The Stata Journal (2004)
4, Number 4, pp. 361–378

Simple thematic mapping

Maurizio Pisati
Department of Sociology and Social Research,
Università degli Studi di Milano-Bicocca, Italy
maurizio.pisati@unimib.it

Abstract. Thematic maps illustrate the spatial distribution of one or more variables of interest within a given geographic unit. In a sense, a thematic map is the spatial analyst's equivalent to the scatterplot in nonspatial analysis. This paper presents the `tmap` package, a set of Stata programs designed to draw five kinds of thematic maps: choropleth, proportional symbol, deviation, dot, and label maps. The first three kinds of maps are intended to depict area data, the fourth is suitable for representing point data, and the fifth can be used to show data of both types.

Keywords: gr0008, tmap choropleth, tmap propsymbol, tmap deviation, tmap dot, tmap label, thematic map, choropleth map, proportional symbol map, deviation map, dot map, label map

1 Introduction

Thematic maps represent the spatial distribution of one or more variables of interest within a given geographical unit (Monmonier 1993; Kraak and Ormeling 1996). For example, a sociologist could use a choropleth map (a.k.a., shaded map) to show how the percent of families below the poverty line varies across the states or the provinces of a given country. In turn, a police officer could be interested in analyzing a dot map showing the locations of drug markets within a given city. As Bailey and Gatrell (1995) have put it, in a sense, a thematic map is the spatial analyst's equivalent to the scatter plot in nonspatial analysis.

Usually to produce state-of-the-art thematic maps, one has to resort to specialized software. In some cases, it is possible to exploit the graphical engine of a general-purpose statistical package to draw simple but effective thematic maps. Up until version 7, Stata offered very limited mapping capabilities (see, for instance, Pisati 2001). The brand-new graphics engine introduced in Stata 8, however, is quite flexible and makes it possible to draw several kinds of maps in a relatively simple manner.

The purpose of this paper is to present the `tmap` package, a set of Stata programs designed to draw five kinds of thematic map: choropleth, proportional symbol, deviation, dot, and label maps. The first three kinds of maps are intended to depict area data, the fourth is suitable for representing point data, and the fifth can be used to show data of both types.

`tmap` exploits the possibility offered by the new Stata graphics to overlay a large number of different graphs, each of which is used to create a distinct element of the

desired map. Specifically, `graph twoway area` is used to draw the outlines of the geographic areas of interest and to fill them with the appropriate colors, while `graph twoway scatter` is used to plot symbols or labels as required.

All the programs included in the `tmap` package require Stata version 8.2 or later.

2 The tmap package

2.1 Syntax

`tmap choropleth` *quantvar* [`if` *exp*] [`in` *range*], `id(`*varname*`) map(`*filename*`)`
[`clmethod(quantile|eqint|stdev|custom|unique) clnumber(`#`)`
`clbreaks(`*numlist*`) eirange(`*numlist*`) palette(`*colorscheme*`)`
`colors(`*colorstyle_list*`) ocolor(`*colorstyle*`) osize(`*linewidthstyle*`) legpos(`#`)`
`legtitle(`*string*`) legformat(`*format*`) legbox nolegend`]

`tmap propsymbol` *quantvar* [`if` *exp*] [`in` *range*], `xcoord(`*varname*`)`
`ycoord(`*varname*`) map(`*filename*`)` [`scolor(`*colorstyle*`) sshape(`*symbolstyle*`)`
`ssize(`#`) ocolor(`*colorstyle*`) osize(`*linewidthstyle*`) fcolor(`*colorstyle*`)`]

`tmap deviation` *quantvar* [`if` *exp*] [`in` *range*], `xcoord(`*varname*`)`
`ycoord(`*varname*`) map(`*filename*`)` [`center(mean|median) scolor(`*colorstyle*`)`
`sshape(`*symbolstyle*`) ssize(`#`) ocolor(`*colorstyle*`) osize(`*linewidthstyle*`)`
`fcolor(`*colorstyle*`)`]

`tmap dot` [`if` *exp*] [`in` *range*], `xcoord(`*varname*`) ycoord(`*varname*`)`
`map(`*filename*`)` [`by(`*varname*`) marker(color|shape) scolor(`*colorstyle_list*`)`
`sshape(`*symbolstyle_list*`) ssize(`#`) ocolor(`*colorstyle*`) osize(`*linewidthstyle*`)`
`fcolor(`*colorstyle*`) legpos(`#`) legtitle(`*string*`) legbox nolegend`]

`tmap label` *labvar* [`if` *exp*] [`in` *range*], `xcoord(`*varname*`) ycoord(`*varname*`)`
`map(`*filename*`)` [`lcolor(`*colorstyle*`) lsize(`#`) llength(`#`) ocolor(`*colorstyle*`)`
`osize(`*linewidthstyle*`) fcolor(`*colorstyle*`)`]

2.2 Description

Let R denote the geographical unit object of analysis (e.g., a country, a region, a city). The programs that make up the `tmap` package can be used to represent the spatial distribution within R of two types of data:

- *Area data*: in this case, we are interested in showing how the value taken on by a given quantitative variable *quantvar* (e.g., the percent of active population, the number of homicides per 10,000 residents, the average family income) varies across a set of n mutually exclusive subareas of R (e.g., states, provinces, districts). Let $A_i (i = 1, \ldots, n)$ denote each of these subareas and P_i denote the corresponding polygons—the ordered sets of x, y coordinate pairs that define the outline (i.e., the geographic boundaries) of each subarea A_i.

