Folium Documentation

Release 0.2.0

Rob Story

July 20, 2016

Contents

CHAPTER 1

Python Data. Leaflet.js Maps.

Folium builds on the data wrangling strengths of the Python ecosystem and the mapping strengths of the Leaflet.js library. Manipulate your data in Python, then visualize it in on a Leaflet map via Folium.

1.1 Concepts

Folium makes it easy to visualize data that's been manipulated in Python on an interactive Leaflet map. It enables both the binding of data to a map for choropleth visualizations as well as passing Vincent/Vega visualizations as markers on the map.

The library has a number of built-in tilesets from OpenStreetMap, Mapbox, and Stamen, and supports custom tilesets with Mapbox or Cloudmade API keys. Folium supports both GeoJSON and TopoJSON overlays, as well as the binding of data to those overlays to create choropleth maps with color-brewer color schemes.

1.2 Contents

1.2.1 Quickstart

Getting Started

To create a base map, simply pass your starting coordinates to Folium:

```
import folium
map_osm = folium.Map(location=[45.5236, -122.6750])
```

To dispaly it in a Jupyter notebook, simply ask for the object representation:

```
map_osm
```

To save it in a file:

```
map_osm.save('/tmp/map.html')
```

Folium defaults to OpenStreetMap tiles, but Stamen Terrain, Stamen Toner, Mapbox Bright, and Mapbox Control room tiles are built in:

```
folium.Map(location=[45.5236, -122.6750],
           tiles='Stamen Toner',
           zoom_start=13)
```

Folium also supports Cloudmade and Mapbox custom tilesets- simply pass your key to the API_key keyword:

```
folium.Map(location=[45.5236, -122.6750],
           tiles='Mapbox',
           API_key='your.API.key')
```

Lastly, Folium supports passing any Leaflet.js compatible custom tileset:

```
folium.Map(location=[45.372, -121.6972],
           zoom_start=12,
           tiles='http://{s}.tiles.yourtiles.com/{z}/{x}/{y}.png',
           attr='My Data Attribution')
```

Markers

Folium supports the plotting of numerous marker types, starting with a simple Leaflet style location marker with popup text:

```
map_1 = folium.Map(location=[45.372, -121.6972],
                   zoom_start=12,
                   tiles='Stamen Terrain')
folium.Marker([45.3288, -121.6625], popup='Mt. Hood Meadows').add_to(map_1)
folium.Marker([45.3311, -121.7113], popup='Timberline Lodge').add_to(map_1)
map_1
```

Folium supports colors and marker icon types (from bootstrap)

```
map_1 = folium.Map(location=[45.372, -121.6972],
                   zoom_start=12,
                   tiles='Stamen Terrain')
folium.Marker([45.3288, -121.6625],
              popup='Mt. Hood Meadows',
              icon=folium.Icon(icon='cloud')
             ).add_to(map_1)
folium.Marker([45.3311, -121.7113],
              popup='Timberline Lodge',
              icon=folium.Icon(color='green')
             ).add_to(map_1)
folium.Marker([45.3300, -121.6823],
              popup='Some Other Location',
              icon=folium.Icon(color='red',icon='info-sign')
              ).add_to(map_1)
map_1
```

Folium also supports circle-style markers, with custom size and color:

```
map_2 = folium.Map(location=[45.5236, -122.6750],
                   tiles='Stamen Toner',
                   zoom_start=13)
folium.Marker([45.5244, -122.6699],
              popup='The Waterfront'
             ).add_to(map_2)
folium.CircleMarker([45.5215, -122.6261],
                    radius=500,
                    popup='Laurelhurst Park',
                    color='#3186cc',
                    fill_color='#3186cc',
                   ).add_to(map_2)
map_2
```

Folium has a convenience function to enable lat/lng popovers:

```
map_3 = folium.Map(
    location=[46.1991, -122.1889],
    tiles='Stamen Terrain',
    zoom_start=13)
map_3.add_child(folium.LatLngPopup())
map_3
```

