CARTO-SOM

Cartogram creation using self-organizing maps

by

Roberto André Pereira Henriques

Dissertation submitted in partial fulfilment of the requirements for the
degree of

Mestre em Ciência e Sistemas de Informação Geográfica

[Master in Geographical Information Systems and Science]

Instituto Superior de Estatística e Gestão de Informação
da

Universidade Nova de Lisboa

CARTO-SOM

Cartogram creation using self-organizing maps

by

Roberto André Pereira Henriques

Dissertation submitted in partial fulfilment of the requirements for the
degree of

Mestre em Ciência e Sistemas de Informação Geográfica

[Master in Geographical Information Systems and Science]

Instituto Superior de Estatística e Gestão de Informação
da

Universidade Nova de Lisboa

CARTO-SOM

Cartogram creation using self-organizing maps

Dissertation supervised by

Professor Doutor Fernando Lucas Bação

Professor Doutor Victor José de Almeida e Sousa Lobo

November 2005

ACKNOWLEDGMENTS

I would like to express my great gratitude towards my supervisors, Professor Doutor Fernando Bação and Professor Doutor Victor Lobo. They were extremely patience, spending long hours discussing and reviewing this dissertation. Also, their encouragment and pragmatic view of the problems was fundamental to me and to this work. May our collaboration continue on for many years.

I would like to thank the friendship from my colleagues from LabNT. I want also to thank Miguel Loureiro for all the discussions we had about my dissertation and for his great review. Good luck on your dissertation!

Finally, I would like also to express my gratitude to my family for being my support in this dissertation. To my wife Nucha, I must thank all the patience and help she gave me in this time. To my parents I must thank all the oportunities and belief they give me allowing the conclusion of this stage. To my brothers my special thanks for all the support and friendship. Also, my gratitude goes to my parents-in-law and brother-in-law.

CARTO-SOM

Cartogram creation using self-organizing maps

ABSTRACT

The basic idea of a cartogram is to distort a map. This distortion comes from the substitution of area for some other variable (in most examples population). The objective is to scale each region according to the value it represents for the new variable, while keeping the map recognizable. The first cartograms were created to show the geographic distribution of population, in the context of human geography. Cartograms can be seen as variants of a map. The difference between a map and a cartogram is the variable that defines the size of the regions. In a map this variable is the geographic area of the regions, while in the cartogram any other georeferenced variable may be used. In this dissertation we present a general method for constructing density-equalizing projections or cartograms, using the basic SOM algorithm, providing a tool for geographic data presentation and analysis.

CARTO-SOM

Cartogram creation using self-organizing maps

RESUMO

A ideia base do cartograma é a distorção do mapa, usando uma variável estatística, que não a área, para redimensionar cada região mantendo o mapa reconhecível. Os primeiros cartogramas foram criados para mostrar a distribuição geográfica da população. Os cartogramas pode ser vistos como variantes de um mapa no quais se a variável que define o tamanho das regiões é uma variável não geográfica. Nesta dissertação é apresentado um método geral para construir cartogramas usando um tipo de rede neuronal (SOM), fornecendo uma ferramenta para a apresentação e a análise geográfica dos dados.

KEYWORDS

Cartograms

Neural networks

Kohonen Self-Organizing Maps

Geographic Information Systems

Population

Matlab

PALAVRAS-CHAVE

Cartogramas

Redes Neurais

Mapas Auto-organizáveis de Kohonen

Sistema de Informação Geográfica

População

Matlab

ABBREVIATIONS

BGRI	<i>Information Referencing Geographic Basis (Base Geral de Referenciação de Informação)</i>
BMP	<i>Best matching pattern</i>
BMU	<i>Best matching unit</i>
ce	<i>Cartogram error</i>
ESRI	<i>Environmental Systems Research Institute</i>
GA	<i>Genetic algorithms</i>
GIS	<i>Geographic Information System</i>
ICA	<i>International Cartographic Ass</i>
INE	<i>Portuguese National Statistics Institute (Instituto Nacional de Estatística)</i>
Kd	<i>Kocmoud area error</i>
Km	<i>Keim area error</i>
mqe	<i>Quadratic mean error</i>
NCGIA	<i>National Center for Geographic Information & Analysis</i>
qe	<i>Quantization error</i>
SA	<i>Simulated annealing</i>
se	<i>Simple average error</i>
SOM	<i>Self-Organizing Map</i>
te	<i>Topologic error</i>
USA	<i>United States of America</i>
we	<i>Weighed error</i>

TABLE OF CONTENTS

Acknowledgments	iii
Abstract	iv
Resumo	v
Keywords	vi
Palavras-chave	vi
Abbreviations	vii
List of Tables	xi
List of Figures	xii
1. Introduction	1
1.1. State of Art	1
1.2. Objectives	1
1.3. General overview	2
2. Cartograms	3
2.1. Cartograms definition	3
2.1.1. Types of cartograms	6
2.1.1.1. Non-continuous Cartograms	6
2.1.1.2. Continuous cartograms	7
2.1.1.3. Dorling Cartograms	8
2.1.1.4. Pseudo-cartograms	9
2.2. Cartogram applicability	9
2.3. Cartograms algorithms	10
2.3.1. Contiguous Area Cartogram	10
2.3.2. Contiguous Area Cartogram using the Constraint-based Method ..	12
2.3.2.1. Simulated Annealing (SA)	13
2.3.3. Rubber-map Method	16
2.3.4. Pseudo-cartogram Method	17
2.3.5. Medial-axes-based Cartogram	18
2.3.6. RecMap: Rectangular Map Approximations	20
2.3.7. Diffusion cartogram	21
2.3.8. Line Integral Method	22
2.3.9. Cartograms comparison	23
3. Self-Organizing Map	25

3.1.	Introduction.....	25
3.2.	Overview	25
3.3.	SOM algorithm	27
3.3.1.	Sequential training.....	29
3.3.2.	Batch Training	30
3.3.3.	Learning rate functions	31
3.3.4.	Neighbourhood functions.....	32
3.4.	General considerations on SOM	33
3.5.	SOM training evaluation	35
3.6.	SOM visualization.....	36
4.	SOM based cartogram methodology	39
4.1.	Carto-SOM variants.....	43
4.1.1.	Variant 1 (no frame).....	44
4.1.2.	Variant 2 (mean density frame)	45
4.1.3.	Variant 3 (variable density frame).....	46
4.1.4.	Variant 4 (uneven shape)	48
4.2.	Carto-SOM labelling process	50
4.3.	Quantitative quality measures for cartograms	53
4.3.1.	Error measures for each individual region	54
4.3.2.	Global cartogram error	55
5.	Experimental results	57
5.1.	Artificial dataset	57
5.1.1.	Dataset characteristics	57
5.1.2.	Experimental settings	58
5.1.3.	Results.....	59
5.1.3.1.	Maps	59
5.1.3.2.	Quantitative evaluation	60
5.2.	Portugal population based cartograms.....	61
5.2.1.	Dataset characteristics	61
5.2.2.	Experimental settings	64
5.2.3.	Results.....	68
5.2.3.1.	Maps	68
5.2.3.2.	Quantitative evaluation	70
5.3.	USA population based cartograms.....	72
5.3.1.	Dataset characteristics	72

5.3.2.	Experimental settings	74
5.3.3.	Results.....	77
5.3.3.1.	Maps	77
5.3.3.2.	Quantitative evaluation	79
5.4.	Comparative analysis and results	81
5.4.1.	Map Folding.....	81
5.4.2.	Visual comparison overview	82
5.4.3.	Quantitative evaluation overview.....	83
6.	Conclusion	87
	References.....	91
	Appendixes	97
	Appendix 1 - Cartograms	99
	Appendix 2 - Matlab routines	109
	Appendix 3 – SOM training tests.....	131
	Appendix 4 – <i>soMGis</i> application.....	133
	Appendix 5 – Cartogram error.....	139

LIST OF TABLES

Table 1 – Cartogram comparison table.....	23
Table 2 – SOM parameters used in the toy cartogram;	59
Table 3 – Error evaluation made on the Portuguese dataset cartograms.....	61
Table 4 – Portugal layer metadata	62
Table 5 – SOM parameters used in the several tests for Portugal;	65
Table 6 – Tests made on the Carto-SOM and Dougenik cartograms using the Portuguese dataset.	71
Table 7 – USA layer metadata.....	72
Table 8– SOM parameters used in the several tests for the USA;	75
Table 9 – Error evaluation made on the USA dataset cartograms.....	80
Table 10 – Global error evaluation made on the Carto-SOM and Dougenik cartograms.	84
Table 11 – Keim area error calculated for the Carto-SOM variant 2 and Dougenik cartograms	85

LIST OF FIGURES

Figure 1 – Different map representation of USA population data and soil pH.	4
Figure 2 – USA population cartogram.....	5
Figure 3 – USA 2004 presidential election results. Represented in blue are counties with more votes for Kerry and in red are counties with more votes for Bush. (Gastner and Newman 2004)	6
Figure 4 – USA population non-continuous cartogram.	7
Figure 5 – USA population continuous cartogram.	8
Figure 6 – England and United Kingdom Dorling cartograms. (NCGIA 2002).....	8
Figure 7 – Tobler pseudo-cartogram example. (Tobler 1986).	9
Figure 8 – Continuous area cartogram example. a) exerted forces in California state b) resulting cartogram.....	11
Figure 9 – Key vertices identification and border simplification. (House and Kocmoud 1998).....	14
Figure 10 – USA population cartogram using the Continuous Cartogram Construction using the Constraint-based Method algorithm. (House and Kocmoud 1998).	16
Figure 11 – USA population cartogram using the Rubber Map Method. (Tobler 1973).	17
Figure 12 – World population cartogram using the Pseudo-cartogram Method. (Tobler 1986).	18
Figure 13 – Medial axes. a) rectangle object b) triangle object c) polygon. (Keim, North et al. 2005).	18
Figure 14 – Medial-axes-based algorithm a) medial-axes creation b) design of scanlines perpendicular to the medial axes c) scaling factor calculation. (Keim, North et al. 2005)	19
Figure 15 – Medial-axes-based cartogram. USA 2000 presidential elections 2000 (Bush-blue; Core-red). (Keim, North et al. 2005).	19
Figure 16 – RecMap algorithm example. a) Variant 1 (no empty space) b) Variant 2 (preserved topology). (Heilmann, Keim et al. 2004).....	21
Figure 17 – Diffusion Cartogram example. a) 2004 USA presidential elections b) 2004 USA presidential elections cartogram using diffusion cartogram (Gastner and Newman 2004)	22

Figure 18 – Line Integral Method Cartogram example.....	23
Figure 19 –SOM structure.....	26
Figure 20 – Best match unit in an $N \times M$ SOM structure.....	27
Figure 21 – Voronoi regions. Space division where all the interior points are closer to the corresponding generator than to any other.....	30
Figure 22 – Learning rate functions. (Vesanto, Himberg et al. 1999)	32
Figure 23 - Neighbourhood functions.....	33
Figure 24 – SOM visualizations. (Vesanto, Himberg et al. 1999),(Vesanto 2000)...	37
Figure 25 - SOM train.....	39
Figure 26 – Proposed methodology example	41
Figure 27 – SOM rectangular shape superposed in a non-regular shape dataset; a) two regions with a non regular shape; b) random point creation based on a region attribute; c) SOM initialization with an even density.	42
Figure 28 – SOM rectangular shape superposed in a non-regular shape dataset; a) SOM training; b) labelling process; c) units migration to its original position.	42
Figure 29 – Cartogram creation; a) units labelling; b) space allocation to the regions based on the units labels.	42
Figure 30 – Input space nomenclature; a) SOM mapped in the input space; b) input space area definition.	43
Figure 31 – Variant 1 cartogram.	44
Figure 32 – Methodology for variant 2.	46
Figure 33 – Methodology for variant 3.	48
Figure 34 – Methodology for variant 4.	50
Figure 35 – Standard labelling process.....	51
Figure 36 – Cartogram using the standard labelling process.....	52
Figure 37 – Cartogram using the inverse labelling process	52
Figure 38 – Density differences based on standard and inverse labelling.....	53
Figure 39 – Artificial dataset.....	57
Figure 40 – Artificial dataset. High densities in the 2 nd and 4 th quadrants and low densities in the 1 st and 3 rd quadrants.....	58
Figure 41 – Carto-SOM and Dougenik cartogram of the artificial dataset.	60
Figure 42 – Portuguese population data.	62
Figure 43 – Portuguese population density.....	63

Figure 44 – Different point datasets created from Portuguese 2001 population values.	64
Figure 45 – Units mapped in the input space after the training process for each variant.....	66
Figure 46 – Quantization error and topographic error for Portuguese tests.....	67
Figure 47 – Portugal population traditional maps and cartograms using the Carto-SOM methodology	69
Figure 48 – Portugal population cartogram using Dougenik’s cartogram algorithm.....	70
Figure 49 – Distribution of the Portuguese cartogram error (ce) for variant 2.....	71
Figure 50 – USA population data.	73
Figure 51 – USA population density.....	74
Figure 52 – Different point datasets created from USA 2001 population values.	74
Figure 53 – Units mapped in the input space after the training process for each variant.....	76
Figure 54 – Quantization error and topographic error for USA tests.....	77
Figure 55 – USA population cartograms.	78
Figure 56 – Dougenik Cartogram for the USA population.....	79
Figure 57 – Distribution of the USA cartogram error (ce) for variant 2.....	80
Figure 58 – Network folds in variant 4 due to the irregular shape of the border and to ocean units’ immobilization.	81
Figure 59 – Change of the radius parameter. On both tests a 50 x 25 unit’s network was used.....	82

1. Introduction

1.1. State of Art

A cartogram is a presentation of statistical data in geographical distribution on a map (Tobler 1979; Tobler 2005). The basic idea of a cartogram is to distort a map. This distortion comes from the substitution of area for some other variable (in most examples population). The objective is to scale each region according to the value it represents for the new variable, while keeping the map recognizable. The use of cartograms is previous to the use of computerized maps and computer visualization (Tobler 2004). The first cartograms were created to show the geographic distribution of population, in the context of human geography (Raisz 1934). Typically, cartograms are applied to portrait demographic (Tobler 1986), electoral (House and Kocmoud 1998) and epidemiological data (Gusein-Zade and Tikunov 1993; Merrill 2001). Cartograms can be seen as variants of a map. The difference between a map and a cartogram is the variable that defines the size of the regions. In a map this variable is the geographic area of the regions, while in the cartogram any other georeferenced variable may be used.

The self-organizing map (SOM) (Kohonen 1982) was introduced in 1981 and is a neural network particularly suited for data clustering and data visualization. The SOM's basic idea is to map high-dimensional data into one or two dimensions, maintaining the most relevant features of the data patterns. The SOMs objective is to extract and illustrate the essential structures in a dataset through a map, usually known as U-matrix, resulting from an unsupervised learning process (Kaski and Kohonen 1996).

1.2. Objectives

In this dissertation we present a new algorithm to create cartograms based on the SOM. Usually, when creating cartograms, areas with a high value on the selected variable “grow”, occupying the geographic space made available by areas with

smaller values. This grow/shrink process is focused on creating an equal-density map where high values will be represented by larger areas, and small values by smaller areas. In our proposal the cartogram will be created based on the unit's movement during the learning process of the SOM.

1.3. General overview

This thesis is organized as follows: on section 2 we define cartograms, its applicability and review some existing cartograms construction algorithms.

Section 3 introduces the self-organizing maps (SOM) used in the proposed methodology. A general overview of this type of neural network is made along with the algorithm presentation and some parameters used in the training.

Section 4 presents the proposed methodology in cartogram construction using the SOM. This new methodology includes four different variants used to achieve the best cartogram as possible. A new labeling process (a process used in the SOM) is proposed in this section. In order to evaluate the cartogram quality some quantitative measures are reviewed in this section, while a new quantitative measure is also proposed.

In section 5 we implemented the methodology using 3 different datasets: artificial data, Portuguese population from 2001 and USA population from 2001. Cartograms produced for the four variants using SOM different parameters are also presented. Cartograms using another algorithm (Dougenik continuous cartograms) are also produced to perform comparisons. Some tests were made to quantitative evaluate these cartograms.

In section 6 presents some general conclusions and further work.

2. Cartograms

2.1. Cartograms definition

Maps have always been an important part of human life, constituting one of the oldest ways of human communication. The oldest maps known in existence today were created around 2300 BC in ancient Babylonia on clay tablets. Today, maps continue to play a vital role in our society, supporting storage and communication of numerous geographic (and other) phenomena. Since its invention maps have never ceased to improve and can be found at the root of many Human brilliant endeavours, such as the Portuguese discoveries. The advent of personal computers and the related increase in geoprocessing capabilities had a tremendous impact not only on the availability of maps but also on the availability of map making tools. Today almost everybody has access to tools which make maps at the push of a button.

Cartography can be defined *“as the science of preparing all types of maps and charts and includes every operation from original survey to final printing of maps. Cartography can also be seen as the art, science and technology of making maps, together with their study as scientific documents and works of art”* (ICA (International Cartographic Association) 1973).

There are four major map presentations: choropleth maps, dot maps, proportional symbol maps or isopleths maps. Choropleth maps are representations in which each spatial unit is filled with a uniform color or pattern and are appropriate for data that have been scaled or normalized in some way. In Figure 1a we present a choropleth map where the colors represent the population present at each USA state in 2001. Dot maps are made by placing a dot or some other symbol in the approximate location of one or more instances of the variable being mapped producing a visual effect of density. The same data present in the choropleth map is used to build a dot map in Figure 1b. Proportional symbol maps scale icons according to the data they represent. Proportional symbol maps are not dependent on the size of the spatial unit associated with its attribute (Figure 1c). Finally, isopleths maps differ from choropleth maps in that the data is not

aggregated to a pre-defined unit but to the data based unit. This means that new spatial units are build according to data attributes (Figure 1d).

Figure 1 – Different map representation of USA population data and soil pH.

Using the choropleth map one can discover which is the most populated state (California) but due to the color class one cannot order the three states that are next (Texas, Florida and New York) in terms of population size. Analysing the dots or the proportional symbols one can identify which are the most populated states in the US but it is difficult to represent anything else in those maps.

A different way of representing the same data is the cartogram (Figure 2a). In the cartogram one can confirm the population through the size of each state. This is achieved using a distortion process which makes each state size representative of the population value. One of the advantages of this method is the fact that other variables can be added to the map. In Figure 2b an example is shown where the number of families *per* state is mapped on a population cartogram, thus resulting in the representation of two variables.

Figure 2 – USA population cartogram

“Cartograms can be defined as a purposely-distorted thematic map that emphasizes the distribution of a variable by changing the area of objects on the map” (Changming and Lin 1999). *“Area cartograms are deliberate exaggerations of a map according to some external geography-related parameter (variable) that communicates information about regions through their spatial dimensions”* (Dougenik, Chrisman et al. 1985). *“These dimensions have no correspondence to the real world but are a representation of one variable other than the area”* (Kocmoud 1997). Object distortions allow the representation of one variable in the object sizes. In a population cartogram, for example, the sizes of regions are proportional to the number of inhabitants. While population is the most commonly used variable, any social, economic, demographic and geographical variable can be used to build a cartogram. A cartogram can also be seen in the context of traditional map generalisation. While the map represents objects based on their real area, cartograms uses some other variable instead (Heilmann, Keim et al.).

In Figure 3a, a map of the counties of the United States in which each state is colored in red if George W. Bush (Republican presidential candidate) had more votes or in blue if John F. Kerry (Democratic presidential candidate) had more votes. Analysing Figure 3a it is clear that the red area is larger than the blue area. As we know George Bush won the 2004 presidential elections. But does this map represents what real happened in the 2004 elections? Did Mr. Bush won with such a big difference (as the map suggests)? The map can give a wrong

impression since the size of each county does not represent the number of voters. A cartogram using the number of voters in each county can improve the interpretation (Figure 3b). The cartogram gives us a very different idea of the results. We can guess that George Bush won the elections but with a much smaller difference than the traditional map would suggest.

Figure 3 – USA 2004 presidential election results. Represented in blue are counties with more votes for Kerry and in red are counties with more votes for Bush. (Gastner and Newman 2004)

2.1.1. Types of cartograms

Cartograms can be classified according to a number of characteristics like topology, perimeter and the shape. According to NCGIA (2002) there exists four major types of cartograms: non-continuous cartograms, continuous cartograms (Dougenik, Chrisman et al. 1985), Dorling cartograms (Dorling 1994) and pseudo-cartograms (Tobler 1986).

2.1.1.1. Non-continuous Cartograms

Non-continuous cartograms are the simplest cartograms to build. These cartograms do not necessarily preserve topology (Olson 1976). This means that the object connectivity with adjacent objects is not preserved. Each object is allowed to grow or shrink, positioning its borders in a way that the resulting area represents one variable. A non-continuous cartogram is shown in Figure 4 using the population variable. In this example the cartogram was produced by shrinking

all the states. Each state shrink factor is given by the value of population resulting in similar shapes for the most populated states (California, Texas and New York) and in smaller shapes for the less populated states. Different results are possible if expansion processes are used, causing in some cases different objects to overlap which is another way of building non-topological cartograms.

Figure 4 – USA population non-continuous cartogram.

2.1.1.2. Continuous cartograms

Continuous cartograms (Figure 5) differ from non-continuous cartograms due to the topology preservation. In this case topology is maintained (the objects remain connected with their original neighbours) usually causing great distortions in shape. These are the most difficult cartograms to produce since the objects must have the appropriate size to represent the attribute value and at the same time maintain topology and the original shape of objects (as best as possible), so that the cartogram can be easily interpreted.

As we know from cartography, when projecting real earth three dimensional data to a plane, at least one the factors (area, shape, orientation or distances) is distorted (Robinson, Morrison et al. 1984). The same happens in a cartogram, area and distances must change according to the variable of interest, while the shape and orientation should stay, if possible, the same.

Figure 5 – USA population continuous cartogram.

2.1.1.3. Dorling Cartograms

Although Dorling area cartograms do not maintain shape, topology or the objects centroids it has prove to be a useful type of cartogram. Shape expansion/shrink processes are used, producing new shapes which represent the variable of interest. Dorling cartograms differ on the type of the shapes. Instead of preserving the original shape of objects new regular shapes are used, normally circles. This type of cartogram can be seen as a generalization of rectangles cartograms proposed by (Raisz 1934). In Figure 6 two examples of circular cartograms are presented representing England and United Kingdom population.

Figure 6 – England and United Kingdom Dorling cartograms. (NCGIA 2002)

2.1.1.4. Pseudo-cartograms

Finally, pseudo-cartograms are representations that are visually similar to cartograms but do not follow the same building rules. Tobler cartogram is the most famous pseudo-cartogram and it is obtained using a grid which is superimposed on the map. Distortions on the grid are then made to produce an equal-density grid. Applying the movement on each grid cell to the geographic objects will result on a representation where the variable of interest has an equal density.

Figure 7 – Tobler pseudo-cartogram example. (Tobler 1986).

2.2. Cartogram applicability

Cartograms are used in geographic research and are very popular in electoral and population studies. Cartograms can be classified into two categories, linear and area. In the first case, using a non geographical variable, proportional shapes are drawn, maintaining the map as much as possible recognizable. In order to make a cartogram recognizable, objects should be placed as close as possible to their original positions and neighbours should be preserved. Another type of cartogram is the representation of time distance between places. In these cases, while preserving topology, objects are approached or moved away according to the time distance between them. In general, it can be said that cartograms will always be a worse geographical representation of reality than traditional maps. The advantage is the fact that using distortions will improve the understanding of other variables distribution. In other words the map is used as a metaphor, which

is familiar to the reader, in order to convey more abstract concepts and distributions.

2.3. Cartograms algorithms

In this section we describe some of the major algorithms for building cartograms. There are other algorithms which we will not review. The algorithms selected represent different approaches to the problem and constitute the most widely used. The following algorithms are reviewed:

- Contiguous Area Cartogram (Dougenik, Chrisman et al. 1985)
- Contiguous Area Cartogram using the Constraint-based Method (House and Kocmoud 1998)
- *Rubber-map Method* (Tobler 1973)
- Pseudo-cartogram method (Tobler 1986)
- Medial-Axes-based Cartograms (Keim, North et al. 2005)
- RecMap:Rectangular Map Aproximations (Heilmann, Keim et al. 2004)
- Diffusion Cartogram (Gastner and Newman 2004)
- Line Integral Method (Gusein-Zade and Tikunov 1993)

2.3.1. Contiguous Area Cartogram

This is one of the most famous algorithms to build continuous (the authors opted for the use of the word contiguous to express the topology preservation characteristic of this method) cartograms. This algorithm uses a model of forces exerted from each polygon centroid, acting on each boundary coordinates in inverse proportion to distance (Dougenik, Chrisman et al. 1985). The concept of force model can be explained in the formula:

$$F_{ij} = \frac{(P_j - q_j)P_j}{d_{ij}}$$

where:

F_{ij} = is the force applied by polygon j on vertex i

$$P_j = \frac{\sqrt{\text{Actual_area}}}{\sqrt{\pi}}$$

$$q_j = \frac{\sqrt{\text{Desired_area}}}{\sqrt{\pi}}$$

d_{ij} = distance from centroid j to vertex i .