- *Point data*: in this case, the phenomenon we want to depict is represented by a set of n point locations $s_i (i = 1, \ldots, n)$, belonging to the geographical unit R, at which some "events" of interest (e.g., residential burglaries, car accidents, Catholic churches) have occurred. Each point location is defined by a pair of (x, y) coordinates and, possibly, by the value of a categorical variable denoting the type of event.

`tmap choropleth` represents the spatial distribution of area data by means of choropleth maps in which each subarea A_i is colored or shaded according to a discrete scale based on its value on the variable of interest, *quantvar* (Bailey and Gatrell 1995). The number of classes k that make up the discrete scale and the corresponding class breaks can be based on several different criteria. In `tmap choropleth`, the number of classes must be between 2 and 9, i.e., $2 \leq k \leq 9$. The class breaks can be determined by means of one of the following four methods:

1. *Quantiles*: class breaks correspond to quantiles of the distribution of *quantvar* so that each class includes approximately the same number of subareas.

2. *Equal intervals*: class breaks correspond to values that divide the distribution of *quantvar* into k equal-width intervals.

3. *Standard deviates*: the distribution of *quantvar* is divided into k classes whose width is defined as a fraction, p, of its standard deviation, sd. Following the suggestions of Evans (1977), this proportion p varies with k as follows: $k = 2$, $p = \infty$; $k = 3$, $p = 1.2$; $k = 4$, $p = 1$; $k = 5$, 6, or 7, $p = 0.8$; $k = 8$ or 9, $p = 0.6$. Class intervals are centered on the arithmetic mean, m, which is a class midpoint if k is odd and a class boundary if k is even; the lowest and highest classes are open-ended. For example, if $k = 5$, then the corresponding class intervals are defined as follows: $[-\infty, m - 1.2\text{sd}]$, $(m - 1.2\text{sd}, m - 0.4\text{sd}]$, $(m - 0.4\text{sd}, m + 0.4\text{sd}]$, $(m + 0.4\text{sd}, m + 1.2\text{sd}]$, and $(m + 1.2\text{sd}, \infty]$. It should be noted that, in any given dataset, one or more classes might be empty for lack of observations.

4. *Custom*: class breaks are specified by the user.

In addition, `tmap choropleth` allows the user to represent the spatial distribution of a categorical variable, provided it takes on a maximum of nine different values.

Users should be aware that choropleth maps can be misleading. In particular, physically large areas tend to dominate the display in a way that may be unsuitable for some kind of data (Bailey and Gatrell 1995). For example, when mapping data on political

participation, the large and sparsely populated rural areas may tend to dominate the choropleth map at the expense of the physically smaller and more densely populated urban areas, regardless of the distribution of the phenomenon of interest. To avoid this problem, one may use proportional symbol maps described below.

`tmap propsymbol` represents the spatial distribution of area data by means of proportional symbol maps, i.e., maps where the value taken on by the variable of interest, *quantvar*, in each subarea A_i is represented by a symbol whose size is proportional to the value itself (Bailey and Gatrell 1995). The symbols are superimposed onto a base map depicting the whole geographical unit R or its detailed subareas.

`tmap deviation` represents the spatial distribution of area data by means of deviation maps, a particular kind of proportional symbol map. In the first place, the values taken on by the variable of interest, *quantvar*, are re-expressed as deviations from the center, m, of their distribution, defined as either the mean or the median of *quantvar*, so that (say) $D = quantvar - m$. The value of D in each subarea is then represented by a symbol whose size is proportional to the value itself expressed in absolute terms and whose filling reflects the sign of the value—positive values are represented by solid symbols and negative values by hollow symbols of the same shape. Users should be aware that when the number of subareas is large relative to the size of the whole geographical unit of interest R, it may become difficult to distinguish between solid and hollow symbols. In these cases, use of `tmap deviation` should be avoided.

`tmap dot` represents the spatial distribution of point data by means of dot maps, i.e., maps where the locations s_i at which some "events" of interest have occurred are indicated by symbols whose color or shape can vary according to the type of event (Bailey and Gatrell 1995).

`tmap label` is an auxiliary program that allows the user to superimpose onto a base map the values taken on by a numeric or string variable, *labvar*, at different locations l_i of R. This program can be used, for example, to plot subarea names or to represent the spatial distribution of a given quantitative variable of interest in numeric form.

