UNIVERSITY OF BUENOS AIRES
Faculty of Exact and Natural Sciences

Metric Multidimensional Scaling in Problems with Large Data Sets

Thesis presented to obtain the Master's degree from the University of Buenos Aires in Mathematical Statistics

Eng. Pedro Camilo Cosatto Ammann

Thesis Advisor: Dr. Daniela Andrea Rodríguez

Defense Date: July 31, 2023.

Note: This is a quick and easy translation made with CHAT GPT. You may find some typos or inconsistencies. Original is in Spanish.

Abstract

In this work, we describe and apply multidimensional scaling (MDS) methods to samples with a large amount of data. Multidimensional scaling is a set of object representation techniques based on distances, similarities, or dissimilarities between them. These methods have severe limitations when the sample size increases, due to computational difficulties. We analyze three different algorithms to overcome this problem: two of them based on the idea of 'divide and conquer', and one of them based on an interpolation method. Then, we apply one of them to a clustering problem. The studied methods accurately and precisely reproduce the solution that would be obtained with classic methods, although some aspects to improve were discovered, especially with the appearance of outliers. Based on the application problem, we believe that these variants provide advantages to MDS as a dimension reduction method, putting it at the same level as other techniques commonly used in the treatment of large samples, such as Principal Component Analysis or t-SNE.

Keywords: Multidimensional scaling, classical scaling, big data, dimensionality reduction, Procrustes transformations, interpolation, unsupervised learning

Índice general

Capítulo 1

Multidimensional Scaling

1.1. Introduction

Multidimensional Scaling, or MDS, is a set of methods aimed at finding coordinates in a vector space for objects or individuals solely based on proximity or dissimilarity data between them. These methods emerged in the field of psychometrics in the early 20th century when there was a need to spatially locate individuals responding to behavioral tests. Visualization in two or three dimensions allowed for a clearer understanding of results and the validation of experimental hypotheses. Nowadays, MDS covers a wide range of practical problems, including:

- Mapping perceptions for market exploration, where a group of consumers judges the similarity between different products. The goal is to map consumers and products to identify groups and relationships.
- Ordering species in biology and ecology based on various measures of similarity. For example, visualizing the proximity between different microorganisms like viruses or bacteria.
- Genome mapping.
- Mapping similarities between catastrophic weather events such as earthquakes, floods, and forest fires to understand associations or common features among them.
- Mapping correlations between different time series, especially climatic ones.
- Locating objects in space to track them, using only a few simple proximity measurements with existing objects.
- Simplified representation of networks and graphs, such as communication networks, electrical circuits and wiring, social networks, protein interaction graphs, Markov chains, among others.
- Dimension reduction of samples with many variables. This can be used for descriptive purposes or as part of statistical learning techniques. While reducing the number of variables may result in information loss, it often simplifies computation or improves the quality of results in many cases.

The variety of MDS methods differ in aspects such as the type of geometry in which the representation is performed, the way of finding a solution, and the modeling of measurement errors, among others. In this chapter, we will present the basic concepts of scaling, the most common methods for obtaining the solution, and provide some examples as an introduction. More application examples can be found in [8].

1.1.1. Proximity Measures

In multivariate problems, one typically starts with a matrix $\mathbf{X}_{n\times p}$ containing the n individuals in the sample as rows and the p variables as columns, representing a configuration of points in $\mathbb{R}^p$. In MDS, on the other hand, the starting information is a measure of proximity or dissimilarity between objects, usually in the form of an $n \times n$ matrix. The goal is to *produce* a configuration like $\mathbf{X}$. Proximity information can be obtained in various ways.

Dissimilarities and Distances

Given a set of objects $\{O_1, O_2, ..., O_n\}$, we will call the *dissimilarity* δ_{ij} of O_i to O_j any real number that satisfies:

(a) $\delta_{ij} \geq 0$ (non-negativity),
(b) $\delta_{ij} = \delta_{ji}$ (symmetry),
(c) $\delta_{ii} = 0$ for $i = 1, 2, ..., n$ (identity).

A higher dissimilarity δ_{ij} implies that the objects O_i and O_j are *less similar*. If pairs of objects have associated variable measurements, let's say vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ in $\mathbb{R}^p$, we will call the *distance* $d_{ij} = d(\mathbf{x}_i, \mathbf{x}_j)$ from O_i to O_j a function of these variables that allows obtaining a dissimilarity between them. If a distance also satisfies the following properties, we call it a *metric distance*:

(d) $d_{ij} = d_{ji} = 0$ if and only if $x_i = x_j$, and
(e) for a third object O_k, $d_{ik} + d_{jk} \geq d_{ij}$ (triangular inequality).

An example of dissimilarity is the score of an observer on a scale, for example, on a range from 0 (identical) to 5 (not similar at all), for two objects. Another example could be the count of how many attributes two objects have in common, out of a predefined total number of attributes. In these examples, the measures are not considered distances because they do not have associated variables. The most common distances are the Euclidean distance

$$d_{ij} = d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{a = 1p(x_{ia} - x_{ja})^2} = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^t(\mathbf{x}_i - \mathbf{x}_j)}, \tag{1.1}$$

which measures the *straight-line* distance between points represented in a Euclidean space, or the Mahalanobis distance

$$d_{ij} = d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^t\mathbf{S}^{-1}(\mathbf{x}_i - \mathbf{x}_j)}, \tag{1.2}$$

where $\mathbf{S}$ is the covariance matrix obtained in the configuration where $\mathbf{x}_i$ and $\mathbf{x}_j$ are found, used to standardize the original Euclidean distance.

Similarities

On the other hand, in some problems, proximity measures increase as two objects become more *similar*. For these cases, we use a *similarity* s_{ij} between O_i and O_j with the following properties:

(a) $s_{ij} \geq 0$,
(b) $s_{ij} = s_{ji}$ (symmetry),

(c) $s_{ij} \leq s_{ii}$ and $s_{ji} \leq s_{jj}$ for all $i \neq j$ (identity).

