Filled Area on Tile Maps in Python

How to make an area on tile-based maps in Python with Plotly.

There are three different ways to show a filled area on a tile-based map:

- Using a Scattermap (https://plotly.com/python/reference/scattermap/) trace and setting the `fill` attribute to 'toself'
- Using a map layout (i.e. by minimally using an empty Scattermap (https://plotly.com/python/reference/scattermap/) trace) and adding a GeoJSON layer
- Using the Choroplethmap (https://plotly.com/python/tile-county-choropleth/) trace type

Filled Scattermap Trace

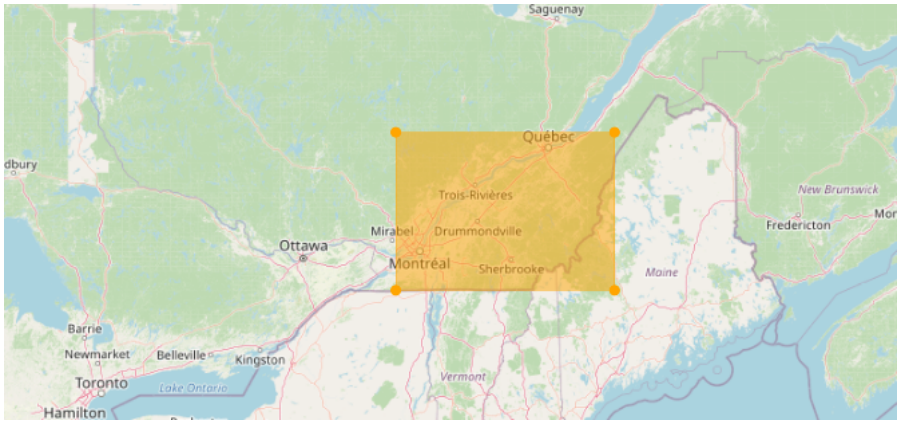
The following example uses `Scattermap` and sets `fill = 'toself'`

```
import plotly.graph_objects as go

fig = go.Figure(go.Scattermap(
    fill = "toself",
    lon = [-74, -70, -70, -74], lat = [47, 47, 45, 45],
    marker = { 'size': 10, 'color': "orange" }))

fig.update_layout(
    map = {
        'style': "open-street-map",
        'center': {'lon': -73, 'lat': 46 },
        'zoom': 5},
    showlegend = False)

fig.show()
```



© OpenStreetMap (https://www.openstreetmap.org/copyright) contributors

Multiple Filled Areas with a `Scattermap` trace

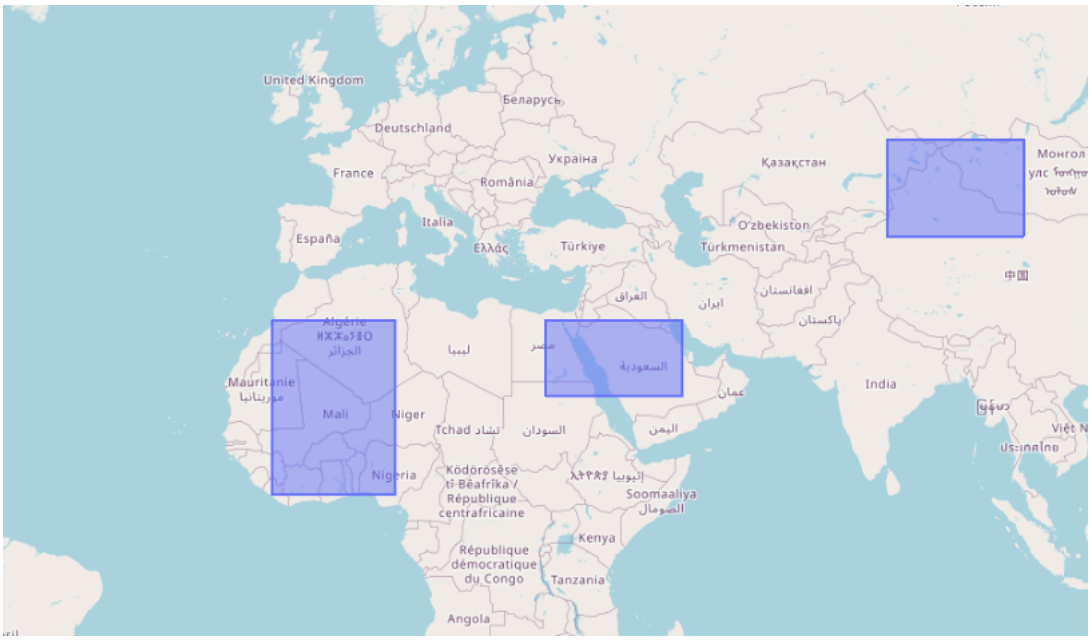
The following example shows how to use `None` in your data to draw multiple filled areas. Such gaps in trace data are unconnected by default, but this can be controlled via the connectgaps (https://plotly.com/python/reference/scattermap/#scattermap-connectgaps) attribute.

```
import plotly.graph_objects as go

fig = go.Figure(go.Scattermap(
    mode = "lines", fill = "toself",
    lon = [-10, -10, 8, 8, -10, None, 30, 30, 50, 50, 30, None, 100, 100, 80, 80, 100],
    lat = [30, 6, 6, 30, 30,    None, 20, 30, 30, 20, 20, None, 40, 50, 50, 40, 40]))

fig.update_layout(
    map = {'style': "open-street-map", 'center': {'lon': 30, 'lat': 30}, 'zoom': 2},
    showlegend = False,
    margin = {'l':0, 'r':0, 'b':0, 't':0})

fig.show()
```



