30538 Problem Set 5: Web Scraping

Tarini Dewan

2024-11-09

Due 11/9 at 5:00PM Central. Worth 100 points + 10 points extra credit.

Submission Steps (10 pts)

1. This problem set is a paired problem set.
2. Play paper, scissors, rock to determine who goes first. Call that person *Partner 1*.
    - Partner 1 (name and cnet ID): Tarini Dewan, tarinidewan
3. Partner 1 will accept the `ps5` and then share the link it creates with their partner. You can only share it with one partner so you will not be able to change it after your partner has accepted.
4. "This submission is our work alone and complies with the 30538 integrity policy." Add your initials to indicate your agreement: TD
5. "I have uploaded the names of anyone else other than my partner and I worked with on the problem set **here**": TD (1 point)
6. Late coins used this pset: 1 Late coins left after submission: 2
7. Knit your `ps5.qmd` to an PDF file to make `ps5.pdf`,
    - The PDF should not be more than 25 pages. Use `head()` and re-size figures when appropriate.
8. (Partner 1): push `ps5.qmd` and `ps5.pdf` to your github repo.
9. (Partner 1): submit `ps5.pdf` via Gradescope. Add your partner on Gradescope.
10. (Partner 1): tag your submission in Gradescope

```
import pandas as pd
import altair as alt
import time
import requests
from bs4 import BeautifulSoup
import numpy as np
from datetime import datetime

import warnings
warnings.filterwarnings('ignore')
alt.renderers.enable("png")
```

```
RendererRegistry.enable('png')
```

Step 1: Develop initial scraper and crawler

1. Scraping (PARTNER 1)

```
url = 'https://oig.hhs.gov/fraud/enforcement/'
response = requests.get(url)
soup = BeautifulSoup(response.content, 'lxml')

# find all relevant li tags that contain an h2 element
tag = soup.find_all('li')
li_tag = soup.find_all(lambda t: t.name == 'li' and t.find_all('h2'))
len(li_tag)

# display the first li_tag
#li_content = [li.contents for li in li_tag]
#for item in li_content[0]:
#  print(item)

# create empty lists to store the data
titles = []
dates = []
categories = []
links = []

# loop over each li_tag and save all the relevant attribute info
for li in li_tag:
```

```
  # get title
  title = li.find('a').text.strip() if li.find('a') else None
  titles.append(title)
  # get date
  date = li.find('span').text if li.find('span') else None
  dates.append(date)
  # get category
  cat = li.find('ul').text if li.find('ul') else None
  categories.append(cat)
  # get link
  link = li.find('a').get('href') if li.find('a') else None
  links.append(link)
# Attribution: ChatGPT
# Query: I want to loop over each li_tag and save the title for each
 ↪  li_tag[0], li_tag[1], and so on

# add entire path to the link
for i in range(len(links)):
  links[i] = 'https://oig.hhs.gov' + links[i]

# create dataframe to store the data
oig_df = pd.DataFrame({
  'Title': titles,
  'Date': dates,
  'Category': categories,
  'Link': links})

# save the data
oig_df.to_csv(f'/Users/tarini_dewan/Desktop/UChicago/Python_2/oig_df.csv')
```

```
# read in the oig data (for the first page) to display head()
oig_df =
 ↪  pd.read_csv('/Users/tarini_dewan/Desktop/UChicago/Python_2/oig_df.csv')
print(oig_df.head())
```

```
   Unnamed: 0                                              Title  \
0           0  Pharmacist and Brother Convicted of $15M Medic...
1           1  Boise Nurse Practitioner Sentenced To 48 Month...
2           2  Former Traveling Nurse Pleads Guilty To Tamper...
3           3  Former Arlington Resident Sentenced To Prison ...
4           4  Paroled Felon Sentenced To Six Years For Fraud...
```

```
              Date                    Category  \
0  November 8, 2024  Criminal and Civil Actions
1  November 7, 2024  Criminal and Civil Actions
2  November 7, 2024  Criminal and Civil Actions
3  November 7, 2024  Criminal and Civil Actions
4  November 7, 2024  Criminal and Civil Actions

                                                Link
0  https://oig.hhs.gov/fraud/enforcement/pharmaci...
1  https://oig.hhs.gov/fraud/enforcement/boise-nu...
2  https://oig.hhs.gov/fraud/enforcement/former-t...
3  https://oig.hhs.gov/fraud/enforcement/former-a...
4  https://oig.hhs.gov/fraud/enforcement/paroled-...
```