In this method each polygon centroid applies a force F_{ij} to their vertices. This force will move to vertices away if F_{ij} is a positive value and will bring them close if F_{ij} is negative. The intensity of the force exerted on each polygon is determined by the selected variable and is proportional to the difference between the current and the desired polygon area. The current area is assumed to be the actual area of polygon, while the desired area is the distorted area of the final cartogram. As shown in Figure 8 , the California State will exert a positive F_{ij} force moving the vertices away from the centroid. This is the result of having a higher value of population in California than in the neighbourhood states. Inversely, the Nevada State will suffer a negative F_{ij} force making its vertices moving closer to the centroid.

Figure 8 – Continuous area cartogram example. a) exerted forces in California state b) resulting cartogram

In each iteration, the sum of the forces exerted on every vertex of the map is calculated in order to displace the vertex. In the end, the area of every polygon is calculated and compared to the "desired area". When the difference between the two areas is smaller than a specified threshold then the process is stopped and the cartogram finished.

2.3.2. Contiguous Area Cartogram using the Constraint-based Method

House and Kocmoud (1998) propose that the cartogram construction problem should be seen as a constrained optimization problem. Optimization problems are function maximization or minimization problems in which the objective is to find the best of all possible solutions. These problems were first formalized in XVII century physics and geometry. An example of such a problem is the travelling salesman problem (TSP) (Lawler, Lenstra et al. 1985). The TSP consists on finding the best path between points (cities) through a weighted graph (paths between cities) ending at the starting point. The possible solutions are all the combinations between cities passage order. The best solution is the one with the lowest cost.

In the cartogram problem, the map is assumed to be a connected collection of polygons and an area function determines each region area, according to the total map area. The goal is to rescale each object maximizing the map recognition using the following constraints:

- Each rescaled region area is given by an area assignment function;
- Regions topology is maintained;

The search space is given by the space of all maps that maintain topology and respect the area assignment function values to each region. Within these possible solutions the best map is the most recognizable one. The recognizable factor is difficult to define and measure but the authors assumed two features to characterize this factor.

- Region borders orientation similarity between original map and cartogram;
- Each border length should stay as much as possible proportional to their original size;

The optimization problem posed is a high-dimensional problem with a non trivial solution. The objective function and the constraints are conflicting. For example, two regions sharing borders can have an inverse behaviour in terms of getting the final area. Thus, according to the objective function, one of these regions will grow in area and the second will shrink. In this case maintaining topology makes it impossible to retain the borders proportion in both regions. Due to the problem dimensionality simulated annealing (Metropolis, Rosenbluth et al. 1953; Kirkpatrick, Gelatt Jr. et al. 1983; Mitchell 1997) was used as the optimization strategy of the algorithm.

2.3.2.1. Simulated Annealing (SA)

The name of this algorithm is inspired in the process of annealing (process of gradual cooling of a liquid). In an annealing process a melt, initially at high temperature and disordered, is slowly cooled so that the system at any time is approximately in thermodynamic equilibrium. As the cooling proceeds, the system becomes more ordered and approaches a "frozen" ground state at $T=0$. As an example of simulated annealing we can think of a ball (its position represents the actual function value) exploring an irregular surface (possible solutions) reaching a valley after some time. Due to the potential energy in the ball, it will not stop, continuing searching for deeper valleys even if in order to reach them some mountains have to be transposed. As the potential energy lowers and exploration capacity on the ball will decrease stopping in some valley.

In the cartogram problem the temperature or the potential energy is substituted by map coarseness or refinement. The cartogram construction is divided into two tasks: adjusting regions sizes in agreement with selected variable and retaining the regions shape to make the cartogram recognizable. The algorithm implemented is shown bellow.

```

Regions := LoadFullResolutionMap (MapDataFile);
coarseness := max_coarseness;
AreaTargets := CalculateDesiredArea (Regions);
repeat
    SimplifyMap (Regions, coarseness);
    repeat
        AchieveAreas (Regions, AreaTargets);
        RestoreShapeWhileMaintainingArea (Regions);
    until adverse effect of area upon shape increases;
    ReconstructMapToFullResolution (Regions, coarseness);
    coarseness := coarseness / 2;
until coarseness < min_coarseness;

```

Initially the map is loaded along with the variable of interest in order to perform the cartogram. In this phase the coarseness is defined. Area targets are then calculated for each region according to the original area and the selected variable. An iterative process begins ending only if the coarseness reaches the value defined by the user. In this phase the first operation is to identify in the map all the vertices that are not key vertices. Key vertices are those that will provoke topology changes when removed (Figure 9a). If all non key vertices were eliminated the map would get too simple (Figure 9b).

Figure 9 – Key vertices identification and border simplification. (House and Kocmoud 1998)

To solve this problem new vertex connecting the key ones are added (Figure 9c). The number of new vertices added is dependent on the key vertices distance. Thus we will have a maximum distance, initially defined, between any two vertices

on the map. Iteratively this distance will decrease improving map detail. A similar procedure is taken in Douglas-Peucker algorithm (Robinson, Morrison et al. 1984) allowing line generalisation on maps. The next step is to achieve region areas using linear physically based area springs elements that exert force outward along their length when compressed and inward when stretched. An n vertex (x_i, y_i) region area is given by:

$$A = \frac{1}{2} \sum_{i=0}^{n-1} (y_i + y_{i \oplus 1})(x_i + x_{i \oplus 1})$$

Where:

A is the area of a region;

y_i is y coordinate of vertex i ;

x_i is x coordinate of vertex i ;

and the operator $\oplus$ is addition module n and $i \in [0, \dots, n-1]$

The force projected by the springs is proportional to the percentage area error of a region:

$$\mathcal{E}_{Area} = 100(A - A_{desired}) / A_{desired}$$

Where:

$\mathcal{E}_{Area}$ is the percentage error area of a region;

A is the region area and $A_{desired}$ is the region desired area.

The area spring force to be applied to each region vertex is:

$$F_{AS} = \frac{K_{AS} \mathcal{E}_{Area}}{N_{vertices}} u$$

Where:

K_{AS} is a scaling parameter defined by the user

$N_{vertices}$ is the number of each region vertices and

u is the direction that divides the angle formed between the adjacent vertex and vertices.

After calculating the forces applied in each vertex the regions are rebuild and the coarseness factor is update. If this factor is bellow the user threshold then the process is finished and the produced map is the cartogram. On Figure 10, USA population cartogram using this algorithm is shown.

Figure 10 – USA population cartogram using the Continuous Cartogram Construction using the Constraint-based Method algorithm. (House and Kocmoud 1998).

2.3.3. Rubber-map Method

Rubber-map cartogram (Tobler 1973) was one of the first methods for cartogram creation. This method is based on map distortion as if it is a rubber surface. On a rubber map, for example, every person might be represented by a dot. The rubber sheet is stretched so that all the dots are at an equal distance from each other.

Tobler's method initially divides the map into a regular grid, computing a density value for each grid cell. On each cell vertex a displacement direction is computed in order to minimize the density error of its four adjacent cells. This displacement process continues until no improvement can be made. A mean square error criterion has been adopted to measure the efficacy of the iterations. In the end of the process a distorted grid is obtained (Figure 11).

Since there is a one-to-one mapping between the initial and final grids a double bivariate interpolation is used to project the map vertices to and from the final grid. Figure 11 shows on the left the original map space and on the right the cartogram space. On the top of this figure is the grid used to obtain an equal-density population map.

Figure 11 – USA population cartogram using the Rubber Map Method. (Tobler 1973).

One of the disadvantages of this method is the dependence on the selected coordinate system. Another disadvantage is the time needed to converge to a minimum since each cell will take into account only the four adjacent cells density.

2.3.4. Pseudo-cartogram Method

Tobler introduced in 1986 a method that produces pseudo-cartograms (Tobler 1986). This method is similar to the rubber-map method and seeks to reduce the area error by adjusting the lines of latitude and longitude on the map. Tobler's pseudo-cartogram algorithm starts with an orthogonal grid superimposed on the map, after which the horizontal and vertical lines of the grid are moved compressing or expanding in order to obtain an equal density approximation. This method provides a useful tool for effectively minimizing region area error. However this method, like the rubber-map method, depends on the chosen

coordinate axes. Also Tobler's method is known to produce a large cartographic error and is mostly used as a pre-processing step for cartogram construction.

Figure 12 – World population cartogram using the Pseudo-cartogram Method. (Tobler 1986).

2.3.5. Medial-axes-based Cartogram

This cartogram method is based on the medial axes of the map (Keim, North et al. 2002; Keim, North et al. 2004; Keim, North et al. 2005). The medial axis has been used as a simplification of the shape of an object, and thus could be used as a starting point to the computation of cartograms.

In a polygon the medial axes are defined by connecting all the centers of possible circumferences drawn inside the object with a radius as big as possible. In Figure 18 we can visualize the construction of the medial axes. These axes can also be obtained using Delaunay triangulation (Robinson, Morrison et al. 1984).

Figure 13 – Medial axes. a) rectangle object b) triangle object c) polygon. (Keim, North et al. 2005)

Based on the medial axes concept this method uses scanlines to guide the process of cartogram creation. The scanline approach uses a line drawn through the map to inspect the direction in which to extend or contract polygons. These axes represent the orientation of each polygon (Figure 14a). Based on this, scanlines are used to guide the process of cartogram creation (Figure 14b). These segments are perpendicular to the axes created before and represent possible breaking zones on the map. Expansion or compression is applied on these zones which will have a small effect in the general appearance of the map. The next step is to calculate the scaling factor (Figure 14c) determining if the break zone will be expanded or contracted. This scaling factor is obtained by the error function that depends on the actual and desired area on each polygon.

Figure 14 – Medial-axes-based algorithm a) medial-axes creation b) design of scanlines perpendicular to the medial axes c) scaling factor calculation. (Keim, North et al. 2005)

An example of this cartogram is shown in Figure 15 where the 2000 election results were mapped.

Figure 15 – Medial-axes-based cartogram. USA 2000 presidential elections 2000 (Bush-blue; Core-red). (Keim, North et al. 2005)

2.3.6. RecMap: Rectangular Map Approximations

The basic idea on this algorithm is based on representing regions through rectangles in a map. Each rectangle area is proportional to some variable. In order to understand all the information represented in the cartogram rectangles are placed as much as possible in the original position and topology is preserved. Two variants of the algorithm are presented. Both create cartograms where the area on each rectangle is proportional to the study variable. The difference between these two methods is related with the fact that the first method does not allow any empty space, while the second variant preserves the shape polygons.

Five different components were added to the objective function in order to evaluate the quality of produced cartograms. These are area, shape, topology, polygon relative position and empty space. Area is evaluated based on difference between actual area and desired area; topology is quantified checking neighbourhood differences between original map and cartogram; polygons relative position based on the mass centers (centroids) evaluate if the rectangles position is corresponding to original polygons; finally the empty space, allows to compare the existing empty space in the initial map and on the cartogram.

In order to achieve high quality cartograms several ones are built and based on an optimization search the best one is selected. To solve this optimization problem Genetic Algorithms (GA) are used. Genetic algorithms are a part of evolutionary computing, which is a rapidly growing area of artificial intelligence. Evolution has proven to be a very powerful mechanism in finding good solutions to difficult problems. One can look at the natural selection as an optimisation method, which tries to produce adequate solutions to particular environments (Bação, Lobo et al. 2005b). In each iteration a new population of n individuals (cartogram) is built. Each one of these is evaluated based on an objective function allowing the selection of the bests. This process is iterative and will stop after some threshold is reached. The best cartogram is obtained as the best solution.

Figure 16 – RecMap algorithm example. a) Variant 1 (no empty space) b) Variant 2 (preserved topology). (Heilmann, Keim et al. 2004)

2.3.7. Diffusion cartogram

This method is based on the fact that a true cartogram will have an equal-density for all regions. Thus, once the areas of regions have been scaled to be proportional to their population, density is the same everywhere (Gastner and Newman 2004). To create a cartogram, given a particular population density, diffusing from high-density areas into low-density ones is allowed. The linear process of the diffusion of the elementary physics appears to be a good solution to this method.

The population is defined as a density function $\rho(r)$, where r represents the geographic position. This population is allowed to flow in such a way that on the limit of time ($t \rightarrow \infty$) the population density becomes uniform. The tracking of this density diffusion will be then applied to the original geographic objects allowing these to equalise their one density. An example using the diffusion cartogram is shown in Figure 17 where 2004 USA presidential election data is used.

Figure 17 – Diffusion Cartogram. a) 2004 USA presidential elections b) 2004 USA presidential elections cartogram using diffusion cartogram (Gastner and Newman 2004)

2.3.8. Line Integral Method

Zade and Tikunov's Line Integral Method applies radial transformations such that the density of the map is made uniform (Gusein-Zade and Tikunov 1993). The algorithm initially divides the map into a grid of cells. Radial transformations are applied to each cell making its density uniform while leaving the density on other cells unchanged. This process is repeated for every cell changing the shape of the overall map. Since the result will only contain the vertices on the edges of the region boundaries, this transformation is applied in practice as a line integral around each of the region boundaries. While the radial methods produce reasonable results in terms of area error, they also produce a “ballooning” effect that can render regions unrecognizable.

Figure 18 – Line Integral Method Cartogram example.

2.3.9. Cartograms comparison

A summary of all the studied algorithms is presented in Table 1 where some of the most important cartogram characteristics are related to each algorithm.

Cartogram characteristics	Contiguous Area Cartogram (Dougenik, Chrisman et al. 1985)	Contiguous Area Cartogram using the Constraint-based Method (House and Kocmoud 1998)	<i>Rubber-map Method</i> (Tobler 1973)	Pseudo-cartogram (Tobler 1986)	Medial-Axes-based Cartograms (Keim, North et al. 2005)	RecMap:Rectangular Map Aproximations (Heilmann, Keim et al. 2004)	Diffusion Cartogram (Gastner and Newman 2004)	Line Integral Method (Gusein-Zade and Tikunov 1993)
Cartogram type	CC	CC	CC	CC	CC	DC	CC	CC
Radial process	Y	N	N	N	N	N	N	Y
Maintaining Topology	Y	Y	Y	Y	Y	Nn	Y	Y
Optimization algorithms used	-	Sa	-	-	-	Ga	-	-

Table 1 – Cartogram comparison table (legend in the following page).

Legend:

CC – continuous cartogram

DC – Dorling cartogram

Y – yes

N – no

Nn – not necessarily

Ga – genetic algorithms

Sa – simulated annealing

3. Self-Organizing Map

3.1. Introduction

Self Organizing Maps (SOM) were first proposed by Tuevo Kohonen in the beginning of the 1980s (Kohonen 1982), and constitute the product of his work on associative memory and vector quantization. Since then there have been many excellent papers and books on SOM, but his book Self Organizing Maps (edited originally as (Kohonen 1995), and later revised in 1997 and 2001 (Kohonen 2001)) is generally regarded as the main reference on the subject.

Kohonen's SOMs draw some inspiration from the way we believe the human brain works. Research has shown that the cerebral cortex of the human brain is divided into functional subdivisions and that the neuron activity decreases as the distance to the region of initial activation increases (Kohonen 2001). There are several public-domain implementations of SOM, of which we would like to highlight the SOM_PAK and Matlab SOM Toolbox, both developed by Kohonen's research group. The SOM Toolbox is currently in version 2.0, with the latest revision dated 17th of March 2005, and it is publicly available at <http://www.cis.hut.fi/projects/somtoolbox>. The SOM Toolbox has, besides the core Matlab routines, an excellent graphic-based user interface, that makes it a very simple tool to use, simplifying the experiments with SOMs. In this thesis a Visual Basic based software, using ESRI's MapObjects component (ESRI 2005b), was also created in order to improve the connection between geographic data and the SOM algorithm (this approach will be described in appendix 4 (Henriques and Bação 2004)).

3.2. Overview

The SOM's basic idea is to map high-dimensional data into one or two dimensions, maintaining the topological relations between the data patterns. The SOMs main objective is to “*extract and illustrate*” the essential structures in a dataset, through a map resulting from an unsupervised learning process (Kaski

and Kohonen 1996; Kaski, Nikkilä et al. 1998). The SOM is normally used as a tool for mapping high-dimensional data into a one, two, or three dimensional feature maps. This map is an n -dimensional (usually with $n=1$ or 2) grid of neurons or units. That grid forms what is usually referred to as the output space, as opposed to the input space which is the original space where the data patterns lie (Figure 19).

The main advantage of the SOM is that it allows us to have some idea of the structure of the data through the observation of the map. This is possible mainly due to preservation of topological relations, *i.e.*, patterns that are close in the input space will be mapped to units that are close in the output space. The output space is usually 2-dimensional, and in most implementations it is a rectangular grid of units (the grid can also be hexagonal (Kohonen 2001)). Single-dimensional SOMs are also common (e.g. for solving the travelling salesman problem) (Bação and Lobo 2005), and some authors have used 3-dimensional SOMs (Seiffert and Michaelis 1995; Kim and Cho 2004). Using higher dimensional SOMs is rare, for although it poses no theoretical problem, the output space is difficult, if not impossible, to visualize.

Figure 19 –SOM structure.

Each unit of the SOM, also called neuron, is represented by a vector of dimension n , $\mathbf{m}_i=[m_{i1},...,m_{in}]$, where n equals the dimension of the input space. In the training phase, a given training pattern $\mathbf{x}$ is presented to the network, and the closest unit

is selected. This unit is called the BMU (best match unit) (Figure 20). The unit's vector values (synaptic weights in neural network jargon) and those of its neighbours are then modified in order to get closer to the data patten $\mathbf{x}$:

$$\mathbf{m}_i := \mathbf{m}_i + \alpha(t)h_{ci}(t)(\mathbf{x} - \mathbf{m}_i)$$

Where $\alpha(t)$ is the learning rate and $h_{ci}(t)$ is the neighbourhood function centred in unit c . Both parameters decrease with time in the learning phase.

Figure 20 – Best match unit in an $N \times M$ SOM structure.

Independently of the quality of the training phase there will always be some residual distance between the training pattern and its representative unit. This difference is known as quantization error. This value is used to measure the accuracy of the map's representation of data.

3.3. SOM algorithm

The SOM algorithm can be easily described as shown in bellow:

```

Define the network size, learning rate and neighbourhood radius
Randomly initiate the unit's weights
For n iterations
    For each individual from dataset
        Present individual to the network
        Find the BMU
        Update the BMU weights
        Update BMU neighbours' weight
    Update learning rates and neighbourhood radius

```

The first step is to define the network size, the initial learning rate and neighbourhood radius. There are no theoretical results indicating the optimal values for these initial parameters. This way the user's experience plays a major role in the definition of these parameters and can be of paramount importance in the outcome of the method. The next step is the initialization of the unit's weights. These may be randomly generated, providing they have the same dimensionality as the training patterns. The next step is to initialize the training phase of the algorithm. For a number of iterations defined by the user, each pattern from the dataset is selected and presented to the network. Based on Euclidean distance the nearest unit (BMU) is found. The update phase consists on the update of the unit weights and depends on the distance of each unit to the BMU and to the training pattern, and on the neighbourhood function and learning rate.

In order for the SOM to convergence to a stable solution both the learning rate and neighbourhood radius should converge to zero. Usually these parameters decrease in a linear fashion but other functions can be used. Additionally, the update of both parameters can be done after each individual data pattern is presented to the network (iteration) or after all the data patterns have been presented (epoch). The former case is known as sequential training and the latter is usually known as batch training.

3.3.1. Sequential training

The basic SOM algorithm consists in three major phases: competition, cooperation and updating (Silva 2004). In the first phase, competition, all units compete in order to find the BMU for a given training pattern. In the cooperation phase, the BMU neighbourhood is defined based on the neighbourhood radius. Finally, in the update phase the BMU's weights along with those of its neighbours are updated in order to be closer to the data pattern. The algorithm can be written as follows:

```
Let
X be the set of  $n$  training patterns  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ 
W be a  $p \times q$  grid of units  $\mathbf{w}_{ij}$  where  $i$  and  $j$  are their coordinates
on that grid
 $\alpha$  be the learning rate, assuming values in  $]0,1[$ , initialized to a
given initial learning rate
 $r$  be the radius of the neighbourhood function
 $h(\mathbf{w}_{ij}, \mathbf{w}_{mn}, r)$ , initialized to a given initial radius

1 Repeat
2   For  $k=1$  to  $n$ 
3     For all  $\mathbf{w}_{ij} \in W$ , calculate  $d_{ij} = ||\mathbf{x}_k - \mathbf{w}_{ij}||$ 
4     Select the unit that minimizes  $d_{ij}$  as the winner winner
5     Update each unit  $\mathbf{w}_{ij} \in W$ :  $\mathbf{w}_{ij} = \mathbf{w}_{ij} + \alpha h(\mathbf{winner}, \mathbf{w}_{ij}, r) ||\mathbf{x}_k - \mathbf{w}_{ij}||$ 
6     Decrease the value of  $\alpha$  and  $r$ 
7 Until  $\alpha$  reaches 0
```

In this type of training, for each randomly selected training pattern presented to the network, a BMU, i.e. the closest unit, is found. The BMU is then updated according to the weights of the training pattern and the learning rate. Initially this learning rate is high allowing bigger adjustments of the units. The unit's mobility will decrease proportionality with the decrease of the learning rate. Based on the neighbourhood rate, a group of surrounding units is also moved closer to the training pattern.

3.3.2. Batch Training

The difference in batch training when compared to sequential training relies on the unit's updating process, and on the non-obligation to randomly present the training patterns to the network (and sometimes the learning rate may also be omitted (Vesanto 2000)). In this algorithm units are updated only after an epoch, *i.e.* after all training patterns are presented once. In each epoch the input space is divided according to the distance between the map units. The division of the input space is made using Voronoi regions (known also as Thiessen polygons). These regions are polygons which include all points which are closer to a unit than to any other (Figure 21),

Figure 21 – Voronoi regions. Space division where all the interior points are closer to the corresponding generator than to any other.

<http://www.cs.cornell.edu/Info/People/chew/Delaunay.html>

The new units' weights are in this case calculated according to (Vesanto 2000):

$$m_i(t+1) = \frac{\sum_{j=1}^N h_{bi}(t)x_j}{\sum_{j=1}^N h_{bi}(t)}$$

Where t is the time, b is the BMU for the training pattern $\mathbf{x}_i$, and $h_{bi}(t)$ is a neighbourhood kernel centered on the winner unit. The new weight vectors are a

weighted average of the training patterns where the weight of each data pattern is the neighbourhood function value $h_{bi}(t)$ to it's BMU b (Vesanto 2000).

Another way to get the new units' weights, computationally more efficient, is using the Voronoi set centroids n_j :

$$n_j = \frac{1}{N_j} \sum_{x_i \in V_j} x_i$$

$$mj = \frac{\sum_{i=1}^M N_i h_{ij} n_i}{\sum_{i=1}^M N_i h_{ij}}$$

Where N_j is the number of points in Voronoi set V_j .

3.3.3. Learning rate functions

The learning rate (α) assumes values in $[0, 1]$, having an initial value ($\alpha(t_0)$) given by the user. As already discussed, the learning rate decreases to zero during the training phase. There are several different functions that can be used to control the learning rate behaviour. In Matlab SOM toolbox implementation, the available functions to control the decrease of the learning rate are "linear", "power" and "Inv" as shown in Figure 22.

Figure 22 – Learning rate functions. “**Linear**” (in red) $\alpha(t) = \alpha_0(1 - t/T)$; “**Power**” (in black) $\alpha(t) = \alpha_0(0.005/\alpha_0)^{t/T}$ e “**Inv**” (in blue) $\alpha(t) = \alpha_0/(1 + 100t/T)$, where T is the training length and α_0 is the learning initial rate. (Vesanto, Himberg et al. 1999)

3.3.4. Neighbourhood functions

Neighbourhood function, h , can have values in $[0, 1]$, and is a function of the position of two units (a winner unit and another unit) and a given radius r . h has a high value for units that are close in the output space, and decreases with distance increase. Usually, the neighbourhood function is a radial function with a maximum at the center, monotonically decreasing up to a radius r (sometimes called the neighbourhood radius) and is zero from there onwards. Matlab SOM toolbox has implemented several neighbourhood functions, as presented in Figure 23.

