Beta Release

April 25, 2018

1 Strava-Vis: Beta Release

Alex Howard & Taylor Pellerin

```
In [1]: import pandas as pd
        from datetime import *
        import numpy as np
        import math
        import  plotly.plotly  as py
        import plotly.graph_objs as go
        import json
        from geoplotlib.layers import BaseLayer
        from geoplotlib.core import BatchPainter
        import geoplotlib
        from geoplotlib.colors import colorbrewer
        from geoplotlib.utils import epoch_to_str, BoundingBox, read_csv
```

```
In [2]: metres_mile = 1609.34
        workout_type_dict = {0:'Run',1:'Race',2:'Long Run',3:'Workout'}
```

1.1 Data Pre-Processing

```
In [3]: activities_df = pd.read_json('alex_all_acts.json')
        data = json.load(open('alex_activities_latlng.json'))
```

```
In [4]: activities_df = activities_df[['average_speed','distance','moving_time','name','start_
        activities_df = activities_df[activities_df.type == 'Run']
```

```
In [5]: activities_df['pace_mile'] = metres_mile / activities_df.average_speed
        activities_df['pace_km'] = 1000 / activities_df.average_speed
```

```
In [6]: activities_df['date'] = pd.to_datetime(activities_df.start_date_local.apply(lambda x :
        activities_df.drop(['average_speed','start_date_local', 'type', 'id'], axis = 1, inpla
```

```
In [7]: activities_df.workout_type = activities_df.workout_type.fillna(0)
        activities_df.workout_type = activities_df.workout_type.apply(lambda x : workout_type_
```

```
In [8]: activities_df['miles'] = activities_df.distance / metres_mile
        activities_df['Distance (Kilometres)'] = activities_df.distance / 1000
```

```
In [9]: activities_df['size'] = activities_df.moving_time.astype('float').apply(lambda x : matl
        sizeref = 20*max(activities_df['size'])/(100**2)
        activities_df['year'] = activities_df.date.apply(lambda x: x.year)
```

```
In [10]: activities_text = []
         for i in range(len(activities_df)):
             row = activities_df.iloc[i,]
             activities_text.append('{}<br>{}<br>'.format(row['name'].encode('ascii','ignore')

         activities_df['text'] = activities_text
```

```
In [11]: activities_df.head(2)
```

```
Out[11]:    distance  moving_time         name workout_type   pace_mile     pace_km  \
         0   10324.0         2649  Morning Run          Run  412.968950  256.607647
         1    4347.8         1156           WD          Run  427.902154  265.886732

                 date     miles  Distance (Kilometres)       size  year  \
         0 2018-04-18  6.415052                10.3240  51.468437  2018
         1 2018-04-17  2.701604                 4.3478  34.000000  2018

                                                 text
         0  Morning Run<br>2018-04-18 00:00:00<br>6.4 mile...
         1  WD<br>2018-04-17 00:00:00<br>2.7 miles<br>427...
```

1.2 1. Bubble Chart

```
In [12]: data = []
         for run_type in ['Run','Workout','Long Run','Race']:
             trace = go.Scatter(
                 x=activities_df['miles'][activities_df['workout_type'] == run_type],
                 y=activities_df['pace_mile'][activities_df['workout_type'] == run_type],
                 mode='markers',
                 hoverinfo='text',
                 opacity = 0.8,
                 name=run_type,
                 hovertext = activities_df['text'][activities_df['workout_type'] == run_type],
                 marker=dict(
                     symbol='circle',
                     sizemode='area',
                     sizeref=sizeref,
                     size=activities_df['size'][activities_df['workout_type'] == run_type],
                     line=dict(
                         width=2
                     ),
                 )
             )
             data.append(trace)
```

```
layout = go.Layout(
    title='Run Summary',
    hovermode='closest',
    xaxis=dict(
        title='Distance (Miles)',
        gridcolor='rgb(255, 255, 255)',
        range=[0, 20],
        zerolinewidth=1,
        ticklen=5,
        gridwidth=2,
    ),
    yaxis=dict(
        title='Pace (Seconds per Mile)',
        gridcolor='rgb(255, 255, 255)',
        range=[0,600],
        zerolinewidth=1,
        ticklen=5,
        gridwidth=2,
    ),
    paper_bgcolor='rgb(243, 243, 243)',
    plot_bgcolor='rgb(243, 243, 243)',
)

fig = go.Figure(data=data, layout=layout)

py.iplot(fig, filename='bubble_chart_test.fig')
```

```
Out[12]: <plotly.tools.PlotlyDisplay object>
```