All the programs included in the `tmap` package require that the geographic boundaries of the whole geographical unit R or of its subareas A_i be stored in an external file. This file must always include the following three variables: `_ID`, which contains the numeric identifier of the polygons representing R or its subareas A_i; `_X`, which contains the x-coordinates of the polygons; and `_Y`, which contains the y-coordinates of the polygons. The coordinates of each polygon must be arranged so as to correspond to consecutive nodes. Moreover, each polygon must be closed; i.e., the last pair of coordinates of each polygon must be equal to the first pair.

2.3 Common options

`map(`*filename*`)` is required. It specifies the name of the file containing the information needed to draw the base map. In `tmap choropleth`, *filename* must contain the coordinates of the polygons P_i representing the different subareas A_i of the geographical

unit of interest R. In all the other cases, *filename* can contain either the coordinates of the polygons representing the different subareas A_i, or the coordinates of the polygons representing the whole geographical unit of interest R. *filename* must follow the format described in the previous section.

`ocolor(`*colorstyle*`)` specifies the outline color of the polygons making up the base map (see [G] ***colorstyle***). The default is `ocolor(black)`.

`osize(`*linewidthstyle*`)` specifies the outline thickness of the polygons making up the base map (see [G] ***linewidthstyle***). The default is `osize(thin)`.

`fcolor(`*colorstyle*`)` (available with all commands but `tmap choropleth`) specifies the fill color of the polygons making up the base map (see [G] ***colorstyle***). The default is `fcolor(white)`.

2.4 Options for tmap choropleth

`id(`*varname*`)` is required. It specifies the name of the numeric variable that uniquely identifies the different subareas A_i of the geographical unit of interest R. The values taken on by *varname* must correspond to the values taken on by the identifier `_ID` contained in the file specified with option `map(`*filename*`)`.

`clmethod(quantile|eqint|stdev|custom|unique)` specifies the method to be used for determining the class breaks.

`clmethod(quantile)` is the default and requests that the quantiles method be used.

`clmethod(eqint)` requests that the equal intervals method be used.

`clmethod(stdev)` requests that the standard deviates method be used.

`clmethod(custom)` requests that class breaks be specified by the user with option `clbreaks(`*numlist*`)`.

`clmethod(unique)` requests that the variable of interest *quantvar* be treated as a categorical variable taking on a maximum of nine different values.

`clnumber(#)` specifies the number of classes k in which the variable of interest *quantvar* should be divided. This option accepts only numbers between 2 and 9. The default is `clnumber(4)`.

`clbreaks(`*numlist*`)` is required if option `clmethod(custom)` is specified. It specifies a list of numbers defined as follows: the first element of the list is the minimum value of *quantvar* to be considered; the second to kth elements of the list are the class breaks; and the last element of the list is the maximum value of *quantvar* to be considered. For example, suppose that we want to divide the values of *quantvar* into the following four classes: $[10, 15]$, $(15, 20]$, $(20, 25]$, and $(25, 50]$; for this we must specify `clbreaks(10 15 20 25 50)`.

`eirange(`*numlist*`)` specifies the range of values (minimum and maximum) to be considered in the calculation of class breaks when option `clmethod(eqint)` is specified. This option overrides the default range $\big[\min(quantvar),\max(quantvar)\big]$.

`palette(`*colorscheme*`)` specifies the color scheme to be used for representing the different classes in which *quantvar* has been divided. *colorscheme* is one of the following:

`Blues`	`BrBG`	`Greens`	`Greys`
`Paired`	`PuRd`	`Purples`	`RdBu`
`RdGy`	`Reds`	`Set1`	`Set3`
`YlOrBr`	`Custom`		

The default is `palette(Greys)` when `clmethod(quantile)` or `clmethod(eqint)` is specified; `palette(RdBu)` when `clmethod(stdev)` is specified; and `palette(Paired)` when `clmethod(unique)` is specified. If option `palette(Custom)` is specified, option `colors(`*colorstyle_list*`)` must be specified as well.

`colors(`*colorstyle_list*`)` specifies a custom list of colors to be used for representing the different classes in which *quantvar* has been divided (see [G] ***colorstyle***). The number of elements of the list must equal k, i.e., the desired number of classes. `colors()` is only effective if used with `palette(Custom)`.

`legpos(#)` specifies the position of the map legend (see [G] ***legend_option***). The default is `legpos(7)`.

`legtitle(`*string*`)` specifies the title of the map legend. By default, no title is used.

`legformat(`*format*`)` specifies the format of the numeric values appearing in the map legend (see [U] **15.5 Formats: controlling how data are displayed**). The default is `legformat(%8.2f)`.

`legbox` requests that a box be drawn around the map legend.

`nolegend` requests that the map legend be suppressed.

2.5 Options for tmap propsymbol

`xcoord(`*varname*`)` is required. It specifies the name of the variable containing the x-coordinate of the centroid of each subarea A_i. *varname* must be expressed in the same units as the x-coordinates of the polygons making up the base map specified with option `map(`*filename*`)`.

`ycoord(`*varname*`)` is required. It specifies the name of the variable containing the y-coordinate of the centroid of each subarea A_i. *varname* must be expressed in the same units as the y-coordinates of the polygons making up the base map specified with option `map(`*filename*`)`.

