

us-presidential-eda-from-1976

April 16, 2024

0.1 Let us analyze the dataset for US presidential election from 1976

0.1.1 This kernel/notebook does the following

1. *Performs EDA on US presidential election data consolidated from all states from 1976-2020*
2. *Does some analysis on various parties*
3. *Performs analysis at a state level*
4. *Uses animation from Plotly and BokehJS*

0.2 Before starting install Bokeh and Holoviews Libraries by:

pip install bokeh

pip install holoviews

0.2.1 Import the libraries

```
[ ]: from bokeh.plotting import figure, output_file, show
      from bokeh.io import output_notebook
      from bokeh.models import ColumnDataSource, NumeralTickFormatter
      from bokeh.models.tools import HoverTool
      from bokeh.transform import dodge

      import random
      import math
      import numpy as np
      import pandas as pd
      import matplotlib.pyplot as plt

      pd.set_option('display.max_columns', 100)
      import matplotlib.pyplot as plt
      import seaborn as sns
      from scipy.stats import norm
      import holoviews as hv
      from holoviews import opts
      hv.extension('bokeh')
      import os
```

```
[ ]: import plotly.express as px
import plotly.graph_objs as go
from plotly import tools
from plotly.offline import iplot, init_notebook_mode
init_notebook_mode()
```

```
[ ]: #Let us define a random color function
def rand_color():
    return "#" + "".join(random.sample("0123456789abcdef", 6))
```

```
[ ]: #Let us define the output to a notebook
output_notebook()
```

Bokeh Case Study - US presidential elections data 1976-2020

- Using a freely available dataset at Harvard dataverse site.
- Leveraging the US presidential vote data set from 1976 to 2020 across various US States.
- Tabulating data for the top two parties across the years.

0.3 Load the dataset

```
[ ]: #We are using the US Presidential vote data set from 1976 to 2016 across states
#The source file is available at Harvard Dataverse site
```

```
df = pd.read_csv('1976-2020-president.csv')
#Dropping the irrelevant columns notes and version
df.drop(['notes', 'version'],axis=1, inplace=True)
df['party'] = df['party_detailed'].str.lower()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4287 entries, 0 to 4286
Data columns (total 14 columns):
#   Column                Non-Null Count  Dtype
---  -
0   year                  4287 non-null  int64
1   state                 4287 non-null  object
2   state_po              4287 non-null  object
3   state_fips            4287 non-null  int64
4   state_cen             4287 non-null  int64
5   state_ic              4287 non-null  int64
6   office                4287 non-null  object
7   candidate             4000 non-null  object
8   party_detailed        3831 non-null  object
9   writein               4284 non-null  object
10  candidatevotes        4287 non-null  int64
11  totalvotes            4287 non-null  int64
12  party_simplified       4287 non-null  object
```

```

13 party          3831 non-null    object
dtypes: int64(6), object(8)
memory usage: 469.0+ KB

```

```

[ ]: #Printing top 5 values
df.head()

```

```

[ ]:   year  state state_po  state_fips  state_cen  state_ic  office \
0  1976  ALABAMA      AL          1         63      41  US PRESIDENT
1  1976  ALABAMA      AL          1         63      41  US PRESIDENT
2  1976  ALABAMA      AL          1         63      41  US PRESIDENT
3  1976  ALABAMA      AL          1         63      41  US PRESIDENT
4  1976  ALABAMA      AL          1         63      41  US PRESIDENT

```

```

           candidate          party_detailed writein \
0          CARTER, JIMMY          DEMOCRAT    False
1          FORD, GERALD          REPUBLICAN    False
2          MADDOX, LESTER  AMERICAN INDEPENDENT PARTY    False
3  BUBAR, BENJAMIN ""BEN""          PROHIBITION    False
4          HALL, GUS          COMMUNIST PARTY USE    False

```

```

      candidatevotes  totalvotes  party_simplified          party
0          659170      1182850          DEMOCRAT          democrat
1          504070      1182850          REPUBLICAN          republican
2           9198      1182850          OTHER  american independent party
3           6669      1182850          OTHER          prohibition
4           1954      1182850          OTHER          communist party use

```

```

[ ]: #Printing unique values from two columns as specified
print(df['party'].unique())
print(df['party_simplified'].unique())

```

```

['democrat' 'republican' 'american independent party' 'prohibition'
 'communist party use' 'libertarian' nan 'independent' 'socialist workers'
 'american' 'peace & freedom' 'u.s. labor' 'no party affiliation'
 'socialist labor' 'american party of iowa' 'socialist u.s.a.'
 'conservative' 'human rights' "mccarthy '76" 'people's"
 'international development bank' 'industrial government party'
 'socialist' 'liberal party' 'free libertarian' 'labor'
 'constitution party' 'concerned citizens' 'statesman' 'citizens'
 'workers world' 'national unity campaign' 'anderson coalition'
 'nominated by petition' 'respect for life' 'right-to-life'
 'middle class candidate' 'down with lawyers' "natural people's league"
 'populist' 'alliance' 'united sovereign citizens' 'workers league'
 'big deal party' 'national unity' 'new alliance'
 'national economic recovery' 'third world assembly' 'solidarity'
 'patriotic party' 'other' 'progressive' 'grassroots' 'consumer'
 'write-in' 'united citizens' 'liberty union party' 'america first'

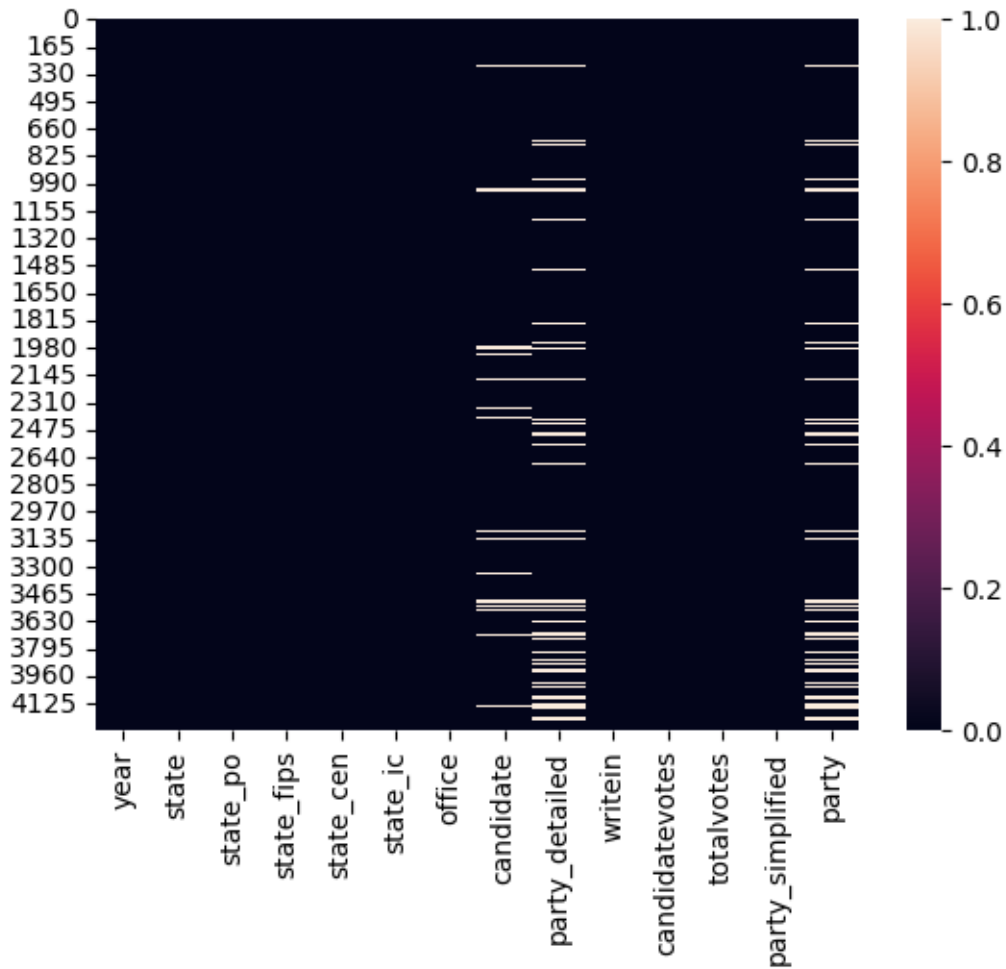
```

