Universitat Politècnica de Catalunya
Facultat de Matemàtiques i Estadística

Master thesis

# Multidimensional Scaling for Big Data

Cristian Pachón García

Advisor: Pedro Delicado Useros

Dept. d'Estadística i Investigació Operativa

# Contents

**1 Classical Multidimensional Scaling** **2**
1.1 Introduction to Multidimensional Scaling . . . . . 2
1.2 Principal coordinates . . . . . 3
1.3 Building principal coordinates . . . . . 5
1.4 Procrustes transformation . . . . . 6
1.5 Multidimensional Scaling with R . . . . . 7

**2 Algorithms for Multidimensional Scaling with Big Data** **8**
2.1 Why is it needed? . . . . . 8
2.2 Divde and Conquer Multidimensional Scaling . . . . . 9
2.3 Fast Multidimensional Scaling . . . . . 10
2.4 Multidimensional Scaling based on Gower interpolation . . . . . 10

**3 Simulation study** **11**

**4 Conclusions** **12**

**Bibliography** **13**

**A Code** **14**

**B Problems encountered during the development** **15**

# Chapter 1

# Classical Multidimensional Scaling

## 1.1 Introduction to Multidimensional Scaling

Multidimensional Scaling (MDS) is a family of methods that represents measurements of dissimilarity (or similarity) among pairs of objects as Euclidean distances between points of a low-dimensional space. The data, for example, may be correlations among intelligence tests and the MDS representation is a plane that shows the tests as points. The graphical display of the correlations provided by MDS enables the data analyst to literally "look" at the data and to explore their structure visually. This often shows regularities that remain hidden when studying arrays of numbers.

Given a square matrix $\mathbf{D}$ $n \times n$, the goal of MDS is to obtain a configuration matrix $\mathbf{X}$ $n \times p$ with orthogonal columns that can be interpreted as the matrix of $p$ variables for the $n$ observations, where the Eucliden distance between the rows of $\mathbf{X}$ is approximately equat to $\mathbf{D}$. The columns of $\mathbf{X}$ are called *principal coordinates*.

This approach arises two questions: is it (always) possible to find this low dimensional configuration $\mathbf{X}$? How is it obtained? In general, it is not possible to find a set of $p$ variables that reproduces *exactly* the initial distance. However, it is possible to find a set of variables which distance is approximately the initial distance matrix $\mathbf{D}$.

As classical example, consider the distances between European cities as in the Table (1.1). One would like to get a representation in a 2-dimensional space such that the distances would be almost the same as in the Table (1.1). The representation of these coordinates are displayed in Figure (1.1).

| | Athens | Barcelona | Brussels | Calais | Cherbourg | $\cdots$ |
|---|---|---|---|---|---|---|
| Athens | 0 | 3313 | 2963 | 3175 | 3339 | $\cdots$ |
| Barcelona | 3313 | 0 | 1318 | 1326 | 1294 | $\cdots$ |
| Brussels | 2963 | 1318 | 0 | 204 | 583 | $\cdots$ |
| Calais | 3175 | 1326 | 204 | 0 | 460 | $\cdots$ |
| Cherbourg | 3339 | 1294 | 583 | 460 | 0 | $\cdots$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ |

Table 1.1: Distances between European cities (just 5 of them are shown)

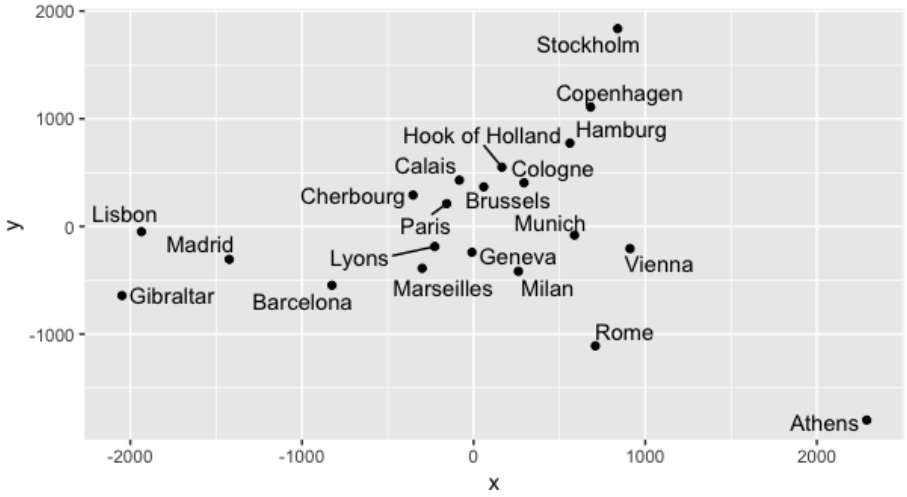



Figure 1.1: MDS on the Eurepean cities.

