Final Portfolio

INFO 526: Data Visualization & Analysis

Antonios J. Bokas
University of Arizona, School of Information

May 4, 2024

I. Reflection

Throughout this course, I excelled at curating data and learned how to reduce data complexity for explanations and visualizations. I particularly gained new skills related to the following Learning Criteria:

- 2 (making different types of plots)
- 3 (making professional Markdown reports)
- 7 (reducing visual load)
- 8 (pivoting data)
- 9 (creating summary statistics)
- 10-13 (evaluating plots for professional and design flaws)
- 14 (writing effective captions)

Before this course, I had never made any plots besides bar and line plots in R. I wanted to learn analysis and visualization in Python, as it is my main tool at work. I think I achieved my goal in spades. My new ability to create Markdown/PDF reports already helped me at work over the past two months. Peers and superiors alike applauded the professionalism of the technical reports I created with this technique. Likewise, the critiques of my peers and professor improved my judgement when selecting visuals, summarizing data, and explaining my outputs. I used dataframe pivoting and aggregation throughout the course to create honest, effective, and unique assessments.

My assessments answered simple and complex questions. The research question often dictated the type of approach and visual I employed. I developed skills in formatting table and plot aesthetics, which made my outputs palatable and clear. Finally, I made substantial contributions to classmates to help them succeed in data evaluation, transformation, and visualization.

The following visualizations include a table and line, bar, and scatter plots. The final plot includes a geographical component, which is relevant to my profession in geospatial intelligence. The visuals are from the assignments, but I took particular care to make as many recommended improvements as possible. Some visuals that I hope to explore more after this course include polar graphs, heatmaps, violin plots, and flowcharts.

II. Visualizations

Year	President	Employed (millions)	% Change	SD (thousands)
2001 - 2004	George W. Bush	2.75	NaN	10.59
2005 - 2008	George W. Bush	2.74	-0.36	16.49
2009 - 2012	Barrack Obama	2.87	4.74	74.38
2013 - 2016	Barrack Obama	2.76	-3.83	26.57
2017 - 2020	Donald Trump	2.84	2.90	61.60
2021	Joseph Biden	2.88	1.41	NaN

Table 1. Mean U.S. federal employment, percentage change, and standard deviation in all industries by presidential term. Research question: *How has the number of federal employees changed over time?*

Table notes:

- One factor of federal employment is discretionary funding, per the Center on Budget and Policy Priorities.
- Discretionary funding reached an all-time high at the start of President Obama's first term, when federal employment was very high. Congress then capped discretionary funding in the middle of his first term, when federal employment drastically dropped. This behavior is evidenced by the high standard deviation for his first term.
- Discretionary funding increased again into President Biden's first term, per the Congressional Budget Office. This coincided with a return to federal employment levels of President Obama's first term.