GeoJSON Layers

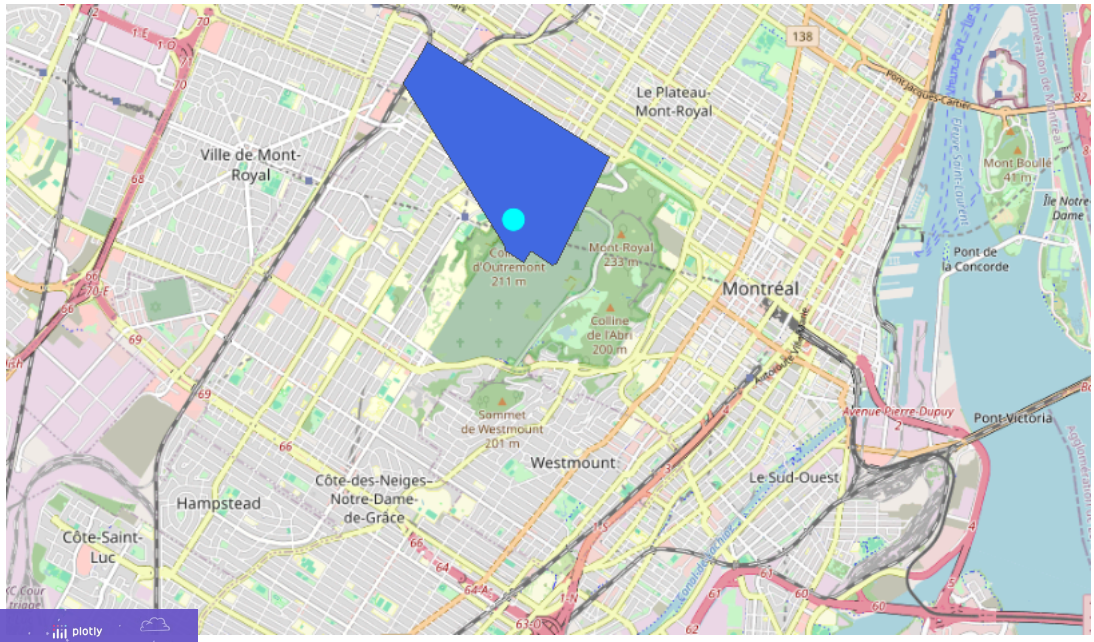
In this map we add a GeoJSON layer.

```
import plotly.graph_objects as go

fig = go.Figure(go.Scattermap(
    mode = "markers",
    lon = [-73.605], lat = [45.51],
    marker = {'size': 20, 'color': ["cyan"]}))

fig.update_layout(
    map = {
        'style': "open-street-map",
        'center': { 'lon': -73.6, 'lat': 45.5},
        'zoom': 12, 'layers': [{
            'source': {
                'type': "FeatureCollection",
                'features': [{
                    'type': "Feature",
                    'geometry': {
                        'type': "MultiPolygon",
                        'coordinates': [[[
                            [-73.606352888, 45.507489991], [-73.606133883, 45.50687600],
                            [-73.605905904, 45.506773980], [-73.603533905, 45.505698946],
                            [-73.602475870, 45.506856969], [-73.600031904, 45.505696003],
                            [-73.599379992, 45.505389066], [-73.599119902, 45.505632008],
                            [-73.598896977, 45.505514039], [-73.598783894, 45.505617001],
                            [-73.591308727, 45.516246185], [-73.591380782, 45.516280145],
                            [-73.596778656, 45.518690062], [-73.602796770, 45.521348046],
                            [-73.612239983, 45.525564037], [-73.612422919, 45.525642061],
                            [-73.617229085, 45.527751983], [-73.617279234, 45.527774160],
                            [-73.617304713, 45.527741334], [-73.617492052, 45.527498362],
                            [-73.617533258, 45.527512253], [-73.618074188, 45.526759105],
                            [-73.618271651, 45.526500673], [-73.618446320, 45.526287943],
                            [-73.618968507, 45.525698560], [-73.619388002, 45.525216750],
                            [-73.619532966, 45.525064183], [-73.619686662, 45.524889290],
                            [-73.619787038, 45.524770086], [-73.619925742, 45.524584939],
                            [-73.619954486, 45.524557690], [-73.620122362, 45.524377961],
                            [-73.620201713, 45.524298907], [-73.620775593, 45.523650879]
                        ]]]
                    }
                }]
            },
            'type': "fill", 'below': "traces", 'color': "royalblue"}]},
    margin = {'l':0, 'r':0, 'b':0, 't':0})

fig.show()
```



