



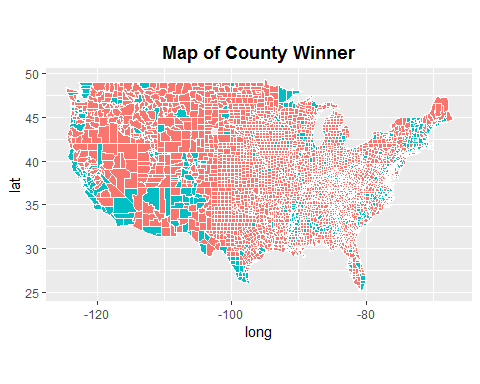
counties = map\_data("county")  
ggplot(data = counties) +  
geom\_polygon(aes(x = long, y = lat, fill = subregion, group = group), color = "white") +  
coord\_fixed(1.3) +  
guides(fill=FALSE)



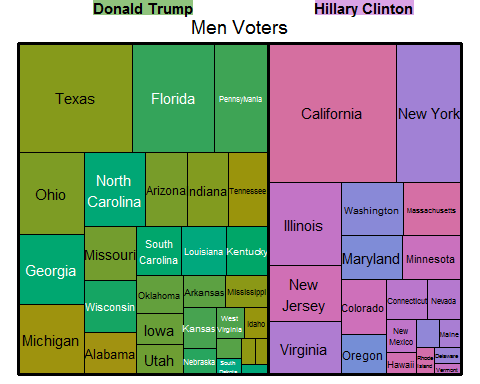
states = map\_data("state")  
states=states%>%mutate(fips=state.abb[match(region,tolower(state.name))])%>%left\_join(state\_winner)  
#state\_winner$state=as.character(state\_winner$state)  
#states=left\_join(states,state\_winner,by=c("fips"="state"))  
ggplot(data = states) +  
geom\_polygon(aes(x = long, y = lat, fill = candidate, group = group), color = "white") +  
coord\_fixed(1.3) +ggtitle("Map of State Winner")+theme(plot.title=element\_text(hjust=0.5,face="bold"))+  
guides(fill=FALSE) # color legend is unnecessary and takes too long



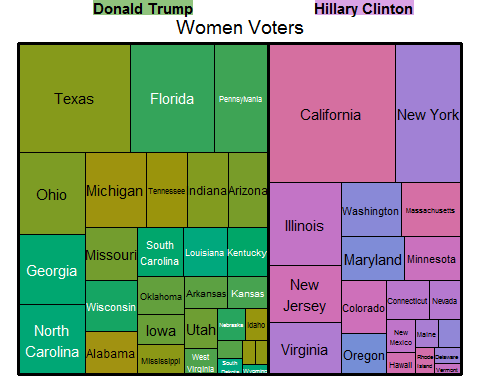
county.fips=county.fips%>%separate(col="polyname",c("state","county"),sep=",")  
counties=left\_join(counties,county.fips,by=c("region"="state","subregion"="county"))  
counties$fips=as.character(counties$fips)  
county\_winner$fips=as.character(county\_winner$fips)  
county=left\_join(county\_winner,counties,by=c("fips"="fips"))  
ggplot(data = county) +  
geom\_polygon(aes(x = long, y = lat, fill = candidate, group = fips), color = "white") +  
coord\_fixed(1.3) +ggtitle("Map of County Winner")+theme(plot.title=element\_text(hjust=0.5,face="bold"))+  
guides(fill=FALSE)



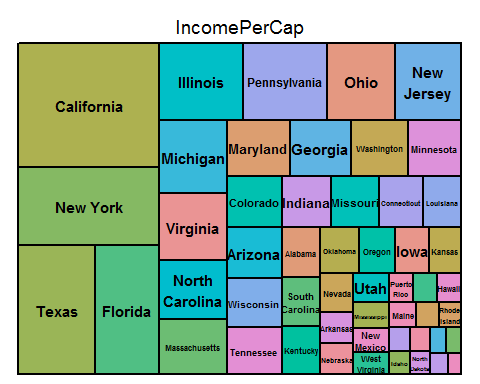
census\_state=census%>%na.omit()%>%mutate(state = state.abb[match(State, state.name)]) %>% left\_join(state\_winner)  
treemap(census\_state,index=c("candidate","State"),vSize="Men",type="index",title="Men Voters",ymod.label=c(2.1,0))



