

Supplementary material to:

An analysis about the accuracy of Probabilistic Geographic Profiling in relation to the number of observations

Fig. 1S. Descriptive Plot.

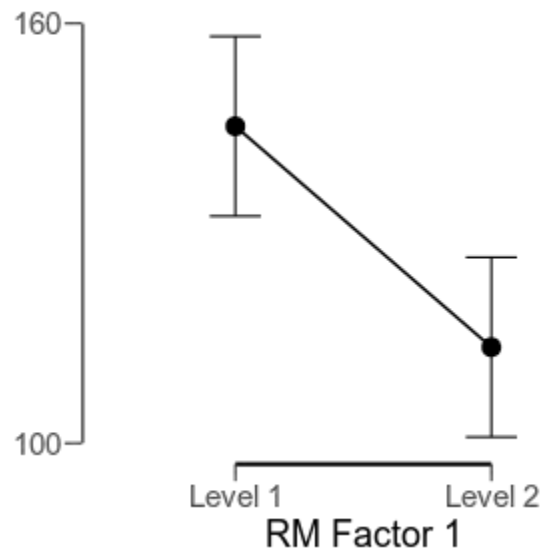


Fig. 2S (ex Figure 8). Distance between the curves with delta = 10%.

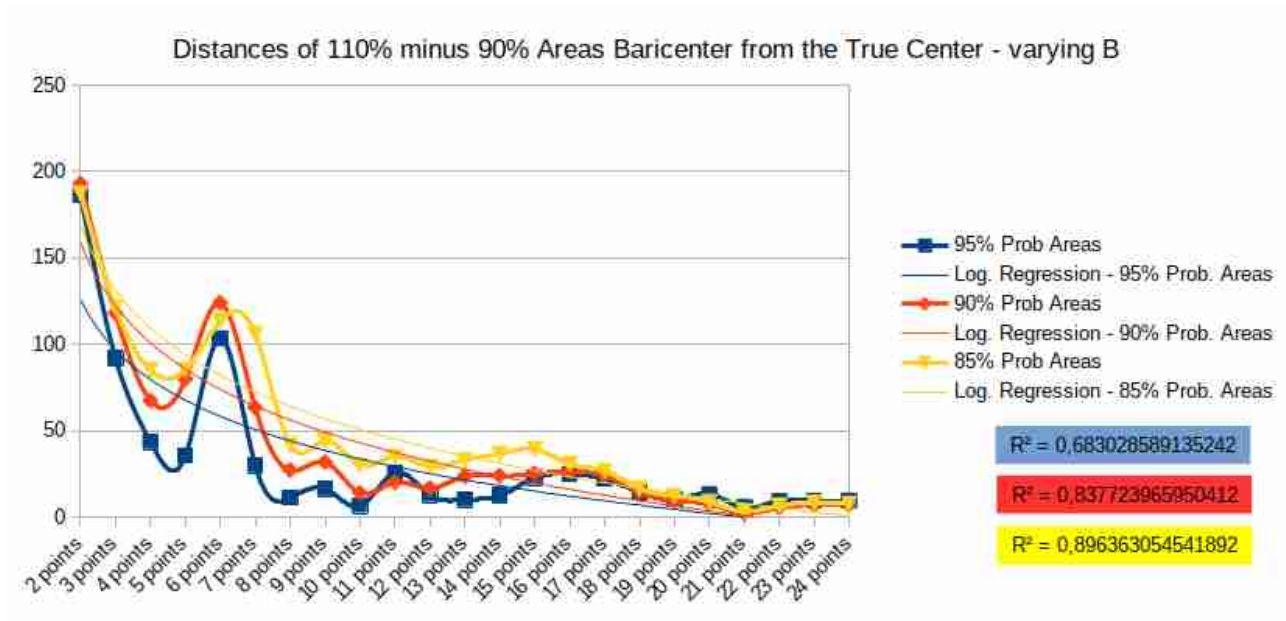


Fig. 3S. Distances of 120% minus 80% areas baricenter from the true center by variation of the B value.

