Milestone_3

November 28, 2018

1 CS 109A

1.1 Milestone 3

1.1.1 Colleen Driscoll, Oliver Mayor and Pooja Tyagi

1.1.2 28 November 2018

1.2 Progress Summary

In Milestone 3, we present the results of exploratory data analysis with respect to U.S. House of Representative elections since 1990, dependent on data availability.

Because results for the 2018 election are not yet finalized (some districts have not yet finished counting), we report base model results to 'predict' the results of the 2016 election.

1.2.1 Research Question: Which candidates will be elected to the U.S. House of Representatives in 2018?

To address this question, the team has accumulated economic, social, and political data on U.S. Congressional Districts and the representatives elected from them. To train accurate models, this data has been collected for the past several decades, based on what is readily available.

1.2.2 Data Summary

Political Outcomes

- As our main outcome of interest, we aim to predict which candidate will win the election for each of the 435 races for Congress. Simplifying this, without loss of generality, we model whether the *Democratic* candidate in each district will win; additionally, the two-party nature of politics allows us to operationalize the binary outcome as one where we predict that the Democrat candidate will win the election if her odds of winning are greater than those of the Republican candidate (that is, if the log-odds is greater than zero). In training datasets, if the Democratic proportion of the two-party vote is greater than 0.5, then the Democrat candidate wins. (Note: There have been a handful of independent members of the House of Representatives of 435 [no more than two per term]. However, as all of the top candidates 2018 election were either Democrats or Republicans, it is safe to ignore third parties/independent candidates in our analysis).
- **Data sources:**

- 2018 election: As of writing, the results of at least one Congressional district have not yet been finalized; upon the certification of all results, the test set data will be compiled either from official state election boards or downloaded from other academics/media.
- 1980 - 2016 outcomes: Data collected based on official records by the Constituency-Level Elections Archive. Following the modeling plan outlined above, we calculated the Democratic share of the two-party vote for each district across 19 elections.

Political Explanatory Variables - Candidate data * Incumbency status: Much research in political science has shown the large positive effect on a candidate's chances of being elected if she is the current holder of the seat (the incumbent). Taking this into account, we create a binary variable for whether the incumbent is running in the election. Next, we combine this binary variable with another that indicates whether the incumbent is a Democrat or Republican, forming an interaction term. When the interaction term indicates that there is a Democratic incumbent running for re-election, we expect predicted Democratic vote share to be higher. When there is a Republican incumbent running for re-election, we expect Democratic vote share to be lower. * Ideological position(s): Political scientists have developed techniques to estimate the ideological position of elected representatives, especially those in Congress. Poole and Rosenthal have developed DW-NOMINATE (Dynamic Weighted NOMINAl Three-step Estimation) scores, which are an aggregate measure of a Congressmember's lifetime public voting record in office over two dimensions, which broadly reflect differences in preferences on economic and social policy. We also include the Nokken-Poole score, which does not make assumptions about ideological stability in members of Congress over time, allowing for more fluctuations within the same legislator over time. Both measures provide for different understandings of the ideology of the incumbent (essentially, how long a memory constituents have); thus, we include both in our dataset.

- Contextual data
  - District prior vote share: For each district, we have a long record of voting. It is likely the case that a district's partisanship, as measured by its most recent vote results, predicts the next future results very well. However, politics is cyclical and all data is subject to random variation. Thus, we include results from multiple years prior to 2018 to train the model. A potential issue with this is the changing nature of districts over time due to redrawing of district boundaries (redistricting). In the next Milestone, we will have a systematic method to deal with redistricting, based on weighting results from districts that once composed the current district.

Socio-economic Data

In the next Milestone, we present data from the American Community Survey. There, we obtain variables on median district age, unemployment rate, household income, educational attainment, and racial/ethnic backgrounds. Regression results using these covariates are not presented at this stage; however, summary statistics are, and these variables will be incorporated in the final model to maximize predictive power.

```
In [307]: import numpy as np
          import pandas as pd
          import re
          import matplotlib.pyplot as plt
          %matplotlib inline
          from sklearn.linear_model import LogisticRegression
          from sklearn.cross_validation import train_test_split
```

```
/anaconda3/lib/python3.6/site-packages/sklearn/cross_validation.py:41: DeprecationWarning: Thi
  "This module will be removed in 0.20.", DeprecationWarning)
```

```
In [212]: data_folder = "https://raw.githubusercontent.com/cdriscoll92/CS-109A-Final-Project/ma
          # local_data_folder = '/Users/poojatyagi/Dropbox (MIT)/CS 109A Final project/Data'
          local_data_folder = "/Users/colleendriscoll/Dropbox/Classes/CS 109A/CS 109A Final pro
```

```
In [21]: ## Reading in state abbreviations file to get the correct district ID columns
         state_abbs = pd.read_csv(data_folder + "state_abbreviations_correspondence_table.csv")
```