treemap(census\_state,index=c("candidate","State"),vSize="Women",type="index",title="Women Voters",ymod.label=c(2.1,0))



treemap(census,index="State",vSize="IncomePerCap",type="index")



census.del=census%>%  
 na.omit()%>%  
 dplyr::mutate(Employed=Employed\*100/(Men+Women),  
 Citizen=Citizen\*100/(Men+Women),  
 Men=Men\*100/(Men+Women),  
 Minority=Hispanic+Black+Native+Asian+Pacific)%>%  
 dplyr::select(-c("Hispanic","Black","Native","Asian","Pacific","Women","Walk","PublicWork","Construction"))  
   
census.subct=census.del%>%  
 dplyr::group\_by(State,County)%>%  
 add\_tally(wt=TotalPop)%>%  
 mutate(CountyTotal=n,Weight=TotalPop/CountyTotal)%>%  
 select(-n)  
census.ct=census.subct%>%  
 dplyr::group\_by(State,County)%>%  
 dplyr::summarize\_at(c(3:27),funs(round(sum(.\*Weight),2)))

## Warning: funs() is soft deprecated as of dplyr 0.8.0  
## please use list() instead  
##   
## # Before:  
## funs(name = f(.))  
##   
## # After:   
## list(name = ~ f(.))  
## This warning is displayed once per session.

head(census.ct)

## # A tibble: 6 x 27  
## # Groups: State [1]  
## State County Men White Citizen Income IncomeErr IncomePerCap  
## <fct> <fct> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 Alab~ Autau~ 48.4 75.8 73.8 51696. 7771. 24974.  
## 2 Alab~ Baldw~ 48.8 83.1 75.7 51074. 8745. 27317.  
## 3 Alab~ Barbo~ 53.8 46.2 76.9 32959. 6031. 16824.  
## 4 Alab~ Bibb 53.4 74.5 77.4 38887. 5662. 18431.  
## 5 Alab~ Blount 49.4 87.8 73.4 46238. 8696. 20532.  
## 6 Alab~ Bullo~ 53.0 22.2 75.4 33293. 9000. 17580.  
## # ... with 19 more variables: IncomePerCapErr <dbl>, Poverty <dbl>,  
## # ChildPoverty <dbl>, Professional <dbl>, 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>, Unemployment <dbl>, Minority <dbl>

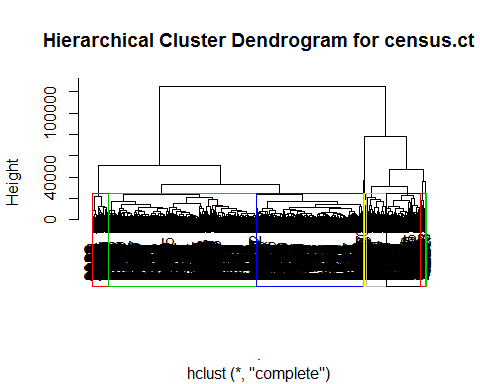
ct.pc=census.ct%>%  
 ungroup%>%  
 dplyr::select(Men:Minority)%>%  
 prcomp(scale=TRUE,center=TRUE)  
  
subct.pc=census.subct%>%  
 ungroup%>%  
 dplyr::select(Men:Minority)%>% #Totalpop?  
 prcomp(scale=TRUE,center=TRUE)  
ct.pca=ct.pc$rotation[,1:2]  
subct.pca=subct.pc$rotation[,1:2]  
  
index\_ct=order(abs(ct.pca[,1]),decreasing=TRUE)  
index\_subct=order(abs(subct.pca[,1]),decreasing=TRUE)  
ct.pca=ct.pca[index\_ct,]  
subct.pca=subct.pca[index\_subct,]  
ct.pca

