2016 Election Analysis

Due December 17, 2020, midnight

Instructions and Expectations

- You are allowed and encouraged to work with one partner on this project. Include your names, perm numbers, and whether you are taking the class for 131 or 231 credit.
- You are welcome and encouraged to write up your report as a research paper (e.g. abstract, introduction, methods, results, conclusion) as long as you address each of the questions below. Alternatively, you can format the assignment like a long homework by addressing each question in parts.
- There should be no raw R *output* in the paper body! All of your results should be formatted in a professional and visually appealing manner. That means, either as a polished visualization or for tabular data, a nicely formatted table (see the documentation for kable and kableExtra packages (https://haozhu233.github.io/kableExtra/awesome_table_in_pdf.pdf). If you feel you must include extensive raw R output, this should be included in an appendix, not the main report.
- All R code should be available from your Rmarkdown file, but does not need to be shown in the body of the report! Use the chunk option `echo=FALSE` to exclude code from appearing in your writeup. In addition to your Rmarkdown, you should turn in the writeup as either a pdf document or an html file (both are acceptable).

Predicting voter behavior is complicated for many reasons despite the tremendous effort in collecting, analyzing, and understanding many available datasets. The 2016 presidential election analysis is a typical example in data analysis and statistical machine learning. For our final project, we will analyze the 2016 presidential election dataset.

Background

The presidential election in 2012 did not come as a surprise. Some predicted the outcome of the election correctly including Nate Silver (https://en.wikipedia.org/wiki/Nate_Silver), and many speculated his approach (https://www.theguardian.com/science/grrlscientist/2012/nov/08/nate-sliver-predict-us-election).

Despite the success in 2012, the 2016 presidential election came as a big surprise (https://fivethirtyeight.com/features/the-polls-missed-trump-we-asked-pollsters-why/) to many, and it was a clear example that even the current state-of-the-art technology can surprise us.

Answer the following questions in one paragraph for each.

1. What makes voter behavior prediction (and thus election forecasting) a hard problem?

2. What was unique to Nate Silver's approach in 2012 that allowed him to achieve good predictions?

3. What went wrong in 2016? What do you think should be done to make future predictions better?

Data

```
## set the working directory as the file location
setwd(getwd())
## put the data folder and this handout file together.
## read data and convert candidate from string to factor
election.raw <- read_delim("data/election/election.csv", delim = ",") %>% mutate(cand
idate=as.factor(candidate))

census_meta <- read_delim("data/census/metadata.csv", delim = ";", col_names = FALSE)
census <- read_delim("data/census/census.csv", delim = ",")
```

Election data

The meaning of each column in `election.raw` is clear except `fips`. The accronym is short for Federal Information Processing Standard (https://en.wikipedia.org/wiki/FIPS_county_code).

In our dataset, `fips` values denote the area (US, state, or county) that each row of data represent. For example, `fips` value of 6037 denotes Los Angeles County.

```
kable(election.raw %>% filter(county == "Los Angeles County"))  %>% kable_styling(boo
tstrap_options = c("striped", "hover", "condensed", "responsive"), full_width=FALSE)
```

county	fips	candidate	state	votes
Los Angeles County	6037	Hillary Clinton	CA	2464364
Los Angeles County	6037	Donald Trump	CA	769743
Los Angeles County	6037	Gary Johnson	CA	88968
Los Angeles County	6037	Jill Stein	CA	76465
Los Angeles County	6037	Gloria La Riva	CA	21993

Some rows in `election.raw` are summary rows and these rows have `county` value of `NA`. There are two kinds of summary rows:

- Federal-level summary rows have `fips` value of `US`.
- State-level summary rows have names of each states as `fips` value.

4. Report the dimension of `election.raw` after removing rows with `fips=2000`. Provide a reason for excluding them. Please make sure to use the same name `election.raw` before and after removing those observations.

Census data

Following is the first few rows of the `census` data:

State	County	TotalPop	Men	Women	Hispanic	White	Black	Native	Asian	Pacific	Citizen	I
Alabama	Autauga	1948	940	1008	0.9	87.4	7.7	0.3	0.6	0.0	1503	
Alabama	Autauga	2156	1059	1097	0.8	40.4	53.3	0.0	2.3	0.0	1662	
Alabama	Autauga	2968	1364	1604	0.0	74.5	18.6	0.5	1.4	0.3	2335	
Alabama	Autauga	4423	2172	2251	10.5	82.8	3.7	1.6	0.0	0.0	3306	
Alabama	Autauga	10763	4922	5841	0.7	68.5	24.8	0.0	3.8	0.0	7666	
Alabama	Autauga	3851	1787	2064	13.1	72.9	11.9	0.0	0.0	0.0	2642	

Census data: column metadata

Column information is given in `metadata`.

