Data and Info Visualization - Choropleth Maps

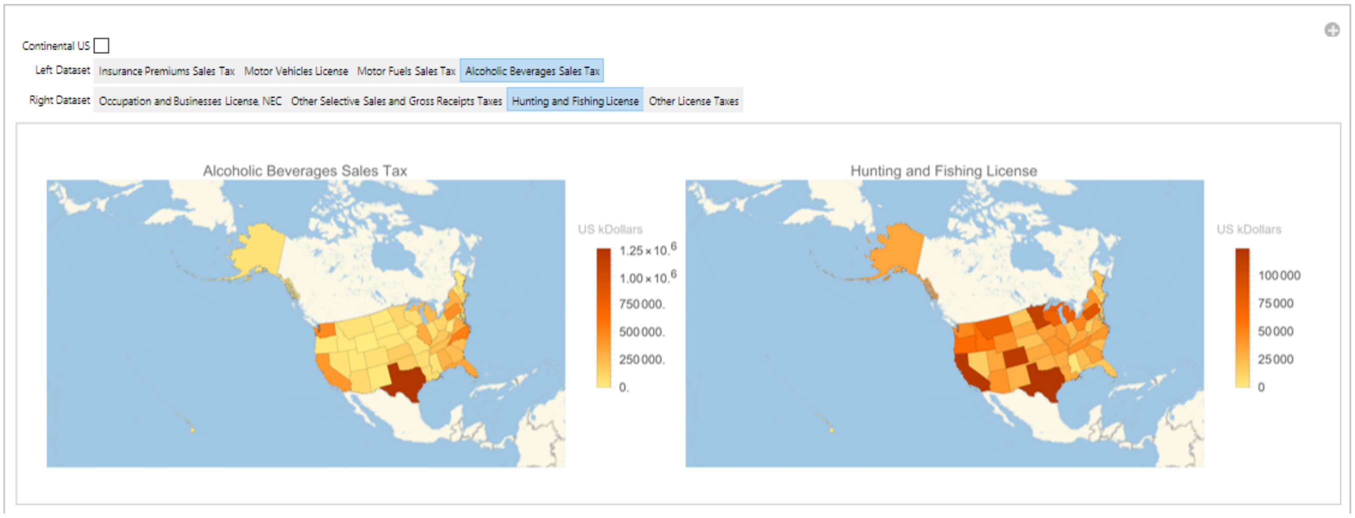
Andrew Struthers, Gihane Ndjeuha, Kollin Trujillo, Nathan Chapman, Nick Haviland

February 2023

1 Mathematica Choropleth Map

1.1 Data Selection

The choropleth visualization we made in Mathematica displayed eight different rows of the census dataset including: “Insurance Premiums Sales Tax”, “Motor Vehicle License”, “Motor Fuels Sales Tax”, “Alcoholic Beverage Sales Tax”, “Occupation and Business License, NEC”, “Other Selective Sales and Gross Receipts Taxes”, “Hunting and Fishing Licenses”, and “Other License Taxes”. These eight rows were selected out of the many available because each one of these rows had a complete dataset, i.e. an entry for each state. This selection would allow us to view each state relative to one another, without having to worry about handling missing data. Below, we can see a few different examples of the visualization of these data in Mathematica. Notice, there is an option to show the contiguous states only. This is useful if we need a more zoomed-in version of the visualization, at the cost of losing visual data about Hawaii and Alaska.