`scolor(`*colorstyle*`)` specifies the color of the symbols (see [G] ***colorstyle***). The default is `scolor(black)`.

`sshape(`*symbolstyle*`)` specifies the shape of the symbols (see [G] ***symbolstyle***). The default is `sshape(Oh)`, i.e., a hollow circle.

`ssize(#)` specifies a multiplier that affects the size of the symbols. For example, to increase the size of all the symbols by 50%, specify `ssize(1.5)`. The default is `ssize(1)`.

2.6 Options for tmap deviation

`xcoord(`*varname*`)` is required. It specifies the name of the variable containing the x-coordinate of the centroid of each subarea. *varname* must be expressed in the same units as the x-coordinates of the polygons making up the base map specified with option `map(`*filename*`)`.

`ycoord(`*varname*`)` is required. It specifies the name of the variable containing the y-coordinate of the centroid of each subarea. *varname* must be expressed in the same units as the y-coordinates of the polygons making up the base map specified with option `map(`*filename*`)`.

`center(mean|median)` specifies the center of the distribution of *quantvar* to be taken as the reference value. `center(mean)` is the default requesting that the reference value be the arithmetic mean of *quantvar*. `center(median)` requests that the reference value be the median of *quantvar*.

`scolor(`*colorstyle*`)` specifies the color of the symbols (see [G] ***colorstyle***). The default is `scolor(black)`.

`sshape(`*symbolstyle*`)` specifies the shape of the symbols (see [G] ***symbolstyle***). For the reasons explained in section 2.2, this option accepts only solid symbolstyles, expressly `O D T S o d t s`. The default is `sshape(O)`, i.e., a circle.

`ssize(#)` specifies a multiplier that affects the size of the symbols. For example, to increase the size of all the symbols by 50%, specify `ssize(1.5)`. The default is `ssize(1)`.

2.7 Options for tmap dot

`xcoord(`*varname*`)` is required. It specifies the name of the variable containing the x-coordinates of the locations at which the events of interest have occurred. *varname* must be expressed in the same units as the x-coordinates of the polygons making up the base map specified with option `map(`*filename*`)`.

`ycoord(`*varname*`)` is required. It specifies the name of the variable containing the y-coordinates of the locations at which the events of interest have occurred. *varname* must be expressed in the same units as the y-coordinates of the polygons making up the base map specified with option `map(`*filename*`)`.

`by(`*varname*`)` specifies the name of a categorical variable denoting the type of event that occurred at each location. Although the program does not impose any restriction, it is advisable that *varname* take a maximum of nine different values.

`marker(color|shape)`, when `by(`*varname*`)` is specified, specifies whether the different types of events should be indicated by symbols having the same shape but different colors, or by symbols having the same color but different shapes. `marker(color)` is the default and requests that the different types of events be indicated by symbols having the same shape but different colors. `marker(shape)` requests that the different types of events be indicated by symbols having the same color but different shapes.

`scolor(`*colorstyle_list*`)` specifies the colors of the symbols (see [G] ***colorstyle***). When `by(`*varname*`)` is not specified or is specified along with `marker(shape)`, the default is `scolor(black)`. When `by(`*varname*`)` is specified along with `marker(color)`, the default is `scolor(black red blue green orange ltblue lime sienna yellow)`.

`sshape(`*symbolstyle_list*`)` specifies the shapes of the symbols (see [G] ***symbolstyle***). When `by(`*varname*`)` is not specified or is specified along with `marker(color)`, the default is `sshape(o)`. When `by(`*varname*`)` is specified along with `marker(shape)`, the default is `sshape(o oh s sh t th d dh x)`.

`ssize(#)` specifies a multiplier that affects the size of the symbols. For example, to increase the size of all the symbols by 50%, specify `ssize(1.5)`. The default is `ssize(1)`.

`legpos(#)` specifies the position of the map legend (see [G] ***legend_option***). The default is `legpos(7)`.

`legtitle(`*string*`)` specifies the title of the map legend. By default, no title is used.

`legbox` requests that a box be drawn around the map legend.

`nolegend` requests that the map legend be suppressed.

2.8 Options for tmap label

`xcoord(`*varname*`)` is required. It specifies the name of the variable containing the x-coordinates of the locations at which the labels of interest should be plotted. *varname* must be expressed in the same units as the x-coordinates of the polygons making up the base map specified with option `map(`*filename*`)`.

`ycoord(`*varname*`)` is required. It specifies the name of the variable containing the y-coordinates of the locations at which the labels of interest should be plotted. *varname* must be expressed in the same units as the y-coordinates of the polygons making up the base map specified with option `map(`*filename*`)`.

`lcolor(`*colorstyle*`)` specifies the color of the labels (see [G] ***colorstyle***). The default is `lcolor(black)`.

`lsize(#)` specifies a multiplier that affects the size of the labels. For example, to increase the size of all the labels by 50%, specify `lsize(1.5)`. The default is `lsize(1)`.