Figure 23 - Neighbourhood functions. From the left, “bubble” $h_{ci}(t) = 1(\sigma_{ci} - d_{ci})$; “Gaussian” $h_{ci}(t) = e^{-d_{ci}^2/2\sigma_{ci}^2}$; “Cutgass” $h_{ci}(t) = e^{-d_{ci}^2/2\sigma_{ci}^2} 1(\sigma_{ci} - d_{ci})$ and “Ep” $h_{ci}(t) = \max\{0, 1 - (\sigma_{ci} - d_{ci})^2\}$,

Where σ_t is the neighbourhood radius at time t , and $d_{ci} = \|r_c - r_i\|$ is the distance between units c and i in the output space. a) Neighborhood function using a one-dimension output space. b) Neighborhood function using a two-dimension output space and a neighbourhood radius $\sigma_t=2$. (Vesanto, Himberg et al. 1999)

3.4. General considerations on SOM

The SOM objective is to adjust the units to the data in the input space, so that the network is (the best possible) representative of the training dataset. One of the problems related to this adjustment on the network is the fact that the network tends to underestimate high probability regions and overestimate low probability areas (Bishop, Svensen et al. 1997).

This problem is usually called magnification factor (Cottrell, Fort et al. 1998). A basic theorem for the case one-dimensional data shows that the unit's density is proportional to the training patterns density with a magnification factor of 2/3 (Ritter and Schulten 1986; Bauer, Der et al. 1996). Another problem of the SOM algorithm lies in the fact that the user has to choose the values of several parameters heuristically (size of the network, learning rate and neighbourhood radius). Some research has been made on this subject and (Haese and Goodhill 2001) proposed an algorithm that estimates the learning parameters during the training of SOMs automatically, using a Kalman filter. Another proposal was the

use of Genetic Algorithms (GA) in order to estimate the training parameters of a SOM (Silva and Rosa 2002).

The SOM dimension and size depends on the problem at hand. This is mainly an empirical process (Kohonen, 2001), where usually a two-dimension SOM is used due to its representation in the space. The size of the SOM must also take into consideration the size of the dataset of training patterns. Three fundamentally different approaches are possible when choosing the size of the SOM, commonly known as “k-means SOMs”, “emergent SOMs”, and “average SOMs” (Bação, Lobo et al. 2005a). In “k-means” SOMs, the number of units should be equal to the expected number of clusters, and thus each cluster should be represented by a single unit. In “emergent SOMs”, a very large number of units is used (sometimes as many or more than the number of training patterns), so as to obtain very large SOMs. These very large SOMs allow for very clear U-Matrices (Ultsch and Siemon 1990) and are useful for detecting quite clearly the underlying structure of the data. Finally, most authors will use far less units than training patterns available (Lobo 2002), but still many more than the expected number of clusters. This leads to SOMs where each unit maps a large number of training patterns, and thus covers a fair amount of input space. However, each expected cluster can still be represented by a number of different units. In this case, if there is a clear separation in the input space between the different clusters, there will be units that, because they are “pulled” both ways, will end up being positioned in the regions between the clusters, and may not map any of the training patterns.

The training of a SOM is more effective if it is done in two phases: the unfolding phase, and the fine-tuning phase. In the unfolding phase the objective is to spread the units in the region of the input space where the data patterns are located. In this phase the neighbourhood function should have a large initial radius so that all units have high mobility and the map can quickly cover the input space.

The fine tuning phase, as the name implies, is the process of small adjustments in order to reduce the quantization error, and center the units in the areas where the density of patterns is highest. Usually, in this phase the learning rate and the neighbourhood radius are smaller than the ones used in the unfolding phase. As

these two parameters are smaller, the map will need more time to adjust its weights and that is why the number of iterations or epochs is normally higher.

In order to best understand the units' position and their relation with the training patterns one can proceed by giving labels to each unit, according to some attribute (from now on called label) of its training patterns. This process is called the labelling, or calibration (Kohonen 2001). Some variations are possible in this process: each unit can get the labels from all its training patterns; each unit can get only a unique label from its training patterns (say, the label of the nearest training pattern); or each unit can get the most frequent label from its training patterns.

3.5. SOM training evaluation

There are several ways to evaluate the quality of a SOM after the training phase. In this thesis we focus on the quantization error (qe) and the topographic error (te) (Kohonen 2001). The quantization error is given by the average distance between a unit and the training patterns mapped to it, *i.e.* all the input data patterns that share it as BMU.

$$qe = \frac{\sum_{k=1}^n \| \mathbf{x}_k - \mathbf{w}_{BMU} \|}{n}$$

Where $\mathbf{x}_k$ is the training pattern, $\mathbf{w}_{BMU}$ is the BMU for the training pattern $\mathbf{x}_k$ and n is the number of existing training patterns.

The topographic error (te) evaluates the topology preservation of the SOM. Considering that for each data pattern the closest unit will be its BMU, the second closest unit will be called the second BMU, or BMU2. Topographic error measures the proportion of all training patterns for which first and second BMUs are not adjacent units.

$$te = \frac{1}{n} \sum_{k=1}^n u(\mathbf{x}_k)$$

Where $\mathbf{x}_k$ is the training pattern, n is the number of data vectors and $u(\mathbf{x}_k)$ equals one (1) if the BMU and the BMU2 are not adjacent and zero (0) otherwise.

3.6. SOM visualization

There are several ways to visualize the SOM in order to improve the understanding of the data patterns processed. One of the ways is to use the output space matrix and map the units along with their labels (Figure 24a). This map allows users to visualize the way training patterns were mapped into the SOM. Another possible way to present the SOM is to map the units in the input space. This process is possible only if the input space has one, two or three dimensions, since the representation of more than three dimensions is not feasible. Another possibility in this process is to map the units using only one, two or three vector components (Figure 24b) but in this case not all information is being used and the results may be misleading. The visualization of the training patterns' positions on the map is often not enough to see structures, because high dimensional distances are distorted in the projection. Units that are immediate neighbours on the map can still be separated by large distances in the input space.

The U-Matrices (Ultsch and Siemon 1990) are computed by finding the distances in the input space of neighbouring units in the output space. There are two ways to visualize a U-matrix. The most common is to use a color code to depict distances, corresponding to the values of the U-Mat. Usually a grayscale is used, with the highest value being represented with black and the lowest with white (Figure 24c). Another way is to plot these distances in form of a 3D landscape with mountains and valleys. A mountain region indicates large distances between units, while low distances between the units form valleys.

Component planes are also used in the SOM visualization and exploration. In a component plane each unit is colored according to the weight of each variable in the SOM (Figure 24d). Through the analysis of the component planes, data patterns can be detected. For instance, it is quite simple to identify variables which are correlated (their component planes will have the same shape), and it is also possible to have an improved understanding of the contributions of each variable to the SOM.

Figure 24 – SOM visualizations. (Vesanto, Himberg et al. 1999),(Vesanto 2000)

4. SOM based cartogram methodology

In this dissertation we present a new algorithm to build cartograms based on the Self-Organizing Map (SOM). Usually, when creating cartograms, areas with a high value on the variable of interest “grow”, occupying the geographic space made available by areas with smaller values. This grow/shrink process is focused on creating an equal-density map where high values are represented by larger areas, and small values by smaller areas. In our proposal the cartogram is built based on the unit’s movement during the learning process of the SOM (Kaski and Lagus 1996; Skupin 2003; Henriques, Bação et al. 2005).

As already described, the SOM is usually used to project high-dimensional data into lower dimensions. However, two-dimension training patterns can be used to project a SOM with the same dimension. In this case, data visualization is not the goal, since it is possible to visualize data without the use of a SOM. The SOM will adapt the units, as best as possible, to the training patterns. One can imagine a surface of points (Figure 25a), with known positions and with different densities. Using these points as training patterns the SOM (Figure 25b) will adapt to the surface (Figure 25c). As expected, more units are projected into higher density areas and fewer units will be placed in low density areas.

Figure 25 - SOM train

As already described in section 2, the cartogram objective is to make an equal density map based on a variable of interest. In our surface example (Figure 25a) a perfect cartogram would be produced by moving the units (Figure 25c) to equidistant positions. If we think of the surface as a rubber-like surface, we can imagine the surface being distorted along with the movement of the units. The new cartogram construction methodology is based on the surface distortion (Figure 25c).

An example of this process is shown in Figure 26, using a simplified example with only two regions. The two regions are geographically identical but have distinct values for the variable of interest, denoted as p . Assuming variable p to be population, the region color represents the value of population (dark color corresponds to a higher value of population). In the following step (Figure 26b) we randomly generate points (represented with triangles) inside each region. The number of points generated is a linear function of the value of population. In Figure 26c a two-dimensional SOM (4 x 4) is initialized. The SOM units are initialized with an even density, contrary to the usual practice, in which the units are randomly initialized in the input space. After the training phase (Figure 26d), where the units are adapted to the training patterns, the algorithm proceeds with the labelling process (Figure 26e). This process gives each unit an attribute based on the region in which the unit falls in. In Figure 26e each unit color represents its label. Figure 26f represents the mapping of each unit to its initial position. A space transformation is then performed based on the unit's position and label resulting on a population cartogram (Figure 26h).

Figure 26 – Proposed methodology example

As already described in section 0, the SOM is made of an n-dimensional grid of units or neurons. In this particular case two dimensional SOMs are used and therefore the network will have a rectangle shape. This means that, in any complex shape, units will eventually be initially positioned outside the regions. As shown in Figure 27 when using non-regular shape regions some units, at the initialization phase, are positioned outside the regions area.

Figure 27 – SOM rectangular shape superposed in a non-regular shape dataset; a) two regions with a non regular shape; b) random point creation based on a region attribute; c) SOM initialization with an even density.

The initialization of some units in areas without any training patterns will cause the units to move to areas populated with training patterns. The Figure 28 shows the application, to this specific case, of processes already described of training, labelling and units migration to its original position.

Figure 28 – SOM rectangular shape superposed in a non-regular shape dataset; a) SOM training; b) labelling process; c) units migration to its original position.

The next phase on the cartogram creation is the mapping of the units in the input space (Figure 29a) with the label and the allocation of the space to the original regions (Figure 29b).

Figure 29 – Cartogram creation; a) units labelling; b) space allocation to the regions based on the units labels.

When comparing the produced cartogram with the original regions shape we verify that the cartogram regions will occupy all the area on the input space and the original regions shape is lost.

In order to better describe this problem we define in the input space three different areas (Figure 30). We assume the input space as our domain area (D); the area occupied by the original regions as the region area (ra) and the remaining area as the buffer area (δ).

Figure 30 – Input space nomenclature; a) SOM mapped in the input space; b) input space area definition.

The input space can now be defined as:

$$D - \delta = ra$$

As already shown when no training patterns exists in the buffer area (δ) the produced cartogram will occupy the domain area (D) and the original shape of the region area (ra) will be lost. In order to try to solve this problem we create and test four variants of the methodology. The first variant follows the already explained methodology where the ra is reshaped to the D area.

4.1. Carto-SOM variants

The four variants are the following:

1. Variant 1 (no frame), where standard SOM algorithm was used with training patterns generated only inside the region area (ra)
2. Variant 2 (mean density frame), where standard SOM algorithm was used with training patterns generated in the entire domain (D). The training patterns generated in the buffer area (δ) are proportional to the mean density of the regions on the ra .
3. Variant 3 (variable density frame), where standard SOM algorithm was used with training patterns generated in the entire domain (D). In this variant the buffer area (δ) is divided according to the nearest region of the ra and its density was used.
4. Variant 4 (uneven shape), where a variant of SOM algorithm was used with training patterns generated only inside the region area (ra)

4.1.1. Variant 1 (no frame)

This is the first variant tested to construct cartograms and the standard SOM algorithm was used. The SOM algorithm used only differs from the original one in the initialization phase. Usually, units are randomly initialized but in this methodology units were equally spread along the data space (Baçã, Lobo et al. 2005c). This variant uses the methodology already explained and produces a cartogram where the domain area is occupied, causing a big distortion on the regions shape and making the cartogram difficult to recognize.

Figure 31 – Variant 1 cartogram.

4.1.2. Variant 2 (mean density frame)

The variant 2 tries to improve the resulting shapes by generating training patterns were in the entire domain (D) (Figure 32a), in contrast to the variant 1 in which the training patterns were only generated to in ra . The training patterns generated in the buffer area (δ) are proportional to the mean density of the regions on the ra . The green triangles in Figure 32b represent the mean density. In Figure 32c the SOM is initialized in the same way used in variant 1. The SOM training (Figure 32c) and labelling (Figure 32d) is then performed. In Figure 32f is possible to see the units' migration to its original position and in Figure 32g the final network is presented. As we can see in this figure some units have the label of the δ area. This fact causes the inclusion of a region representing the δ area in the produced cartogram (Figure 32h). Due to the limited number of units presented in the figure, the cartogram does not maintain the general shape of the regions. In practice with a bigger number of units this property is generally maintained.

From the cartogram presented in Figure 32i we can argue that regions grow through the δ area if they have a density higher than the mean density of ra . In an inverse way if ra density is lower than the mean density then the δ area will grow towards the region.

Figure 32 – Methodology for variant 2.

4.1.3. Variant 3 (variable density frame)

In variant 3 the δ area is also populated with training patterns. The difference in this variant relies on the fact that the training patterns generated in the δ area are not proportional to the mean density. The δ area is divided according to the proximity of each region using the Voronoi tessellation, also known as Thiessen polygons (Robinson, Morrison et al. 1984). This variant tries to solve a problem that can arise in variant 2 if most of the regions have a density lower than the mean density. In these cases the produced cartogram will crush all these regions and the δ area will grow and occupy the spaces made available.

As shown in Figure 33a the δ area is subdivided in two smaller areas (δ_1 and δ_2) according to the regions in the ra (ra_1 and ra_2). Each of these smaller areas is populated with random patterns in a way that:

$$\frac{tp_{ra_i}}{A_{ra_i}} = \frac{tp_{\delta_i}}{A_{\delta_i}}$$

Where:

tp_{ra_i} is the number of training patterns generated in the region ra_i

tp_{δ_i} is the number of training patterns generated in the region δ_i

A_{ra_i} is the area of the region ra_i

A_{δ_i} is the area of the region δ_i

As we can see in the Figure 33b the density of the training patterns in the δ_i areas is similar to the density in the ra_i areas Figure 33c shows the SOM initialization, the training (Figure 33d), the labelling phases (Figure 33e) and the movement of the units to its original position (Figure 33f). The final units' positions and their labels are presented in Figure 33g. As we can see in this figure due to the density equality between region areas (ra) and its neighbouring buffer areas (δ) the produced cartogram generally maintains the shape on the original regions. The objective in this variant is to force the region areas to compete with each other, in order to achieve its desired area, occupying as less as possible the δ area.

Figure 33 – Methodology for variant 3.

4.1.4. Variant 4 (uneven shape)

In variant 4 some changes are made in the SOM algorithm, in order to ensure the preservation of the regions shape between the original map and the cartogram. In appendix 2 the Matlab source code for this SOM is presented. We proposed a preliminary step where all the units positioned in the δ area are excluded from the network and therefore don't participate in the training phase. The algorithm is explained in bellow:

Let
 X be the set of n training patterns $\mathbf{x}_1, \mathbf{x}_2$
 Y be the set of m calibrating patterns y_1, y_2 with labels S (for sea) and L (for land)
 W be a $p \times q$ grid of units $\mathbf{w}_{ij}$ where i and j are their coordinates on that grid
 a be the learning rate, assuming values in $]0,1[$, initialized to a given initial learning rate
 r be the radius of the neighbourhood function $h(\mathbf{w}_{ij}, \mathbf{w}_{mn}, r)$, initialized to a given initial radius
 I be the list of $\mathbf{w}_{ij} \in W$ which when calibrated with Y have label L

- 1 Initialize W based on Geo-Som criteria (unit will be uniformly distributed in a geographic space)
- 2 Label the units of W with the calibrating patterns Y
- 3 Select for I all unit of W which have label L (select for I all units which are in land)
- 4 For $k=1$ to n
 - 5 For all $\mathbf{w}_{ij} \in I$, calculate $d_{ij} = ||\mathbf{x}_k - \mathbf{w}_{ij}||$
 - 6 Select the unit that minimizes d_{ij} as the winner $\mathbf{w}_{winner}$
 - 7 Update each unit $\mathbf{w}_{ij} \in I$: $w_{ij} = w_{ij} + a h(w_{winner}, w_{ij}, r) ||\mathbf{x}_k - \mathbf{w}_{ij}||$
 - 8 Decrease the value of a and r
- 9 Until a reaches 0

In this variant the training patterns are randomly generated inside the regions areas (ra) (Figure 34b). The next step is the initialization of the network (using the same process as in the other variants) and based on the units position those who falls outside the regions area (ra), i.e. those that are located in the buffer area (δ) are excluded from the network (in Figure 34c those units are represented in white). The following steps are the SOM training (Figure 34d), labelling (Figure 34e) and the migration of the units to their original position. Notice that, as already defined, none of the excluded units participate on these phases. From this step forward the methodology is similar to those presented before for the other variants. In Figure 34i we present the produced cartogram. Due to the small number of units used in these examples it's difficult to demonstrate that the original regions shapes are maintained. With the increase of the number of units used in the SOM the maintenance of the regions border also increases.

Figure 34 – Methodology for variant 4.

4.2. Carto-SOM labelling process

The labelling process (or calibration), already explained in section 3.4 consists on giving each unit an attribute (known as label) derived from the training patterns. This process can be seen as a division of the input space based on the data characteristics. In the two-dimensional example presented in Figure 35a we can see the units (in red) mapped in the input space. The input space will be then divided based on each unit (Figure 35b). This space allocation is based on Thiessen polygons, already explained in this document. Based on these regions each training data will be assigned to one unit and its selected attribute (label) will correspond to the unit's label.

Figure 35 – Standard labelling process

Usually this process works well and gives each unit one or several labels. Several variants of this process are implemented on Matlab SOM Toolbox: each unit can get all the attributes from its training patterns; each unit can get all different attributes from its training patterns or each unit can get the most frequent attribute from its training patterns.

In our case a problem arises due to the number of units and training patterns used. Since we use a training dataset which is too small when compared with the units' number (on appendix 3 the number of training datasets and units used for the several tests is shown) sometimes after the labelling process we have several units without any label. This case is represented in the Figure 35b by the green unit, i.e. no training pattern will be allocated to this unit. This causes a problem in the cartogram construction, because without label these units will return a discontinuous cartogram. An example of this problem is also shown in Figure 36 when using the Portuguese and USA dataset. As we can see on both pictures there exists a high number on non labelled units making the cartogram discontinuous and difficult to read.

a) Portugal cartogram with standard labelling

b) USA cartogram with standard labelling

Figure 36 – Cartogram using the standard labelling process

a) Portugal cartogram with inverse labelling

b) USA cartogram with inverse labelling

Figure 37 – Cartogram using the inverse labelling process

This way a different process was used in order to produce better cartograms. Since we use a number of units too high when compared to the number of training patterns the labelling is now based on the distance between units and training patterns. For each unit the closest training pattern is calculated and its label is used. One training pattern could be used to label several units if it is the nearest training patterns for all of them. We can think of this training pattern as the best match pattern (BMP) (in analogy to BMU) for the units. Using this process all the units will always be labelled and the discontinuous cartogram problem

disappears. This labelling process will be identified in this work as inverse labelling since we calculate the BMP which is an inverse process to the BMU. Two examples using the inverse labelling are presented in Figure 37 using Portugal and USA population values. These cartograms, as desired, are continuous and easier to understand.

Based on these two different labelling procedures different cartograms are built. In Figure 38 we present for each Carto-SOM variant the density variation for Portuguese regions when using the standard and inverse labelling. Analysing Figure 38 we can detect a higher density difference for the variant 1. This is consistent with our explanation about the null labelling due to the high number of units compared with the training patterns. In variant 1 the training patterns dataset is the smaller used (only points inside study area borders) and the number of units is the bigger used (all units participate in the train).

Figure 38 – Density differences based on standard and inverse labelling.

4.3. Quantitative quality measures for cartograms

While it is rather subjective to compare the shape of different cartograms, it is relatively easy to define a numerical value that characterizes how well the

cartogram distributes the available area between the different regions, according to the given variable of interest. Various such measures have been proposed. We will now review some of them.

4.3.1. Error measures for each individual region

Keim, North *et al.* (2004) propose an area error function in order to determine the cartogram error in each region. This relative area error E_{rel}^j of a region p_j is give by:

$$E_{rel}^j = \frac{|A_{desire}^j - A_{actual}^j|}{A_{desire}^j + A_{actual}^j}$$

Where A_{desire}^j is the region j optimal area in the cartogram in order to create a perfect cartogram and A_{actual}^j is the real area for the region j in the cartogram.

Another possible error measure to evaluate the cartogram accuracy is the amount of error area of a region (Kocmoud 1997) given by:

$$\mathcal{E}_{area} = \frac{100 \times (A - A_{desire})}{A_{desire}},$$

which is the percentage difference between the current region area A and the target area A_{desire} . In this dissertation we defined another measure, which we called cartogram error (ce), is defined as follows:

Let N be number of regions in the map

Let a_i be the area occupied by region i

Let v_i be the value of the variable of interest v in region i

The cartogram error of region i relative to variable v is

$$ce(i, v) = \frac{a_i}{\sum_{j=1}^{j=N} a_j} - \frac{v_i}{\sum_{j=1}^{j=N} v_j}$$

The first term in the equation is the percentage of total area occupied by the given region, and the second is the percentage of the variable of interest contained in that region. Thus, this measure is no more than the difference between the percentage of area occupied by a region, and the percentage of the variable of interest contained in that region. A negative value indicates that the area in the cartogram is smaller than it should be, if it were to accurately represent the value of interest. For a perfect cartogram, this error is zero, since in that case each region occupies an area strictly proportional to the value of its variable of interest. Since the cartogram error is a percentual error it has values in $[-1, 1]$. The extreme vales of -1 and 1 would mean that all the area is used up by a region where the value of interest is zero, or *vice-versa*.

4.3.2. Global cartogram error

The error measures described above are relative to each individual region. We need to define a global error that may define the quality of the cartogram.

Based on the error of each region, we may define a global measure of the “goodness” of a cartogram, relative to the variable if interest v , by its weighed mean, simple mean or quadratic mean. The quadratic mean error (mqe), defined as:

$$mqe(v) = \frac{\sqrt{\sum_{i=1}^{i=N} e(i, v)^2}}{N}$$

The weighed mean error (we) is defined as:

$$we(v) = \sum_{i=1}^{i=N} |a_i \times e(i, v)|$$

The simple average error (*se*) is defined as:

$$se(v) = \sum_{i=1}^{i=N} |e(i, v)|$$

These measures can be applied to both a cartogram and to an ordinary geographical map. The global error of a geographical map is a measure of how much the variable of interest is misrepresented by the geographical area of the different regions, and thus how much distortion is necessary to obtain an accurate cartogram.

In all the measures presented above the perfect cartogram will produce an error equal to zero, while the increase of the distance of the error to zero means a decrease of the cartogram quality.

Another possible measure that can be used to assess the quality of the representation is the Pearsons R^2 . This coefficient measures how well the ratios between different areas and variables of interest of each region adjust to a straight line. The problem with using the R^2 is that in order to adequately measure the proportionality, between areas and variables of interest, it is necessary to guarantee that the line passes at the origin, and it should preferably have a slope of 1. Because of these difficulties we decided not to use R^2 as a quality measure.

5. Experimental results

The Carto-SOM methodology was tested using an artificial dataset, Portugal's population data in 2001 and USA's population data also in 2001. For each dataset Carto-SOM's four variants were applied using different SOM parameters, and compared with Dougenik's cartograms.

5.1. Artificial dataset

We shall now describe the characteristics of this data set, the maps and quantitative results obtained.