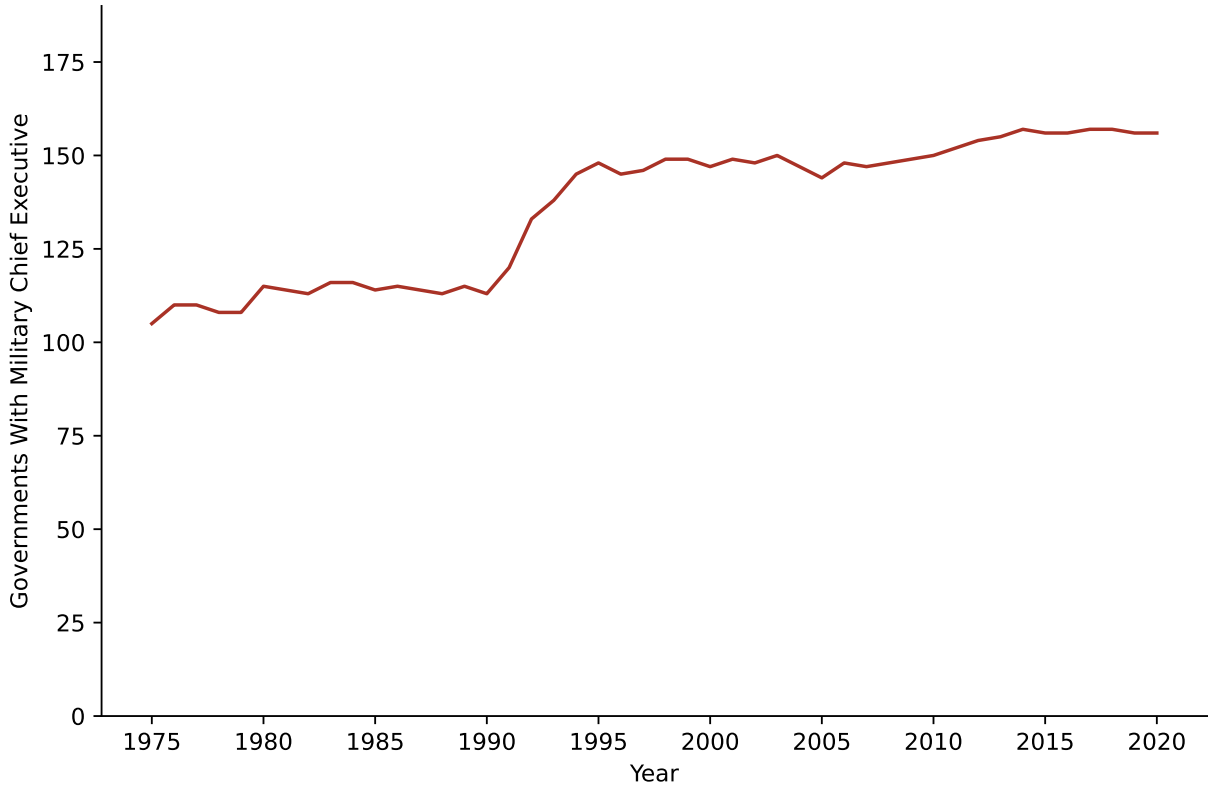

Figure 1. Annual total number of governments with a military officer as chief executive. Research question: *How has the number of political governments led by military members changed over time?*

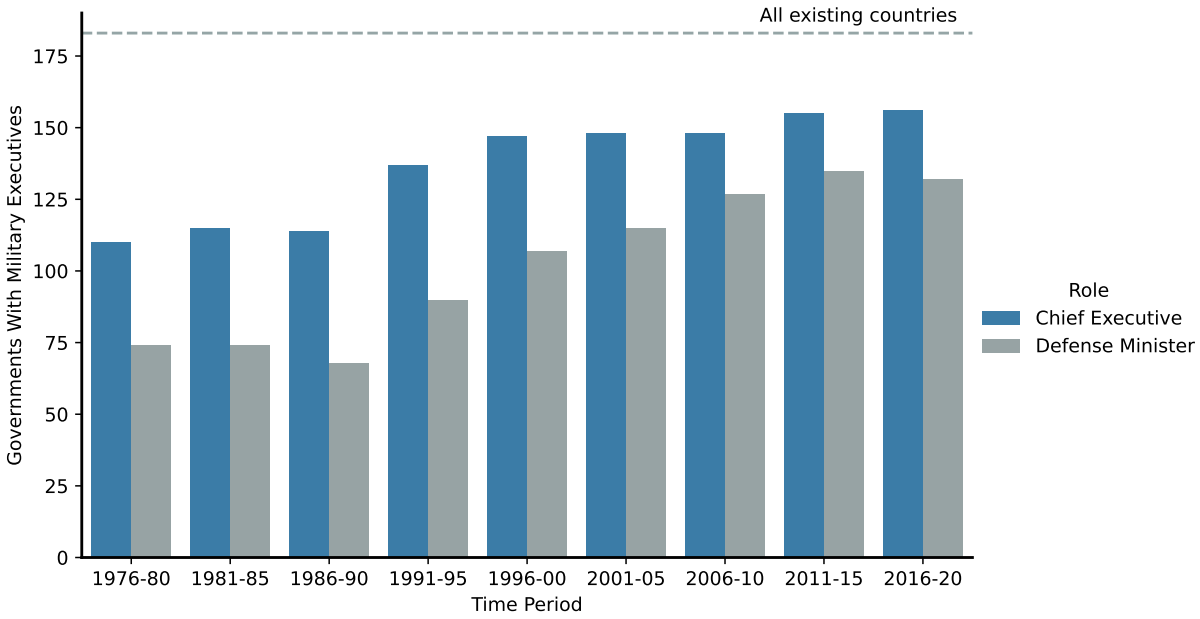

Figure 2. Annual total number of governments with a military officer as chief executive or defense minister. The dotted line is the total number of countries in the dataset and thus the theoretical maximum y-value. Research question: *How has the number of political governments led by military members changed over time?*