'independents for economic recovery' 'natural law' 'taxpayers party'
 'democrat/republican' 'unaffiliated' 'campaign for a new tomorrow'
 'equal justice and opportunity' 'more perfect democracy'
 'justice, industry, and agriculture' 'independent voters'
 'larouche for president party' 'socialist party usa'
 'tisch independent citizens' 'independent american' '6 million jobs'
 'ron daniels independent' 'america first populist'
 'independents for larouche' 'freedom for larouche'
 'labor-farm-laborista-agrario' 'third party' 'reform party' 'green'
 'u.s. taxpayers party' 'looking back party' 'liberty, ecology, community'
 'socialist equality party' 'independent grassroots' 'independence'
 'freedom' 'independent nomination' 'patriot party' 'unenrolled'
 'democratic-farmer-labor' 'reform party minnesota' 'citizens first'
 'working families' 'progressive/green' 'vermont grassroots'
 'alaskan independence party' 'american constitution party'
 'concerns of people' 'petitioning candidate' 'd.c. statehood green'
 'constitution party of florida' 'socialist party of florida'
 'protecting working families' 'green-rainbow' 'better life'
 'christian freedom party' 'nebraska party' 'peace and justice'
 'pacific green' 'nonpartisan' 'personal choice' 'wisconsin green'
 'socialism and liberation party' "america's independent party"
 'boston tea party' "heartquake '08" 'objectivist party'
 'u.s. pacifist party' 'ecology party of florida' 'new'
 'louisiana taxpayers party' 'vote here' 'peace party' 'independent green'
 'mountain party' 'we the people' 'justice' "america's party"
 'american third position' 'american independent'
 'constitutional government' 'nsa did 911' 'new mexico independent party'
 'american delta party' 'better for america' 'veterans party of america'
 'independent people of colorado' 'american solidarity party'
 'nutrition party' 'kotlikoff for president'
 'nonviolent resistance/pacifist' 'approval voting party'
 'new independent party iowa' 'legal marijuana now' 'workers world party'
 "women's equality" 'party for socialism and liberation'
 'life and liberty party' 'unity party' 'prohibition party'
 'progressive party' 'socialist workers party'
 'independent american party' 'non-affiliated' 'dc statehood green'
 'american shopping party' 'geneology know your family history party'
 'becoming one nation' 'c.u.p' 'freedom and prosperity'
 'life , liberty, constitution' 'the birthday party' 'bread and roses'
 'us taxpayers party' 'grumpy old patriots' 'american solidarity'
 'constitution' 'liberty union' 'boiling frog' 'bull moose'
 'approval voting']
 ['DEMOCRAT' 'REPUBLICAN' 'OTHER' 'LIBERTARIAN']

0.4 Checking for a heatmap on Null values

```
[ ]: import seaborn as sns
      #Checking null values and plotting them
      sns.heatmap(df.isnull())
```

```
[ ]: <AxesSubplot: >
```



```
[ ]: #Printing datatypes of each columns
      df.dtypes
```

```
[ ]: year          int64
      state         object
      state_po      object
      state_fips    int64
      state_cen     int64
      state_ic      int64
```

```

office          object
candidate       object
party_detailed  object
writein         object
candidatevotes  int64
totalvotes      int64
party_simplified object
party           object
dtype: object

```

```

[ ]: #Printing unique values in state, year and party columns
cols=["state","year","party"]
for i in cols:
    print("Number of unique values in ", i, " are : ",len(df[i].unique()), " :␣
↪",df[i].unique())

```

```

Number of unique values in state are : 51 : ['ALABAMA' 'ALASKA' 'ARIZONA'
'ARKANSAS' 'CALIFORNIA' 'COLORADO'
'CONNECTICUT' 'DELAWARE' 'DISTRICT OF COLUMBIA' 'FLORIDA' 'GEORGIA'
'HAWAII' 'IDAHO' 'ILLINOIS' 'INDIANA' 'IOWA' 'KANSAS' 'KENTUCKY'
'LOUISIANA' 'MAINE' 'MARYLAND' 'MASSACHUSETTS' 'MICHIGAN' 'MINNESOTA'
'MISSISSIPPI' 'MISSOURI' 'MONTANA' 'NEBRASKA' 'NEVADA' 'NEW HAMPSHIRE'
'NEW JERSEY' 'NEW MEXICO' 'NEW YORK' 'NORTH CAROLINA' 'NORTH DAKOTA'
'OHIO' 'OKLAHOMA' 'OREGON' 'PENNSYLVANIA' 'RHODE ISLAND' 'SOUTH CAROLINA'
'SOUTH DAKOTA' 'TENNESSEE' 'TEXAS' 'UTAH' 'VERMONT' 'VIRGINIA'
'WASHINGTON' 'WEST VIRGINIA' 'WISCONSIN' 'WYOMING']

```

```

Number of unique values in year are : 12 : [1976 1980 1984 1988 1992 1996
2000 2004 2008 2012 2016 2020]

```

```

Number of unique values in party are : 173 : ['democrat' 'republican'
'american independent party' 'prohibition'
'communist party use' 'libertarian' nan 'independent' 'socialist workers'
'american' 'peace & freedom' 'u.s. labor' 'no party affiliation'
'socialist labor' 'american party of iowa' 'socialist u.s.a.'
'conservative' 'human rights' "mccarthy '76" "people's"
'international development bank' 'industrial government party'
'socialist' 'liberal party' 'free libertarian' 'labor'
'constitution party' 'concerned citizens' 'statesman' 'citizens'
'workers world' 'national unity campaign' 'anderson coalition'
'nominated by petition' 'respect for life' 'right-to-life'
'middle class candidate' 'down with lawyers' "natural people's league"
'populist' 'alliance' 'united sovereign citizens' 'workers league'
'big deal party' 'national unity' 'new alliance'
'national economic recovery' 'third world assembly' 'solidarity'
'patriotic party' 'other' 'progressive' 'grassroots' 'consumer'
'write-in' 'united citizens' 'liberty union party' 'america first'
'independents for economic recovery' 'natural law' 'taxpayers party'
'democrat/republican' 'unaffiliated' 'campaign for a new tomorrow'
'equal justice and opportunity' 'more perfect democracy'

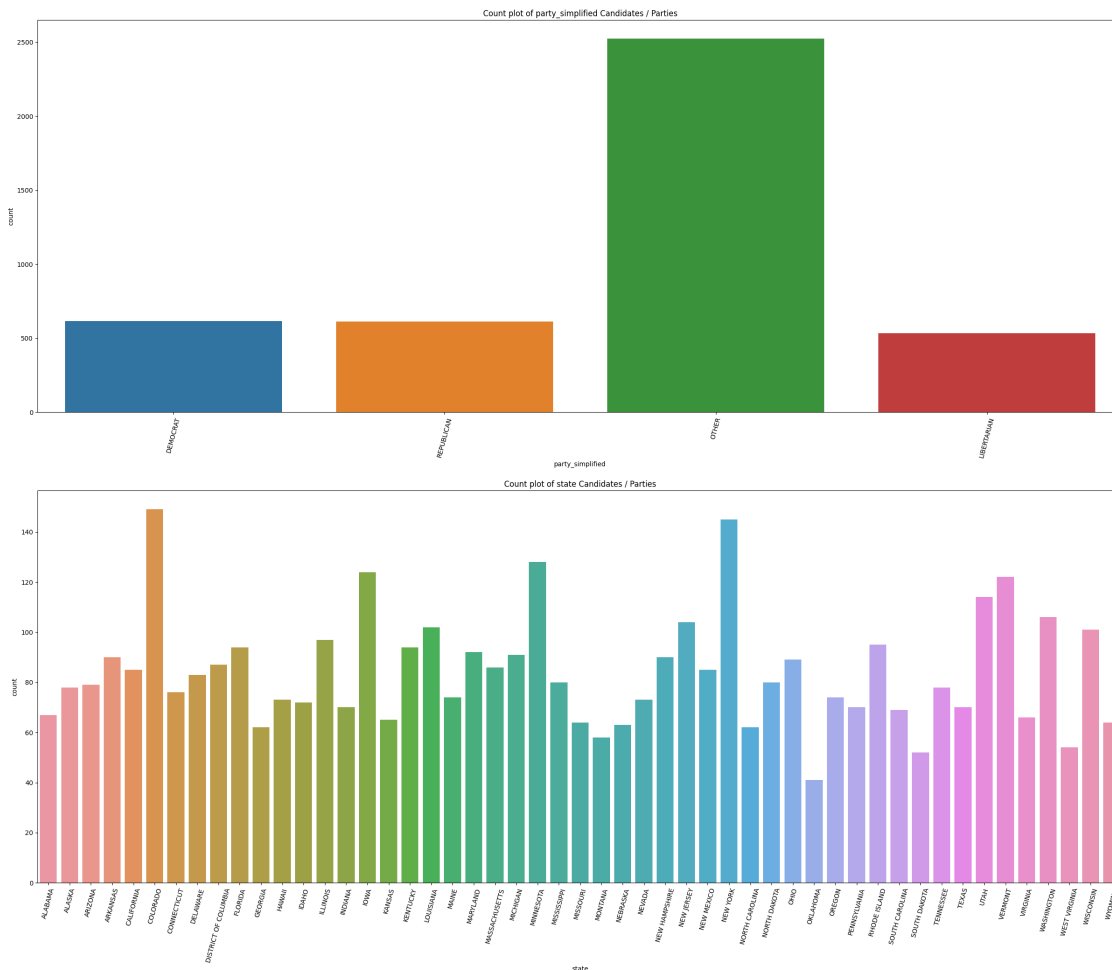
```

