J. Comp. Int. Sci. (2017) 8(1):47-51
http://epacis.net/jcis/PDF_JCIS/JCIS-0121.pdf
jcis@epacis.net

Pan-american Association of Computational Interdisciplinary Sciences

pyGHS: Computing Geometric Histogram Separation in Binomial Proportion Patterns

R. Sautter[1] and P. H. Barchi.
Lab for Computing and Applied Mathematics (LABAC),
National Institute for Space Research (INPE), São José dos Campos, SP, Brazil
Received on January 30, 2017 / accepted on March 15, 2017

Abstract

This is a short technical report on pyGHS, a code designed in python to calculate the geometric separation between two histograms that represent a pattern of binomial proportion.

Keywords: Computational Discriminant Analysis, Histogram separation.

1. Program summary

- *Program title:* pyGHS (*python Geometric Histogram Separation*)
- *Program URL:* https://github.com/GFEC-INPE/pyGHS
- *Licensing provisions:* GNU General Public License.
- *Hardware for which the program has been tested:* Intel Core I7 950 Quad-Core, 1GB.
- *Operating systems:* Linux, Mac OS X.
- *Programming language:* Python.
- *Has the code been vectorized or parallelized?:* No
- *Nature of the problem:* The challenge in Computational Discriminant Analysis [1] is to project a dataset onto a lower-dimensional space with good class-separability and also reduce computational costs. Although there are a variety of techniques to differentiate binomial proportions, it still lacks more efficient approaches. Such need comes from the fact that discrete histograms from real data are not always statistically well defined [2, 3].
- *Method of solution:* Recently, based on the operation of the box counting algorithm, the measure of the δ_{GHS} separation parameter was proposed based only on the geometric properties of the histograms [4]. It is a measure calculated straightforward on the histograms files that maximizes the mutual separation considering the quantitative balance between areas and heights for typical binomial patterns.
- *Typical running time:* 0.048 seconds for each 1MB (data input).
- *Unusual features of the program:* None.
- *Restrictions*: Pandas driver should be installed.

[1]E-mail Corresponding Author: rubens.sautter@gmail.com

```python
import numpy
import pandas as pd
from math import sqrt
import sys

def ghs(data1,data2):
        '''
        GHS - Geometric Histogram Separation
        ----------------------------------------------
        Input:
        data1,data2 - The data samples - list or numpy.ndarray

        Output:
        BCA, BCL, GHS - The metrics - tuple of float64
        ----------------------------------------------
        This function measures the Box Counting Area (BCA) and
        the Box Counting Linear (BCL) between data1 and data2.
        The GHS is measured as average of BCA square root and BCL.
        '''
        hist1, bins1 = numpy.histogram(data1)
        hist2, bins2 = numpy.histogram(data2)
        both = numpy.concatenate((bins1,bins2))
        # n is the number of bins (average between the number of both bins)
        n = len(both)/2
        rnge = (numpy.min(both),numpy.max(both))
        hist1, bins1 = numpy.histogram(data1,bins=n,range=rnge,normed=True)
        hist2, bins2 = numpy.histogram(data2,bins=n,range=rnge,normed=True)

        # since both histograms have same bins, dy is the intersection:
        dx = (rnge[1]-rnge[0])/n
        dy = numpy.minimum(hist1,hist2)

        # ao is the relative area
        ao = numpy.sum(dy*dx)

        a_height = numpy.max(hist1)
        b_height = numpy.max(hist2)
        c_height = numpy.max(dy)

        bcl = (a_height+b_height-2.0*c_height)/(a_height+b_height)
        bca = 1.0 - (ao) / (2.0 - ao)
        ghs = ( sqrt(bca) + bcl ) / 2.0
        return bca,bcl,ghs

if __name__ == "__main__":
        '''
        Call structres:
                python ghs.py data1.csv data2.csv parameter
                python ghs.py data1.csv data2.csv
        '''
        d1 = pd.read_csv(sys.argv[1]).dropna()
        d2 = pd.read_csv(sys.argv[2]).dropna()
        if len(sys.argv)== 4:
                bca,bcl,ghs = ghs(d1[sys.argv[3]], d2[sys.argv[3]])
        else:
                bca,bcl,ghs = ghs(d1, d2)
        print("BCA:"+str(bca)+"\nBCL:"+str(bcl)+"\nGHS:"+str(ghs))
```

Figure 1: The pyGHS code.