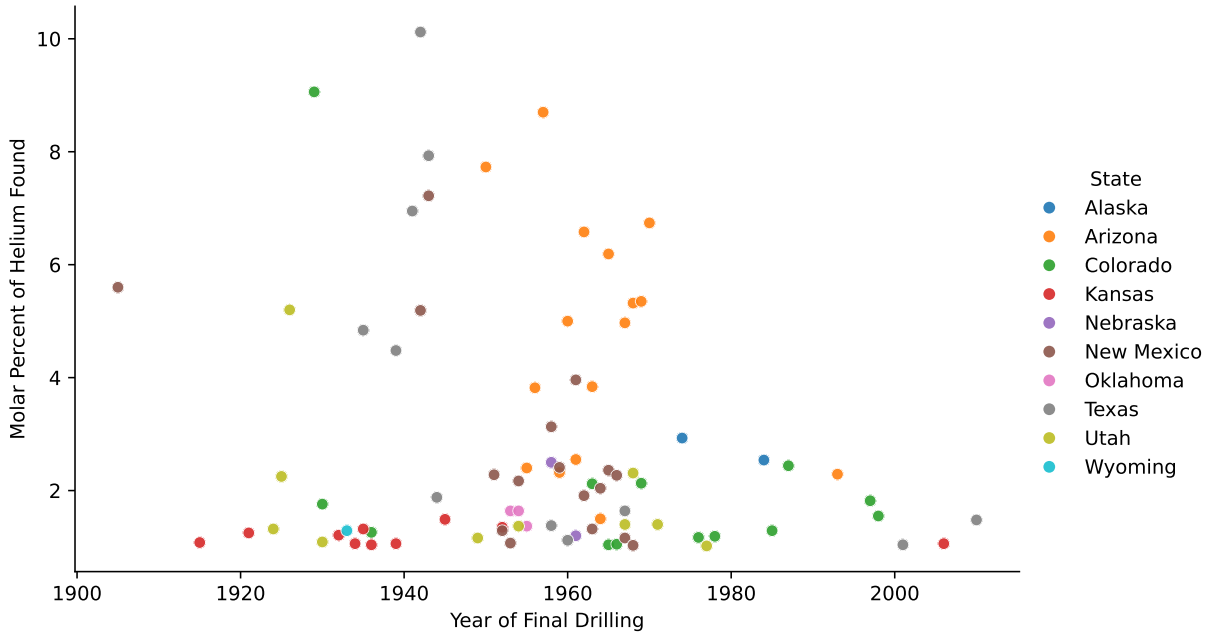

Figure 3. Mean molar percent of helium found since 1900 per year of last drilling and U.S. state, where molar percent was greater than 1. A *mole* is a unit of measure in chemistry that indicates the amount of a substance such as an element, compound, ion, or electron, per Oklahoma State University. Thus, helium molar percent is the ratio of helium to other substances found at the drill site. Research questions: *Is there a distinct change in some variable over time? How even is the representation of data in different categorical variables?*