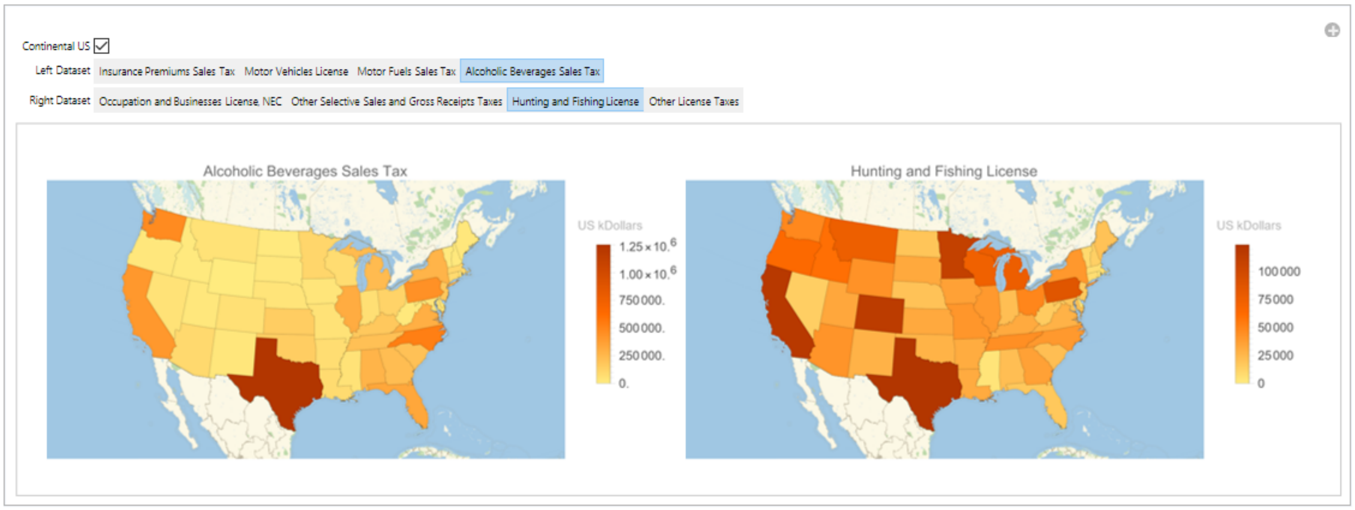


Continental US ☐

Left Dataset: Insurance Premiums Sales Tax | Motor Vehicles License | Motor Fuels Sales Tax | [✓ Alcoholic Beverages Sales Tax]

Right Dataset: Occupation and Businesses License, NEC | Other Selective Sales and Gross Receipts Taxes | [✓ Hunting and Fishing License] | Other License Taxes

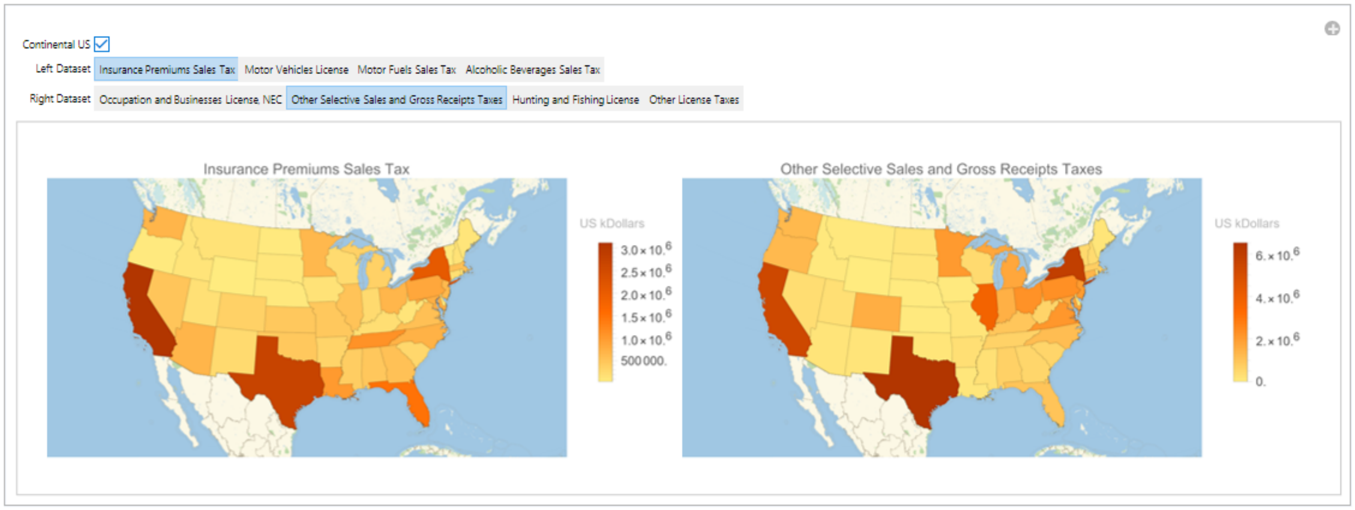
Figure 1: State map comparing Alcoholic Beverages Sales Tax to Hunting and Fishing Licenses


Continental US ✓

Left Dataset: Insurance Premiums Sales Tax | Motor Vehicles License | Motor Fuels Sales Tax | [✓ Alcoholic Beverages Sales Tax]

Right Dataset: Occupation and Businesses License, NEC | Other Selective Sales and Gross Receipts Taxes | [✓ Hunting and Fishing License] | Other License Taxes

Figure 2: Contiguous map of Alcoholic Beverages Sales Tax to Hunting and Fishing Licenses


