# estimating the footprint of a conference. Code examples

March 29, 2018

## 0.1 Examples of code used to calculate the footprint of a conference

the code and further explanations will also be hosted at https://github.com/mfastudillo?tab=repositories

import statements

```
In [1]: import pandas as pd
        import numpy as np
        import math
        import matplotlib.pyplot as plt

        from geopy import distance
        from geopy import Point
        import geopy.geocoders
        from scipy.spatial.distance import cdist, euclidean

        import json
        import requests
```

```
In [2]: %matplotlib inline
```

## 0.2 geolocate

Several geocoders are available, in this case I use the one provided by Google.

```
In [3]: gg= geopy.geocoders.GoogleV3()
```

The geolocation algorithm does a good job even with poorly described address, as far as they are unequivocal. It works much better when an API key is used, but for demonstration purposes it is not necessary.

```
In [4]: Udes_address='Universite de Sherbrooke, Sherbrooke'
        MIT_address='MIT'
        Caltech_address='Caltech'
        conference_address=('Portsmouth, NH 03801, USA')
```

```
In [5]: conference_loc=gg.geocode(conference_address)
```

```
In [8]: udes_loc=gg.geocode(Udes_address)
```

```
In [9]: MIT_loc=gg.geocode(MIT_address)

In [10]: caltech_loc=gg.geocode(Caltech_address)

In [11]: print(conference_loc)
         print(udes_loc)
         print(caltech_loc)
         print(MIT_loc)
```

```
Portsmouth, NH 03801, USA
2500 Boulevard de l'Université, Immeuble K1, Sherbrooke, QC J1K 2R1, Canada
1200 E California Blvd, Pasadena, CA 91125, USA
77 Massachusetts Ave, Cambridge, MA 02139, USA
```

### 0.3 calculate *as the crow flies* distance

The distance between two points is calculated as the geodesic distance using the Vicenty formula

```
In [12]: #example: road distance between the conference and Sherbrooke University
         distance.distance((conference_loc.latitude,conference_loc.longitude),
                           (udes_loc.latitude,udes_loc.longitude)).kilometers
```

```
Out[12]: 272.4397133919147
```

### 0.4 Calculate distance by road

example of estimation of distance travelled by road using the Google Maps API. Distance by road between Sherbrooke University and the place where the conference took place.

```
In [13]: base_url='https://maps.googleapis.com/maps/api/distancematrix/json?'

In [14]: req={'origins':'Universite de Sherbrooke, Sherbrooke',
              'destinations':'Portsmouth, NH 03801, USA',
              'mode': 'driving'}

In [15]: r =requests.get(base_url,params=req)

In [16]: x = json.loads(r.text)

In [17]: x
```

```
Out[17]: {'destination_addresses': ['Portsmouth, NH 03801, USA'],
          'origin_addresses': ["2500 Boulevard de l'Université, Immeuble K1, Sherbrooke, QC J1
          'rows': [{'elements': [{'distance': {'text': '394 km', 'value': 394170},
              'duration': {'text': '3 hours 58 mins', 'value': 14254},
              'status': 'OK'}]}],
          'status': 'OK'}
```

```
In [18]: x['rows'][0]['elements'][0]['distance']['text']
```

```
Out[18]: '394 km'
```

### 0.5 calculating the optimal location

The location wich minimises the distance to all the participants is calculated using a vectorized implementation of the algorithm provided here. The code of this function was written by Orson Peters

```
In [19]: def geometric_median(X, eps=1e-6):
             y = np.mean(X, 0)

             while True:
                 D = cdist(X, [y])
                 nonzeros = (D != 0)[:, 0]

                 Dinv = 1 / D[nonzeros]
                 Dinvs = np.sum(Dinv)
                 W = Dinv / Dinvs
                 T = np.sum(W * X[nonzeros], 0)

                 num_zeros = len(X) - np.sum(nonzeros)
                 if num_zeros == 0:
                     y1 = T
                 elif num_zeros == len(X):
                     return y
                 else:
                     R = (T - y) * Dinvs
                     r = np.linalg.norm(R)
                     rinv = 0 if r == 0 else num_zeros/r
                     y1 = max(0, 1-rinv)*T + min(1, rinv)*y

                 if euclidean(y, y1) < eps:
                     return y1

                 y = y1
```

```
In [20]: origins=np.array([(udes_loc.latitude,udes_loc.longitude),
                            (MIT_loc.latitude,MIT_loc.longitude),
                            (caltech_loc.latitude,caltech_loc.longitude)])
```

```
In [21]: ideal_location=geometric_median(origins,eps=1e-8)
```

```
In [22]: ideal_location
```

```
Out[22]: array([ 43.83938554, -72.42544609])
```

### 0.6 optimal location visualization

```
In [23]: odf=pd.DataFrame(origins,index=['udes','MIT','Caltech'],columns=['lat','long'])
```

```
In [24]: #origins
         odf.plot(kind='scatter',x='long',y='lat')
```

```
#optimal location (minimization of euclidean distance)
plt.plot(ideal_location[1],ideal_location[0],'o',color='r');
plt.annotate('ideal meeting point',
             xy=(ideal_location[1],ideal_location[0]),
             xytext=(ideal_location[1]-20,ideal_location[0]),
             arrowprops=dict(facecolor='black', shrink=0.05)
            );
plt.title('the location that minimises the sum of the distance\
          \n travelled by participats.');
```