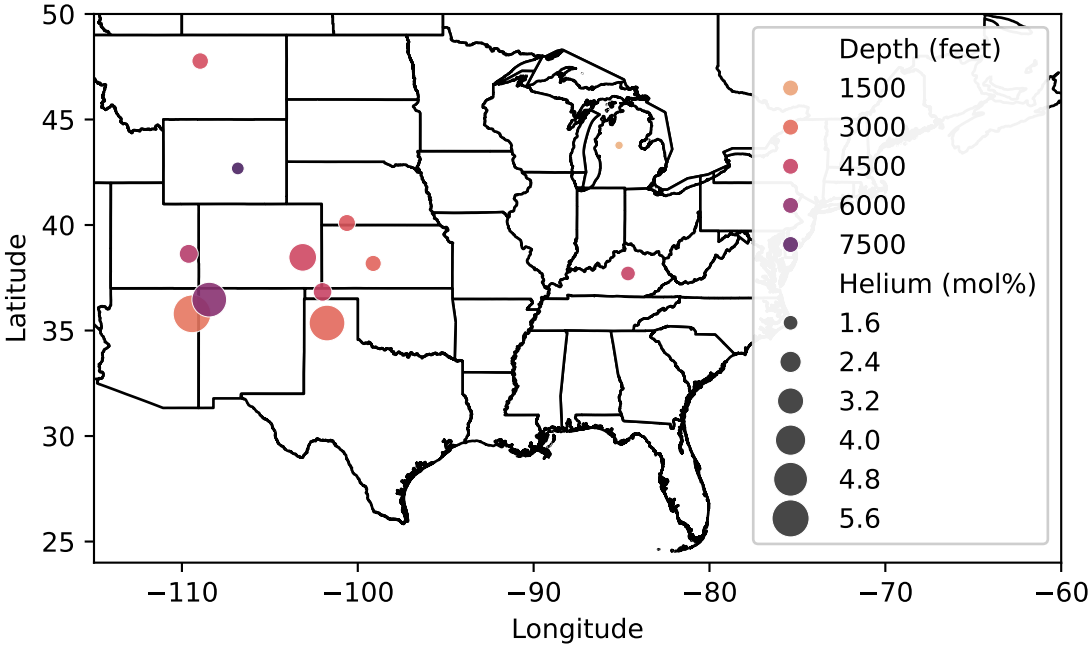

Figure 4. Mean molar percent of helium found, average drill depth, and average site latitude/longitude since 1900 per state, where molar percent was greater than 1. Research questions: *Are any variables correlated in a way that seems meaningful? Are there regional differences, or category differences in a variable?*

```
#!/usr/bin/env python

import warnings

from matplotlib import pyplot

import geopandas as geo

import numpy as np

import pandas as pd

import seaborn as sb

# TABLE 1


def set_presidents(row):
    if row.year < 2005:
        return 'George W. Bush'
    elif row.year < 2009:
        return 'George W. Bush'
    elif row.year < 2013:
        return 'Barrack Obama'
    elif row.year < 2017:
        return 'Barrack Obama'
    elif row.year < 2021:
        return 'Donald Trump'
    else:
        return 'Joseph Biden'


def bin_years(row):
    if row.year < 2005:
        return '2001 - 2004'
    elif row.year < 2009:
        return '2005 - 2008'
    elif row.year < 2013:
        return '2009 - 2012'
    elif row.year < 2017:
        return '2013 - 2016'
    elif row.year < 2021:
        return '2017 - 2020'
    else:
        return '2021'


# LOAD

bls = pd.read_csv('/Users/tonybokas/Documents/personal/info526/code/'
                  'BLS National Wages.csv')

# CLEAN AND TRANSFORM
```

```
bls.drop(columns=['area_title', 'area_fips'], inplace=True)
bls.drop_duplicates(inplace=True)

bls = (bls[['year', 'own_code', 'industry_code', 'annual_avg_emplvl']]
       .query('own_code == 1 and industry_code == "10"')
       .sort_values(['year', 'own_code']))

bls.drop(columns=['own_code', 'industry_code'], inplace=True)
bls['President'] = ''
bls.President = bls.apply(set_presidents, axis=1)
bls.year = bls.apply(bin_years, axis=1)

bls = (bls.groupby(['year', 'President'], as_index=False)
          ['annual_avg_emplvl']
          .agg(['mean', 'std']))

bls['mean'] = bls.apply(lambda row: round(row['mean']/1000000, 2), axis=1)
bls['std'] = round(bls['std']/1000, 2)
bls['% Change'] = round(bls['mean'].pct_change() * 100, 2)
bls = bls[['year', 'President', 'mean', '% Change', 'std']]

# VISUALIZE

new_names = {'year': 'Year',
             'mean': 'Employed (millions)',
             'std': 'SD (thousands)'}

bls = bls.rename(columns=new_names)

# Format before table output:
bls['Employed (millions)'] = bls.apply(
    lambda row: '{:,}'.format(row['Employed (millions)']), axis=1
    )

bls['% Change'] = bls.apply(lambda row: '{:.2f}'.format(row['% Change']),
                            axis=1)
bls.replace('nan', 'NaN', inplace=True)
```

```
# FIGURE 1

# Per the dataset definitions, MILITARY of 1 means the role
# is held by a military officer.
def count_mil_chief(row):
    return 1 if row.military == 0 else 0


# DEFMIN follows the same rules as MILITARY:
def count_def_mil(row):
    return 1 if row.defmin == 0 else 0

# LOAD

pol = pd.read_excel('/Users/tonybokas/Documents/personal/info526/code/'
```

```
                    'Database of Political Institutions.xlsx')

# CLEAN AND TRANSFORM

mil = pol[['year', 'military']].copy(deep=True)
mil.military = mil.apply(count_mil_chief, axis=1)
mil.year = mil.year.dt.strftime('%Y').astype('int')
mil = mil.groupby(['year'], as_index=False).sum().sort_values('year')

# VISUALIZE

plots = sb.relplot(data=mil,
                   kind='line',
                   x='year',
                   y='military',
                   aspect=1.5,
                   color='#a93226')

for plot in plots.axes.flat:
    plot.set(xlabel='Year',
             ylabel='Governments With Military Chief Executive',
             xticks=[i for i in range(1975, 2025, 5)],
             ylim=[0, 190])

pyplot.tight_layout()
pyplot.show()
```

```
# FIGURE 2

# CLEAN AND TRANSFORM

countries = pol.countryname.nunique()

mil = pol[['year', 'military', 'defmin']].copy(deep=True)

mil.military = mil.apply(count_mil_chief, axis=1)