Continental US ✓

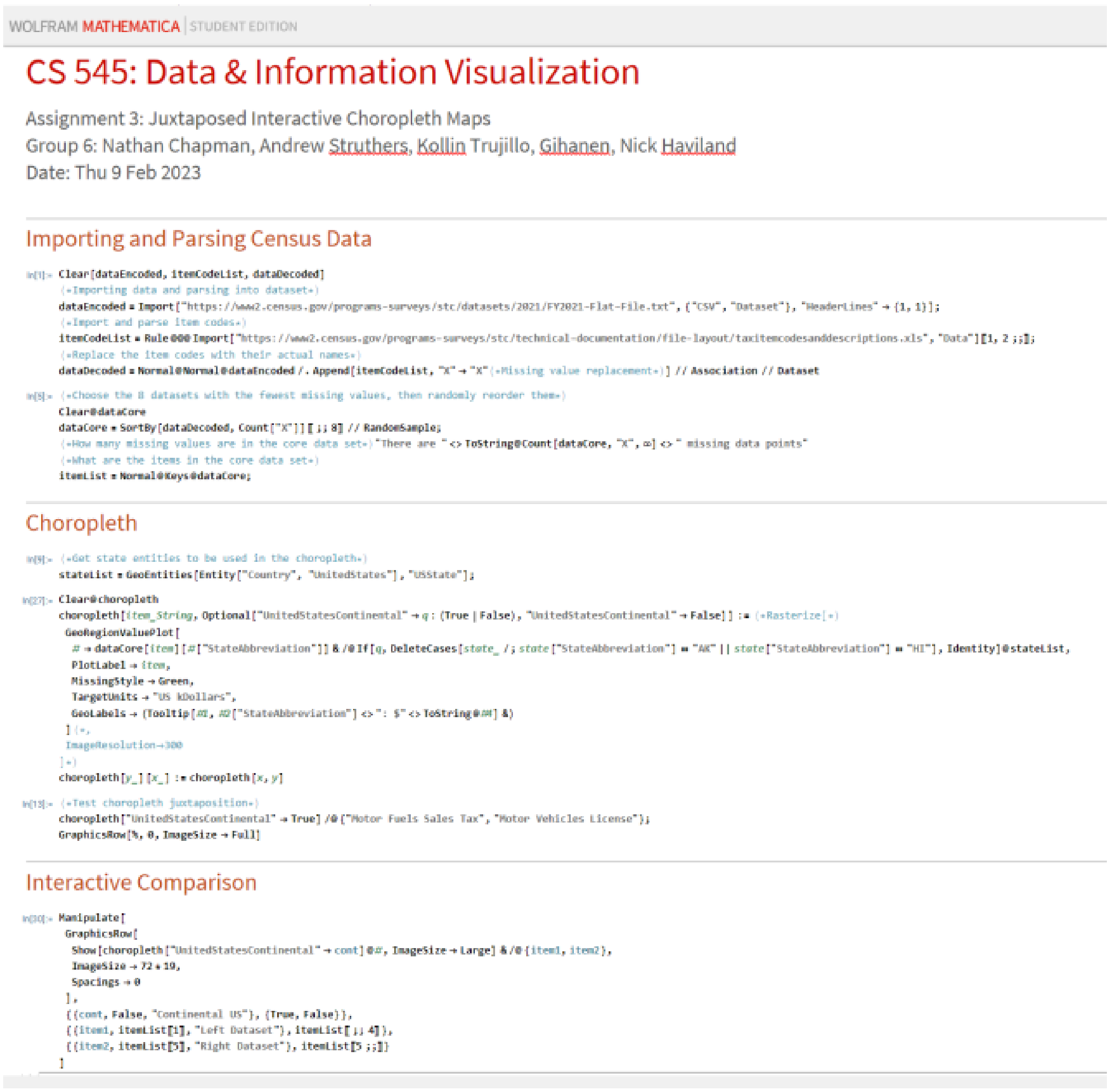
Left Dataset: [✓ Insurance Premiums Sales Tax] | Motor Vehicles License | Motor Fuels Sales Tax | Alcoholic Beverages Sales Tax

Right Dataset: Occupation and Businesses License, NEC | [✓ Other Selective Sales and Gross Receipts Taxes] | Hunting and Fishing License | Other License Taxes

Figure 3: Map of Insurance Premiums Sales Tax vs Other Selective Sales and Gross Receipts Taxes

1.2 Programming Language and Code

We attempted a choropleth visualization in Mathematica, since there was past expereince in making interactive UI and visualization components in Mathematica. As we can see from the figures above, it is entirely possible to read data and visualize it with an active UI display that users can interact with. This was a fun challenge to see if it was possible in a language that wouldn't necessarily be the "go-to" language for visualizing data. Below we can see a screenshot of the Mathematica notebook responsible for this visualization.