5.1.1. Dataset characteristics

The artificial dataset used in these tests was created to test the proposed cartogram construction methodology. We defined 4 rectangular regions with equal areas, forming a larger rectangle. We defined a variable of interest (which may represent population, income, or any other parameter), that has high values in two of the regions, and low values in the other two. To make the problem more challenging, the different values of this regions form a chess pattern, *i.e.*, the regions of quadrant 1 and 3 have a high value for that variable (the same for both regions), and the regions 2 and 4 have a low value (see Figure 39)

Figure 39 – Artificial dataset.

To use the Carto-SOM, we must generate points for each of the regions. Points were randomly generated with x positions assuming values in $[0, 50]$ and y positions with values in $[0, 100]$. These points were distributed in four quadrants. The density of points has a given value (relatively high, given by the value of interest) in the 1st and 3rd quadrants, and a different value (relatively low, also given by the value of interest) in the 2nd and 4th quadrants (Figure 40). Around four hundred and fifty points were generated for 1st and 3rd quadrants and one thousand seven hundred and fifty were generated in the remaining quadrants.

Figure 40 – Artificial dataset. High densities in the 2nd and 4th quadrants and low densities in the 1st and 3rd quadrants.

5.1.2. Experimental settings

Since the regions in this dataset occupy a rectangular area, all four variants of the Carto-SOM will produce similar results, and a single experiment is sufficient.

Based on the artificial dataset (Figure 40) created to test our methodology, we trained the SOM using the SOM-PAK software available at http://www.cis.hut.fi/research/som_lvq_pak.shtml. The SOM parameters used in this test are presented in the Table 2.

X	y	Iter1	Iter2	n1	n2	L1	L2
50	100	30000	50000	25	10	0.5	0.2

Table 2 – SOM parameters used in the toy cartogram; x and y are the number of units in x and y used in the SOM; $iter1$ and $iter2$ are the number of iterations for the first and second phases (the unfolding and fine tune phases); $n1$ and $n2$ are the neighbourhood radius used in the first and second train (the function was “bubble”) and $L1$ and $L2$ are the first and second phase learning rates.

For Dougenik’s method, we used an ArcGIS script file to produce the cartograms (Schmid 2005). This script allows the ArcGIS user to select an input layer (a new layer based on the artificial dataset was created) and to choose the number of iterations used. In this test we used 5 iterations to produce Dougenik’s cartograms.

5.1.3. Results

Carto-SOM and Dougenik’s cartogram construction methods were applied to the artificial dataset producing several cartograms. The results are presented in two phases: in the first phase we present the cartograms while in the second phase we evaluate the cartogram quality based on the quantitative measures already presented.

5.1.3.1. Maps

The results are presented in Figure 41 where we can see the produced cartograms based on the Carto-SOM and Dougenik’s methods. High density areas tend to grow, occupying the low density areas. The topology was in this case preserved by distorting the original shape of each region. Visually these are good results, since the cartograms represent the variable density by the size of each region, while maintaining topology, *i.e.*, all areas have the same neighbours, and important points, such as the junction of all four regions in the center, are retained.

a) Carto-SOM

b) Dougenik cartogram

Figure 41 – Carto-SOM and Dougenik cartogram of the artificial dataset.

5.1.3.2. Quantitative evaluation

Some tests were performed in order to analyse the cartogram accuracy. In Table 3 we present for the Dougenik's and Carto-SOM cartograms, 9 different error values evaluating the global quality of each cartogram. A simple average error, a mean quadratic error and a weighed error were calculated for the cartogram error (ce), the Keim area error (km) and for the Kocmoud area error (kd).

As we can see from the table (Table 3), the Carto-SOM has a significantly lower error than the cartograms produced by Dougenik's method. All the Carto-SOM variants, for the artificial dataset, have the same error values. This is explained due to the fact that the artificial dataset input space defines a rectangle making the cartograms produced using the four variants the same.

	Error measures	se	mqe	We
Dougenik	ce	5,42%	0,60%	5,42%
	km	14,64%	1,83%	10,03%
	kd	35,08%	4,58%	21,67%
Carto-SOM variant 1	ce	2,06%	0,23%	2,06%
	km	6,19%	0,80%	3,98%
	kd	13,37%	1,75%	8,26%
Carto-SOM variant 2	ce	2,06%	0,23%	2,06%
	km	6,19%	0,80%	3,98%
	kd	13,37%	1,75%	8,26%
Carto-SOM variant 3	ce	2,06%	0,23%	2,06%
	km	6,19%	0,80%	3,98%
	kd	13,37%	1,75%	8,26%
Carto-SOM variant 4	ce	2,06%	0,23%	2,06%
	km	6,19%	0,80%	3,98%
	kd	13,37%	1,75%	8,26%

Table 3 – Error evaluation made on the artificial dataset cartograms.

5.2. Portugal population based cartograms

We shall now describe the characteristics for the Portuguese dataset, the maps and quantitative results obtained.

5.2.1. Dataset characteristics

Portuguese population data were collected from 2001 Portuguese census in the Portuguese National Statistics Institute (INE) website at www.ine.pt. Portuguese administrative regions (*distritos*) were used as the spatial units for population aggregation. The geographic information was obtained from the Information Referencing Geographic Basis (originally *Base Geográfica de Referenciação de Informação*, BGRI). This is also an INE product used to georeference national statistics. Metadata for this product is explicit in the table bellow.

Projected Coordinate System:	Lisboa Hayford Gauss IGeoE
Projection:	Transverse Mercator
False Easting:	200000,00000000
False Northing:	300000,00000000
Central Meridian:	-8,13190611
Scale factor	1,00000000
Latitude Of Origin:	39,66666667
Linear Unit:	Meter (1,000000)

Table 4 – Portugal layer metadata

In Figure 42a population values for Portugal are presented. Two new datasets were created based on this information. In both cases the population values were interpolated for the buffer area (δ) of Portugal. In the first case a mean population density was used (Figure 42b) while in the second case the δ area was divided based on Voronoi regions and population densities were obtained from the nearest coastal region (Figure 42c).

Figure 42 – Portuguese population data.

In Figure 43 Portuguese population density is represented. From this representation we can see that, generally, the regions with a higher population in Portugal are also the high density regions. In Figure 43b the density in the δ area is equal to the mean of all Portuguese regions. In Figure 43c the color scale presented in the new δ areas formed shows that these regions have the same density as their Portuguese neighbouring regions.

Figure 43 – Portuguese population density.

Three different datasets were created based on these population datasets. The first dataset concerns the population data for Portugal (Figure 42a). The second and third population datasets include the δ area differing on the density population according to the example from Figure 42.

The reason that led to the creation of these datasets including Portugal's frame (δ area) is related to the fact that a two-dimensional SOM usually has a rectangular shape. The SOM is a matrix ($n \times m$) of units that will adapt to the training data. Thus in order to have training patterns in the δ area a frame was created in the area of study.

Based on the Portuguese regions population datasets (Figure 42) the training patterns were randomly created. The number of patterns created for each Portuguese region is proportional to its population value. As we can see in Figure 44 regions with high population values will have more patterns than those with low population. In the Portuguese case, coastal regions are the most populated and so more patterns were created in those regions.

Figure 44 – Different point datasets created from Portuguese 2001 population values.

5.2.2. Experimental settings

The population variable was aggregated in Portuguese administrative regions (*Distritos*). For each Carto-SOM variant we applied 6 tests using different SOM parameters. These parameters control the size of the SOM, the number of epochs used in the first and second training phases, the neighbourhood radius and the learning rate for the first and second training phases. A table showing the

SOM parameters used during training (Table 5) is presented. In order to evaluate the training phase, two quality measures were included in the same table. These measures are the quantization error (*qe*) and the topographic error (*te*) already explained in the section 3.

Variants	Test	Parameters								Error	
		<i>x</i>	<i>y</i>	<i>iter1</i>	<i>iter2</i>	<i>n1</i>	<i>n2</i>	<i>L1</i>	<i>L2</i>	<i>qe</i>	<i>te</i>
1	1	50	25	50	75	20	8	0,5	0,1	3243,40	0,0954
	2	75	50	50	75	20	8	0,5	0,1	1920,70	0,1350
	3	50	25	50	75	25	10	0,5	0,1	3356,10	0,1045
	4	75	50	50	75	25	10	0,5	0,1	2051,80	0,1168
	5	50	25	50	75	25	10	0,7	0,2	3062,70	0,0963
	6	75	50	50	75	25	10	0,5	0,1	1818,70	0,1285
2	1	50	25	50	75	20	8	0,5	0,1	5704,40	0,0419
	2	75	50	50	75	20	8	0,5	0,1	3389,50	0,0519
	3	50	25	50	75	25	10	0,5	0,1	5923,10	0,0384
	4	75	50	50	75	25	10	0,5	0,1	3569,70	0,0540
	5	50	25	50	75	25	10	0,7	0,2	5436,40	0,0445
	6	75	50	50	75	25	10	0,5	0,1	3235,80	0,0545
3	1	50	25	50	75	20	8	0,5	0,1	4673,80	0,1285
	2	75	50	50	75	20	8	0,5	0,1	2790,10	0,0807
	3	50	25	50	75	25	10	0,5	0,1	4786,50	0,1030
	4	75	50	50	75	25	10	0,5	0,1	2902,80	0,0816
	5	50	25	50	75	25	10	0,7	0,2	4544,50	0,1067
	6	75	50	50	75	25	10	0,5	0,1	---	---
4	1	50	25	50	75	8	5	0,5	0,1	3774,90	0,2016
	2	75	50	50	75	8	5	0,5	0,1	2294,10	0,1757
	3	50	25	50	75	10	8	0,5	0,1	4043,80	0,2145
	4	75	50	50	75	10	8	0,5	0,1	2408,70	0,2004
	5	50	25	50	75	8	5	0,7	0,2	3520,70	0,1718
	6	75	50	50	75	8	5	0,5	0,1	2062,60	0,2324

Table 5 – SOM parameters used in the several tests for Portugal; *x* and *y* are the number of units in *x* and *y* used in the SOM; *iter1* and *iter2* are the number of epochs for the first and second train (the unfolding and fine tune phases); *n1* and *n2* are the neighbourhood rate used in the first and second train and *L1* and *L2* are the first and second train learning rate. Also two quality parameters are presented, the *qe* (quantization error) and *te* (topographic error).

In Figure 45 we present for each variant the units mapped in the input space after the training phase. In Figure 45a and Figure 45d, for variants 1 and 4, units adapt to the training patterns created only inside Portugal boundaries (*ra*). In the

remaining variants (Figure 45b and Figure 45c) training patterns were created along the Portuguese domain area (D).

Figure 45 – Units mapped in the input space after the training process for each variant.

For the construction of the Dougenik cartogram, the geographic layer used was presented in the Figure 42a, using the variable population from 2001. The CartoCreator script (Schmid 2005) was, as in the artificial dataset, used in the ArcGIS environment. A set of 5 iterations was also selected.

In order to evaluate the training phase the qe and te were calculated for each test performed. For test 6 from variant 3 the qe and te calculation was not possible

due to computer¹ memory problems. In order to perform this test we need to increase the computer memory or change the standard SOM algorithm for qe and te calculation. Based on this table two graphs were build in order to evaluate the two measures (Figure 46).

Figure 46 – Quantization error and topographic error for Portuguese tests.

As we can see in Figure 46a quantization error is lower for the first variant, specifically for the second, fourth and sixth tests. In these tests a bigger (50 x 75 units) network was used meaning that the network could better adapt to the training patterns. Also in variant 1 the training patterns were only generated inside regions. This means a lower number of training patterns and consequently a better adaptation of the SOM. The second best result of qe is achieved for the variant 4. Again the number of training patterns used is lower (the same patterns dataset used in variant 1 is used) improving the results. In these variants where some units are excluded (units that are in the δ area) the number of units considered was lower than in the other variants.

In Figure 46b topographic error (te) is presented for each test. te has lower values for variant 2 meaning that this was the variant with less “cross-over” between units. For this test the highest values are achieved for the variant 4.

¹ Computer used was a Intel(R) Pentium (R) M processor 1,6 GHz, 768 MB RAM

5.2.3. Results

Carto-SOM and Dougenik's cartogram construction methods were, in this section applied to the Portuguese dataset producing several cartograms. The results are presented in two phases: in the first phase we present the cartograms while in the second phase we evaluate the cartogram quality based on the quantitative measures already presented.

5.2.3.1. Maps

The cartograms produced in this set of tests can be found in Appendix 1. In Figure 47 we present one cartogram result from each variant. Each cartogram was selected as the best for its variant based on the evaluations made in the next section. The first map is the original shape of Portugal using the Hayford-Gauss IPCC Lisbon projection. A choropleth map of population density is also present, representing the value of population through the color lightness.

a) Portugal's Hayford-Gauss IPCC projection

b) Portugal's population

c) Variant 1

d) Variant 2

e) Variant 3

f) Variant 4

Figure 47 – Portugal's population traditional maps and cartograms using the Carto-SOM methodology.

In Figure 48, the cartogram produced with Dougenik's method is presented. Five iterations were used to produce this cartogram.

Figure 48 – Portuguese population cartogram using Dougenik's cartogram algorithm.

5.2.3.2. Quantitative evaluation

As in the artificial dataset cartograms, some tests were performed in order to analyse the cartograms accuracy. In Table 6 we present 9 different error values to evaluate each cartogram global quality. A simple average error, a mean quadratic error and a weighed error were calculated for the cartogram error (*ce*), the Keim area error (*km*) and for the Kocmoud area error (*kd*).

As we can see from the table, the Carto-SOM has, generally, a lower error than the cartograms produced by Dougenik's method. When analysing the error achieved using the cartogram error (*ce*) the Carto-SOM seems to be a worse cartogram than Dougenik's cartogram. If we consider the Keim area error or the Kocmoud area error than the inverse occurs, *i.e.*, Carto-SOM produces similar or lower global cartogram errors.

In the Carto-SOM variants, for the Portuguese population dataset, we may consider the four cartogram variants very similar in terms of area error, suggesting that the Carto-SOM methodology is a robust method for producing cartograms.

	Error measures	se	mqe	we
Dougenik	ce	1,04%	0,33%	1,70%
	km	12,29%	3,80%	8,65%
	kd	31,16%	10,70%	18,64%
Carto-SOM variant 1	ce	1,16%	0,36%	1,99%
	km	12,38%	3,29%	10,45%
	kd	26,14%	6,85%	20,91%
Carto-SOM variant 2	ce	1,24%	0,46%	2,57%
	km	11,69%	3,23%	11,22%
	kd	26,89%	7,77%	22,37%
Carto-SOM variant 3	ce	1,22%	0,47%	2,48%
	km	12,54%	3,91%	10,69%
	kd	31,57%	10,72%	22,04%
Carto-SOM variant 4	ce	1,33%	0,51%	2,49%
	km	13,80%	4,00%	11,87%
	kd	32,40%	10,07%	23,85%

Table 6 – Tests made on the Carto-SOM and Dougenik cartograms using the Portuguese dataset.

In Figure 49 we present the *ce* for each region for the cartogram produced in variant 2. As we can see, in the Portuguese case, the error is very high for two regions (Lisbon and Oporto), and decreases for less populated areas.

Figure 49 – Distribution of the Portuguese cartogram error (*ce*) for variant 2.

5.3. USA population based cartograms

In this section we will present the characteristics for the USA dataset, and the maps and quantitative results obtained.

5.3.1. Dataset characteristics

The third dataset used to test the methodology is the population of the United States of America for the year of 2001. This data was collected from the ArcGIS CD installation from ESRI (ESRI 2005a). This information uses the GCS North American 1927 system of geographic coordinates. In order to have a map representation where the area is less distorted when compared to reality one proceeds to a data projection process. The projection system used was the Albers equal area (North America Albers Equal Conic Area). This projection system maintains the area of regions while distorting other characteristics such as shape and distance. Metadata for this product is explicit in the Table 7.

Projected Coordinate System:	North America Albers Equal Area Conic
Projection:	Albers
False Easting:	0,00000000
False Northing:	0,00000000
Central Meridian:	-96,00000000
Standard_Parallel_1:	20,00000000
Standard_Parallel_2:	60,00000000
Latitude Of Origin:	40,00000000
Linear Unit:	Meter (1,000000)

Table 7 – USA layer metadata

The process applied to Portugal's dataset was also used in the USA dataset to produce the training datasets. Thus we created three different training datasets. In the first one the patterns were created only in the regions area (ra) while on the other two the δ area was also populated. The difference between those two datasets refers to the density used for the δ area. While in the second pattern dataset a mean density of all the regions was used, in the third dataset new δ_i areas were created based on the neighbouring ra_i areas. Each of these new areas

was given a density similar to that of the neighbouring region, and training patterns were then created.

In Figure 50a population values for USA are presented. Two new datasets were created based on this information where the buffer areas were populated.

Figure 50 – USA population data.

In Figure 51 USA population density is represented. In Figure 51b the calculated density for the δ area is equal to the mean of USA population density. In Figure 51c we can see by the color scale that the new δ areas have the same density as its USA states' neighbours.

Figure 51 – USA population density.

Again as in the Portugal dataset, three different training datasets were created based on this polygon population data (Figure 52). Data patterns were randomly generated inside each polygon proportionally to its value of population.

Figure 52 – Different point datasets created from USA 2001 population values.

5.3.2. Experimental settings

Population variable was aggregated in states for the USA. Several tests were applied using different SOM parameters. For each Carto-SOM variant we applied 6 tests using different SOM parameters. These parameters control the size of the

SOM, the number of epochs used in the first and second train, the neighbourhood radius and the learning rate for the first and second training phases. The parameters used were the same applied in the Portuguese cartograms. A table showing the SOM training parameters used (Table 8) is presented. As in the Portuguese case study the quantization error (*qe*) and the topographic error (*te*) were calculated to evaluate the SOM training phase.

Variants	Test	Parameters								Error	
		<i>x</i>	<i>y</i>	<i>iter1</i>	<i>iter2</i>	<i>n1</i>	<i>n2</i>	<i>L1</i>	<i>L2</i>	<i>qe</i>	<i>te</i>
1	1	50	25	50	75	20	8	0,5	0,1	27617	0,16184
	2	75	50	50	75	20	8	0,5	0,1	---	---
	3	50	25	50	75	25	10	0,5	0,1	28069	0,16042
	4	75	50	50	75	25	10	0,5	0,1	---	---
	5	50	25	50	75	25	10	0,7	0,2	26975	0,16459
	6	75	50	50	75	25	10	0,5	0,1	---	---
2	1	50	25	50	75	20	8	0,5	0,1	57542	0,06696
	2	75	50	50	75	20	8	0,5	0,1	57542	0,06633
	3	50	25	50	75	25	10	0,5	0,1	61029	0,05740
	4	75	50	50	75	25	10	0,5	0,1	36350	0,06633
	5	50	25	50	75	25	10	0,7	0,2	54266	0,07419
	6	75	50	50	75	25	10	0,5	0,1	31871	0,07802
3	1	50	25	50	75	20	8	0,5	0,1	46079	0,08854
	2	75	50	50	75	20	8	0,5	0,1	26934	0,07423
	3	50	25	50	75	25	10	0,5	0,1	47102	0,08412
	4	75	50	50	75	25	10	0,5	0,1	27701	0,07510
	5	50	25	50	75	25	10	0,7	0,2	44674	0,09425
	6	75	50	50	75	25	10	0,5	0,1	26031	0,07207
4	1	50	25	50	75	8	5	0,5	0,1	33267	0,16227
	2	75	50	50	75	8	5	0,5	0,1	19747	0,17125
	3	50	25	50	75	10	8	0,5	0,1	34045	0,16463
	4	75	50	50	75	10	8	0,5	0,1	20420	0,16802
	5	50	25	50	75	8	5	0,7	0,2	32746	0,18475
	6	75	50	50	75	8	5	0,5	0,1	19237	0,16558

Table 8– SOM parameters used in the several tests for the USA; *x* and *y* are the number of units in *x* and *y* used in the SOM; *iter1* and *iter2* are the number of epochs for the first and second train (the unfolding and fine tune phases); *n1* and *n2* are the neighbourhood rate used in the first and second train and *L1* and *L2* are the first and second train learning rate. Also two quality parameters are presented, the *qe* (quantization error) and *te* (topographic error).

In Figure 53 we present for each variant the units mapped in the input space after training (units are presented in red while the training patterns are blue dots). As

we can see in Figure 53a and Figure 53d variants 1 and 4 units adapt to the training patterns created only inside USA boundaries. In the other two variants (Figure 53b and Figure 53c) training patterns were created in the USA domain area and therefore units will occupy this space.

Figure 53 – Units mapped in the input space after the training process for each variant

Quantization error (qe) and topographic error (te) were calculated for each test as already presented in Table 8. For the same reason as presented in Portugal's dataset tests, due to computer memory limitations, it was not possible to calculate the qe and te in some cases (tests 2, 4, and 6 from variant 1). Based on this table two graphs were built in order to analyse qe and te evolution (Figure 54).

Figure 54 – Quantization error and topographic error for USA tests.

As we can see in Figure 54 the qe is lower for the first and fourth variant. As already pointed out in tests applied to Portugal's datasets this is caused by the fact that a lower number of training patterns is used in these variants. Variant 3 presents the highest qe in all tests. In this variant, a bigger dispersion on the training patterns was used and consequently the network adaptation is a harder task to perform. The te error has higher values for the first and fourth variant and lower values for the second and third variants. We can conclude that in the datasets where training patterns were generated for the δ area the network folding has a lower probability of occurring.

5.3.3. Results

Carto-SOM and Dougenik's cartogram construction methods were, in this section applied to the USA dataset producing several cartograms. The results are presented in two phases: in the first phase we present the cartograms while in the second phase we evaluate the cartogram quality based on the quantitative measures already presented.

5.3.3.1. Maps

The cartograms produced during this set of tests can be found in Appendix 1. In Figure 55 we present one cartogram produced from each variant. Each cartogram was selected as the best for its variant based on the statistics presented on the quantitative evaluation section. The first map is the original shape of the USA

using Albers Equal Conic Area projection. A choropleth map of population density, is also present, representing the value of population through the color lightness.

Figure 55 – USA population cartograms.

In Figure 56 the cartograms produced using Dougenik's method are presented. Five iterations were used to produce this cartogram.

Figure 56 – Dougenik Cartogram for the USA population.

5.3.3.2. Quantitative evaluation

As in other datasets used in this dissertation, some tests were performed in order to analyse the cartograms' accuracy. In Table 9 we present for Dougenik's and Carto-SOM cartograms, some error values evaluating each cartogram's global quality. A simple average error, a mean quadratic error and a weighed error were calculated for the cartogram error (ce), the Keim area error (km) and for the Kocmoud area error (kd).

As we can see from the table, the Carto-SOM has, for variant 2, lower errors than the cartograms produced by Dougenik's method. Variant 3 cartograms produces similar errors to Dougenik's and variant 1 and four are generally worse. For this specific dataset, Carto-SOM variant 2 proves to be an effective cartogram construction methodology.

	Error measures	se	mqe	we
Dougenik	ce	0,48%	0,10%	0,83%
	km	16,02%	3,08%	11,09%
	kd	45,01%	10,74%	23,24%
Carto-SOM variant 1	ce	0,69%	0,13%	1,06%
	km	20,75%	3,65%	16,02%
	kd	53,47%	11,38%	32,90%
Carto-SOM variant 2	ce	0,33%	0,07%	0,54%
	km	9,31%	1,65%	8,13%
	kd	19,55%	3,52%	15,98%
Carto-SOM variant 3	ce	0,55%	0,11%	0,85%
	km	17,31%	3,17%	13,19%
	kd	44,43%	9,55%	26,64%
Carto-SOM variant 4	ce	0,67%	0,13%	0,98%
	km	23,87%	4,24%	16,78%
	kd	54,90%	10,34%	32,35%

Table 9 – Error evaluation made on the USA dataset cartograms.

In Figure 57 we present the *ce* for each region for the cartogram produced in variant 2. As we can see, in this case the error is, except for the Texas state, higher for the less density populated states.

Figure 57 – Distribution of the USA cartogram error (*ce*) for variant 2.

5.4. Comparative analysis and results

In this section an overview on the produced cartograms is presented. Some considerations are also made on the Carto-SOM methodology, based on the quantitative evaluation. Also, some SOM training issues are analysed in the map folding section.

5.4.1. Map Folding

One of the most difficult problems in training a SOM is the unfolding process. Although it is not easy to identify unfolded maps of very high dimensional data, a good choice of learning parameters can greatly reduce the risk that they will occur. In the cases presented in this dissertation, since a two dimensional input space is used, it is possible to check visually if the SOM unfolds correctly.

In variant 4, due to the exclusion of the units from the δ area and because the coast is sometimes very irregular the network folds tend to increase. In Figure 58 we present an example extracted from the Portuguese dataset.