Reading in and cleaning data from the Constituency-Level Elections Archive:

```
In [4]: def clea_clean(clea_file_name, state_abb_df):
            ## Read in data
            clea_results = pd.read_csv(clea_file_name)
            democrat_code = 180
            republican_code = 583
            election_month_int = 11

            ## Subsetting to only Democrats and Republicans
            clea_results = clea_results[(clea_results.pty == democrat_code) |
                                        (clea_results.pty == republican_code)]
            ## Only general elections (November)
            clea_results = clea_results[clea_results.mn == election_month_int]

            ## Extracting district number from constituency name
            ## There are some states with only one district that then don't
            ## have a district number listed -- therefore filling those NAs with 1s
            clea_results['dist_num'] = clea_results.cst_n.str.findall('[0-9]+').\
            str[0].fillna(1)

            ## Lowercase state name to match CLEA listing
            state_abb_df['state_name_lower'] = state_abb_df.state_name.str.lower()

            ## Merging CLEA with state abbreviation correspondence table
            clea_merged = pd.merge(clea_results, state_abb_df,
                                   how = 'right',
                                   left_on = 'sub',
                                   right_on = 'state_name_lower')

            ## Creating distict ID variable to merge on later
            clea_merged['dist_id'] = clea_merged['state_abb']+ "_" + \
            clea_merged['dist_num'].astype(str)

            ## Grouping CLEA by district-year to get the democratic share of the
            ## two-party vote
            grouped = clea_merged.groupby(['dist_id', 'yr'])
```

```
    years = []
    dist_ids = []
    dem_shares = []

    for name, group in grouped:
        dem_share = 0
        years.append(group.yr.values[0])
        dist_ids.append(group.dist_id.values[0])

        if democrat_code in group.pty.values: ## If a Democrat ran
            total_votes = np.sum(group.cv1.values)
            dem_votes = np.sum(group.cv1[group.pty == democrat_code].values)
            dem_share = dem_votes/total_votes
        dem_shares.append(dem_share)

    dem_vote_share_dict = {'year': years,
                           'dist_id': dist_ids,
                           'dem_vote_share': dem_shares
                          }
    dem_vote_share = pd.DataFrame(dem_vote_share_dict)

    return dem_vote_share
```

```
In [321]: clea_cleaned = clea_clean(data_folder + "election_results/clea_20180507.csv",
                                    state_abbs)
```

```
In [322]: results_2016 = clea_cleaned[clea_cleaned.year == 2016]
          results_2016 = results_2016[['dist_id', 'dem_vote_share']]
```

```
In [327]: ## If Democratic vote share is >0.50, the Democrat won; else
          ## the Republican did
          results_2016['dem_won_2016'] = np.round(results_2016.dem_vote_share, 0)
          results_2016.drop('dem_vote_share', axis = 1, inplace = True)
```

```
In [328]: results_2016[20:25]
```

Out[328]:

	dist_id	dem_won_2016
353	AZ_9	1.0
372	CA_1	0.0
391	CA_10	0.0
410	CA_11	1.0
429	CA_12	1.0

```
In [329]: clea_cleaned = pd.merge(clea_cleaned, results_2016,
                                  on = "dist_id", how = "left")
```

NOMINATE Data summary

```
In [6]: def drop_secondary_members(nominate_df):
            ## Districts where there was more than one member of Congress serving,
```

```
    ## assign the one who voted the most number of times to the district
    multiple_member_districts = nominate_df.dist_id\
    [nominate_df.dist_id.duplicated()]

    nominate_df['main_member'] = 1
    for district in multiple_member_districts:
        member_votes = nominate_df.nominate_number_of_votes\
        [nominate_df.dist_id == district]

        orders = np.argsort(member_votes)

        lowest_score_index = nominate_df['main_member']\
        [nominate_df.dist_id == district][orders == 0].index

        nominate_df.loc[lowest_score_index, 'main_member'] = 0

    ## Only keeping the main member in each district
    nominate_df = nominate_df[nominate_df.main_member == 1]
    nominate_df.drop('main_member', axis = 1, inplace = True)

    return nominate_df

def nom_scores_clean(nom_file_name, cols_keep):
    nominate_scores = pd.read_csv(nom_file_name)
    nominate_scores = nominate_scores[cols_keep]

    ## Dropping president
    nominate_scores = nominate_scores[nominate_scores['state_abbrev']\
                                      != "USA"]

    ## Dropping members who didn't vote (they can't provide ideology measures then)
    missing_vote_num_indices = nominate_scores.nominate_number_of_votes.isna()\
    == True
    nominate_scores = nominate_scores[~missing_vote_num_indices]

    ## District ID column
    nominate_scores['dist_id'] = nominate_scores.state_abbrev + '_' + \
    nominate_scores.district_code.astype(str)

    nominate_scores = drop_secondary_members(nominate_scores)

    nominate_scores.drop('nominate_number_of_votes', axis = 1,
                        inplace = True)

    ## Election year during which this Congress was in session (not the one that
    ## produced this Congress!)
    session_length = 2
    congress_start_year = 1788
```

```
        nominate_scores['year'] = congress_start_year + session_length*\
        nominate_scores['congress']

        return nominate_scores
```

```
In [7]: session_nums = ['096', '097', '098', '099', '100', '101', '102', '103', '104',
                        '105', '106', '107', '108', '109', '110', '111', '112', '113',
                        '114', '115']

        nominate_csvs = [data_folder + "nominate_scores/H" + x + "_members.csv" \
                         for x in session_nums]
```

```
In [8]: nom_cols_keep = ['congress', 'icpsr', 'district_code',
                         'state_abbrev', 'party_code', 'bioname', 'born',
                         'nominate_dim1', 'nominate_dim2','nominate_number_of_votes',
                         'nokken_poole_dim1', 'nokken_poole_dim2']
```

```
In [9]: nom_combined = nom_scores_clean(nominate_csvs[0],
                                        nom_cols_keep)

        for file_path in nominate_csvs[1:]:
            df = nom_scores_clean(file_path, nom_cols_keep)
            nom_combined = nom_combined.append(df, ignore_index = True)
```

```
/anaconda3/lib/python3.6/site-packages/pandas/core/frame.py:3694: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  errors=errors)
```

```
In [332]: merged_elections_ideology = pd.merge(clea_cleaned, nom_combined, how = "left",
                                               on = ["year", "dist_id"])
          merged_elections_ideology['age'] = merged_elections_ideology['year'] - \
          merged_elections_ideology['born']
          merged_elections_ideology.drop(['district_code','state_abbrev',
                                          'bioname', 'born'],
                                          axis = 1, inplace = True)
```

```
In [333]: merged_elections_ideology[2000:2005]
```

Out[333]:

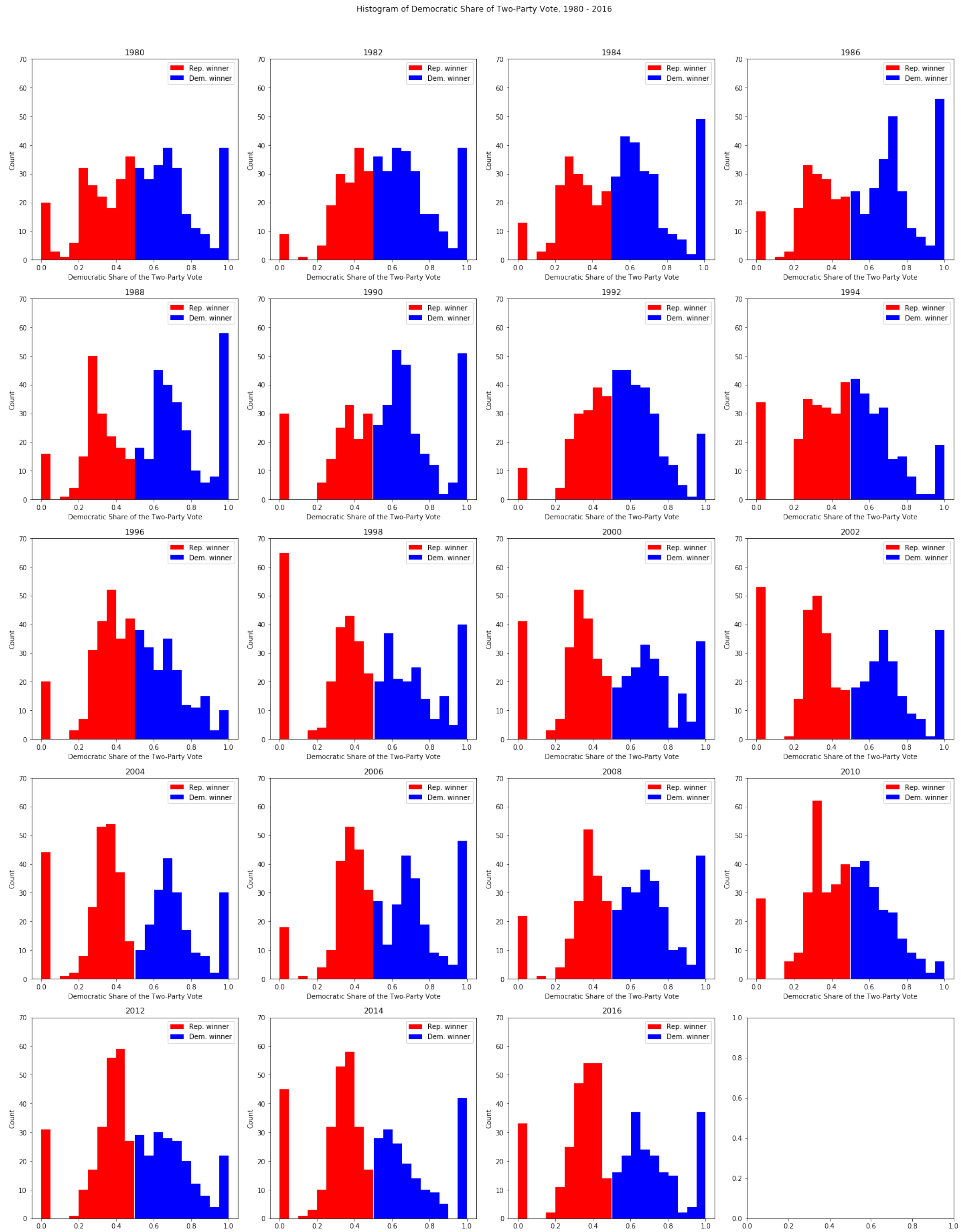
	year	dist_id	dem_vote_share	dem_won_2016	congress	icpsr	\
2000	2014	GA_1	0.390865	0.0	113.0	29338.0	
2001	2016	GA_1	0.000000	0.0	114.0	21513.0	
2002	1980	GA_10	0.802200	0.0	96.0	14404.0	
2003	1982	GA_10	1.000000	0.0	97.0	14404.0	
2004	1984	GA_10	1.000000	0.0	98.0	14404.0	

```
      party_code  nominate_dim1  nominate_dim2  nokken_poole_dim1  \
2000       200.0          0.540          0.302              0.690
2001       200.0          0.572          0.370              0.551
2002       100.0         -0.028          0.639             -0.063
2003       100.0         -0.028          0.639             -0.025
2004       100.0         -0.028          0.639             -0.003

      nokken_poole_dim2   age
2000             -0.059  59.0
2001              0.272  59.0
2002              0.675  58.0
2003              0.838  60.0
2004              0.631  62.0
```

In [249]:
```
col_n, row_n = 4,5

fig, ax  = plt.subplots(nrows=row_n, ncols=col_n, figsize=(5*col_n,5*row_n))
fig.suptitle("Histogram of Democratic Share of Two-Party Vote, 1980 - 2016",
            y = 1.03)
for i, year in enumerate(merged_elections_ideology.year.unique()):
    histogram_values = merged_elections_ideology.dem_vote_share[
        merged_elections_ideology.year == year].values
    republican_winners = [x for x in histogram_values if x < 0.5]
    democrat_winners = [x for x in histogram_values if x >= 0.5]
    ax[i // 4, i % 4].hist(republican_winners, color = "red",
                           label = "Rep. winner")
    ax[i // 4, i % 4].hist(democrat_winners, color = "blue",
                           label = "Dem. winner")
    ax[i // 4, i % 4].set_ylim(0, 70)
    ax[i // 4, i % 4].set_title(year)
    ax[i // 4, i % 4].set_xlabel("Democratic Share of the Two-Party Vote")
    ax[i // 4, i % 4].set_ylabel("Count")
    ax[i // 4, i % 4].legend()
plt.tight_layout()
plt.show();
```



Above, we see the proportion of votes given to the Democratic candidate in each district since 1980. If the Democratic share of the vote is over 50%, then by American electoral rules, the Democrat wins the election (blue shaded bars). If the Democratic share of the vote is below 50%, then the Republican share is over 50%, given the American two-party system, so the Republican candidate wins.

One striking element of the figure above is the number of extreme values at 0 and 1, which indicate candidates that run unopposed. The number of candidates that ran unopposed seems to be highest between 1998 and 2004, although has remained high over the range of years available. Obviously, if a candidate runs unopposed, predicting the outcome is trivial.