WOLFRAM MATHEMATICA | STUDENT EDITION

CS 545: Data & Information Visualization

Assignment 3: Juxtaposed Interactive Choropleth Maps
Group 6: Nathan Chapman, Andrew Struthers, Kollin Trujillo, Gihanen, Nick Haviland
Date: Thu 9 Feb 2023

Importing and Parsing Census Data

```
In[1]:= Clear[dataEncoded, itemCodeList, dataDecoded]
        (*Importing data and parsing into dataset*)
        dataEncoded = Import["https://www2.census.gov/programs-surveys/stc/datasets/2021/FY2021-Flat-File.txt", {"CSV", "Dataset"}, "HeaderLines" → {1, 1}];
        (*Import and parse item codes*)
        itemCodeList = Rule @@@ Import["https://www2.census.gov/programs-surveys/stc/technical-documentation/file-layout/taxitemcodesanddescriptions.xls", "Data"][[1, 2 ;;]];
        (*Replace the item codes with their actual names*)
        dataDecoded = Normal@Normal@dataEncoded /. Append[itemCodeList, "X" → "X"(*Missing value replacement*)] // Association // Dataset

In[5]:= (*Choose the 8 datasets with the fewest missing values, then randomly reorder them*)
        Clear@dataCore
        dataCore = SortBy[dataDecoded, Count["X"]][[ ;; 8]] // RandomSample;
        (*How many missing values are in the core data set*)"There are " <> ToString@Count[dataCore, "X", ∞] <> " missing data points"
        (*What are the items in the core data set*)
        itemList = Normal@Keys@dataCore;
```

Choropleth

```
In[9]:= (*Get state entities to be used in the choropleth*)
        stateList = GeoEntities[Entity["Country", "UnitedStates"], "USState"];

In[27]:= Clear@choropleth
        choropleth[item_String, Optional["UnitedStatesContinental" → q : (True | False), "UnitedStatesContinental" → False]] := (*Rasterize[*)
         GeoRegionValuePlot[
          # → dataCore[item][#["StateAbbreviation"]] & /@ If[q, DeleteCases[state_ /; state["StateAbbreviation"] == "AK" || state["StateAbbreviation"] == "HI"], Identity]@stateList,
          PlotLabel → item,
          MissingStyle → Green,
          TargetUnits → "US kDollars",
          GeoLabels → (Tooltip[#1, #2["StateAbbreviation"] <> ": $" <> ToString@#4] &)
         ](*,
         ImageResolution→300
        ]*)
        choropleth[y_][x_] := choropleth[x, y]

In[13]:= (*Test choropleth juxtaposition*)
        choropleth["UnitedStatesContinental" → True] /@ {"Motor Fuels Sales Tax", "Motor Vehicles License"};
        GraphicsRow[%, 0, ImageSize → Full]
```

Interactive Comparison

```
In[30]:= Manipulate[
         GraphicsRow[
          Show[choropleth["UnitedStatesContinental" → cont]@#, ImageSize → Large] & /@ {item1, item2},
          ImageSize → 72 * 19,
          Spacings → 0
         ],
         {{cont, False, "Continental US"}, {True, False}},
         {{item1, itemList[[1]], "Left Dataset"}, itemList[[ ;; 4]]},
         {{item2, itemList[[5]], "Right Dataset"}, itemList[[5 ;;]]}
        ]
```

Figure 4: Mathematica notebook for choropleth visualization

1.3 Interesting Findings

One finding we came up with was that, of the states with high insurance premiums, those states also had high gross tax. We noticed that these two aren't necessarily related, but came to the conclusion that it showed a general trend of "tax is high for this, so tax is high for that". For example, we can see that California has a high insurance premium sales tax, and they also have high selective sales and gross tax. Since insurance premiums and selective sales aren't related, the natural conclusion is that California just has high tax rates all around. This is backed up by many California residents complaining about their high taxes, and them moving to states with historically lower taxes. We also noticed that the states with the most alcoholic beverage tax and the most hunting/fishing licenses were pretty highly correlated, except for the states that are much more generally “Christian/Catholic”. There is also a large correlation between states with easy access to large bodies of water and the number of fishing licenses purchased. We can see in Figure 2, for example, that the West Coast, Texas, and states around the Great Lakes, all have a relatively large number of fishing licenses.

