How to create maps in Plotly with non-US locations

People in the US will never know our pain.

Kerry Halupka Follow
Jul 27 · 5 min read ★

Photo by Annie Spratt on Unsplash

Plotly makes it really easy to display geospatial data in an interactive choropleth map, particularly if your data pertains to the US. Unfortunately, if the location of your data is outside of the US, and more granular than country-level, things get a bit murky. You might feel like the poor cousin when sifting through the depths of Stack Overflow and GitHub Issues to find clues to achieving your mapping goals. I'm here to say I've been there, I feel your pain, and there is a solution. Let's jump in!

Step 1: Get some data to plot

For this tutorial, I'm going to display unemployment data from Victoria, Australia. I got my data from the Australian Bureau of Statistics Stat portal. This portal allows you to select the geospatial granularity of your data. I selected to download "Census 2016, G43 Labour force status by age by sex (LGA)", where LGA refers to Local Government Area, which is the spatial granularity of the data. Let's load it in and reshape it a bit:

```
import geopandas as gpd
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable
import plotly.graph_objects as go

emp_df = pd.read_csv('./data/ABS_C16_G43_LGA_26072020234812892.csv') #read ir
emp_df = emp_df[['LGA_2016', 'Labour force status', 'Region', 'Value']] #sele
emp_df['LGA_2016'] = emp_df['LGA_2016'].astype('str') # we will join on this
emp_df = emp_df.pivot(index='LGA_2016', columns='Labour force status', values
emp_df['percent_unemployed'] = emp_df['Total Unemployed']/(emp_df['Total Unem
emp_df.head()
```

	LGA_2016	Total	Total Employed	Total Unemployed	percent_unemployed
0	20110	10384	5488	216	0.037868
1	20260	9798	4708	257	0.051762
2	20570	82219	44716	3415	0.070952
3	20660	99850	58915	3429	0.055001
4	20740	27459	12537	857	0.063984

The column “LGA_2016” is referring to the ID or code for each Local Government Area (LGA). By pivoting the data we’ve created one row, and therefore one unemployment rate, for each LGA. This is important since when we create our map we will need one value for each geospatial area we show.

Step 2: Get geometries corresponding to your data

The key to creating a Plotly choropleth with data outside of the US is to have a GeoJSON with geometries corresponding to your data. While sometimes you may already have a GeoJSON, it’s more likely you’ll have to create one, which is the case for me.

I’m using The Australian Bureau of Statistics again to access geometries of my data in ESRI Shapefile format, via this link (I downloaded “Local Government Areas ASGS Ed 2020 Digital Boundaries in ESRI Shapefile Format”). This format can be loaded in to Python using Geopandas, by pointing it at the ‘.shp’ file:

```
lga_gdf = gpd.read_file('./data/1270055003_lga_2020_aust_shp/LGA_2020_AUST.shp
lga_gdf = lga_gdf[lga_gdf['STE_NAME16']=='Victoria'] #Select the data for the
lga_gdf['LGA_CODE20'] = lga_gdf['LGA_CODE20'].astype('str') # we will join on
lga_gdf.head()
```

choropleth_example2.py hosted with ❤ by GitHub view raw

	LGA_CODE20	LGA_NAME20	STE_CODE16	STE_NAME16	AREASQKM20	geometry
131	20110	Alpine (S)	2	Victoria	4788.1568	POLYGON ((146.67057 -36.56828, 146.67056 -36.5...
132	20260	Ararat (RC)	2	Victoria	4211.1171	POLYGON ((143.18569 -37.18385, 143.18598 -37.1...
133	20570	Ballarat (C)	2	Victoria	739.0321	POLYGON ((143.85331 -37.68123, 143.85320 -37.6...
134	20660	Banyule (C)	2	Victoria	62.5402	POLYGON ((145.08875 -37.69136, 145.08925 -37.6...
135	20740	Bass Coast (S)	2	Victoria	865.8095	MULTIPOLYGON (((145.11016 -38.51961, 145.10991...