## 1.2 Principal coordinates

Given a matrix $\mathbf{X}$ $n \times p$, the matrix of $n$ individuals over $p$ variables, it is possible to obtain a new one with mean equal to 0 by column from the previous one:

$$\widetilde{\mathbf{X}} = \Big(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}'\Big)\mathbf{X} = \mathbf{P}\mathbf{X},$$

where

$$\mathbf{P} = \Big(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}'\Big).$$

This new matrix, $\widetilde{\mathbf{X}}$, has the same dimensions as the orginial one but its columns mean is $\mathbf{0}$. From this matrix, it is possible to build two square semi-positive definite matrices: the covariance matrix $\mathbf{S}$, defined as $\widetilde{\mathbf{X}}'\widetilde{\mathbf{X}}/n$ and the cross-prodructs matrix $Q = \widetilde{\mathbf{X}}\widetilde{\mathbf{X}}'$. The last matrix can be interpreted as a similarity matrix between the $n$ elements. The term $ij$ is obtained as follows:

$$q_{ij} = \sum_{s=1}^{p} x_{is}x_{js} = \mathbf{x_i}'\mathbf{x_j} \tag{1.1}$$

where $\mathbf{x_i}'$ is the i-th row from $\widetilde{\mathbf{X}}$. Given the scalar product formula, $\mathbf{x_i}'\mathbf{x_j} =\mid \mathbf{x_i} \mid\mid \mathbf{x_i} \mid \cos\theta_{ij}$, if the elements $i$ and $j$ have similar coordinates, then $\cos\theta_{ij} \simeq 1$ and $q_{ij}$ will be large. On the contrary, if the elements are very different, then $\cos\theta_{ij} \simeq 0$ and $q_{ij}$ will be small. So, $\widetilde{\mathbf{X}}\widetilde{\mathbf{X}}'$ can be interpreted as the similarity matrix between the elements.

The distances between elements can be deduced from the similarity matrix. The Eucliden distance between two elements is calculated in the following way:

$$d_{ij}^2 = \sum_{s=1}^{p}(x_{is} - x_{js})^2 = \sum_{s=1}^{p} x_{is}^2 + \sum_{s=1}^{p} x_{js}^2 - 2\sum_{s=1}^{p} x_{is}x_{js}. \tag{1.2}$$

This expression can be obtained directly from the matrix $\mathbf{Q}$:

$$d_{ij}^2 = q_{ii} + q_{jj} - 2q_{ij}. \tag{1.3}$$

We have just seen that, given the matrix $\widetilde{\mathbf{X}}$, it is possible to get the similarity matrix $\mathbf{Q} = \widetilde{\mathbf{X}}\widetilde{\mathbf{X}}'$ and from it, to get the distance matrix $\mathbf{D}$. Let diag($\mathbf{Q}$) be the vector that contains the diagonal terms of $\mathbf{Q}$ and $\mathbf{1}$ be the vector of ones, the matrix $\mathbf{D}$ is given by

$$\mathbf{D} = \text{diag}(\mathbf{Q})\mathbf{1}' + \mathbf{1}\,\text{diag}(\mathbf{Q})' - 2\mathbf{Q}.$$

The problem we are dealing with goes in the opposite direction. We want to rebuild $\widetilde{\mathbf{X}}$ from a square distance matrix $\mathbf{D}$, with elements $d_{ij}^2$. The first step is to obtain $\mathbf{Q}$ and afterwards, to get $\widetilde{\mathbf{X}}$. The theory needed to get the solution can be found in (Peña 2002). Here, we summarise it.

The first step is to find out a way to obtain the matrix $\mathbf{Q}$ given $\mathbf{D}$. We can assume without loss of generality that the mean of the variables is equal to 0. This is a consequence of the fact that the distance between two points remains the same if the variables are expressed in terms of the mean:

$$d_{ij}^2 = \sum_{s=1}^{p}(x_{is} - x_{js})^2 = \sum_{s=1}^{p}[(x_{is} - \overline{x_s}) - (x_{js} - \overline{x_s})]^2. \tag{1.4}$$

The previous condition means that we are lookig for a matrix $\widetilde{\mathbf{X}}$ such that $\widetilde{\mathbf{X}}'\mathbf{1} = 0$. It also means that $\mathbf{Q1} = 0$, i.e, the sum of all the elements of a column of $\mathbf{Q}$ is 0. Since the matrix is symmetric, the previous condition should state for the rows as well.