`llength(#)` specifies the maximum number of characters of the labels to be displayed. The default is `llength(12)`.

3 Examples

For illustrative purposes, I will take the city of Milano, Italy, as the geographical unit of interest. First, let us analyze the spatial distribution of some area data, namely, the distribution of the percent foreign population (2001) across the 20 former administrative districts of Milano.

```
. use milano-areadata, clear
. describe
Contains data from milano-areadata.dta
  obs:            20
 vars:             6                          9 Nov 2004 08:13
 size:           380 (99.9% of memory free)
```

variable name	storage type	display format	value label	variable label
id	byte	%9.0g		District number
district	byte	%40.0f	zona	District name
group	byte	%2.0f		District group
x	float	%9.0g		x-coordinate of district centroid
y	float	%9.0g		y-coordinate of district centroid
foreign01	float	%5.1f		% foreign pop. (2001)

```
Sorted by:  id
. list id district foreign01, noobs sep(0)
```

id	district	forei~01
1	Centro Storico	11.8
2	Centro Direzionale-Greco-Zara	11.6
3	Venezia-Buenos Ayres	12.2
4	Vittoria-Romana-Molise	9.3
5	Ticinese-Genova	9.8
6	Magenta-Sempione	10.1
7	Bovisa-Dergano	13.7
8	Affori-Bruzzano-Comasina	6.6
9	Niguarda-CàGranda-Bicocca	4.6
10	Monza-Padova	12.1
11	Centro Studi-Argonne	9.9
12	Feltre-Carnia-Cimiano-Ortica-Lambrate	8.7
13	Forlanini-Taliedo	6.6
14	Corvetto-Rogoredo-Vigentina	9.6
15	Chiesa Rossa-Gratosoglio	6.7
16	Barona-Ronchetto	4.9
17	Lorenteggio-Inganni	8.1
18	Baggio-Forze Armate	5.5
19	San Siro-QT8-Gallaratese	7.5
20	Vialba-Certosa-Quarto Oggiaro	8.3

The spatial distribution of the variable of interest `foreign01` can be represented by a choropleth map with four classes based on the quantiles method (figure 1).

```
. tmap choropleth foreign01, id(id) map("milano-areamap.dta")
> clmethod(quantile) clnumber(4)
```

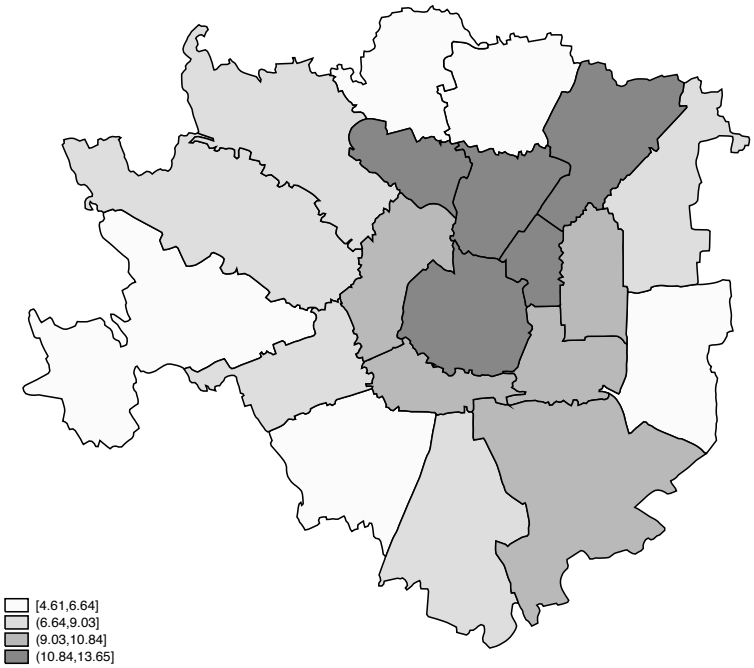

Figure 1: Percentage foreign population, by administrative district, Milano, 2001.

(*Continued on next page*)

Alternatively, we can use a choropleth map with user-defined class breaks (figure 2).

```
. tmap cho foreign01, id(id) map("milano-areamap.dta")
> clmethod(custom) clbreaks(4 6 8 12 14)
```