Figure 58 – Network folds in variant 4 due to the irregular shape of the border and to ocean units' immobilization.

Due to this higher probability to fold of variant 4 we decided to change some parameters in order to minimize this error. Thus, instead of using a radius of 25 or 20 for the first training phase and 10 or 5 for the second we used 10 or 8 and 8 or 5 respectively. This parameter controls the mobility of the network in the initial phase of training. By decreasing it we try to give less initial mobility to the units, thus decreasing the probability of folding. In Figure 59 we present the resulting cartograms when using a smaller and a bigger radius for the Portuguese case study. Because the SOM in the Figure 59a suffers a big amount of folds the cartogram is geographically inverted with East regions mapped in the West and *vice-versa*. When using a smaller radius in the training process (Figure 59b) the probability of getting this type of inverted cartogram is smaller.

a) Radius equal to 25 and 10 for the first and second training phase

b) Radius equal to 10 and 8 for the first and second training phase

Figure 59 – Change of the radius parameter. On both tests a 50 x 25 unit's network was used.

5.4.2. Visual comparison overview

By analysing visually the cartograms produced (Appendix 1) in Carto-SOM methods we conclude that the first variant cartogram is not an effective

cartogram since the original regions shape is changed and the “recognizability” is lost. Variant 2 has the best quantitative evaluation and visually is also a good representation since it is possible to recognize the original regions on the cartogram. In variant 3, although quantitative parameters present acceptable results, this representation loses the original shape becoming visually difficult to interpret. Finally, in variant 4, due to the borders maintenance, a very good visual representation is achieved.

Dougenik cartograms also present good visual results, but have the disadvantage of performing radial transformations on the shape of the states. If several iterations are performed the shape will become circular, in the cases where the original density is higher than the mean density (Florida’s case), or will be smashed on the opposite situation (states with a low value of population, e.g. Wyoming, Kansas, South Dakota, etc.). With a high number of iterations the cartogram error will be lower, but due to these radial transformations the cartogram tends to become difficult to interpret.

5.4.3. Quantitative evaluation overview

In Table 10 we present the cartogram error for each cartogram produced in this dissertation. From Dougenik’s and Carto-SOM cartograms 9 different error values were produced, evaluating each cartogram’s global quality. A simple average error, a mean quadratic error and a weighed error were calculated for the cartogram error (*ce*), the Keim area error (*km*) and for the Kocmoud area error (*kd*).

		Artificial			USA			Portugal		
		<i>se</i> (%)	<i>mge</i> (%)	<i>we</i> (%)	<i>se</i> (%)	<i>mge</i> (%)	<i>we</i> (%)	<i>se</i> (%)	<i>mge</i> (%)	<i>we</i> (%)
Dougenik	<i>ce</i>	5,42	0,60	5,42	0,48	0,10	0,83	1,04	0,33	1,70
	<i>km</i>	14,64	1,83	10,03	16,02	3,08	11,09	12,29	3,80	8,65
	<i>kd</i>	35,08	4,58	21,67	45,01	10,74	23,24	31,16	10,70	18,64
Carto-Som variant 1	<i>ce</i>	2,06	0,23	2,06	0,69	0,13	1,06	1,16	0,36	1,99
	<i>km</i>	6,19	0,80	3,98	20,75	3,65	16,02	12,38	3,29	10,45
	<i>kd</i>	13,37	1,75	8,26	53,47	11,38	32,90	26,14	6,85	20,91
Carto-Som variant 2	<i>ce</i>	2,06	0,23	2,06	0,33	0,07	0,54	1,24	0,46	2,57
	<i>km</i>	6,19	0,80	3,98	9,31	1,65	8,13	11,69	3,23	11,22
	<i>kd</i>	13,37	1,75	8,26	19,55	3,52	15,98	26,89	7,77	22,37
Carto-Som variant 3	<i>ce</i>	2,06	0,23	2,06	0,55	0,11	0,85	1,22	0,47	2,48
	<i>km</i>	6,19	0,80	3,98	17,31	3,17	13,19	12,54	3,91	10,69
	<i>kd</i>	13,37	1,75	8,26	44,43	9,55	26,64	31,57	10,72	22,04
Carto-Som variant 4	<i>ce</i>	2,06	0,23	2,06	0,67	0,13	0,98	1,33	0,51	2,49
	<i>km</i>	6,19	0,80	3,98	23,87	4,24	16,78	13,80	4,00	11,87
	<i>kd</i>	13,37	1,75	8,26	54,90	10,34	32,35	32,40	10,07	23,85

Table 10 – Global error evaluation made on the Carto-SOM and Dougenik cartograms.

In general the results achieved by Carto-SOM are superior to the ones of Dougenik's method. Particularly, if we consider only variant 2 of the Carto-SOM, which is the one that produces better visual results, then there is only one case where this does not occur (cartogram error (*ce*), Keim and Kocmoud error using a weighed average, for the Portuguese dataset it is not better the Dougenik's). Making a general overview on the presented results, we can argue that the artificial dataset cartogram, produces the lowest errors when using the Carto-SOM methodology. A similar result is achieved in the USA cartograms, especially in variant 2, as they present results that are always better than those produced with the Dougenik method. One possible explanation for this fact could be related

with the magnification factor (Cottrell, Fort et al. 1998) applied by the SOM. As already described, the SOM tends to underestimate high density regions and overestimate low density areas. Since, in the Portuguese case study, we have two major regions (Lisbon and Oporto) with a very high population density, Carto-SOM underestimates these regions producing cartogram with a higher error on this regions (cartogram error for each region are presented in appendix 5).

In Table 11 we present a summary of the errors, using only Keim area error, comparing the cartograms produced by Dougenik and the Carto-SOM variant 2 methods.

		Artificial			USA			Portugal		
		<i>se</i> (%)	<i>mqe</i> (%)	<i>we</i> (%)	<i>se</i> (%)	<i>mqe</i> (%)	<i>we</i> (%)	<i>se</i> (%)	<i>mqe</i> (%)	<i>we</i> (%)
Dougenik	<i>km</i>	14,64	1,83	10,03	16,02	3,08	11,09	12,29	3,80	8,65
Carto-Som variant 2	<i>km</i>	6,19	0,80	3,98	9,31	1,65	8,13	11,69	3,23	11,22

Table 11 – Keim area error calculated for the Carto-SOM variant 2 and Dougenik cartograms

Analysing this table allows us to argue that Carto-SOM variant 2 produces very good results, even when compared with Dougenik. This variant is also, visually the most similar to Dougenik, having the advantage of not using radial operations (operations leading to circular shapes).

6. Conclusion

In this dissertation we have presented a general method for constructing density-equalizing projections or cartograms providing a tool for geographic data presentation and analysis. The proposed methodology uses SOM units to represent the distribution of the selected variable (in our work we used population). Based on this distribution we distort the original shapes in such a way that the final region densities in the cartogram are closer to the mean density.

Cartograms were made using three different datasets. We first used an artificial dataset, to get some preliminary evaluation on the method, and then we used Portuguese 2001 population dataset and USA 2001 population dataset. In order to achieve a better cartogram construction methodology several variants of the initial idea were implemented. Six tests, changing the SOM parameters, were performed in each Carto-SOM variants (four variants).

In order to compare the results with other cartograms the Dougenik cartogram construction algorithm was used to produce cartograms for the mentioned datasets.

In order to evaluate the cartograms quality, tests were also evaluated using quantitative measures. These tests indicate that the cartograms created by the Carto-SOM methodology are good and accurate representations of the study variables.

Analyzing the quantitative results we can conclude that Carto-SOM proves to be an efficient cartogram construction algorithm, presenting in most cases results better than the benchmark methodology. This comparison is made on several different measures proving the reliability of the proposed method.

Although not proved rigorously in this dissertation, the increase of units in the Carto-SOM has the same effect that the increase of iterations in Dougenik's cartograms, *i.e.* the obtained cartograms are better approximations to the ideal cartogram.

Although the results presented high-quality cartograms, in most cases we think that variant 2 represents the best solution in cartogram construction. Variant 1, is not a real cartogram, since due to changes in the shape of regions, it is not possible to recognize the original map. Variant 3 is a particular case of variant 2, since they differ only in the δ area population. Variant 3 produces in most cases a larger error than variant 2 leading us to believe that variant 2 will in general be better. Finally for variant 4, although we have compared this variant with the others we think they aren't fully comparable. The main reason that leads us to this conclusion is related with the number of units used on the SOM. In this variant, since all units that are initialized in the δ area will be excluded from the process, the number of units used for this variant is smaller than in the other variants. In order to, efficiently, evaluate this variant, tests with a higher number of units should be performed in future work.

Another advantage we can list in Carto-SOM methodology has to do with the input dataset. The proposed method has the ability to use continuous information, i.e. although a polygon dataset was used (USA states and Portuguese administrative regions) it is possible to use a raster or point datasets.

It is important to consider the differences that exist in terms of programming development and implementation. The Carto-SOM constitutes only a prototype and it was not developed with computation efficiency issues in mind. This explains that while computational efficiency tests were not rigorously made in this work, we believe that Dougenik's implementation in the ArcGIS environment (Schmid 2005) is, at this phase, a more efficient solution. The implementation of the Carto-SOM methodology in C language will surely improve the computational efficiency of the method.

Being a computation intensive task, building cartograms can benefit from improvements in computing processing. Parallel processing constitutes one of the most promising solutions for computer intensive tasks. Nevertheless, it is not always easy and efficient to adapt algorithms to the parallel processing paradigm. In this context, the Carto-SOM presents important advantages over Dougenik's method, as it is easily adapted to parallel processing, contrary to Dougenik's method. This can be a major issue in years to come as parallel processing

becomes increasingly available. This will not only allow us to drastically reduce the processing time but also to increase the number of SOM units that can be used.

Although the proposed algorithm is an effective cartogram generation algorithm, promising directions for further research still remain. It would be interesting to include in the algorithm methods for computing the final cartogram shape giving it a more realistic boundary instead of using cells. Another improvement in this method would be to increase of the number of units used in the SOM training.

References

- Baçaõ, F. and V. Lobo (2005). One dimensional self-organizing maps to optimize marine patrol activities. IEEE Conference, (Brest, France: IEEE).
- Baçaõ, F., V. Lobo and M. Painho (2005a). Applications of different self-organizing maps variants to geographical information science problems. Self-Organizing Maps: applications in geographic information science. ed. by P. Agarwal and A. Skupin, to be published by Wiley.
- Baçaõ, F., V. Lobo and M. Painho (2005b). Applying genetic algorithms to zone design. Soft Computing, **9**: 341-348.
- Baçaõ, F., V. Lobo and M. Painho (2005c). The self-organizing map, the Geo-SOM and relevant variants for geosciences. Computers and Geosciences, **31**: 155-163.
- Bauer, H.-U., R. Der and M. Herrmann (1996). Controlling the Magnification Factor of Self-Organizing Feature Maps. Neural Computation **8**: 757-771.
- Bishop, C. M., M. Svensen and C. K. I. Williams (1997). Magnification factors for the SOM and GTM algorithms. Proceedings 1997 Workshop on Self-Organizing Maps, (Helsinki: Helsinki University of Technology), 333-338.
- Changming, D. and L. Lin (1999). Constructing contiguous area cartogram using arcview avenue. The proceedings of Geoinformatics'99 Conference, (Ann Arbor: The Association of Chinese Professionals in GIS), 1-7.
- Cottrell, M., J. C. Fort and G. Pages (1998). Theoretical Aspects of the SOM algorithm. Neurocomputing, Elsevier **21**: 119-138.
- Dorling, D. (1994). Cartograms for Visualizing Human Geography. Visualization in Geographical Information Systems. ed. by D. J. U. H.M. Hearshaw. New York, John Wiley & Sons: 85-102.

Dougenik, J., N. Chrisman and D. Niemeyer (1985). An algorithm to construct continuous area cartograms. Professional Geographer, **37**: 75-81.

ESRI (2005a) ArcGIS Desktop (version 9.1) [Software]. (Redlands: ESRI)

ESRI (2005b) MapObjects-Windows Edition (version 2.3) [Software]. (Redlands: ESRI)

Gastner, M. and M. E. J. Newman (2004). Diffusion-based method for producing density-equalizing maps. Proceedings of the National Academy of Sciences of the United States of America), 7499-7504.

Gusein-Zade, S. and V. Tikunov (1993). A new technique for constructing continuous cartograms. Cartography and Geographic Information Systems, **20 (3)**: 167-173.

Haese, K. and G. J. Goodhill (2001). Auto-SOM: recursive parameter estimation for guidance of self-organizing feature maps. Neural Computation, **13**: 595-619.

Heilmann, R., D. A. Keim, C. Panse and M. Sips (2004). RecMap: rectangular map approximations. IEEE Symposium on Information Visualization, (Austin, Texas: IEEE), 33-40.

Henriques, R. and F. Bação (2004). soMGis: uma ferramenta para construir regiões. V Congresso da Geografia Portuguesa, (Campus de Azurém, Guimarães: Associação Portuguesa de Geógrafos),

Henriques, R., F. Bação and V. Lobo (2005). Carto-SOM: cartogram creation using self-organizing maps. 14th European Colloquium on Theoretical and Quantitative Geography [CD-ROM], (Tomar: e-Geo-Geography and Regional Planning Research Centre),

House, D. and C. Kocmoud (1998). Continuous cartogram construction. Proceedings of IEEE Visualization, (Research Triangle Park, NC: IEEE), 197-204.

ICA (International Cartographic Association) (1973). Multilingual dictionary of technical terms in cartography. Wiesbaden, Franz Steiner Verlag.

Kaski, S. and T. Kohonen (1996). Exploratory data analysis by the self-organizing map: structures of welfare and poverty in the world. Neural Networks in Financial Engineering. ed. by N. Apostolos-Paul, Yaser Refenes, Yaser Abu-Mostafa, John Moody and A. Weigend. Singapore, World Scientific: 498-507.

Kaski, S. and K. Lagus (1996). Comparing self-organizing maps. Proceedings of ICANN96 International Conference on Artificial Neural Networks, (Germany: Springer), 809-814.

Kaski, S., J. Nikkilä and T. Kohonen (1998). Methods for interpreting a self-organized map in data analysis. Proceedings of ESANN'98, 6th European Symposium on Artificial Neural Networks, (Bruges, Belgium: D-Facto), 185-190.

Keim, D. A., S. C. North and C. Panse (2004). CartoDraw: a fast algorithm for generating contiguous cartograms. IEEE Transactions on Visualization and Computer Graphics, **10(1)**: 95-110.

Keim, D. A., S. C. North and C. Panse (2005). Medial-axes based cartograms. IEEE Computer Graphics and Applications, **25(3)**: 60-68.

Keim, D. A., S. C. North, C. Panse and J. o. Schneidewind (2002). Efficient cartogram generation: a comparison. Proceedings of the IEEE Symposium on Information Visualization 2002: 33.

Kim, M. Y. and H. Cho (2004). Three-dimensional map building for mobile robot navigation environments using a self-organizing neural network. Journal of Robotic Systems **21**: 323-343.

Kirkpatrick, S., C. D. Gelatt Jr. and M. P. Vecchi (1983). Optimization by Simulated Annealing. Science **220**: 671-680.

Kocmoud, C. (1997). Constructing continuous cartograms: a constraint-based approach. Master of Science, Texas A&M University, Texas.

Kohonen, T. (1982). Self-organizing formation of topologically correct feature maps. RecMap: rectangular map approximations **43 (1)**: 59-69.

Kohonen, T. (1995). Self-Organizing Maps. (1st edition) Berlin, Springer.

Kohonen, T. (2001). Self-Organizing Maps. (3rd edition) Berlin, Springer.

Lawler, E., J. Lenstra and A. Rinnooy Kan (1985). The traveling salesman problem: a guided tour of combinatorial optimization. New York, Wiley.

Lobo, V. (2002). Ship noise classification: a contribution to prototype based classifier design. Doctor in Informatics, University of Lisbon, Lisbon.

Merrill, D. W. (2001). Use of a density equalizing map projection in analysing childhood cancer in four California counties. Statistics in Medicine, **20**: 1499-1513.

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller (1953). Equation of state calculations by fast computing machines. Journal of Chemical Physics **21**: 1087-1092.

Mitchell, T. (1997). Machine Learning. McGraw-Hill.

National Atmospheric Deposition Program (2003). Lab pH (measured at the Central Analytical Laboratory) ([url: http://www.geo.ed.ac.uk/home/ded.html](http://www.geo.ed.ac.uk/home/ded.html), accessed in 04-02-2004).

- NCGIA (2002). Cartogram Central (url: http://www.ncgia.ucsb.edu/projects/Cartogram_Central/, accessed in 10-03-2005).
- Olson, J. (1976). Noncontiguous area cartograms. The Professional Geographer **28**, 4: 371-380.
- Raisz, E. (1934). The rectangular statistical cartogram. The Geographical Review **24**: 292-296.
- Ritter, H. and K. Schulten (1986). On the stationary state of Kohonen's self-organizing sensory mapping. Biological Cybernetics **54**: 99-106.
- Robinson, A. H., J. L. Morrison, P. C. Muehrcke, A. J. Kimerling and S. C. Guptill (1984). Elements of Cartography. New York, John Wiley & Sons.
- Schmid, S. (2005). CartoCreator: cartogram creator (url: <http://arcscripts.esri.com/details.asp?dbid=14226>, accessed in 10-08-2005).
- Seiffert, U. and B. Michaelis (1995). Three-dimensional self-organizing maps for classification of image properties. 2nd New Zealand Two-Stream International Conference on Artificial Neural Networks and Expert Systems(ANNES'95), (Dunedin, New Zealand: IEEE Computer Society),
- Silva, M. A. (2004). Mapas auto-organizáveis na análise exploratória de dados geoespaciais multivariados. Mestrado, Instituto Nacional de Pesquisas Espaciais, São José dos Campos.
- Silva, N. C. and N. C. Rosa (2002). Estimative of SOM learning parameters using genetic algorithms. 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002), (Orlando, FL USA: SCI 2002), 31-36.
- Skupin, A. (2003). A novel map projection using an artificial neural network. Proceedings of 21st International Cartographic Conference, (Durban, South Africa: ICC), 1165-1172.

Tobler, W. (1973). A continuous transformation useful for districting. Annals, New York Academy of Sciences **219**: 215-220.

Tobler, W. (1979). Cartograms and cartosplines. Proceedings, Workshop on Automated Cartography and Epidemiology, (Washington, D.C.: Department of Health, Education and Welfare), 53-58.

Tobler, W. (1986). Pseudo-cartograms. The American Cartographer **13,1**: 43-50.

Tobler, W. (2004). Thirty-five years of computer cartograms. Annals, Assoc. American Geographers, **94(1)**: 58-73.

Tobler, W. (2005). What is a cartogram? (url: http://www.geog.ucsb.edu/~tobler/publications/pdf_docs/cartography/projections/cartograms/A_Cartogram_is.pdf, accessed in 01-02-2005).

Ultsch, A. and H. P. Siemon (1990). Kohonen's self-organizing neural networks for exploratory data analysis. Proceedings of the International Neural Network Conference, (Paris: Kluwer), 305-308.

Vesanto, J. (2000). Using SOM in data mining. Licenciate, Helsinki University of Technology, Espoo, Finland.

Vesanto, J., J. Himberg, E. Alhoniemi and Parhankangas (1999). Self-organizing map in Matlab: the SOM Toolbox. Proceedings of the Matlab DSP Conference, (Espoo, Finland: Comsol Oy.), 35-40.

Appendixes

Appendix 1 - Cartograms

Portuguese cartograms

Figure A1. 1 - Variant 1 final cartograms for Portugal.

Figure A1. 2 - Variant 2 final cartograms for Portugal.

Figure A1. 3 - Variant 3 final cartograms for Portugal.

Figure A1. 4 - Variant 4 final cartograms for Portugal.

USA cartograms

Figure A1. 6 - Variant 1 final cartograms for USA.

Figure A1. 7 - Variant 2 final cartograms for USA.

Figure A1. 8 - Variant 3 final cartograms for USA.

Figure A1. 9 - Variant 4 final cartograms for USA.

Figure A1. 10 – Dougenik cartograms for USA dataset.

Appendix 2 - Matlab routines

```
function carto_cartograma_semMar(xNeurons,yNeurons,niterations_1,...
niterations_2,radius_ini_1,radius_ini_2,alpha_ini_1,alpha_ini_2);

% Algorithm to train the SOM using only the units inside the study area

disp('Iniciando...');

% Batch para fazer cartogramas

%load USA txt file
load usa.txt -ascii
[num_dados num_features]=size(usa);
num_features=num_features-1; % os ficheiros incluem labels

%create Som Data
sD=som_data_struct(usa(:,1:2));
clear usa;

#####
%Som parameters: only if we are using this code as a macro
#####
% xNeurons=50;
% yNeurons=25;
% niterations_1 = 2;
% niterations_2 = 2;
% radius_ini_1 = 8;
% radius_ini_2 = 5;
% alpha_ini_1 = 0.5;
% alpha_ini_2 = 0.1;
#####
#####

%load DADOS temporarios (neuronios com labels) file
% aqui vou carregar dados com labels que servira para identificar os
% neuronios que estao no mar; assim posso treinar apenas os neuronios que
% estao em terra
% por defeito estes dados tem:
% label 0 para as zonas em terra
% e label -1 para o mar

%carrego ficheiro
load N.txt -ascii;

%create Som Data
sDn=som_data_struct(N(:,1:2));

%apanho os labels
n_labels = N(:,num_features+1);
labs = num2cell(n_labels);
[n1,n2]=size(n_labels);
sDn.labels = mat2cell(n_labels,ones(1,n1),[n2]); % criar lista de labels

%substituir os labels -1 por ""
for k = 1:length(sDn.labels)
    if (sDn.labels{k,1}== -1)
        sDn.labels{k,1} = "";
```

```

    end
end

%initialize som
sM1 = som_randinit_roberto( sD, 'msize',[xNeurons yNeurons],'rect','sheet');
sM1.neigh = 'bubble';
sM1=som_autolabel(sM1,sDn,'vote');

%plot
%figure(1);
%som_show(sM1,'empty','Labels')
%som_show_add('label',sM1, 'TextSize', 12)
%pause % Strike any key to add labels...

% train SOM
disp('comecei a primeira fase');
sM1 = som_seqtrain_roberto(sM1,sD,...
    'radius_ini',radius_ini_1,...
    'radius_fin',0,...
    'alpha_ini',alpha_ini_1,...
    'trainlen', niterations_1,...
    'epochs');
    %'epochs', 'tracking', 3);

h = figure(2);
title('Treino 1');
plot(sD.data(:,1),sD.data(:,2),'b. ');
hold on;
som_grid(sM1,'Coord',sM1.codebook,'LineColor', [0.9 0.9 0.9], 'Marker','.', 'MarkerColor' ,[0.9 0.1
0.1]);
drawnow;

%save figure
%saveas(h,'Testes\treino1','jpg')
saveas(h,'Testes\treino1')

%pause % Strike any key to add labels...

disp('comecei a segunda fase');
sM1 = som_seqtrain_roberto(sM1,sD,...
    'radius_ini',radius_ini_2,...
    'radius_fin',0,...
    'alpha_ini',alpha_ini_2,...
    'trainlen', niterations_2,...
    'epochs');
    %'epochs', 'tracking', 3);

h=figure(3);
title('Treino 2');
plot(sD.data(:,1),sD.data(:,2),'b. ');
hold on;

som_grid(sM1,'Coord',sM1.codebook,'LineColor', [0.9 0.9 0.9], 'Marker','.', 'MarkerColor' ,[0.9 0.1
0.1]);
drawnow;

%save figure
saveas(h,'Testes\treino2')
%pause % Strike any key to add labels...

% progress