This last condition could be interpreted as *no object is more similar to another than to itself*. Depending on the problem, additional constraints may be imposed on the similarity coefficient. For example, a common restriction is to impose that $s_{ij} \leq 1$, where the value 1 indicates that two objects are identical. Another frequent case is to impose that $s_{ij} \in [-1, 1]$. The advantage of these cases is that dissimilarities δ_{ij} can be derived very easily. For example, with δ_{ij} $= 1 - s_{ij}$, or with the transformation

$$\delta_{ij} = \sqrt{s_{ii} + s_{jj} - 2s_{ij}} \tag{1.3}$$

which also returns a dissimilarity with distance properties and corresponds to Euclidean distances under certain conditions. A comprehensive approach to these topics can be found in [6] and [2]. We will use the following notation to differentiate between different cases:

- $\mathbf{\Delta}_{n\times n}$ for a dissimilarity matrix δ_{ij} in general.
- $\mathbf{D}_{n\times n}$ for a distance matrix d_{ij}. It is necessarily linked to a point configuration, so we could write, for example, $\mathbf{D_X}$ to indicate that it is a distance associated with $\mathbf{X}$.
- $\mathbf{S}_{n\times n}$ to refer to a similarity matrix s_{ij}.

Definición (k-dimensional Scaling)**.** A configuration $\mathbf{Z}_{n\times k}$ is a *k-dimensional scaling* of a dissimilarity matrix $\mathbf{\Delta}_{n\times n}$ if a distance matrix $\mathbf{D_Z}$ approximates it optimally in the sense of some loss function $L(\mathbf{D_Z}, \mathbf{\Delta}) : \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times n} \to \mathbb{R}$. The distance used for scaling is typically the Euclidean distance, and that is what we will use in this work. We will call $\mathcal{Z}$ the vector space where the scaling is found.

The MDS problem consists of finding a scaling $\mathbf{Z}$ to visualize the points. The columns of $\mathbf{Z}$ could be thought of as observations of new variables $Z_1, Z_2, ..., Z_k$ for the n objects, but unlike other methods, we will not emphasize their interpretation. In practical applications, it is usually imposed that $k \leq 3$ to be able to graph the objects in a visible space. Another interesting aspect of MDS methods is that they help determine how many dimensions are important for representing objects, formulate hypotheses about these structures, or confirm previous hypotheses. In this chapter, we will provide an introduction to the main MDS techniques and show some examples.

1.2. Classic Scaling

The first method of multidimensional scaling is classic scaling (CMDS - Classic Multidimensional Scaling), introduced by Torgerson (1952, 1958) and Gower (1966). It is closely related to other techniques such as principal component analysis (PCA) or factor analysis (FA).

1.2.1. Representation of Euclidean Distances

As an introduction, let's assume that we start with the Euclidean distances d_{ij} of a group of n objects. Our first problem is to construct a vector configuration $\mathbf{X}$ that has this distance matrix. Rewrite 1.1, squaring it, and observe that it consists of three parts:

$$
\begin{aligned}
d_{ij}^2 = d^2(\mathbf{x}_i, \mathbf{x}_j) &= (\mathbf{x}_i - \mathbf{x}_j)^t(\mathbf{x}_i - \mathbf{x}_j) \\
&= \mathbf{x}_i^t\mathbf{x}_i + \mathbf{x}_j^t\mathbf{x}_j - 2\mathbf{x}_i^t\mathbf{x}_j \\
&= \sum_{a=1}^{p} x_{ia}^2 + \sum_{a=1}^{p} x_{ja}^2 - 2\sum_{a=1}^{p} x_{ia}\, x_{ja}.
\end{aligned}
\tag{1.4}
$$

The first two terms of the last equality are the squared norms of both vectors, $b_{ii} = ||\mathbf{x}_i||^2$ and $b_{jj} = ||\mathbf{x}_j||^2$, while the third term has the dot product, let's call it $b_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$, between them. For the dot product, we will consider that the reference point is the center of the point configuration, or the vector given by the means of the columns of $\mathbf{X}$. For simplicity, we also assume that this vector is the origin. Now let's look for a matrix expression for this relationship. We have $\mathbf{D}^{(2)} = \{d_{ij}^2\}$, the squared distance matrix, and it is convenient to define $\mathbf{c}$ as a column vector containing the n norms $b_{ii} = ||\mathbf{x}_i||^2$ for each point, and $\mathbf{1}_n$ a column vector of ones. In this way, $\mathbf{D}^{(2)}$ can be written as:

$$
\mathbf{D}^{(2)} = \mathbf{c}\mathbf{1}^t + \mathbf{1}\mathbf{c}^t - 2\mathbf{X}\mathbf{X}^t. \tag{1.5}
$$

The first two terms $\mathbf{c}\mathbf{1}^t$ and $\mathbf{1}\ \mathbf{c}^t$ are two $n \times n$ matrices that have the squared norms of the vectors. In the third term, the matrix $\mathbf{X}\mathbf{X}^t$ is symmetric and contains all the inner products b_{ij} off the diagonal and the squared norms b_{ii} on the diagonal. Note that if we apply the centering operator $\mathbf{H} = \mathbf{I}_n - n^{-1}\mathbf{1}_n\mathbf{1}_n^t$ to the left and right of $\mathbf{D}^{(2)}$, which consists of centering its rows and then centering the resulting columns, we get:

$$
\begin{aligned}
\mathbf{H}\mathbf{D}^{(2)}\mathbf{H} &= \mathbf{H}(\mathbf{c}\mathbf{1}^t + \mathbf{1}\mathbf{c}^t - 2\mathbf{X}\mathbf{X}^t)\mathbf{H} \\
&= \mathbf{H}\mathbf{c}\mathbf{1}^t\mathbf{H} + \mathbf{H}\mathbf{1}\mathbf{c}^t\mathbf{H} - 2\mathbf{H}\mathbf{X}\mathbf{X}^t\mathbf{H} \\
&= -2\mathbf{X}\mathbf{X}^t.
\end{aligned}
$$

The first and second terms of the second equality are also centered on rows and columns, so the whole expression is centered. This is an interesting feature, as it indicates that the distances between the points $\mathbf{x}_i$ are proportional to the off-diagonal elements of the centered distance matrix, and their expression can be given by:

$$
-\frac{1}{2}\mathbf{H}\mathbf{D}^{(2)}\mathbf{H} = \mathbf{X}\mathbf{X}^t. \tag{1.6}
$$

1.2.2. Eigenvalue Decomposition of the Double-Centered Matrix

The next step is to calculate the eigenvalues and eigenvectors of the double-centered matrix $\mathbf{H}\mathbf{D}^{(2)}\mathbf{H}$. The eigenvalues of this matrix are related to the singular values of $\mathbf{X}$ and can be used to obtain a low-dimensional representation of the data.