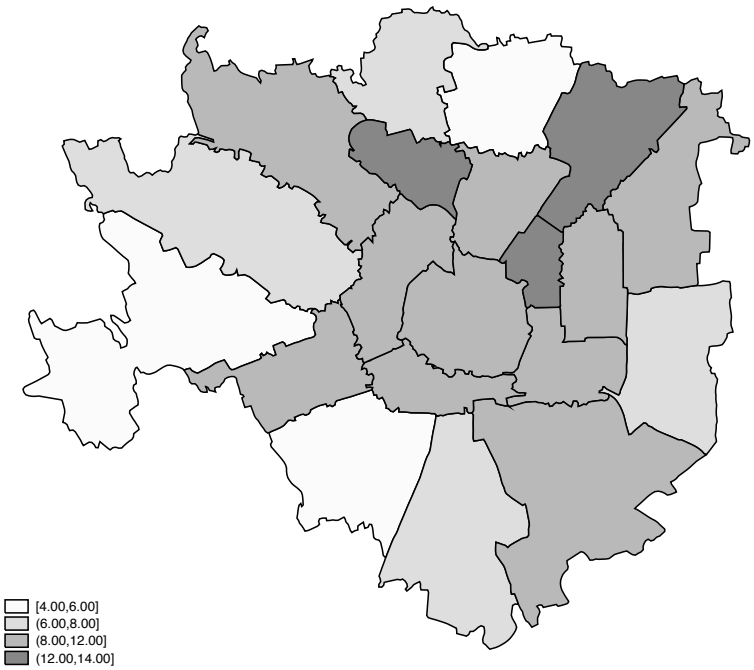

Figure 2: Percentage foreign population, by administrative district, Milano, 2001.

(*Continued on next page*)

The appearance of a choropleth map can be modified, e.g., by specifying a custom palette, changing the color and the thickness of the polygon outlines, adding a title to the legend, and changing the format of the legend (figure 3).

```
. tmap cho foreign01, id(id) map("milano-areamap.dta")
> clmethod(quantile) clnumber(4) palette(Custom) colors(gs13 gs9 gs5 gs2)
> ocolor(white) osize(medium) legtitle("% foreign pop.") legformat("%4.1f")
```

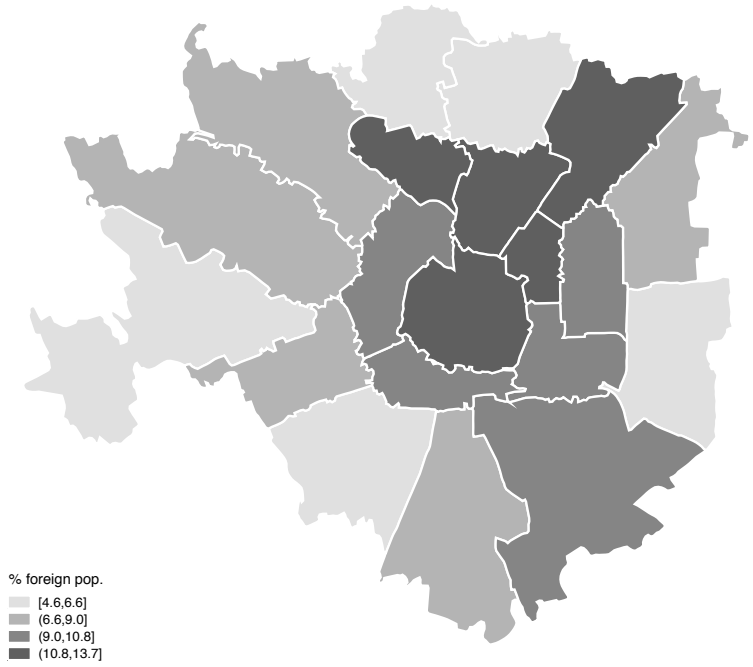

Figure 3: Percentage foreign population, by administrative district, Milano, 2001.

(*Continued on next page*)

The same data can be depicted by means of a proportional symbol map, with the overall boundaries of Milano taken as the base map (figure 4).

```
. tmap propsymbol foreign01, x(x) y(y) map("milano-generalmap.dta") sshape(O)
```

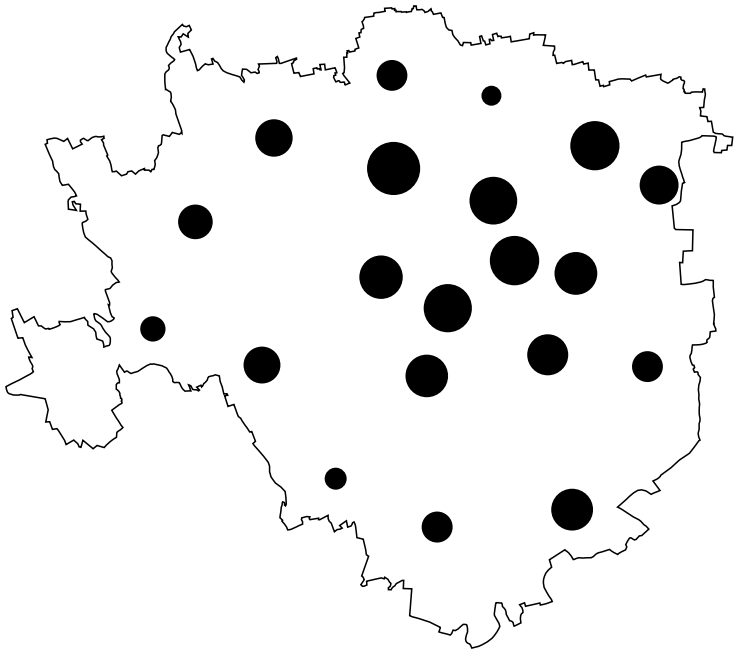

Figure 4: Percentage foreign population, by administrative district (symbol size proportional to variable value), Milano, 2001.