```

```

disp('autolabeling');

load usa.txt -ascii
disp('autolabeling 1');

usa_labels=usa(:,num_features+1);
%usa_labels=num2str(usa_labels);
[n1,n2]=size(usa_labels);
disp('autolabeling 2');

label_cell=mat2cell(usa_labels,ones(1,n1),[n2]); % criar lista de labels

sMlixo=som_randinit_roberto( sD, 'msize',[xNeurons yNeurons],'rect','sheet');
disp('autolabeling 3');

sMar=sM1.labels;
sM1.labels=sMlixo.labels;
clear sMlixo;
clear usa;
clear usa_labels;
disp('autolabeling 4');

sD.labels=label_cell;
sM1=som_autolabel(sM1,sD,'vote');
clear label_cell;
disp('autolabeling 5');

M= labels2mat(sM1.labels);

%Alteracao do colormap
jet1=jet;
jet1(1,:)= [1 1 1];

% progresso
disp('plot...');
figure(4);
title('Cartograma');
outMat=reshape(M,sM1.topol.msize(1),sM1.topol.msize(2));
outMat=[outMat ; zeros(1,sM1.topol.msize(2))];
outMat=[outMat zeros(sM1.topol.msize(1)+1,1)];
h = pcolor(flipud(outMat));
%h = pcolor(outMat);
COLORMAP(jet1);

%save figure
saveas(h,'Testes\carto')

%#####
% Atribuir um label ao neuronios que nao sao mar e que estao sem label
%#####

l_c = cellfun('isempty',sMar);
%indices dos neuronios de terra
iTerra=find(l_c == 0 );
iMar=find(l_c ~= 0 );

%Bmus=som_bmus(sD,M(iTerra));
Bmus=carto_nearestDado (sM1.codebook, sD);

%Bmus=cell2mat(sD.labels(Bmus));
Bmus(iMar)=0;

```

```

figure(5);

outMat=reshape(Bmus,sM1.topol.msize(1),sM1.topol.msize(2));
outMat=[outMat ; zeros(1,sM1.topol.msize(2))];
outMat=[outMat zeros(sM1.topol.msize(1)+1,1)];
h = pcolor(flipud(outMat));
%h = pcolor(outMat);
title('Cartograma com BMUs ao contrario');
COLORMAP(jet1);

%save figure
saveas(h,'Testes\carto1')
#####

%save labels to ascii
save 'testes\labels.txt' M -ascii;

%save BMUs to ascii
save 'testes\BMUs.txt' Bmus -ascii;

#####
%calculate area for each state
Estados=labels2mat(sD.labels);
EstadoMin=min(Estados);
EstadoMax=max(Estados);
DeltaX=range(sD.data(:,1))/xNeurons;
DeltaY=range(sD.data(:,2))/yNeurons;

EstadosDiferentes=unique(Estados);
area=zeros(length(EstadosDiferentes),3);

for estado=EstadoMin:EstadoMax
    area(estado,1)= estado;
    area(estado,2)=length(find(Bmus == estado));
    area(estado,3)=area(estado,2)*DeltaX*DeltaY;
end;

save 'testes\contagem.txt' area -ASCII;
#####

%quantization and topographic error
[qe , te] = som_quality(sM1,sD);

%save parameters to ascii
%A = [ xNeurons yNeurons ; niterations_1 niterations_2 ; radius_ini_1 radius_ini_2 ; alpha_ini_1
alpha_ini_2 ; qe te];
%save 'testes\param.txt' A -ASCII;

fid = fopen('testes\parametros.txt','w');
fprintf(fid,'%s \t %3g\n','xNeurons' , xNeurons);
fprintf(fid,'%s \t %3g\n','yNeurons' , yNeurons);
fprintf(fid,'%s \t %5g\n','niterations_1' , niterations_1);
fprintf(fid,'%s \t %5g\n','niterations_2' , niterations_2);
fprintf(fid,'%s \t %5g\n','radius_ini_1' , radius_ini_1);
fprintf(fid,'%s \t %5g\n','radius_ini_2' , radius_ini_2);
fprintf(fid,'%s \t %5g\n','alpha_ini_1' , alpha_ini_1);
fprintf(fid,'%s \t %5g\n','alpha_ini_2' , alpha_ini_2);
fprintf(fid,'%s \t %5.5g\n','qe' , qe);
fprintf(fid,'%s \t %5.5g\n','te' , te);

```

```
fclose(fid);
```

```
disp('End');
```

```
beep;
```

```
%save usa_total
```

```
function carto_cartograma_comMar(xNeurons,yNeurons,niterations_1,...  
niterations_2,radius_ini_1,radius_ini_2,alpha_ini_1,alpha_ini_2);
```

```
% algorithm using all units in the network
```

```
disp('Iniciando...');
```

```
% Batch para fazer cartogramas
```

```
%load USA txt file
```

```
load usa.txt -ascii
```

```
[num_dados num_features]=size(usa);
```

```
num_features=num_features-1; % os ficheiros incluem labels
```

```
%create Som Data
```

```
sD=som_data_struct(usa(:,1:2));
```

```
clear usa;
```

```
%carrego ficheiro
```

```
load N.txt -ascii;
```

```
%create Som Data
```

```
sDn=som_data_struct(N(:,1:2));
```

```
%#####
```

```
%Som parameters
```

```
%#####
```

```
% xNeurons=75;
```

```
% yNeurons=50;
```

```
% niterations_1 = 1;
```

```
% niterations_2 = 1;
```

```
% radius_ini_1 = 25;
```

```
% radius_ini_2 = 10;
```

```
% alpha_ini_1 = 0.5;
```

```
% alpha_ini_2 = 0.1;
```

```
%#####
```

```
%#####
```

```
%initialize som
```

```
sM1 = som_randinit_roberto( sD, 'msize',[xNeurons yNeurons], 'rect', 'sheet');
```

```
sM1.neigh = 'bubble';
```

```
%sM1=som_autolabel(sM1,sDn,'vote');
```

```
%plot
```

```
%figure(1);
```

```
%som_show(sM1,'empty','Labels')
```

```
%som_show_add('label',sM1, 'TextSize', 12)
```

```
%pause % Strike any key to add labels...
```

```
% train SOM
```



```

disp('comecei a primeira fase');
sM1 = som_seqtrain(sM1,sD,...
    'radius_ini',radius_ini_1,...
    'radius_fin',0,...
    'alpha_ini',alpha_ini_1,...
    'trainlen', niterations_1,...
    'epochs');
    %'epochs', 'tracking', 3);

h = figure(2);
title('Treino 1');
plot(sD.data(:,1),sD.data(:,2),'b. ');
hold on;
som_grid(sM1,'Coord',sM1.codebook,'LineColor', [0.9 0.9 0.9], 'Marker','.', 'MarkerColor' ,[0.9 0.1
0.1]);
drawnow;

%save figure
saveas(h,'Testes\treino1')

%pause % Strike any key to add labels...

disp('comecei a segunda fase');
sM1 = som_seqtrain(sM1,sD,...
    'radius_ini',radius_ini_2,...
    'radius_fin',0,...
    'alpha_ini',alpha_ini_2,...
    'trainlen', niterations_2,...
    'epochs');
    %'epochs', 'tracking', 3);

h=figure(3);
title('Treino 2');
plot(sD.data(:,1),sD.data(:,2),'b. ');
hold on;

som_grid(sM1,'Coord',sM1.codebook,'LineColor', [0.9 0.9 0.9], 'Marker','.', 'MarkerColor' ,[0.9 0.1
0.1]);
drawnow;

%save figure
saveas(h,'Testes\treino2')

%pause % Strike any key to add labels...

% progress
disp('autolabeling');

load usa.txt -ascii
disp('autolabeling 1');

usa_labels=usa(:,num_features+1);
%usa_labels=num2str(usa_labels);
[n1,n2]=size(usa_labels);
disp('autolabeling 2');

label_cell=mat2cell(usa_labels,ones(1,n1),[n2]); % criar lista de labels

sMlixo=som_randinit_roberto( sD, 'msize',[xNeurons yNeurons],'rect','sheet');
disp('autolabeling 3');

```

```

sM1.labels=sMlixo.labels;
clear sMlixo;
clear usa;
clear usa_labels;

disp('autolabeling 4');

sD.labels=label_cell;
sM1=som_autolabel(sM1,sD,'vote');
clear label_cell;
disp('autolabeling 5');

M= labels2mat(sM1.labels);

%Alteracao do colormap
jet1=jet;
jet1(1,:)= [1 1 1];

% Figura com o resultado do autolabel
disp('plot...');
figure(4);
title('Cartograma');
outMat=reshape(M,sM1.topol.msize(1),sM1.topol.msize(2));
outMat=[outMat ; zeros(1,sM1.topol.msize(2))];
outMat=[outMat zeros(sM1.topol.msize(1)+1,1)];
h = pcolor(flipud(outMat));
COLORMAP(jet1);

%save figure
saveas(h,'Testes\carto')

%save labels
save 'testes\labels.txt' M -ascii;

disp('BMUS');

% Figura com o resultado do BMU (nosso)
figure(5);
Bmus=carto_nearestDado (sM1.codebook, sD);
outMat=reshape(Bmus,sM1.topol.msize(1),sM1.topol.msize(2));
outMat=[outMat ; zeros(1,sM1.topol.msize(2))];
outMat=[outMat zeros(sM1.topol.msize(1)+1,1)];
h = pcolor(flipud(outMat));
title('Cartograma com BMUs ao contrario');
COLORMAP(jet1);

%save figure
saveas(h,'Testes\carto1')

%save bmus
save 'testes\BMUs.txt' Bmus -ascii;

%#####
%calculate area for each state
Estados=labels2mat(sD.labels);
EstadoMin=min(Estados);
EstadoMax=max(Estados);
DeltaX=range(sDn.data(:,1))/xNeurons;

```

```

DeltaY=range(sDn.data(:,2))/yNeurons;

EstadosDiferentes=unique(Estados);
area=zeros(length(EstadosDiferentes),3);

i=0;
for estado=EstadoMin:EstadoMax
    i=i+1;
    area(i,1)= estado;
    area(i,2)=length(find(M == estado));
    area(i,3)=area(i,2)*DeltaX*DeltaY;
end;

save 'testes\contagem.txt' area -ASCII;
%#####

%quantization and topographic error
[qe , te] = som_quality(sM1,sD);

%save parameters to ascii
%A = [ xNeurons yNeurons ; niterations_1 niterations_2 ; radius_ini_1 radius_ini_2 ; alpha_ini_1
alpha_ini_2 ; qe te];
%save 'testes\param.txt' A -ASCII;

fid = fopen('testes\parametros.txt','w');
fprintf(fid,'%s \t %3g\n','xNeurons' , xNeurons);
fprintf(fid,'%s \t %3g\n','yNeurons' , yNeurons);
fprintf(fid,'%s \t %5g\n','niterations_1' , niterations_1);
fprintf(fid,'%s \t %5g\n','niterations_2' , niterations_2);
fprintf(fid,'%s \t %5g\n','radius_ini_1' , radius_ini_1);
fprintf(fid,'%s \t %5g\n','radius_ini_2' , radius_ini_2);
fprintf(fid,'%s \t %5g\n','alpha_ini_1' , alpha_ini_1);
fprintf(fid,'%s \t %5g\n','alpha_ini_2' , alpha_ini_2);
fprintf(fid,'%s \t %5.5g\n','qe' , qe);
fprintf(fid,'%s \t %5.5g\n','te' , te);
fclose(fid);

disp('End');
beep;

```

function sMap = som_randinit_geo(D, varargin)

```

%SOM_RANDOMINIT Initialize a Self-Organizing Map with random values.
%
% sMap = som_randinit(D, [[argID,] value, ...])
%
% sMap = som_randinit(D);
% sMap = som_randinit(D,sMap);
% sMap = som_randinit(D,'munits',100,'hexa');
%
% Input and output arguments ([]'s are optional):
% D          The training data.
%            (struct) data struct
%            (matrix) data matrix, size dlen x dim
% [argID, (string) Parameters affecting the map topology are given
% value] (varies) as argument ID - argument value pairs, listed below.
%
% sMap      (struct) map struct
%
% Here are the valid argument IDs and corresponding values. The values
% which are unambiguous (marked with '*') can be given without the

```

```

% preceeding argID.
% 'munits'      (scalar) number of map units
% 'msize'       (vector) map size
% 'lattice'     *(string) map lattice: 'hexa' or 'rect'
% 'shape'       *(string) map shape: 'sheet', 'cyl' or 'toroid'
% 'topol'       *(struct) topology struct
% 'som_topol','sTopol' = 'topol'
% 'map'         *(struct) map struct
% 'som_map','sMap'   = 'map'
%
% For more help, try 'type som_randinit' or check out online documentation.
% See also SOM_MAP_STRUCT, SOM_LINIT, SOM_MAKE.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% DETAILED DESCRIPTION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% som_randinit
%
% PURPOSE
%
% Initializes a SOM with random values.
%
% SYNTAX
%
% sMap = som_randinit(D)
% sMap = som_randinit(D,sMap);
% sMap = som_randinit(D,'munits',100,'hexa');
%
% DESCRIPTION
%
% Initializes a SOM with random values. If necessary, a map struct
% is created first. For each component (xi), the values are uniformly
% distributed in the range of [min(xi) max(xi)].
%
% REQUIRED INPUT ARGUMENTS
%
% D          The training data.
%            (struct) Data struct. If this is given, its '.comp_names' and
%            '.comp_norm' fields are copied to the map struct.
%            (matrix) data matrix, size dlen x dim
%
% OPTIONAL INPUT ARGUMENTS
%
% argID (string) Argument identifier string (see below).
% value (varies) Value for the argument (see below).
%
% The optional arguments can be given as 'argID',value -pairs. If an
% argument is given value multiple times, the last one is used.
%
% Here are the valid argument IDs and corresponding values. The values
% which are unambiguous (marked with '*') can be given without the
% preceeding argID.
% 'dlen'      (scalar) length of the training data
% 'data'      (matrix) the training data
%            *(struct) the training data
% 'munits'    (scalar) number of map units
% 'msize'     (vector) map size
% 'lattice'   *(string) map lattice: 'hexa' or 'rect'
% 'shape'     *(string) map shape: 'sheet', 'cyl' or 'toroid'
% 'topol'     *(struct) topology struct
% 'som_topol','sTopol' = 'topol'
% 'map'       *(struct) map struct

```

```

% 'som_map','sMap'      = 'map'
%
% OUTPUT ARGUMENTS
%
% sMap    (struct) The initialized map struct.
%
% EXAMPLES
%
% sMap = som_randinit(D);
% sMap = som_randinit(D,sMap);
% sMap = som_randinit(D,sTopol);
% sMap = som_randinit(D,'msize',[10 10]);
% sMap = som_randinit(D,'munits',100,'hexa');
%
% SEE ALSO
%
% som_map_struct  Create a map struct.
% som_lininit     Initialize a map using linear initialization algorithm.
% som_make        Initialize and train self-organizing map.

% Copyright (c) 1997-2000 by the SOM toolbox programming team.
% http://www.cis.hut.fi/projects/somtoolbox/

% Version 1.0beta ecco 100997
% Version 2.0beta juuso 101199

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% check arguments

% data
if isstruct(D),
    data_name = D.name;
    comp_names = D.comp_names;
    comp_norm = D.comp_norm;
    D = D.data;
    struct_mode = 1;
else
    data_name = inputname(1);
    struct_mode = 0;
end
[dlen dim] = size(D);

% varargin
sMap = [];
sTopol = som_topol_struct;
sTopol.msize = 0;
munits = NaN;
i=1;
while i<=length(varargin),
    argok = 1;
    if ischar(varargin{i}),
        switch varargin{i},
            case 'munits',    i=i+1; munits = varargin{i}; sTopol.msize = 0;
            case 'msize',    i=i+1; sTopol.msize = varargin{i};
                                munits = prod(sTopol.msize);
            case 'lattice',   i=i+1; sTopol.lattice = varargin{i};
            case 'shape',     i=i+1; sTopol.shape = varargin{i};
            case {'som_topol','sTopol','topol'}, i=i+1; sTopol = varargin{i};
            case {'som_map','sMap','map'}, i=i+1; sMap = varargin{i}; sTopol = sMap.topol;
            case {'hexa','rect'}, sTopol.lattice = varargin{i};
            case {'sheet','cyl','toroid'}, sTopol.shape = varargin{i};
        end
    else
        argok = 0;
    end
    if argok,
        i=i+1;
    end
end

```

```

        otherwise argok=0;
    end
elseif isstruct(varargin{i}) & isfield(varargin{i},'type'),
    switch varargin{i}.type,
        case 'som_topol',
            sTopol = varargin{i};
        case 'som_map',
            sMap = varargin{i};
            sTopol = sMap.topol;
        otherwise argok=0;
    end
else
    argok = 0;
end
if ~argok,
    disp(['(som_topol_struct) Ignoring invalid argument #' num2str(i)]);
end
i = i+1;
end

if ~isempty(sMap),
    [munits dim2] = size(sMap.codebook);
    if dim2 ~= dim, error('Map and data must have the same dimension.'); end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% create map

% map struct
if ~isempty(sMap),
    sMap = som_set(sMap,'topol',sTopol);
else
    if ~prod(sTopol.msize),
        if isnan(munits),
            sTopol = som_topol_struct('data',D,sTopol);
        else
            sTopol = som_topol_struct('data',D,'munits',munits,sTopol);
        end
    end
    sMap = som_map_struct(dim, sTopol);
end

if struct_mode,
    sMap = som_set(sMap,'comp_names',comp_names,'comp_norm',comp_norm);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% initialization

% train struct
sTrain = som_train_struct('algorithm','randinit');
sTrain = som_set(sTrain,'data_name',data_name);

munits = prod(sMap.topol.msize);
%%%
%%% Modified by V.Lobo, 04/04/06 (yy/mm/dd)
%%%
sMap.codebook = zeros([munits dim]); % Initialize all weights to 0
% set interval of GEO component to correct value
ma_x = max(D(:,1)); mi_x = min(D(:,1)); % find max and min geo-coordinates of the data

```

```

ma_y = max(D(:,2)); mi_y = min(D(:,2));
intervalx=(ma_x-mi_x)/(sMap.topol.msize(2)-1); % find interval bewtween coords
x=mi_x:intervalx:ma_x; % distrite x evenly along the axis
intervaly=(ma_y-mi_y)/(sMap.topol.msize(1)-1);
y=ma_y:-intervaly:mi_y;

k=1;
for i=1:sMap.topol.msize(2)
    for j=1:sMap.topol.msize(1)
        sMap.codebook(k,1:2)=[ x(i) y(j) ];
        k=k+1;
    end;
end;
%%

% Atribuir os outros par^ametros do vizinho mais proximo
if dim > 2
    Dist_total = dist( sMap.codebook(:,1:2),D(:,1:2));
    [lixo, bmp ] = min( Dist_total,[], 2 );
    sMap.codebook(:,3:end)=D(bmp,3:end);
end;

% training struct
sTrain = som_set(sTrain,'time',datestr(now,0));
sMap.trainhist = sTrain;

return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

function [sMap, sTrain] = som_seqtrain_roberto(sMap, D, varargin)

```

%SOM_SEQTRAIN Use sequential algorithm to train the Self-Organizing Map.
%
% [sM,sT] = som_seqtrain(sM, D, [[argID,] value, ...])
%
% sM = som_seqtrain(sM,D);
% sM = som_seqtrain(sM,sD,'alpha_type','power','tracking',3);
% [M,sT] = som_seqtrain(M,D,'ep','trainlen',10,'inv','hexa');
%
% Input and output arguments ([]'s are optional):
% sM (struct) map struct, the trained and updated map is returned
% (matrix) codebook matrix of a self-organizing map
% size munits x dim or msize(1) x ... x msize(k) x dim
% The trained map codebook is returned.
% D (struct) training data; data struct
% (matrix) training data, size dlen x dim
% [argID, (string) See below. The values which are unambiguous can
% value] (varies) be given without the preceeding argID.
%
% sT (struct) learning parameters used during the training
%
% Here are the valid argument IDs and corresponding values. The values which
% are unambiguous (marked with **) can be given without the preceeding argID.
% 'mask' (vector) BMU search mask, size dim x 1
% 'msize' (vector) map size
% 'radius' (vector) neighborhood radiuses, length 1, 2 or trainlen
% 'radius_ini' (scalar) initial training radius
% 'radius_fin' (scalar) final training radius
% 'alpha' (vector) learning rates, length trainlen
% 'alpha_ini' (scalar) initial learning rate

```

```

% 'tracking' (scalar) tracking level, 0-3
% 'trainlen' (scalar) training length
% 'trainlen_type' *(string) is the given trainlen 'samples' or 'epochs'
% 'train' *(struct) train struct, parameters for training
% 'sTrain','som_train' = 'train'
% 'alpha_type' *(string) learning rate function, 'inv', 'linear' or 'power'
% 'sample_order'*(string) order of samples: 'random' or 'ordered'
% 'neigh' *(string) neighborhood function, 'gaussian', 'cutgauss',
% 'ep' or 'bubble'
% 'topol' *(struct) topology struct
% 'som_topol','sTopol' = 'topol'
% 'lattice' *(string) map lattice, 'hexa' or 'rect'
% 'shape' *(string) map shape, 'sheet', 'cyl' or 'toroid'
%
% For more help, try 'type som_seqtrain' or check out online documentation.
% See also SOM_MAKE, SOM_BATCHTRAIN, SOM_TRAIN_STRUCT.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% DETAILED DESCRIPTION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% som_seqtrain
%
% PURPOSE
%
% Trains a Self-Organizing Map using the sequential algorithm.
%
% SYNTAX
%
% sM = som_seqtrain(sM,D);
% sM = som_seqtrain(sM,sD);
% sM = som_seqtrain(...,'argID',value,...);
% sM = som_seqtrain(...,value,...);
% [sM,sT] = som_seqtrain(M,D,...);
%
% DESCRIPTION
%
% Trains the given SOM (sM or M above) with the given training data
% (sD or D) using sequential SOM training algorithm. If no optional
% arguments (argID, value) are given, a default training is done, the
% parameters are obtained from SOM_TRAIN_STRUCT function. Using
% optional arguments the training parameters can be specified. Returns
% the trained and updated SOM and a train struct which contains
% information on the training.
%
% REFERENCES
%
% Kohonen, T., "Self-Organizing Map", 2nd ed., Springer-Verlag,
% Berlin, 1995, pp. 78-82.
% Kohonen, T., "Clustering, Taxonomy, and Topological Maps of
% Patterns", International Conference on Pattern Recognition
% (ICPR), 1982, pp. 114-128.
% Kohonen, T., "Self-Organized formation of topologically correct
% feature maps", Biological Cybernetics 43, 1982, pp. 59-69.
%
% REQUIRED INPUT ARGUMENTS
%
% sM The map to be trained.
% (struct) map struct
% (matrix) codebook matrix (field .data of map struct)
% Size is either [munits dim], in which case the map grid
% dimensions (msize) should be specified with optional arguments,

```



```

%           or [msize(1) ... msize(k) dim] in which case the map
%           grid dimensions are taken from the size of the matrix.
%           Lattice, by default, is 'rect' and shape 'sheet'.
% D           Training data.
%           (struct) data struct
%           (matrix) data matrix, size [dlen dim]
%
% OPTIONAL INPUT ARGUMENTS
%
% argID (string) Argument identifier string (see below).
% value (varies) Value for the argument (see below).
%
% The optional arguments can be given as 'argID',value -pairs. If an
% argument is given value multiple times, the last one is
% used. The valid IDs and corresponding values are listed below. The values
% which are unambiguous (marked with '*') can be given without the
% preceeding argID.
%
% 'mask'      (vector) BMU search mask, size dim x 1. Default is
%              the one in sM (field '.mask') or a vector of
%              ones if only a codebook matrix was given.
% 'msize'     (vector) map grid dimensions. Default is the one
%              in sM (field sM.topol.msize) or
%              'si = size(sM); msize = si(1:end-1);'
%              if only a codebook matrix was given.
% 'radius'    (vector) neighborhood radius
%              length = 1: radius_ini = radius
%              length = 2: [radius_ini radius_fin] = radius
%              length > 2: the vector given neighborhood
%              radius for each step separately
%              trainlen = length(radius)
% 'radius_ini' (scalar) initial training radius
% 'radius_fin' (scalar) final training radius
% 'alpha'     (vector) learning rate
%              length = 1: alpha_ini = alpha
%              length > 1: the vector gives learning rate
%              for each step separately
%              trainlen is set to length(alpha)
%              alpha_type is set to 'user defined'
% 'alpha_ini' (scalar) initial learning rate
% 'tracking'   (scalar) tracking level: 0, 1 (default), 2 or 3
%              0 - estimate time
%              1 - track time and quantization error
%              2 - plot quantization error
%              3 - plot quantization error and two first
%              components
% 'trainlen'   (scalar) training length (see also 'tlen_type')
% 'trainlen_type' *(string) is the trainlen argument given in 'epochs'
%              or in 'samples'. Default is 'epochs'.
% 'sample_order'*(string) is the sample order 'random' (which is the
%              the default) or 'ordered' in which case
%              samples are taken in the order in which they
%              appear in the data set
% 'train'      *(struct) train struct, parameters for training.
%              Default parameters, unless specified,
%              are acquired using SOM_TRAIN_STRUCT (this
%              also applies for 'trainlen', 'alpha_type',
%              'alpha_ini', 'radius_ini' and 'radius_fin').
% 'sTrain', 'som_train' (struct) = 'train'
% 'neigh'      *(string) The used neighborhood function. Default is
%              the one in sM (field '.neigh') or 'gaussian'
%              if only a codebook matrix was given. Other

```