mil.defmin = mil.apply(count_def_mil, axis=1)
mil.year = mil.year.dt.strftime('%Y').astype('int')

mil = (mil.groupby('year', as_index=False)
          .sum()
          .sort_values('year')
          .query('year > 1975'))  # must cut off to make bins equal

year_groups = ['1976-80',
               '1981-85',
               '1986-90',
               '1991-95',
               '1996-00',
               '2001-05',
               '2006-10',
               '2011-15',
               '2016-20']
```

```
mil['year_bins'] = pd.qcut(mil.year, q=9, labels=year_groups)
mil.drop(columns=['year'], inplace=True)

mil.rename(columns={'military': 'Chief Executive',
                    'defmin': 'Defense Minister'},
           inplace=True)

mil = (mil.groupby('year_bins', as_index=False, observed=True)
          .agg('mean')
          .round()
          .melt(id_vars='year_bins',
                value_vars=['Chief Executive', 'Defense Minister'],
                var_name='Role',
                value_name='mem_count'))

# VISUALIZE

plots = sb.catplot(data=mil,
                   kind='bar',
                   x='year_bins',
                   y='mem_count',
                   hue='Role',
                   aspect=1.5,
                   palette=['#2980b9', '#95a5a6'])  # regal colors

for plot in plots.axes.flat:
    plot.set(xlabel='Time Period',
             ylabel='Governments With Military Executives',
             ylim=[0, 190])
    plot.axhline(183, ls='--', color='#95a5a6')
    plot.text(6.35, 187, 'All existing countries')
    plot.spines[['left', 'bottom']].set_linewidth(1.5)

plots.fig.subplots_adjust(top=0.95, bottom=0.15)
pyplot.show()
```

```
# FIGURE 3

def format_date(series: pd.Series) -> pd.Series:
    series = series.astype('str').str.strip()

    exp = {r' \d{2}:\d{2}:\d{2}': '',             # remove time
           r'\.0$': '',                           # remove decimals
           r'\d{1,2}/\d{1,2}/(\d{4}).*': r'\1',   # find year
           r'.*([1-2][0-9][0-9][0-9]).*': r'\1',  # find year
           r'0000|.*[A-Za-z]+.*': '0'}            # clean up missing

    series.replace(exp, inplace=True, regex=True)

    return series.astype('int')


def format_text(series: pd.Series) -> pd.Series:
    series = series.astype('str').str.strip()
```

```
    exp = {r'\.0$': '',      # remove decimals
           r'^0$': None,     # replace zeroes,
           r'^nan$': None,   # nan,
           np.nan: None}     # and NaN with None
    series.replace(exp, inplace=True, regex=True)

    return series


def format_floats(series: pd.Series) -> pd.Series:
    series = series.astype('str').str.strip()
    exp = {r'< ': '', None: np.nan}
    series.replace(exp, inplace=True, regex=True)

    return series.astype('float64')


# OUTPUT SETTINGS

warnings.filterwarnings('ignore')

# LOAD

gas = pd.read_excel('/Users/tonybokas/Documents/personal/info526/code/'
                    'Natural Gas Dataset.xlsx')

# CLEAN AND TRANSFORM

# Very hard to clean and not so interesting for now:
gas.drop(columns='PUBLICATION DATE', inplace=True)

gas.rename(
    columns={
        'SOURCE': 'Source',
        'ALASKA_QUAD_NAMES': 'Alaska Quad Names',
        'WELL NAME': 'Well Name',
        'LAT': 'Latitude',
        'LONG': 'Longitude',
        'FIELD': 'Field',
        'STATE': 'State',
        'COUNTY': 'County',
        'FORMATION': 'Formation',
        'AGE': 'Formation Age',
        'DEPTH': 'Depth (feet)',
        'FINAL_DRILLING_DATE': 'Final Drilling',
        'FINAL_SAMPLING_DATE': 'Final Sampling',
        'SAMPLE DATE BEFORE COMPLETION': 'Sample Before Completion',
        'HE': 'Helium',
        'CO2': 'Carbon Dioxide',
        'H2': 'Hydrogen',
        'N2': 'Nitrogen',
        'H2S': 'Hydrogen Sulfide',
        'AR': 'Argon',
```

```
        'O2': 'Oxygen',
        'C1': 'Methane',
        'C2': 'Ethane',
        'C3': 'Propane',
        'N-C4': 'Butane',
        'I-C4': 'Isobutane',
        'N-C5': 'Pentane',
        'I-C5': 'Isopentane',