## PC1 PC2  
## IncomePerCap 0.351554149 0.06774833  
## ChildPoverty -0.343598516 -0.05151568  
## Poverty -0.342308398 -0.07877015  
## Employed 0.327837484 0.04408699  
## Income 0.320033809 0.13541019  
## Unemployment -0.290698724 0.02929755  
## Professional 0.250165786 -0.11120410  
## Minority -0.226531530 -0.04583274  
## White 0.223026235 0.04854335  
## IncomePerCapErr 0.194524378 0.03709535  
## Service -0.181802543 -0.09568071  
## WorkAtHome 0.174893904 -0.37459686  
## IncomeErr 0.169869206 0.14264964  
## Production -0.118759249 0.18642678  
## SelfEmployed 0.097754207 -0.39843550  
## Drive -0.094144381 0.36318698  
## Carpool -0.076714956 -0.06565115  
## Transit 0.070752693 0.01459493  
## MeanCommute -0.058604318 0.22788393  
## PrivateWork 0.056472885 0.44667344  
## FamilyWork 0.048683798 -0.27489064  
## Office -0.014999663 0.25846663  
## OtherTransp -0.009661447 -0.15281350  
## Men 0.006812396 -0.16893052  
## Citizen 0.004687668 -0.04178918

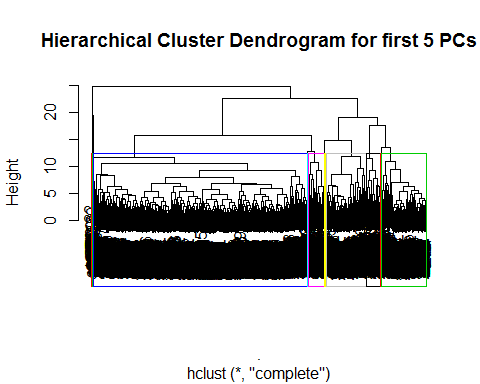
subct.pca

## PC1 PC2  
## IncomePerCap 0.31884859 -0.178244989  
## Professional 0.30686496 -0.154052613  
## Poverty -0.30444951 -0.076564983  
## Income 0.30250072 -0.150118580  
## ChildPoverty -0.29772431 -0.047486186  
## Service -0.26881316 -0.072117588  
## Unemployment -0.25260334 -0.088744162  
## Minority -0.24175412 -0.305499716  
## White 0.24016342 0.308680193  
## Employed 0.22131412 -0.039399388  
## IncomePerCapErr 0.21354592 -0.218871311  
## Production -0.20705014 0.207928331  
## IncomeErr 0.19977349 -0.233208031  
## WorkAtHome 0.17388608 -0.119399969  
## Carpool -0.16297605 0.042373782  
## Citizen 0.16133721 0.217143677  
## Drive 0.07776443 0.436917687  
## SelfEmployed 0.07114733 -0.042856495  
## Transit -0.05635290 -0.437751840  
## OtherTransp -0.04464935 -0.162830018  
## PrivateWork -0.04261844 -0.003276937  
## Men 0.01740104 0.041612414  
## FamilyWork 0.01554012 0.027351590  
## Office -0.01443059 0.084172911  
## MeanCommute 0.01026841 -0.285348602

ct\_hc=census.ct%>%   
 ungroup()%>%   
 dplyr::select(-c(State,County))%>%  
 dist(scale(.),method="euclidean")%>%  
 hclust(method="complete")  
ct.pc\_hc=ct.pc$x[,1:5]%>%  
 dist(scale(.),method="euclidean")%>%  
 hclust(method="complete")  
  
  
plot(ct\_hc,main="Hierarchical Cluster Dendrogram for census.ct")  
rect.hclust(ct\_hc,k=10, border = 2:11)



ct.cut=cutree(ct\_hc,k=10)  
  
plot(ct.pc\_hc,main="Hierarchical Cluster Dendrogram for first 5 PCs")  
rect.hclust(ct.pc\_hc,k=10, border = 2:11)



ct.pc.cut=cutree(ct.pc\_hc,10)

index1=ct.cut[which(census.ct$County=="San Mateo")]  
index2=ct.pc.cut[which(census.ct$County=="San Mateo")]  
ct.M=census.ct[which(ct.cut==index1),]  
ct.pc.M=census.ct[which(ct.pc.cut==index2),]  
ct.M\_ave=ct.M%>%ungroup%>%select(c(Men:Minority))%>%  
   
 summarize\_at(vars(Men:Minority),mean)  
  
ct.pc.M\_ave=ct.pc.M%>%ungroup%>%summarize\_at(vars(Men:Minority),mean)  
ct.M\_ave