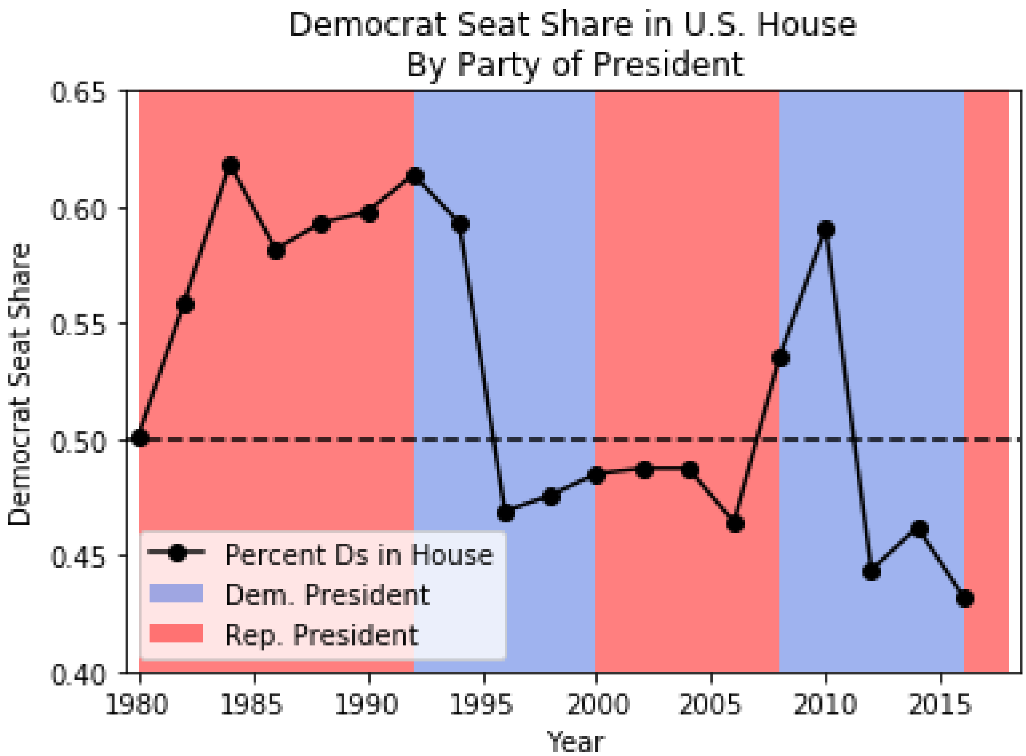
Finally, the margins of victory are worth noting. While many margins of victory were small in 1980 (there are many observations in the range (0.4, 0.6), 2004 provides an example of well-divided elections, whose result was very clear. That said, there does not appear to be a pattern in when the results will be well-separated across the two parties; such distinctions could be made through more thorough analysis of more covariates.

In [93]:
```
ntl_df = pd.read_csv(data_folder + "national_government_makeup.csv")
ntl_df = ntl_df[ntl_df.year < 2018] ## Dropping 2018 results
```

In [95]:
```
## Years in which there was a Republican/Democratic President
R_pres_year_ranges = [[1980.1,1992], [2000.1, 2008], [2016.1, 2018]]
D_pres_year_ranges = [[1978, 1980], [1992.1,2000], [2008.1, 2016]]
```

In [102]:
```
year_low, year_high = 1980, 2018
x = np.arange(year_low, year_high+2, 2)

fig, ax = plt.subplots()
ax.plot(ntl_df.year.values,ntl_df.house.values, "o-",
       color = "black",
       label = "Percent Ds in House")
ax.axhline(0.5, color = "black", linestyle = "--")
ax.set_xlim(year_low-0.5, year_high+0.5)
ax.set_ylim(0.4, 0.65)
## Republican presidents
for year_range in R_pres_year_ranges:
    ax.fill_between(x, 0.7, where = (x > year_range[0] - 1) &
                    (x < year_range[1] + 1),
                    facecolor = "red",
                   alpha = 0.5)
## Democrat presidents
for year_range in D_pres_year_ranges:
    ax.fill_between(x, 0.7, where = (x > year_range[0] - 1) &
                    (x < year_range[1]+1),
                    facecolor = "royalblue",
                   alpha = 0.5)
ax.fill_between(x, 0, facecolor = "royalblue", alpha =0.5,
               label = "Dem. President")
ax.fill_between(x, 0, facecolor = "red", alpha =0.5,
               label = "Rep. President")
plt.title("Democrat Seat Share in U.S. House\nBy Party of President")
plt.xlabel("Year")
plt.ylabel("Democrat Seat Share")
plt.legend()
plt.show();
```



Above, we see the percent seats held by Democrats in the House over time and under Presidents from different parties. This plot shows the midterm advantage enjoyed by the non-presidential party in American politics -- at midterm elections (two years after the President was (re)elected), voters tend to support the party in opposition. This is seen in the graph above by steep drop-offs in the proportion of House seats held be Democrats 1994-1996 and 2008-2010. Additionally, Republicans lost a large share of seats 1982-1984 and 2006-2008.

In terms of modeling, this suggests that we should include a dummy variable indicating whether the election was a Presidential or midterm election. Given that the 2018 election is a midterm election under a Republican President, we expect the Democratic Party (the opposition party) to gain seats this election.