'justice, industry, and agriculture' 'independent voters'
 'larouche for president party' 'socialist party usa'
 'tisch independent citizens' 'independent american' '6 million jobs'
 'ron daniels independent' 'america first populist'
 'independents for larouche' 'freedom for larouche'
 'labor-farm-laborista-agrario' 'third party' 'reform party' 'green'
 'u.s. taxpayers party' 'looking back party' 'liberty, ecology, community'
 'socialist equality party' 'independent grassroots' 'independence'
 'freedom' 'independent nomination' 'patriot party' 'unenrolled'
 'democratic-farmer-labor' 'reform party minnesota' 'citizens first'
 'working families' 'progressive/green' 'vermont grassroots'
 'alaskan independence party' 'american constitution party'
 'concerns of people' 'petitioning candidate' 'd.c. statehood green'
 'constitution party of florida' 'socialist party of florida'
 'protecting working families' 'green-rainbow' 'better life'
 'christian freedom party' 'nebraska party' 'peace and justice'
 'pacific green' 'nonpartisan' 'personal choice' 'wisconsin green'
 'socialism and liberation party' "america's independent party"
 'boston tea party' "heartquake '08" 'objectivist party'
 'u.s. pacifist party' 'ecology party of florida' 'new'
 'louisiana taxpayers party' 'vote here' 'peace party' 'independent green'
 'mountain party' 'we the people' 'justice' "america's party"
 'american third position' 'american independent'
 'constitutional government' 'nsa did 911' 'new mexico independent party'
 'american delta party' 'better for america' 'veterans party of america'
 'independent people of colorado' 'american solidarity party'
 'nutrition party' 'kotlikoff for president'
 'nonviolent resistance/pacifist' 'approval voting party'
 'new independent party iowa' 'legal marijuana now' 'workers world party'
 'women's equality' 'party for socialism and liberation'
 'life and liberty party' 'unity party' 'prohibition party'
 'progressive party' 'socialist workers party'
 'independent american party' 'non-affiliated' 'dc statehood green'
 'american shopping party' 'geneology know your family history party'
 'becoming one nation' 'c.u.p' 'freedom and prosperity'
 'life , liberty, constitution' 'the birthday party' 'bread and roses'
 'us taxpayers party' 'grumpy old patriots' 'american solidarity'
 'constitution' 'liberty union' 'boiling frog' 'bull moose'
 'approval voting']

```

[ ]: cols = ["party_simplified", "state"]
fig, axes = plt.subplots(nrows=2, ncols=1, figsize=[30, 24])
for i in range(0, len(cols)):
    axes[i] = sns.countplot(x = cols[i], data = df, ax=axes[i])
    axes[i].set_xticklabels(axes[i].get_xticklabels(), rotation=75)
    axes[i].set_title("Count plot of " + cols[i] + " Candidates / Parties")
    print("*****50)
  
```

```
*****
*****
```

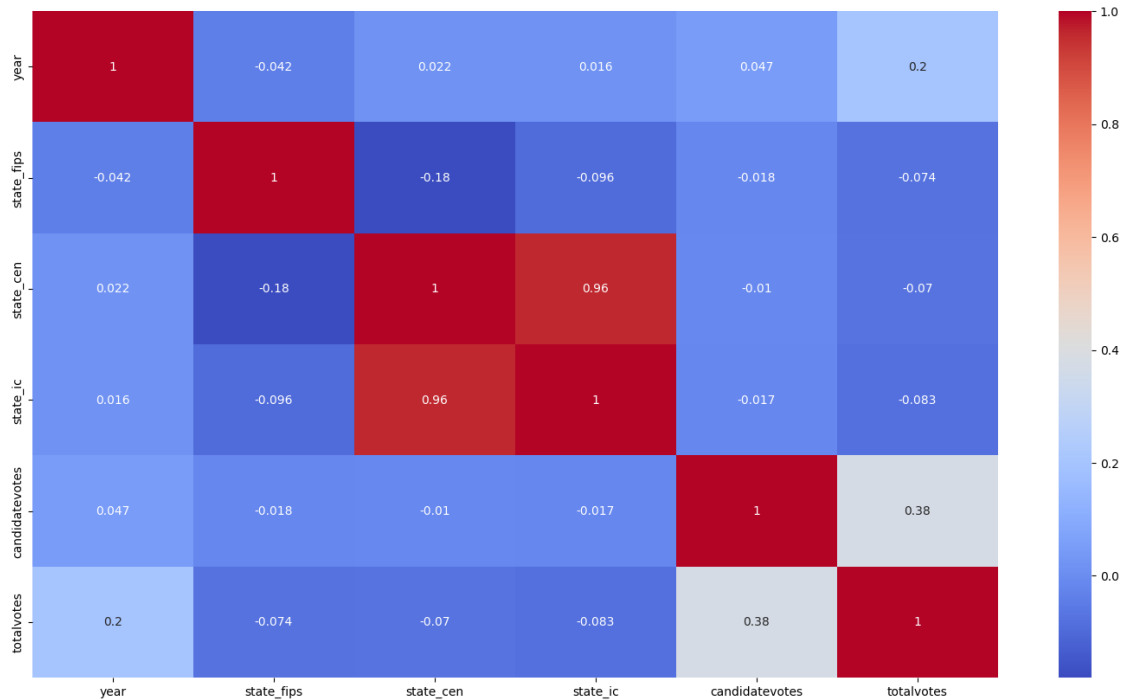


```
[ ]: #Plotting a heatmap to check most correlated columns
plt.figure(figsize=(18,10))
sns.heatmap(df.corr(),cmap='coolwarm',annot=True)
```

```
/var/folders/md/bwhknjsx4t3fc85j8f04l_w40000gn/T/ipykernel_24345/238214810.py:3:
FutureWarning:
```

The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.

```
[ ]: <AxesSubplot: >
```

0.4.1 Bivariate Analysis

```
[ ]: df.columns
```

```
[ ]: Index(['year', 'state', 'state_po', 'state_fips', 'state_cen', 'state_ic',
            'office', 'candidate', 'party_detailed', 'writein', 'candidatevotes',
            'totalvotes', 'party_simplified', 'party'],
           dtype='object')
```

```
[ ]: #Plotting scatter plots for voting method
cols=['writein', 'candidatevotes',
      'totalvotes']
#Creating a figure and axes for subplots, with 1 row and 3 columns, and a
↳specific size
fig, axes = plt.subplots(nrows=1,ncols=3,figsize=[20,12])
#Iteration through each column for the column list keeping cols on x axis and
↳year on y axis respectively in each plot
for i in range(0,len(cols)):
    axes[i]=sns.scatterplot( x= cols[i], y="year",
                            data = df,hue="writein",
                            size = "totalvotes",
                            sizes=(50,200),
                            hue_norm=(0, 6),
                            cmap="viridis",
```

```

ax=axes[i])
#Setting up titles for subplots
axes[i].set_title("Total Votes vs "+cols[i])

```

/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/seaborn/relational.py:573: UserWarning:

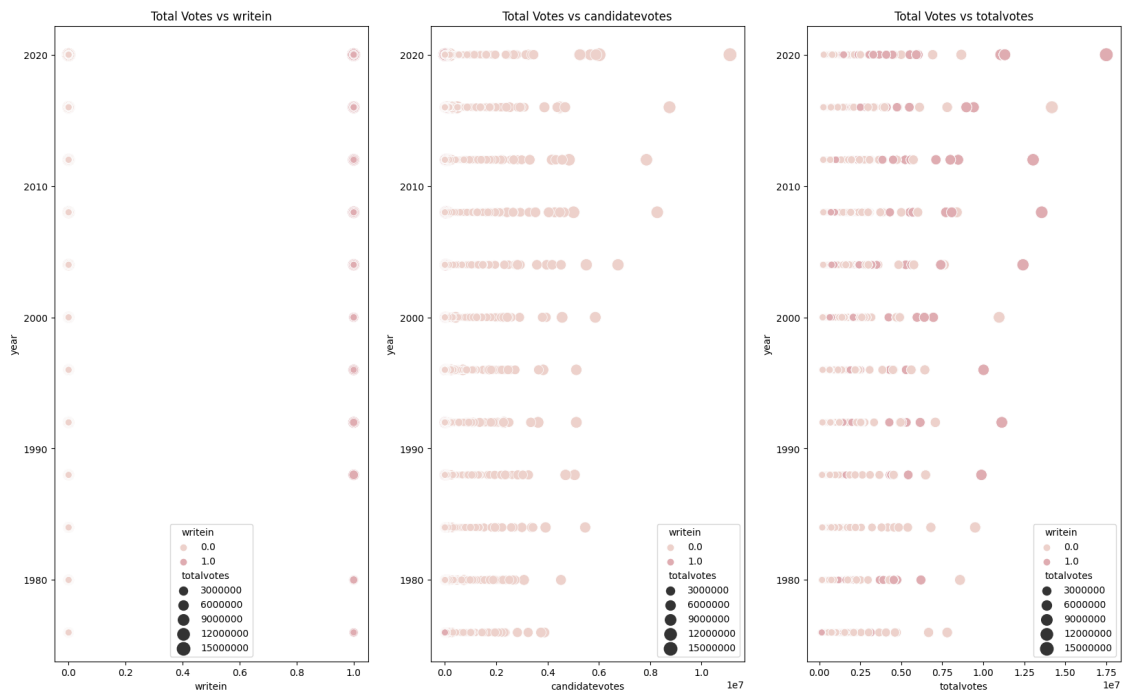
No data for colormapping provided via 'c'. Parameters 'cmap' will be ignored