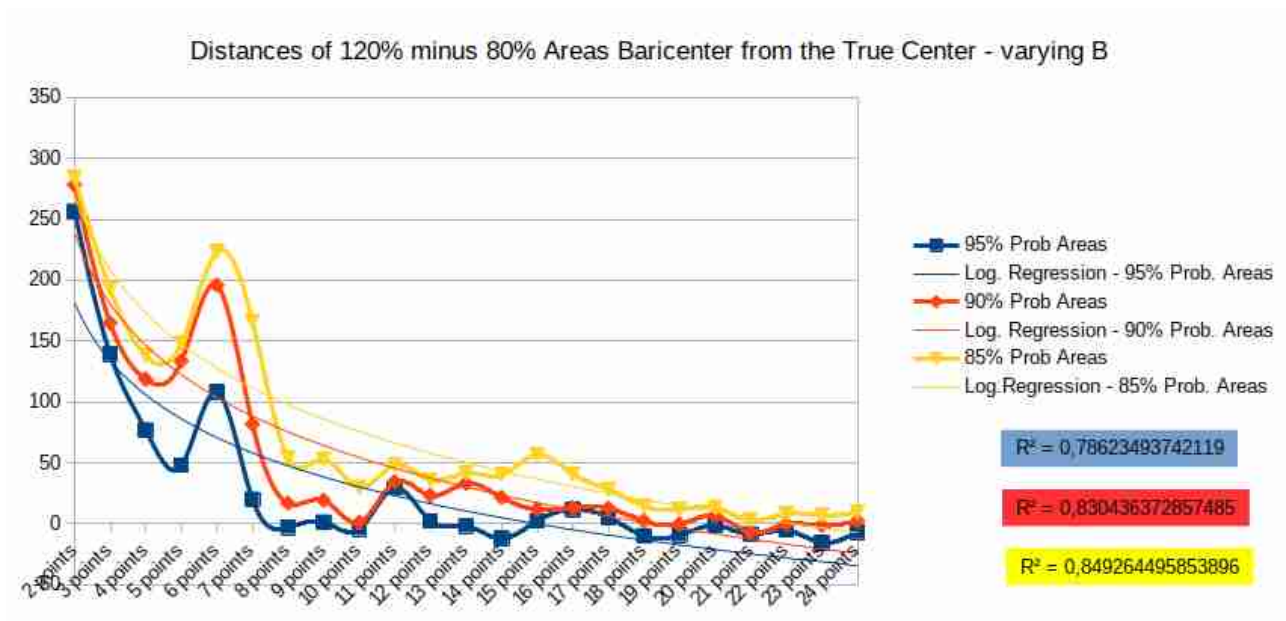


Table 1S. List of the cases related to Boston strangler case.

N.	Name	Surname	Age	Date	Address	City
1	Anna	Slesers	56	June 14 1962	77, Gainsborough St. Back Bay	Boston
2	Mary	Mullen	85	June 28 1962	1435, Commonwealth Ave.	Boston
3	Nina	Nichols	68	June 30 1962	1940, Commonwealth Ave.	Boston
4	Helen	Blake	65	June 30 1962	73, Newhall St.	Lynn
5	Ida	Irga	75	August 19 1962	7, Grove St. Beacon Hill	Boston
6	Jane	Sullivan	67	August 21 1962	435, Columbia Rd. Dorchester South	Boston
7	Sophie	Clark	20	December 5 1962	315, Huntington Ave. Back Bay	Boston
8	Patricia	Bissette	23	December 31 1962	515, Park Drive Back Bay	Boston
9	Mary	Brown	69	March 6 1963	319, Park Ave.	Lawrence
10	Beverly	Samans	23	May 6 1963	4, University Rd.	Cambridge
11	Evelyn	Corben	58	September 6 1963	224, Lafayette St.	Salem
12	Joann	Graff	23	November 23 1963	54, Essex St.	Lawrence
13	Mary	Sullivan	19	January 4 1964	44-A, Charles St.	Boston

Table 2S. Table of values conversion to the pixel position on the map.

Case ID	Real Distance from the offender's home (km)	X Coordinate	Y Coordinate	Distance from the offender's home (pixels)	B Values (Km)	B Values (pixels)
Home	0,00	493	633	0		
1	9,63	476	761	129,12	5,7544	75
2	10,41	421	748	135,68	6,5216	85
3	12,08	404	762	156,72	7,2889	95
4	11,50	635	587	149,26	8,0561	105
5	7,40	498	729	96,13	8,8233	115
6	12,85	498	800	134,73	9,5906	125
7	9,80	479	767	121,84	10,3578	135
8	9,32	459	750	431,73	11,1251	145
9	33,00	385	215	98,18	11,8923	155
10	7,36	439	715	227,79	12,6596	165
11	17,41	686	512	412,28	13,4268	175
12	31,79	406	230	412,28	14,1941	185
13	7,85	495	736	103,02	14,9613	195

Mean	13,88 km	200,68 pix					
Standard Deviation	8,65 km	128,82 pix					
Median	10,41 km	135,68 pix				Pixel Linear Dimension.	0,0767 Km

 How the Pareto algorithm works (see Python script My_Pareto_1.py)

Pseudo-algorithm:

It generates a sequence of $n_{max}/2$ points with a Pareto distribution as follows:

1. Generates a sequence of $n_{max}/2$ points with a Pareto distribution
2. Unites the two sequences one after the other and randomly mixes the data (radiuses vector)
3. Generates a sequence of n_{max} points with uniform distribution (angles vector)
4. Multiplies the radiuses vector so that the maximum is 95% of $\min(\text{image_width}/2, \text{image_height}/2)$ (normalization on the map).
5. Converts the two sequences in a two dimensional vector
6. Converts the vector from polar coordinates to cartesian coordinates
7. Translates the resulting vector so the centre coincides with that of the image ($\text{image_width}/2, \text{image_height}/2$)
8. Visualizes the radial and spatial distribution
9. Writes data and reconstruction parameters

Execution of reconstructions

Pseudo-algorithm:

For $n = 1$ to n_{max}

1. *extracts the first n points from the data set*
2. *executes the reconstruction of the PGF after the parameters provided by the previous procedure (it will generate 2 images)*
3. *discards the redundant images*

Execution of counts

One critical aspect of PGF for the reconstruction of the areas with highest probability is the use of a B parameter that can be as close as possible to the real B value (B_{exact}).

In our case we know B_{exact} , since we provided it during the procedure of generation of the points (on the map). For this reason, we executed a series of reconstructions based on the same data set, with B values corresponding to 90%, 80% and 60% of B_{exact} in order to verify how the use of incorrect values of B affect the identification of the spread centre and in which amount. For this purpose it is needed to investigate the shape and dimension of the red zone.

The choice of the above mentioned B values allows testing of B values that are not too far from the real value, since no objective criterion is currently known to define a range of “feasible” B values only on the base of the events distribution, while in the real world, the B value should be evaluated case by case by analysing the type of event and cause (for instance, the biology and propagation mode of an invading organism). For too high B values the buffer zone could enlarge so much to create a “border effect”, and then some of the points/events may localize outside the map and then be truncated. In this last case we would have an alteration of the results based on counting.

Currently the choice of B values is justified either on the basis of a priori knowledge of the spreading mechanism, the events and the cause (spreading centre) or on the basis of previous analogous studies. For this reason the choice of B appears to be crucial and can affect the feasibility of the PGF reconstruction. Further investigations will be necessary to provide insights on this last topics.


```

#           - vers 0.0.1 - initial working codes
#
# Instruction for bypass UTF-8 Exceptions
# encoding=utf8
import sys
reload(sys)
#
import random as rnd
import numpy as np
import time
import argparse
import math
import matplotlib.pyplot as plt

def Random_Angles(n_max, factor):
    #
    # function that generates a sequence of random angles
    #
    Angles=[]
    #
    # scale_factor=factor # unused here...
    #
    RAND=int(time.time())
    rnd.seed(RAND)
    i=0
    base=0.0
    while i<n_max :
        Angle=base+((rnd.uniform(0,2*math.pi)))
        Angles.append(Angle)
        base += Angle
        i += 1
    return Angles

def my_pareto(alpha,n_max, Max_Value):
    #
    # function that generates a vector of given length containing the
    values of the pareto distribution
    #
    i=0
    Values=[]
    while i<n_max:
        valore=(int(rnd.paretovariate(alpha)))
        if valore < Max_Value:
            Values.append(valore)
            i += 1
    return Values

def polar_2_rect_coords(Radius,Angle):
    #
    # function that transforms 2 values that express polar coordinates
    # of one point into two other values that express
    # the same point in Cartesian coordinates
    #
    x=Radius*math.sin(Angle)

```

```

    y=Radius*math.cos (Angle)
    return x,y

def vect_pol_2_rect (radiuses,Angles):
    #
    # function that transforms 2 vectors that
    # express polar coordinates of one point into two other
    # vectors that express the same point in Cartesian coordinates
    #
    coord_X=[]
    coord_Y=[]
    for i in range (len (radiuses)):
        temp1,temp2=polar_2_rect_coords (radiuses[i],Angles[i])
        coord_X.append(temp1)
        coord_Y.append(temp2)
    return coord_X,coord_Y

#
# ***** main *****
#
# initially it generates data in polar coordinates and then it will convert
# them to cartesian coordinates
# hence the radial distribution will be that of the Rossmo formula, while
# the radial one
# will be uniform, so to reach the maximal number of necessary hypotheses
# for the correct working
# of the method
#
# B is equal to Rossmo formula "B"
# alpha is Rossmo formula "f" & "g" ( assuming f and g are equals )
#
B = 100
alfa=0.4
n_max=12
#
# random variation amplitude
#
x0=28
Values_a=[]
Values_b=[]
Angles=[]
#
# init of random numbers generation
#
RAND=int (time.time ())
rnd.seed (RAND)
rng=rnd.getstate ()
rnd.randrange (0, x0)
Max_Value=45
#
#
# generate a pareto distribution for Values variing from zero to Max_Value
#
Values_a=my_pareto (alfa,n_max, Max_Value)
#