Now we can see that both the data frames have a column containing a code for each LGA, which is what we will be joining on. Each row also has a “geometry” value, which is the geospatial outline of the LGA.

Side note: For plotting of Australian data the ABS will be your best bet for acquiring a shape file. I’m not as well versed in data for other countries (please leave a comment if you know of websites providing geo data from your country), but if you’re stuck http://www.diva-gis.org /gdata contains shape files, free to download, for most countries.

Step 3: Merge the data and the geometries (Optional: display using Geopandas)

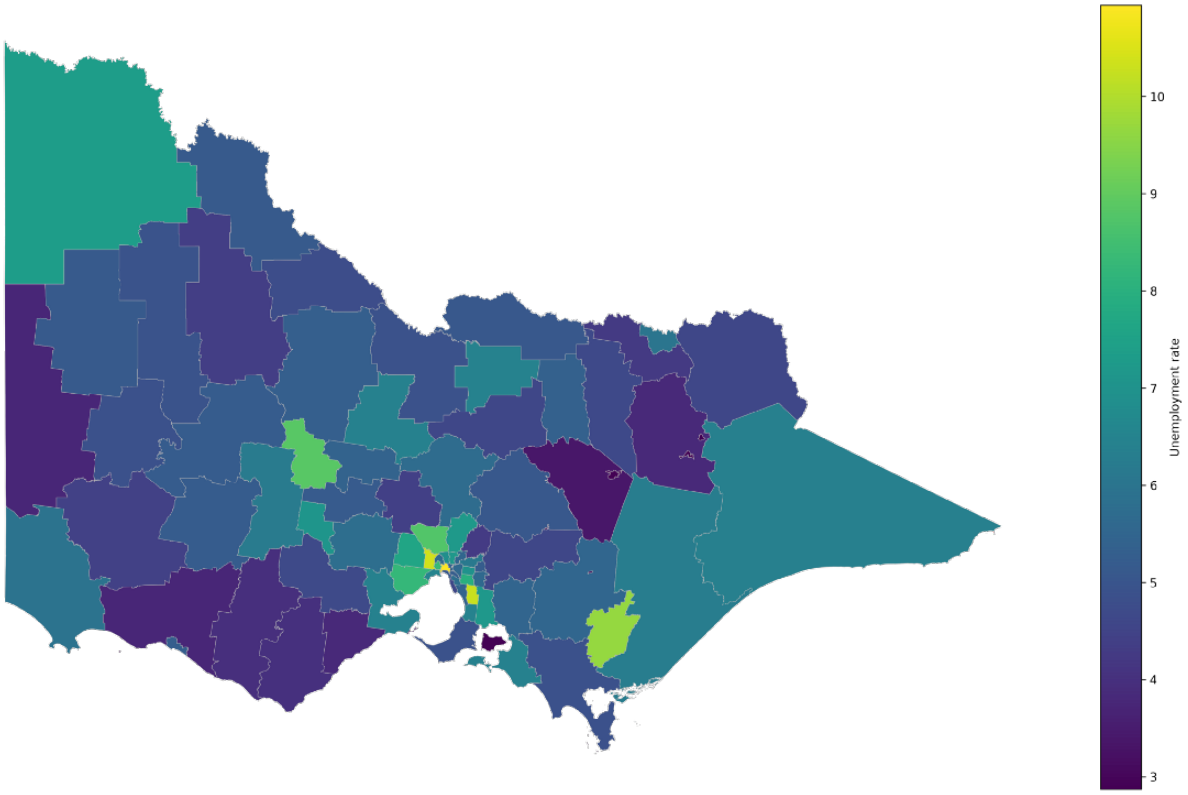
If both of your data frames have matching keys like mine, then it’s a simple matter to join them on that column, and drop rows with no data in the required columns.

```
df_merged = pd.merge(lga_gdf[['LGA_CODE20', 'geometry', 'LGA_NAME20']], emp_df
df_merged = df_merged.dropna(subset=['percent_unemployed', 'geometry']).set_ir
```