Data wrangling

5. Remove summary rows from `election.raw` data: i.e.,

```
* Federal-level summary into a `election_federal`.

* State-level summary into a `election_state`.

* Only county-level data is to be in `election`.
```

6. How many named presidential candidates were there in the 2016 election? Draw a bar chart of all votes received by each candidate. You can split this into multiple plots or may prefer to plot the results on a log scale. Either way, the results should be clear and legible!

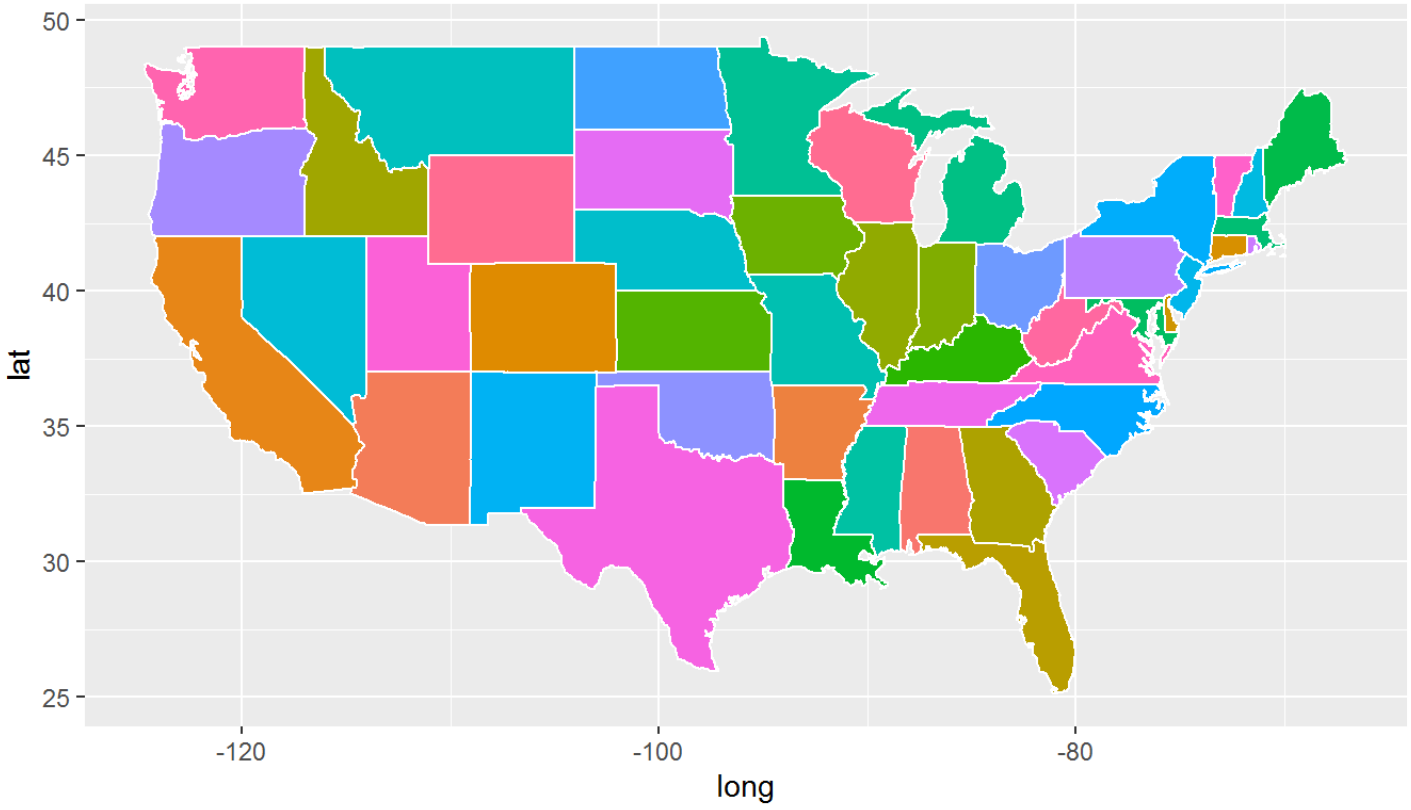
7. Create variables `county_winner` and `state_winner` by taking the candidate with the highest proportion of votes. Hint: to create `county_winner`, start with `election`, group by `fips`, compute `total` votes, and `pct = votes/total`. Then choose the highest row using `top_n` (variable `state_winner` is similar).

