Bubble Maps in Python

How to make bubble maps in Python with Plotly.

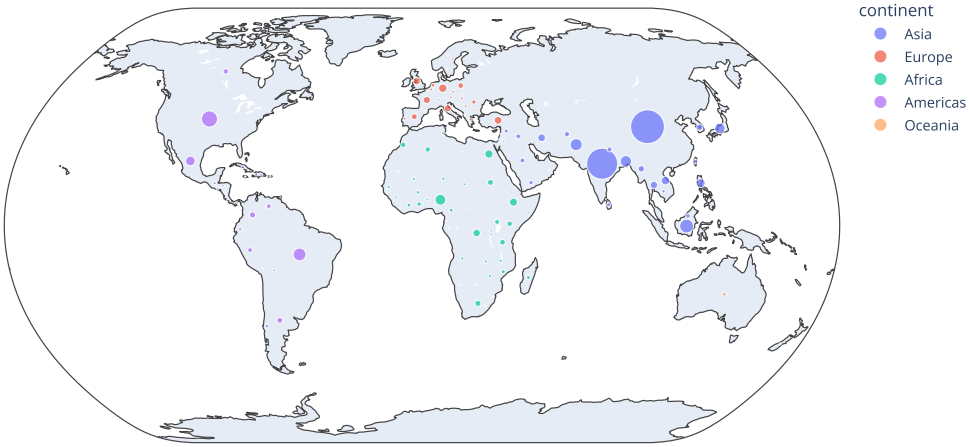
Base Map Configuration

Plotly figures made with Plotly Express (/python/plotly-express/) px.scatter_geo, px.line_geo or px.choropleth functions or containing go.Choropleth or go.Scattergeo graph objects (/python/graph-objects/) have a go.layout.Geo object which can be used to control the appearance of the base map (/python/map-configuration/) onto which data is plotted.

Bubble map with Plotly Express

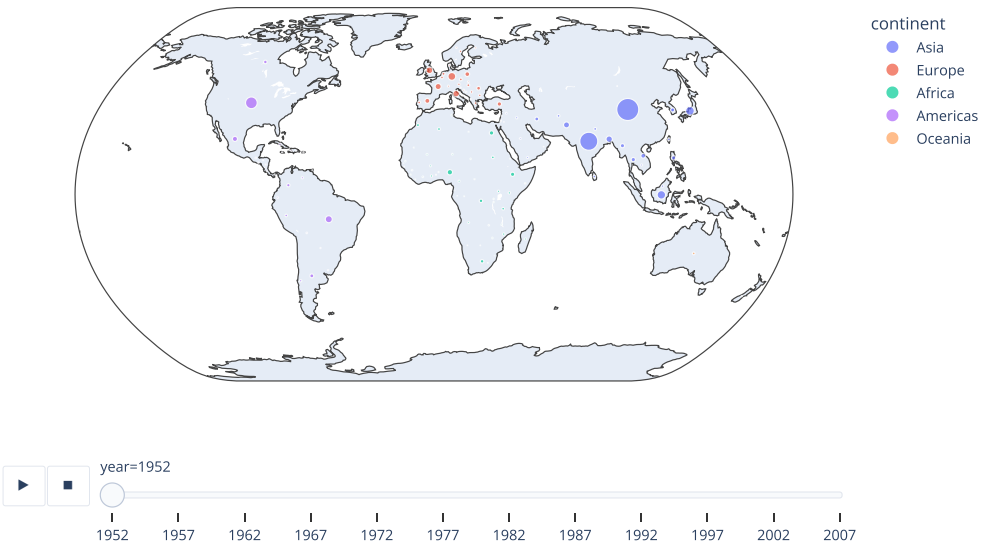
Plotly Express (/python/plotly-express/) is the easy-to-use, high-level interface to Plotly, which operates on a variety of types of data (/python/px-arguments/) and produces easy-to-style figures (/python/styling-plotly-express/). With px.scatter_geo, each line of the dataframe is represented as a marker point. The column set as the size argument gives the size of markers.

```
import plotly.express as px
df = px.data.gapminder().query("year==2007")
fig = px.scatter_geo(df, locations="iso_alpha", color="continent",
                     hover_name="country", size="pop",
                     projection="natural earth")
fig.show()
```



Bubble Map with animation

```
import plotly.express as px
df = px.data.gapminder()
fig = px.scatter_geo(df, locations="iso_alpha", color="continent",
                     hover_name="country", size="pop",
                     animation_frame="year",
                     projection="natural earth")
fig.show()
```