2. Crawling (PARTNER 1)

```
# create an empty list to store the agencies
agencies = []

# new url becomes the input into a new request
for i in range(len(links)):
  url = links[i]
  #print(url)
  response = requests.get(url)
  soup = BeautifulSoup(response.content, 'lxml')

# extract the agency text from the link
  ul_tag = soup.find('ul', class_ = 'usa-list usa-list--unstyled margin-y-2')
  agency = np.nan
  for ul in ul_tag:
    if 'Agency' in ul.get_text():
      agency = ul.get_text().replace('Agency:', '').replace('November 7,
 ↪  2024;', '').strip()

  agencies.append(agency)

# add the agency column to df
oig_df['Agency'] = agencies
```

```
# save the data
oig_df.to_csv(f'/Users/tarini_dewan/Desktop/UChicago/Python_2/oig_df_full.csv')
```

```
# read in the oig data (with agency info) to display head()
oig_df_full =
 ↪  pd.read_csv('/Users/tarini_dewan/Desktop/UChicago/Python_2/oig_df_full.csv')
print(oig_df_full.head())
```

```
   Unnamed: 0.1  Unnamed: 0  \
0             0           0
1             1           1
2             2           2
3             3           3
4             4           4

                                               Title              Date  \
0  Pharmacist and Brother Convicted of $15M Medic...  November 8, 2024
1  Boise Nurse Practitioner Sentenced To 48 Month...  November 7, 2024
2  Former Traveling Nurse Pleads Guilty To Tamper...  November 7, 2024
3  Former Arlington Resident Sentenced To Prison ...  November 7, 2024
4  Paroled Felon Sentenced To Six Years For Fraud...  November 7, 2024

                    Category  \
0  Criminal and Civil Actions
1  Criminal and Civil Actions
2  Criminal and Civil Actions
3  Criminal and Civil Actions
4  Criminal and Civil Actions

                                                Link  \
0  https://oig.hhs.gov/fraud/enforcement/pharmaci...
1  https://oig.hhs.gov/fraud/enforcement/boise-nu...
2  https://oig.hhs.gov/fraud/enforcement/former-t...
3  https://oig.hhs.gov/fraud/enforcement/former-a...
4  https://oig.hhs.gov/fraud/enforcement/paroled-...

                                      Agency
0                      U.S. Department of Justice
1        U.S. Attorney's Office, District of Idaho
2  U.S. Attorney's Office, District of Massachusetts
3  U.S. Attorney's Office, Eastern District of Vi...
4  U.S. Attorney's Office, Middle District of Flo...
```