```

%           possible values is 'cutgauss', 'ep' and 'bubble'.
% 'topol'   *(struct) topology of the map. Default is the one
%           in sM (field 'topol').
% 'sTopol', 'som_topol' (struct) = 'topol'
% 'alpha_type'*(string) learning rate function, 'inv', 'linear' or 'power'
% 'lattice' *(string) map lattice. Default is the one in sM
%           (field sM.topol.lattice) or 'rect'
%           if only a codebook matrix was given.
% 'shape'   *(string) map shape. Default is the one in sM
%           (field sM.topol.shape) or 'sheet'
%           if only a codebook matrix was given.
%
% OUTPUT ARGUMENTS
%
% sM        the trained map
% (struct) if a map struct was given as input argument, a
% map struct is also returned. The current training
% is added to the training history (sM.trainhist).
% The 'neigh' and 'mask' fields of the map struct
% are updated to match those of the training.
% (matrix) if a matrix was given as input argument, a matrix
% is also returned with the same size as the input
% argument.
% sT (struct) train struct; information of the accomplished training
%
% EXAMPLES
%
% Simplest case:
% sM = som_seqtrain(sM,D);
% sM = som_seqtrain(sM,sD);
%
% To change the tracking level, 'tracking' argument is specified:
% sM = som_seqtrain(sM,D,'tracking',3);
%
% The change training parameters, the optional arguments 'train',
% 'neigh','mask','trainlen','radius','radius_ini', 'radius_fin',
% 'alpha', 'alpha_type' and 'alpha_ini' are used.
% sM = som_seqtrain(sM,D,'neigh','cutgauss','trainlen',10,'radius_fin',0);
%
% Another way to specify training parameters is to create a train struct:
% sTrain = som_train_struct(sM,'dlen',size(D,1),'algorithm','seq');
% sTrain = som_set(sTrain,'neigh','cutgauss');
% sM = som_seqtrain(sM,D,sTrain);
%
% By default the neighborhood radius goes linearly from radius_ini to
% radius_fin. If you want to change this, you can use the 'radius' argument
% to specify the neighborhood radius for each step separately:
% sM = som_seqtrain(sM,D,'radius',[5 3 1 1 1 1 0.5 0.5 0.5]);
%
% By default the learning rate (alpha) goes from the alpha_ini to 0
% along the function defined by alpha_type. If you want to change this,
% you can use the 'alpha' argument to specify the learning rate
% for each step separately:
% alpha = 0.2*(1 - log([1:100]));
% sM = som_seqtrain(sM,D,'alpha',alpha);
%
% You don't necessarily have to use the map struct, but you can operate
% directly with codebook matrices. However, in this case you have to
% specify the topology of the map in the optional arguments. The
% following commads are identical (M is originally a 200 x dim sized matrix):
% M = som_seqtrain(M,D,'msize',[20 10],'lattice','hexa','shape','cyl');
%

```

```

% M = som_seqtrain(M,D,'msize',[20 10],'hexa','cyl');
%
% sT= som_set('som_topol','msize',[20 10],'lattice','hexa','shape','cyl');
% M = som_seqtrain(M,D,sT);
%
% M = reshape(M,[20 10 dim]);
% M = som_seqtrain(M,D,'hexa','cyl');
%
% The som_seqtrain also returns a train struct with information on the
% accomplished training. This is the same one as is added to the end of the
% trainhist field of map struct, in case a map struct is given.
% [M,sTrain] = som_seqtrain(M,D,'msize',[20 10]);
%
% [sM,sTrain] = som_seqtrain(sM,D); % sM.trainhist{end}==sTrain
%
% SEE ALSO
%
% som_make      Initialize and train a SOM using default parameters.
% som_batchtrain Train SOM with batch algorithm.
% som_train_struct Determine default training parameters.

% Copyright (c) 1997-2000 by the SOM toolbox programming team.
% http://www.cis.hut.fi/projects/somtoolbox/

% Version 1.0beta juuso 220997
% Version 2.0beta juuso 101199

%%%%%%%%%%%%%%
%% Check arguments

error(nargchk(2, Inf, nargin)); % check the number of input arguments

% map
struct_mode = isstruct(sMap);
if struct_mode,
    sTopol = sMap.topol;
else
    orig_size = size(sMap);
    if ndims(sMap) > 2,
        si = size(sMap); dim = si(end); msize = si(1:end-1);
        M = reshape(sMap,[prod(msize) dim]);
    else
        msize = [orig_size(1) 1];
        dim = orig_size(2);
    end
    sMap = som_map_struct(dim,'msize',msize);
    sTopol = sMap.topol;
end
[munits dim] = size(sMap.codebook);

% data
if isstruct(D),
    data_name = D.name;
    D = D.data;
else
    data_name = inputname(2);
end
D = D(find(sum(isnan(D),2) < dim),:); % remove empty vectors from the data
[dlen ddim] = size(D); % check input dimension
if dim ~= ddim, error('Map and data input space dimensions disagree.');
```

```

sTrain = som_set('som_train','algorithm','seq','neigh', ...
                sMap.neigh,'mask',sMap.mask,'data_name',data_name);

radius    = [];
alpha     = [];
tracking  = 1;
sample_order_type = 'random';
tlen_type = 'epochs';

i=1;
while i<=length(varargin),
    argok = 1;
    if ischar(varargin{i}),
        switch varargin{i},
            % argument IDs
            case 'msize', i=i+1; sTopol.msize = varargin{i};
            case 'lattice', i=i+1; sTopol.lattice = varargin{i};
            case 'shape', i=i+1; sTopol.shape = varargin{i};
            case 'mask', i=i+1; sTrain.mask = varargin{i};
            case 'neigh', i=i+1; sTrain.neigh = varargin{i};
            case 'trainlen', i=i+1; sTrain.trainlen = varargin{i};
            case 'trainlen_type', i=i+1; tlen_type = varargin{i};
            case 'tracking', i=i+1; tracking = varargin{i};
            case 'sample_order', i=i+1; sample_order_type = varargin{i};
            case 'radius_ini', i=i+1; sTrain.radius_ini = varargin{i};
            case 'radius_fin', i=i+1; sTrain.radius_fin = varargin{i};
            case 'radius',
                i=i+1;
                l = length(varargin{i});
                if l==1,
                    sTrain.radius_ini = varargin{i};
                else
                    sTrain.radius_ini = varargin{i}(1);
                    sTrain.radius_fin = varargin{i}(end);
                    if l>2, radius = varargin{i}; tlen_type = 'samples'; end
                end
            case 'alpha_type', i=i+1; sTrain.alpha_type = varargin{i};
            case 'alpha_ini', i=i+1; sTrain.alpha_ini = varargin{i};
            case 'alpha',
                i=i+1;
                sTrain.alpha_ini = varargin{i}(1);
                if length(varargin{i})>1,
                    alpha = varargin{i}; tlen_type = 'samples';
                    sTrain.alpha_type = 'user defined';
                end
            case {'sTrain','train','som_train'}, i=i+1; sTrain = varargin{i};
            case {'topol','sTopol','som_topol'},
                i=i+1;
                sTopol = varargin{i};
                if prod(sTopol.msize) ~= munits,
                    error('Given map grid size does not match the codebook size.');
```

```

    case 'som_topol',
        sTopol = varargin{i};
        if prod(sTopol.msize) ~= munits,
            error('Given map grid size does not match the codebook size.');
```

end

```

        case 'som_train', sTrain = varargin{i};
        otherwise argok=0;
    end
else
    argok = 0;
end
if ~argok,
    disp(['(som_seqtrain) Ignoring invalid argument #' num2str(i+2)]);
end
i = i+1;
end

% training length
if ~isempty(radius) | ~isempty(alpha),
    lr = length(radius);
    la = length(alpha);
    if lr>2 | la>1,
        tlen_type = 'samples';
        if lr> 2 & la<=1, sTrain.trainlen = lr;
        elseif lr<=2 & la> 1, sTrain.trainlen = la;
        elseif lr==la, sTrain.trainlen = la;
        else
            error('Mismatch between radius and learning rate vector lengths.')
```

end

end

```

end
if strcmp(tlen_type,'samples'), sTrain.trainlen = sTrain.trainlen/dlen; end

% check topology
if struct_mode,
    if ~strcmp(sTopol.lattice,sMap.topol.lattice) | ...
        ~strcmp(sTopol.shape,sMap.topol.shape) | ...
        any(sTopol.msize ~= sMap.topol.msize),
        warning('Changing the original map topology.');
```

end

end

```

sMap.topol = sTopol;
% complement the training struct
sTrain = som_train_struct(sTrain,sMap,'dlen',dlen);
if isempty(sTrain.mask), sTrain.mask = ones(dim,1); end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% initialize
%M      = sMap.codebook;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Alterado por Roberto Henriques
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%removo os neuronios do mar ficando apenas
%os neuronios do continente

l_c = cellfun('isempty',sMap.labels);
%indices dos neuronios de terra
iTerra=find(l_c == 0 );
```

```

%indices dos neuronios de mar
iMar=find(l_c == 1 );

%neuronios em Terra
M=sMap.codebook(iTerra,:);
[munits dim] = size(M);

%sMap.topol = sTopol;
%M = zeros(size(sMap.codebook));
% for k = 1:length(sMap.labels)
%     if sum(sMap.labels{k,1}) > 0
%         M(k)=sMap.codebook(k , :)
%     end
% end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

mask    = sTrain.mask;
trainlen = sTrain.trainlen*dlen;

% neighborhood radius
if length(radius)>2,
    radius_type = 'user defined';
else
    radius = [sTrain.radius_ini sTrain.radius_fin];
    rini = radius(1);
    rstep = (radius(end)-radius(1))/(trainlen-1);
    radius_type = 'linear';
end

% learning rate
if length(alpha)>1,
    sTrain.alpha_type = 'user defined';
    if length(alpha) ~= trainlen,
        error('Trainlen and length of neighborhood radius vector do not match.')
    end
    if any(isnan(alpha)),
        error('NaN is an illegal learning rate.')
    end
else
    if isempty(alpha), alpha = sTrain.alpha_ini; end
    if strcmp(sTrain.alpha_type,'inv'),
        % alpha(t) = a / (t+b), where a and b are chosen suitably
        % below, they are chosen so that alpha_fin = alpha_ini/100
        b = (trainlen - 1) / (100 - 1);
        a = b * alpha;
    end
end

% initialize random number generator
rand('state',sum(100*clock));

% distance between map units in the output space
% Since in the case of gaussian and ep neighborhood functions, the
% equations utilize squares of the unit distances and in bubble case
% it doesn't matter which is used, the unitdistances and neighborhood
% radiuses are squared.
Ud = som_unit_dists(sTopol).^2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Action

```

```

update_step = 100;
mu_x_1 = ones(munits,1);
samples = ones(update_step,1);
r = samples;
alfa = samples;

qe = 0;
start = clock;
if tracking > 0, % initialize tracking
    track_table = zeros(update_step,1);
    qe = zeros(floor(trainlen/update_step),1);
end

for t = 1:trainlen,

    % Every update_step, new values for sample indeces, neighborhood
    % radius and learning rate are calculated. This could be done
    % every step, but this way it is more efficient. Or this could
    % be done all at once outside the loop, but it would require much
    % more memory.
    ind = rem(t,update_step); if ind==0, ind = update_step; end
    if ind==1,
        steps = [t:min(trainlen,t+update_step-1)];
        % sample order
        switch sample_order_type,
            case 'ordered', samples = rem(steps,dlen)+1;
            case 'random', samples = ceil(dlen*rand(update_step,1)+eps);
        end

        % neighborhood radius
        switch radius_type,
            case 'linear', r = rini+(steps-1)*rstep;
            case 'user defined', r = radius(steps);
        end
        r=r.^2; % squared radius (see notes about Ud above)
        r(r==0) = eps; % zero radius might cause div-by-zero error

        % learning rate
        switch sTrain.alpha_type,
            case 'linear', alfa = (1-steps/trainlen)*alpha;
            case 'inv', alfa = a ./ (b + steps-1);
            case 'power', alfa = alpha * (0.005/alpha).^((steps-1)/trainlen);
            case 'user defined', alfa = alpha(steps);
        end
    end

    %#####

    % find BMU
    x = D(samples(ind),:); % pick one sample vector
    known = ~isnan(x); % its known components
    Dx = M(:,known) - x(mu_x_1,known); % each map unit minus the vector
    [qerr bmu] = min((Dx.^2)*mask(known)); % minimum distance(^2) and the BMU

    %este bmu refere-se ao indice do neuronio mais proximo na Matriz M ; De
    %notar que neste caso M e uma submatriz do M inicial (M possui agora so
    %os neuronios da terra) Agora tenho que apanhar o BMU referente ao M
    %inicial

    bmu=iTerra(bmu);

    %#####

```

```

% tracking
if tracking>0,
    track_table(ind) = sqrt(qerr);
    if ind==update_step,
        n = ceil(t/update_step);
        qe(n) = mean(track_table);
        trackplot(M,D,tracking,start,n,qe);
    end
end

% neighborhood & learning rate
% notice that the elements Ud and radius have been squared!
% (see notes about Ud above)
switch sTrain.neigh,
case 'bubble', h = (Ud(:,bmu)<=r(ind));
case 'gaussian', h = exp(-Ud(:,bmu)/(2*r(ind)));
case 'cutgauss', h = exp(-Ud(:,bmu)/(2*r(ind))) .* (Ud(:,bmu)<=r(ind));
case 'ep', h = (1-Ud(:,bmu)/r(ind)) .* (Ud(:,bmu)<=r(ind));
end
h = h*alfa(ind);

%alteracao Roberto Henriques
% so atualizo os neuronios da terra
h = h(iTerra,:);
% update M
M(:,known) = M(:,known) - h(:,ones(sum(known),1)).*Dx;

%mTerra=[M iTerra];
%mMar=[sMap.codebook(iMar,:) iMar];
%mTotal = [mTerra;mMar];
%Lixo = sortrows(mTotal,3);
%Lixo = Lixo(:,1:2);

%mostrar figura
%clf;
%plot (D(:,1),D(:,2),'r');
%hold on;
%som_grid(sMap,'Coord',Lixo,'LineWidth',2,'LineColor',[0.5 0.8 0.5]);
%drawnow;

end; % for t = 1:trainlen

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Build / clean up the return arguments

if tracking, fprintf(1,'\n'); end

% update structures
sTrain = som_set(sTrain,'time',datestr(now,0));
if struct_mode,
    %sMap = som_set(sMap,'codebook',M,'mask',sTrain.mask,'neigh',sTrain.neigh);

%alteracao Roberto Henriques
% A matriz M deve ser do mesmo tamanho que o sMap original ou seja tenho
% que lhe adicionar os neuronios do mar nesta fase para que o sMap fique
% com o tamanho original

mTerra=[M iTerra];
mMar=[sMap.codebook(iMar,:) iMar];
mTotal = [mTerra;mMar];
M = sortrows(mTotal,3);

```



```

M = M(:,1:2);
sMap = som_set(sMap,'codebook',M,'mask',sTrain.mask,'neigh',sTrain.neigh);
tl = length(sMap.trainhist);
sMap.trainhist(tl+1) = sTrain;
else
    sMap = reshape(M,orig_size);
end

return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% subfunctions

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [] = trackplot(M,D,tracking,start,n,qe)