1.4 Hardships and Lessons Learned

One of the interesting problems was that, while the dataset was originally in a human readable format, parsing it into a nice computationally manipulable format was more work than we were expecting. This could be because we wanted to be stubborn and turn it into an actual Mathematica Dataset structure. Additionally, for the visualization part, it turned out to be a considerable challenge to customize how the actual graphic of each region was presented, e.g. we weren’t able to place the state abbreviation on top of the displayed region. If the program was left to annotate the regions automatically with their names, then it works just fine, but the whole state name is used instead of the abbreviation, making the whole thing hard to read due to overlapping text. This is further demonstrated in Figure 5 below. Finally, as was pointed out when discussing the results, if we include all the state data then we need to show both Alaska and Hawaii, which really throws off the scale if you want to see them in the correct geographic locations, since we are now visualizing a lot of ocean as well as Canada and Mexico. This resulted in a tradeoff between more data being shown versus being able to present most of the data more clearly, hence the option to only view the contiguous United States.

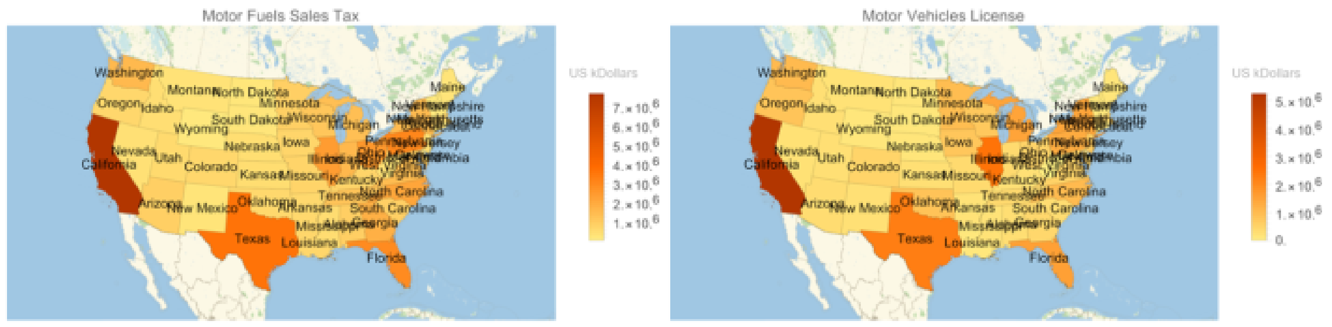


Figure 5: Example of Mathematica’s automatic annotation

2 Python Choropleth Map

2.1 Data Selection

We also decided to try and accomplish this visualization using Python and the Plotly. For the data selection, we used the transpose feature in Excel to transform rows into columns and vice versa. Then we got a column of states which is represented by item, and the other columns T01, T10 etc.. represent the colors. We again used eight rows of data with no missing values, so that we didn't have to make a decision that would impact the visualization provided. The rows we used were "Alcoholic Beverage Sales Tax", "Insurance Premiums Sales Tax", "Motor Fuels Sales Tax", "Tobacco Products Sales Tax", "Other Selective Sales and Gross Receipts Taxes", "Hunting and Fishing Licenses", "Motor Vehicle License", and "Occupation and Business License, NEC".