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code (PARTNER 2)
- Step 1: Check if inputs make sense
  - Make sure year is 2013 or later and month is between 1 and 12
  - Create a date using the month and year
- Step 2: Set up our starting point
  - Create a flag to keep track if we're still looking (date_not_reached)
  - Start at page 0
  - Create empty lists to store attributes
- Step 3: Main Loop: Keep going until we hit our target date
  - While we haven't reached our date:
    - If it's the first page (page 0), use main website URL
    - If not first page, add page number to URL
    - Get the webpage content
    - Find all entries on the page
    - For each entry we find:
      - Get the date
      - If this date is older than what we want, stop looking
      - If not, save the attributes (excluding agency)
    - Move to next page
    - Wait half a second
- Step 4: Put all our collected data into a dataframe and save as a CSV file
- b. Create Dynamic Scraper (PARTNER 2) The final dataframe contains 1554 enforcement actions. The earliest enforcement action recorded is a Criminal and Civil Action dated 03/01/2023.

```
def scraper(month, year):
  # set the condition for the year input
  if year < 2013:
    return print('Restrict the year to post 2013')
  # set the condition for the month input
  if month > 12 or month < 0:
    return print('Invalid month')
```

```
# set target date
date_target = datetime(year, month, 1)

date_not_reached = True
page_number = 0

# create empty lists to store the data
titles = []
dates = []
categories = []
agencies = []
links =[]
while date_not_reached == True:

  if page_number == 0:
    # use the URL based on the page number
    url = 'https://oig.hhs.gov/fraud/enforcement/'
  else:
    url = f'https://oig.hhs.gov/fraud/enforcement/?page={page_number}'
  response = requests.get(url)
  soup = BeautifulSoup(response.content, 'lxml')

  # find all relevant li tags that contain an h2 element
  tag = soup.find_all('li')
  li_tag = soup.find_all(lambda t: t.name == 'li' and t.find_all('h2'))

  # loop over each li_tag and save all the relevant attribute info
  for li in li_tag:
    # get date
    date = li.find('span').text if li.find('span') else None
    date = datetime.strptime(date, '%B %d, %Y')
    # check if the date is before the target date
    if date < date_target:
      date_not_reached = False
      break
    dates.append(date)
    # get title
    title = li.find('a').text.strip() if li.find('a') else 'None'
    titles.append(title)
    # get links
    link = li.find('a').get('href') if li.find('a') else None
    links.append(link)
```

```
      # get category
      if len(li.find('ul')) >1:
        ul_tag = li.find('ul')
        joined_text = ', '.join(li.get_text(strip=True) for li in
↪  ul_tag.find_all('li'))
        categories.append(joined_text)
      else:
        cat = li.find('ul').text if li.find('ul') else None
        categories.append(cat)

    # Increment the page number
    page_number+=1
    # Wait for half a second before the next request
    time.sleep(0.5)

  # Create a DataFrame with the collected data
  ym_df = pd.DataFrame({
  'Title': titles,
  'Date': dates,
  'Category': categories,
  'Link': links})

  # save df to a CSV file

↪  ym_df.to_csv(f'/Users/tarini_dewan/Desktop/UChicago/Python_2/enforcement_actions_{year}_

  return ym_df

# run scraper function for January 2023
scraper(1, 2023)
```

```
jan_2023 =
↪  pd.read_csv('/Users/tarini_dewan/Desktop/UChicago/Python_2/enforcement_actions_2023_1.csv
print(jan_2023.head())
```

```
   Unnamed: 0                                              Title        Date
   \
0           0  Pharmacist and Brother Convicted of $15M Medic...  2024-11-08
1           1  Boise Nurse Practitioner Sentenced To 48 Month...  2024-11-07
2           2  Former Traveling Nurse Pleads Guilty To Tamper...  2024-11-07
3           3  Former Arlington Resident Sentenced To Prison ...  2024-11-07
```

```
4           4  Paroled Felon Sentenced To Six Years For Fraud...  2024-11-07

                     Category  \
0  Criminal and Civil Actions
1  Criminal and Civil Actions
2  Criminal and Civil Actions
3  Criminal and Civil Actions
4  Criminal and Civil Actions

                                                 Link
0  /fraud/enforcement/pharmacist-and-brother-conv...
1  /fraud/enforcement/boise-nurse-practitioner-se...
2  /fraud/enforcement/former-traveling-nurse-plea...
3  /fraud/enforcement/former-arlington-resident-s...
4  /fraud/enforcement/paroled-felon-sentenced-to-...
```

- c. Test Partner's Code (PARTNER 1) The final dataframe contains 3042 enforcement actions. The earliest enforcement action recorded is a Criminal and Civil Action dated 04/01/21.

```
# run scraper function for January 2021
scraper(1, 2021)
```