Bubble Map with go.Scattergeo

United States Bubble Map

Note about sizeref:

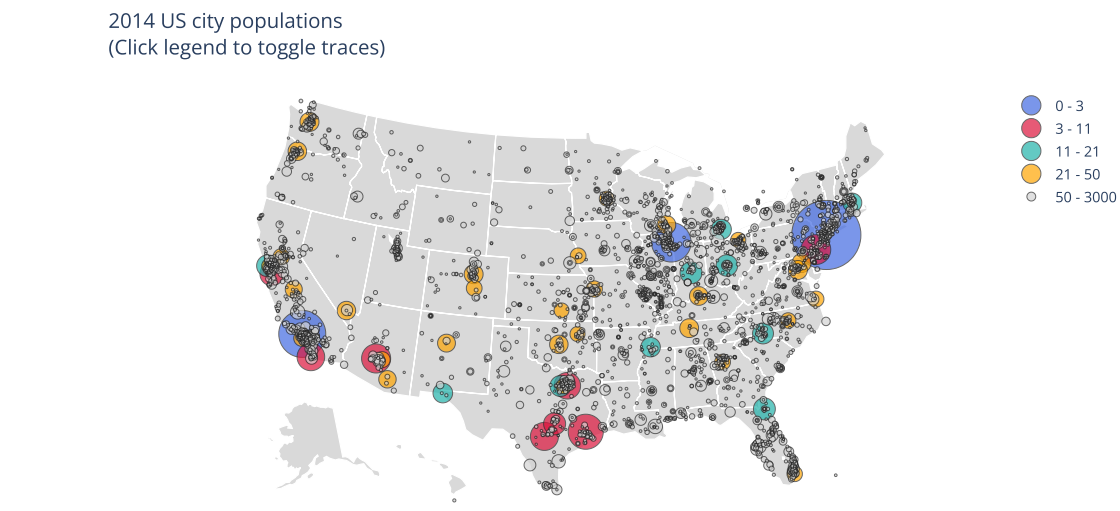
To scale the bubble size, use the attribute sizeref. We recommend using the following formula to calculate a sizeref value:

sizeref = 2. * max(array of size values) / (desired maximum marker size ** 2)

Note that setting sizeref to a value greater than 1, decreases the rendered marker sizes, while setting sizeref to less than 1, increases the rendered marker sizes.

See https://plotly.com/python/reference/scatter/#scatter-marker-sizeref (https://plotly.com/python/reference/scatter/#scatter-marker-sizeref) for more information. Additionally, we recommend setting the sizemode attribute: https://plotly.com/python/reference/scatter/#scatter-marker-sizemode (https://plotly.com/python/reference/scatter/#scatter-marker-sizemode) to area.

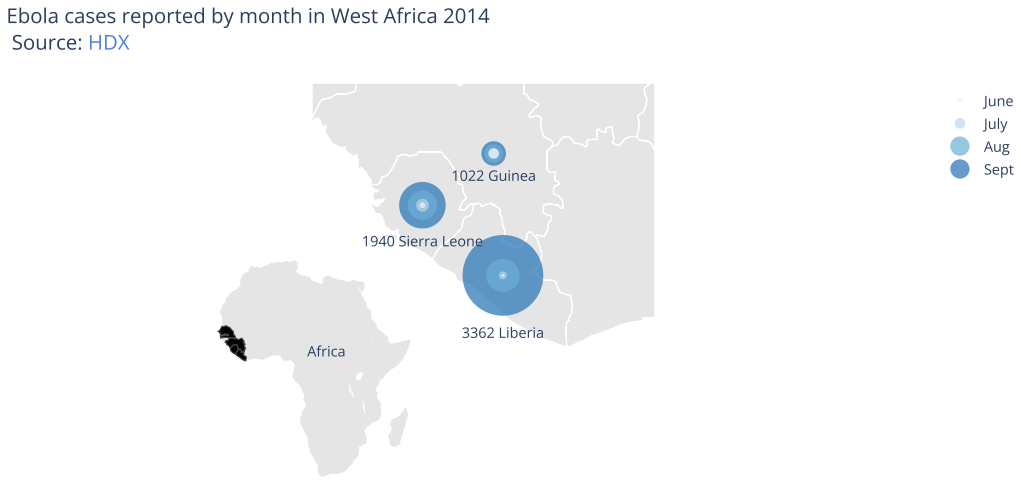
```
import plotly.graph_objects as go

import pandas as pd

df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/2014_us_cities.csv')
df.head()

df['text'] = df['name'] + '<br>Population ' + (df['pop']/1e6).astype(str)+' million'
limits = [(0,3),(3,11),(11,21),(21,50),(50,3000)]
colors = ["royalblue","crimson","lightseagreen","orange","lightgrey"]
cities = []
scale = 5000

fig = go.Figure()

for i in range(len(limits)):
    lim = limits[i]
    df_sub = df[lim[0]:lim[1]]
    fig.add_trace(go.Scattergeo(
        locationmode = 'USA-states',
        lon = df_sub['lon'],
        lat = df_sub['lat'],
        text = df_sub['text'],
        marker = dict(
            size = df_sub['pop']/scale,
            color = colors[i],
            line_color='rgb(40,40,40)',
            line_width=0.5,
            sizemode = 'area'
        ),
        name = '{0} - {1}'.format(lim[0],lim[1])))

fig.update_layout(
        title_text = '2014 US city populations<br>(Click legend to toggle traces)',
        showlegend = True,
        geo = dict(
            scope = 'usa',
            landcolor = 'rgb(217, 217, 217)',
        )
    )

fig.show()
```