To establish these constrains, we sum up (1.3) at row level:

$$\sum_{i=1}^{n} d_{ij}^2 = \sum_{i=1}^{n} q_{ii} + nq_{jj} = t + nq_{jj} \tag{1.5}$$

where $t = \sum_{i=1}^{n} q_{ii} = \text{Trace}(\mathbf{Q})$, and we have used that the condition $\mathbf{Q1} = 0$ implies $\sum_{i=1}^{n} q_{ij} = 0$. Summing up (1.3) at column level:

$$\sum_{j=1}^{n} d_{ij}^2 = t + nq_{ii}. \tag{1.6}$$

Summing up (1.5) we obtain:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}^2 = 2nt \tag{1.7}$$

Replacing in (1.3) $q_{jj}$ obtained in (1.5) and $q_{ii}$ obtained in (1.6), we have the following expression:

$$d_{ij}^2 = \frac{1}{n}\sum_{i=1}^{n} d_{ij}^2 - \frac{t}{n} + \frac{1}{n}\sum_{j=1}^{n} d_{ij}^2 - \frac{t}{n} - 2q_{ij} \tag{1.8}$$

Let $d_{i.}^2 = \frac{1}{n}\sum_{j=1}^{n} d_{ij}^2$ and $d_{.j}^2 = \frac{1}{n}\sum_{i=1}^{n} d_{ij}^2$ be the row-mean and column-mean of the elements of $\mathbf{D}$. Using (1.7), we have that

$$d_{ij}^2 = d_{i.}^2 + d_{.j}^2 - d_{..}^2 - 2q_{ij} \tag{1.9}$$

where $d_{..}$ is the mean of all the elements of $\mathbf{D}$, given by

$$d_{..}^2 = \frac{1}{n^2}\sum\sum d_{ij}^2.$$

Finally, from (1.9) we get the expression:

$$q_{ij} = -\frac{1}{2}(d_{ij}^2 - d_{i.}^2 - d_{.j}^2 + d_{..}^2). \tag{1.10}$$

The previous expression shows how to build the matrix of similarities $\mathbf{Q}$ from the distance matrix $\mathbf{D}$.

The next step is to obtain the matrix $\mathbf{X}$ given the matrix $\mathbf{Q}$. Let's assume that the similarity matrix is positive definite of range $p$. Therefore, it can be represented as

$$\mathbf{Q} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}'$$

where $\mathbf{V}$ is a $n \times p$ matrix that contains the eigenvectors with not nulls eigenvalues of $\mathbf{Q}$. $\mathbf{\Lambda}$ is a diagonal matrix $p \times p$ that contains the eigenvalues.

Re-writing the previous expression, we obtain

$$\mathbf{Q} = (\mathbf{V}\mathbf{\Lambda}^{1/2})(\mathbf{\Lambda}^{1/2}\mathbf{V}'). \tag{1.11}$$

Getting

$$\mathbf{Y} = \mathbf{V}\mathbf{\Lambda}^{1/2}.$$

We have obtained a matrix with dimensions $n \times p$ with $p$ uncorrelated variables that reproduce the initial metric. It is important to notice that if one starts from $\mathbf{X}$ (i.e $\mathbf{X}$ is known) and calculates from these variables the distance matrix in (1.2) and after that it is applied the method explained, the matrix obtained is not the same as $\mathbf{X}$, but its principal components. This happens since the distance between the rows in a matrix does not change if:

- The row-mean values are modified by adding the same row vector to all the rows in $\mathbf{X}$.
- Rows are rotated, i.e, $\mathbf{X}$ is postmultiplied by an orthogonal matrix.

By (1.3), the distance is a function of the terms of the similarity matrix $\mathbf{Q}$ and this matrix is invariant given any rotation, reflexion or translation of the variables

$$\mathbf{Q} = \widetilde{\mathbf{X}}\widetilde{\mathbf{X}'} = \widetilde{\mathbf{X}}\mathbf{A}\mathbf{A}'\widetilde{\mathbf{X}'}$$

for any orthogonal $\mathbf{A}$ matrix. The matrix $\mathbf{Q}$ only contains information about the space generated by the variables $\mathbf{X}$. Any rotation, reflexion or translation keeps the distance unchaged. Therefore, the solution is not unique.