1.3. Distance Scaling

So far, the idea behind a set of dissimilarities was implicitly assumed to have an unknown configuration of points that one wants to approximate. From this perspective, a broader view of the MDS problem emerges with the works of Kruskal (1964) and Shepard (1962), suggesting a direct regression for distances.

General Model Setup

Given a set of objects $\{O_1, O_2, ..., O_n\}$, assume they have an associated vector configuration in finite dimensions in a space $\mathcal{Z}$, with their Euclidean distances d_{ij}. For these objects, there are *observations* of dissimilarity δ_{ij} in the form

$$\delta_{ij} = d_{ij} + \varepsilon_{ij}, \tag{1.7}$$

where ε_{ij} are random variables capturing measurement errors, sampling effects, or bias due to the measurement method. With this, one seeks a scaling $\mathbf{Z}$ whose distances, let's call them $d_{ij(\mathbf{Z})}$ to differentiate them from the true distances, *approximate* them by minimizing the sum of squares

$$L(\mathbf{D_Z}, \mathbf{\Delta}) = \sigma^2_{abs} = j < i\big(\delta_{ij} - d_{ij(\mathbf{Z})}\big)^2, \tag{1.8}$$

called *Stress*. To find $\mathbf{Z}$, normal equations are formulated (see Gower in [4]), and optimization algorithms are implemented. A coverage of different resolution methods that emerged in the field of MDS for minimizing *Stress* and its variants can be found in [2] and [8]. Kruskal's approach is nowadays the common denominator for most MDS methods. Some authors call it *least squares scaling* or *distance scaling* to distinguish it from CMDS.

Incorporation of a Mapping Function

Instead of trying to fit the original dissimilarities directly, a more flexible approach is to transform them beforehand. A *mapping function* f, non-decreasing and continuous, is used so that now

$$f(\delta_{ij}) = d_{ij} + \varepsilon_{ij}. \tag{1.9}$$

A simple case is to propose a linear relationship in parameters β_0 and β_1. With this idea, observe that without loss of generality, 1.9 can be reformulated as follows:

$$d_{ij} = f(\delta_{ij}) + \varepsilon_{ij} = \beta_0 + \beta_1\ \delta_{ij} + \varepsilon_{ij}. \tag{1.10}$$

Transformations like this allow giving metric properties to dissimilarities that may not have them (see Chapter 19 of [1]). Other examples of parametric modeling involve monotonic polynomials in δ_{ij}, splines, and exponential or logarithmic functions. If the mapping is strictly increasing, the scaling is called *metric*, while if it only seeks to preserve the order of dissimilarities, the scaling is called *ordinal* or non-metric. The features of the modeling will largely depend on the specific application.

We can see that the problem has two parts that are worked on simultaneously. On the one hand, the proposed mapping function must be adjusted, whether parametric or not. This produces new dissimilarities $\hat{f}(\delta_{ij})$ called *disparities* $\hat{d}_{ij}$. Then, a configuration of points in $\mathbb{R}^{n\times k}$ must be found whose Euclidean distances approximate these disparities. Suppose f is parametric and depends on certain coefficients $\vartheta \in \Theta$, and let $\mathcal{U}$ be the set of all configurations of n points of dimension k. We can reformulate the problem 1.8 as

$$\min_{\mathbf{U}\in\mathcal{U},\vartheta\in\Theta}\Big\{\text{Stress}(\mathbf{D_U},\boldsymbol{\Delta}) = j < i e_{ij}^2 = j < i\Big(\hat{f}_\vartheta(\delta_{ij}) - d_{ij(\mathbf{U})}\Big)^2\Big\}, \tag{1.11}$$

where e_{ij} are the residuals of the fit. To find the solution with a linear model like 1.10, in this work, we will use the SMACOF method, *stress majorization of a complicated function*, applied to MDS by De Leeuw (1977,1988). This method is detailed in Chapter 8 of [1].

ANOVA can be postulated for multidimensional scaling (see Gower in [4]) as

$$j < l\delta_{ij}^2 = j < l d_{ij(\mathbf{Z})}^2 + j < l\Big(\delta_{ij} - d_{ij(\mathbf{Z})}\Big)^2 \tag{1.12}$$

where δ_{ij} can be replaced by $\hat{d}_{ij}$ for the case of transformed dissimilarities. From this decomposition, a widely used error measure in MDS emerges, Kruskal's *Stress-1*

$$\sigma_1 = \sqrt{\frac{j < i\Big(\delta_{ij} - d_{ij}(\mathbf{Z})\Big)^2}{j < i d_{ij}(\mathbf{Z})^2}} = \frac{||\boldsymbol{\Delta} - \mathbf{D_Z}||_F}{||\mathbf{D_Z}||_F}, \tag{1.13}$$

which plays a role analogous to the normalized *Strain* coefficient of **??** for CMDS. Some general features and properties of distance scaling solutions are:

(a) If the SMACOF algorithm is applied, the optimal solution has columns centered at the origin.
(b) If $\mathbf{T}$ is an orthogonal matrix of $k \times k$ and $\mathbf{t}$ is a vector in $\mathbb{R}^k$, the transformation $\mathbf{Z}' = \mathbf{ZT} + \mathbf{t}$ is also an optimal solution, just like in CMDS.
(c) Optimal solutions for different values of $k \leq r$ are not nested as in CMDS; changing the dimensionality requires the optimization process to be performed again.

Let's continue with the example of Argentine routes to show the peculiarities of this approach and illustrate the differences with CMDS.

Ejemplo (Continuation - Argentine Cities)**.** For the example of Argentine cities, we propose scaling the distances by the given routes in the table **??**, adjusting the model given by 1.10. The SMACOF algorithm was used for resolution.