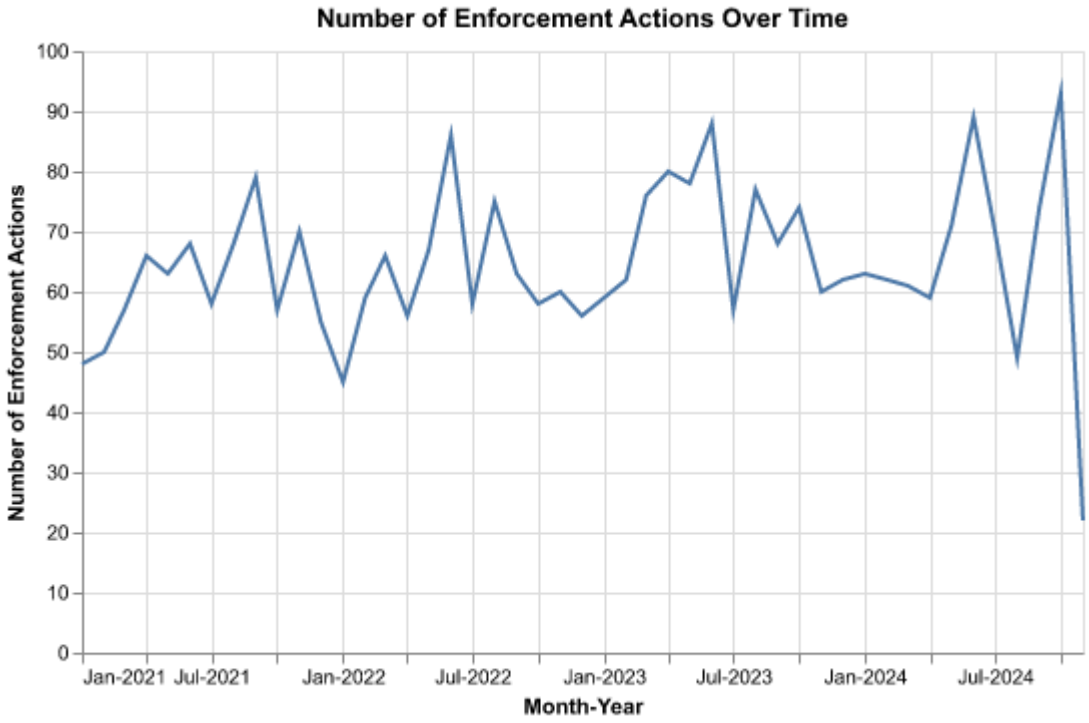
Step 3: Plot data based on scraped data

1. Plot the number of enforcement actions over time (PARTNER 2)

```
# read in the data
enforce_df =
 ↪  pd.read_csv('/Users/tarini_dewan/Desktop/UChicago/Python_2/enforcement_actions_2021_1.csv
print(enforce_df.head())

#  line chart that shows the number of enforcement actions over time
enf_time = alt.Chart(enforce_df).mark_line().encode(
    alt.X('yearmonth(Date):T', title = 'Month-Year').axis(format = '%b-%Y'),
    alt.Y('count(Category):Q',
    axis = alt.Axis(labelFontSize=10), sort = '-x', title = 'Number of
 ↪  Enforcement Actions')).properties(
    title = 'Number of Enforcement Actions Over Time'
    ).properties(
```

```
    width=500,
    height=300,
)
enf_time
```

```
   Unnamed: 0                                              Title        Date
   \
0           0  Pharmacist and Brother Convicted of $15M Medic...  2024-11-08
1           1  Boise Nurse Practitioner Sentenced To 48 Month...  2024-11-07
2           2  Former Traveling Nurse Pleads Guilty To Tamper...  2024-11-07
3           3  Former Arlington Resident Sentenced To Prison ...  2024-11-07
4           4  Paroled Felon Sentenced To Six Years For Fraud...  2024-11-07

                    Category  \
0  Criminal and Civil Actions
1  Criminal and Civil Actions
2  Criminal and Civil Actions
3  Criminal and Civil Actions
4  Criminal and Civil Actions

                                                Link
0  /fraud/enforcement/pharmacist-and-brother-conv...
1  /fraud/enforcement/boise-nurse-practitioner-se...
2  /fraud/enforcement/former-traveling-nurse-plea...
3  /fraud/enforcement/former-arlington-resident-s...
4  /fraud/enforcement/paroled-felon-sentenced-to-...
```




```
from tqdm import tqdm
agency_full = []

# add full link path to the links collected for headlines from each page
enforce_df['Full Link'] = enforce_df['Link'].apply(lambda x:
 ↪  'https://oig.hhs.gov' + x)
full_links = list(enforce_df['Full Link'])

# loop over each full link
for link in tqdm(full_links):
  url = link
  response = requests.get(url)
  soup = BeautifulSoup(response.content, 'lxml')

# extract the agency text from the links
  ul_tag = soup.find('ul', class_ = 'usa-list usa-list--unstyled margin-y-2')
  agency = np.nan
  for ul in ul_tag:
    if 'Agency' in ul.get_text():
      agency = ul.get_text().replace('Agency:', '').strip()
```

```
  # Append the extracted agency to the agency_full list
  agency_full.append(agency)

# add agency column to main January 2021 dataframe
enforce_df['Agency'] = agency_full
enforce_df.to_csv('/Users/tarini_dewan/Desktop/UChicago/Python_2/enforce_df.csv')
```

