Final Project - PSTAT 231

Elijah Castro - 3646908 and Cyrus Tabatabai-Yazdi - 8279861

December 06, 2022

Data

In order to analyze and visualize the 2020 presidential election, we will start by loading in two data sets. The first data set is election data, which contains county-level election results and will be called `election.raw`. The second data set contains the 2017 United States county-level census data and will be called `census`.

Election Data

1.

Below, we can see the dimension of the `election.raw` data set:

32177 and *5*

This means it contains 32177 rows and 5 columns.

Next, from the table below, we can see there are no missing values in the data set.

state	county	candidate	party	total_votes
0	0	0	0	0

Lastly, we compute the total number of distinct values in the `state` column in `election.raw` to verify that the data contains all states and a federal district. We confirm that this is indeed the case:

51

Census Data

2.

Below, we can see the dimension of the `census` data set:

3220 and *37*

This means it contains 3220 rows and 37 columns.

Next, from the table below, we can see there is one missing value in the data set. This is located in the `ChildPoverty` column.

name	value
ChildPoverty	1

We now compute the total number of distinct values in the column `county` in the `census` data set:

3220

Lastly, if we were to compare the values of the total number of distinct counties in `census` with that in `election.raw`, we see that the values are much different. `election.raw` has less distinct counties than `census`.

2825

This can most likely be attributed to the fact that `election.raw` does count the total number of distinct counties, but many states share common county names. Thus, we are only including one of the counties with the same name, when in fact, we should also be counting the other state's county that has the same name. We remove this issue with having a unique county ID for each county in the `census` data set.

Data Wrangling

3.

Here, we will construct aggregated data sets from `election.raw` data.

First is county-level data, which is keeping `election.raw` as is. Below shows the first few rows of the vote count by county:

state	county	candidate	party	total_votes
Delaware	Kent	Joe Biden	DEM	44552
Delaware	Kent	Donald Trump	REP	41009
Delaware	Kent	Jo Jorgensen	LIB	1044
Delaware	Kent	Howie Hawkins	GRN	420
Delaware	New Castle	Joe Biden	DEM	195034
Delaware	New Castle	Donald Trump	REP	88364

Next, we create a state-level summary called `election.state`, and its first few rows are shown as follows:

state	candidate	party	total_votes
Alabama	Donald Trump	REP	1441168
Alabama	Jo Jorgensen	LIB	25176
Alabama	Joe Biden	DEM	849648
Alabama	Write-ins	WRI	7312
Alaska	Brock Pierce	IND	825
Alaska	Don Blankenship	CST	1127

Lastly, we create a federal-level summary called `election.total`, and its first few rows are shown as follows:

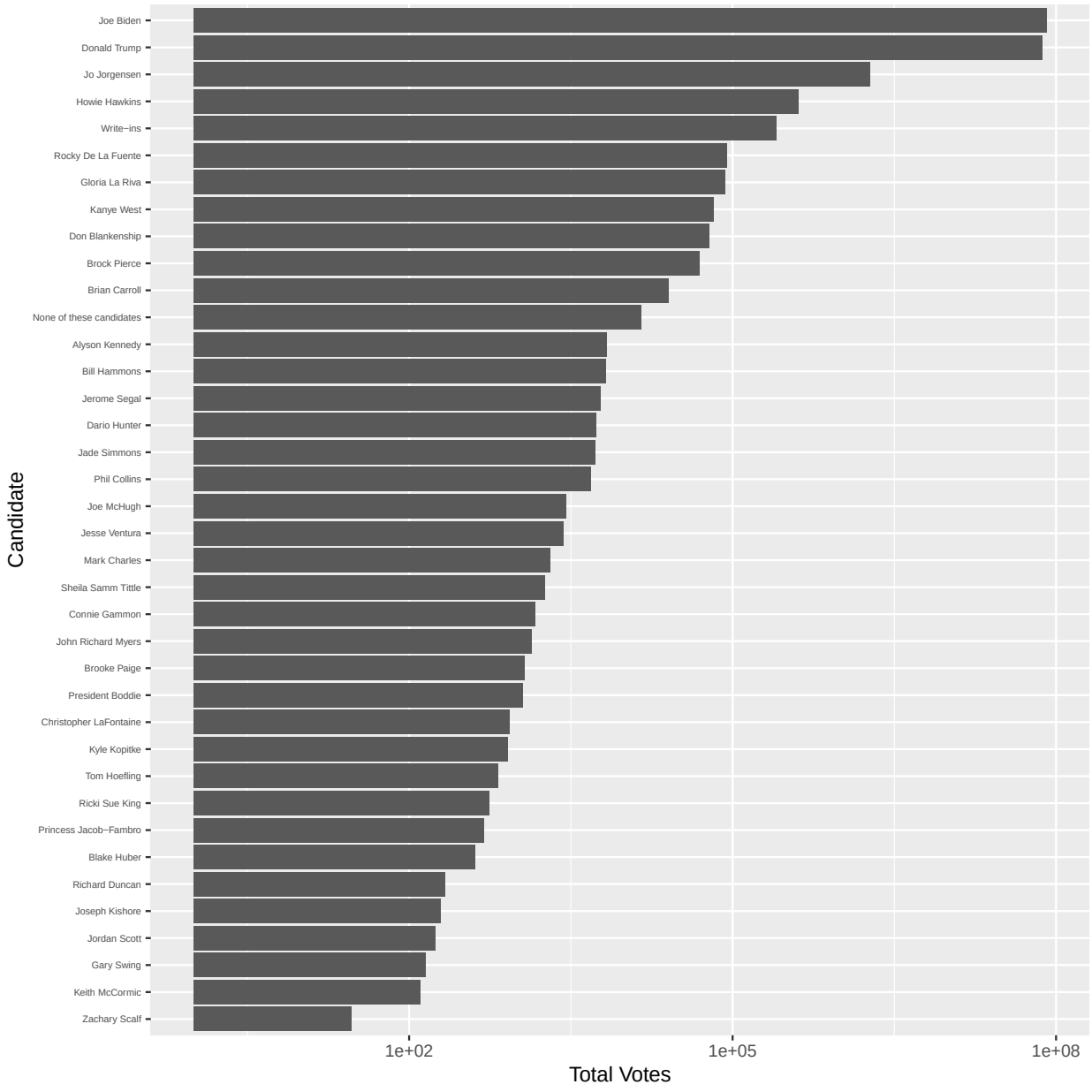
candidate	party	total_votes
Alyson Kennedy	SWP	6791
Bill Hammons	UTY	6647
Blake Huber	APV	409
Brian Carroll	ASP	25256
Brock Pierce	IND	49552
Brooke Paige	GOP	1175

4.

The number of named presidential candidates in the 2020 election is as follows:

38

Below is a bar chart of all votes received by each candidate:



5.

Here, we create data sets `county.winner` and `state.winner` by taking the candidate with the highest proportion of votes in both county level and state level respectively. Below are the first few rows of each new data set:

County Level

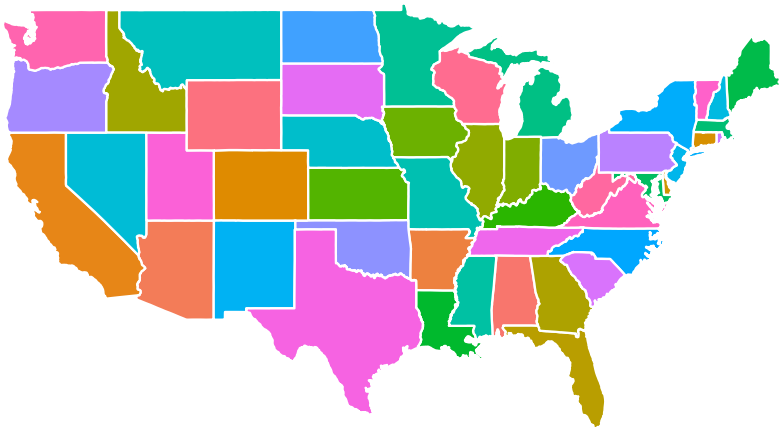
state	county	candidate	party	prop_votes
Alabama	Autauga	Donald Trump	REP	71.4
Alabama	Baldwin	Donald Trump	REP	76.2
Alabama	Barbour	Donald Trump	REP	53.5
Alabama	Bibb	Donald Trump	REP	78.4
Alabama	Blount	Donald Trump	REP	89.6
Alabama	Bullock	Joe Biden	DEM	74.7