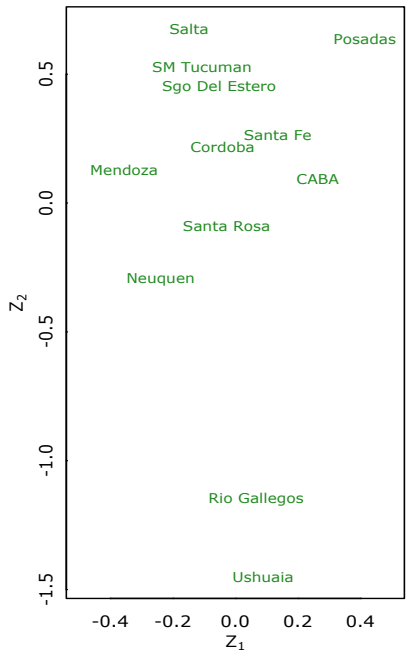
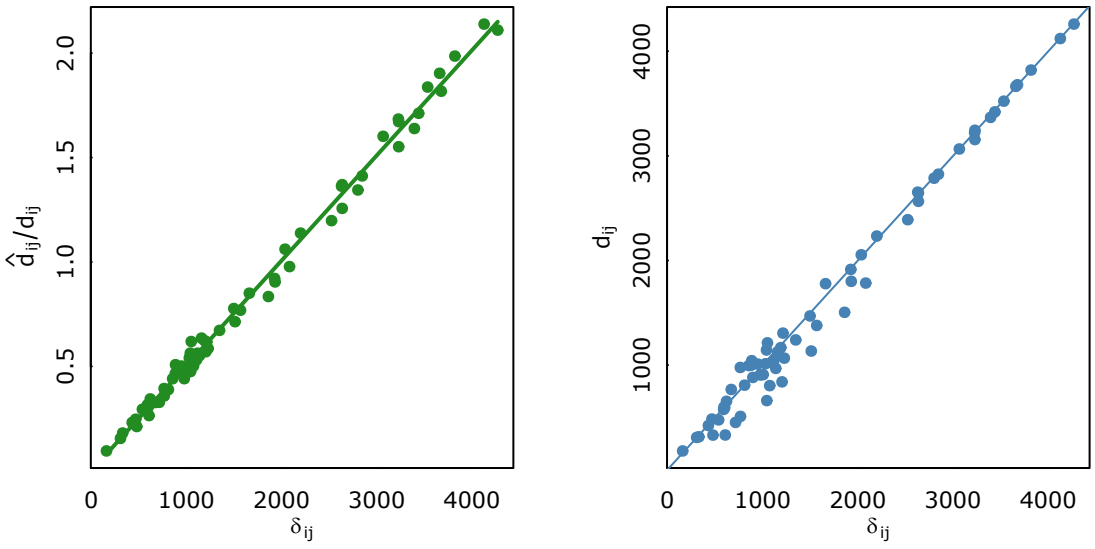


Figura 1.1: Scaling distances by routes allowing a linear transformation



(a) Distance scaling **(b)** CMDS

Figura 1.2: Shepard diagrams, MDS of distances by route

The graphs in figure 1.2 show the fit of distances (or disparities) in relation to the original dissimilarities.

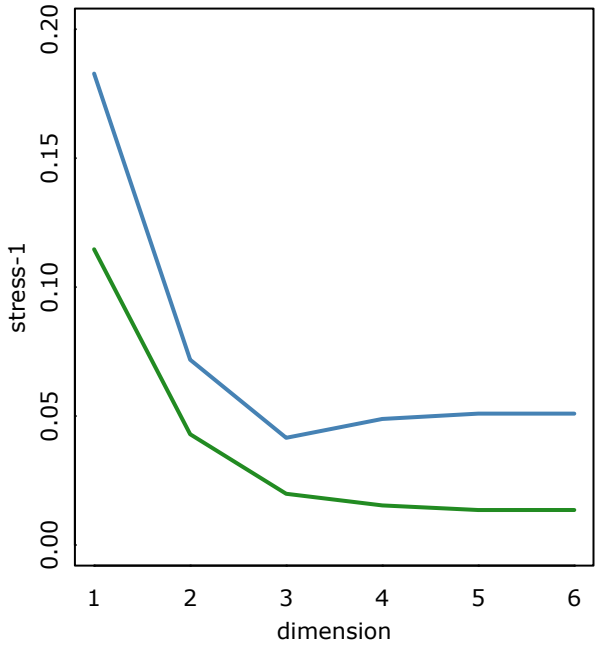


Figura 1.3: Stress-1 as a function of k

The graph shows how the Stress-1 coefficient given in 1.13 decreases with distance scaling (in green) and with CMDS (in blue). It is noteworthy that CMDS does not work by minimizing this loss.

1.4. MDS as a Dimension Reduction Technique

Given a configuration of points $\mathbf{X}_{n\times p}$, let's show an example of how to use multidimensional scaling to find a dimension reduction similar to principal components and construct a biplot.

Ejemplo (Wine Quality)**.** Data from 100 bottles of red wine from different European wineries are available. For each bottle, 4 quality parameters were considered: Free sulfur dioxide (`free.sulfur.dioxide`), density (`density`), `pH`, and `alcohol`. For each bottle, a discrete scale from 1 to 5 was used for the named parameters, where 1 indicates a very low or nil level, and 5 indicates a very high level.

obs	`total.sulfur.dioxide`	`density`	`pH`	`alcohol`
1	2	1	5	5
2	2	1	4	1
3	3	2	1	3
4	1	2	2	1
5	2	1	4	2
6	...	...	...	...