2. Plot the number of enforcement actions categorized: (PARTNER 1)

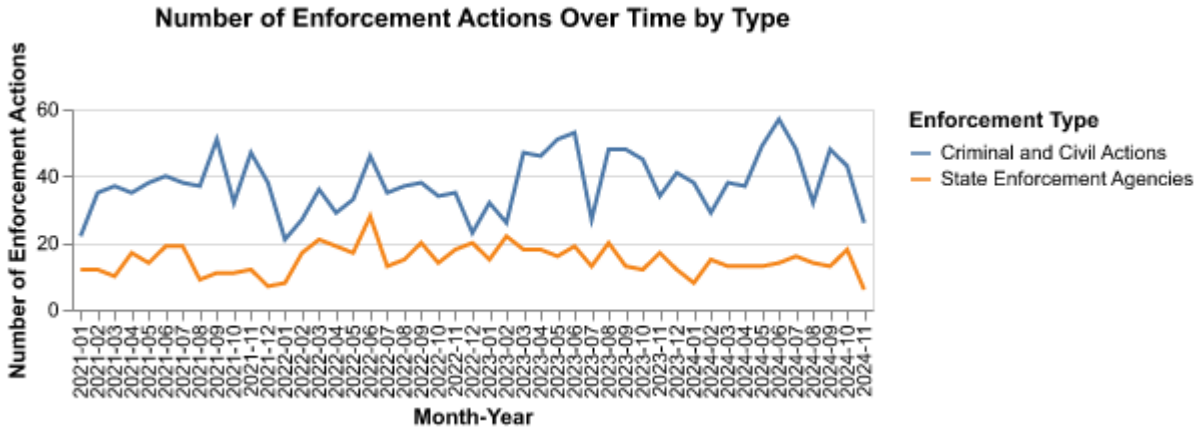
- based on "Criminal and Civil Actions" vs. "State Enforcement Agencies"

```
enforce_df =
 ↪  pd.read_csv('/Users/tarini_dewan/Desktop/UChicago/Python_2/enforce_df.csv')
print(enforce_df.head())

# create binary columns for "Criminal and Civil Actions" and "State
 ↪  Enforcement Agencies"
enforce_df['Criminal and Civil Actions'] =
 ↪  enforce_df['Category'].apply(lambda x: 1 if 'Criminal and Civil Actions'
 ↪  in x else 0)
enforce_df['State Enforcement Agencies'] =
 ↪  enforce_df['Category'].apply(lambda x: 1 if 'State Enforcement Agencies'
 ↪  in x else 0)

# Convert 'Date' column to datetime and extract 'year_month'
enforce_df['Date'] = pd.to_datetime(enforce_df['Date'], errors='coerce')
enforce_df['year_month'] = enforce_df['Date'].dt.strftime('%Y-%m')

# Group the data by 'year_month' and sum the counts
enforce_df_group = enforce_df.groupby('year_month').agg({
    'Criminal and Civil Actions': 'sum',
    'State Enforcement Agencies': 'sum'
}).reset_index()

# Reshape data from wide to long format for Altair
enforce_df_melt = pd.melt(
    enforce_df_group,
    id_vars=['year_month'],
    value_vars=['Criminal and Civil Actions', 'State Enforcement Agencies'],
    var_name='Action Type',
    value_name='Count'
```

```
)

# Create a line chart using Altair
chart = alt.Chart(enforce_df_melt).mark_line().encode(
    x=alt.X('year_month:N', title='Month-Year'),
    y=alt.Y('Count:Q', title='Number of Enforcement Actions'),
    color=alt.Color('Action Type:N', title='Enforcement Type')
).properties(
    title='Number of Enforcement Actions Over Time by Type',
    width=400,
    height=100
)
chart
```

```
   Unnamed: 0.1  Unnamed: 0  \
0             0           0
1             1           1
2             2           2
3             3           3
4             4           4

                                               Title        Date  \
0  Pharmacist and Brother Convicted of $15M Medic...  2024-11-08
1  Boise Nurse Practitioner Sentenced To 48 Month...  2024-11-07
2  Former Traveling Nurse Pleads Guilty To Tamper...  2024-11-07
3  Former Arlington Resident Sentenced To Prison ...  2024-11-07
4  Paroled Felon Sentenced To Six Years For Fraud...  2024-11-07

                    Category  \
0  Criminal and Civil Actions
1  Criminal and Civil Actions
2  Criminal and Civil Actions
3  Criminal and Civil Actions
4  Criminal and Civil Actions

                                                Link  \
0  /fraud/enforcement/pharmacist-and-brother-conv...
1  /fraud/enforcement/boise-nurse-practitioner-se...
2  /fraud/enforcement/former-traveling-nurse-plea...
3  /fraud/enforcement/former-arlington-resident-s...
4  /fraud/enforcement/paroled-felon-sentenced-to-...
```