Visualization

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

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

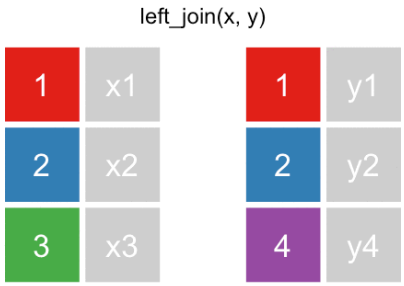
```
states <- map_data("state")

ggplot(data = states) +
  geom_polygon(aes(x = long, y = lat, fill = region, group = group), color = "white")
+
  coord_fixed(1.3) +
  guides(fill=FALSE)  # color legend is unnecessary and takes too long
```



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

8. Draw county-level map by creating `counties = map_data("county")`. Color by county

9. Now color the map by the winning candidate for each state. First, combine `states` variable and `state_winner` we created earlier using `left_join()`. Note that `left_join()` needs to match up values of states to join the tables. A call to `left_join()` takes all the values from the first table and looks for matches in the second table. If it finds a match, it adds the data from the second table; if not, it adds missing values:



Here, we'll be combing the two datasets based on state name. However, the state names are in different formats in the two tables: e.g. `AZ` vs. `arizona`. Before using `left_join()`, create a common column by creating a new column for `states` named `fips = state.abb[match(some_column, some_function(state.name))]`. Replace `some_column` and `some_function` to complete creation of this new column. Then `left_join()`. Your figure will look similar to state_level New York Times map (https://www.nytimes.com/elections/results/president).

10. The variable `county` does not have `fips` column. So we will create one by pooling information from `maps::county.fips`. Split the `polyname` column to `region` and `subregion`. Use `left_join()` combine `county.fips` into `county`. Also, `left_join()` previously created variable `county_winner`. Your figure will look similar to county-level New York Times map (https://www.nytimes.com/elections/results/president).

11. Create a visualization of your choice using `census` data. Many exit polls noted that demographics played a big role in the election (https://fivethirtyeight.com/features/demographics-not-hacking-explain-the-election-results/). Use this Washington Post article (https://www.washingtonpost.com/graphics/politics/2016-election/exit-polls/) and this R graph gallery (https://www.r-graph-gallery.com/) for ideas and inspiration.

12. The `census` data contains high resolution information (more fine-grained than county-level). In this problem, we aggregate the information into county-level data by computing `TotalPop` -weighted average of each attributes for each county. Create the following variables:

- *Clean census data* `census.del`: start with `census`, filter out any rows with missing values, convert {`Men`, `Employed`, `Citizen`} attributes to percentages (meta data seems to be inaccurate), compute `Minority` attribute by combining {Hispanic, Black, Native, Asian, Pacific}, remove these variables after creating `Minority`, remove {`Walk`, `PublicWork`, `Construction`}.
  Many columns seem to be related, and, if a set that adds up to 100%, one column will be deleted.

- *Sub-county census data,* `census.subct` : start with `census.del` from above, `group_by()` two attributes { `State` , `County` }, use `add_tally()` to compute `CountyTotal` . Also, compute the weight by `TotalPop/CountyTotal` .
- *County census data,* `census.ct` : start with `census.subct` , use `summarize_at()` to compute weighted sum
- *Print few rows of* `census.ct` :

Dimensionality reduction

13. Run PCA for both county & sub-county level data. Save the first two principle components PC1 and PC2 into a two-column data frame, call it `ct.pc` and `subct.pc` , respectively. Discuss whether you chose to center and scale the features before running PCA and the reasons for your choice. What are the three features with the largest absolute values of the first principal component? Which features have opposite signs and what does that mean about the correaltion between these features?

14. Determine the number of minimum number of PCs needed to capture 90% of the variance for both the county and sub-county analyses. Plot proportion of variance explained (PVE) and cumulative PVE for both county and sub-county analyses.

Clustering

15. With `census.ct` , perform hierarchical clustering with complete linkage. Cut the tree to partition the observations into 10 clusters. Re-run the hierarchical clustering algorithm using the first 5 principal components of `ct.pc` as inputs instead of the originald features. Compare and contrast the results. For both approaches investigate the cluster that contains San Mateo County. Which approach seemed to put San Mateo County in a more appropriate clusters? Comment on what you observe and discuss possible explanations for these observations.