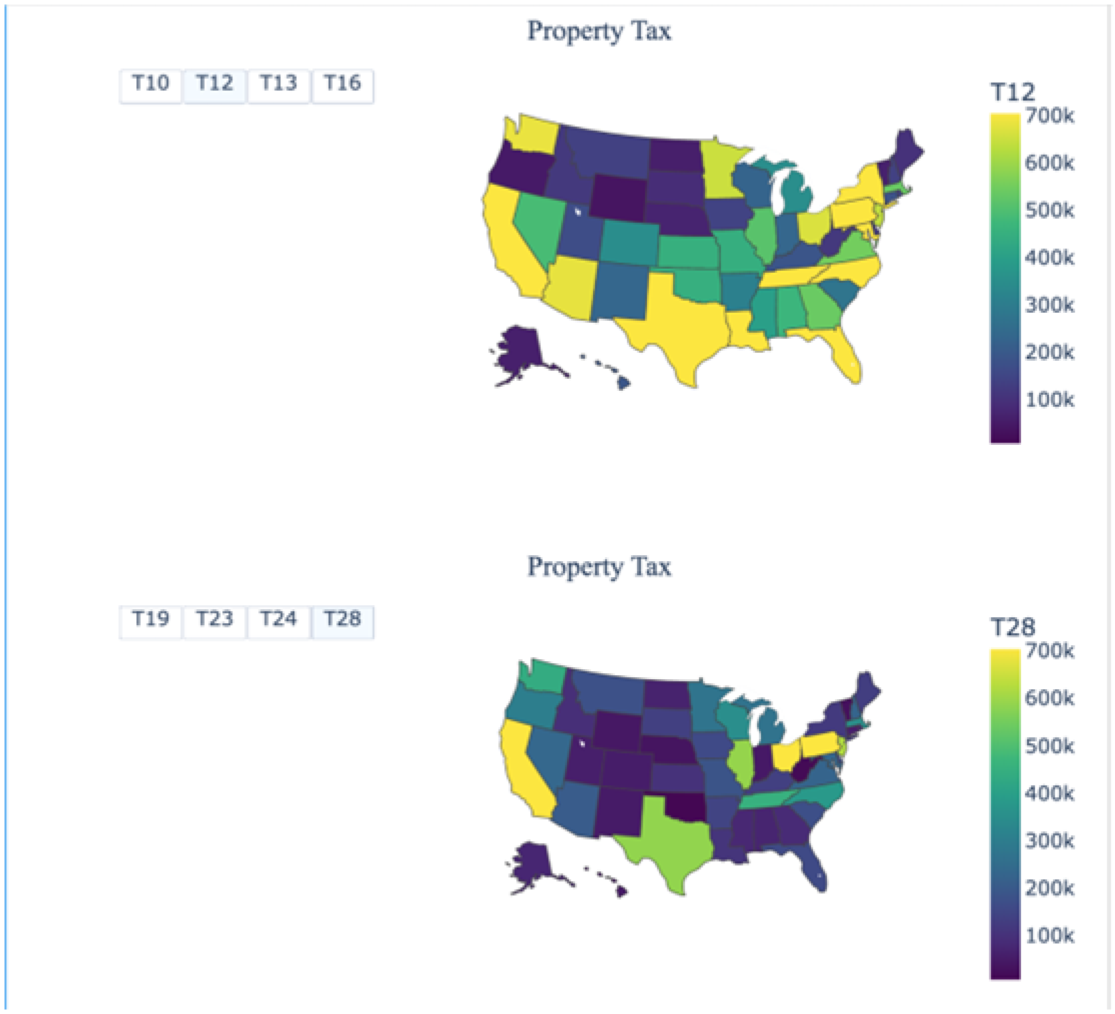
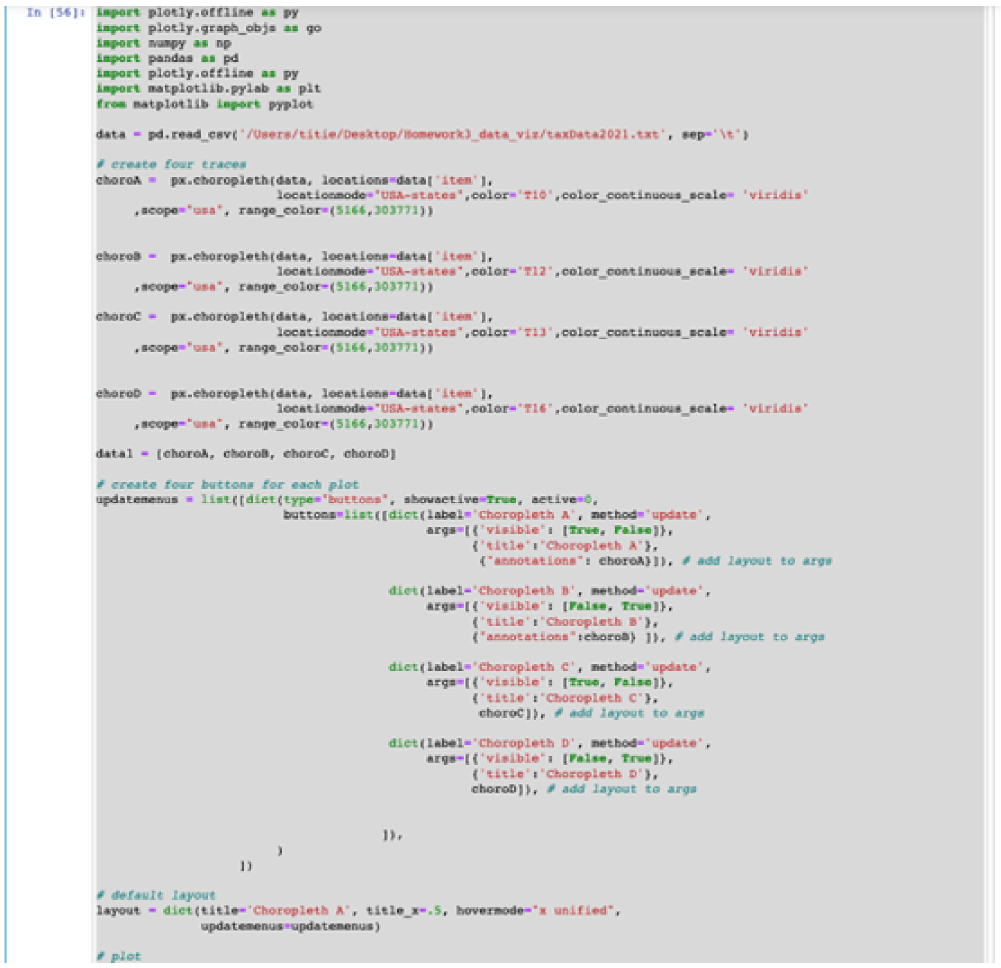