In [220]:
```
## Getting csv files on socioeconomic data
#state_csv_files = os.listdir(local_data_folder + "Socio-economicData/2018")
state_csv_files = os.listdir(local_data_folder+ "Socio-economicData/2018/")
state_csv_files = [x for x in state_csv_files if "csv" in x]
socioeconomic_file_paths = [local_data_folder + "Socio-economicData/2018/" + x \
                            for x in state_csv_files]

len(socioeconomic_file_paths)
```

Out[220]: 50

In [221]:
```
#index of data in CSV
medage_index=16
```

```
unemprate_index=96
medhhincome_index=198
bachdeg_index=240
totpop_index = 19
black_index = 22
asian_index = 24
hispanic_index = 29
```

In [226]:
```
## Dictionary to hold data from districts
data_dict = {
'state_name':[],
'district_num':[],
'median_age':[],
'unemp_rate':[],
'median_HH_income':[],
'bachelor_deg_perc':[],
'total_pop':[],
'black_pop':[],
'asian_pop':[],
'hispanic_pop':[]
}

for csv_end in state_csv_files:
    full_file_path = local_data_folder + "Socio-economicData/2018/" +csv_end

    state_name = csv_end.partition("_District")[0]

    df = pd.read_csv(full_file_path)
    data_columns = np.arange(3, len(df.columns), 2)

    ## Getting variables of interest from each data frame
    data_dict['median_age'].extend(df.iloc[medage_index,data_columns].values)
    data_dict['unemp_rate'].extend(df.iloc[unemprate_index,data_columns].values)
    data_dict['median_HH_income'].extend(df.iloc[medhhincome_index,data_columns].valu
    data_dict['bachelor_deg_perc'].extend(df.iloc[bachdeg_index,data_columns].values
    data_dict['total_pop'].extend(df.iloc[totpop_index,data_columns].values)
    data_dict['black_pop'].extend(df.iloc[black_index,data_columns].values)
    data_dict['asian_pop'].extend(df.iloc[asian_index,data_columns].values)
    data_dict['hispanic_pop'].extend(df.iloc[hispanic_index,data_columns].values)
    data_dict['state_name'].extend([state_name for i in range(len(data_columns))])

    district_names = list(df.columns[data_columns].values)
    data_dict['district_num'].extend(np.arange(len(data_columns))+1)

SE_data_df = pd.DataFrame(data_dict)
```

In [227]:
```
state_abbs[:5]
```

Out[227]:
```
   state_name state_abb state_name_lower
```

```
0     Alabama        AL          alabama
1      Alaska        AK           alaska
2     Arizona        AZ          arizona
3    Arkansas        AR         arkansas
4  California        CA       california
```

```
In [228]: SE_data_merged = pd.merge(SE_data_df, state_abbs, how = "left",
                                    on = "state_name")
          SE_data_merged['dist_id'] = SE_data_merged['state_abb'] + "_"+\
          SE_data_merged['district_num'].astype(str)
          SE_data_merged[:5]
```

```
Out[228]:   state_name  district_num median_age unemp_rate median_HH_income  \
          0    Alabama             1       40.0        5.8            47984
          1    Alabama             2       38.5        6.2            46579
          2    Alabama             3       38.1        5.3            46484
          3    Alabama             4       40.7        6.0            43218
          4    Alabama             5       39.5        4.7            54707

            bachelor_deg_perc total_pop black_pop asian_pop hispanic_pop state_abb  \
          0              25.0    713410    198799     10717        21976        AL
          1              23.1    673776    207087      7686        24457        AL
          2              21.7    710488    187176     11727        20464        AL
          3              17.9    685553     49177      3863        45965        AL
          4              31.9    718713    129234     12993        37350        AL

            state_name_lower dist_id
          0          alabama    AL_1
          1          alabama    AL_2
          2          alabama    AL_3
          3          alabama    AL_4
          4          alabama    AL_5
```

```
In [237]: population_columns = ["total_pop", "black_pop", "asian_pop", "hispanic_pop"]
          for col in population_columns:
              SE_data_merged[col] = SE_data_merged[col].astype(float)
```

```
In [230]: SE_data_merged['black_perc'] = SE_data_merged['black_pop']/SE_data_merged['total_pop
          SE_data_merged['asian_perc'] = SE_data_merged['asian_pop']/SE_data_merged['total_pop
          SE_data_merged['hispanic_perc'] = SE_data_merged['hispanic_pop']/SE_data_merged['tota
```

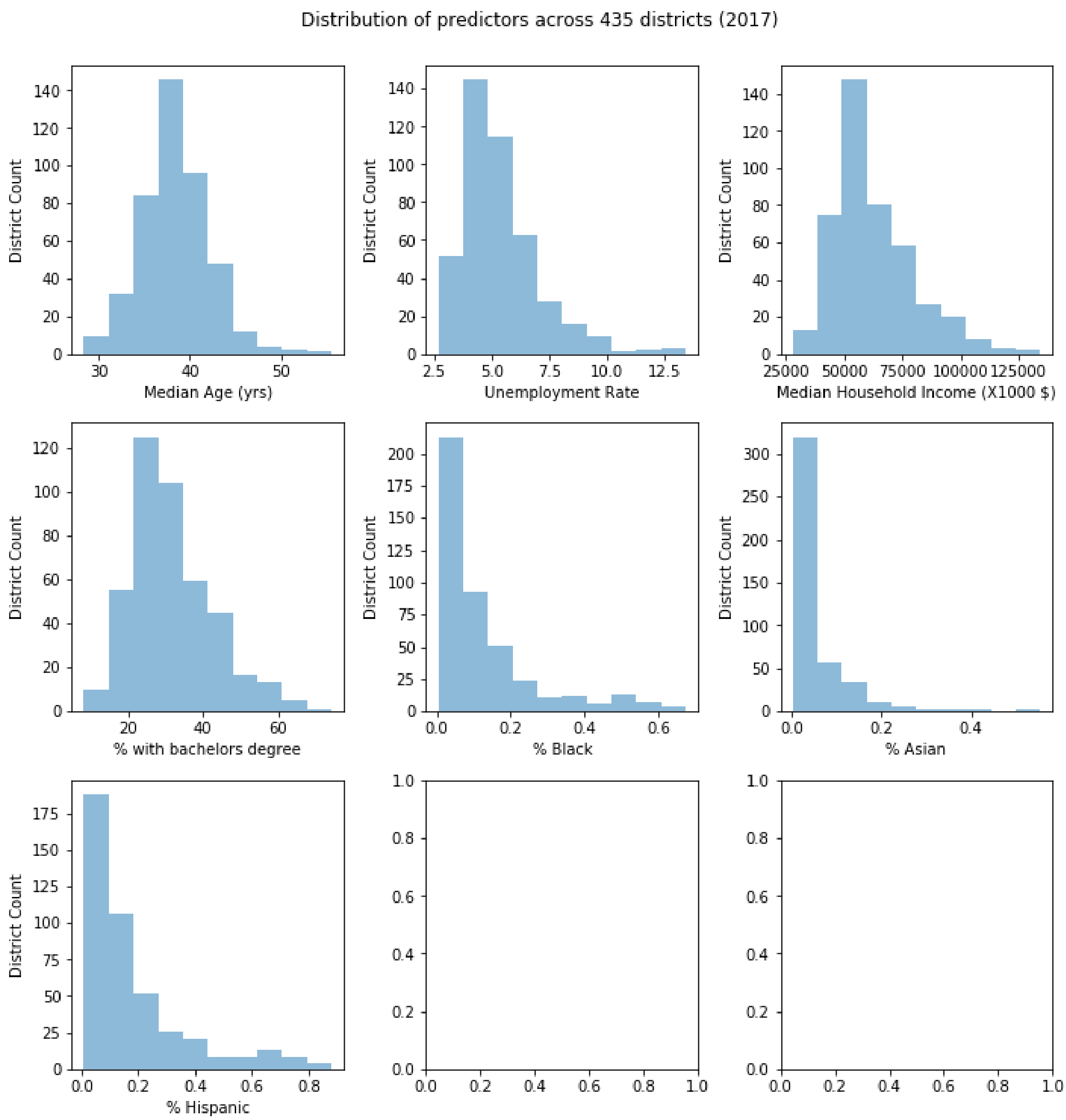
Below, we see histograms of demographic predictors in 2017 -- interestingly, we see slight right-skewed distributions in all of these variables, except for median age, potentially. This indicates that we might need to transform our variables (especially racial/ethnic variables, which appear most skewed) to help ensure that our errors are normally distributed and that our models are not overly influenced by outliers.

```
In [245]: fig, ax = plt.subplots(nrows=3, ncols=3, figsize=(10,10))
          ax = ax.flatten()
```

```
fig.suptitle('Distribution of predictors across 435 districts (2017)', y=1.03)
labels=['Median Age (yrs)', 'Unemployment Rate', 'Median Household Income (X1000 $)'
        '% with bachelors degree',
        '% Black', '% Asian', '% Hispanic']

for i, col_i in enumerate([2, 3, 4, 5, 13, 14, 15]):
    ax[i].hist(SE_data_merged.iloc[:,col_i].dropna().astype(float), alpha=0.5)
    ax[i].set_ylabel('District Count')
    ax[i].set_xlabel(labels[i])

plt.tight_layout()
```