State Level

state	candidate	party	prop_votes
Alabama	Donald Trump	REP	62.03
Alaska	Donald Trump	REP	48.52
Arizona	Joe Biden	DEM	49.37
Arkansas	Donald Trump	REP	62.4
California	Joe Biden	DEM	63.5
Colorado	Joe Biden	DEM	55.4

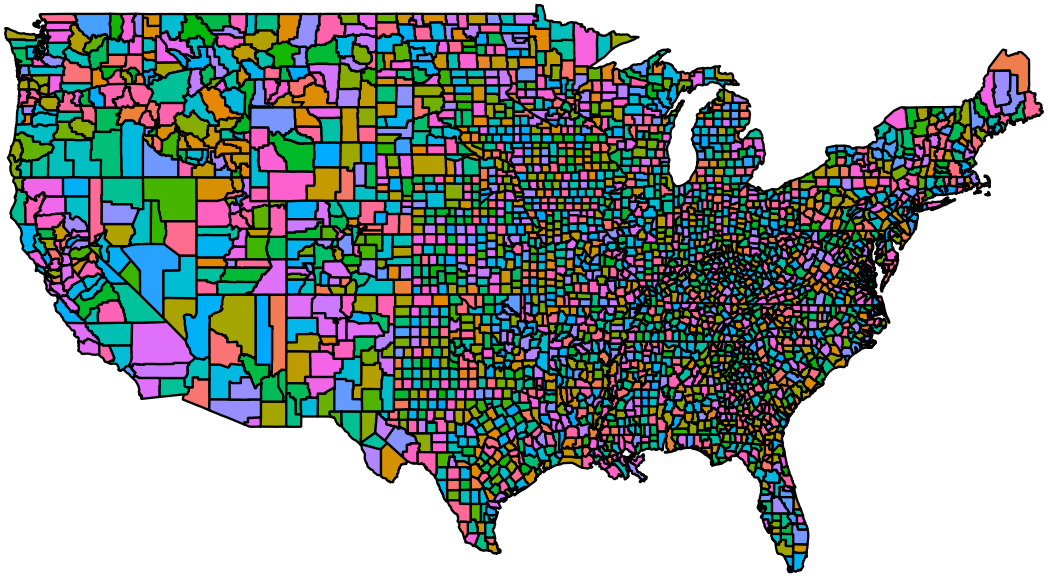
Visualization

Below is a visual of a state-level map colored by state.



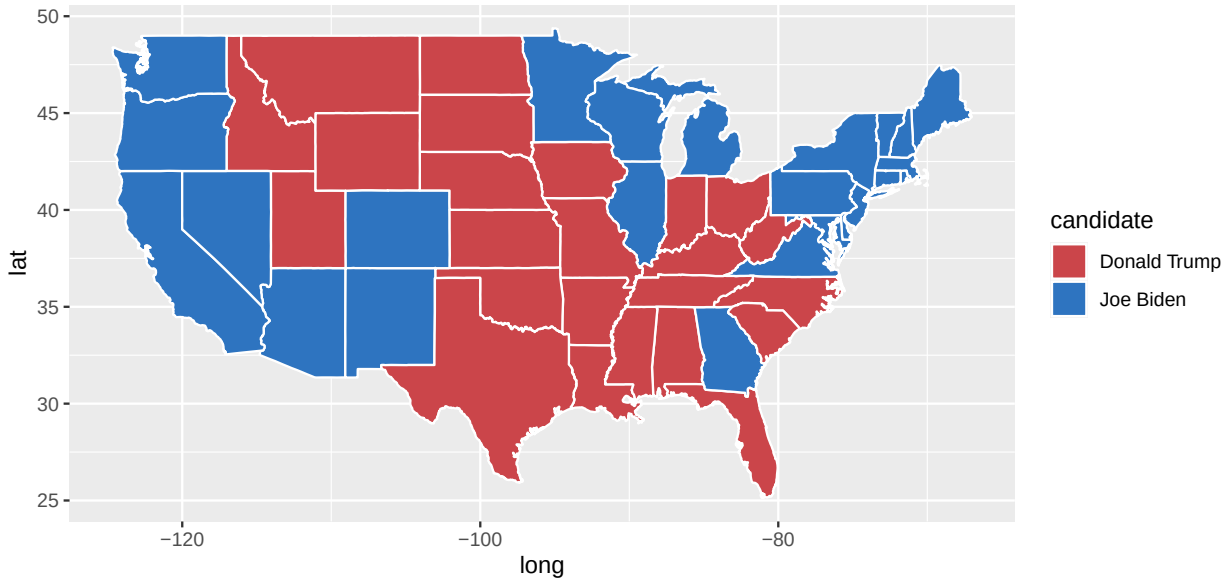
6.

Similar to the map above, we now create a map based on county-level and color by county.



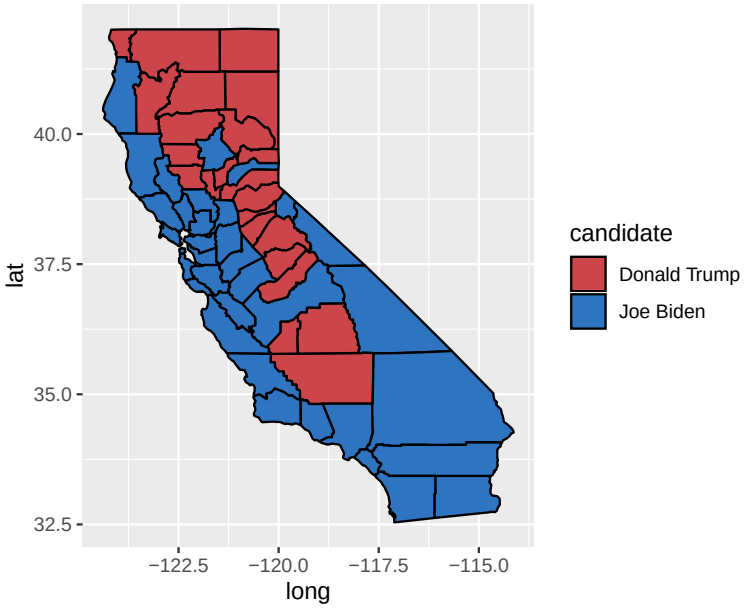
7.

Now, at the state-level, we create a map colored by the winning candidate for each state.



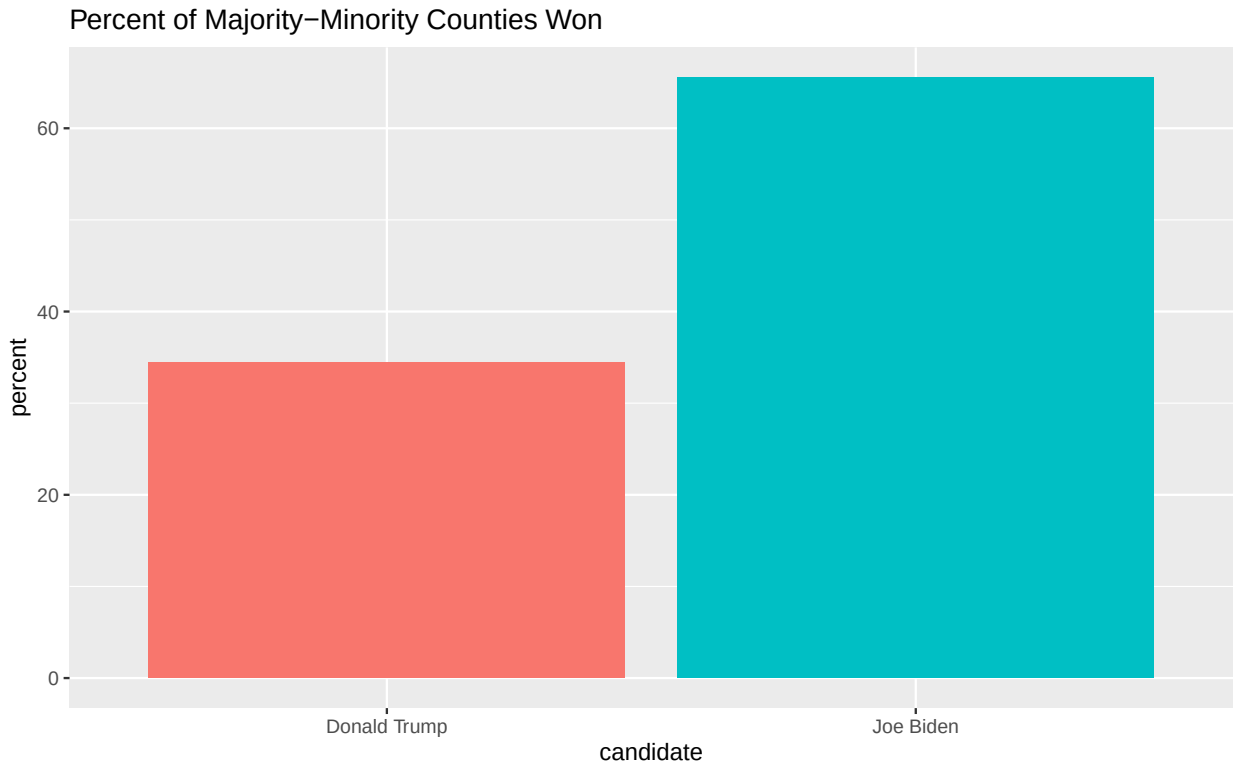
8.