        'C6+': 'Higher Hydrocarbons'
        },
    inplace=True
    )

# Make state names titlecase:
exp = r'([A-Z])([A-Z]+\s?)'
repl = lambda j: j.group(1) + j.group(2).lower()
gas['State'] = gas['State'].str.replace(exp, repl, regex=True)

# The American Petroleum Institute (API) value is categorical:
exp = {r'\.0$': '', r'^nan$': None, np.nan: None}
gas['API'] = gas['API'].astype('str').replace(exp, regex=True)

# This is actually a boolean attribute:
gas['Sample Before Completion'].fillna(value=False, inplace=True)
gas['Sample Before Completion'] = \
    gas['Sample Before Completion'].astype('bool')

# This dataset presents big date problems. Not only are some months and
# days missing, so set to "00", which is out-of-bounds for str-date
# conversion, date inputs are inconsistent. E.g., day/month/year
# and month/day/year. My best choice at this time is to just collect
# the year as an integer:
date_columns = ['Final Drilling', 'Final Sampling']

for column in date_columns:
    gas[column] = format_date(gas[column])

# Clean up NULL values in text columns:
text_columns = ('ID',
                'Source',
                'BLM ID',
                'USGS ID',
                'Alaska Quad Names',
                'API',
                'Well Name',
                'Field',
                'State',
                'County',
                'Formation')

for column in text_columns:
    gas[column] = format_text(gas[column])
```

```
# Make all element columns numeric:
element_columns = ('Helium',
                   'Carbon Dioxide',
                   'Hydrogen',
                   'Hydrogen Sulfide',
                   'Methane',
                   'Ethane',
                   'Propane',
                   'Butane',
                   'Isobutane',
                   'Pentane',
                   'Isopentane')

for column in element_columns:
    gas[column] = format_floats(gas[column])

helium = gas[['State', 'Helium', 'Final Drilling']].copy(deep=True)

helium = (helium.groupby(['State', 'Final Drilling'],
                         as_index=False,
                         observed=True)
                .agg('mean').round(2))

helium = helium[helium['Final Drilling'] > 1900][helium['Helium'] > 1]

# VISUALIZE

plots = sb.relplot(data=helium,
                   kind='scatter',
                   x='Final Drilling',
                   y='Helium',
                   aspect=1.5,
                   hue='State',
                   palette='tab10',
                   alpha=0.9)

for plot in plots.axes.flat:
    plot.set(xlabel='Year of Final Drilling',
             ylabel='Molar Percent of Helium Found')

plots.fig.subplots_adjust(top=0.95, bottom=0.15)
pyplot.show()
```

```
# FIGURE 4

# LOAD

us_canada = geo.read_file('/Users/tonybokas/Documents/personal/info526/code/'
                          'USA_Canada_ShapefileMerge.shp')

# CLEAN AND TRANSFORM

coordinates = geo.points_from_xy(gas['Longitude'], gas['Latitude'])
wells = geo.GeoDataFrame(gas, geometry=coordinates)
```

```
plots, plot = pyplot.subplots()

us_canada.plot(ax=plot, color='white', edgecolor='black', aspect=1.2)

# knitr outputs return unless I capture and assign to variable:
i = plot.set_xlim(-115, -60)
i = plot.set_ylim(24, 50)

helium = (gas[gas['Final Drilling'] > 1900]
             [gas['Helium'] > 1]
             [['State', 'Latitude', 'Longitude', 'Helium', 'Depth (feet)']]
          .copy(deep=True))

helium = (helium.groupby('State', as_index=False)
                .agg('mean')
                .rename(columns={'Helium': 'Helium (mol%)'}))

# VISUALIZE

plots = sb.scatterplot(data=helium,
                       x='Longitude',
                       y='Latitude',
                       size='Helium (mol%)',
                       sizes=(10, 200),
                       hue='Depth (feet)',
                       palette='flare',
                       alpha=0.9)

pyplot.legend(facecolor='white', framealpha=.95)
pyplot.show()
```