```



```

# generate a second pareto distribution for Values varying from zero to
Max_Value
#
Values_b=my_pareto(alfa,n_max, Max_Value)
#
# flip the distribution around simmetry axe x=1
#
for i in range(len(Values_b)):
    Values_b[i]=1-(Values_b[i])
#
# concatenate the lists Values_a and Values_b, so we have a two tiles
pareto distribution with max in x=1
#
Values=Values_a+Values_b
#
# Multiply for B the radius otherwise the maximum is in 1 ( normalized
distribution )
# to simulate the dead zone in polar coordinates

for i in range (len(Values)):
    Values[i]=Values[i]*B
#
# shuffle the vector values
#
np.random.shuffle(Values)
#
# create the vector containing the angles
# they are generated with a uniform distribution
#
scale_factor=3.0/47.0 #(ratio between 2 prime numbers, just to make sure
itâ€™s irrational)
Angles=Random_Angles((2*n_max),(math.pi*scale_factor))
#
# Polar to rectagnular coords conversion
#
Coord_X,Coord_Y=vect_pol_2_rect(Values,Angles)
#
# generate and plot distribution histogram
#
hist1,_=np.histogram(Values,2*n_max)
plt.figure()
#
# Instructions to construct the graph of the generated data
#
Range=np.linspace(hist1.min(), hist1.max(), 20)
Range_A=np.array(Range)
plt.subplot(2,1,1)
plt.title('Distribution of Points Distances From Center - Histogram ')
plt.hist(Values,bins=n_max)
plt.subplot(2,1,2)
plt.title('Spatial Distribution of Generated Points')
#plot raw data
plt.scatter(Coord_X,Coord_Y,marker=".",color="red")
#plot distribution center

```

```

plt.scatter(np.zeros(1),np.zeros(1),marker="*",color="black")
#print data labes
for i in range(len(Coord_X)):
    plt.text(Coord_X[i],Coord_Y[i], i,fontsize=9)
plt.text(0.3, 0.3, "Diffusion Center", fontsize=9)
plt.xlabel('X Coord')
plt.ylabel('Y Coord')
plt.show()
#
# I print the data to video, then I have to save data to csv
#
for i in range(len(Coord_X)):
    print i,";", int(Coord_X[i]),";",int(Coord_Y[i]), ";100"
print
#
# Saving data to csv - with semicolon as separator of values
#
#
Output_file_name="out_file.csv"
#
# surely not the most elegant way to do it, but I will improve it later
#
my_File = open(Output_file_name,â€wâ€)
for i in range(len(Coord_X)):
    my_string=str(i)+";"+str(int(Coord_X[i]))+";"+str(int(Coord_Y[i])) +
";100"
    my_File.write=(my_string)
print
my_File.close()
#
print "End"
#

```

Linear correlation between the curves obtained with the different B values

The linear correlation between the curves obtained with the different B values (correlation of data obtained with the simulation with the theoretical data obtained with the authors' provided parameters) produce the results shown in Table 2 (ANOVA with $p < 10^{-15}$):

Table 3S

Model Summary

Model	R	R ²	Adjusted R ²	RMSE
1	0.9816	0.9635	0.9618	24.8706

ANOVA

Model		Sum of Squares	df	Mean Square	F	p
1	Regression	343174.3334	1	343174.3334	554.8074	1.3913e -16
	Residual	12989.4823	21	618.5468		
	Total	356163.8157	22			

Coefficients

Model		Unstandardized	Standard Error	Standardized	t	p
1	(Intercept)	-11.4123	8.4353		-1.3529	0.1905
	Media_95_B110	1.3780	0.0585	0.9816	23.5544	1.391e -16

Table 3S. Ex table 2. Linear Regression between curves obtained with the different B values

Table 4S. Ex table 3. Bayesian Repeated Measures ANOVA

Model Comparison

Models	P(M)	P(M data)	BF_M	BF₁₀	error %
RM Factor 1	0.5000	0.9496	18.8511	1.0000	
Null model (incl. subject)	0.5000	0.0504	0.0530	0.0530	1.0701

Note. All models include subject.

Descriptives

Descriptives

RM Factor 1	Mean	SD	N
Level 1	145.2928	127.2370	23.0000
Level 2	113.7168	90.6332	23.0000