/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/seaborn/relational.py:573: UserWarning:

No data for colormapping provided via 'c'. Parameters 'cmap' will be ignored

/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/seaborn/relational.py:573: UserWarning:

No data for colormapping provided via 'c'. Parameters 'cmap' will be ignored

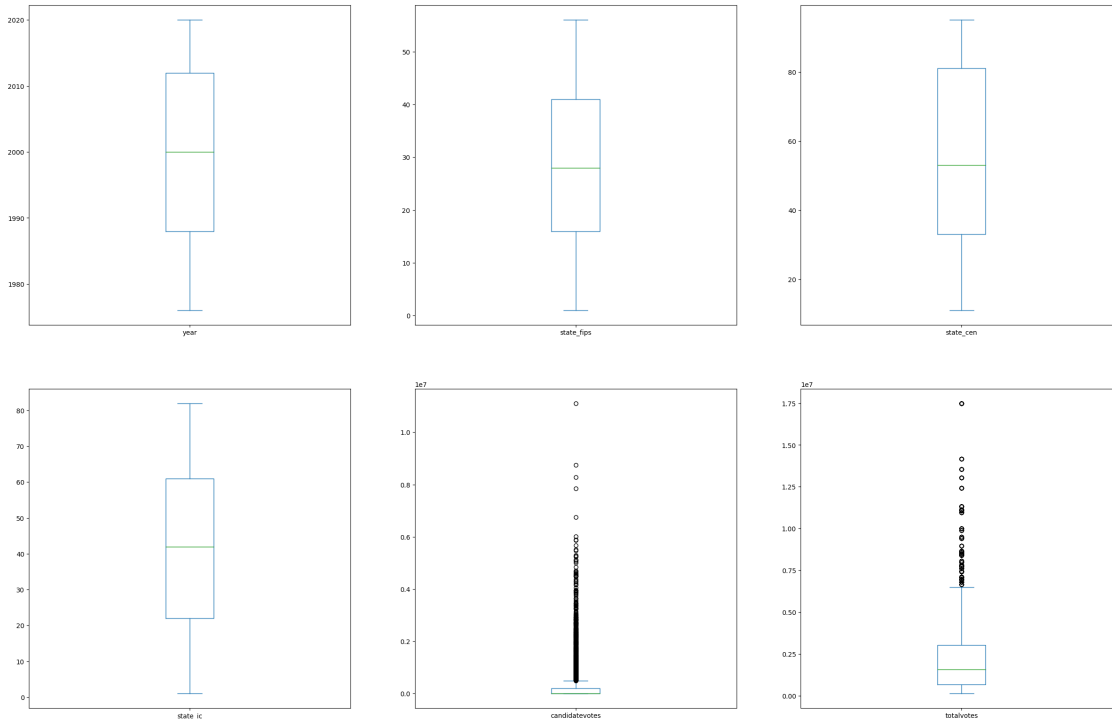


```

[ ]: #Plotting Boxplots for checking mathematical values and outliers for each
      ↪ numerical value in the dataframe
df.plot(kind="box",subplots=True,layout=(3,3),figsize=(30,30))

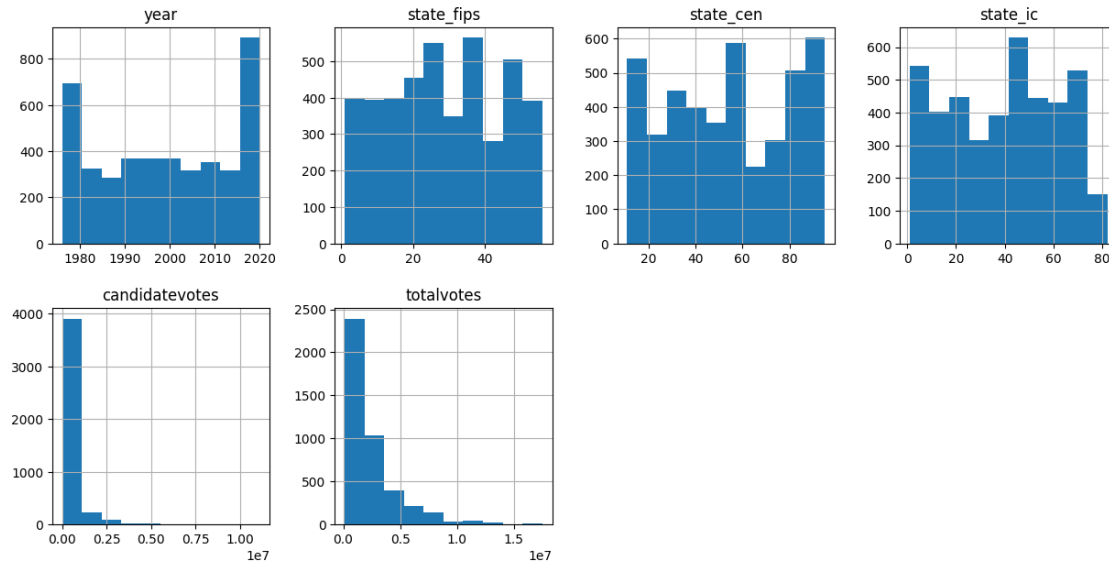
```

```
[ ]: year                AxesSubplot(0.125,0.653529;0.227941x0.226471)
    state_fips           AxesSubplot(0.398529,0.653529;0.227941x0.226471)
    state_cen            AxesSubplot(0.672059,0.653529;0.227941x0.226471)
    state_ic             AxesSubplot(0.125,0.381765;0.227941x0.226471)
    candidatevotes       AxesSubplot(0.398529,0.381765;0.227941x0.226471)
    totalvotes           AxesSubplot(0.672059,0.381765;0.227941x0.226471)
    dtype: object
```



```
[ ]: #Plotting histograms to see spread of various columns
df.hist(figsize=(15,15), layout=(4,4), bins=10)
```

```
[ ]: array([[<AxesSubplot: title={'center': 'year'}>,
            <AxesSubplot: title={'center': 'state_fips'}>,
            <AxesSubplot: title={'center': 'state_cen'}>,
            <AxesSubplot: title={'center': 'state_ic'}>],
            [<AxesSubplot: title={'center': 'candidatevotes'}>,
            <AxesSubplot: title={'center': 'totalvotes'}>, <AxesSubplot: >,
            <AxesSubplot: >],
            [<AxesSubplot: >, <AxesSubplot: >, <AxesSubplot: >,
            <AxesSubplot: >],
            [<AxesSubplot: >, <AxesSubplot: >, <AxesSubplot: >,
            <AxesSubplot: >]], dtype=object)
```



```
[ ]: df.skew()
```

```
/var/folders/md/bwhknjsx4t3fc85j8f04l_w40000gn/T/ipykernel_24345/1665899112.py:1
: FutureWarning:
```

The default value of `numeric_only` in `DataFrame.skew` is deprecated. In a future version, it will default to `False`. In addition, specifying `'numeric_only=None'` is deprecated. Select only valid columns or specify the value of `numeric_only` to silence this warning.

```
[ ]: year          -0.048871
     state_fips     0.029350
     state_cen      -0.026119
     state_ic       -0.080905
     writein        2.471984
     candidatevotes  4.506062
     totalvotes     2.216780
     dtype: float64
```