Figure 6: Visualizing Insurance Premiums Sales Tax against Occupation and Business Licenses

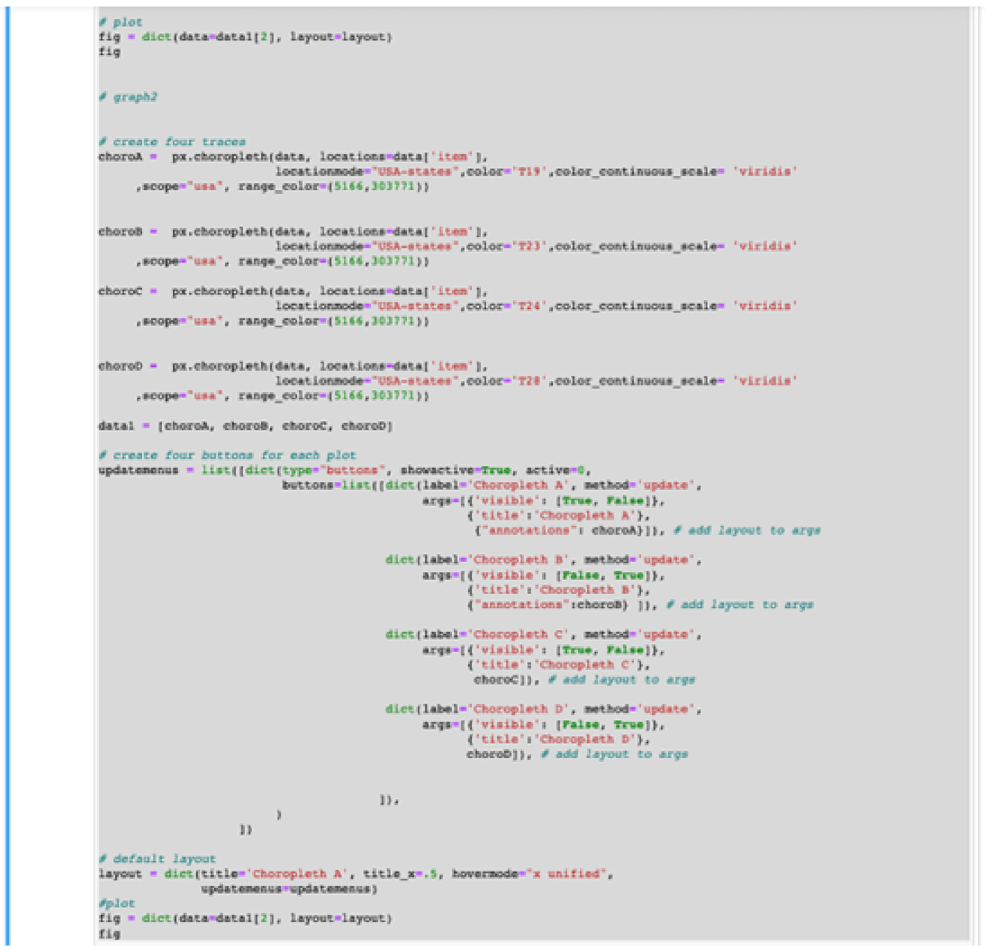
2.2 Programming Language and Code

We used Python to accomplish this visualizaiton, with the use of a few packages, namely Plotly, Numpy, Pandas, and Matplotlib. Plotly and Matplotlib made up the core of the choropleth visualization, and Numpy and Pandas were used to read and process the data from the Excel file.



```
In [56]: import plotly.offline as py
         import plotly.graph_objs as go
         import numpy as np
         import pandas as pd
         import plotly.offline as py
         import matplotlib.pylab as plt
         from matplotlib import pyplot

         data = pd.read_csv('/Users/titie/Desktop/Homework3_data_viz/taxData2021.txt', sep='\t')

         # create four traces
         choroA =  px.choropleth(data, locations=data['item'],
                                locationmode="USA-states",color='T10',color_continuous_scale= 'viridis'
              ,scope="usa", range_color=(5166,303771))


         choroB =  px.choropleth(data, locations=data['item'],
                                locationmode="USA-states",color='T12',color_continuous_scale= 'viridis'
              ,scope="usa", range_color=(5166,303771))

         choroC =  px.choropleth(data, locations=data['item'],
                                locationmode="USA-states",color='T13',color_continuous_scale= 'viridis'
              ,scope="usa", range_color=(5166,303771))


         choroD =  px.choropleth(data, locations=data['item'],
                                locationmode="USA-states",color='T16',color_continuous_scale= 'viridis'
              ,scope="usa", range_color=(5166,303771))

         data1 = [choroA, choroB, choroC, choroD]

         # create four buttons for each plot
         updatemenus = list([dict(type="buttons", showactive=True, active=0,
                                 buttons=list([dict(label='Choropleth A', method='update',
                                                    args=[{'visible': [True, False]},
                                                          {'title':'Choropleth A'},
                                                           {"annotations": choroA}]), # add layout to args

                                               dict(label='Choropleth B', method='update',
                                                    args=[{'visible': [False, True]},
                                                          {'title':'Choropleth B'},
                                                          {"annotations":choroB} ]), # add layout to args

                                               dict(label='Choropleth C', method='update',
                                                    args=[{'visible': [True, False]},
                                                          {'title':'Choropleth C'},
                                                           choroC]), # add layout to args

                                               dict(label='Choropleth D', method='update',
                                                    args=[{'visible': [False, True]},
                                                          {'title':'Choropleth D'},
                                                          choroD]), # add layout to args


                                              ]),
                                 )
                             ])

         # default layout
         layout = dict(title='Choropleth A', title_x=.5, hovermode="x unified",