Finally, at the county-level, we create a map of California colored by the winning candidate for each county.



9.

For our additional visualization of choice using the `census` data, we created a bar plot of the percent of majority-minority counties won for each candidate, where majority-minority describes counties where the minority population is the majority, or in other words, the minority population is greater than 50% of the total population in any respective county. We can see Joe Biden won more often in these counties than Donald Trump did during this election.



10.

Here, we want to clean county-level census data in the `census` data set. We do this by filtering out any rows with missing values, converting {`Men`, `Employed`, `VotingAgeCitizen`} attributes to percentages, computing `Minority` attribute by combining {`Hispanic`, `Black`, `Native`, `Asian`, `Pacific`}, removing these variables after creating `Minority`, and removing {`IncomeErr`, `IncomePerCap`, `IncomePerCapErr`, `Walk`, `PublicWork`, `Construction`}.

```
census.clean <- census %>%
  drop_na() %>%
  mutate(Men=Men/TotalPop * 100,
         Employed=Employed /TotalPop * 100,
          VotingAgeCitizen=VotingAgeCitizen/TotalPop * 100) %>%
  mutate(Minority=Hispanic + Black + Native + Asian + Pacific) %>%
  dplyr::select(-(Hispanic:Pacific),
                -c(IncomeErr,IncomePerCap,IncomePerCapErr,
                   Walk,PublicWork,Construction))
```

Below are the first five rows of this cleaned `census` data:

Table 8: Table continues below

CountyId	State	County	TotalPop	Men	Women
1001	Alabama	Autauga County	55036	48.88	28137
1003	Alabama	Baldwin County	203360	48.94	103833
1005	Alabama	Barbour County	26201	53.34	12225
1007	Alabama	Bibb County	22580	54.26	10329
1009	Alabama	Blount County	57667	49.4	29177

Table 9: Table continues below

VotingAgeCitizen	Income	Poverty	ChildPoverty	Professional	Service
74.53	55317	13.7	20.1	35.3	18
76.4	52562	11.8	16.1	35.7	18.2
77.36	33368	27.2	44.9	25	16.8
78.22	43404	15.2	26.6	24.4	17.6
73.72	47412	15.6	25.4	28.5	12.9

Table 10: Table continues below

Office	Production	Drive	Carpool	Transit	OtherTransp	WorkAtHome
23.2	15.4	86	9.6	0.1	1.3	2.5
25.6	10.8	84.7	7.6	0.1	1.1	5.6
22.6	24.1	83.4	11.1	0.3	1.7	1.3
19.7	22.4	86.4	9.5	0.7	1.7	1.5
23.3	19.5	86.8	10.2	0.1	0.4	2.1

Table 11: Table continues below

MeanCommute	Employed	PrivateWork	SelfEmployed	FamilyWork
25.8	43.81	74.1	5.6	0.1
27	44.02	80.7	6.3	0.1
23.4	33.88	74.1	6.5	0.3
30	36.19	76	6.3	0.3
35	37.07	83.9	4	0.1

Unemployment	Minority
5.2	22.8
5.5	15.4
12.4	52.8
8.2	24.8
4.9	10.9

Dimensionality Reduction

11.

We now use the county-level `census.cleaned` data to run PCA.

First, we save the first two principal components PC1 and PC2 into a two-column data frame called `pc.county`.

```
pr.out <- prcomp(census.clean[,4:ncol(census.clean)],scale=T,center=T)
pc.county <- pr.out$x[,1:2]
```

We chose to center and scale the data as the scales of the features and the variation are widely different due to some features representing percentages, and thus have a fixed width (0-100), while others like `Income` are on another scale of magnitude. This can be shown in the aggregated columns we did above.

Next, we can see the three features with the largest absolute values of the first principal component are:

ChildPoverty	Poverty	Employed
0.3899	0.3868	0.3729

Lastly, the features in PC1 that have opposite signs between them are shown below:

Features with Positive Loadings

TotalPop, *Men*, *Women*, *Income*, *Professional*, *Transit*, *WorkAtHome*, *Employed*, *PrivateWork*, *SelfEmployed* and *FamilyWork*

Features with Negative Loadings

VotingAgeCitizen, *Poverty*, *ChildPoverty*, *Service*, *Office*, *Production*, *Drive*, *Carpool*, *OtherTransp*, *MeanCommute*, *Unemployment* and *Minority*

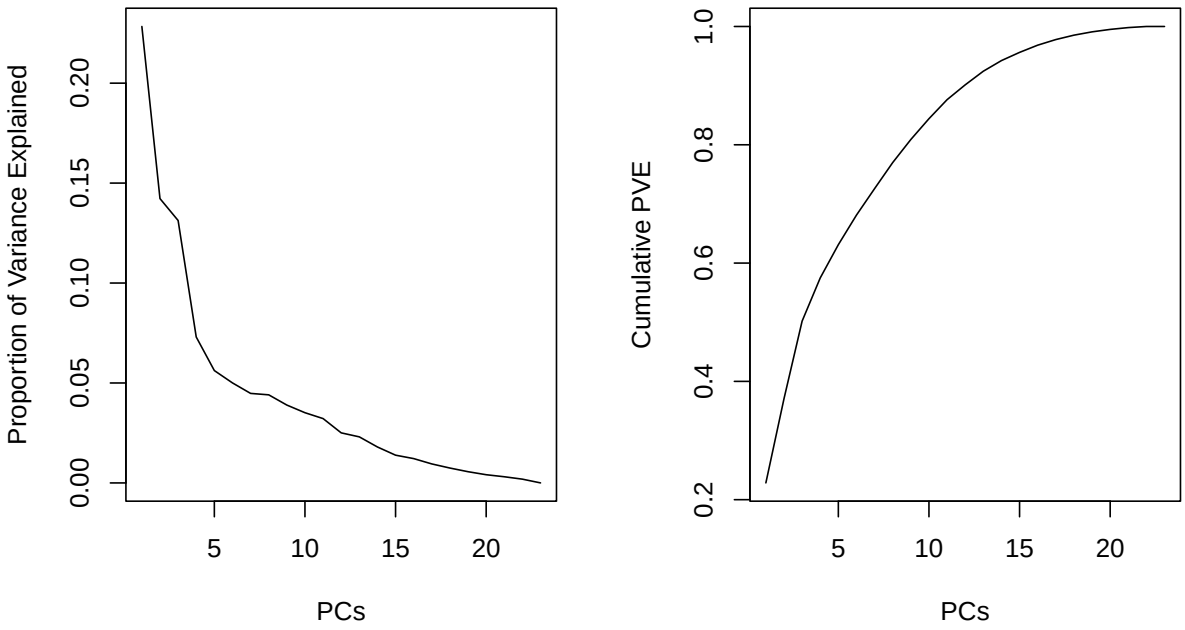
Correlation between these features depends on the signs that the loadings have, which was shown above. That is, the loadings with negative signs indicate that the features and the 1st principal component are negatively correlated, and the loadings with positive signs indicate that the features and the 1st principal component are positively correlated.

12.

First, we want to determine the minimum number of PCs needed to capture 90% of the variance for the analysis. Based on the result below, we need 12 principal components to explain 90% of the variation in the data.

12

Plots of proportion of variance explained (PVE) and cumulative PVE are also shown below:



Clustering

13.