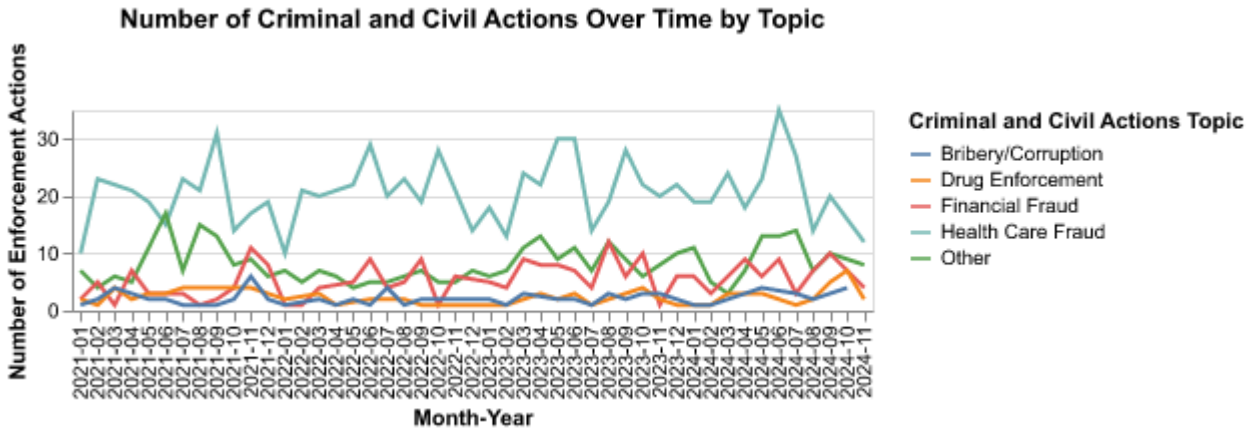
```
                                                Full Link  \
0  https://oig.hhs.gov/fraud/enforcement/pharmaci...
1  https://oig.hhs.gov/fraud/enforcement/boise-nu...
2  https://oig.hhs.gov/fraud/enforcement/former-t...
3  https://oig.hhs.gov/fraud/enforcement/former-a...
4  https://oig.hhs.gov/fraud/enforcement/paroled-...

                                               Agency
0                          U.S. Department of Justice
1  November 7, 2024; U.S. Attorney's Office, Dist...
2  U.S. Attorney's Office, District of Massachusetts
3  U.S. Attorney's Office, Eastern District of Vi...
4  U.S. Attorney's Office, Middle District of Flo...
```