1.3 2. Parallel Coordinates

```
In [13]: activities_grouped_df = activities_df.groupby(['date'], as_index = False)['miles'].sur
         activities_grouped_df['dow'] = activities_grouped_df.date.apply(lambda x : x.weekday(
         activities_grouped_df['week_start'] = activities_grouped_df.date.apply(lambda x : x -

         miles_per_week = activities_grouped_df.groupby(['week_start'], as_index = False).miles
         by_week_df = pd.DataFrame(activities_grouped_df.week_start.unique(), columns = ['week_
```

```
In [14]: for i in range(7):
             by_week_df['{}'.format(i)] = i

         for i in range(7):
             by_week_df = pd.merge(by_week_df, activities_grouped_df, left_on = ['week_start',
```

```
In [15]: by_week_df = by_week_df[['week_start','miles','miles_1','miles_2','miles_3','miles_4'
         by_week_df.columns = ['week_start','miles_0','miles_1','miles_2','miles_3','miles_4',
         by_week_df['year'] = by_week_df['week_start'].apply(lambda x : x.year)
         by_week_df.fillna(0, inplace = True)
         by_week_df = pd.merge(by_week_df, miles_per_week, how='left', on='week_start')
```

```
In [16]: days_dict = {0:'Monday',1:'Tuesday',2:'Wednesday',3:'Thursday',4:'Friday',5:'Saturday
```

```
In [17]: dimensions = list()

         for i in range(7):
             dimensions.append(
                     dict(range = [0,20],
                         constraintrange = [0,20],
                         label = '{}'.format(days_dict[i]), values = by_week_df['miles_{}'.forr
```

```
In [18]: data = [
             go.Parcoords(
                 line = dict(color = by_week_df['miles'],
                            colorscale = 'Hot',
                            showscale = True,
                            reversescale=True),
                 opacity=0.5,
                 dimensions = dimensions,hoverinfo='text')

         ]

         layout = go.Layout(
             plot_bgcolor = '#E5E5E5',
             paper_bgcolor = '#E5E5E5',
             title = 'Miles per week broken down by day'
         )

         fig = go.Figure(data = data, layout = layout)
         py.iplot(fig, filename = 'parcoords')
```

```
Out[18]: <plotly.tools.PlotlyDisplay object>
```

1.4 3. Miles Per Week

```
In [19]: by_week_df['week_end'] = by_week_df['week_start'].apply(lambda x: (x + timedelta(days=
```

```
In [20]: data = []
         for i in range(7):
             data.append(go.Bar(
                 x=by_week_df['week_start'],
                 y=by_week_df['miles_{}'.format(i)],
                 name=days_dict[i]))
```

```
In [21]: layout = go.Layout(
             barmode='stack'
         )
```

```
In [22]: layout = dict(
             barmode='stack',
             hovermode='closest',
             title='Miles per week',
             xaxis=dict(
                 rangeselector=dict(
                     buttons=list([
                         dict(count=1,
                              label='1m',
                              step='month',
                              stepmode='backward'),
                         dict(count=6,
                              label='6m',
                              step='month',
                              stepmode='backward'),
                         dict(count=1,
                             label='YTD',
                             step='year',
                             stepmode='todate'),
                         dict(count=1,
                             label='1y',
                             step='year',
                             stepmode='backward'),
                         dict(step='all')
                     ])
                 ),
                 rangeslider=dict(),
                 type='date'
             )
         )
```

```
In [23]: fig = go.Figure(data=data, layout=layout)
         py.iplot(fig, filename='stacked-bar')
```

```
Out[23]: <plotly.tools.PlotlyDisplay object>
```

1.5 4. Geographic Visualisation:

```
In [24]: class AllTrailsLayer(BaseLayer):

             def __init__(self):
                 self.data = read_csv('alex.csv')
                 self.cmap = colorbrewer(self.data['runner_id'], alpha=220)
                 self.t = self.data['timestamp'].min()
                 self.painter = BatchPainter()


             def draw(self, proj, mouse_x, mouse_y, ui_manager):
                 self.painter = BatchPainter()
                 df = self.data.where((self.data['timestamp'] > self.t) & (self.data['timestam
                 for taxi_id in set(df['runner_id']):
                     grp = df.where(df['runner_id'] == taxi_id)
                     self.painter.set_color(self.cmap[taxi_id])
                     x, y = proj.lonlat_to_screen(grp['lon'], grp['lat'])
                     self.painter.points(x, y, 10)

                 self.t += 2*60

                 if self.t > self.data['timestamp'].max():
                     self.t = self.data['timestamp'].min()

                 self.painter.batch_draw()
                 ui_manager.info(epoch_to_str(self.t))


             # this should get modified as well moving forward. Might be too small
             def bbox(self):
                 return BoundingBox(north=37.801421, west=-122.517339, south=37.730097, east=-
```

```
In [25]: geoplotlib.add_layer(AllTrailsLayer())
         geoplotlib.show()
```

```
In [26]: class FollowTrailsLayer(BaseLayer):

             def __init__(self):
                 self.data = read_csv('alex.csv')
                 self.data = self.data.where(self.data['runner_id'] == list(set(self.data['runn
                 self.t = self.data['timestamp'].min()
                 self.painter = BatchPainter()


             def draw(self, proj, mouse_x, mouse_y, ui_manager):
                 self.painter = BatchPainter()
                 self.painter.set_color([0,0,255])
                 df = self.data.where((self.data['timestamp'] > self.t) & (self.data['timestamp
                 proj.fit(BoundingBox.from_points(lons=df['lon'], lats=df['lat']), max_zoom=14)
                 x, y = proj.lonlat_to_screen(df['lon'], df['lat'])
                 self.painter.linestrip(x, y, 10)
                 self.t += 30
                 if self.t > self.data['timestamp'].max():
                     self.t = self.data['timestamp'].min()

                 self.painter.batch_draw()
                 ui_manager.info(epoch_to_str(self.t))
```

```
In [27]: geoplotlib.add_layer(FollowTrailsLayer())
         geoplotlib.show()
```