0.4.2 Let us see how many parties have contested in each state since 1976 elections

```
[ ]: #Grouping the DataFrame by state and counting the number of unique parties per
     ↪state
     parties_per_state = df.groupby('state')['party_detailed'].nunique().
     ↪reset_index().sort_values('party_detailed',ascending = False)
     #Creating a bar plot using Plotly Express (px) to visualize the number of
     ↪unique parties per state
```

```
fig = px.bar(parties_per_state, x='state', y='party_detailed',
             color='party_detailed', height=600)
#Displaying the plot
fig.show()
```

0.4.3 Let us see how many votes have been polled by 3rd to 20th parties since 1976 elections

0.4.4 Top-5 major parties

```
[ ]: #Groupby dataframe by party_detailed and calculated sum of candidate votes for
      each party
vote_per_party= df.groupby('party_detailed')['candidatevotes'].sum().
      reset_index().sort_values('candidatevotes',ascending = False)
#Plotting a bar graph keeping party name on x axis and total votes on y axis
      with the color scheme
fig = px.bar(vote_per_party.head(5), x='party_detailed', y='candidatevotes',
             color='candidatevotes', height=600)
#Plotting the bar graph using plotly express
fig.show()
```

0.4.5 Top 20 non major parties

```
[ ]: # Filtering out rows corresponding to major parties (DEMOCRAT, REPUBLICAN,
      INDEPENDENT, LIBERTARIAN)
# "~" is used to negate the condition, keeping rows where 'party_detailed' is
      not in the specified list
vote_per_party_nodemrep= vote_per_party[~vote_per_party["party_detailed"].
      isin(["DEMOCRAT","REPUBLICAN","INDEPENDENT","LIBERTARIAN"])]
#Selecting top 20 parties according to highest number of candidate votes among
      non major parties
fig = px.bar(vote_per_party_nodemrep.head(20), x='party_detailed',
             y='candidatevotes', color='candidatevotes', height=600)
#Plotting the bar graph using plotly express
fig.show()
```

0.5 Let us see top-25 vote getters across 40 years

```
[ ]: #Groupby DataFrame 'df' by 'candidate' and calculate the sum of
      'candidatevotes' for each candidate
vote_per_candidate= df.groupby('candidate')['candidatevotes'].sum().
      reset_index().sort_values('candidatevotes',ascending = False)
#Selecting top 25 candidates by the highest number of candidate votes
fig = px.bar(vote_per_candidate.head(25), x='candidate', y='candidatevotes',
             color='candidatevotes', height=600)
#Plotting the bar graph using plotly express
```

```
fig.show()
```

0.5.1 Trying a pie chart for visualising states contribution

```
[ ]: #Grouping the dataframe 'df' by state and calculating sum of total votes for
      ↪ each state
df2=df.groupby('state')['totalvotes'].sum().reset_index().
      ↪sort_values('totalvotes',ascending = False)
# Representing only large states keeping states less than 200 million votes to
      ↪ 'other states' category
df2.loc[df2['totalvotes'] < 2.e8, 'state'] = 'Other States'
#Creating a pie chart to see the distribution of votes across states
fig = px.pie(df2, values='totalvotes', names='state', title='Total Votes')
#Plotting the pie chart using plotly express
fig.show()
```

```
[ ]: #Creating a distribution plot using Holoviews for the 'totalvotes' column of
      ↪ DataFrame 'df2'
#Setting the plot variables which are customisable to appearance
pr = hv.Distribution(df2['totalvotes']).opts(title="Statewide Vote
      ↪Distribution", color="purple", xlabel="State Vote size", ylabel="Vote
      ↪Density",xformatter='%d', xrotation=90)\
      .opts(opts.Distribution(width=400,
      ↪height=300,tools=['hover'],show_grid=True))
#Displaying the plot using holoviews
pr
```

```
[ ]: :Distribution [totalvotes] (Density)
```

0.6 Let us try some more plotly charts

```
[ ]: #Plotting a bar plot for each state showing total number of votes according to
      ↪ year given by a color configuration
fig = px.histogram(df, x='state', y='totalvotes', color='year')
fig
```

```
[ ]: fig = px.histogram(df, x='state', y='totalvotes', color='party_simplified')
fig
```

```
[ ]: fig = px.histogram(df, x='state', y='totalvotes', color='year',
      ↪facet_row='party_simplified', height=3600, hover_name='year')
fig
```

```
[ ]: #Creating a plotly express plot for parallel_categories year and parties
fig = px.parallel_categories(df, dimensions=['year', 'party_simplified'],
      ↪color='year', color_continuous_scale='armyrose')
```

```

#Updating layout of the plot to enable autosizing
fig.update_layout(autosize=True)
#Displaying the plot
fig

```

```

[ ]: #Plotting a density contour plot using plotly express
#This plot provides a visual representation of the density of data points
↳across different states ('state') and years ('year'), with the contour lines
↳representing the density of candidate votes ('candidatevotes').
#The color of the contour lines indicates the party affiliation
↳('party_simplified'). Hovering over the plot reveals additional information
↳about the candidate votes.
fig = px.density_contour(df, x='state', y='year', z='candidatevotes',
↳color='party_simplified', hover_name='candidatevotes')
#Displaying the plot
fig

```

0.7 Let us try some Maps

0.7.1 States and Total votes

```

[ ]: #Creating a choropleth map using Plotly Express
#locations: Column specifying the location codes or names ('state_po' - State
↳postal codes)
#color: Column specifying the values to be represented by color ('totalvotes' -
↳Total votes)
#range_color: Range of values to map to the color scale (0 to 8,000,000)
#locationmode: Specify the location mode ('USA-states' for US states)
#scope: Scope of the map ('usa' for USA)
#title: Title of the choropleth map
fig = px.choropleth(df, locations='state_po', color="totalvotes",
                    range_color=(0, 8000000),
                    locationmode = 'USA-states',
                    scope="usa",
                    title='USA Presidential Vote Counts'
                    )
#Updating the layout of plot to adjust margins
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
#Showing the choropleth map
fig.show()

```

0.7.2 Top 5 political parties and states

```

[ ]: #Group the DataFrame by 'party' and calculate the sum of 'candidatevotes' for
↳each party
vote_per_party= df.groupby('party')['candidatevotes'].sum().reset_index().
↳sort_values('candidatevotes',ascending = False)

```

```

#Iterating over the top 5 parties
for index, row in vote_per_party.head(5).iterrows():
    party_name = row['party']
    title_head = 'USA Presidential Vote Counts - ' + party_name
    print(title_head)
    df_r = df.loc[df['party'] == party_name]
    fig = px.choropleth(df_r,
                        locations='state_po',
                        color="candidatevotes",
                        range_color=(0, 8000000),
                        locationmode = 'USA-states',
                        title=title_head,
                        scope="usa")
    fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
    fig.show()

```

USA Presidential Vote Counts - democrat

USA Presidential Vote Counts - republican

USA Presidential Vote Counts - independent

USA Presidential Vote Counts - libertarian

USA Presidential Vote Counts - reform party

0.7.3 Democratic party across several states with timestamp

```

[ ]: #Creating a copy of the dataframe to avoid modifying the original
df_temp = df.copy()

#Filtering the dataframe to only include rows where 'party' is in the specified
↪ list
df_temp = df_temp[df_temp.party.isin(['democrat', 'republican', 'independent',
                                       'reform', 'libertarian', 'alliance'])]

#Grouping the data by 'state', 'state_po', 'party', and 'year'
#and sum the 'candidatevotes' for each group
df_temp = df_temp.groupby(['state', 'state_po', 'party',
                           ↪ 'year'])["candidatevotes"].sum()

#Resetting the index to create new columns from the previous index
df_temp = df_temp.reset_index()

#Filtering the data to only include rows where 'party' is 'democrat'
datadump = df_temp[df_temp['party'] == 'democrat']

#Creating a choropleth using plotly express
px.choropleth(datadump,

```



```

        locations='state_po', # Use 'state_po' for locations
        color="candidatevotes", # Color based on 'candidatevotes'
        animation_frame="year", # Animate by 'year'
        color_continuous_scale="Inferno", # Use 'Inferno' color scale
        locationmode='USA-states', # Set location mode to US states
        scope="usa", # Set scope to USA
        range_color=(100000, 5000000), # Color range from 100,000 to 5,000,000
        title='Total Votes by Party - Democrats', # Set title
        height=600 # Set chart height
    )

```

0.7.4 Independent parties across several states

```

[ ]: #Filtering data for independent candidates and sort by year:
datadump = df_temp[df_temp['party'] == 'independent'].sort_values('year',
    ↪ascending=True)
#Creating an animated choropleth map using Plotly Express:
px.choropleth(datadump,
    locations = 'state_po',
    color="candidatevotes",
    animation_frame="year",
    color_continuous_scale="Inferno",
    locationmode='USA-states',
    scope="usa",
    range_color=(1000, 100000),
    title='Total Votes by Party - Independents',
    height=600
)