West Africa

```
import plotly.graph_objects as go

import pandas as pd

df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/2014_ebola.csv')
df.head()

colors = ['rgb(239,243,255)','rgb(189,215,231)','rgb(107,174,214)','rgb(33,113,181)']
months = {6:'June',7:'July',8:'Aug',9:'Sept'}

fig = go.Figure()

for i in range(6,10)[::-1]:
    df_month = df.query('Month == %d' %i)
    fig.add_trace(go.Scattergeo(
            lon = df_month['Lon'],
            lat = df_month['Lat'],
            text = df_month['Value'],
            name = months[i],
            marker = dict(
                size = df_month['Value']/50,
                color = colors[i-6],
                line_width = 0
            )))

df_sept = df.query('Month == 9')
fig['data'][0].update(mode='markers+text', textposition='bottom center',
                      text=df_sept['Value'].map('{:.0f}'.format).astype(str)+' '+\
                      df_sept['Country'])

# Inset
fig.add_trace(go.Choropleth(
        locationmode = 'country names',
        locations = df_sept['Country'],
        z = df_sept['Value'],
        text = df_sept['Country'],
        colorscale = [[0,'rgb(0, 0, 0)'],[1,'rgb(0, 0, 0)']],
        autocolorscale = False,
        showscale = False,
        geo = 'geo2'
    ))
fig.add_trace(go.Scattergeo(
        lon = [21.0936],
        lat = [7.1881],
        text = ['Africa'],
        mode = 'text',
        showlegend = False,
        geo = 'geo2'
    ))

fig.update_layout(
    title = go.layout.Title(
        text = 'Ebola cases reported by month in West Africa 2014<br> \
Source: <a href="https://data.hdx.rwlabs.org/dataset/rowca-ebola-cases">\
HDX</a>'),
    geo = go.layout.Geo(
        resolution = 50,
        scope = 'africa',
        showframe = False,
        showcoastlines = True,
        landcolor = "rgb(229, 229, 229)",
        countrycolor = "white" ,
        coastlinecolor = "white",
        projection_type = 'mercator',
        lonaxis_range= [ -15.0, -5.0 ],
        lataxis_range= [ 0.0, 12.0 ],
        domain = dict(x = [ 0, 1 ], y = [ 0, 1 ])
    ),
    geo2 = go.layout.Geo(
        scope = 'africa',
        showframe = False,
        landcolor = "rgb(229, 229, 229)",
        showcountries = False,
        domain = dict(x = [ 0, 0.6 ], y = [ 0, 0.6 ]),
        bgcolor = 'rgba(255, 255, 255, 0.0)',
    ),
    legend_traceorder = 'reversed'
```

```
fig.show()
```



Reference

See function reference for px.(scatter_geo) (https://plotly.com/python-api-reference/generated/plotly.express.scatter_geo) or https://plotly.com/python/reference/choropleth/ (https://plotly.com/python/reference/choropleth/) and https://plotly.com/python/reference/scattergeo/ (https://plotly.com/python/reference/scattergeo/) for more information and chart attribute options!

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
```