Mapbox Maps

> Mapbox traces are deprecated and may be removed in a future version of Plotly.py.

The earlier examples using go.Scattermap use Maplibre (https://maplibre.org/maplibre-gl-js/docs/) for rendering. This trace was introduced in Plotly.py 5.24 and is now the recommended way to draw filled areas on tile-based maps. There is also a trace that uses Mapbox (https://docs.mapbox.com), called go.Scattermapbox.

To use the Scattermapbox trace type, in some cases you *may* need a Mapbox account and a public Mapbox Access Token (https://www.mapbox.com/studio). See our Mapbox Map Layers (/python/mapbox-layers/) documentation for more information.

Here's one of the earlier examples rewritten to use Scattermapbox.

```
import plotly.graph_objects as go

fig = go.Figure(go.Scattermapbox(
    fill = "toself",
    lon = [-74, -70, -70, -74], lat = [47, 47, 45, 45],
    marker = { 'size': 10, 'color': "orange" }))

fig.update_layout(
    mapbox = {
        'style': "open-street-map",
        'center': {'lon': -73, 'lat': 46 },
        'zoom': 5},
    showlegend = False)

fig.show()
```

Reference

See https://plotly.com/python/reference/scattermap/ (https://plotly.com/python/reference/scattermap/) for available attribute options, or for go.Scattermapbox, see https://plotly.com/python/reference/scattermapbox/ (https://plotly.com/python/reference/scattermapbox/).

What About Dash?

Dash (https://dash.plot.ly/) is an open-source framework for building analytical applications, with no Javascript required, and it is tightly integrated with the Plotly graphing library.

Learn about how to install Dash at https://dash.plot.ly/installation (https://dash.plot.ly/installation).

Everywhere in this page that you see fig.show(), you can display the same figure in a Dash application by passing it to the figure argument of the Graph component (https://dash.plot.ly/dash-core-components/graph) from the built-in dash_core_components package like this:

```
import plotly.graph_objects as go # or plotly.express as px
fig = go.Figure() # or any Plotly Express function e.g. px.bar(...)
# fig.add_trace( ... )
# fig.update_layout( ... )

from dash import Dash, dcc, html

app = Dash()
app.layout = html.Div([
    dcc.Graph(figure=fig)
])

app.run(debug=True, use_reloader=False)  # Turn off reloader if inside Jupyter
```

nap

(https://dash.plotly.com/tutorial?utm_medium=graphing_libraries&utm_content=python_footer)

JOIN OUR MAILING LIST

Sign up to stay in the loop with all things Plotly — from Dash Club to product updates, webinars, and more!

SUBSCRIBE (HTTPS://GO.PLOT.LY/SUBSCRIPTION)

Products

Dash (https://plotly.com/dash/)
Consulting and Training (https://plotly.com/consulting-and-oem/)

Pricing

Enterprise Pricing (https://plotly.com/get-pricing/)

About Us

Careers (https://plotly.com/careers)
Resources (https://plotly.com/resources/)
Blog (https://medium.com/@plotlygraphs)

Support

Community Support (https://community.plot.ly/)
Documentation (https://plotly.com/graphing-libraries)

Terms of Service (https://community.plotly.com/tos) Privacy Policy (https://plotly.com/privacy/)