In [256]:
```
## This data applies for the 2018 election (it is the most recent)

SE_data_merged['year'] = 2018
```

In [259]:
```
## Merging in current socioeconomic data -- will do
## when we have more data than 2018
# combined_data = pd.merge(merged_elections_ideology,
#                          SE_data_merged,
#                          on = ["dist_id", "year"],
#                          how = "left")
```

In [378]:
```
## Data for 2005 - 2017
unemp_df = pd.read_csv(local_data_folder +
                       "congressional_district_unemployment_2005-2017.csv")
```

In [255]:
```
unemp_df[:5]
```

Out[255]:

	dist_id	year	unemployment
0	AL_1	2005	6.3
1	AL_2	2005	6.7
2	AL_3	2005	8.8
3	AL_4	2005	6.9
4	AL_5	2005	7.0

In [334]:
```
## Merging in unemployment data
combined_data = pd.merge(merged_elections_ideology,
                         unemp_df,
                         on = ["dist_id", "year"],
                         how = "left")
combined_data.drop_duplicates(inplace = True)
```

In [335]:
```
combined_data[10:15]
```

Out[335]:

	year	dist_id	dem_vote_share	dem_won_2016	congress	icpsr	party_code	\
10	2000	AK_1	0.192064	0.0	106.0	14066.0	200.0	
11	2002	AK_1	0.188273	0.0	107.0	14066.0	200.0	
12	2004	AK_1	0.239302	0.0	108.0	14066.0	200.0	
13	2006	AK_1	0.414254	0.0	109.0	14066.0	200.0	
16	2008	AK_1	0.472837	0.0	110.0	14066.0	200.0	

	nominate_dim1	nominate_dim2	nokken_poole_dim1	nokken_poole_dim2	age	\
10	0.279	0.014	0.295	0.181	67.0	
11	0.279	0.014	0.383	0.215	69.0	
12	0.279	0.014	0.353	0.142	71.0	
13	0.279	0.014	0.367	-0.047	73.0	
16	0.279	0.014	0.291	0.269	75.0	