Click-for-marker functionality will allow for on-the-fly placement of markers:

```
map_4 = folium.Map(location=[46.8527, -121.7649],
                   tiles='Stamen Terrain',
                   zoom_start=13)
folium.Marker([46.8354, -121.7325], popup='Camp Muir').add_to(map_4)
map_4.add_child(folium.ClickForMarker(popup="Waypoint"))
map_4
```

Folium also supports the Polygon marker set from the Leaflet-DVF:

```
map_5 = folium.Map(location=[45.5236, -122.6750],
                   zoom_start=13)

folium.RegularPolygonMarker(
    [45.5012, -122.6655],
    popup='Ross Island Bridge',
    fill_color='#132b5e',
    number_of_sides=3,
    radius=10
    ).add_to(map_5)
folium.RegularPolygonMarker(
    [45.5132, -122.6708],
    popup='Hawthorne Bridge',
    fill_color='#45647d',
    number_of_sides=4,
    radius=10
    ).add_to(map_5)
folium.RegularPolygonMarker(
    [45.5275, -122.6692],
    popup='Steel Bridge',
    fill_color='#769d96',
    number_of_sides=6,
    radius=10
    ).add_to(map_5)
folium.RegularPolygonMarker(
    [45.5318, -122.6745],
    popup='Broadway Bridge',
    fill_color='#769d96',
    number_of_sides=8,
    radius=10
    ).add_to(map_5)
map_5
```

Vincent/Vega Markers

Folium enables passing vincent visualizations to any marker type, with the visualization as the popover:

```
import json

buoy_map = folium.Map(
    [46.3014, -123.7390],
    zoom_start=7,
    tiles='Stamen Terrain'
    )

folium.RegularPolygonMarker(
    [47.3489, -124.708],
    fill_color='#43d9de',
    radius=12,
    popup=folium.Popup(max_width=450).add_child(
        folium.Vega(json.load(open('vis1.json')), width=450, height=250))
    ).add_to(buoy_map)

folium.RegularPolygonMarker(
    [44.639, -124.5339],
    fill_color='#43d9de',
    radius=12,
    popup=folium.Popup(max_width=450).add_child(
        folium.Vega(json.load(open('vis2.json')), width=450, height=250))
    ).add_to(buoy_map)

folium.RegularPolygonMarker(
    [46.216, -124.1280],
    fill_color='#43d9de',
    radius=12,
    popup=folium.Popup(max_width=450).add_child(
        folium.Vega(json.load(open('vis3.json')), width=450, height=250))
    ).add_to(buoy_map)

buoy_map
```

For more information about popups, please visit Popups.ipynb

GeoJSON/TopoJSON Overlays

Both GeoJSON and TopoJSON layers can be passed to the map as an overlay, and multiple layers can be visualized on the same map:

```
ice_map = folium.Map(location=[-59.1759, -11.6016],
                     tiles='Mapbox Bright', zoom_start=2)

folium.GeoJson(open('antarctic_ice_edge.json'),
               name='geojson'
              ).add_to(ice_map)

folium.TopoJson(open('antarctic_ice_shelf_topo.json'),
                'objects.antarctic_ice_shelf',
                name='topojson',
               ).add_to(ice_map)

folium.LayerControl().add_to(ice_map)
ice_map
```

Choropleth maps

Folium allows for the binding of data between Pandas DataFrames/Series and Geo/TopoJSON geometries. Color Brewer sequential color schemes are built-in to the library, and can be passed to quickly visualize different combinations:

```
import folium
import pandas as pd

state_geo = r'us-states.json'
state_unemployment = r'US_Unemployment_Oct2012.csv'

state_data = pd.read_csv(state_unemployment)

#Let Folium determine the scale
map = folium.Map(location=[48, -102], zoom_start=3)
map.geo_json(geo_path=state_geo, data=state_data,
             columns=['State', 'Unemployment'],
             key_on='feature.id',
             fill_color='YlGn', fill_opacity=0.7, line_opacity=0.2,
             legend_name='Unemployment Rate (%)')
map
```

```
/home/bibmartin/miniconda/envs/py35/lib/python3.5/site-packages/folium-0.2.0.dev0-py3.5.egg/folium/fo
  warnings.warn('This method is deprecated. '
/home/bibmartin/miniconda/envs/py35/lib/python3.5/site-packages/folium-0.2.0.dev0-py3.5.egg/folium/fo
  return self.choropleth(*args, **kwargs)
```