We seek a two-dimensional representation of the data. First, we perform a principal component analysis with the correlation matrix of centered data, obtaining:

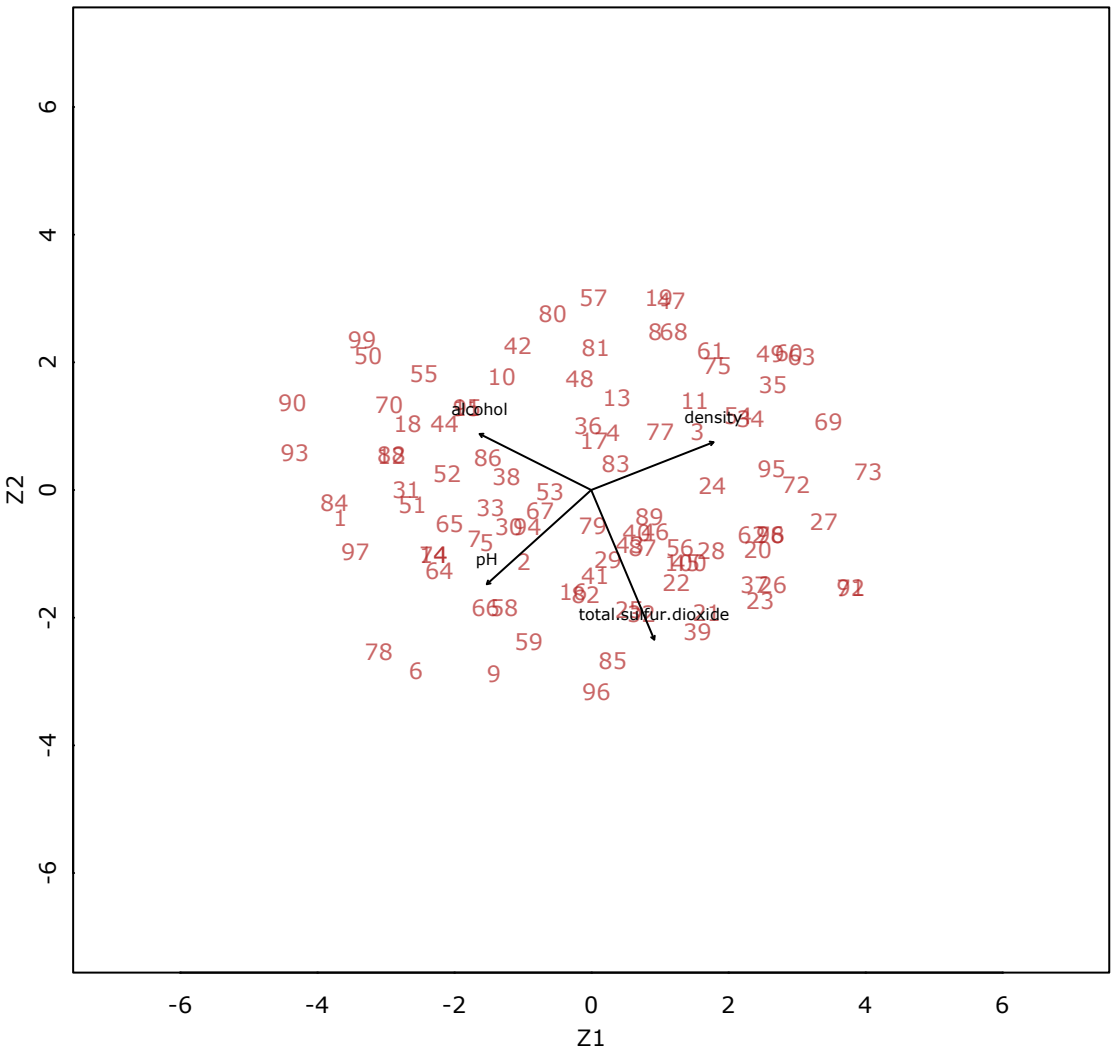


Figura 1.4: A Principal Component Biplot

Let $\hat{\mathbf{\Sigma}}$ be the covariance matrix of the data. We decompose it as

$$\hat{\mathbf{\Sigma}} = \mathbf{\Gamma}_2 \mathbf{\Lambda}_2 \mathbf{\Gamma}_2^{\mathrm{t}},$$

where $\mathbf{\Gamma}_2$ has the eigenvectors associated with the first two eigenvalues in columns, and $\mathbf{\Lambda}_2$ is diagonal. For constructing the biplot in Figure 1.4, we project $\mathbf{X}$ into the 2-dimensional space according to $\mathbf{\Lambda}_2$, forming the points representing observations. Then, we project the matrix $\mathrm{diag}(3, 3, 3, 3)$ with the same transformation to represent canonical marks of size 3 with arrows. This scale was chosen to adjust the relationship between the size of points and the size of arrows. The representation given by principal component analysis and the biplot is such that the Euclidean distances of the scores approximate the Euclidean distances of the original data, while the angles between the arrows preserve the correlations between variables.

Linear MDS Biplot

In this particular problem, it might be interesting to propose a different metric for the observations. Consider the Minkowski distance of order k for $\mathbf{x}_i$ and $\mathbf{x}_j$ given by

$$\left(\sum_{k=1}^{p} |X_{ik} - X_{jk}|^k\right)^{1/k}, \tag{1.14}$$

which, according to the order k, produces different forms for the distance contour lines. Adjusting this distance can be of interest for specific problems where Euclidean distance may not be the most suitable. For k = 1, the Minkowski distance is called *city-block* or Manhattan distance, as it measures the distance from one point to another by traversing a grid. For k = 2, it is the classic Euclidean distance, while for higher values of k, more importance is given to the larger of the two components in the distance calculation.

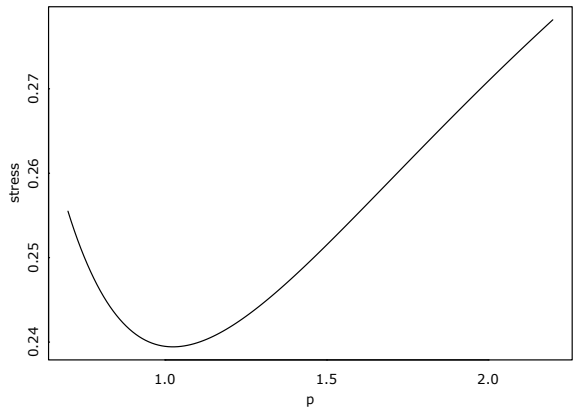


Figura 1.5: Adjustment of the Minkowski Power

In the graph 1.5, we see the levels of Stress-1 for different MDS configurations obtained by varying the value of k. The optimal level for this problem is k = 1.02. We perform MDS with this metric and seek a biplot analogous to that in Figure 1.4. Let $\mathbf{Z}$ be the two-dimensional scaling obtained, and we need to have projection directions to represent canonical marks in $\mathbb{R}^2$. We call $\mathbf{B}$ the matrix with p rows and 2 columns that has projection directions for the original data. We seek an approximate solution of

$$\mathbf{XB} = \mathbf{Z}$$

by least squares, obtaining $\hat{\mathbf{B}} = (\mathbf{X}^t\mathbf{X})^{-1}\mathbf{X}^t\mathbf{Z}$. The points of the biplot correspond to the coordinates of $\mathbf{Z}$, and the arrows mark the points resulting from projecting $\text{diag}(3, 3, 3, 3)$ according to $\hat{\mathbf{B}}$.

