

Program for Optimal Linear Ordering (POLO)

Diansheng Guo

Department of Geography
University of South Carolina

Introduction

Linear ordering is to transform a set of multivariate data objects to a one dimensional order, which preserves multivariate patterns (similarities) as much possible. Linear ordering is widely used in visualization research to accentuate patterns. When applied in geographical analysis, a linear ordering can be used to preserve spatial or multivariate patterns and facilitate the detection of space-time or space-multivariate relationships.

This program implements a linear ordering method based on the complete linkage hierarchical clustering and an optimal cluster ordering method. The method is named CLO_OPT, which outperforms other existing ordering methods (see the evaluation provided in (Guo and Gahegan 2006)). Such an ordering method can help in a variety of geographic analysis problems, such as multivariate spatial clustering, spatio-temporal visualization (Guo et al. 2006), and spatial interaction analysis (Guo 2007).

References:

- Guo, D. (2007). "Visual Analytics of Spatial Interaction Patterns for Pandemic Decision Support", International Journal of Geographical Information Science, **21**(8), pp. 859-877.
- Guo, D. and M. Gahegan (2006). "Spatial Ordering and Encoding for Geographic Data Mining and Visualization", Journal of Intelligent Information Systems, **27**(3), pp.243-266.
- Guo, D., J. Chen, A. M. MacEachren, and K. Liao (2006), "A Visualization System for Spatio-Temporal and Multivariate Patterns (VIS-STAMP)", IEEE Transactions on Visualization and Computer Graphics, **12**(6), pp. 1461-1474.

Java

To run this program, it is recommended to have the latest version of Java installed on your machine. You can visit <http://java.com/en/download/index.jsp> and test if you have the latest version. .

Then you simply run the **linearordering.jar** file using one of the following options:

- Double click the jar file;
- Right click → open with... → Java ...; or
- Execute in a command window with "**java -jar linearordering.jar**".

(If for some reason, your downloading software actually saves the file as “linearordering.ZIP”, you need to rename the file back to “linearordering.JAR” before running it using one of the above options.)

Input Data Format:

The input data should be in a CSV format, with **attribute names** on the first line, **attribute weights** on the second line, and the **actual data values** starting on the third line (see Figure 1 or the example data file accompanying this manual).

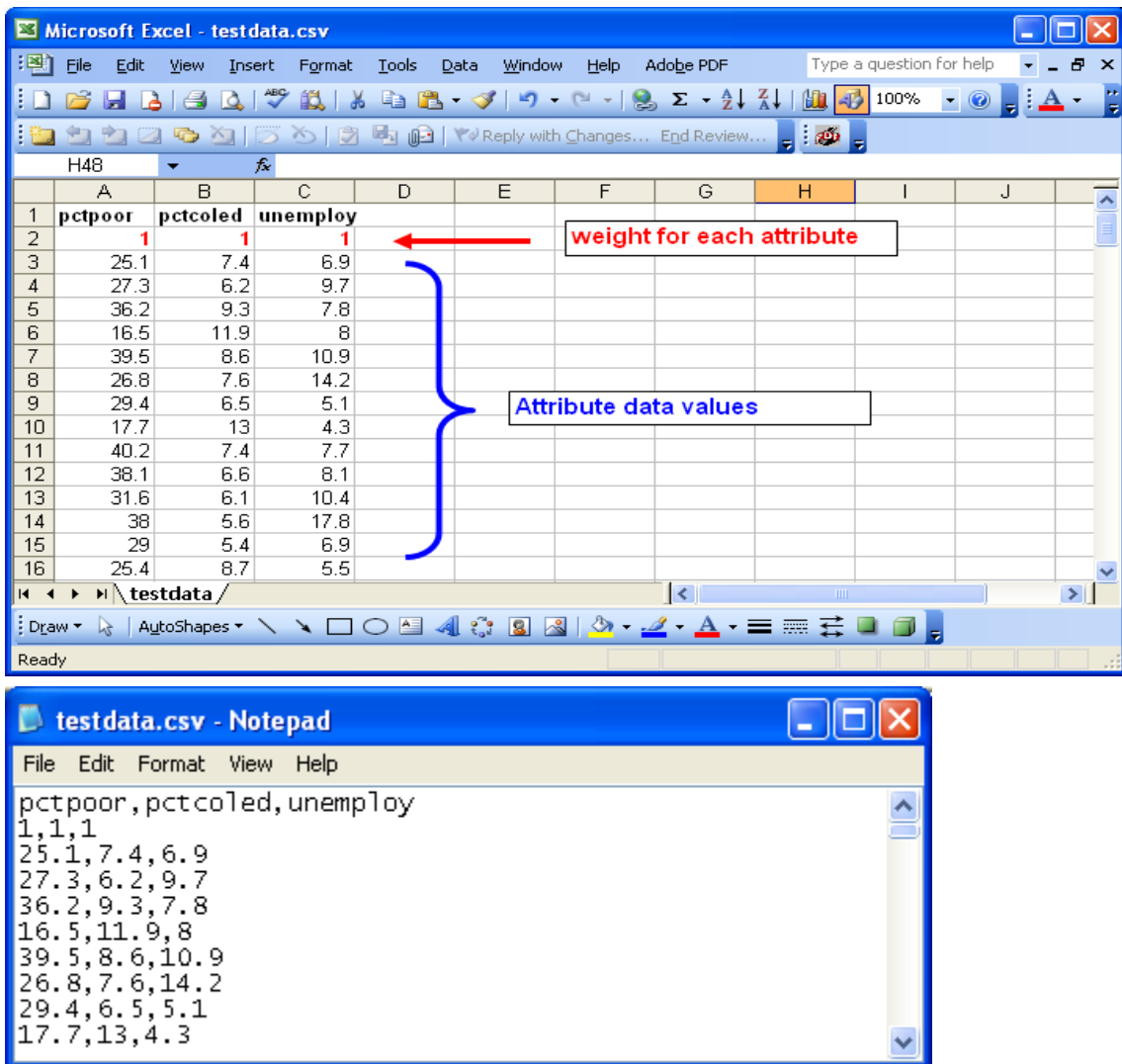


Figure 1: Input data format.