	unemployment
10	NaN

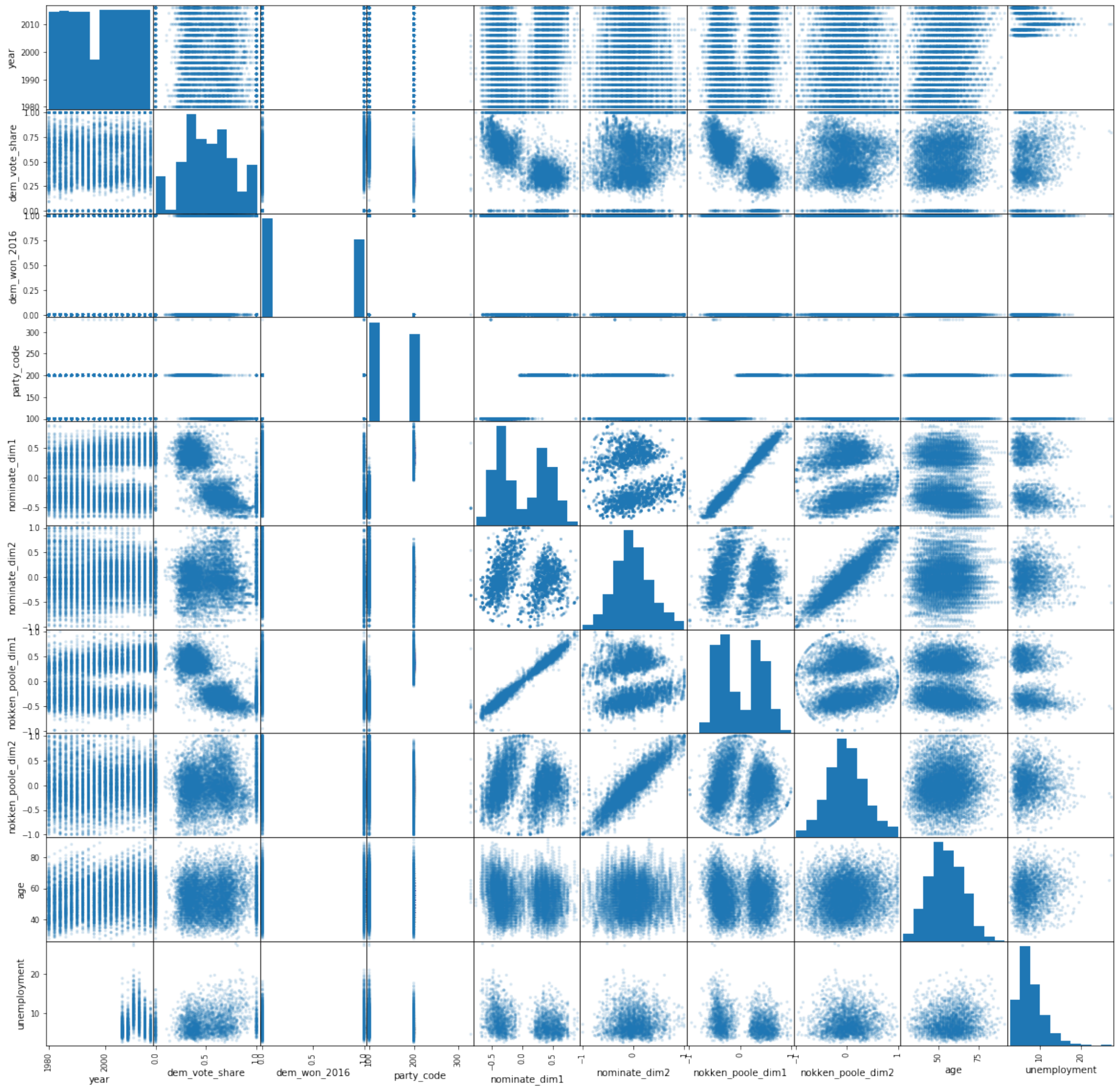
```
11            NaN
12            NaN
13            9.4
16            7.7
```

```
In [336]: ## Dropping some columns for the scatter matrix
          cols_to_drop_for_scatter = ['dist_id', 'congress', 'icpsr']

          to_scatter_df = combined_data.drop(cols_to_drop_for_scatter,
                                             axis = 1,
                                            inplace = False)
```

Below, we see a scatter matrix for all of the predictors in the dataset. One interesting aspect of the scatter matrix is that the ideological measures are split at 0.5 -- that is, there are few observations at the median of the distribution. This may prove problematic, as voting records are likely highly collinear with the incumbent's party. Interestingly, however, there is a high level of variance between ideological scores and the ultimate democratic vote share, indicating that incumbent ideology is a good, though not perfect, predictor of the election outcome.

```
In [337]: pd.plotting.scatter_matrix(to_scatter_df, alpha = 0.2,
                                    figsize = (20,20))
          plt.savefig(local_data_folder + "scatter_matrix.png");
```



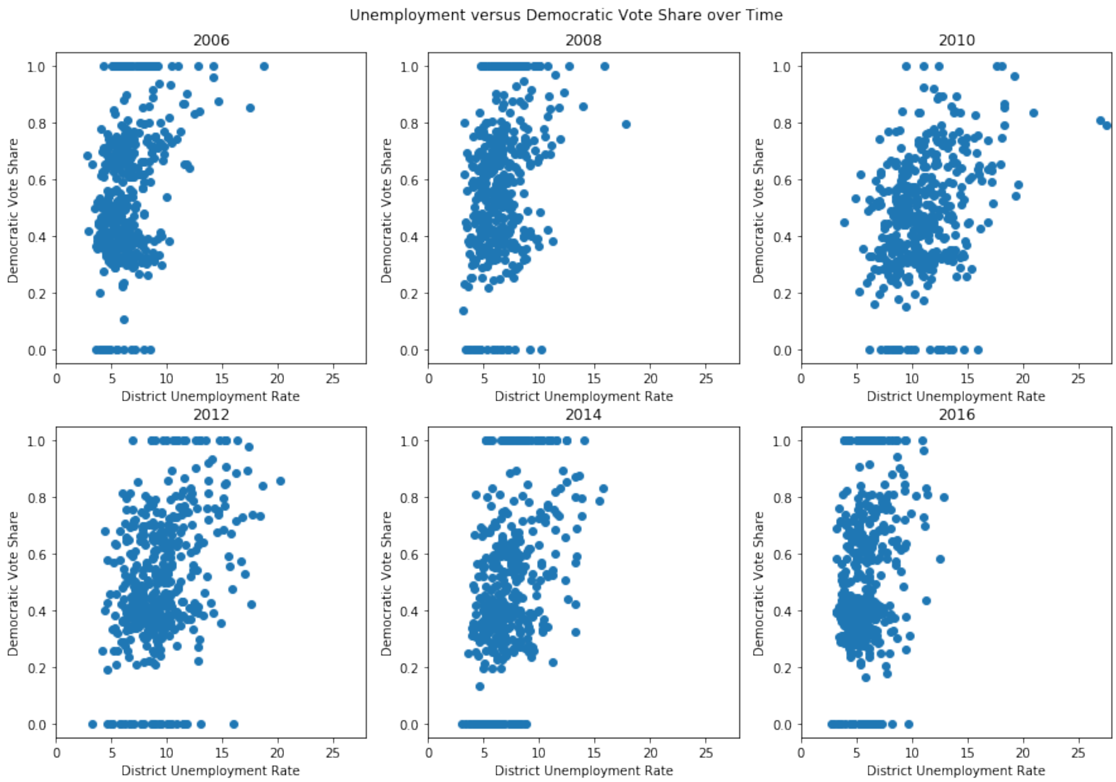

```
In [306]: years_no_NA_unemp = np.arange(2006, 2017, 2)

          col_n, row_n = 3, 2

          fig, ax  = plt.subplots(nrows=row_n, ncols=col_n, figsize=(5*col_n,5*row_n))
          # plt.tight_layout()
          ax = ax.flatten()
          fig.suptitle("Unemployment versus Democratic Vote Share over Time", y = 0.93)
          for i, year_i in enumerate(years_no_NA_unemp):
              this_year_df = combined_data[combined_data.year == year_i]
              ax[i].plot(this_year_df['unemployment'],
                         this_year_df['dem_vote_share'], "o")
              ax[i].set_title(year_i)
              ax[i].set_xlim(0, 28)
```

```
        ax[i].set_ylabel('Democratic Vote Share')
        ax[i].set_xlabel("District Unemployment Rate")

    plt.show();
```