2. Program Input Description

The pyGHS is a sequential and small program (see Figure 1) which has as input: $data_1$ and $data_2$ representing the respective histograms previously identified and visualized by the user.

Six typical patterns of binomial proportions are shown in Figure 1. These data were generated to represent samples compatible with data from scientific experiments. We sampled 5.000 points of the Gaussian and Gamma distributions using scipy [5]. Taking into account the balance between areas and heights, the patterns are arranged from the smallest to the largest separation between the histograms of each binomial proportion.

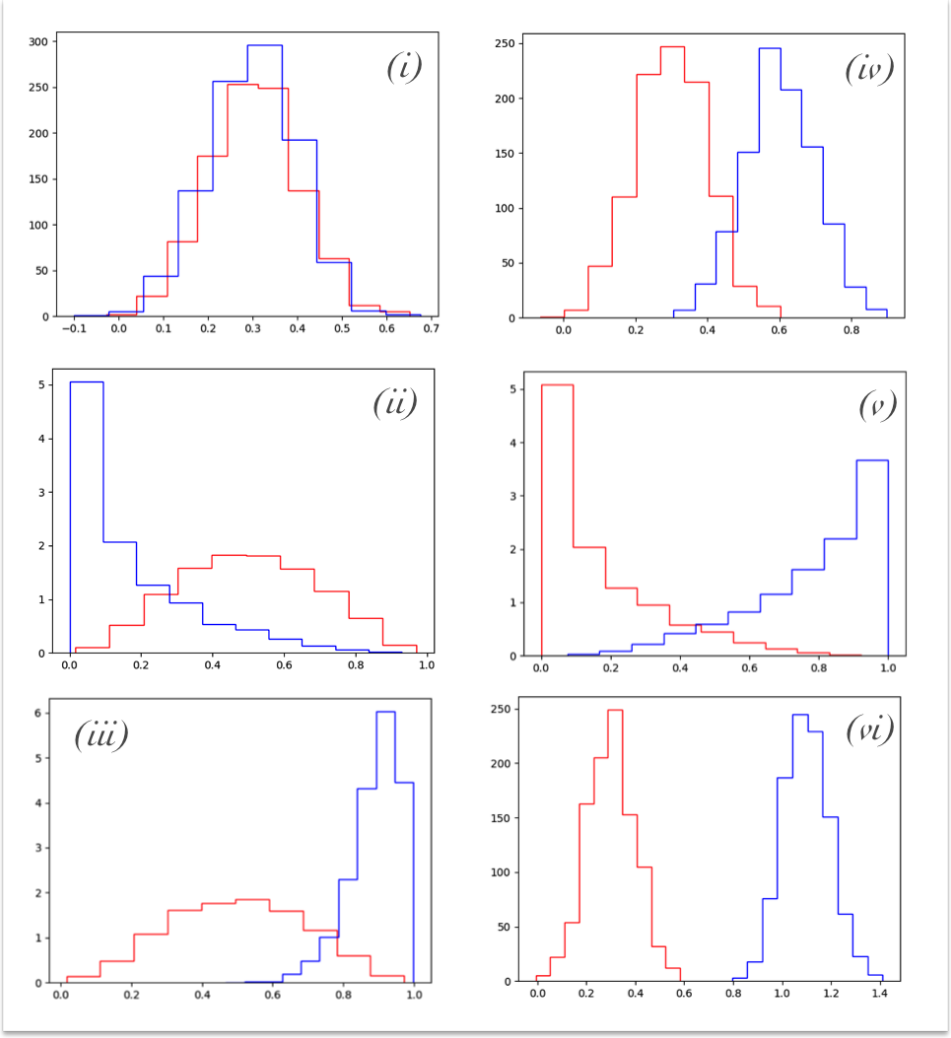

Figure 2: Six typical patterns of binomial proportions: (i) Symmetric-Symmetric-High-superposition; (ii) Left-Skewed-Symmetric-medium superposition; (iii) Right-Skewed-Symmetric-Low Superposition; (iv) Symmetric-Symmetric-Low-superposition; (v) Symmetric-Skewed-Low-Superposition; (vi) Symmetric-Symmetric-No-Superposition.

3. Program Output Description

The output of the program provides the following parameters: δ_{BCA}, δ_{BCL} and δ_{GHS}.

