Contour map: Popular Locations

James Premo

This contour library uses data given in geographic format, latitude (N°) and longitude (E°). A map is generated from openstreetmap [1] then plotted using matplotlib[2] and geotiler[3] libraries. Then a vector field is generated using gaussian kernel density estimation(KDE) from the scipy library [4] . The input for the KDE are the points provided to the library, which were assigned to a numpy array (*np.array()*). Then this vector field is applied to the *matplotlib.contourf()* function. Lastly the points are plotted with simple dots using the *matplotlib.scatter()* function. This can work on any city in the world, or any location.

Algorithm:

1. Read points into numpy array (*np.array()*) from a file.
2. Find the rough center of points given using the mid point equation.
3. Retrieve map from openstreetmap using the center of the points as the center of the map.
4. Plot the map onto a matplotlib figure.
5. Using the array of points convert them to cartesian, for use in figure.
6. Apply this array to a gaussian kernel density estimate (*scipy.stats.gaussian_kde()*).
7. Plot the output of the KDE to a contour map (*matplotlib.confourf*).
8. Layer the contour map over the map image. With a lower alpha to the contour to make it translucent.
9. Output figure to PNG file (or any other format supported by matplotlib).

Usage:

Python 3 IDE

```
>>>import contour
>>>contour.Contour('File with points')
```

contour.***Contour***(*filename, pix_size=2000, inch_size=10,dpi=200,zoom=14*)

filename: Filename containing points.
zoom: Zoom of openstreetmap.
pix_size: Size of image downloaded from open-street map.
inch_size: Size of *matplotlib* figure in inches.
dpi: Number of pixels per inch of *matplotlib* figure.

File Format:

The format for the file of points has to be in the format: [*latitude,longitude*]. Separated by a comma with no space, and each coordinate has to be be on new line. The file should be plain text ASCII.

Latitude: Has to be in terms of N°. (Incorrect: *1° S* Correct: *-1° N*)

Longitude: Has to be in terms of E°. (Incorrect: *1° W* Correct: *-1°* E)

Sample data Random points in London:

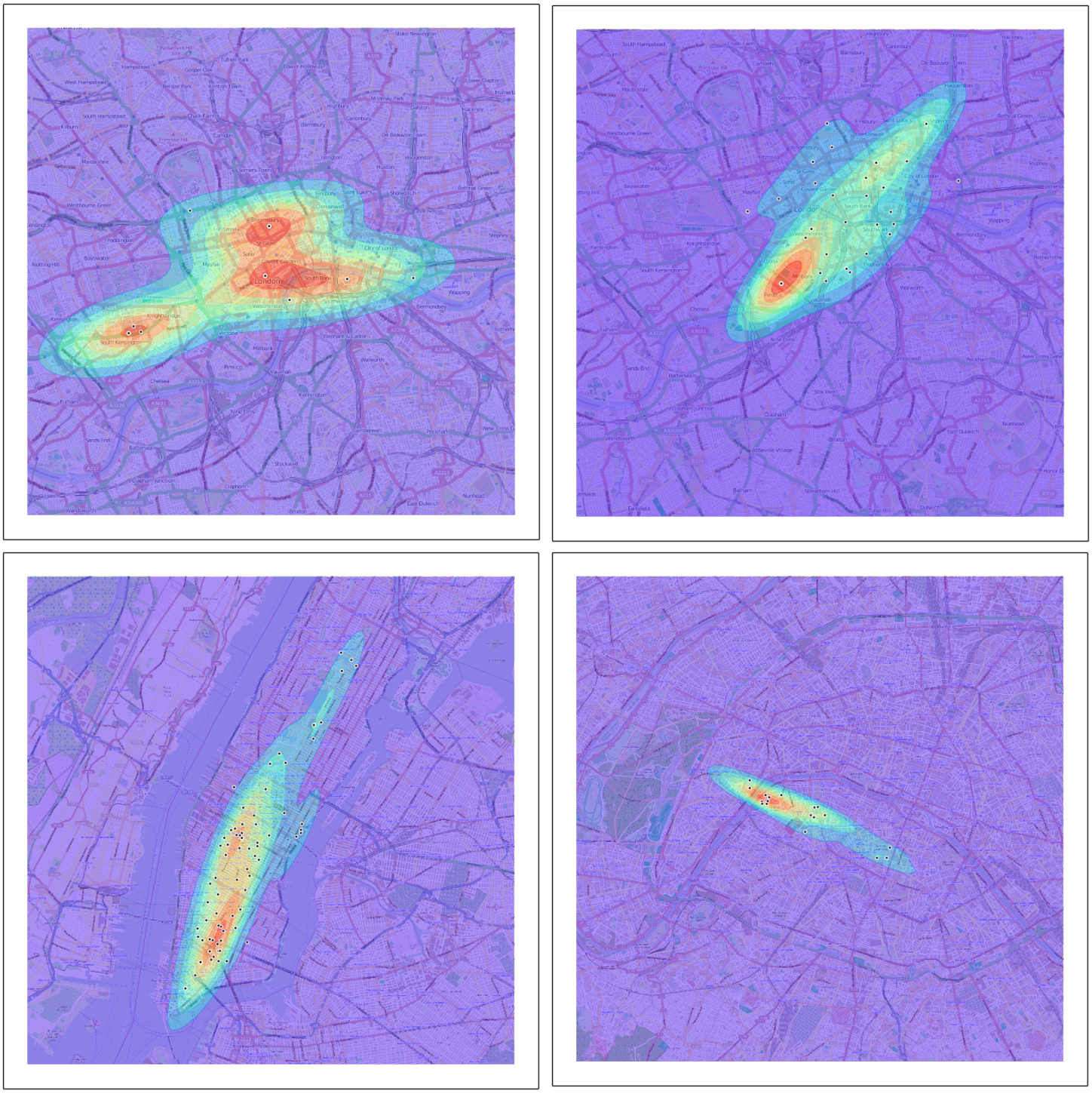
[*Latitude*° N, *Longitude*° E]

```
51.50308,-0.125999
51.50308,-0.125999
51.50308,-0.125999
51.513123,-0.124626
51.509704,-0.137501
51.497523,-0.120678
51.504041,-0.102825
51.517609,-0.103168
51.517823,-0.125484
51.526261,-0.120506
51.51793,-0.092354
51.504148,-0.096989
51.494317,-0.11364
51.49111,-0.136642
51.506926,-0.148487
51.521028,-0.11879
51.513657,-0.074158
51.501904,-0.098362
51.493568,-0.112438
51.504575,-0.113983
51.49763,-0.106602
51.510345,-0.106602
```

Examples:

Top left: London (Popular Places) Top Right: London (Random Points)

Bottom Left: New York City (Popular places) Bottom Right: Paris (Popular Places)



Reference:

[1]: http://openstreetmap.org/
[2]: http://matplotlib.org/
[3]: http://wrobell.it-zone.org/geotiler/
[4]: http://www.scipy.org/
[5]: http://www.numpy.org/