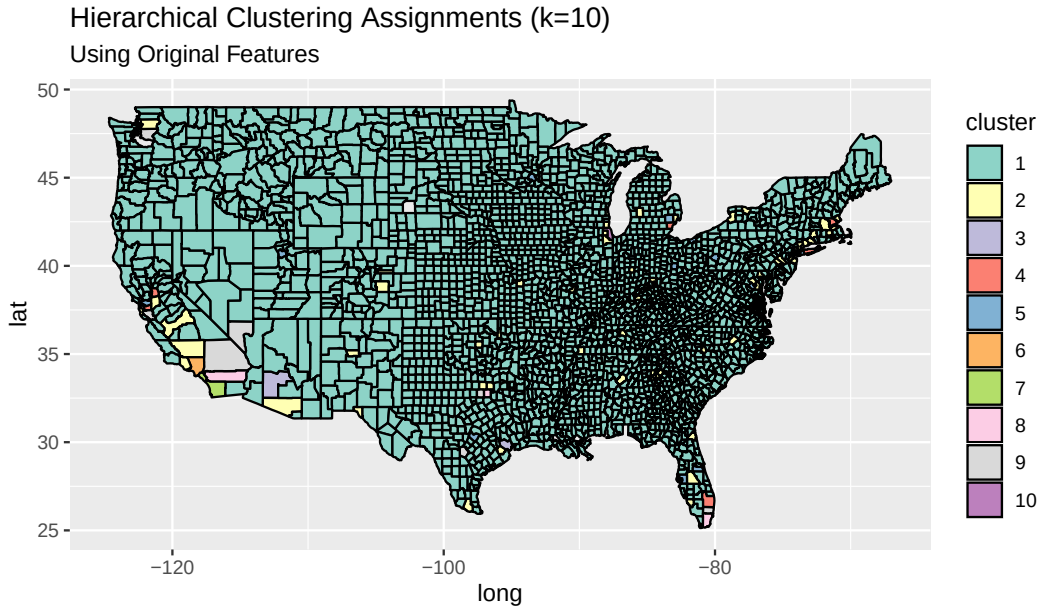
Before performing hierarchical clustering, we will first organize the data by adding the FIPS code to the counties map table in order to make the joins easier. Then, with the reorganized cleaned `census` data, we will now perform hierarchical clustering with complete linkage, and cut the tree to partition the observations into 10 clusters.

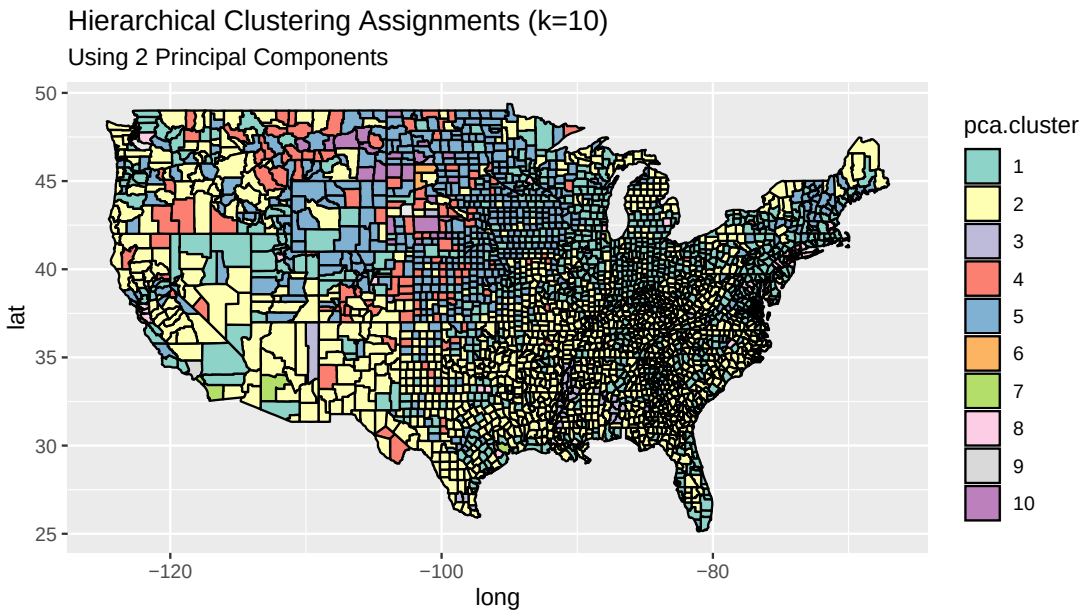
1	2	3	4	5	6	7	8	9	10
3111	69	2	9	12	1	2	5	7	1

Furthermore, we also re-run the hierarchical clustering algorithm using the first 2 principal components from `pc.county` as inputs instead of the original features.

1	2	3	4	5	6	7	8	9	10
907	1433	121	129	555	4	7	37	1	25

Although the first cluster still contains the bulk of the observations for both methods, we notice that the clusters are more spread using the first two principal components instead of being solely concentrated in the first cluster when the original features were used. We now visualize the assignments to gain more insight:





As the maps indicate, the clustering by the first two principal components leads to more variation in the assignment of clusters.

We now investigate the cluster that contains *Santa Barbara County* for both approaches.

Original Features

1

First 2 PCAs

1

Both approaches assigned Santa Barbara to the first cluster, which contains the vast majority of the observations in both methods. In order to get a better understanding for which approach is more appropriate for Santa Barbara, we analyze the descriptive statistics of the features within and between the two clusters.

Table 16: Table continues below

name	TotalPop_mean	Men_mean	Women_mean	VotingAgeCitizen_mean
cluster	58696	50.08	29692	75.24
pca.cluster	195181	49.41	99519	74.05

Table 17: Table continues below

Income_mean	Poverty_mean	ChildPoverty_mean	Professional_mean
48388	16.88	23.16	31.16
58240	12.5	17.04	33.87

Table 18: Table continues below

Service_mean	Office_mean	Production_mean	Drive_mean	Carpool_mean
18.22	21.81	16.03	79.89	9.882
17.08	23.33	15.26	81.88	9.043

Table 19: Table continues below

Transit_mean	OtherTransp_mean	WorkAtHome_mean	MeanCommute_mean
0.678	1.582	4.728	23.31
1.398	1.385	4.125	24.88

Table 20: Table continues below

Employed_mean	PrivateWork_mean	SelfEmployed_mean	FamilyWork_mean
43.24	74.65	7.849	0.2836
47.42	80.16	5.648	0.1622

Unemployment_mean	Minority_mean
6.665	22.31
5.651	19.07

It is hard to differentiate between the two approaches as many of the statistics are similar. However, for the principal component approach, some important statistics of interest are the mean total population and mean income being larger than the approach using the original features. This means for the principal components, larger counties are included inside the first cluster with populations above 100,000 as well as a higher income. These two measures make sense for a county like Santa Barbara as this is a bigger county, in terms of population, relative to most counties in the U.S., and it has a high median income due to the high cost of living and a larger economy.

We next limit to only counties within California in order to see how Santa Barbara's assignment relates to its closer proximity counties.

Table 22: Table continues below

name	TotalPop_mean	Men_mean	Women_mean	VotingAgeCitizen_mean
cluster	157010	50.93	78482	70.63
pca.cluster	877686	49.6	442487	64.8

Table 23: Table continues below

Income_mean	Poverty_mean	ChildPoverty_mean	Professional_mean
56438	16.18	20.46	32.62

Income_mean	Poverty_mean	ChildPoverty_mean	Professional_mean
68727	13.97	18.07	34.83

Table 24: Table continues below

Service_mean	Office_mean	Production_mean	Drive_mean	Carpool_mean
21.52	21.7	10.67	74.02	10.92
19.29	23	10.96	75.39	11.74

Table 25: Table continues below

Transit_mean	OtherTransp_mean	WorkAtHome_mean	MeanCommute_mean
1.73	2.616	7.156	24.25
2.04	3.39	5.23	27.57

Table 26: Table continues below

Employed_mean	PrivateWork_mean	SelfEmployed_mean	FamilyWork_mean
40.92	68.6	10.32	0.4884
44.84	74.99	7.38	0.14

Unemployment_mean	Minority_mean
8.451	36.07
7.79	53.59

Limiting to only counties within California, the first cluster using the first two PCs seems to include the big coastal counties, which can be seen in the larger total population mean, a slightly higher average income, and a higher minority percentage. This makes sense as the coastal cities and counties tend to be more liberal and similar in their political views, while being more diverse and affluent. Thus, we conclude the clustering by the first two PCs leads to a better fit for Santa Barbara County. However, further investigation by changing the number of clusters may also be beneficial in future work.

Classification

14.

We now want to build classification models, so we first need to combine `county.winner` and `census.clean` data. For simplicity, we use the provided code to make necessary changes to merge them into `election.cl` for classification.

