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

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)

- Contextual data
  - District prior vote share: [Filler] A potential issue with this is the changing nature of districts over time due to redrawing of district boundaries (redistricting).

Socio-economic Data - Variables * Age * Unemployment rate * Household income * Education * Proportion black * Proportion hispanic * Proportion Asian

```
In [1]: import numpy as np
        import pandas as pd
        import re
        import matplotlib.pyplot as plt
        %matplotlib inline
```

```
In [105]: data_folder = "https://raw.githubusercontent.com/cdriscoll92/CS-109A-Final-Project/ma
          local_data_folder = "/Users/colleendriscoll/Dropbox/Classes/CS 109A/CS 109A Final pro
```

```
In [3]: ## Reading in state abbreviations file to get the correct district ID columns
        state_abbs = pd.read_csv(data_folder + "state_abbreviations_correspondence_table.csv")
```

CLEA Summary here

```
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(
```

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

            ## Merging CLEA with state abbrevation 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 [5]: clea_cleaned = clea_clean(data_folder + "election_results/clea_20180507.csv",
                                  state_abbs)
```

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 == dist

        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 [10]: merged_elections_ideology = pd.merge(clea_cleaned, nom_combined, how = "left",
                                               on = ["year", "dist_id"])
```

```
In [11]: merged_elections_ideology[2000:2005]
```

Out[11]:

	year	dist_id	dem_vote_share	congress	icpsr	district_code
2000	2014	GA_1	0.390865	113.0	29338.0	1.0
2001	2016	GA_1	0.000000	114.0	21513.0	1.0
2002	1980	GA_10	0.802200	96.0	14404.0	10.0
2003	1982	GA_10	1.000000	97.0	14404.0	10.0
2004	1984	GA_10	1.000000	98.0	14404.0	10.0

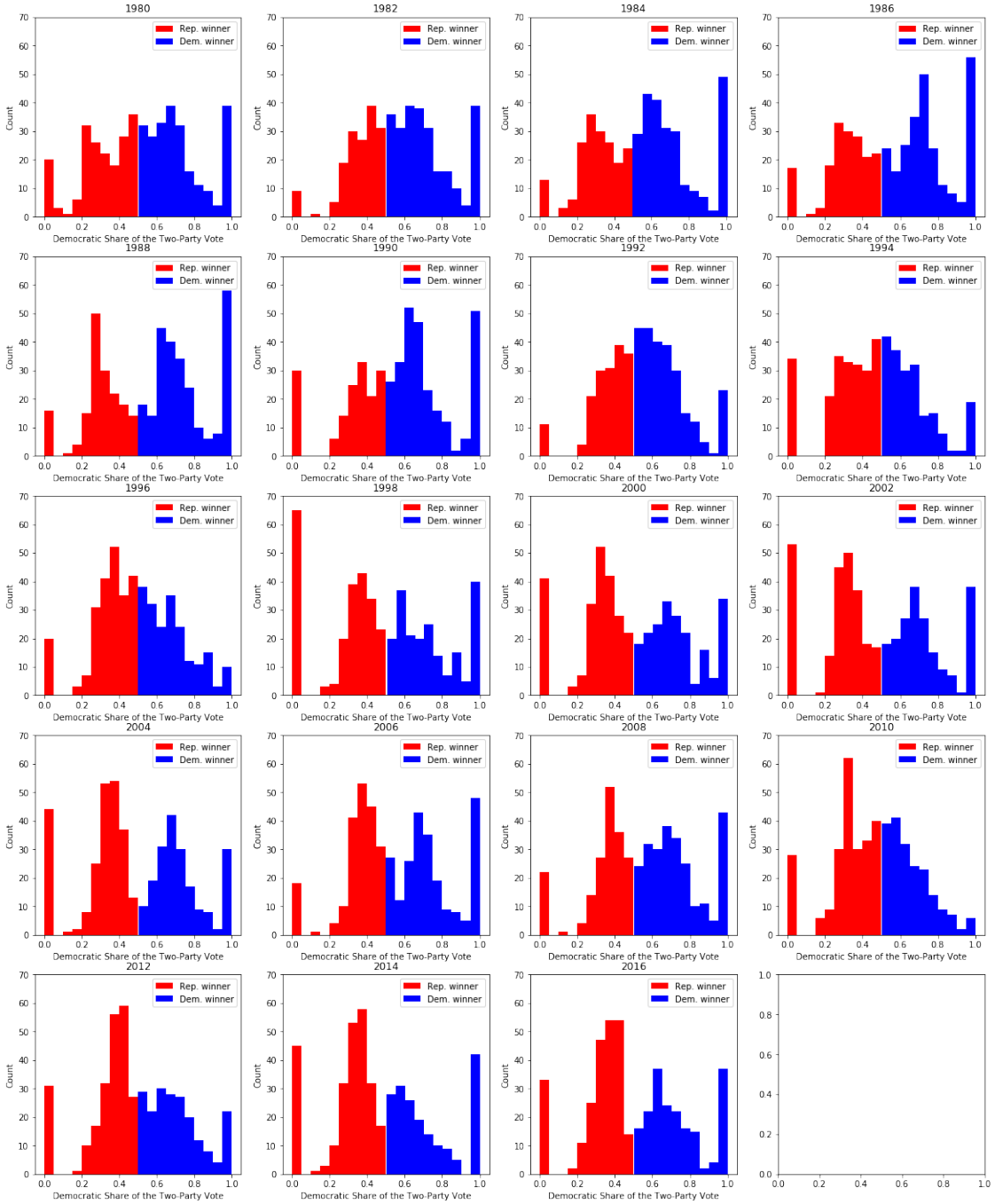
	state_abbrev	party_code	bioname	born
2000	GA	200.0	KINGSTON, Jack	1955.0
2001	GA	200.0	CARTER, Buddy	1957.0
2002	GA	100.0	BARNARD, Druie Douglas, Jr.	1922.0
2003	GA	100.0	BARNARD, Druie Douglas, Jr.	1922.0
2004	GA	100.0	BARNARD, Druie Douglas, Jr.	1922.0

	nominate_dim1	nominate_dim2	nokken_poole_dim1	nokken_poole_dim2

2000	0.540	0.302	0.690	-0.059
2001	0.572	0.370	0.551	0.272
2002	-0.028	0.639	-0.063	0.675
2003	-0.028	0.639	-0.025	0.838
2004	-0.028	0.639	-0.003	0.631