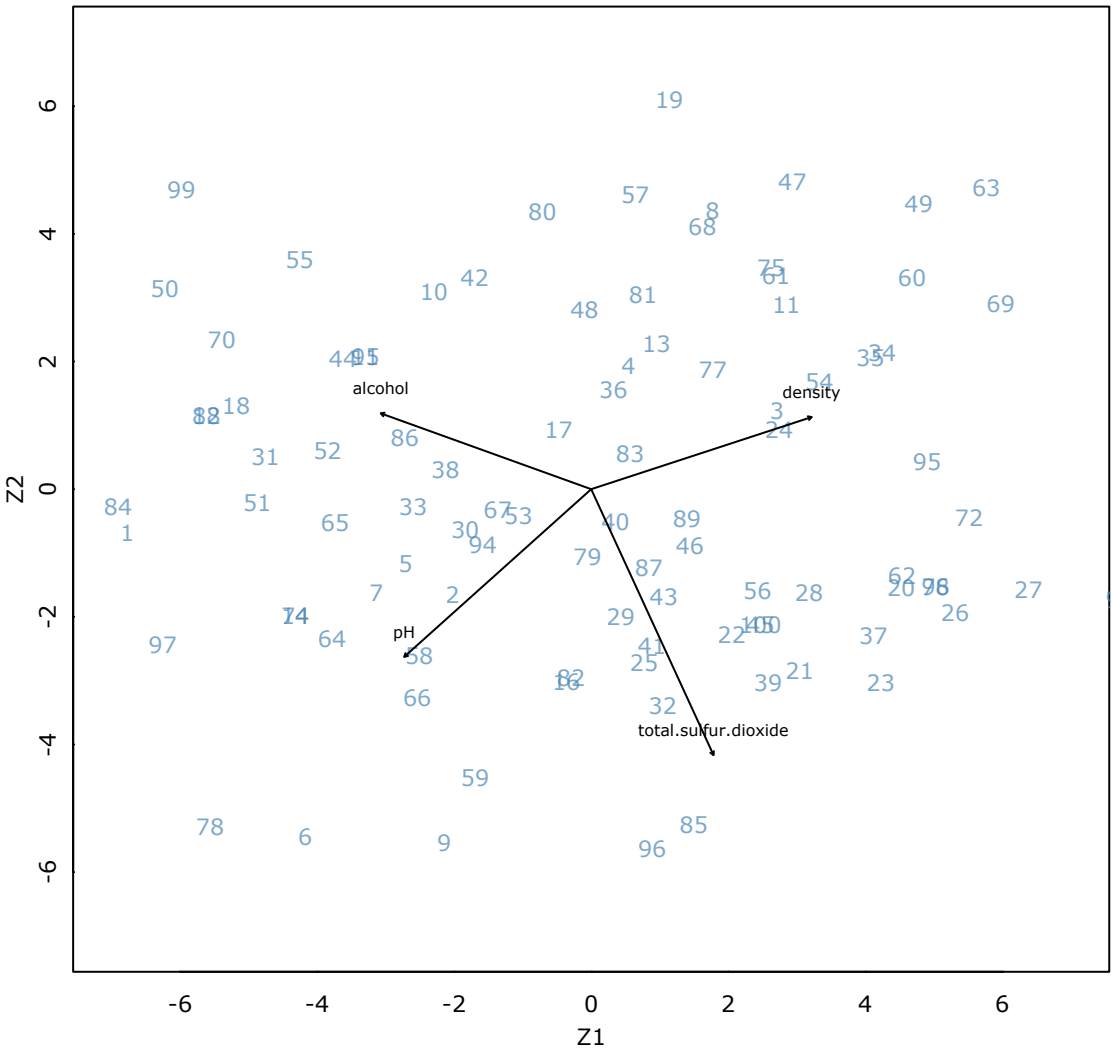


Figura 1.6: MDS Biplot

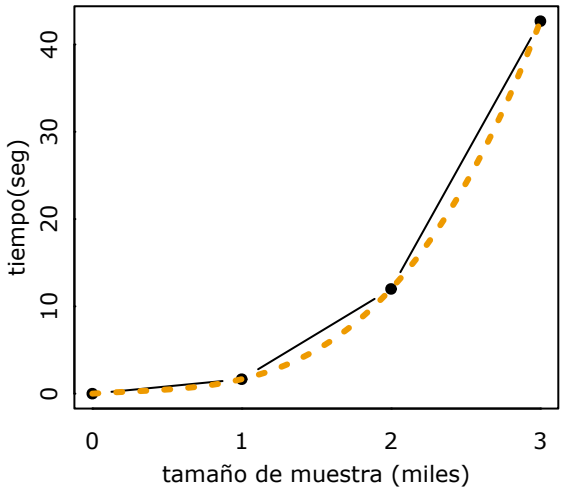
The representation in Figure 1.6 has an approximation error due to the linear regression made when representing canonical marks. This can hide nonlinear structures in the data. In [4], strategies are suggested to create biplots in MDS where the canonical marks of variables are represented by non-linear trajectories. One of these methods is based on incorporating the canonical vectors $\mathbf{e}_k$ and their multiples into the original sample and performing MDS with them.

Capítulo 2

Challenges with Large Datasets

2.1. Introduction

As the sample size increases, multidimensional scaling becomes computationally expensive with standard algorithms because it requires significant memory and time. For the CMDS method, the computation time is $O(n^3)$ since it requires eigenvalue and eigenvector decomposition of a dissimilarity matrix of size $n \times n$. If the problem also requires the calculation of this dissimilarity matrix, the requirement is even higher.