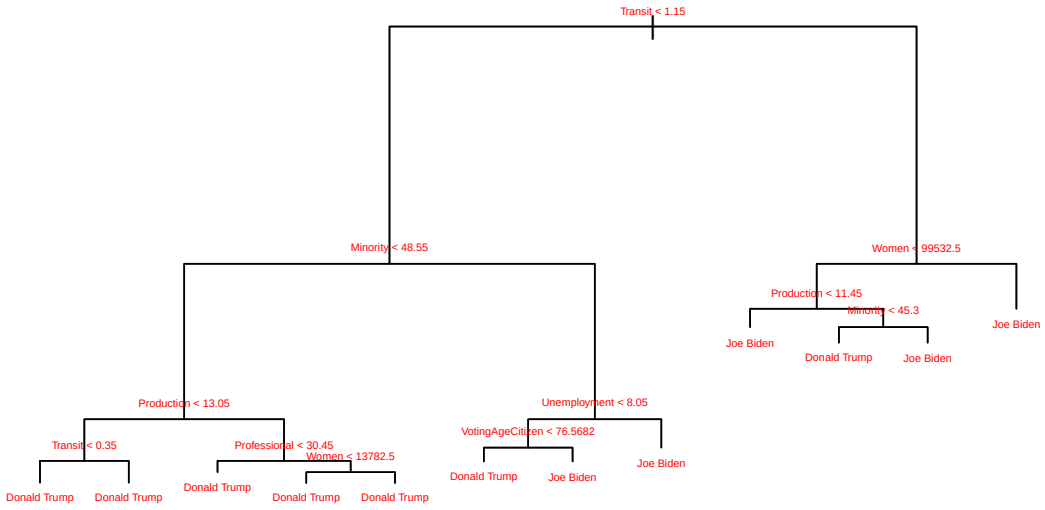
We need to exclude the predictor `party` from `election.cl` as our goal is to use the more involved, numerical features such as `population` and `income` that are more difficult in correctly predicting the winner, not simply the particular political party that a candidate represents.

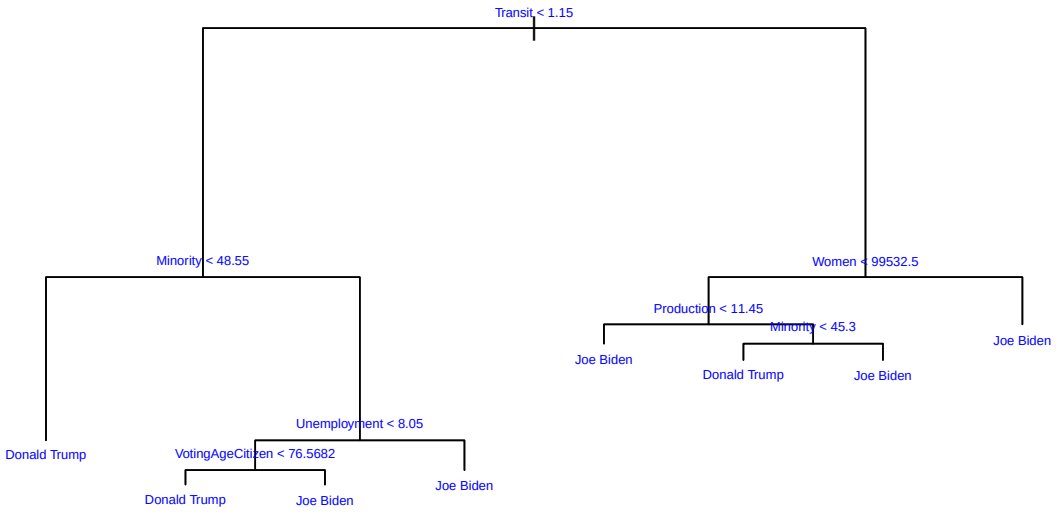
After partitioning the data into 80% training and 20% testing, defining 10 cross-validation folds, and creating the error rate function, we are now ready to perform each classification method that follows.

15. Decision Tree

We first prune the tree to minimize misclassification error. Based on the results using 10-fold cross validation, we find that a tree with 8 terminal nodes is the best tree size.

We now visualize the decision trees before and after it is pruned to its best size, so that we can view the differences.





We now calculate the training and test error rates and save them to the `records` object.

Training Error

0.0708

Test Error

0.1068

```
records['tree','train.error'] <- train_error
records['tree','test.error'] <- test_error
```

Finally, we interpret and discuss the results of the decision tree analysis. First, we note that the pruned tree significantly pruned the nodes, leading to a less complex tree that is less prone to over-fitting. We also note that the test error rate is decently low, which indicates that the pruned tree is minimizing misclassification error and being somewhat successful at accurately predicting voter behavior.

For example, looking at the plot of the pruned tree, we notice from the root node that the most important factor in determining voter behavior is `Transit`, which is the percentage of those who commute via public transportation in their respective county. That is, `Transit` is the predictor that best splits between counties voting for Joe Biden or Donald Trump. From there, we can see right away that if `Transit` percentage is less than 1.15 and the `Minority` percentage is less than 48.55, the decision tree predicts the county winner to be Donald Trump. Similarly, if `Transit` percentage is greater than 1.15 and the population of `Women` exceeds 99532, then the predicted winner is Joe Biden for that county. There are more branches that help predict the county winner for each candidate if the above conditions are not met, but it is rather interesting that

the best sized decision tree only needs at *least* two predictors to decide whether the county winner is Donald Trump or Joe Biden.

16. Logistic Regression

We now run a logistic regression model to predict the winning candidate in each county.

First, we save the training and test errors to the `records` variable.

Training Error

0.0627

Test Error

0.0777

```
records['logistic','train.error'] <- train_error
records['logistic','test.error'] <- test_error
```

Next, we display the significant variables (those at a significance level less than 5%) and their respective estimates:

term	p.value	estimate
VotingAgeCitizen	7.799e-11	0.1679
Professional	1.41e-16	0.3266
Service	2.662e-13	0.3263
Office	0.0004355	0.1657
Production	5.839e-07	0.1974
Drive	0.0001022	-0.1413
Carpool	0.02019	-0.118
Transit	0.003927	0.2554
Employed	1.849e-16	0.2699
PrivateWork	0.01729	0.05007
Unemployment	2.166e-09	0.2888
Minority	6.245e-41	0.13

While our logistic regression model has a lot more significant variables, the majority of them are consistent with our decision tree analysis, such as `Transit`, `Minority`, and `Production`. The only predictor missing as a significant variable in the logistic regression model that was in our decision tree is `Women`.

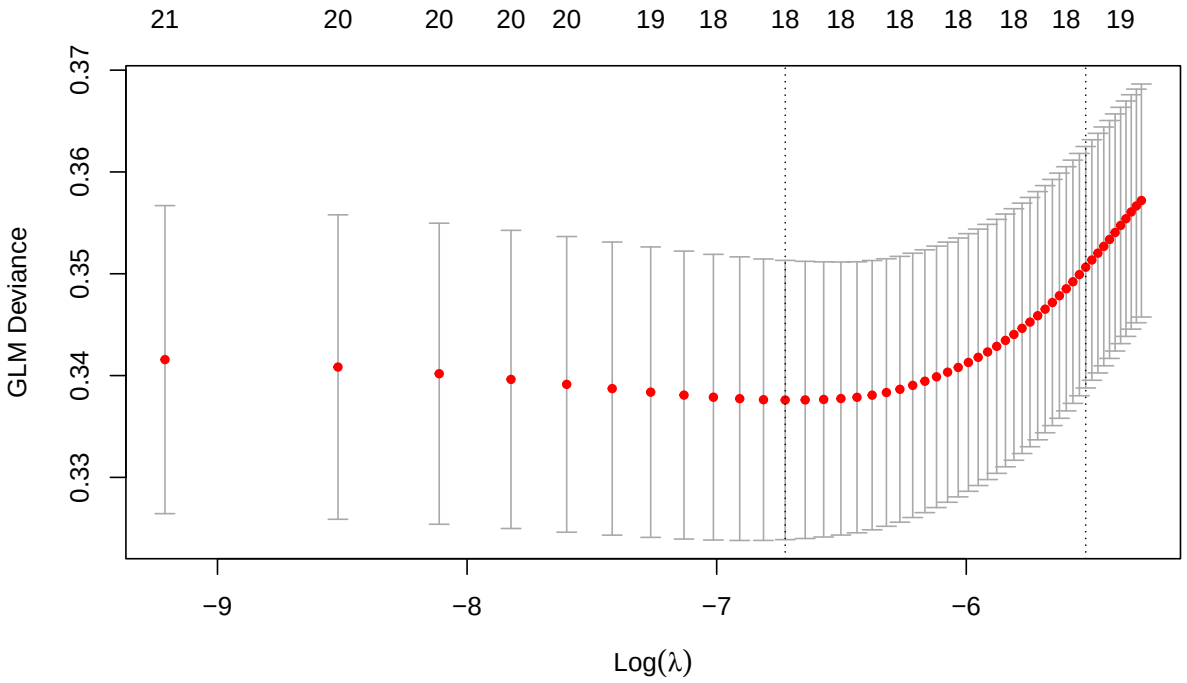
Holding other variables fixed, increasing the percentage of minorities in a county by 1 percentage point would, on average, increase the log odds of Joe Biden winning by 0.13, which agrees with minorities generally having more liberal views. On the other hand, increasing the percent of people who drive to work by 1 percent would decrease the log odds of Joe Biden winning by 0.1413 on average, holding other variables fixed. In the decision tree, counties with more public transit favored Joe Biden so this makes sense that areas with more people driving to work are less likely to favor Joe Biden.