In [28]:
```
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")
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.show();
```

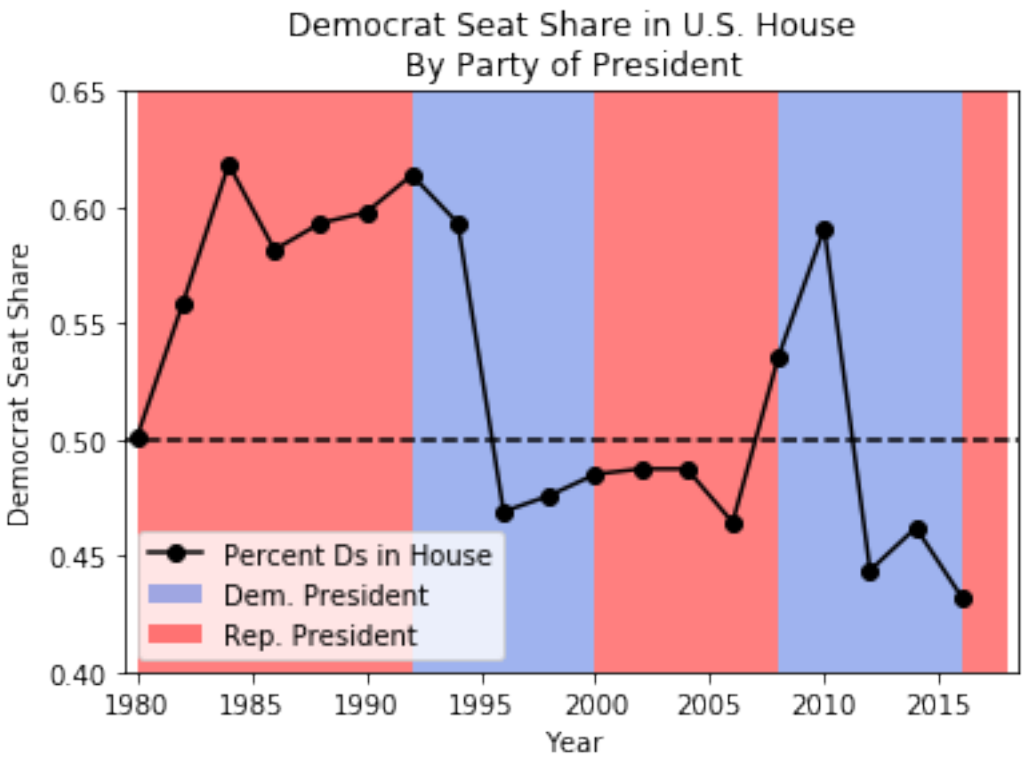



```
In [93]: ntl_df = pd.read_csv(data_folder + "national_government_makeup.csv")
         ntl_df = ntl_df[ntl_df.year < 2018] ## Dropping 2018 results
```

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();
```




```
In [107]: import os

In [138]: ## Getting csv files on socioeconomic data
          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 = [data_folder + "Socio-economicData/" + x \
                                      for x in state_csv_files]

          len(socioeconomic_file_paths)

In [186]: #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 [192]: ## 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 [189]:
```
state_abbs[:5]
```

Out[189]:

	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 [193]:
```
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[193]:

	state_name	district_num	median_age	unemp_rate	median_HH_income	bachelor_deg_perc	total_pop	black_pop	asian_pop	hispanic_pop	state_abb	state_name_lower	dist_id
0	Alabama	1	40.0	5.8	47984	25.0	713410	198799	10717	21976	AL	alabama	AL_1
1	Alabama	2	38.5	6.2	46579	23.1	673776	207087	7686	24457	AL	alabama	AL_2
2	Alabama	3	38.1	5.3	46484	21.7	710488	187176	11727	20464	AL	alabama	AL_3
3	Alabama	4	40.7	6.0	43218	17.9	685553	49177	3863	45965	AL	alabama	AL_4
4	Alabama	5	39.5	4.7	54707	31.9	718713	129234	12993	37350	AL	alabama	AL_5

In [199]:
```
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 [200]:
```
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
```

In [201]:
```
SE_data_merged.columns
```

Out[201]:
```
Index(['state_name', 'district_num', 'median_age', 'unemp_rate',
       'median_HH_income', 'bachelor_deg_perc', 'total_pop', 'black_pop',
       'asian_pop', 'hispanic_pop', 'state_abb', 'state_name_lower', 'dist_id',
       'black_perc', 'asian_perc', 'hispanic_perc'],
      dtype='object')
```

In [ ]:
```
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, 13, 14, 15]):
    ax[i].hist(SE_data_merged.[SE_data_merged.columns[i]].dropna(), alpha=0.5)
    ax[i].set_ylabel('Count')
    ax[i].set_xlabel(labels[i])
```

```
plt.tight_layout()

plt.savefig('sample.pdf', bbox_inches='tight')
```