```

0.7.5 Libertarian party across several states across each year span

```

[ ]: #Filtering data for the Libertarian party and sort by year:
datadump = df_temp[df_temp['party'] == 'libertarian'].sort_values('year',
    ↪ascending=True)
#Creating an animated choropleth map using Plotly Express:
px.choropleth(datadump,
    locations = 'state_po',
    color="candidatevotes",
    animation_frame="year",
    color_continuous_scale="Inferno",
    locationmode='USA-states',
    scope="usa",
    range_color=(1000, 100000),
    title='Total Votes by Party - Libertarian Party',
    height=600
)

```

0.7.6 Republican party across several states for each year

```
[ ]: #Filtering data for Republican party and sort by year
datadump = df_temp[df_temp['party'] == 'republican'].sort_values('year',
    ↪ascending=True)
#Creating an animated choropleth map using Plotly Express
px.choropleth(datadump,
               locations = 'state_po',
               color="candidatevotes",
               animation_frame="year",
               color_continuous_scale="Inferno",
               locationmode='USA-states',
               scope="usa",
               range_color=(100000, 5000000),
               title='Total Votes by Party - Republican',
               height=600
            )
```

0.7.7 Alliance party across several states for each year

```
[ ]: #Filtering data for 'alliance' party and sort by 'year':
datadump = df_temp[df_temp['party'] == 'alliance'].sort_values('year',
    ↪ascending=True)
#Creating an animated choropleth map:
px.choropleth(datadump,
               locations = 'state_po',
               color="candidatevotes",
               animation_frame="year",
               color_continuous_scale="Inferno",
               locationmode='USA-states',
               scope="usa",
               range_color=(1000, 50000),
               title='Total Votes by Party - Alliance Party',
               height=600
            )
```

0.8 Let us try other visualizations

```
[ ]: df_temp = df
state_group = df_temp.groupby(['year', 'state', 'state_po', 'party', 'candidate']).
    ↪agg({'candidatevotes': 'sum'})
state_pcts = state_group.groupby(level=0).apply(lambda x:
    100 * x / float(x.sum()))
state_pcts = state_pcts.reset_index()
```

```
/var/folders/md/bwhknjsx4t3fc85j8f04l_w40000gn/T/ipykernel_24345/3929514380.py:3
: FutureWarning:
```

Not prepending group keys to the result index of transform-like apply. In the future, the group keys will be included in the index, regardless of whether the applied function returns a like-indexed object.

To preserve the previous behavior, use

```
>>> .groupby(..., group_keys=False)
```

To adopt the future behavior and silence this warning, use

```
>>> .groupby(..., group_keys=True)
```

```
[ ]: #Filtering data for Democrat party
datadump = state_pcts[state_pcts['party'] == 'democrat']
#Creating an animated choropleth map using Plotly Express
px.choropleth(datadump,
               locations = 'state_po',
               color="candidatevotes",
               animation_frame="year",
               color_continuous_scale="Inferno",
               locationmode='USA-states',
               scope="usa",
               range_color=(0.01, 0.90),
               title='Total Votes by Party - Democrat Party',
               height=600
            )
```

```
[ ]: #Let us define parties we want to track - We will pick the top-2 parties
parties = ['democrat', 'republican','libertarian','independent']
top2_df=df.loc[df['party'].isin(parties)]
#Let us build pivot tables we shall use for various examples
#Pivot-1 indexed on year and state on the aggregated sum on party votes
table = pd.pivot_table(top2_df, values='candidatevotes', index=['year',
↳ 'state'],
                       columns=['party'], aggfunc=np.sum)
#Pivot-2 indexed on year for the aggregated sum on party votes
table2 = pd.pivot_table(top2_df, values='candidatevotes', index=['year'],
                       columns=['party'], aggfunc=np.sum)
#Pivot-3 indexed on state for the aggregated sum on party votes
table3 = pd.pivot_table(top2_df, values='candidatevotes', index=['state'],
                       columns=['party'], aggfunc=np.sum)
source = ColumnDataSource(df)
```

```
[ ]: #Let us build a line graph for republican and democrat votes
p = figure()
p.line(x='year', y='republican', source=table2,
       line_width=2, color=rand_color(),legend_label='Republican')
```

```

p.circle_dot(x='year', y='republican', source=table2,
             size=10, color=rand_color(),legend_label='Republican')
p.line(x='year', y='democrat', source=table2,
       line_width=2, color=rand_color(),legend_label='Democrat')
p.diamond_cross(x='year', y='democrat', source=table2,
               size=10, color=rand_color(),legend_label='Democrat')
p.line(x='year', y='libertarian', source=table2,
       line_width=2, color=rand_color(),legend_label='libertarian')
p.diamond(x='year', y='libertarian', source=table2,
          size=10, color=rand_color(),legend_label='libertarian')
p.line(x='year', y='independent', source=table2,
       line_width=2, color=rand_color(),legend_label='Independents')
p.diamond(x='year', y='independent', source=table2,
          size=10, color=rand_color(),legend_label='Independents')
p.title.text = 'A Sample Line Chart of total votes collected by Republicans ,
↳ Democrats, Green Party and Independents in the US Presidential Elections'
p.xaxis.axis_label = 'Year'
p.yaxis.axis_label = 'Votes'
p.legend.location = 'top_left'
p.legend.title = 'Parties'
#Let us remove the scientific formatting
p.yaxis.formatter=NumeralTickFormatter(format="00")
#Let us add a hovering tool
hover = HoverTool()
hover.tooltips=[
    ('Year', '@year')
]
p.add_tools(hover)
#show the plot
show(p)

```

```

/var/folders/md/bwhknjsx4t3fc85j8f04l_w40000gn/T/ipykernel_24345/2471642404.py:5
: BokehDeprecationWarning:

```

'circle_dot() method' was deprecated in Bokeh 3.4.0 and will be removed, use "scatter(marker='circle_dot', ...) instead" instead.

```

/var/folders/md/bwhknjsx4t3fc85j8f04l_w40000gn/T/ipykernel_24345/2471642404.py:9
: BokehDeprecationWarning:

```

'diamond_cross() method' was deprecated in Bokeh 3.4.0 and will be removed, use "scatter(marker='diamond_cross', ...) instead" instead.

```

/var/folders/md/bwhknjsx4t3fc85j8f04l_w40000gn/T/ipykernel_24345/2471642404.py:1
3: BokehDeprecationWarning:

```

'diamond() method' was deprecated in Bokeh 3.4.0 and will be removed, use "scatter(marker='diamond', ...) instead" instead.

/var/folders/md/bwhknjsx4t3fc85j8f04l_w40000gn/T/ipykernel_24345/2471642404.py:1
7: BokehDeprecationWarning:

'diamond() method' was deprecated in Bokeh 3.4.0 and will be removed, use
"scatter(marker='diamond', ...) instead" instead.

- For this, we shall build a pivot table on the data and plot the trends. Bar Chart and Difference in Data
- Following program leverages the power of pandas and builds a new column for difference in votes and uses it for plotting the vertical bar chart to show the trend over the years.

```
[ ]: #Defining a function to calculate absolute difference
```

```
def abs_min(a, b): # a, b are input arrays  
    return np.abs(a[:,None]-b).min(axis=0)  
#Defining a function to calculate difference  
def diff_min(a, b): # a, b are input arrays  
    return (a-b)
```

```
[ ]: #Converting the pivot table to a pandas dataframe
```

```
table31_df = pd.DataFrame(table3.to_records())  
#Finding the difference between democrats and republicans  
diffs = diff_min(table31_df.republican.values, table31_df.democrat.values)  
#Creating a new column with difference in votes  
table31_df.insert(3, "difference", diffs, True)  
#Let us choose the top Republican states  
table3_df = table31_df.sort_values(by='republican', ascending=False).head(15)  
states = table3_df['state']  
#Converting the votes as multiples of millions  
republican = table3_df['republican']/1000000  
democrat = table3_df['democrat'] / 1000000  
difference = table3_df['difference'] / 1000000
```

```
[ ]: #Building a dataset for plotting
```

```
data = {'states' : states, 'republican' : republican,  
        'democrat' : democrat, 'difference' : difference }  
source = ColumnDataSource(data=data)  
#Plot a vertical bar chart with dodge by a parameter  
p2 = figure(height=1500, width=1200,x_range=states,  
            y_range=(difference.min(),max(republican.max(), democrat.max())),  
            title="State wise Votes - vote size in millions")  
p2.vbar(x=dodge('states', -0.25, range=p2.x_range), top='republican', width=0.2,  
        source=source, color="#ff0011", legend_label="Republican")  
p2.vbar(x=dodge('states', +0.0, range=p2.x_range), top='democrat', width=0.2,  
        source=source, color="#1100ff", legend_label="Democrat")  
p2.vbar(x=dodge('states', +0.25, range=p2.x_range), top='difference', width=0.2,  
        source=source, color="gold", legend_label="Difference")
```

```

#Plot a line plots
p2.line(x=dodge('states', -0.25, range=p2.x_range), y='republican',
        source=source, line_width=2, color='red', legend_label='Republican')
p2.circle_dot(x=dodge('states', -0.25, range=p2.x_range), y='republican',
              source=source, size=4, color=rand_color(), legend_label='Republican')
p2.line(x='states', y='democrat',
        source=source, line_width=2, color='navy', legend_label='Democrat')
p2.diamond_cross(x='states', y='democrat',
                 source=source, size=4, color=rand_color(), legend_label='Democrat')
p2.line(x=dodge('states', +0.25, range=p2.x_range), y='difference',
        source=source, line_width=2, color='gold', legend_label='Difference')
p2.diamond_dot(x=dodge('states', +0.25, range=p2.x_range), y='difference',
               source=source, size=4, color=rand_color(), legend_label='Difference')
#Add Formatting aspects
p2.x_range.range_padding = 0.1
p2.xgrid.grid_line_color = None
p2.legend.location = "top_right"
p2.legend.orientation = "vertical"
p2.yaxis.formatter=NumeralTickFormatter(format="00")
p2.xaxis.major_label_orientation = math.pi/2
#Add Hover
hover = HoverTool()
hover.tooltips=[('States', '@states')]
p2.add_tools(hover)
#Show the plot
show(p2)