17. Regularization - LASSO

When using logistic regression, we did have some linear combination of variables *perfectly* predicting the winner, i.e., perfect separation. Since this is usually a sign that we are overfitting, we will try to control this through regularization.

First, we will use the `cv.glmnet` function to run a 10-fold cross validation and select the best regularization parameter for the logistic regression with LASSO penalty. We use `lambda = seq(1, 50) * 1e-4` in the `cv.glmnet()` function to set pre-defined candidate values for the tuning parameter λ.

```
set.seed(1)
cv.out.lasso <- cv.glmnet(x_train, y_train, lambda = seq(1,50) * 1e-4,
                          alpha = 1, nfolds = nfold, family = binomial)
```



The optimal value of λ in cross validation is: 0.0012.

The non-zero coefficients in the LASSO regression for the optimal value of λ are displayed below:

coef	s1
(Intercept)	-40.67
TotalPop	1.774e-06
VotingAgeCitizen	0.1578
Poverty	0.04239
Professional	0.2559
Service	0.2572
Office	0.108
Production	0.1319
Drive	-0.1124
Carpool	-0.08469
Transit	0.2303
OtherTransp	0.1045
MeanCommute	0.009245
Employed	0.2279
PrivateWork	0.04181
SelfEmployed	-0.02977
FamilyWork	-0.4102
Unemployment	0.2491
Minority	0.1166

Comparing to the unpenalized logistic regression, we notice some of the non-significant variables from the original logistic regression - `Men`, `Women` `Income`, `ChildPoverty`, and `WorkAtHome` were all set to zero using LASSO, while the remaining non-significant variables (`TotalPop`, `Poverty`, `OtherTransp`, etc. . .) were close to but still non-zero.

Also, comparing coefficient estimates between the unpenalized logistic regression and LASSO regression below, we notice some of the coefficient estimates are similar. However, most seem to be smaller and closer to zero with LASSO, which matches our understanding of LASSO forcing coefficient estimates towards zero and performing variable selection, which it did in this case.

coef	Logistic_Estimates	LASSO_Estimates
(Intercept)	-48.16	-40.67
TotalPop	0.00002807	1.774e-06
Men	0.01449	.
Women	-0.00005175	.
VotingAgeCitizen	0.1679	0.1578
Income	-0.00001475	.
Poverty	0.03181	0.04239
ChildPoverty	0.002474	.
Professional	0.3266	0.2559
Service	0.3263	0.2572
Office	0.1657	0.108
Production	0.1974	0.1319
Drive	-0.1413	-0.1124
Carpool	-0.118	-0.08469
Transit	0.2554	0.2303
OtherTransp	0.1049	0.1045
WorkAtHome	-0.008266	.
MeanCommute	0.02611	0.009245
Employed	0.2699	0.2279
PrivateWork	0.05007	0.04181
SelfEmployed	-0.01718	-0.02977
FamilyWork	-0.5054	-0.4102
Unemployment	0.2888	0.2491
Minority	0.13	0.1166

Lastly, we fit the LASSO model using the best value of lambda and save the training and test errors to the `records` variable.

```
records['lasso','train.error'] <- train_error
records['lasso','test.error'] <- test_error
```

Training Error

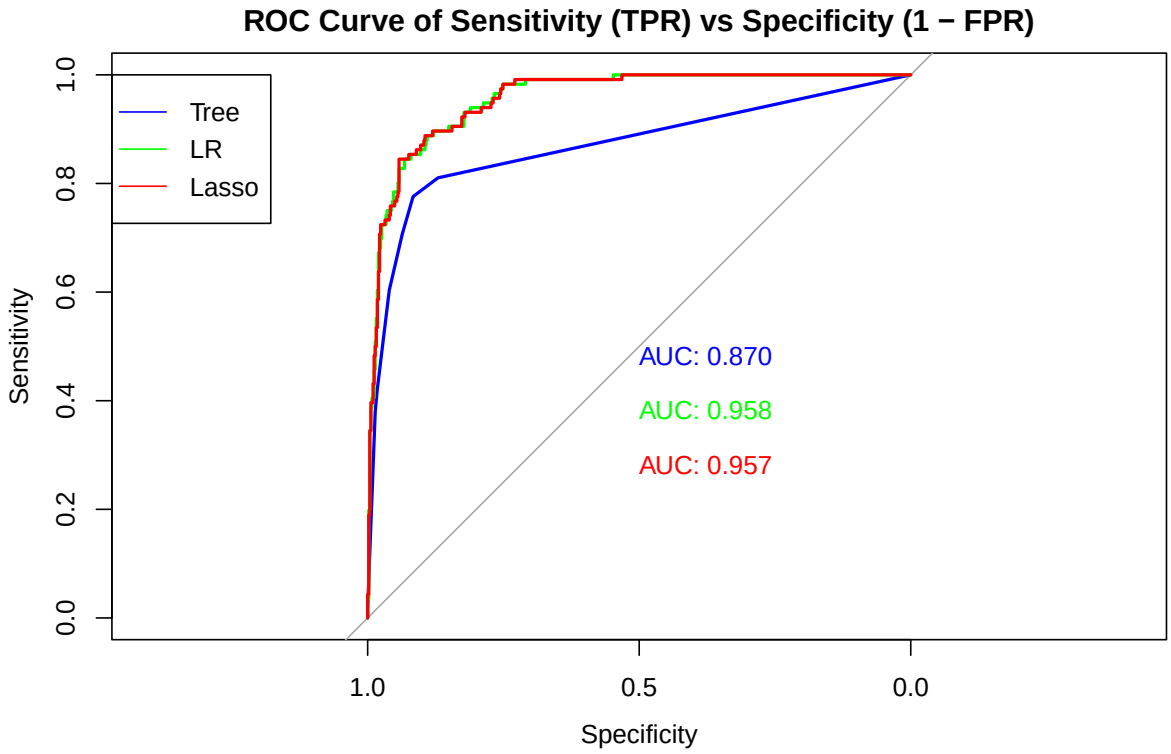
0.0611

Test Error

0.0761

18. ROC Curves

Now, we will compute ROC curves for the decision tree, logistic regression and LASSO logistic regression using predictions on the test data.



As can be seen in the above ROC Curve that plots sensitivity vs. specificity, which is simply TPR vs. (1-FPR), the single decision tree performed the worst in terms of AUC, while logistic and LASSO performed very similarly. However, since LASSO is a simpler model (i.e., less predictors), it would probably be best to choose the LASSO logistic regression model as our final model. Having said that, only considering metrics like AUC or error rate is **not** the only way to choose a model. Interpretability and model complexity play an important role as well as different classifiers being more appropriate for answering different kinds of questions one has about the election.

For example, even though the decision tree performed the worst, it is the most interpretable visually and it might be useful if someone wants to get a better understanding of the features of votes and their preferred candidates. Logistic regression is less interpretable, although its output of a probability and its association with odds makes it a good choice as well. Adding LASSO to logistic regression has the added benefit of preventing overfitting by performing variable selection. This leads to a less complex model and knowledge of which features may not be relevant at all for predicting voter behavior.