From the input data these parameters are calculated as follows:

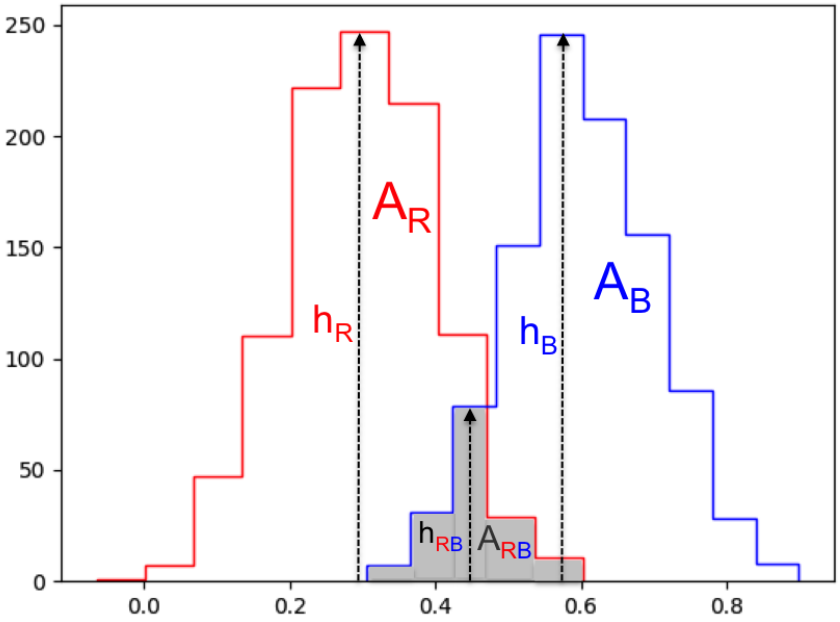

Figure 3: The areas and their respective heights from the binomial histogram pattern are: A_B (blue histogram area), A_R (red histogram area), A_{BR} (intersection area between A_B and A_R) and heights: h_B, h_R, and h_{BR}

1. The areas (amount of bins) covered by each histogram are computed according to the variables: A_R (red histogram area), A_B (blue histogram area), A_{BR} (intersection area between A_B and A_R), as shown in Figure 2. Then, the output is computed as

$$\delta_{BCA} = 1 - \frac{A_{BR}}{A_B + A_R + A_{BR}}. \tag{1}$$

   Note that, δ_{BCA} is a normalized measure which computes the component of the histograms separation based on the area superposition. When A_{BR} is null $\delta_{BCA} = 1$ (totally separated). When the total area is A_{BR} then $\delta_{BCA} = 0$ (total superposition).

2. The heights for each histogram are computed according to the variables: h_B, h_R, and h_{BR} as shown in Figure 2. Then, the output is computed as

$$\delta_{BCL} = \frac{h_a + h_b - 2h_c}{h_a + h_b}. \tag{2}$$

   Note that, δ_{BCL} is a normalized measure which computes the component of the histograms separation based on the height of superposition balanced with the largest distance between the maxima values for h_B and h_R. When h_c is null $\delta_{BCL} = 1$. When $h_B = h_R = h_{BR}$ then $\delta_{BCL} = 0$.

3. Then, considering the previous calculations, the output value performing the Geometric Histogram Separation is computed as

$$\delta_{GHS} = \frac{\sqrt{\delta_{BCA}} + \delta_{BCL}}{2}. \tag{3}$$

   The respective values to the histograms showed in Figure 2 are shown in Table 1.

Pattern	δ_{BCA}	δ_{BCL}	δ_{GHS}
(i)	0.060	0.002	0.124
(ii)	0.786	0.710	0.798
(iii)	0.913	0.748	0.851
(iv)	0.935	0.817	0.892
(v)	0.913	0.898	0.927
(vi)	1.000	1.000	1.000

References

[1] Tao Li, Shenghuo Zhu, and Mitsunori Ogihara. Using Discriminant Analysis for Multi-Class Classification: An Experimental Investigation. Knowledge and Information Systems 10, no. 4 (2006): 45372.)

[2] A Probabilistic Theory of Pattern Recognition Por Luc Devroye,Laszlo Gyrfi,Gabor Lugosi, Springer 1991.

[3] Duda, Richard O, Peter E Hart, and David G Stork. 2001. Pattern Classification. New York: Wiley.

[4] Rosa, R.R., Sautter, R.; Barchi, P.H., Carvalho, R.R., *Binomial histogram separation in computational discriminant analysis inspired by the box-counting algorithm*, in submission to Computer Physics Communication (2017).

[5] Jones, E., Oliphant, T., Peterson, P., Open source scientific tools for *Python*, 2017.