The graph shows the average processing times with CMDS, in dimension $k = 5$, for increasing sample sizes, performed on a MacBook Air with an Intel Core i5 processor and 8 GB of RAM. The orange curve represents the polynomial

$$g(n) = 1.2n - 1.48n^2 + 1.94n^3$$

overlaid on the data points. For a sample of 50,000 data points, the estimated processing time increases to more than 2.5 days. This significantly limits the applications of MDS. In this chapter, we will explore strategies to address this issue: first, two methods based on the idea of *divide and conquer*; and then two methods based on interpolation. In all cases, we start with a set of objects $\{O_1, O_2, ..., O_n\}$, with dissimilarity matrix $\mathbf{\Delta}$, for which we seek a k-dimensional scaling $\mathbf{Z}$.

2.2. Methods of *Divide and Conquer*

A first group of methods, called *Divide and Conquer MDS* (DCO-MDS), is based on dividing the initial sample into small blocks and then combining the different configurations into a unique solution. The procedure is as follows:

- Choose a group of $\mathtt{c}$ reference points, called *landmarks*, randomly from the initial sample.
- Divide the remaining points into $\mathtt{p}$ blocks and include the reference points in each one. Groups of a maximum size $\mathtt{l}$, small enough to perform MDS efficiently, will be formed.
- Apply MDS to each group separately and obtain the scalings $\mathbf{Z}_\mathtt{1}, \mathbf{Z}_\mathtt{2}, ..., \mathbf{Z}_\mathtt{p}$. In these configurations, the solution for the $\mathtt{c}$ reference points will be repeated, but with different orientation and location. Let $\mathbf{Z}_\mathrm{i}^\mathtt{c}$ be these solutions for the i-th group.
- Starting from the second group, align the points $\mathbf{Z}_2^\mathtt{c}$ and find the rotation, translation, and/or dilation necessary to make the vectors coincide with those of $\mathbf{Z}_1^\mathtt{c}$.
- Apply the found transformation to the remaining points of $\mathbf{Z}_\mathtt{2}$. This way, the first two solutions are combined.
- Repeat the operation for the remaining groups $\mathbf{Z}_3^\mathtt{c}, \mathbf{Z}_4^\mathtt{c}, ..., \mathbf{Z}_\mathrm{p}^\mathtt{c}$, all in relation to the first one, and thus obtain the final solution.

The mechanism used to align the different solutions may vary. In [3], Procrustes transformations are used, while in [9], QR decomposition is used. Note that this algorithm can be applied to any MDS method, as long as it is possible to correctly merge the solutions.

The choice of $\mathtt{l}$ and $\mathtt{c}$ will affect the overall efficiency of the algorithm. Additionally, the value of $\mathtt{c}$ must be greater than or equal to the desired dimension k for scaling. Applying this method for a fixed value of $\mathtt{l}$, the computation times are O(n). An important advantage of this method in terms of computation is that it allows *parallelization* of the process, as the scalings of different groups can be performed independently.

2.2.1. Alignment with Procrustes Transformations

The strategy used in [3] to align the different solutions consists of finding an optimal transformation. Suppose we have two point clouds given by the matrices $\mathbf{A}$ and $\mathbf{B}$, both of $\mathbb{R}^{\mathrm{n}\times\mathrm{p}}$ with $\mathrm{n} \geq \mathrm{p}$ in the same space $\mathcal{R}$. We want to transform $\mathbf{B}$ to resemble $\mathbf{A}$ *as closely as possible*, point by point. In other words, we want to *deform* one point cloud to take the shape of the other as much as possible, setting necessary constraints. This procedure is called the Procrustes transformation,[1] and is commonly used in image processing.

A first step is to propose a rigid transformation: a rotation or reflection. For this, we need to find an orthogonal matrix $\mathbf{T}$ that satisfies

$$\mathbf{A} \approx \mathbf{BT}.$$

[1] Quoting [2]: *It is probably the only statistical method that bears the name of a villain. According to Greek mythology, any traveler on the road from Eleusis to Athens was in for a surprise if he accepted lodging with Damastes, who had an inn by the side of the road. If the bed was too large for the guests, Damastes stretched their bodies to the correct size; if the bed was too small, he cut off their limbs. For this, he was given the name "Procrustes,"which means "stretcher..Ev entually, Procrustes met the same fate as his guests at the hands of Theseus.*

We will seek to minimize $||\mathbf{A} - \mathbf{BT}||_{\mathrm{F}}^2$. Let's see that

$$\begin{aligned} ||\mathbf{A} - \mathbf{BT}||_{\mathrm{F}}^2 = \mathrm{tr}(\mathbf{A} - \mathbf{BT})^{\mathrm{t}}(\mathbf{A} - \mathbf{BT}) &= \mathrm{tr}\ \mathbf{A}^{\mathrm{t}}\mathbf{A} - \mathrm{tr}\ \mathbf{T}^{\mathrm{t}}\mathbf{B}^{\mathrm{t}}\mathbf{BT} - 2\mathrm{tr}\ \mathbf{A}^{\mathrm{t}}\mathbf{BT} \\ &= \mathrm{tr}\ \mathbf{A}^{\mathrm{t}}\mathbf{A} - \mathrm{tr}\ \mathbf{B}^{\mathrm{t}}\mathbf{B} - 2\mathrm{tr}\ \mathbf{A}^{\mathrm{t}}\mathbf{BT}. \end{aligned} \tag{2.1}$$

In the second term, we permute the factors using trace properties, then we have $\mathbf{T}^{\mathrm{t}}\mathbf{B}^{\mathrm{t}}\mathbf{BT} = \mathbf{B}^{\mathrm{t}}\mathbf{TT}^{\mathrm{t}}\mathbf{B}$. Since $\mathbf{T}$ is orthogonal, $\mathbf{T}^{\mathrm{t}}\mathbf{T} = \mathbf{I}$. Now, since $\mathrm{tr}\ \mathbf{CD} = \mathrm{tr}\ \mathbf{DC}$,

$$\begin{aligned} \mathrm{tr}\ \mathbf{A}^{\mathrm{t}}\mathbf{A} - \mathrm{tr}\ \mathbf{B}^{\mathrm{t}}\mathbf{B} - 2\mathrm{tr}\ \mathbf{A}^{\mathrm{t}}\mathbf{BT} &= \mathrm{tr}\ \mathbf{A}^{\mathrm{t}}\mathbf{A} - \mathrm{tr}\ \mathbf{B}^{\mathrm{t}}\mathbf{B} - 2\mathrm{tr}\ \mathbf{TA}^{\mathrm{t}}\mathbf{B} \\ &= \mathrm{tr}\ \mathbf{A}^{\mathrm{t}}\mathbf{A} - \mathrm{tr}\ \mathbf{B}^{\mathrm{t}}\mathbf{B} - 2\mathrm{tr}\ \mathbf{A}^{\mathrm{t}}\mathbf{BT}. \end{aligned} \tag{2.2}$$