Folium creates the legend on the upper right based on a D3 threshold scale, and makes the best-guess at values via quantiles. Passing your own threshold values is simple:

```
map = folium.Map(location=[48, -102], zoom_start=3)
map.geo_json(geo_path=state_geo, data=state_data,
             columns=['State', 'Unemployment'],
             threshold_scale=[5, 6, 7, 8, 9, 10],
             key_on='feature.id',
             fill_color='BuPu', fill_opacity=0.7, line_opacity=0.5,
             legend_name='Unemployment Rate (%)',
             reset=True)
map
```

```
/home/bibmartin/miniconda/envs/py35/lib/python3.5/site-packages/folium-0.2.0.dev0-py3.5.egg/folium/fo
  warnings.warn('This method is deprecated. '
```

By binding data via the Pandas DataFrame, different datasets can be quickly visualized. In the following example, the df DataFrame contains six columns with different economic data, a few of which we will visualize:

```
import pandas as pd
unemployment = pd.read_csv('./US_Unemployment_Oct2012.csv')

m = folium.Map([43,-100], zoom_start=4)

m.choropleth(
    geo_str=open('us-states.json').read(),
    data=unemployment,
    columns=['State', 'Unemployment'],
    key_on='feature.id',
    fill_color='YlGn',
```

```
    )
m
```

```
/home/bibmartin/miniconda/envs/py35/lib/python3.5/site-packages/ipykernel/__main__.py:11: FutureWarni
```

For more choropleth example, please visit GeoJSON and choropleth.ipynb

1.2.2 Examples

You shall take a look at folium's example gallery.

If this is not enough, you may find fancier examples here.

1.2.3 Installing

Requirements

```
jinja2
```

Though Folium requires only *jinja2* to run, some functionalities may require *numpy* or *pandas* parameters.

Installation

Easiest

```
$pip install folium
```

Or from the source

$python setup.py install

From Source

Choose the sandbox folder of your choice (*~/sandbox* for example)

```
$ cd ~/sandbox
```

Clone *folium* from github:

```
$ git clone https://github.com/python-visualization/folium
```

Run the installation script

```
$ cd folium
$ python setup.py install
```

Run the tests

To run the tests, you'll also need to install:

```
flake8
pandas
pytest
vincent
```

Then go in folium base folder (*~/sandbox/folium* for example)

```
$ cd ~/sandbox/folium
```

Run the test

```
$ py.test
```

Build the docs

To build the docs, you'll also need to install:

```
sphinx
```

Then go in folium base folder (*~/sandbox/folium* for example)

```
$ cd ~/sandbox/folium
```

Build the docs

```
$ rm -rf docs/_build; sphinx-build -b html docs/ docs/_build/html
```

Then the documentation is in *docs/_build/html/index.html*.

1.2.4 Contributing

Choose the sandbox folder of your choice (*~/sandbox* for example)

```
$ cd ~/sandbox
```

Clone *folium* from github:

```
$ git clone https://github.com/python-visualization/folium
```

Push code to your fors, and write a pull request.

TODO: end up this section.

1.2.5 ModuleDescription

Element

Element

MacroElement

Figure

Html

Div

IFrame

Link

JavascriptLink

CssLink

Map

LegacyMap

Layer

TileLayer

FeatureGroup

LayerControl

Icon

Marker

Popup

FitBounds

Features

WmsTileLayer

RegularPolygonMarker

Vega

GeoJson

TopoJson