Run the program

When you start the program, it will ask for the data file (in CSV format). The program assumes that all columns (attributes) in the data file will be used to derive the ordering. Once the ordering is derived done, the result will be saved in a new file (in the same folder where loaded the input data file). The program will show you a dialog (see below) to provide a brief report on the result.



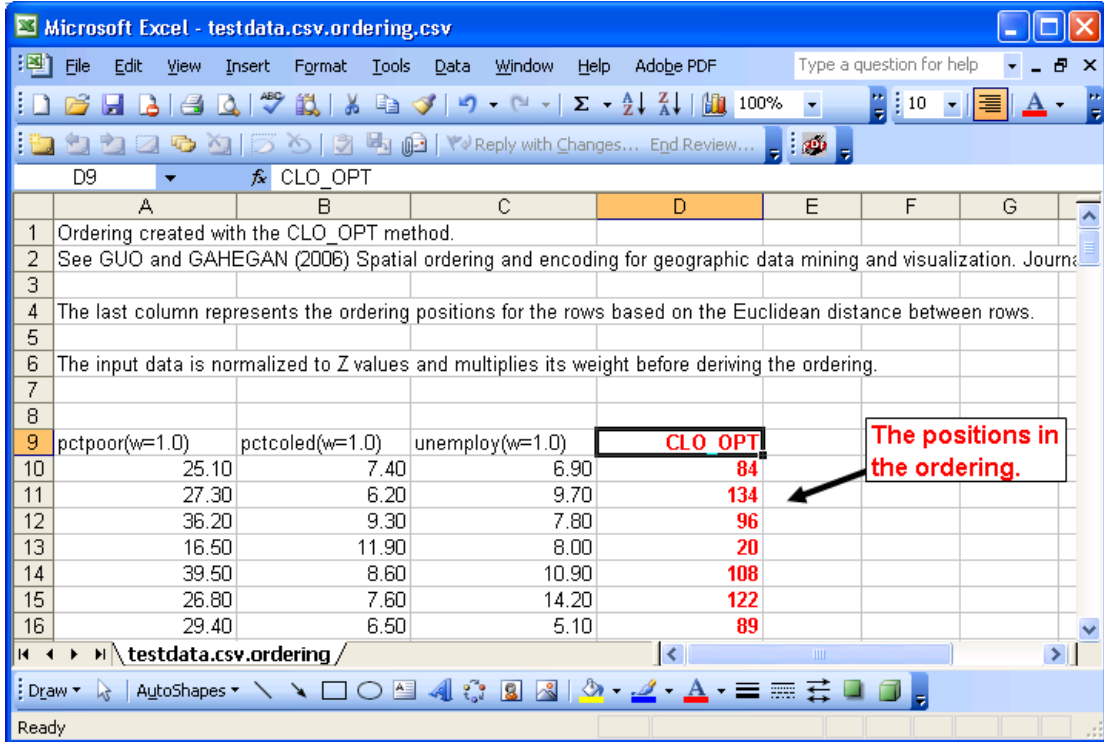
RAM Memory Complexity

The program uses $O(n^2)$ memory, due to the pair-wise similarity matrix. If the data set is too large (e.g, $n > 1000$), the user needs to specify the maximum memory to allocate for Java, by using the following command to start the program:

```
java -jar -Xmx1024m linearordering.jar
```

The above command requests 1G RAM memory to be allocated for running the program. This, of course, requires that the computer should have about 2G RAM memory. With the above command, the program can process and order about 3000-5000 data objects. The more memory is allocated, the larger dataset it can process.

Ordering Result:



Microsoft Excel - testdata.csv.ordering.csv

File Edit View Insert Format Tools Data Window Help Adobe PDF Type a question for help

100% 10

D9 CLO_OPT

	A	B	C	D	E	F	G
1	Ordering created with the CLO_OPT method.						
2	See GUO and GAHEGAN (2006) Spatial ordering and encoding for geographic data mining and visualization. Journ						
3							
4	The last column represents the ordering positions for the rows based on the Euclidean distance between rows.						
5							
6	The input data is normalized to Z values and multiplies its weight before deriving the ordering.						
7							
8							
9	pctpoor(w=1.0)	pctcoled(w=1.0)	unemploy(w=1.0)	CLO OPT			
10	25.10	7.40	6.90	84			
11	27.30	6.20	9.70	134			
12	36.20	9.30	7.80	96			
13	16.50	11.90	8.00	20			
14	39.50	8.60	10.90	108			
15	26.80	7.60	14.20	122			
16	29.40	6.50	5.10	89			

testdata.csv.ordering/

Draw AutoShapes

Ready