Plotly vs. Matplotlib for choropleth maps.

Python notebook using data from Freight Analysis Framework · 5,624 views · data visualization, geospatial analysis

Plotly choropleth map

Plotly is a great library for interactive charts. It is also can be helpful with some choropleth maps. This small kernel was made to do demonstrate Plotly abilities and show some inconviniences.

In [1]:

```
import numpy as np
import pandas as pd
import plotly.graph_objs as go

from IPython.display import HTML
from plotly import __version__
from plotly.offline import download_plotlyjs, init_noteb
ook_mode, plot, iplot
init_notebook_mode(connected=True)
```

In [2]:

```
states_dict = {'01': "AL", '02': "AK", "04": "AZ", "05":
"AR",
              "06": "CA", "08": "CO", "09": "CT", "10":
"DE",
              "11": "DC", "12": "FL", "13": "GA", "15":
"HI",
              "16": "ID", "17": "IL", "18": "IN", "19":
"IA",
              "20": "KS", "21": "KY", "22": "LA", "23":
"ME",
              "24": "MD", "25": "MA", "26": "MI", "27":
"MN",
              "28": "MS", "29": "MO", "30": "MT", "31":
"NE",
              "32": "NV", "33": "NH", "34": "NJ", "35":
"NM",
              "36": "NY", "37": "NC", "38": "ND", "39":
"OH",
              "40": "OK", "41": "OR", "42": "PA", "44":
"RI",
              "45": "SC", "46": "SD", "47": "TN", "48":
"TX",
              "49": "UT", "50": "VT", "51": "VA", "53":
"WA",
              "54": "WV", "55": "WI", "56": "WY"}

years = ['2012', '2013', '2014','2015','2020',
         '2025', '2030', '2035', '2040', '2045']
```

```
2025 , 2030 , 2035 , 2040 , 2045 ]
```

In [3]:

```
states_df = pd.read_csv('../input/FAF4.4_State.csv',
                        dtype = {'dms_orig': str, 'dms_
dest': str})
```

In [4]:

```
states_df['dms_orig'].replace(states_dict, inplace = Tru
e)
states_df['dms_dest'].replace(states_dict, inplace = Tru
e)
```

In [5]:

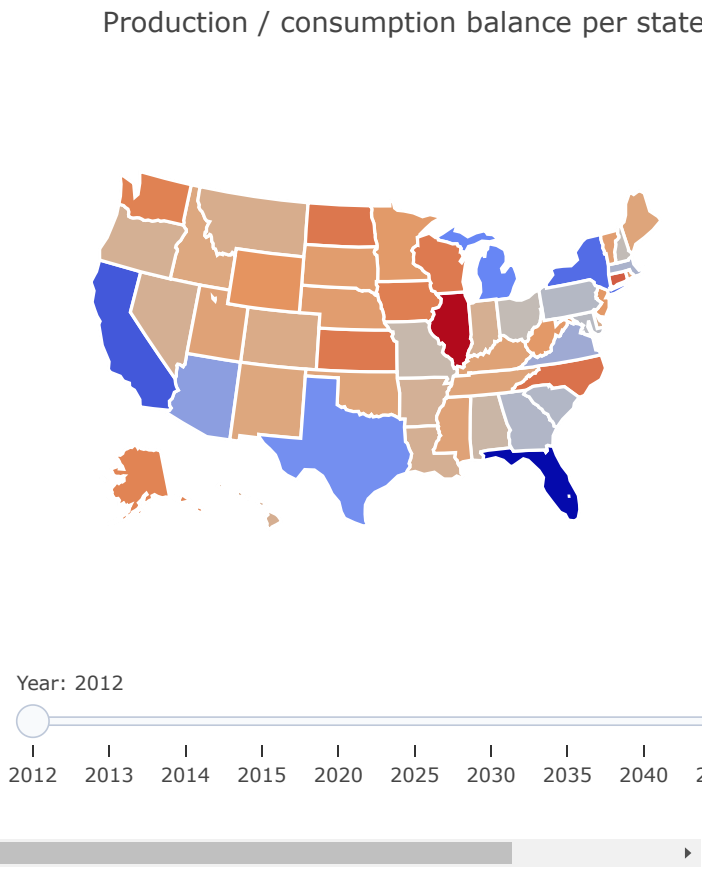
```
# domestic origin by value 2012-2045
dom_origin_df = states_df.loc[pd.isnull(states_df['fr_or
ig'])]

# domestic destination by value 2012-2045
dom_dest_df = states_df.loc[pd.isnull(states_df['fr_des
t'])]

# dataframe for freight balance (outflow from regoin minu
s inflow to region)
dom_origin_bal_df = dom_origin_df[['dms_orig','value_201
2',
                                   'value_2013', 'value_201
4',
                                   'value_2015', 'value_202
0',
                                   'value_2025', 'value_203
0',
                                   'value_2035', 'value_204
0',
                                   'value_2045']].groupby(
'dms_orig', as_index = True).sum()

dom_dest_bal_df = dom_dest_df[['dms_dest','value_2012',
                                   'value_2013', 'value_201
4',
                                   'value_2015', 'value_202
0',
                                   'value_2025', 'value_203
0',
                                   'value_2035', 'value_204
0',
                                   'value_2045']].groupby(
'dms_dest', as_index = True).sum()

dom_dest_bal_df = dom_dest_bal_df.apply(lambda x: x*(-1
))

balance_df = dom_origin_bal_df.add(dom_dest_bal_df, fill
_value = 0.0)
balance_df.reset_index(inplace = True)
```

```
balance_df.columns = ['state']+years
```