choropleth_example3.py hosted with ❤️ by **GitHub** **view raw**

I also like to display my data frame at this point using Geopandas as a sanity check, and to know what my plotly map should end up looking like.

```
# OPTIONAL: Display using geopandas
fig, ax = plt.subplots(1,1, figsize=(20,20))
divider = make_axes_locatable(ax)
tmp = df_merged.copy()
tmp['percent_unemployed'] = tmp['percent_unemployed']*100 #To display percent
cax = divider.append_axes("right", size="3%", pad=-1) #resize the colorbar
tmp.plot(column='percent_unemployed', ax=ax,cax=cax,  legend=True,
         legend_kwds={'label': "Unemployment rate"})
tmp.geometry.boundary.plot(color='#BABABA', ax=ax, linewidth=0.3) #Add some b
ax.axis('off')
```



Unemployment rate in Victoria, Australia, plotted using Geopandas.

NOTE: Merging the data frames isn't strictly necessary for choropleths in Plotly, but it does serve to align the geometries with the data to be plotted. If you don't do this merging step you will need to make sure that the data frame containing your data and the shapefile/geoJSON containing the geometries have corresponding and identically sorted rows.

Step 4: Convert the data frame to a GeoJSON

So now we've got a Geo Dataframe with all of the info in it that we need to map it using Plotly. Except for one little detail: Plotly wants it in GeoJSON format. Luckily, Geopandas can easily convert from data frame to GeoJSON, as below. First though, you will want to make sure your

geometries are in lat/long format (EPSG:4236), which is what Plotly is expecting.

```
df_merged = df_merged.to_crs(epsg=4326) # convert the coordinate reference sys
lga_json = df_merged.__geo_interface__ #covert to geoJSON
```

choropleth_example4.py hosted with ❤️ by GitHub view raw

Your GeoJSON should look something like this:

```
{'type': 'FeatureCollection',
 'features': [{'id': '20110',
   'type': 'Feature',
   'properties': {'LGA_2016': '20110',
    'LGA_CODE20': '20110',
    'LGA_NAME20': 'Alpine (S)',
    'percent_unemployed': 0.037868162692847124},
   'geometry': {'type': 'Polygon',
    'coordinates': (((146.67057224400003, -36.56828108499997),
      (146.670555271, -36.568038086999934),
      (146.67051924600003, -36.567766081999935),
      (146.67051924600003, -36.56750909899995),
      (146.6704312810001, -36.56723709299996),
      ...
```

You can see that this dictionary contains a nested dictionary called ‘features’, in which the index of each of your data frames rows is stored as ‘id’. If you didn’t create your geoJSON using this method then make sure yours has a similar format.

Step 5: Plot!

There’s a couple of things of note in the code below.

First up, you need a Mapbox access token. You can get this by creating a Mapbox account here. For personal use levels, accessing the API will be

free.