## 1.3 Building principal coordinates

Let $\mathbf{D}$ be a square distance matrix. The process to obtain the *principal coordinates* is:

1. Build the matrix $\mathbf{Q} = -\frac{\mathbf{1}}{\mathbf{2}}\mathbf{PDP}$ of cross-products.
2. Obtain the eigenvalues of $\mathbf{Q}$. Take the $r$ greatest eigenvalues. Since $\mathbf{P1} = 0$, where $\mathbf{1}$ is a vector of ones, range$(\mathbf{Q}) = n - 1$, being the vector $\mathbf{1}$ an eigenvector with eigenvalue 0.

3. Obtain the coordinates of the rows in the variables $\mathbf{v_i}\sqrt{\lambda_i}$, where $\lambda_i$ is an eigenvalue of $\mathbf{Q}$ and $\mathbf{v_i}$ is the associated unitary eigenvector. This implies that $\mathbf{Q}$ is apporximated by:

$$\mathbf{Q} \approx (\mathbf{V_r}\mathbf{\Lambda}^{1/2})(\mathbf{\Lambda_r}^{1/2}\mathbf{V_r'}).$$

4. Take as coordinates of the points the following variables:

$$\mathbf{Y_r} = \mathbf{V_r}\mathbf{\Lambda_r}^{1/2}.$$

The method can also be applied if the initial information is not a distance matrix but a similarity matrix. A *similarity function*, $s_{ij}$, between two element $i$ and $j$ is defined as:

- $s_{ii} = 1$.
- $0 \leq s_{ij} \leq 1$.
- $s_{ij} = s_{ji}$.

If the initial information is $\mathbf{Q}$, a similariy matrix, then $q_{ii} = 1$, $q_{ij} = q_{ji}$ and $0 \leq q_{ij} \leq 1$. The associated distance matrix is

$$d_{ij}^2 = q_{ii} + q_{jj} - 2q_{qij} = 2(1 - q_{ij}),$$

and it is easy to see that $\sqrt{2(1 - q_{ij})}$ is a distance and it verifies the triangle inequality.

## 1.4 Procrustes transformation

As we have mentioned before, the MDS solution is not unique. One example of it is shown in Figure (1.2).

Since rotations, translations and reflections are distance-preserving functions, one can find two different MDS configurations for the same set of data. How is it possible to align both solutions? This problem is solved by means of *Procrustes transformations*.

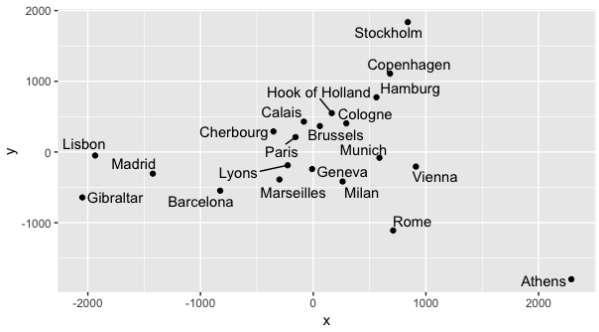
The Procrustes problem is concern with fitting a configuration (testee) to another (target) as closely as possible. In the simple case, both configurations have the same dimensionality and the same number of points, which can be brought into 1-1 correspondence by substantive considerations. Under orthogonal transformations, the testee can be fitted it to the target. In addition to such rigid motions, one may also allow for dilations and for shifts.

All the details are developed in (Borg and Groenen 2005). This is out of the scope of this thesis. However, since it has been a repeatdly used tool, we briefly summarise it.

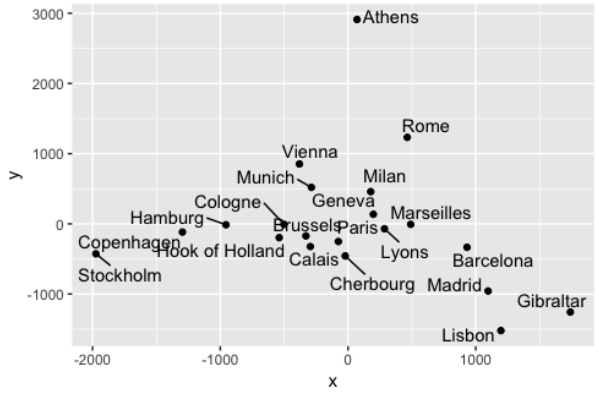
Let $\mathbf{A}$ and $\mathbf{B}$ be two different MDS configurations for the same set of data. Without loss of generality, let's assume that the target is $\mathbf{A}$ and the testee is $\mathbf{B}$. One wants to obtain $s$, $\mathbf{T}$ and $t$ such that

$$\mathbf{A} = s\mathbf{B}\mathbf{T} + \mathbf{1}\mathbf{t}'$$

where $\mathbf{T}$ is an orthogonal matrix. As mentioned before, in (Borg and Groenen 2005) are all the details needed to estimate these parameters.