(*Continued on next page*)

If we prefer to emphasize how the various administrative districts deviate from the mean, we can represent the data by means of a deviation map (figure 5).

```
. tmap deviation foreign01, x(x) y(y) map("milano-generalmap.dta")
```

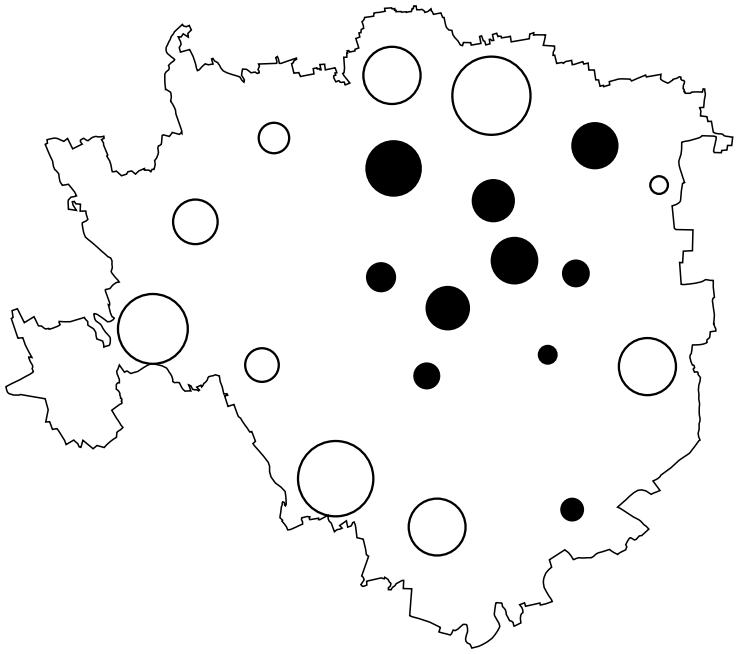

Figure 5: Percentage foreign population, by administrative district: deviations from overall mean (solid symbols denote positive deviations, hollow symbols denote negative deviations, symbol size proportional to absolute value of deviations), Milano, 2001.

(*Continued on next page*)

On the other hand, if we want to see the single values of the distribution, we can use the auxiliary program `tmap label` (figure 6).

```
. tmap label foreign01, x(x) y(y) map("milano-generalmap.dta")
```

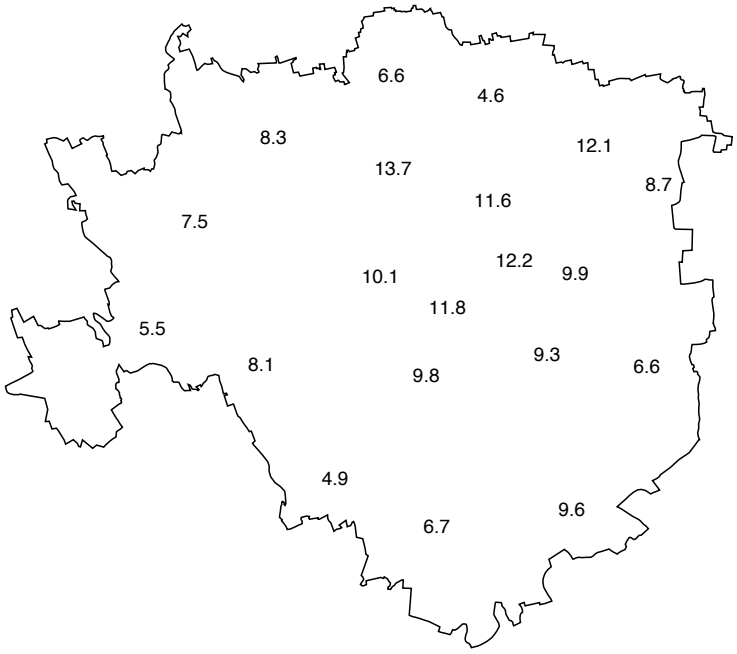

Figure 6: Percentage foreign population, by administrative district, Milano, 2001.

Let us now consider some point data representing the location of police stations in Milano.

```
. use milano-pointdata.dta, clear
. describe
Contains data from milano-pointdata.dta
  obs:            34
 vars:             3                          9 Nov 2004 08:14
 size:           442 (99.9% of memory free)
```

variable name	storage type	display format	value label	variable label
x	float	%9.0g		x-coordinate of location
y	long	%12.0g		y-coordinate of location
type	byte	%16.0g	type	Type of police force

```
Sorted by:
```

First, we can depict the overall spatial distribution of police stations (figure 7).

```
. tmap dot, x(x) y(y) map("milano-generalmap.dta")
```