Source code:

```
#start of code
#The MIT license (MIT)
#
#Copyright (c) 2014 James Premo
#
#Permission is hereby granted, free of charge, to any person obtaining a copy
#of this software and associated documentation files (the "Software"), to deal
#in the Software without restriction, including without limitation the rights
#to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
#copies of the Software, and to permit persons to whom the Software is
#furnished to do so, subject to the following conditions:
#
#The above copyright notice and this permission notice shall be included in
#all copies or substantial portions of the Software.
#
#THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
#IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
#FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
#AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
#LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
#OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
#THE SOFTWARE.

#!/usr/bin/python

import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import geotiler
from scipy.stats import gaussian_kde


class Contour():

    def __init__(self, in_f,pix_size=2000,inch_size=10,dpi=200,zoom=14):

        #init varibles
        self.in_f = in_f
        self.np = np
        self.plt = plt
        self.zoom = zoom
        self.pix_size = pix_size
```

```
        self.inch_size = inch_size
        self.dpi = dpi

        #start fig
        self.fig = self.plt.figure(figsize=(self.inch_size, self.inch_size), dpi=self.dpi)
        self.ax = self.plt.subplot(111)

        #init func.
        self.cor_array()

    def bounds(self):

        #min/max
        xmin = self.a[:,0].min()
        xmax = self.a[:,0].max()
        ymin = self.a[:,1].min()
        ymax = self.a[:,1].max()

        #center
        midx = (xmax + xmin)/2
        midy = (ymax + ymin)/2
        self.center = (float(midx), float(midy))

        self.map_l()

    def map_l(self):

        #download map and plot map
        self.mm = geotiler.Map(center=self.center, zoom=self.zoom, size =(self.pix_size,self.pix_size))
        self.img = geotiler.render_map(self.mm)
        self.ax.imshow(self.img)
        self.contour_l()

    def cor_array(self):

        #open file
        f = open(self.in_f,"r")
        #init array
        initline = f.readline()
        initline = initline.rstrip('\n')
        initline = tuple(float(x) for x in initline.split(","))
        self.a = np.array((initline[1],initline[0]))
        #put rest of data into array
        for line in f:
            if line == '\n':
                break
            line = line.rstrip('\n')
            line = tuple(float(x) for x in line.split(","))
            b = np.array((line[1],line[0]))
            self.a = np.vstack((self.a,b))

        self.bounds()

    def contour_l(self):

        #lat,long to x,y
        x, y = zip(*(self.mm.rev_geocode(p) for p in self.a))
```

```
        self.a = np.array([x,y])

        #data shaping
        X, Y = np.mgrid[0:self.pix_size:100j, 0:self.pix_size:100j]
        positions = np.vstack([X.ravel(),Y.ravel()])
        kernel = gaussian_kde(self.a)
        Z = np.reshape(kernel(positions).T, X.shape)

        #plot
        self.ax.contourf(X,Y,Z,cmap='rainbow',alpha=.5,linewidth=.4)
        self.ax.scatter(x, y, c='black', edgecolor='white', s=10, alpha=0.9)

        self.layer_out()


 def layer_out(self):

        #remove tics
        self.plt.gca().xaxis.set_major_locator(plt.NullLocator())
        self.plt.gca().yaxis.set_major_locator(plt.NullLocator())

        #output png
        self.plt.savefig('test.png', bbox_inches='tight')
        self.plt.close()

Contour('pnts_ny.txt')
#end of code
```

Points list created using, Click Latitude, Longitude.

pnts_ny.txt:

"40.7189,-73.996181\n40.71877,-73.999271\n
40.7189,-73.995667\n40.721632,-73.995323\n
40.721762,-74.001846\n40.716558,-73.998756\n
40.721762,-73.998241\n40.727096,-74.000473\n
40.712394,-74.004421\n40.769102,-73.972492\n
40.768842,-73.977985\n40.771572,-73.974724\n
40.75506,-73.98365\n40.743225,-73.981934\n
40.749858,-73.98983\n40.75714,-73.987427\n
40.762081,-73.97953\n40.762601,-73.990688\n
40.749728,-73.988628\n40.738933,-73.985195\n
40.748948,-73.987942\n40.749728,-73.978329\n
40.751158,-73.991718\n40.744526,-73.993607\n
40.746867,-73.982449\n40.751548,-73.990345\n
40.746997,-73.995495\n40.747777,-73.982449\n
40.752719,-73.98777\n40.749078,-73.994122\n
40.744266,-73.983307\n40.746347,-73.9888\n
40.749078,-73.987942\n40.752719,-73.982964\n
40.75571,-73.973007\n40.747257,-73.99292\n
40.738413,-73.997211\n40.716168,-73.996181\n
40.715777,-74.002361\n40.722673,-74.003391\n
40.725405,-73.995323\n40.717339,-73.997898\n
40.708881,-74.007854\n40.725665,-73.996181\n
40.737893,-73.989658\n40.733991,-73.996353\n
40.736852,-73.9991\n40.728917,-73.996868\n
40.715908,-74.002533\n40.724234,-73.994465\n
40.731909,-73.999958 \n40.725015,-73.990173\n
40.729958,-73.988457 \n40.735161,-73.98674\n
40.740754,-73.986053 \n40.746217,-73.985367\n
40.750378,-73.987942\n40.728267,-73.991203\n
40.716168,-73.993263\n40.721242,-73.986053\n
40.724885,-74.003563\n40.720721,-73.996525\n
40.722022,-73.999271\n40.750378,-73.967085\n
40.749403,-73.968716\n40.751223,-73.967085\n
40.752784,-73.966742\n40.779307,-73.96245\n
40.775537,-73.962708\n40.779177,-73.962536\n
40.779892,-73.959961\n40.798477,-73.952923\n
40.796528,-73.949318\n40.794968,-73.947601\n
40.793604,-73.952665\n"