## # A tibble: 1 x 25  
## Men White Citizen Income IncomeErr IncomePerCap IncomePerCapErr Poverty  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 49.2 67.7 68.7 1.00e5 14146. 45278. 5545. 6.65  
## # ... with 17 more variables: ChildPoverty <dbl>, Professional <dbl>,  
## # 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>, Unemployment <dbl>,  
## # Minority <dbl>

ct.pc.M\_ave

## # A tibble: 1 x 25  
## Men White Citizen Income IncomeErr IncomePerCap IncomePerCapErr Poverty  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 49.4 73.0 72.2 69597. 10203. 32873. 4210. 11.0  
## # ... with 17 more variables: ChildPoverty <dbl>, Professional <dbl>,  
## # 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>, Unemployment <dbl>,  
## # Minority <dbl>

SanM=census.ct%>%ungroup%>%filter(County=="San Mateo")%>%select(c(Men:Minority))  
compare=data.frame(rbind(SanM,ct.M\_ave,ct.pc.M\_ave))  
row.names(compare)=c("San Mateo","Cluster Mean","PC Cluster Mean")  
compare

## Men White Citizen Income IncomeErr  
## San Mateo 49.20000 40.64000 64.20000 100369.92 16123.02  
## Cluster Mean 49.21115 67.74385 68.69962 100257.27 14146.49  
## PC Cluster Mean 49.44953 73.02311 72.17606 69597.31 10202.79  
## IncomePerCap IncomePerCapErr Poverty ChildPoverty  
## San Mateo 47881.29 6115.550 8.010000 9.710000  
## Cluster Mean 45278.06 5544.617 6.651923 7.982308  
## PC Cluster Mean 32873.33 4209.829 10.953731 13.903264  
## Professional Service Office Production Drive  
## San Mateo 45.74000 18.29000 22.30000 7.340000 69.93000  
## Cluster Mean 49.25231 14.97154 22.71115 6.439231 74.22846  
## PC Cluster Mean 38.60298 17.13583 23.98117 10.513057 78.67938  
## Carpool Transit OtherTransp WorkAtHome MeanCommute  
## San Mateo 10.680000 9.260000 2.600000 5.080000 26.83000  
## Cluster Mean 8.365769 7.783462 1.511154 5.764231 31.06462  
## PC Cluster Mean 9.269793 2.910984 1.601658 4.930518 26.33808  
## Employed PrivateWork SelfEmployed FamilyWork Unemployment  
## San Mateo 51.72000 79.77000 8.370000 0.1700000 6.690000  
## Cluster Mean 51.38731 77.59154 5.817308 0.1338462 5.912692  
## PC Cluster Mean 49.29461 78.15329 6.169819 0.1536528 6.707176  
## Minority  
## San Mateo 55.53000  
## Cluster Mean 29.80115  
## PC Cluster Mean 24.64487

#Classification

tmpwinner = county\_winner %>% ungroup %>%  
mutate(state = state.name[match(state, state.abb)]) %>% ## state abbreviations  
mutate\_at(vars(state, county), tolower) %>% ## to all lowercase  
mutate(county = gsub(" county| columbia| city| parish", "", county)) ## remove suffixes  
tmpcensus = census.ct %>%ungroup%>% mutate\_at(vars(State, County), tolower)  
election.cl = tmpwinner %>%  
left\_join(tmpcensus, by = c("state"="State", "county"="County")) %>%  
na.omit  
## saves meta information to attributes  
attr(election.cl, "location") = election.cl %>% select(c(county, fips, state, votes, pct))  
election.cl = election.cl %>% select(-c(county, fips, state, votes, pct))

set.seed(10)  
n = nrow(election.cl)  
in.trn= sample.int(n, 0.8\*n)  
trn.cl = election.cl[ in.trn,]  
tst.cl = election.cl[-in.trn,]

set.seed(20)  
nfold = 10  
folds = sample(cut(1:nrow(trn.cl), breaks=nfold, labels=FALSE))  
  
  
calc\_error\_rate = function(predicted.value, true.value){  
return(mean(true.value!=predicted.value))  
}  
records = matrix(NA, nrow=3, ncol=2)  
colnames(records) = c("train.error","test.error")  
rownames(records) = c("tree","knn","lda")