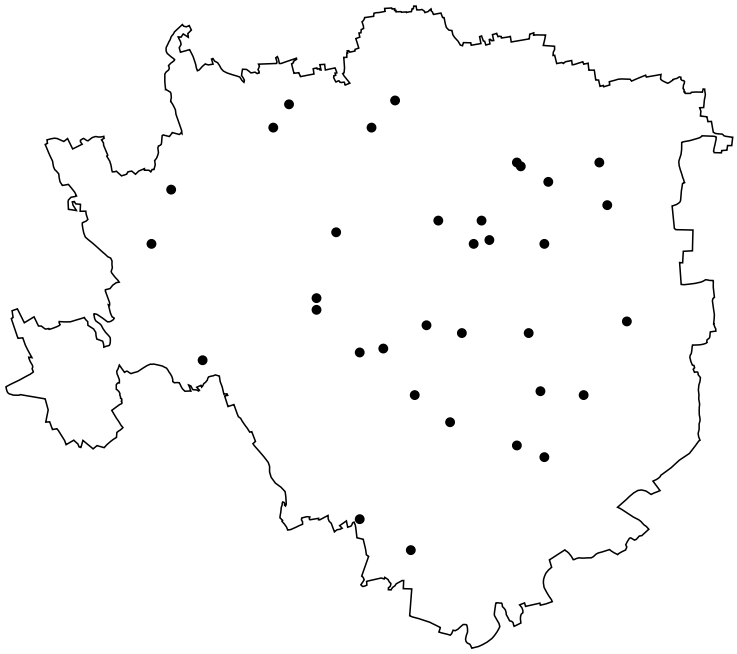

Figure 7: Location of police stations, Milano, 2004.

(*Continued on next page*)

Second, we can show how the spatial distribution of police stations varies by type of police force (figure 8).

```
. tmap dot, x(x) y(y) map("milano-generalmap.dta") by(type) marker(shape)
```

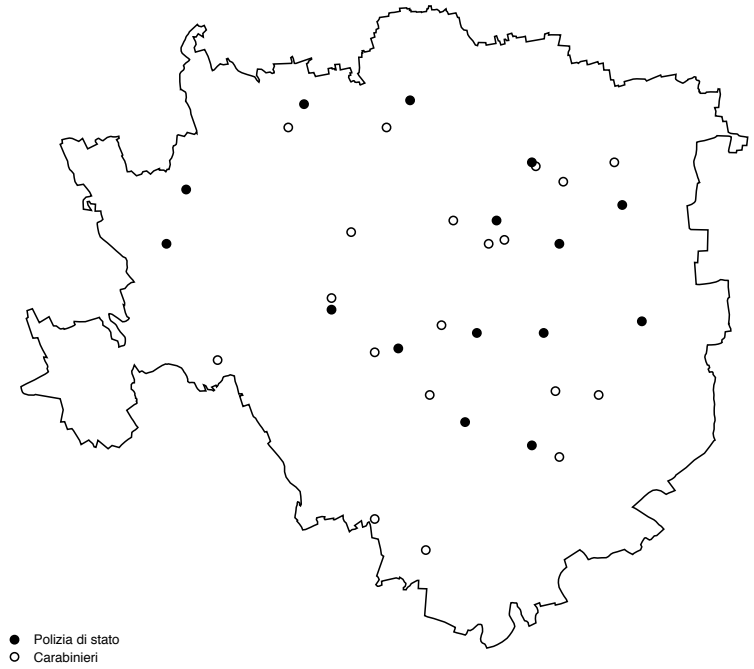

Figure 8: Location of police stations, by police force, Milano, 2004.

4 Limitations

Within its scope, the `tmap` package has two main limitations:

- Enclaves (e.g., San Marino within Italy) are not explicitly handled, so they can get confused with the subareas of interest. More generally, the `tmap` package is not suited for dealing with complex mapping tasks.
- Being based on sersets (see [P] **serset**), the `tmap` package can represent only base maps made of a maximum of about 1,950 polygons.

5 Acknowledgments

The color schemes used in `tmap choropleth` were designed by Cynthia A. Brewer, Department of Geography, Pennsylvania State University, University Park, Pennsylvania, USA. The color schemes are used with Dr. Brewer's permission and are from the ColorBrewer map design tool available at `ColorBrewer.org`. I wish to thank an anonymous reviewer and Nick Cox for their valuable suggestions and for helping improve the `tmap` package. Any remaining errors are mine.

6 References

Bailey, T. C. and A. C. Gatrell. 1995. *Interactive Spatial Data Analysis*. Harlow: Longman.

Evans, I. S. 1977. The selection of class intervals. *Transactions of the Institute of British Geographers* 2(1): 98–124.

Kraak, M. J. and F. J. Ormeling. 1996. *Cartography: Visualization of Spatial Data*. Harlow: Longman.

Monmonier, M. 1993. *Mapping It Out*. Chicago: University of Chicago Press.

Pisati, M. 2001. sg162: Tools for spatial data analysis. *Stata Technical Bulletin* 60: 21–37. In *Stata Technical Bulletin Reprints*, vol. 10, 277–298. College Station, TX: Stata Press.

About the Author

Maurizio Pisati is an associate professor of Sociology at the Università degli Studi di Milano-Bicocca, Italy.