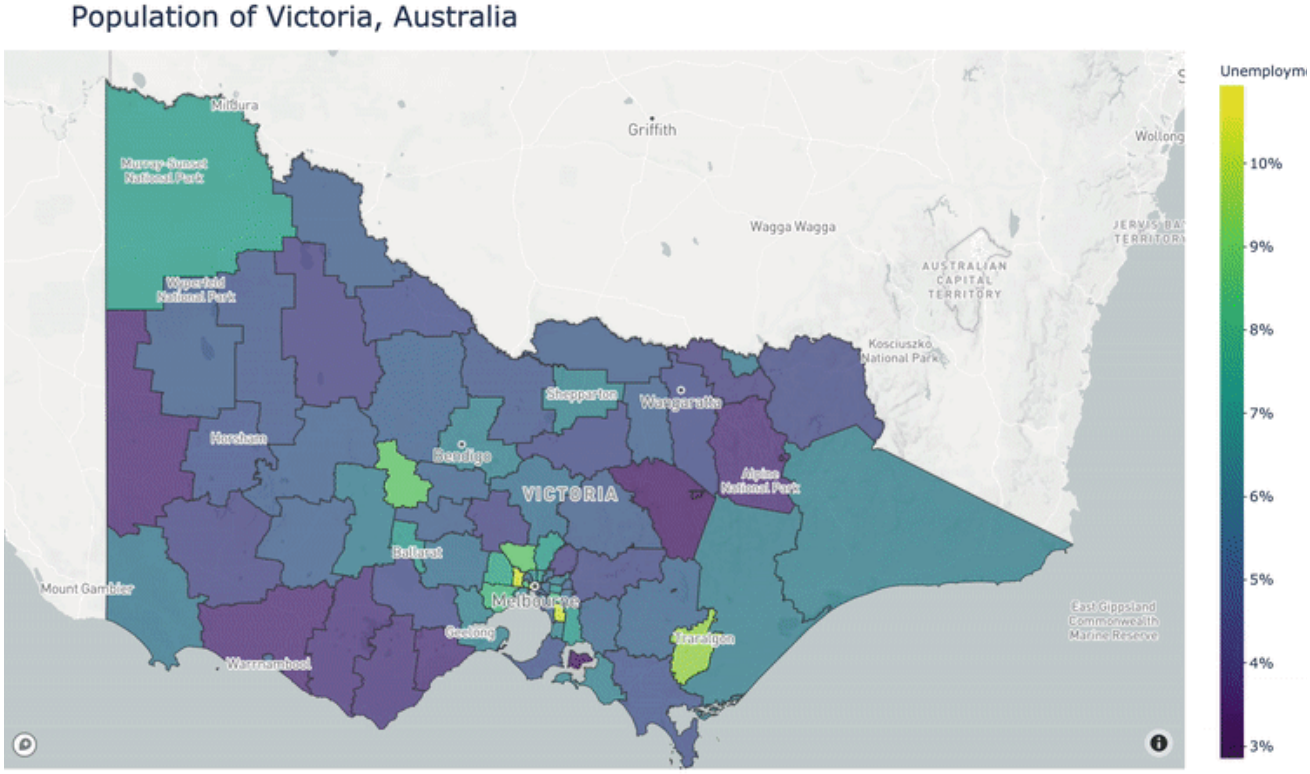
Secondly, we assign the data for the map, using the geoJSON for the geometries, and the merged data frame for the colors and text.

Finally, we set some parameters to define the layout of the map, including the title, centre lat/long, and zoom level.

```
MAPBOX_ACCESSTOKEN = 'your mapbox token here'

zmin = df_merged['percent_unemployed'].min()
zmax = df_merged['percent_unemployed'].max()

# Set the data for the map
data = go.Choroplethmapbox(
        geojson = lga_json,             #this is your GeoJSON
        locations = df_merged.index,    #the index of this dataframe should a
        z = df_merged.percent_unemployed, #sets the color value
        text = df_merged.LGA_NAME20,    #sets text for each shape
        colorbar=dict(thickness=20, ticklen=3, tickformat='%',outlinewidth=0)
        marker_line_width=1, marker_opacity=0.7, colorscale="Viridis", #adjus
        zmin=zmin, zmax=zmax,           #sets min and max of the colorbar
        hovertemplate = "<b>%{text}</b><br>" +
                    "%{z:.0%}<br>" +
                    "<extra></extra>")  # sets the format of the text shown w

# Set the layout for the map
layout = go.Layout(
    title = {'text': f"Population of Victoria, Australia",
            'font': {'size':24}},       #format the plot title
    mapbox1 = dict(
        domain = {'x': [0, 1],'y': [0, 1]},
        center = dict(lat=-36.5 , lon=145.5),
        accesstoken = MAPBOX_ACCESSTOKEN,
        zoom = 6),
    autosize=True,
    height=650,
    margin=dict(l=0, r=0, t=40, b=0))

# Generate the map
```

Data Science Data Visualization Programming Python Data

This will make a beautiful, interactive map, showing the unemployment rate ir
Victoria. Nice!



Plotly choropleth of Victoria, Australia, showing unemployment rate.

You can customise this tutorial using your own data and shapefiles. To see the
on GitHub click here.