Classification

In order to train classification models, we need to combine `county_winner` and `census.ct` data. This seemingly straightforward task is harder than it sounds. Following code makes necessary changes to merge them into `election.cl` for classification.

```
tmpwinner <- county_winner %>% ungroup %>%
  mutate(state = state.name[match(state, state.abb)]) %>%               ## state abbr
eviations
  mutate_at(vars(state, county), tolower) %>%                           ## to all low
ercase
  mutate(county = gsub(" county| columbia| city| parish", "", county))  ## remove suf
fixes
tmpcensus <- census.ct %>% mutate_at(vars(State, County), tolower)

election.cl <- tmpwinner %>%
  left_join(tmpcensus, by = c("state"="State", "county"="County")) %>%
  na.omit

## save meta information
election.meta <- election.cl %>% select(c(county, fips, state, votes, pct, total))

## save predictors and class labels
election.cl = election.cl %>% select(-c(county, fips, state, votes, pct, total))
```

Using the following code, partition data into 80% training and 20% testing:

```
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,]
```

Using the following code, define 10 cross-validation folds:

```
set.seed(20)
nfold <- 10
folds <- sample(cut(1:nrow(trn.cl), breaks=nfold, labels=FALSE))
```

Using the following error rate function:

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

Classification

16. Decision tree: train a decision tree by `cv.tree()`. Prune tree to minimize misclassification error. Be sure to use the `folds` from above for cross-validation. Visualize the trees before and after pruning. Save training and test errors to `records` variable. Intepret and discuss the results of the decision tree analysis. Use this plot to tell a story about voting behavior in the US (remember the NYT infographic? (https://archive.nytimes.com/www.nytimes.com/imagepages/2008/04/16/us/20080416_OBAMA_GRAPHIC.html))

17. Run a logistic regression to predict the winning candidate in each county. Save training and test errors to `records` variable. What are the significant variables? Are the consistent with what you saw in decision tree analysis? Interpret the meaning of a couple of the significant coefficients in terms of a unit change in the variables.

18. You may notice that you get a warning `glm.fit: fitted probabilities numerically 0 or 1 occurred`. As we discussed in class, this is an indication that we have perfect separation (some linear combination of variables *perfectly* predicts the winner). This is usually a sign that we are overfitting. One way to control overfitting in logistic regression is through regularization. Use the `cv.glmnet` function from the `glmnet` library to run K-fold cross validation and select the best regularization parameter for the logistic regression with LASSO penalty. Reminder: set `alpha=1` to run LASSO regression, set `lambda = c(1, 5, 10, 50) * 1e-4` in `cv.glmnet()` function to set pre-defined candidate values for the tuning parameter λ. This is because the default candidate values of λ in `cv.glmnet()` is relatively too large for our dataset thus we use pre-defined candidate values. What is the optimal value of λ in cross validation? What are the non-zero coefficients in the LASSO regression for the optimal value of λ? How do they compare to the unpenalized logistic regression? Save training and test errors to the `records` variable.

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

Taking it further (Most important part of your project)

20. This is an open question. Interpret and discuss any overall insights gained in this analysis and possible explanations. Use any tools at your disposal to make your case: visualize errors on the map, discuss what does/doesn't seems reasonable based on your understanding of these methods, propose possible directions (collecting additional data, domain knowledge, etc). In addition, propose and tackle *at least* one more interesting question. Creative and thoughtful analyses will be rewarded! *This part will be **worth up to a 20%** of your final project grade!*

Some possibilities for further exploration are:

- Data preprocessing: we aggregated sub-county level data before performing classification. Would classification at the sub-county level before determining the winner perform better? What implicit assumptions are we making?

- Exploring additional classification methods: KNN, LDA, QDA, SVM, random forest, boosting etc. (You may research and use methods beyond those covered in this course). How do these compare to logistic regression and the tree method?
- Bootstrap: Perform boostrap to generate plots similar to ISLR Figure 4.10/4.11. Discuss the results.
- Use linear regression models to predict the `total` vote for each candidate by county. Compare and contrast these results with the classification models. Which do you prefer and why? How might they complement one another?
- Conduct an exploratory analysis of the "purple" counties-- the counties which the models predict Clinton and Trump were roughly equally likely to win. What is it about these counties that make them hard to predict?
- Instead of using the native attributes (the original features), we can use principal components to create new (and lower dimensional) set of features with which to train a classification model. This sometimes improves classification performance. Compare classifiers trained on the original features with those trained on PCA features.