```

/var/folders/md/bwhknjsx4t3fc85j8f04l_w40000gn/T/ipykernel_24345/642709768.py:17
: BokehDeprecationWarning:

'circle_dot() method' was deprecated in Bokeh 3.4.0 and will be removed, use
"scatter(marker='circle_dot', ...) instead" instead.

/var/folders/md/bwhknjsx4t3fc85j8f04l_w40000gn/T/ipykernel_24345/642709768.py:21
: BokehDeprecationWarning:

'diamond_cross() method' was deprecated in Bokeh 3.4.0 and will be removed, use
"scatter(marker='diamond_cross', ...) instead" instead.

/var/folders/md/bwhknjsx4t3fc85j8f04l_w40000gn/T/ipykernel_24345/642709768.py:25
: BokehDeprecationWarning:

'diamond_dot() method' was deprecated in Bokeh 3.4.0 and will be removed, use
"scatter(marker='diamond_dot', ...) instead" instead.

- For the this, let us see how both the parties performed in one of their bellwether states over the years.

- We shall take one state for each party to plot the performance and show the trend.

```
[ ]: from bokeh.models import FixedTicker
from bokeh.palettes import Turbo256
table41_df = pd.DataFrame(table.to_records())
diffs = abs_min(table41_df.republican.values, table41_df.democrat.values)
table41_df.insert(3, "difference", diffs, True)
table4_df = table41_df.sort_values(by='democrat', ascending=False)
states = table4_df['state']
#Change the values in 1000s of vote
republican = table4_df['republican']/1000
democrat = table4_df['democrat'] / 1000
difference = table4_df['difference'] / 1000
year = table4_df['year'].sort_values(ascending=True).unique()
table4_df.republican.fillna(0)
table4_df.difference.fillna(0)
table4_df.democrat.fillna(0)
tab4_pivot = pd.pivot_table(table4_df,
    ↪values=['republican','democrat','difference'],
                                index=['year'], columns=['state'], aggfunc=np.sum,
    ↪margins=True)
flat_tab4_df = pd.DataFrame(tab4_pivot.to_records())
tabcols = [flat_tab4_df.columns]
years = table4_df['year']
states = table4_df['state']
republican = table4_df['republican']/1000
democrat = table4_df['democrat'] / 1000
difference = table4_df['difference'] / 1000
votegroup = ['democrat', 'republican','difference']
source = ColumnDataSource(data=dict(x=tabcols, democrat=democrat,
    ↪republican=republican, difference=difference,))
p4 = figure(width=900, height=800) #, x_axis_type="datetime")
years = flat_tab4_df.year
values = flat_tab4_df["('democrat', 'CALIFORNIA')"]
rvalues = flat_tab4_df["('republican', 'CALIFORNIA')"]
#Plotting for a democrat state - California
p4.vbar(years, top = values, width = .9, fill_alpha = .5, line_alpha = .5,
    fill_color = rand_color(), line_color=rand_color(), line_dash='dashed')
p4.line(years,rvalues,line_width=4,line_color="red",line_dash="dotted")
p4.circle(years,rvalues,radius=.2,fill_color='yellow',line_color=rand_color())
hover = HoverTool()
hover.tooltips=[('Votes', '@top'),('Year', '@x')]
p4.x_range.range_padding = 0.1
p4.xgrid.grid_line_color = None
p4.yaxis.formatter=NumeralTickFormatter(format="00")
p4.xaxis.major_label_orientation = math.pi/2
p4.add_tools(hover)
```

```

show(p4)

p5 = figure(width=900, height=800)
years = flat_tab4_df.year
#Plotting for a republican state - Texas
values = flat_tab4_df["('republican', 'TEXAS')"]
dvalues = flat_tab4_df["('democrat', 'TEXAS')"]
divalues = flat_tab4_df["('difference', 'TEXAS')"]
p5.vbar(years, top = values, width = .9, fill_alpha = .5, line_alpha = .5,
        fill_color = rand_color(), line_color=rand_color(), line_dash='dotted')
p5.line(years,dvalues,line_width=4,line_color="navy",line_dash="dotted")
p5.circle(years,dvalues,radius=.2,fill_color='yellow',line_color=rand_color())
p5.line(years,divalues,line_width=2,line_color=rand_color(),line_dash="dashdot")
hover = HoverTool()
hover.tooltips=[('Votes', '@top'),('Year', '@x')]
p5.x_range.range_padding = 0.1
p5.xgrid.grid_line_color = None
p5.yaxis.formatter=NumeralTickFormatter(format="00")
p5.xaxis.major_label_orientation = math.pi/2
p5.add_tools(hover)
show(p5)

```

/var/folders/md/bwhknjsx4t3fc85j8f04l_w40000gn/T/ipykernel_24345/427574933.py:26

: BokehUserWarning:

ColumnDataSource's columns must be of the same length. Current lengths:
('democrat', 612), ('difference', 612), ('republican', 612), ('x', 1)

```

[ ]: #Preparing the data for the plot
data = [go.Bar(
    x=df['year'].unique(),
    y=df.groupby(['year','state'])['candidate'].count(),
    textposition = 'auto',
    marker=dict(
        color=df['totalvotes'],
        line=dict(
            color='rgb(8,48,107)',
            width=1.5),
    ),
    opacity=0.6
)]

#Defining the layout of the plot
layout = {
    'xaxis': {'title': 'Year'},
    'yaxis': {'title': 'No. of Candidates'},
    'barmode': 'relative',

```



```

    'title': 'Total Number of Candidates'
};
#Creating and displaying the plot
iplot({'data': data, 'layout': layout})

```

```

[ ]: #Data for the chart is defined as a list of go.Bar objects
data = [go.Bar(
    #X-axis data: unique state values from the 'df' dataframe's 'state' column
    x=df['state'].unique(),

    #Y-axis data: sum of 'totalvotes' grouped by 'year' and 'party' from 'df'
    y=df.groupby(['year', 'party'])['totalvotes'].sum(),

    #Text positioning for data labels on bars (automatic positioning)
    textposition='auto',

    #Defining bar appearance
    marker=dict(
        color='mediumvioletred', #Bar color
        line=dict( #Line around bars
            color='rgb(8,48,107)', #Line color
            width=1.5 #Line width
        )
    ),

    #Bar transparency
    opacity=0.6
)]

#Layout for the chart is (titles, mode, etc.)
layout = {
    'xaxis': {'title': 'Year'}, #X-axis title
    'yaxis': {'title': 'Total Votes'}, #Y-axis title
    'barmode': 'relative', #Stacking bars on top of each other (proportionally)
    'title': 'Total Number of Candidates' #Chart title
};
#Generating the chart using plotly's iplot function
iplot({'data': data, 'layout': layout})

```

```

[ ]: #Creating a new pandas DataFrame with election data
df_simple =
    ↪df[['year', 'state', 'state_po', 'candidate', 'candidatevotes', 'totalvotes', 'party_simplified',
    ↪copy()

```

```

[ ]: #Creating a new column 'party' and copy values from 'party_simplified'
df_simple['party'] = df_simple['party_simplified']
#Deleting the original 'party_simplified' column