- based on five topics

```
# define keywords for each topic
enforce_df_crim = enforce_df[enforce_df['Criminal and Civil Actions'] == 1]
topics = {
    'Health Care Fraud': ['health', 'insurance', 'medical', 'medicare',
    ↪  'prescription'],
    'Financial Fraud': ['bank', 'financial', 'transaction', 'investment',
    ↪  'fraud'],
    'Drug Enforcement': ['drug', 'trafficking', 'narcotic', 'opioid'],
    'Bribery/Corruption': ['bribery', 'corruption', 'bribe', 'kickback'],
    'Other': []
}

# function to categorize topics based on keywords
```

```
def categorize_topic(title):
    title = title.lower()
    for topic, keywords in topics.items():
        if any(keyword in title for keyword in keywords):
            return topic
    return 'Other'

# apply function to categorize each action
enforce_df_crim['Topic'] = enforce_df_crim['Title'].apply(categorize_topic)

chart = alt.Chart(enforce_df_crim).mark_line().encode(
    x=alt.X('year_month:N', title='Month-Year'),
    y=alt.Y('count():Q', title='Number of Enforcement Actions'),
    color=alt.Color('Topic:N', title='Criminal and Civil Actions Topic')
).properties(
    title='Number of Criminal and Civil Actions Over Time by Topic',
    width=400,
    height=100
)
chart
```



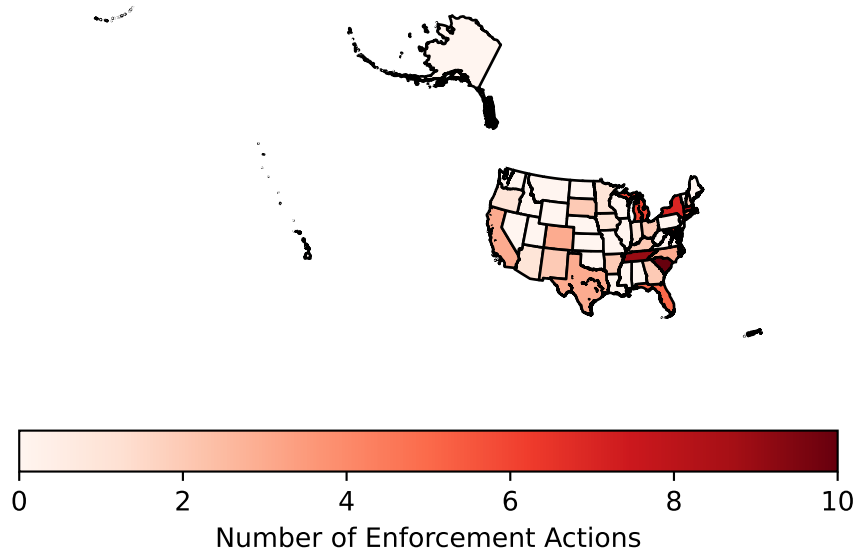
Step 4: Create maps of enforcement activity