(a)




(b)

Figure 1.2: Two different solutions of MDS.

## 1.5 Multidimensional Scaling with R

All the algorithms have been coded in R, since it has a widely statistics packages already implemented. We have used two packages for developing our MDS approaches:

- Package: stats. From this one we have used the function cmdscale to do the MDS. The output of this function is:
  - The new coordinates for the individuals.
  - All the eigenvalues found.
- Package: MCMCpack. From this one we have used the function procrustes to do the Procrustes transformation. The output of this function is:
  - The dilation coeficient $s$.
  - The orthogonal matrix $\mathbf{T}$.
  - The translation vector $\mathbf{t}$.

# Chapter 2

# Algorithms for Multidimensional Scaling with Big Data

## 2.1 Why is it needed?

In this chapter we present the algorithms developed so that MDS can be applied when we are dealing with large datasets. The natural question here is why we need them if there are already some implementations. To answer this question, let's take a look at the Figures (2.1) and (2.2).

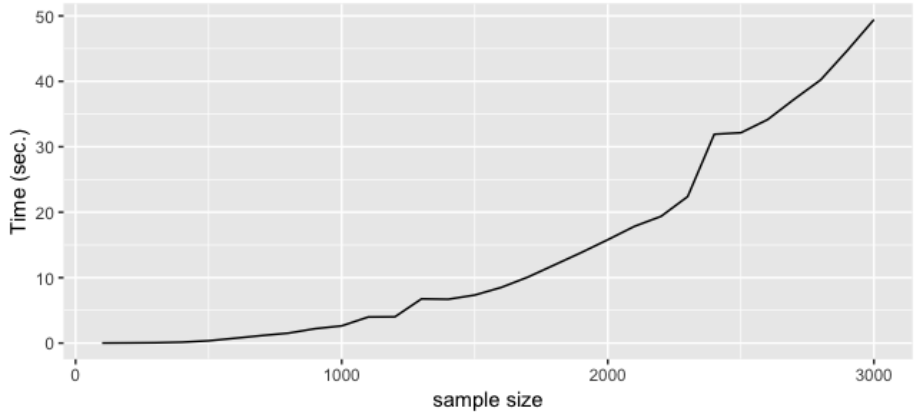



Figure 2.1: Elapsed time to compute MDS.

Figure (2.1) shows the time needed to compute MDS as a function of the sample size. As we can see, the time grows exponentially. Apart of the time issue, there is another one related to the memory needed to compute the distance matrix. Figure (2.2) points out that it is required at least 400MB to store the distance matrix when the dataset is close to 10.000 observations.

In order to solve these problems, we have developed three algorithms:

- *Divide and Conquer MDS:* Before this thesis, *Pedro Delicado* had done some work about MDS with big datasets and he had already created a first approach, which is this one. The algorithm is based on the idea of dividng and conquering. Given a big data set, it is divided into $p$ small ones, then MDS is perform over all the partitions and finally the $p$ solutions are stitched so that all the points lie on the same system of coordinates.

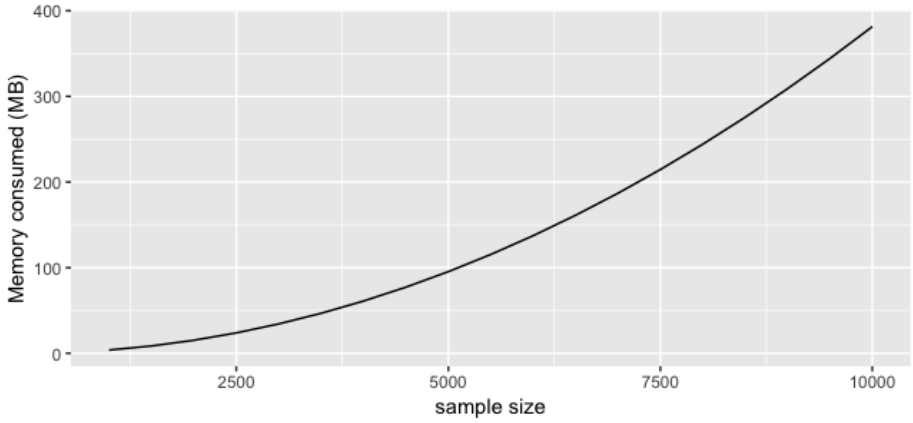



Figure 2.2: Memory consumed to compute distance.