In [6]:

```
scl = [[0.0, 'rgb(84,39,143)'], [0.1, 'rgb(117,107,177)'
], [0.2, 'rgb(158,154,200)'],
       [0.3, 'rgb(188,189,220)'], [0.4, '218,218,235)'],
[0.5, 'rgb(240,240,240)'],
       [0.6, 'rgb(255,214,151)'],[0.8, 'rgb(250,195,10
4)'], [0.9, 'rgb(250,177,58)'],
       [1.0, 'rgb(252,153,6)']]

data_bal = []

data_2012 = [dict(type='choropleth',
                colorscale = scl,
                autocolorscale = False,
                locations = balance_df['state'],
                z = balance_df['2012'].astype(float)/100
0,
                locationmode = 'USA-states',
                text = balance_df['state'],
                marker = dict(line = dict(color = 'rgb(2
55,255,255)',
                                          width = 2)),
                visible = True,
                colorbar = dict(title = "Billions USD"
))]

data_bal.extend(data_2012)

for i in years[1:]:
    data_upd = [dict(type='choropleth',
                     colorscale = scl,
                     autocolorscale = False,
                     locations = balance_df['state'],
                     z = balance_df[i].astype(float)/10
00,
                     locationmode = 'USA-states',
                     text = balance_df['state'],
                     marker = dict(line = dict(color =
'rgb(255,255,255)',
                                               width =
2)),
                     visible = False,
                     colorbar = dict(title = "Billions
 USD"))]

    data_bal.extend(data_upd)


# set menues inside the plot
steps = []
yr = 0
for i in range(0,len(data_bal)):
    step = dict(method = "restyle",
                args = ["visible", [False]*len(data_bal
```

```
)],
                label = years[yr])
    step['args'][1][i] = True
    steps.append(step)
    yr += 1

sliders = [dict(active = 10,
                currentvalue = {"prefix": "Year: "},
                pad = {"t": 50},
                steps = steps)]

# Set the layout
layout = dict(title = 'Production / consumption balance
 per state',
              geo = dict(scope='usa',
                         projection=dict( type='albers u
sa' ),
                         showlakes = True,
                         lakecolor = 'rgb(255, 255, 25
5)'),
              sliders = sliders)

fig = dict(data=data_bal, layout=layout)
iplot( fig, filename='d3-cloropleth-map')
```



The matplotlib version of this map as well as detailed analysis of US Freight can be found here: www.kaggle.com/ievgenvp/us-freight-animated-novel

(3d map, under comment "# dataframe for freight balance (outflow from regoin minus inflow to region)")

Which library is better for making choropleth maps? Here are my thoughts.

Did you find this Kernel useful?
Show your appreciation with an upvote

10

Data

Data Sources

- Freight Analysis F...
  - F 42 columns
  - F 42 columns
  - FAF4 User Gui...
  - CFS_AREA_sha...