1. Map by State (PARTNER 1)

```
import geopandas as gpd
import matplotlib.pyplot as plt
from shapely import wkt

# read census shapefile
filepath =
 ↪  '/Users/tarini_dewan/Desktop/UChicago/Python_2/problem-set-4-shreya-and-tarini/gz_2010_us
usa_df = gpd.read_file(filepath)

# read US District Attorney shapefine
usda_df =
 ↪  pd.read_csv('/Users/tarini_dewan/Desktop/UChicago/Python_2/US_Attorney_Districts_Shapefil

# Convert WKT strings to geometry objects
usda_df['geometry'] = usda_df['the_geom'].apply(wkt.loads)
usda_df = gpd.GeoDataFrame(usda_df, geometry='geometry', crs='EPSG:4326')

# Dissolve geometries by 'STATE' to get state-level geometries
usda_df_new = usda_df.dissolve(by='STATE')

# extract state name from agency
def get_state(agency):
    if isinstance(agency, str) and 'State of' in agency:
        return agency.replace('State of', '').strip()
    return np.nan

# apply function to create a new 'state' column in enforce_df
enforce_df['state'] = enforce_df['Agency'].apply(get_state)

# number of enforcement actions per state
state_df = pd.DataFrame(enforce_df['state'].value_counts())

# merge the state counts with District Attorney GeoDataFrame
map_state = pd.merge(state_df, usda_df_new, how = 'right', left_on = 'state',
 ↪  right_on = 'STATE')

# fill missing counts with zero
map_state['count'] = map_state['count'].fillna(0)

# Define the Albers equal-area projection for the United States
albers_crs = "+proj=aea +lat_1=20 +lat_2=60 +lat_0=40 +lon_0=-96 +x_0=0
 ↪  +y_0=0 +ellps=GRS80 +datum=NAD83 +units=m +no_defs"
```

```
# convert to a GeoDataFrame with the crs
map_state = gpd.GeoDataFrame(map_state, geometry='geometry', crs='EPSG:4326')

# Project to Albers
map_state_albers = map_state.to_crs(albers_crs)

# map showing the number of enforcement actions per state
plt.figure(figsize=(50, 30), dpi=1000)
ax = map_state_albers.plot(
    column='count',
    cmap='Reds',
    edgecolor='black',
    vmin=0,
    vmax=map_state_albers['count'].max(),
    legend=True,
    legend_kwds={'label': "Number of Enforcement Actions", 'orientation':
 ↪  "horizontal"}
)
ax.set_axis_off()
plt.title('Number of Enforcement Actions for Each State')
plt.show()
```

```
<Figure size 50000x30000 with 0 Axes>
```



2. Map by District (PARTNER 2)

```
import re

# extract district name from agency info
def get_district(agency):
    if isinstance(agency, str) and 'District' in agency:
        # Find everything from 'District' onwards using regex
        district_match = re.search(r'(?:Southern |Eastern |Western |Northern
 ↪  |Central )?District.*$', agency)
        if district_match:
            district = district_match.group(0)

            # remove special characters but keep spaces and alphanumeric
             ↪  chars
            district = re.sub(r'[^a-zA-Z0-9\s]', '', district)

            # remove extra whitespace and convert multiple spaces to single
             ↪  space
            district = ' '.join(district.split())
```

```
            return district if district else np.nan
    return np.nan
# Attribution: ChatGPT
# Query: I want to extract everything that comes after 'District,' regardless
 ↪  of what comes before it and removing special characters

# Create district column by applying the get_district function
enforce_df['district'] = enforce_df['Agency'].apply(get_district)

# Create a DataFrame with the count of enforcement actions each district
district_df = pd.DataFrame(enforce_df['district'].value_counts())

# merge district counts with the USDA DataFrame
map_district = pd.merge(district_df, usda_df, how = 'right', left_on =
 ↪  'district', right_on = 'Judicial District ')

# Fill missing counts with zero
map_district['count'] = map_district['count'].fillna(0)

# Convert to a GeoDataFrame with crs
map_district = gpd.GeoDataFrame(map_district, geometry='geometry',
 ↪  crs='EPSG:4326')

# Project to Albers
map_district_albers = map_district.to_crs(albers_crs)

# map showing number of enforcement actions for each district
plt.figure(figsize=(50, 30), dpi=1000)
ax = map_district_albers.plot(
    column='count',
    cmap='Reds',
    edgecolor='black',
    vmin=0,
    vmax=map_state_albers['count'].max(),
    legend=True,
    legend_kwds={'label': "Number of Enforcement Actions", 'orientation':
 ↪  "horizontal"}
)
ax.set_axis_off()
plt.title('Number of Enforcement Actions for Each District')
plt.show()
```

```
<Figure size 50000x30000 with 0 Axes>
```