XTrain=trn.cl%>%select(-candidate)  
YTrain=trn.cl$candidate  
XTest=tst.cl%>%select(-candidate)  
YTest=tst.cl$candidate

do.chunk <- function(chunkid, folddef, Xdat, Ydat, k){  
train = (folddef!=chunkid)  
Xtr = Xdat[train,]  
Ytr = Ydat[train]  
Xvl = Xdat[!train,]  
Yvl = Ydat[!train]  
## get classifications for current training chunks  
predYtr = knn(train = Xtr, test = Xtr, cl = Ytr, k = k)  
## get classifications for current test chunk  
predYvl = knn(train = Xtr, test = Xvl, cl = Ytr, k = k)  
data.frame(train.error = calc\_error\_rate(predYtr, Ytr),  
val.error = calc\_error\_rate(predYvl, Yvl))  
}

#setup=tree.control(nobs=nrow(trn.cl),minsize=5,mindev=1e-5)  
tree.cl=tree(candidate~.,data=trn.cl)  
summary(tree.cl)

##   
## Classification tree:  
## tree(formula = candidate ~ ., data = trn.cl)  
## Variables actually used in tree construction:  
## [1] "Transit" "White" "Unemployment" "Citizen"   
## [5] "Professional" "Drive" "Employed" "Production"   
## [9] "total"   
## Number of terminal nodes: 15   
## Residual mean deviance: 0.3191 = 778.8 / 2441   
## Misclassification error rate: 0.05741 = 141 / 2456

treecv=cv.tree(tree.cl,K=nfold,rand=folds,method="misclass")  
tree.cl\_size=min(treecv$size[treecv$dev==min(treecv$dev)])  
tree.cl\_pruned=prune.tree(tree.cl,best=tree.cl\_size,method="misclass")  
pred.train=predict(tree.cl\_pruned,trn.cl,type="class")  
pred.test=predict(tree.cl\_pruned,tst.cl,type="class")  
train.err1=calc\_error\_rate(pred.train,YTrain)  
test.err1=calc\_error\_rate(pred.test,YTest)  
records[1,]=c(train.err1,test.err1)  
records

## train.error test.error  
## tree 0.06107492 0.07003257  
## knn NA NA  
## lda NA NA

kv=c(1, seq(5, 50, length.out=10))  
error.folds=NULL  
for (j in kv){  
tmp = ldply(1:nfold, do.chunk, folddef=folds, Xdat=XTrain,Ydat=YTrain,k = j)  
error.folds = rbind(error.folds, tmp)  
}  
  
num=c(rep(0:10,each=10))  
df=cbind(num,error.folds)  
colnames(df)=c("N","train.error","test.error")  
average=df%>%  
 group\_by(N) %>%  
 summarise\_at(.funs=funs(mean),.var=vars(train.error,test.error))  
best\_n=min(average$N[average$test.error==min(average$test.error)])  
knn\_pred\_train=knn(train=XTrain,test=XTrain,cl=YTrain,k=best\_n)  
knn\_pred\_test=knn(train=XTrain,test=XTest,cl=YTrain,k=best\_n)  
train.err2=calc\_error\_rate(knn\_pred\_train,YTrain)  
test.err2=calc\_error\_rate(knn\_pred\_test,YTest)  
records[2,]=c(train.err2,test.err2)  
records