- *Fast MDS:* during the phase of gathering information, we found an article that solved the problem of scalability (Tynia, Jinze, Leonard, and Wei 2006). The authors use a divide and conquer idea together with recursive programming.
- *MDS based on Gower interpolation formula:* this algorithm uses Gower interpolation formula which allows to add a new set of points to an existing MDS configuration (Handl 1996).

In the next sections we are going to provide a description of the algorithms. If further details about the implementation are needed, the code is provided in Appendix A.

## 2.2 Divde and Conquer Multidimensional Scaling

- The first step is to divide the original dataset into $p$ partitions: $\mathbf{X_1}, \ldots, \mathbf{X_p}$. The number of partitions, $p$, is also the number of steps needed to compute the algorithm.
- Calculate the MDS for the first partition: **MDS(1)**. This solution will be used as a guide to align the MDS for the remaining partitions. We will use a new variable, **cum-mds**, that will be growing as long as new partitions are used. Before adding a new MDS, it will be aligned and, after that, added.
- Define **cum-mds** equal to **MDS(1)** and start iterating until the last partition is reached.
- Given a step $k$, $1 < k \leq p$, partitions $k$ and *k-1* are joint, i.e, $\mathbf{X_k} \cup \mathbf{X_{k_1}}$. MDS is calculated on this union, obtaining $\mathbf{MDS_{k,k-1}}$. In order to append the rows of the *k-th* partition to **cum-mds**, the following steps are performed:
  - Take the rows of the partition *k-1* from $\mathbf{MDS_{k,k-1}}$: $\mathbf{MDS_{k,k-1}}\Big|_{k-1}$.
  - Take the rows of the partition *k-1* from **cum-mds**: $\mathbf{cum\text{-}mds}\Big|_{k-1}$.
  - Apply Procrustes to align both solutions. It means that a scalar number $s$, a vector $\mathbf{t}$ and an orthogonal matrix $\mathbf{T}$ are obtained so that

$$\mathbf{cum\text{-}mds}\Big|_{k-1} = s\mathbf{MDS_{k,k-1}}\Big|_{k-1}\mathbf{T} + \mathbf{1t}'.$$

– Take the rows of the partition $k$ from $\mathbf{MDS_{k,k-1}} : \mathbf{MDS_{k,k-1}}\Big|_k$.

– Use the previous Procrustes parameters to append the rows of $\mathbf{MDS_{k,k-1}}\Big|_k$ with **cum-mds**:

$$\mathbf{cum\text{-}mds}\Big|_k := s\mathbf{MDS_{k,k-1}}\Big|_k \mathbf{T} + \mathbf{1t}'.$$

– Add the previous dataset to **cum-mds**, i.e:

$$\mathbf{cum\text{-}mds} = \mathbf{cum\text{-}mds} \bigcup \mathbf{cum\text{-}mds}\Big|_k$$

As we have seen, the algorithm depends on $p$, the number of partitions. How many of them are needed? To answer this question, Let $l \times l$ be the largest matrix that allows to run MDS eficiently. If $n$ is the number of rows of $\mathbf{X}$, then $p$ is $\frac{2n}{l}$.

## 2.3 Fast Multidimensional Scaling

## 2.4 Multidimensional Scaling based on Gower interpolation

# Chapter 3

# Simulation study

# Chapter 4

# Conclusions

# Bibliography

Borg, I. and P. Groenen (2005). *Modern Multidimensional Scaling: Theory and Applications*. Springer.

Handl, A. (1996, October). Biplots : J. C. Gower and D. J. Hand (1996) London: Chapman & Hall, ISBN 0-412-71630-5, [pound sign] 32.00, pp. 277. *Computational Statistics & Data Analysis 22*(6), 655–651.

Peña, D. (2002). *Análisis de datos multivariantes*. Madrid, Spain: McGraw Hill.

Tynia, Y., L. Jinze, M. Leonard, and W. Wei (2006). A fast approximation to multidimensional scaling.

# Appendix A

# Code

# Appendix B

# Problems encountered during the development