l = length(qe);
elap_t = etime(clock,start);
tot_t = elap_t*l/n;
fprintf(1,'\rTraining: %3.0f/ %3.0f s',elap_t,tot_t)
switch tracking
case 1,
case 2,
    plot(1:n,qe(1:n),(n+1):l,qe((n+1):l))
    title('Quantization errors for latest samples')
    drawnow
otherwise,
    %subplot(2,1,1), plot(1:n,qe(1:n),(n+1):l,qe((n+1):l))
    %title('Quantization error for latest samples');
    %subplot(2,1,2), plot(M(:,1),M(:,2),'r*',D(:,1),D(:,2),'b. ');
    plot(M(:,1),M(:,2),'r*',D(:,1),D(:,2),'b. ');
    title('Neuronios da Terra (*) e dados (.)');
    drawnow
end
% end of trackplot

```

Appendix 3 – SOM training tests

Portugal Datasets

SOM parameters used in the several tests: x and y are the number of units in x and y used in the SOM; $iter1$ and $iter2$ are the number of epochs for the first and second training phase (the unfolding and fine tune phases); $n1$ and $n2$ are the neighbourhood radius used in the first and second training phase and $L1$ and $L2$ are the first and second training learning rate; qe is quantization error and te is the topographic error; *number of training patterns* is the number of patterns used in each dataset

Variants	Test	Parameters								Error		Nº Training patterns
		x	y	$iter1$	$iter2$	$n1$	$n2$	$L1$	$L2$	qe	te	
1	1	50	25	50	75	20	8	0,5	0,1	3243,4	0,0954	6687
	2	75	50	50	75	20	8	0,5	0,1	1920,7	0,135	
	3	50	25	50	75	25	10	0,5	0,1	3356,1	0,1045	
	4	75	50	50	75	25	10	0,5	0,1	2051,8	0,1168	
	5	50	25	50	75	25	10	0,7	0,2	3062,7	0,0963	
	6	75	50	50	75	25	10	0,5	0,1	1818,7	0,1285	
2	1	50	25	50	75	20	8	0,5	0,1	5704,4	0,0419	6877
	2	75	50	50	75	20	8	0,5	0,1	3389,5	0,0519	
	3	50	25	50	75	25	10	0,5	0,1	5923,1	0,0384	
	4	75	50	50	75	25	10	0,5	0,1	3569,7	0,054	
	5	50	25	50	75	25	10	0,7	0,2	5436,4	0,0445	
	6	75	50	50	75	25	10	0,5	0,1	3235,8	0,0545	
3	1	50	25	50	75	20	8	0,5	0,1	4673,8	0,1285	19606
	2	75	50	50	75	20	8	0,5	0,1	2790,1	0,0807	
	3	50	25	50	75	25	10	0,5	0,1	4786,5	0,103	
	4	75	50	50	75	25	10	0,5	0,1	2902,8	0,0816	
	5	50	25	50	75	25	10	0,7	0,2	4544,5	0,1067	
	6	75	50	50	75	25	10	0,5	0,1	---	---	
4	1	50	25	50	75	8	5	0,5	0,1	3774,9	0,2016	6687
	2	75	50	50	75	8	5	0,5	0,1	2294,1	0,1757	
	3	50	25	50	75	10	8	0,5	0,1	4043,8	0,2145	
	4	75	50	50	75	10	8	0,5	0,1	2408,7	0,2004	
	5	50	25	50	75	8	5	0,7	0,2	3520,7	0,1718	
	6	75	50	50	75	8	5	0,5	0,1	2062,6	0,2324	

USA Datasets

SOM parameters used in the several tests: x and y are the number of units in x and y used in the SOM; $iter1$ and $iter2$ are the number of epochs for the first and second training phase (the unfolding and fine tune phases); $n1$ and $n2$ are the neighbourhood rate used in the first and second training phase and $L1$ and $L2$ are the first and second training learning rate; qe is quantization error and te is the topographic error; *number of training patterns* is the number of patterns used in each dataset .

		Parameters								Error		N° Training patterns
Variants	Test	x	y	$iter1$	$iter2$	$n1$	$n2$	$L1$	$L2$	qe	te	
1	1	50	25	50	75	20	8	0,5	0,1	27617	0,16184	25396
	2	75	50	50	75	20	8	0,5	0,1	---	---	
	3	50	25	50	75	25	10	0,5	0,1	28069	0,16042	
	4	75	50	50	75	25	10	0,5	0,1	---	---	
	5	50	25	50	75	25	10	0,7	0,2	26975	0,16459	
	6	75	50	50	75	25	10	0,5	0,1	---	---	
2	1	50	25	50	75	20	8	0,5	0,1	57542	0,06696	20828
	2	75	50	50	75	20	8	0,5	0,1	57542	0,06633	
	3	50	25	50	75	25	10	0,5	0,1	61029	0,05740	
	4	75	50	50	75	25	10	0,5	0,1	36350	0,06633	
	5	50	25	50	75	25	10	0,7	0,2	54266	0,07419	
	6	75	50	50	75	25	10	0,5	0,1	31871	0,07802	
3	1	50	25	50	75	20	8	0,5	0,1	46079	0,08854	20828
	2	75	50	50	75	20	8	0,5	0,1	26934	0,07423	
	3	50	25	50	75	25	10	0,5	0,1	47102	0,08412	
	4	75	50	50	75	25	10	0,5	0,1	27701	0,07510	
	5	50	25	50	75	25	10	0,7	0,2	44674	0,09425	
	6	75	50	50	75	25	10	0,5	0,1	26031	0,07207	
4	1	50	25	50	75	8	5	0,5	0,1	33267	0,16227	25396
	2	75	50	50	75	8	5	0,5	0,1	19747	0,17125	
	3	50	25	50	75	10	8	0,5	0,1	34045	0,16463	
	4	75	50	50	75	10	8	0,5	0,1	20420	0,16802	
	5	50	25	50	75	8	5	0,7	0,2	32746	0,18475	
	6	75	50	50	75	8	5	0,5	0,1	19237	0,16558	

Appendix 4 – *soMGis* application

soMGis is a stand-alone application developed in Visual Basic using the MapObjects component developed by ESRI.

The main goal of this tool is to use the self-organizing map in a geographic environment. This is mainly a demo tool to understand better the SOM algorithm.

Interfaces

Figure A4. 1- *soMGis* main interface

- 1 – Menu area
- 2 – Button Toolbar
- 3 – Legend area
- 4 – Map area

File Menu

Figure A4. 2 – File menu

This menu has only one sub-menu:

- Close – allows the user to end the application

Layers Menu

Figure A4. 3 – Layers menu

This menu has three available functions:

- Add... – allows the user to add a new vector layer from different sources (.shp, .dgn, .dxf, etc.)
- Add image... – allows the user to add a new raster layer
- Remove... – allows the user to remove the current layers

SOM Menu

Figure A4. 4 – SOM menu

This menu allows the user to create a patterns dataset, initialize a SOM and train it.

Help Menu

Figure A4. 5 – Help menu

This menu presents the user the about dialog box.

Buttons

The following figure shows each available button and its function.

Button	Function
Zoom in function
Zoom out function
Pan function
Distance and area calculation function
Full extent function
Identify function

Figure A4. 6 – Buttons

The SOM function

To use this function the user has to load some geographical information to the map. By default the soMGis installation package adds some information about Alentejo region and its population in the installation path.

The SOM function, at this phase, works only with polygon data. The first step is to click on the Kohonen SOM sub-menu located on the SOM menu. At this phase a new form is presented (Figure A4. 7). This form will allow the user to create a point dataset based on a selected attribute from the geographic information. This tool will create random points inside each polygon, and the number of points will be proportional to the value on the selected variable.

Figure A4. 7 – Populate form

Using the Alentejo dataset, one can choose the population variable to create the pattern dataset. The maximum number of points for each region can also be input, by default the value is 100. To create the point dataset one should click on the Populate button. In this form it is also possible to create this dataset and export it to the SOM-PAK format or to an ASCII file.

After the pattern dataset creation (Figure A4. 8) the user should click on the “Run SOM” button.

Figure A4. 8 – Random points proportional to the population value

A new form is displayed on the screen. This is the SOM parameters definition form. On this form we can change the following parameters:

<i>Learning rate</i>	Values between 0 and 1
	<i>Neighbourhood rate</i>	The neighbours updated
	<i>Momentum</i>	The leaning rate and Neighbourhood rate decrease on each iteration
	<i>Grid Rows</i>	The SOM number of rows
	<i>Grid Columns</i>	The SOM number of columns
	<i>See Network</i>	Allows the user to see the network on the map
	<i>Start</i>	Starts the training phase
	<i>Initialize</i>	Initialize the network with the defined parameters
	<i>Speed</i>	Speed up or slows the training
	<i>See selected pattern and BMU</i>	Allows the user to see in each iteration the selected pattern and the BMU
	<i>Classify</i>	Allows the clustering of the initial geographic information based on the units position
	<i>Exit</i>	Exits the process

In the following figure the SOM is initialized using some training patterns created based on the Alentejo regions' density.

Figure A4. 9 – SOM initialization

During the training phase (Figure A4. 10) the SOM will adapt to the training patterns in a way that the distances between the units and the training patterns is minimized.

The last function available in soMGis is the classification or clustering process. In this phase, based on the units' final position, Alentejo regions are grouped to form clusters.

Appendix 5 – Cartogram error

Cartogram error (ce) for the Portuguese dataset. Global cartogram evaluation uses the weighed error (*we*), single average error (*se*), and mean quadratic error (*mqe*)

we	8,76%	1,70%	1,99%	2,57%	2,48%	2,49%
se	5,20%	1,04%	1,16%	1,24%	1,22%	1,33%
mqe	1,68%	0,33%	0,36%	0,46%	0,47%	0,51%
Distrito	Original ce	ce Dougenik	ce Carto-SOM 1	ce Carto-SOM 2	ce Carto-SOM 3	ce Carto-SOM 4
Aveiro	-0,040952577	-0,00517817	0,016730079	0,003772794	-7,21645E-05	0,002131478
Beja	0,099278664	0,021729332	0,005278381	0,01233141	0,019429854	0,010017767
Braga	-0,053934612	-0,008566563	-0,007785605	-0,003752102	0,014640762	-0,004255711
Bragança	0,059782114	0,01083325	0,001688792	0,006965315	0,008056254	-0,003065059
Castelo Branco	0,053449634	0,010783763	0,008918252	0,013096863	0,016085153	0,010818391
Coimbra	-0,000236179	-0,000637502	0,014327763	0,00931976	0,004383864	0,012160969
Évora	0,065814607	0,009261422	0,007888576	0,006660493	0,013961507	0,019852694
Faro	0,016339285	0,004392089	-0,016174048	0,005158953	0,000628196	-0,015542011
Guarda	0,044481852	0,009816445	0,006926946	0,003816372	0,003504786	0,006730981
Leiria	-0,007292272	0,000116971	0,010868536	0,006370901	0,010251425	0,010314646
Lisboa	-0,185915643	-0,034699753	-0,040622646	-0,057663938	-0,057242562	-0,078657484
Portalegre	0,055928536	0,008042273	0,003904239	0,002565547	0,008869175	0,008396804
Porto	-0,154481628	-0,027803756	-0,029897353	-0,050443286	-0,048705207	-0,015493971
Santarém	0,02986737	0,008687893	0,014268147	0,013482502	0,002332663	0,024218336
Setúbal	-0,024230659	-0,016294876	-0,000212296	0,003903008	-0,001348341	0,003790359
Viana do Castelo	-0,000548715	0,002372581	-0,00987496	0,002204565	0,003392922	-0,002242788
Vila Real	0,026260305	0,004253141	0,001847042	0,005996844	0,004680141	0,000446955
Viseu	0,016389917	0,00289146	0,011920156	0,016213999	-0,002848428	0,010377644

Keim area error (*km*) for the Portuguese dataset. Global cartogram evaluation uses the weighed error (*we*), single average error (*se*), and mean quadratic error (*mqe*)

we	48,24%	8,65%	10,45%	11,22%	10,69%	11,87%
se	41,43%	12,29%	12,38%	11,69%	12,54%	13,80%
mqe	11,53%	3,80%	3,29%	3,23%	3,91%	4,00%
Distrito	Original km	km Dougenik	km Carto-SOM 1	km Carto-SOM 2	km Carto-SOM 3	km Carto-SOM 4
Aveiro	0,395097676	0,037139146	0,103698131	0,025427054	0,000499297	0,014525957
Beja	0,752409159	0,399448171	0,139097218	0,274028564	0,372941746	0,234680606
Braga	0,470880869	0,053571865	0,04845144	0,022778347	0,079953834	0,025914895
Bragança	0,664593393	0,264199492	0,053007335	0,187561751	0,210747424	0,113077851
Castelo Branco	0,559019816	0,203669906	0,174587844	0,23700286	0,276146368	0,204190368
Coimbra	0,002648549	0,007181387	0,13811655	0,094397631	0,046739845	0,119729978
Évora	0,651596247	0,208346516	0,183118126	0,159147457	0,284047034	0,360674231
Faro	0,169443126	0,051988389	0,253051701	0,060516298	0,007782574	0,240782146
Guarda	0,54949419	0,212086347	0,159623102	0,094734179	0,087677791	0,155810945
Leiria	0,08498224	0,001254807	0,104535132	0,064046831	0,0991884	0,099739086
Lisboa	0,752870017	0,087150616	0,103566967	0,153690886	0,152396644	0,222070373
Portalegre	0,684824298	0,238062562	0,131703516	0,090637884	0,256267259	0,245975759
Porto	0,747720642	0,083424298	0,090273145	0,16238412	0,155916624	0,044833251
Santarém	0,244862199	0,086192184	0,134128104	0,127685641	0,024699551	0,208191509
Setúbal	0,178759706	0,113565366	0,001330448	0,023845003	0,008510581	0,023172732
Viana do Castelo	0,01093734	0,044689582	0,241780281	0,041656693	0,062703487	0,046267028
Vila Real	0,366771823	0,085763833	0,039144496	0,116817803	0,093568546	0,009762013
Viseu	0,169983606	0,034869588	0,129636219	0,168466488	0,036905229	0,114786412

Kocmoud area error (*kd*) for the Portuguese dataset. Global cartogram evaluation uses the weighed error (*we*), single average error (*se*), and mean quadratic error (*mqe*)

we	93,52%	18,64%	20,91%	22,37%	22,04%	23,85%
se	161,86%	31,16%	26,14%	26,89%	31,57%	32,40%
mqe	56,19%	10,70%	6,85%	7,77%	10,72%	10,07%
Distrito	Original <i>kd</i>	<i>kd</i> Dougenik	<i>kd</i> Carto-SOM 1	<i>kd</i> Carto-SOM 2	<i>kd</i> Carto-SOM 3	<i>kd</i> Carto-SOM 4
Aveiro	-0,566408622	-0,071618444	0,231391084	0,052180915	-0,000998096	0,029480141
Beja	6,077843242	1,330270431	0,323142684	0,754929327	1,189496329	0,613288013
Braga	-0,640270573	-0,101695701	-0,092424768	-0,044542098	0,17380396	-0,050520555
Bragança	3,962911761	0,71812805	0,111948776	0,461725556	0,534043044	-0,203180488
Castelo Branco	2,535351185	0,511521309	0,423031918	0,621241803	0,76298952	0,513163852
Coimbra	-0,005283106	-0,014260364	0,320499367	0,208474789	0,098063147	0,272030116
Évora	3,740466295	0,526357858	0,44833441	0,378538327	0,793479593	1,128295617
Faro	0,408022934	0,109678803	-0,403896664	0,12882884	0,015687236	-0,388113493
Guarda	2,439454404	0,538349212	0,379884556	0,209295825	0,192207949	0,369137562
Leiria	-0,156651854	0,002512767	0,23347679	0,136859051	0,220220078	0,221578176
Lisboa	-0,859014085	-0,160328503	-0,187694938	-0,266433388	-0,264486442	-0,363433037
Portalegre	4,345666827	0,624887426	0,303360702	0,199343874	0,689138034	0,652434618
Porto	-0,855652356	-0,154001157	-0,165597301	-0,27939838	-0,269771402	-0,085818959
Santarém	0,648523221	0,188644007	0,309810504	0,292751437	0,050650137	0,52586329
Setúbal	-0,303301351	-0,20396713	-0,00265736	0,048854952	-0,016877524	0,047444892
Viana do Castelo	-0,021638018	0,093560336	-0,389409116	0,086934802	0,133796479	-0,088442103
Vila Real	1,15841915	0,187618551	0,081478425	0,2645384	0,206454764	0,019716499
Viseu	0,40959096	0,072258812	0,297889737	0,405194706	-0,071183418	0,25934173

Cartogram error (ce) for the USA dataset. Global cartogram evaluation uses the weighed error (*we*), single average error (*se*), and mean quadratic error (*mqe*)

we	2,48%	0,83%	1,06%	0,54%	0,85%	0,98%
se	1,69%	0,48%	0,69%	0,33%	0,55%	0,67%
mqe	0,32%	0,10%	0,13%	0,07%	0,11%	0,13%
Distrito	Original ce	ce Dougenik	ce Carto-SOM 1	ce Carto-SOM 2	ce Carto-SOM 3	ce Carto-SOM 4
Florida	-0,037422934	-0,006823519	-0,019575295	-0,010278217	0,014539824	-0,031862522
Louisiana	-0,000122293	0,001837082	0,005669616	-0,000771718	0,005799059	0,009180151
Georgia	-0,009275046	0,001548392	0,008358392	0,000326169	0,00138313	0,00183306
Mississippi	0,006066052	0,002827837	0,007957439	-0,000109351	0,002257068	0,005503181
Alabama	0,001933619	0,00289683	0,001649191	0,003137164	0,001792474	0,003507502
South Carolina	-0,003666766	0,003721476	0,001404521	0,001593856	0,00115052	-0,000814616
Arkansas	0,008397994	0,002594131	0,00852242	0,001427446	0,005843196	0,005192121
Texas	0,01505795	0,009914518	0,007753096	0,017291361	0,004455152	0,011450241
North Carolina	-0,011836274	-0,001715792	0,004184451	0,002946307	-0,000961515	-0,005374799
Tennessee	-0,005712006	-0,000126029	-0,001167345	0,0035629	0,001387321	0,002140292
New Mexico	0,034344011	0,006020259	0,01391591	0,003423897	0,007804415	0,009474449
Oklahoma	0,011462363	0,003972574	0,002649203	0,003646946	0,00219656	0,007808776
Arizona	0,019773808	0,005307828	0,012539592	0,005143725	0,002968146	0,01248152
Kentucky	-0,000517974	0,002274317	0,007858289	0,000608548	0,005165226	0,004428135
Virginia	-0,01155937	0,000176259	-0,001210852	0,000612583	-0,000485483	-0,001877619
Maryland	-0,015121257	-0,004796805	-0,004609682	-0,000859897	-0,009289165	-0,006964085
Kansas	0,018161407	0,003853963	0,006895413	0,001417255	0,008854153	0,00693401
Missouri	0,003981884	0,002593741	0,004926608	0,001082548	0,001822421	0,006079553
West Virginia	0,001917269	0,003134135	0,003531627	0,002577093	0,002894182	0,00127708
Colorado	0,019574666	0,003208981	0,005039896	0,005237873	0,006984796	0,007606758
New Jersey	-0,026599922	-0,008576842	-0,010512245	-0,010641087	-0,015006925	-0,015526991
Indiana	-0,008859021	-0,001243407	0,005669472	-0,001571627	0,001147113	0,002207095
Ohio	-0,025203997	-0,004932012	-0,009041782	-0,006882886	-0,009748426	0,000906977
Nevada	0,029704688	0,012486356	0,017020753	0,001514885	0,013916566	0,008099235
California	-0,065575585	-0,025604069	-0,021929334	-0,002483561	-0,014403737	-0,007748742
Utah	0,020514348	0,003766841	0,001079973	0,001905664	0,003265034	0,005766116
Rhode Island	-0,003308704	0,001672086	-0,001202957	-0,000712727	0,000400018	-0,003190159
Connecticut	-0,0100916	-0,001824227	-0,003668501	-0,003978037	-0,00369618	-0,007810394
Pennsylvania	-0,026941116	-0,000153697	-0,008935733	-0,004179563	-0,005826661	-0,011857047
Illinois	-0,024059833	-0,004148278	-0,008897168	-0,003977588	-0,0126388	-0,002990446
Massachusetts	-0,019246511	-0,010270834	-0,013034372	-0,008321412	-0,011856361	-0,012728431
Nebraska	0,019914655	0,00218407	0,005422802	-6,39982E-05	0,003168542	0,005493622
Iowa	0,008710249	0,002840534	0,00205984	0,002567354	0,00302537	0,005702454
Wyoming	0,030947041	0,003239985	0,003963582	0,001220181	0,002332927	0,003560971
New York	-0,049724385	-0,021078289	-0,023410663	-0,011026081	-0,022144728	-0,033034359
Vermont	0,00110022	0,000483132	-0,000488083	-0,000161263	0,001923299	-0,001666878
New Hampshire	-0,001215992	-0,00076302	-0,002689647	-0,000419193	-0,000278264	-0,001678341
Michigan	-0,020440507	-0,005536794	0,001355186	0,001742499	-0,014073754	0,003893012
South Dakota	0,023159489	0,004674675	0,005479546	0,000310924	0,004444818	0,003089734
Oregon	0,020521137	0,006338739	0,005906779	0,001727258	0,006248704	0,006080642
Wisconsin	0,000118856	-0,001529774	0,008166054	0,005784041	0,005657793	0,004703677
Michigan	-0,028950181	-0,01496315	-0,020471815	-0,012834761	-0,010045557	-0,011437693
Maine	0,006305853	0,001320298	-0,001971115	0,001434566	-0,00036814	-0,002206237
Washington	0,001893246	-0,002120901	-0,00673723	0,002843459	-0,002353268	-0,00295935
Idaho	0,023287274	0,004873961	0,000320989	-0,000642186	0,001512841	0,004230948
Montana	0,046033997	0,008864862	0,004167012	0,000778614	0,001926592	0,005213728
North Dakota	0,02144655	0,004089351	0,001046375	0,000728197	0,004862091	0,003507007
Minnesota	0,011122647	0,003490225	-0,004960211	0,00332184	0,002047615	0,004376665

Keim area error (*km*) for the USA dataset. Global cartogram evaluation uses the weighed error (*we*), single average error (*se*), and mean quadratic error (*mqe*)

we	38,89%	11,09%	16,02%	8,13%	13,19%	16,78%
se	42,45%	16,02%	20,75%	9,31%	17,31%	23,87%
mqe	7,25%	3,08%	3,65%	1,65%	3,17%	4,24%
	Original	km	km	km	km	km
Distrito	km	Dougenik	Carto-SOM 1	Carto-SOM 2	Carto-SOM 3	Carto-SOM 4
Florida	0,502426973	0,064934106	0,212009951	0,101134643	0,114987422	0,398058917
Louisiana	0,00399967	0,056464728	0,155897578	0,02578733	0,158891331	0,230205297
Georgia	0,191705843	0,026153022	0,126613351	0,005625263	0,023427071	0,03081304
Mississippi	0,235839526	0,125777333	0,288182621	0,005594656	0,103005462	0,218742767
Alabama	0,059334679	0,086339643	0,051052362	0,092837767	0,055242793	0,102671914
South Carolina	0,151246289	0,11764941	0,0479114	0,054021159	0,039589873	0,030064262
Arkansas	0,311925449	0,122832714	0,315090766	0,071542116	0,239786946	0,218917883
Texas	0,09339234	0,063518035	0,050368203	0,105779134	0,029576705	0,072642046
North Carolina	0,266178556	0,031431726	0,069178028	0,049726693	0,017373993	0,10553536
Tennessee	0,168947151	0,003199088	0,030435989	0,082695948	0,033912564	0,0513731
New Mexico	0,731773607	0,323516581	0,525040273	0,213827191	0,382701296	0,42942704
Oklahoma	0,325020128	0,143017512	0,10014579	0,132851934	0,08448041	0,247011251
Arizona	0,352263383	0,127384985	0,256436671	0,12393471	0,075471714	0,255552575
Kentucky	0,018890758	0,075279237	0,219531579	0,021318172	0,156036636	0,136816264
Virginia	0,306208758	0,00356183	0,025174423	0,012270813	0,009943561	0,039585739
Maryland	0,700578453	0,150330433	0,143623786	0,02398911	0,3388221	0,234156679
Kansas	0,4947685	0,172056281	0,271036687	0,070995042	0,323148688	0,272140934
Missouri	0,093398844	0,062886247	0,11305306	0,027245033	0,045027346	0,135914307
West Virginia	0,134490485	0,202559531	0,222532644	0,172778173	0,189997862	0,093795325
Colorado	0,392158057	0,095648974	0,142448503	0,147220373	0,187132624	0,200455634
New Jersey	0,841468496	0,172800391	0,220387175	0,223692559	0,3473479	0,363764032
Indiana	0,267184639	0,030496197	0,11889321	0,03885903	0,026576259	0,049908258
Ohio	0,478238641	0,06758616	0,131299238	0,0969109	0,143028373	0,011508001
Nevada	0,672544324	0,46332771	0,540621216	0,094811725	0,490373775	0,358974015
California	0,384192805	0,121544705	0,102315603	0,010623674	0,064923839	0,033909726
Utah	0,56764549	0,194248566	0,064649709	0,108704497	0,172844089	0,269556336
Rhode Island	0,83814414	0,187276302	0,198724104	0,10891922	0,052246426	0,78455549
Connecticut	0,75234363	0,084139704	0,184935504	0,203718627	0,186591247	0,497643517
Pennsylvania	0,470835895	0,001829572	0,118786518	0,052256745	0,074382073	0,163989207
Illinois	0,390323632	0,050866479	0,11584342	0,048671603	0,172987888	0,036155753
Massachusetts	0,782149642	0,30583546	0,422928396	0,234192912	0,370541984	0,408941931
Nebraska	0,6281342	0,156296677	0,315048288	0,005457897	0,211824359	0,317854913
Iowa	0,301990506	0,123646212	0,092817443	0,113099793	0,130640883	0,220725944
Wyoming	0,901258595	0,488646738	0,538960492	0,264639953	0,407607222	0,512258498
New York	0,612585672	0,191939412	0,217803928	0,091983952	0,203627864	0,337562909
Vermont	0,207197115	0,10294888	0,13114476	0,039832561	0,313593583	0,655498038
New Hampshire	0,164391446	0,097201129	0,454080923	0,051160166	0,033386422	0,242024791
Michigan	0,425941719	0,088035572	0,01941954	0,024831833	0,258919432	0,053828507
South Dakota	0,81637917	0,472966672	0,512653232	0,05632702	0,460418052	0,3723114
Oregon	0,463466416	0,210623563	0,199128543	0,067779128	0,208254922	0,203795281
Wisconsin	0,003200077	0,043100711	0,18070955	0,135119611	0,132561315	0,112726479
Michigan	0,733300219	0,279861159	0,426872609	0,230862934	0,172060319	0,200690198
Maine	0,41764686	0,130554913	0,288953743	0,140269127	0,043698577	0,334966597
Washington	0,044179431	0,054606937	0,196862154	0,064913566	0,060954367	0,077875666
Idaho	0,719938815	0,349817389	0,034220966	0,076298824	0,143102181	0,318359119
Montana	0,881011404	0,587770925	0,401279185	0,111295436	0,23656867	0,456102781
North Dakota	0,830583659	0,483154379	0,19302662	0,142708019	0,526393975	0,444966031
Minnesota	0,245558562	0,092670187	0,169797941	0,08859544	0,056532305	0,113534186

Kocmoud area error (km) for the USA dataset. Global cartogram evaluation uses the weighed error (*we*), single average error (*se*), and mean quadratic error (*mqe*)

we	81,09%	23,24%	32,90%	15,98%	26,64%	32,35%
se	203,37%	45,01%	53,47%	19,55%	44,43%	54,90%
mqe	60,49%	10,74%	11,38%	3,52%	9,55%	10,34%
Distrito	Original kd	kd Dougenik	kd Carto-SOM 1	kd Carto-SOM 2	kd Carto-SOM 3	kd Carto-SOM 4
Florida	-0,668820491	-0,121949528	-0,349848532	-0,183691692	0,259854887	-0,569445124
Louisiana	-0,007967473	0,119687582	0,369380714	-0,050278121	0,377814037	0,5980953
Georgia	-0,321733494	0,053710743	0,289936539	0,011314171	0,047978129	0,063585337
Mississippi	0,617251309	0,287746675	0,809709426	-0,011127061	0,229667981	0,559976299
Alabama	0,126154708	0,188997239	0,107597847	0,20467732	0,116946009	0,228839186
South Carolina	-0,262752271	0,266672707	0,100644836	0,114212193	0,082443681	-0,058373566
Arkansas	0,906661781	0,280066793	0,920094958	0,154109555	0,630841433	0,560550239
Texas	0,206025923	0,135652446	0,106079436	0,23658391	0,060956296	0,156664523
North Carolina	-0,420443949	-0,060947759	0,148638579	0,104657665	-0,034154585	-0,190921726
Tennessee	-0,289058664	-0,006377773	-0,059074001	0,180302153	0,070205993	0,108310442
New Mexico	5,456387784	0,956465663	2,210883338	0,543969947	1,239922564	1,505248477
Oklahoma	0,963051319	0,333769975	0,222582255	0,306411186	0,184551834	0,656082183
Arizona	1,087674753	0,291961478	0,689750719	0,282934872	0,163265342	0,68655641
Kentucky	-0,037081028	0,162815067	0,562563642	0,043565071	0,369771112	0,317003804
Virginia	-0,468851179	0,007149125	-0,049112468	0,024846513	-0,01969132	-0,076156757
Maryland	-0,823929589	-0,261369132	-0,251173135	-0,046854229	-0,506149547	-0,379460214
Kansas	1,958581365	0,415623134	0,743622299	0,152841041	0,954858717	0,747784692
Missouri	0,206041749	0,134212623	0,25492632	0,05601623	0,094300806	0,31458525
West Virginia	0,310777601	0,508024208	0,572455272	0,417731175	0,469129284	0,207006932
Colorado	1,290329045	0,211530635	0,332221454	0,345271787	0,460425968	0,501424666
New Jersey	-0,913910282	-0,294679969	-0,36117583	-0,365602549	-0,515602391	-0,533470634
Indiana	-0,42169804	-0,059187404	0,269872419	-0,074810977	0,05460368	0,105059871
Ohio	-0,64703848	-0,126614905	-0,232121147	-0,176697852	-0,250262157	0,023283954
Nevada	4,107696847	1,726669027	2,353705632	0,209485092	1,924444821	1,119998326
California	-0,555114582	-0,216745181	-0,185637585	-0,021023997	-0,121931422	-0,065595139
Utah	2,625833555	0,482155062	0,138236359	0,243924707	0,417923843	0,738061946
Rhode Island	-0,911946046	0,460860934	-0,331559369	-0,196442118	0,110253187	-0,879272732
Connecticut	-0,858671344	-0,155219302	-0,312144422	-0,338482138	-0,314499617	-0,664568719
Pennsylvania	-0,640228997	-0,003652462	-0,212348855	-0,099323184	-0,138464844	-0,28177101
Illinois	-0,561486007	-0,096808644	-0,207633827	-0,092825253	-0,294952557	-0,069788259
Massachusetts	-0,877759784	-0,468413471	-0,594447896	-0,379507789	-0,540723288	-0,580495082
Nebraska	3,378284315	0,37050151	0,919913862	-0,01085654	0,537505469	0,93192759
Iowa	0,865290539	0,282183323	0,204627928	0,255045138	0,300545264	0,566491192
Wyoming	18,25492755	1,91119046	2,338023027	0,719756137	1,376138388	2,100532746
New York	-0,759755816	-0,322062363	-0,3576995	-0,168471252	-0,338356846	-0,504743226
Vermont	0,52269516	0,229527342	-0,231879711	-0,076613414	0,913725674	-0,791904337
New Hampshire	-0,282364571	-0,177180148	-0,624560733	-0,097340381	-0,064615561	-0,389726184
Michigan	-0,597418132	-0,161824805	0,039608253	0,05092831	-0,411335984	0,113781713
South Dakota	8,89201048	1,794826427	2,103854034	0,119378263	1,706573221	1,18629333
Oregon	1,727632452	0,533645428	0,49727966	0,14541431	0,526065592	0,511916787
Wisconsin	0,006420702	-0,082639597	0,441136718	0,312458492	0,305638466	0,254096343
Michigan	-0,846131802	-0,437330498	-0,598333175	-0,375123708	-0,293603181	-0,33429139
Maine	1,434342263	0,300317788	-0,448353937	0,326309387	-0,083737926	-0,501835173
Washington	0,092442938	-0,103558843	-0,328963787	0,138839712	-0,114904786	-0,144498421
Idaho	5,141296649	1,076058888	0,070867072	-0,141780001	0,334000572	0,934096317
Montana	14,80833343	2,851671366	1,340455099	0,25046667	0,619751011	1,677165338
North Dakota	9,805236674	1,869627445	0,478396498	0,332927455	2,222919254	1,603383058
Minnesota	0,650967853	0,204270123	-0,290303026	0,194415179	0,119839407	0,256150173