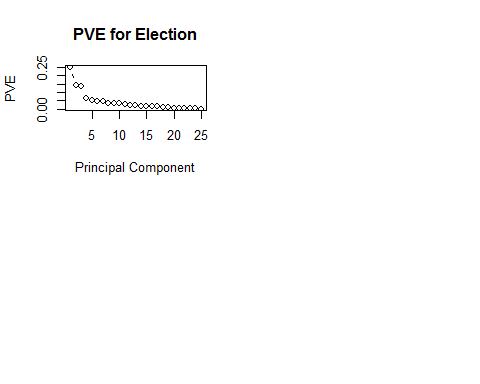
## train.error test.error  
## tree 0.06107492 0.07003257  
## knn 0.08265472 0.14820847  
## lda NA NA

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

trn.pca=trn.cl%>%  
 dplyr::select(Men:Minority)%>%  
 prcomp(scale=TRUE,center=TRUE)  
#names(trn.pca)  
#trn.pca$rotation  
#dim(trn.pca$x)  
trn.dev=trn.pca$sdev  
trn\_pve=trn.dev^2/sum(trn.dev^2)  
trn\_cumu\_pve=cumsum(trn\_pve)  
min.trn.pc=min(which(trn\_cumu\_pve > 0.9))  
print(min.trn.pc)

## [1] 13

par(mfrow=c(2, 2))  
plot(trn\_pve,xlab = "Principal Component",ylab = "PVE",main="PVE for Election",type="b")  
#plot(ele\_cumu\_pve, type="l", lwd=3,  
# xlab = "Principal Component",  
# ylab = "Cumulative PVE",ylim = c(0,1),main="Cumulative PVE for county data")



tr.pca=data.frame(candidate=trn.cl$candidate,trn.pca$x)  
test.pca=tst.cl%>%  
 dplyr::select(Men:Minority)%>%  
 prcomp(scale=TRUE,center=TRUE)  
test.pca=data.frame(candidate=tst.cl$candidate,test.pca$x)  
tr.pca=tr.pca%>%  
 dplyr::rename(Men=PC1,White=PC2,Citizen=PC3,Income=PC4,  
 IncomeErr=PC5,IncomePerCap=PC6,IncomePerCapErr=PC7,  
 Poverty=PC8,ChildPoverty=PC9,Professional=PC10,  
 Service=PC11,Office=PC12,Production=PC13,Drive=PC14,  
 Carpool=PC15,Transit=PC16,OtherTransp=PC17,  
 WorkAtHome=PC18,MeanCommute=PC19,Employed=PC20,  
 PrivateWork=PC21,SelfEmployed=PC22,FamilyWork=PC23,  
 Unemployment=PC24,Minority=PC25)  
test.pca=test.pca%>%  
 dplyr::rename(Men=PC1,White=PC2,Citizen=PC3,Income=PC4,  
 IncomeErr=PC5,IncomePerCap=PC6,IncomePerCapErr=PC7,  
 Poverty=PC8,ChildPoverty=PC9,Professional=PC10,  
 Service=PC11,Office=PC12,Production=PC13,Drive=PC14,  
 Carpool=PC15,Transit=PC16,OtherTransp=PC17,  
 WorkAtHome=PC18,MeanCommute=PC19,Employed=PC20,  
 PrivateWork=PC21,SelfEmployed=PC22,FamilyWork=PC23,  
 Unemployment=PC24,Minority=PC25)

tre.trn.pca=tree(candidate~.,data=tr.pca)  
summary(tre.trn.pca)

##   
## Classification tree:  
## tree(formula = candidate ~ ., data = tr.pca)  
## Variables actually used in tree construction:  
## [1] "Citizen" "Drive" "OtherTransp" "White"   
## [5] "Poverty" "PrivateWork" "Men" "IncomePerCap"  
## Number of terminal nodes: 17   
## Residual mean deviance: 0.4358 = 1063 / 2439   
## Misclassification error rate: 0.07695 = 189 / 2456

trecv.pca=cv.tree(tre.trn.pca,K=nfold,rand=folds,method="misclass")  
tre.cl.pca\_size=min(trecv.pca$size[trecv.pca$dev==min(trecv.pca$dev)])  
tre.cl.pca\_pruned=prune.tree(tre.trn.pca,  
 best=tre.cl.pca\_size,method="misclass")  
tre\_pca.pred.train=predict(tre.cl.pca\_pruned,tr.pca,type="class")  
tre\_pca.pred.test=predict(tre.cl.pca\_pruned,test.pca,type="class")  
train.cpa.err1=calc\_error\_rate(tre\_pca.pred.train,YTrain)  
test.cpa.err1=calc\_error\_rate(tre\_pca.pred.test,YTest)  
pca.records[1,]=c(train.err1,test.err1)  
pca.records