Freight Analysis Framework

Flows of goods among US regions for all modes of transportation

Last Updated: 2 years ago (Version 1)

About this Dataset

The Freight Analysis Framework (FAF) integrates data from a variety of sources to create a comprehensive picture of freight movement among states and major metropolitan areas by all modes of transportation. Starting with data from the 2012 Commodity Flow Survey (CFS) and international trade data from the Census Bureau, FAF incorporates data from agriculture, extraction, utility, construction, service, and other sectors. FAF version 4 (FAF4) provides estimates for tonnage (in thousand tons) and value (in million dollars) by regions of origin and destination, commodity type, and mode. Data are available for the base year of 2012, the recent years of 2013 - 2015, and forecasts from 2020 through 2045 in 5-year intervals.

Inspiration

This dataset should be great for map-based visualizations.

Run Info

Succeeded	True	Run Time	1686.6 seconds
Exit Code	0	Queue Time	0 seconds
Docker Image Name	kaggle/python (Dockerfile)	Output Size	0
Timeout Exceeded	False	Used All Space	False
Failure Message			

Log

Download Log

```
Time   Line #  Log Message
            1  [{
            2    "data": "[NbConvertApp] Converting notebook
               __temp_notebook_source__.ipynb to html\n",
            3    "stream_name": "stderr",
            4    "time": 2.4705342408269644
            5  },{
            6    "data": "[NbConvertApp] Writing 303968 bytes to
```

```
__results__.html\n",
  "stream_name": "stderr",
  "time": 2.639381928369403
}{
  "data": "[NbConvertApp] Converting notebook __temp_notebook_source__.ipynb to notebook\n",
  "stream_name": "stderr",
  "time": 2.227084616199136
},{
  "data": "[NbConvertApp] Executing notebook with kernel: python3\n",
  "stream_name": "stderr",
  "time": 2.2350845467299223
},{
  "data": "Fontconfig warning: ignoring C.UTF-8: not a valid language tag\n",
  "stream_name": "stderr",
  "time": 3.903501395136118
},{
  "data": "[NbConvertApp] Writing 110880 bytes to __notebook__.ipynb\n",
  "stream_name": "stderr",
  "time": 16.99631928279996
}{
  "data": "[NbConvertApp] Converting notebook __notebook__.ipynb to html\n",
  "stream_name": "stderr",
  "time": 2.230880768969655
},{
  "data": "[NbConvertApp] Writing 303950 bytes to __results__.html\n",
  "stream_name": "stderr",
  "time": 2.3987531289458275
}

Complete. Exited with code 0.
```


Comments (4)

Sort by

All Comments | Hotness

Click here to enter a comment...

Aleksey Bil... • Posted on Latest Version • 2 years ago • Options • Reply 1

A handful more options:

- cartopy: http://scitools.org.uk/cartopy/ (matplotlib wrapper)
- geoplot: http://www.residentmar.io/geoplot/index.html (cartopy wrapper)
- geopandas: This package has a plot built-in -- http://geopandas.org/mapping.html.

Not all of these work on Kaggle right now, but all can be useful.

One other package you should absolutely take a look at is https://github.com/jwass/mplleaflet. This nifty bit of code pumps a matplotlib viz onto a Leaflet animated display. I personally find it to be extremely useful, and it addresses some of the intent of the cons you list w.r.t matplotlib.

ievgen Kernel Author • Posted on Latest Version • 2 years ago • Options • Reply

Thanks for links, Aleksey. I will definitely try them for the following projects.

I made some mapping in R using Leaflet and it was very pleasant experience (lot's of tutorials and documentation in the web). Thus I'm looking forward to apply it in Python.

Frank Corri... • Posted on Latest Version • a year ago • Options • Reply 0

Thanks @ievgen. Where did you find the state list to match integer to state abbr? I feel like I just keep missing it in the docs. Thanks in advance.

ievgen Kernel Author • Posted on Latest Version • a year ago • Options • Reply

Hi, Frank. I took it from additional document named "Data Sources and Estimation Methodologies". Here is the link:

https://www.bts.dot.gov/sites/bts.dot.gov/files/legacy/FAF4%20Sources%20and%20Methodologies_v2.pdf

The crosswalk table for states and their relative codes is located on page 67 in Appendix A.

Similar Kernels