```

```
del df_simple['party_simplified']
#Calculating the percentage of votes for each candidate
df_simple['percentage'] = round((df_simple['candidatevotes'] /
    df_simple['totalvotes'])*100,2)
df_simple.head()
```

```
[ ]: year    state state_po    candidate    candidatevotes \
0  1976  ALABAMA    AL    CARTER, JIMMY    659170
1  1976  ALABAMA    AL    FORD, GERALD    504070
2  1976  ALABAMA    AL    MADDOX, LESTER    9198
3  1976  ALABAMA    AL    BUBAR, BENJAMIN ""BEN""    6669
4  1976  ALABAMA    AL    HALL, GUS    1954

    totalvotes    party    percentage
0    1182850    DEMOCRAT    55.73
1    1182850    REPUBLICAN    42.61
2    1182850    OTHER    0.78
3    1182850    OTHER    0.56
4    1182850    OTHER    0.17
```

```
[ ]: import warnings
warnings.filterwarnings('ignore')

#Making a new dataframe and printing relevant columns needed for further
analysis
df_simple2 = df_simple.
    groupby(['state','year','candidate','percentage','party'])['candidatevotes'].
    sum().to_frame('candidatevotes').reset_index()
print(df_simple2.head())
```

```
    state    year    candidate    percentage    party \
0  ALABAMA    1976    BUBAR, BENJAMIN ""BEN""    0.56    OTHER
1  ALABAMA    1976    CARTER, JIMMY    55.73    DEMOCRAT
2  ALABAMA    1976    FORD, GERALD    42.61    REPUBLICAN
3  ALABAMA    1976    HALL, GUS    0.17    OTHER
4  ALABAMA    1976    MACBRIDE, ROGER    0.13    LIBERTARIAN

    candidatevotes
0            6669
1          659170
2          504070
3            1954
4            1481
```

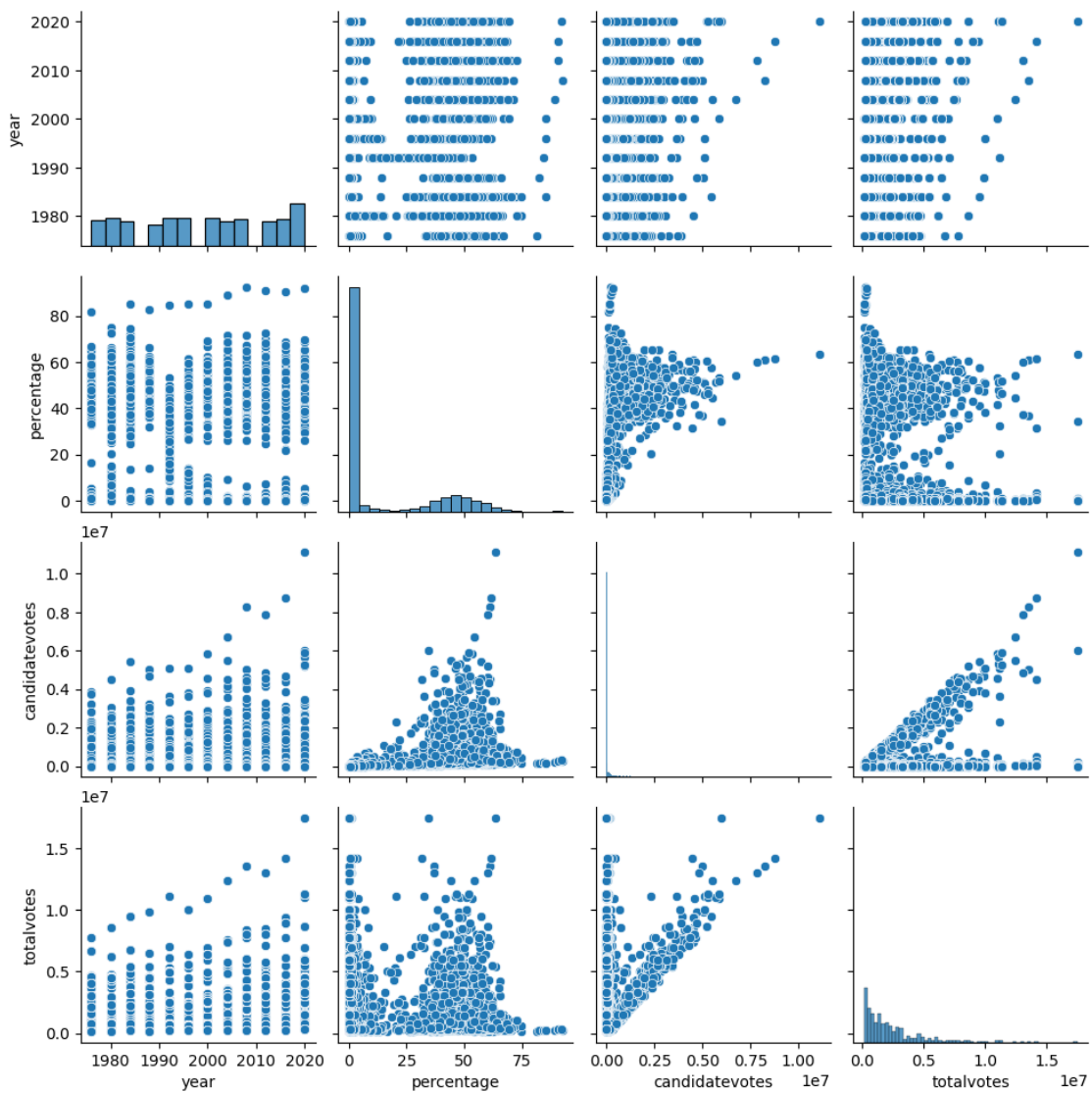
```
[ ]: #Creating a colormap using the "tab10" palette, specifically for mapping a
continuous variable
sns.color_palette("tab10", as_cmap=True)
```

```
[ ]:
```



```
[ ]: #Creating a pair plot using seaborn
sns.pairplot(df_simple[['state', 'year', 'party', 'percentage', 'candidatevotes', 'totalvotes']])
```

```
[ ]: <seaborn.axisgrid.PairGrid at 0x31219b7d0>
```



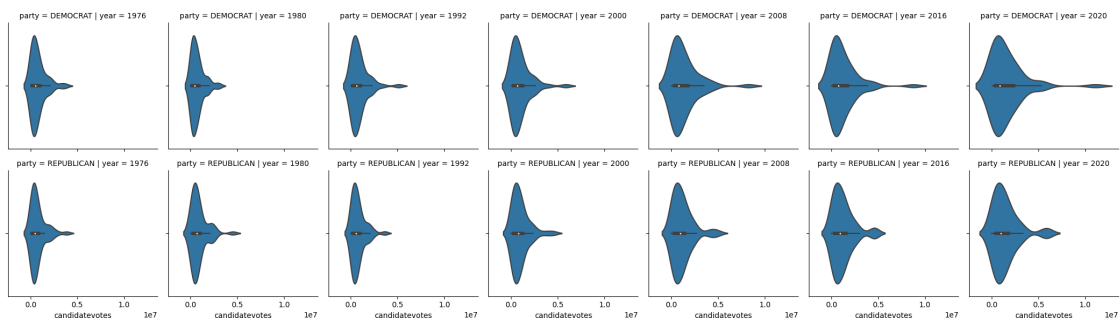
```
[ ]: dft = df_simple[df_simple['party'].isin(['DEMOCRAT', 'REPUBLICAN'])]

#Choosing the years that had a change in leadership
dft = dft[dft['year'].isin([1976,1980,1992, 2000, 2008, 2016, 2020])]
```

```
[ ]: g = sns.FacetGrid(dft, row="party", col="year")
g.map(sns.violinplot, "candidatevotes")

#g2 = sns.FacetGrid(dft, row="year", col="state")
#g2.map(sns.boxenplot, x=dft["state"].head(5), y=dft["candidatevotes"], width=0.
↪05)
```

```
[ ]: <seaborn.axisgrid.FacetGrid at 0x312b5e390>
```



```
[ ]: #Importing the Plotly Express library for creating interactive visualizations
import plotly.express as px

#Creating a scatter plot using Plotly Express
fig = px.scatter(df_simple, # DataFrame containing election data
                x="candidatevotes", # X-axis: Number of votes for the
↪candidate
                y="percentage", # Y-axis: Percentage of votes for the
↪candidate
                color="party", # Color the dots based on the party of the
↪candidate
                animation_frame="year", # Animate the plot by year
                animation_group="state", # Group the animation by state
                size="candidatevotes", # Resize the dots based on the number
↪of votes
                hover_name="state", # Display the state name on hover
                log_x=True, # Use a logarithmic scale for the x-axis
                size_max=55, # Set a maximum size for the dots
                range_x=[10000, 10000000], # Limit the x-axis range to 10,000
↪to 10,000,000 votes
                range_y=[0, 80], # Limit the y-axis range to 0 to 80 percent
```

```
        color_continuous_scale=px.colors.sequential.Viridis # Use the
↪ Viridis color scale
    )
#Displaying the interactive plot
fig.show()
```