## train.error test.error  
## tree 0.06107492 0.07003257  
## knn NA NA  
## lda NA NA

#error of PCA summary(ele.pca) //tr.pca=tr.pca[,1:14] #13 PCAs explained 90%

tree.pca=tree(candidate~.,data=tr.pca) summary(tree.pca) treecv.pca=cv.tree(tree.pca,K=nfold,rand=folds,method=“misclass”) tree.pca\_size=min(treecv.pcadev==min(treecv.pca$dev)]) tree.pca\_pruned=prune.tree(tree.pca,best=tree.pca\_size,method=“misclass”) draw.tree(tree.pca\_pruned,nodeinfo=TRUE)

tre.test=predict(ele.pca,newdata=tst.cl) tre.test=as.data.frame(tre.test) tre.test=tre.test%>% dplyr::rename(Men=PC1,White=PC2,Citizen=PC3,Income=PC4, IncomeErr=PC5,IncomePerCap=PC6,IncomePerCapErr=PC7, Poverty=PC8,ChildPoverty=PC9,Professional=PC10, Service=PC11,Office=PC12,Production=PC13,Drive=PC14, Carpool=PC15,Transit=PC16,OtherTransp=PC17, WorkAtHome=PC18,MeanCommute=PC19,Employed=PC20, PrivateWork=PC21,SelfEmployed=PC22,FamilyWork=PC23, Unemployment=PC24,Minority=PC25) //tre.test.data=tre.test.data[,1:13]#select first 13 components tre.pca.pred=predict(tree.pca\_pruned,tst.cl,type=“class”)

tre.pca.train=predict(tree.pca\_pruned,trn.cl,type=“class”)

train.cpa.err1=calc\_error\_rate(tre.pca.train,YTrain) test.cpa.err1=calc\_error\_rate(tre.pca.pred,YTest) pca.records[1,]=c(train.cpa.err1,test.cpa.err1) pca.records #

XTrain.pca=tr.pca%>%select(-candidate)  
YTrain.pca=tr.pca$candidate  
XTest.pca=test.pca%>%select(-candidate)  
YTest.ca=test.pca$candidate

kv=c(1, seq(5, 50, length.out=10))  
error.folds=NULL  
for (j in kv)  
{  
 tmp=ldply(1:nfold,do.chunk, folddef=folds, Xdat=XTrain.pca,  
 Ydat=YTrain.pca,k = j)  
 error.folds = rbind(error.folds, tmp)  
}  
  
num=c(rep(0:10,each=10))  
df.pca=cbind(num,error.folds)  
colnames(df.pca)=c("N","train.error","test.error")  
ave.pca=df.pca%>%  
 group\_by(N) %>%  
 summarise\_at(.funs=funs(mean),.var=vars(train.error,test.error))  
best\_n.pca=min(ave.pca$N[ave.pca$test.error==min(ave.pca$test.error)])  
knn\_pred\_tr.pca=knn(train=XTrain.pca,test=XTrain.pca,  
 cl=YTrain.pca,k=best\_n.pca)  
knn\_pred\_tst.pca=knn(train=XTrain.pca,test=XTest.pca,  
 cl=YTrain.pca,k=best\_n.pca)  
train.pca.err2=calc\_error\_rate(knn\_pred\_train,YTrain)  
test.pca.err2=calc\_error\_rate(knn\_pred\_test,YTest)  
pca.records[2,]=c(train.pca.err2,test.pca.err2)  
pca.records

## train.error test.error  
## tree 0.06107492 0.07003257  
## knn 0.08265472 0.14820847  
## lda NA NA

1. During modeling the