                       updatemenus=updatemenus)

         # plot
```

Figure 7: Choropleth Python code part one



```
# plot
fig = dict(data=data1[2], layout=layout)
fig


# graph2


# create four traces
choroA =  px.choropleth(data, locations=data['item'],
                        locationmode="USA-states",color='T19',color_continuous_scale= 'viridis'
     ,scope="usa", range_color=(5166,303771))


choroB =  px.choropleth(data, locations=data['item'],
                        locationmode="USA-states",color='T23',color_continuous_scale= 'viridis'
     ,scope="usa", range_color=(5166,303771))

choroC =  px.choropleth(data, locations=data['item'],
                        locationmode="USA-states",color='T24',color_continuous_scale= 'viridis'
     ,scope="usa", range_color=(5166,303771))


choroD =  px.choropleth(data, locations=data['item'],
                        locationmode="USA-states",color='T28',color_continuous_scale= 'viridis'
     ,scope="usa", range_color=(5166,303771))

data1 = [choroA, choroB, choroC, choroD]

# create four buttons for each plot
updatemenus = list([dict(type="buttons", showactive=True, active=0,
                         buttons=list([dict(label='Choropleth A', method='update',
                                            args=[{'visible': [True, False]},
                                                  {'title':'Choropleth A'},
                                                   {"annotations": choroA}]), # add layout to args

                                       dict(label='Choropleth B', method='update',
                                            args=[{'visible': [False, True]},
                                                  {'title':'Choropleth B'},
                                                  {"annotations":choroB} ]), # add layout to args

                                       dict(label='Choropleth C', method='update',
                                            args=[{'visible': [True, False]},
                                                  {'title':'Choropleth C'},
                                                   choroC]), # add layout to args

                                       dict(label='Choropleth D', method='update',
                                            args=[{'visible': [False, True]},
                                                  {'title':'Choropleth D'},
                                                  choroD]), # add layout to args


                                      ]),
                         )
                    ])

# default layout
layout = dict(title='Choropleth A', title_x=.5, hovermode="x unified",
              updatemenus=updatemenus)
#plot
fig = dict(data=data1[2], layout=layout)
fig
```

Figure 8: Choropleth Python code part two

2.3 Interesting Findings

The visualization shows that the Alcoholic Beverages Sales Tax (T10) is low in most of the states and higher in only one state on the first graph and on the second graph the Other Selective Sales and Gross Receipts Taxes(T13) is higher in the majority of states. These findings seemed to align with the findings from the Mathematica visualizations, even though there were slightly different datasets being used.

2.4 Hardships and Lessons Learned

Learning how to connect the buttons in the figure to the appropriate chart, so that when an user clicks on T19 for example, the second chart updates, was a bit tricky to figure out. Using the well-defined Plotly libraries outside of their typical use case was a bit of a challenge, but one that was overcome through some careful integration with existing functionality.