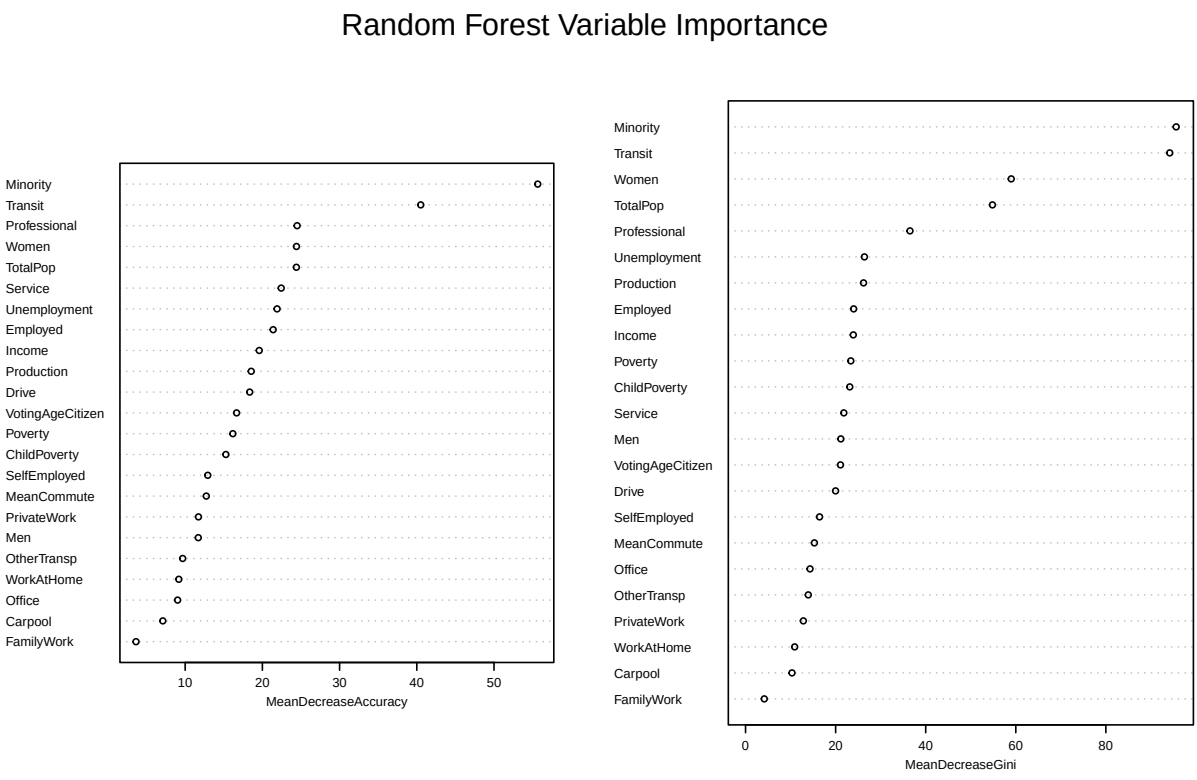
Taking It Further

19.

Random forests have been one of the go-to off-the-shelf classifiers for their strong and robust performance as they combine the power of multiple decision trees and the ensemble approach leads to powerful results. We investigate how random forests perform compared to the previous classifiers.

Based on the summary of the random forest, we find that out-of-bag estimate of error rate is 6.31%. Also, four variables were randomly considered at each split in the trees, and 500 trees were used to fit the data.

Looking at the variable importance plot displayed below, we find that the variables found as important by the random forest such as `Minority` and `Transit` match up with the results from logistic and LASSO regression.



Random Forest Performance

Training Error

0.0004

Test Error

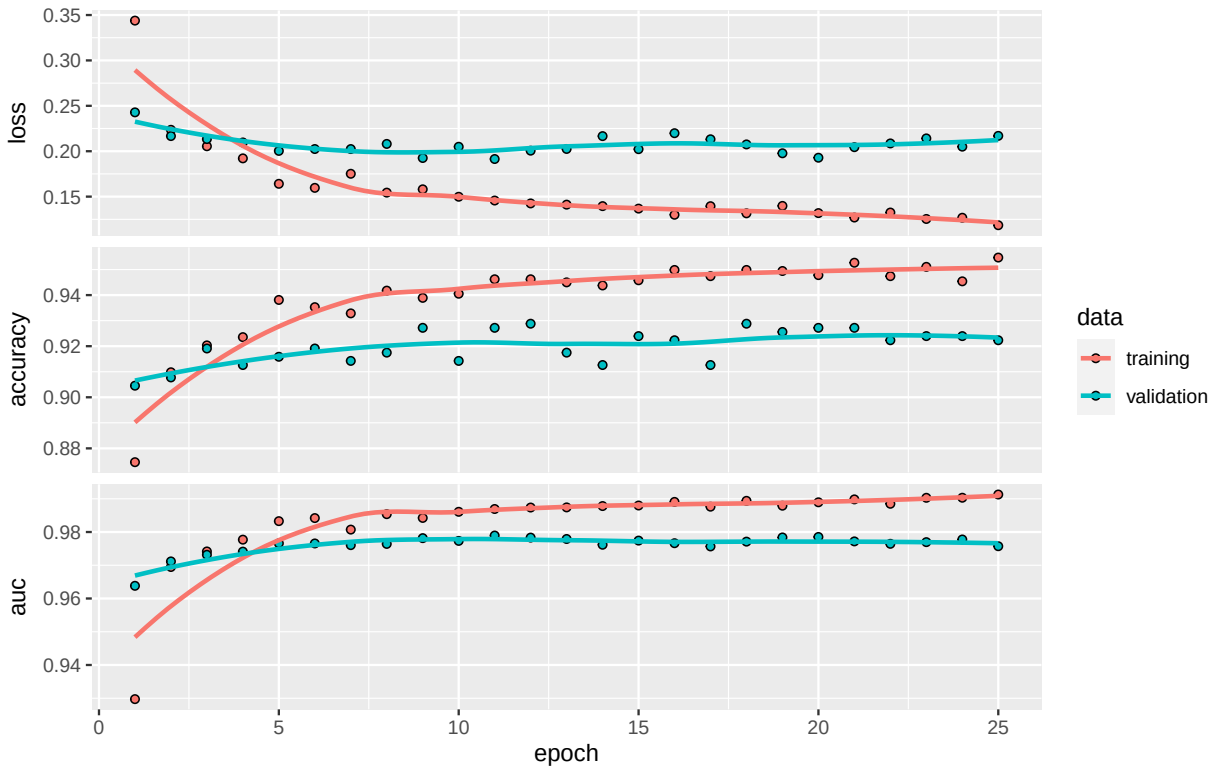
0.0761

Looking at the error rates, random forest did perform better than a single decision tree as well as logistic regression, but only performed the same as LASSO regression.

‘

We next train a neural network to see how deep learning fares on this problem. Deep learning has been a popular approach for classification and regression tasks recently, but it will be interesting to see if it performs better than more classical approaches on this data set as it is relatively small.

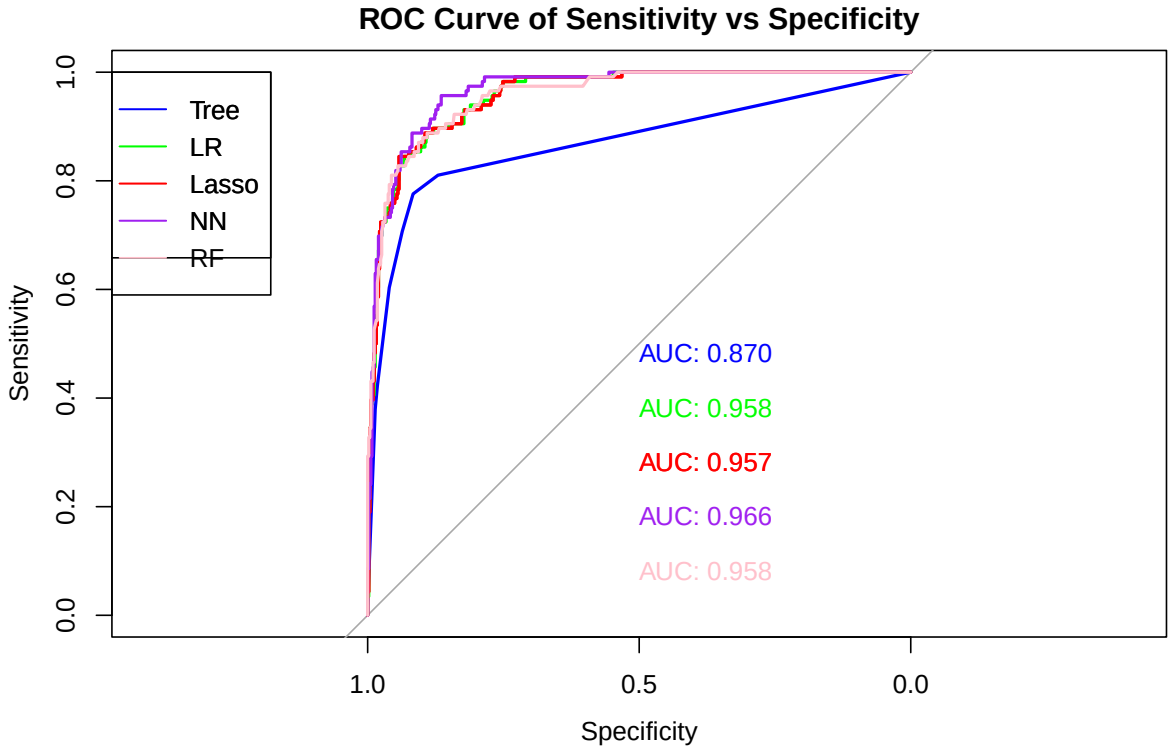
After fitting the model, we display the plots of loss, accuracy, and AUC against each epoch below:



Comparing error rates across all classifiers:

	train.error	test.error
tree	0.07079	0.1068
logistic	0.0627	0.07767
lasso	0.06108	0.07605
random forest	0.0004045	0.07605
NN	0.04531	0.07767

However, accuracy is not the sole metric by which to assess classification. Therefore, we look at the ROC Curves with AUC as well.



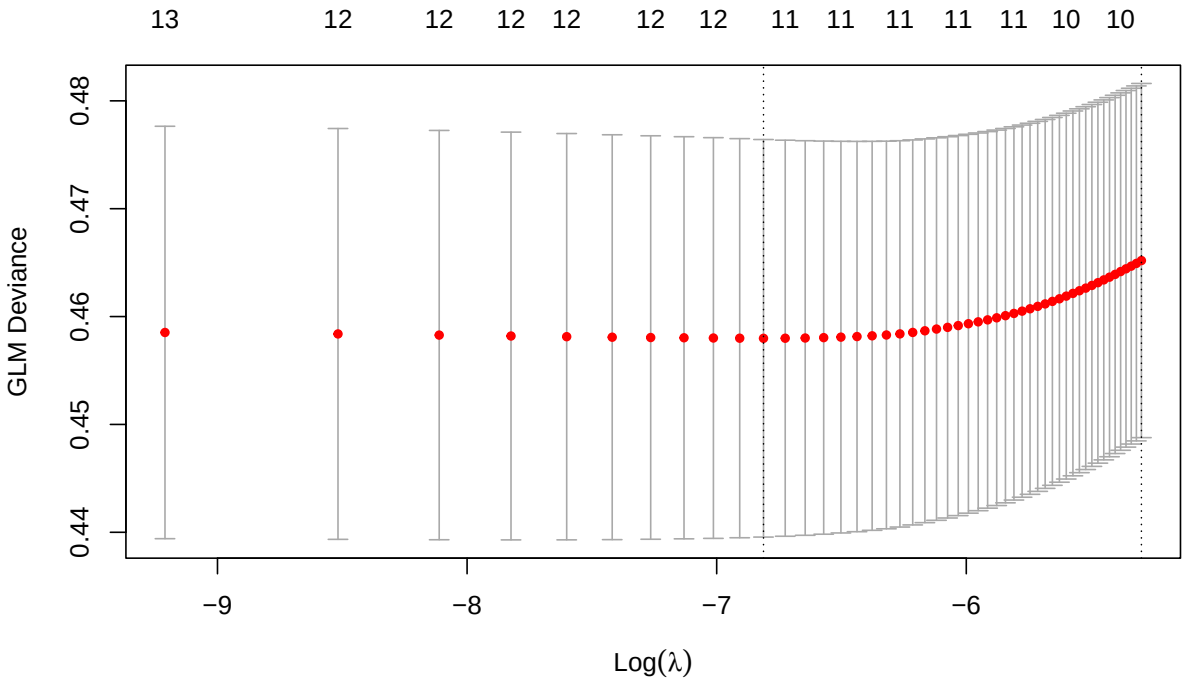
Now, looking at both accuracy and metrics like AUC, we notice that logistic regression, LASSO Regression, neural network, and random forest all perform similarly. However, due to the lack of interpretability of random forests and neural networks, settling for a simpler model like logistic regression or LASSO logistic regression might be beneficial as the performance is very similar to the more complex models, but it still is more interpretable.

20.

In this part, we explore using principal components to create new (and lower dimensional) set of features instead of using the native attributes (the original features) with which to train a classification model. This sometimes improves classification performance, so we will compare classifiers trained on the original features with those trained on PCA features.

Specifically, we want to see how PCA improves the performance of logistic regression, lasso regression, and a single decision tree. Most variables are percentages, but some are numerical such as `Income` and `TotalPop`, so as before, we scale and center due to different scales of the data. We choose the first 13 principal components as they explain 90% of the total variation as found previously.

First, we investigate how PCA improves the performance of LASSO regression.



The optimal value of λ in cross validation is: 0.0011.

We then fit the LASSO model using the best value of lambda and output the training and test errors.

Training Error

0.091

Test Error

0.1003

Next, we investigate how PCA improves the performance of logistic regression.

First, we fit the model and output the summary statistics:

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-2.562	0.1139	-22.5	4.291e-112
PC1	0.1094	0.0367	2.982	0.002868
PC2	-0.974	0.07307	-13.33	1.572e-40
PC3	1.491	0.08889	16.77	3.785e-63
PC4	0.3634	0.06949	5.229	1.706e-07
PC5	-0.004951	0.1072	-0.04619	0.9632
PC6	-0.2067	0.0856	-2.414	0.01577
PC7	-0.6069	0.1048	-5.793	6.911e-09
PC8	0.105	0.09533	1.101	0.2708
PC9	-0.6793	0.1042	-6.516	7.202e-11
PC10	0.7338	0.09966	7.363	1.795e-13
PC11	-0.471	0.1263	-3.729	0.0001923
PC12	0.1053	0.1317	0.7994	0.424
PC13	0.1557	0.1003	1.552	0.1207

(Dispersion parameter for binomial family taken to be 1)

Null deviance:	2185 on 2471 degrees of freedom
Residual deviance:	1099 on 2458 degrees of freedom

Then, we output the training and test errors.

Training Error

0.0906

Test Error

0.0971

Lastly, we investigate how PCA improves the performance of a single decision tree.

First, we prune the tree to minimize misclassification error. Based on the results using 10-fold cross validation, we find that a tree with 12 terminal nodes is the best tree size.

Then, we output the training and test errors.

Training Error

0.0845

Test Error

0.1214

Ultimately, using PCA did not seem to improve the accuracy of any the following classifiers: a single decision tree, logistic regression and LASSO regression.

21.

Lastly, we want to close this analysis by discussing our insights gained overall and provide explanations where appropriate.

From our research, the fact that logistic regression performed as good as, if not better than, more exotic state-of-the-art models is a classic example of interpretability-prediction tradeoff. We tried to sacrifice interpretability for the sake of better predictions by using random forests, neural networks, and even dimension reduction using PCA, but ultimately, this led to worse predictions than those using logistic regression. The failure of the more complex models illustrates the critical importance of balancing interpretability and predictive accuracy in the context of election results. These failures are perhaps indicative that predicting election results accurately depends more on interpretability in order to obtain more meaningful information on which predictors are the most important or insignificant.

Looking back at the 2016 election, most models incorrectly predicted Joe Biden to win, so it is possible many of these models sacrificed interpretability for the sake of being the most sophisticated with presumably more accurate predictions. That sacrifice might have led to less interpretability, which in turn, led to a lack of understanding in where their models could be performing poorly and fostering bias. Thus, is sacrificing interpretability in our models worth it for better predictions? Prediction accuarcy is critical for a model's effectiveness, but interpretability is just as integral as it can lead to even more interesting angles that could warrant further research (such as identifying what general conditions a county would have to meet to be more aligned towards as specific candidate). Furthermore, it can help us identity problems in our models, such as finding the variables that are suspiciously considered less important than others.

In conclusion, when creating a strong model, the goal should not be to create the best predictive models at the expense of interpretability. This concept is important in order to diagnose shortcomings of the model as well as possible alternative routes worth exploring. All things considered, interpretability-prediction tradeoff plays a prominent role in predicting election outcomes.