Interestingly, we see above that the relationship between district-level unemployment and the percent of the vote that the Democratic candidate obtains might not be linear. Particularly, when there is greater variation in unemployment, the apparent relationship between unemployment and greater success for Democratic candidates is stronger. We see this in particular in comparing the elections in 2010 and 2016: in 2010, unemployment surpassed 25% in two districts, while it was no higher than 15% in 2016. These scatterplots suggest that we should include a measure of variation in nationwide unemployment and the unemployment rate itself, or that especially high unemployment rates may be outliers that could bias our results. In any case, care should be taken with these data.

Below, we fit a basic logistic model to predict the 2016 election outcomes (the 2018 outcomes are not yet available in full, as discussed above). We base this model solely on the election outcomes in the district since 1980. Impressively, this baseline model already performs fairly well. However, future models that incorporate more covariates will likely do better.

```
In [350]: # Fit logistic regression on training data

    data_to_fit = combined_data.dropna(inplace = False)
```

```
## Can't predict 2016 results with 2016 results...
x_train = data_to_fit[data_to_fit.year < 2016]
y_train = x_train.dem_won_2016.values

logreg1 = LogisticRegression(C=100000)
logreg1.fit(x_train.dem_vote_share.values.reshape(-1, 1),
            y_train)
```

Out[350]:
```
LogisticRegression(C=100000, class_weight=None, dual=False,
          fit_intercept=True, intercept_scaling=1, max_iter=100,
          multi_class='ovr', n_jobs=1, penalty='l2', random_state=None,
          solver='liblinear', tol=0.0001, verbose=0, warm_start=False)
```

In [361]:
```
# Make Prediction and check the accuracy
y_train_probs=(logreg1.predict_proba(x_train.dem_vote_share\
                                     .values.reshape(-1, 1)))

train_accuracy = logreg1.score(x_train.dem_vote_share\
                               .values.reshape(-1, 1),
                            y_train)*100

print('Accuracy of logistic regression classifier on train set: ',
      np.round(train_accuracy, 2), "%", sep = "")
```

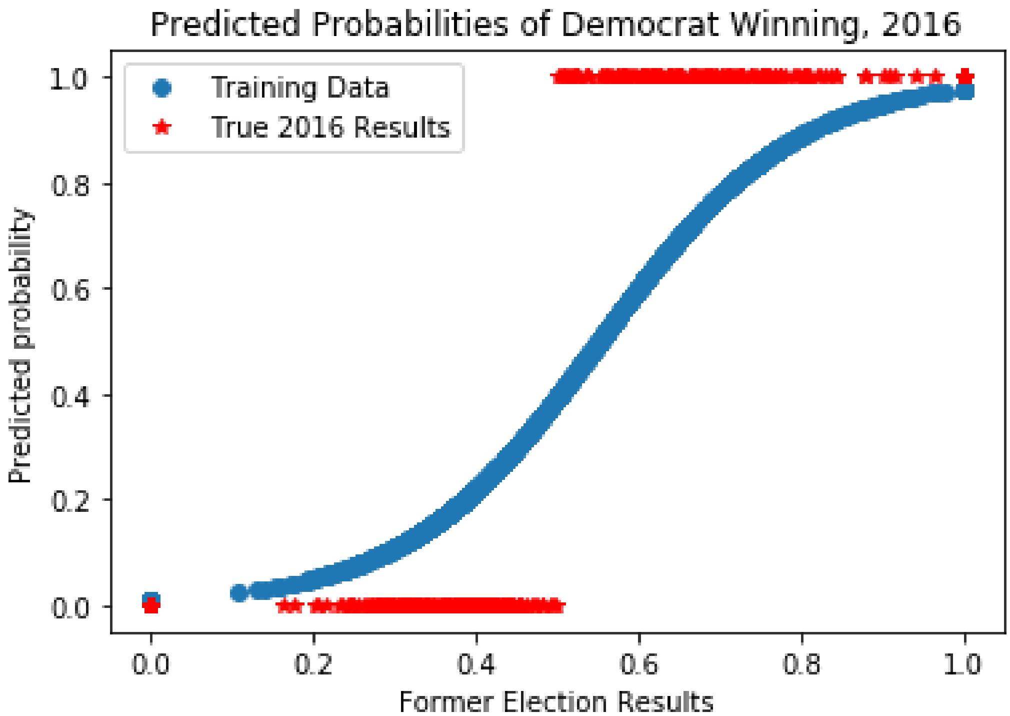
```
Accuracy of logistic regression classifier on train set: 82.83%
```

In [376]:
```
## True outcomes:
x_train_true = data_to_fit[data_to_fit.year == 2016]['dem_vote_share'].values
y_train_true = data_to_fit[data_to_fit.year == 2016]['dem_won_2016'].values

# Plot predicted probabilities

plt.plot(x_train.dem_vote_share.values,
         y_train_probs[:,1],'o',label='Training Data')
plt.plot(x_train_true, y_train_true, "*", color = "red",
        label = "True 2016 Results")
plt.xlabel('Former Election Results')
plt.ylabel('Predicted probability')
plt.title('Predicted Probabilities of Democrat Winning, 2016')
plt.legend()

plt.show();
```


Predicted Probabilities of Democrat Winning, 2016
Training Data
True 2016 Results
Predicted probability
Former Election Results
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0