In the last equality, we used the cyclic invariance of the trace. Taking the derivative with respect to $\mathbf{T}$ and setting it to zero, we obtain

$$\frac{\partial}{\partial \mathbf{T}}||\mathbf{A} - \mathbf{BT}||_{\mathrm{F}}^2 = -2\mathbf{A}^{\mathrm{t}}\mathbf{B} + 2\mathbf{TA}^{\mathrm{t}}\mathbf{BT} = 0. \tag{2.3}$$

Thus, we find that $\mathbf{A}^{\mathrm{t}}\mathbf{B} = \mathbf{TA}^{\mathrm{t}}\mathbf{BT}$.

Notice that we are not imposing a scaling factor in this initial transformation. To find the optimal scale factor s, we will project $\mathbf{A}$ onto $\mathbf{B}$ using $\mathbf{T}$. This gives the solution for s:

$$\arg\min_{\mathrm{s}} ||\mathbf{A} - \mathrm{s}\mathbf{BT}||_{\mathrm{F}}^2 = \mathrm{s}^* = \frac{\mathrm{tr}(\mathbf{A}^{\mathrm{t}}\mathbf{B})}{\mathrm{tr}(\mathbf{B}^{\mathrm{t}}\mathbf{B})}.$$

Thus, $\mathbf{T}$ and s are found, and the transformation $\mathbf{T}^* = \mathrm{s}^*\mathbf{T}$ is optimal for the given point clouds.

This method can be extended to consider translation as well. In that case, we seek to minimize

$$||\mathbf{A} - \mathbf{BT} - \mathbf{c}^{\mathrm{t}}||_{\mathrm{F}}^2,$$

where $\mathbf{c}$ is the translation vector. The solution for $\mathbf{c}^*$ is given by

$$\mathbf{c}^* = \frac{1}{\mathrm{n}}\sum_{\mathrm{i}=1}^{\mathrm{n}}(\mathbf{a}_{\mathrm{i}} - \mathbf{b}_{\mathrm{i}}\mathbf{T}),$$

where $\mathbf{a}_{\mathrm{i}}$ and $\mathbf{b}_{\mathrm{i}}$ are the columns of $\mathbf{A}$ and $\mathbf{B}$, respectively.

In the particular case of aligning a group of solutions $\mathbf{Z}_2$ with a reference solution $\mathbf{Z}_1$, this method is used for every pair $(\mathbf{Z}_{\mathrm{i}}^{\mathsf{c}}, \mathbf{Z}_1^{\mathsf{c}})$, for $\mathrm{i} \geq 2$. It is important to notice that, due to the properties of Procrustes transformations, the transformations of the points of $\mathbf{Z}_2$ are applied *in the same order as the groups are aligned.*

2.2.2. Efficiency Analysis

For the problem at hand, finding an appropriate value of c can be a challenge. This value must be large enough so that the scalings can be correctly aligned. On the other hand, it must be small enough for the method to be efficient in terms of computation. This trade-off will depend on the properties of the sample and the algorithm used for the MDS.

Another important aspect is the value of `l`. The existence of a scaling solution for the sample divided into groups depends on this value. Additionally, the size of the groups must be large enough for the alignment to be efficient, but small enough for the individual MDS problems to be computationally feasible.

The computational complexity of the proposed method is dominated by the application of the MDS algorithm to each group. If `l` is kept constant, the computation times are O(n). As mentioned before, the advantage of this method is that it allows parallelization of the process. If the sample can be efficiently divided into groups, the overall computation time can be significantly reduced.

The results obtained with this method will depend on the choice of `c`, `l`, and the MDS algorithm used. It is recommended to test different configurations to find the most suitable parameters for a given dataset.

2.3. Methods Based on Interpolation

Let's consider a new group of objects $O_{n+1}, O_{n+2}, ..., O_{n+m}$ that we want to include in an existing k-dimensional scaling for a given set of points. We have dissimilarities $\mathbf{\Delta}21$ between the elements of the second group and the first group. The problem of *interpolation* involves finding coordinates $\mathbf{Z}2$ by adding the new points, either in blocks or one at a time, to represent the given dissimilarities. In large data problems, this idea can be used as follows:

- Randomly choose a group of `m` reference points from the initial sample.
- Divide the remaining points into `p` groups of maximum size `m`.
- Apply MDS to the first group and obtain a configuration $\mathbf{Z}_1$ with the desired dimensionality k in a space $\mathcal{Z}$.
- For the subsequent blocks, one by one, interpolate the points to the first solution in $\mathcal{Z}$, obtaining successively $\mathbf{Z}_2, \mathbf{Z}_3, ..., \mathbf{Z}_p$. The final scaling is the superposition of all the blocks. Depending on the method used, it may not be possible to interpolate in blocks, and the new points must be incorporated one at a time, i.e., there will be $p = n - m$ groups of size 1.

In [3], a method proposed by Gower for interpolation is suggested, involving a decomposition of distances. On the other hand, in [7] and [5], optimization methods similar to those used in distance scaling (seen in Section **??**) are employed. Note that this methodology also allows for parallelizing the process since block interpolation is done concerning the original group.

2.3.1. Gower Interpolation

As the first method, we present a series of results found in Appendices 5, 6, and 7 of [4], recommended in [3] as a strategy for performing MDS. We will show a method for interpolating a point and a block of points external to an existing configuration, based on a decomposition of the Euclidean distance similar to classical ANOVA.

Interpolation of a Point

Let $\mathbf{A}$ be a configuration of n objects given by the matrix $\mathbf{A}$ in a k-dimensional space $\mathcal{S}$. A new object is incorporated, and its coordinates are sought based on squared distances from the elements of $\mathbf{A}$ given by

$$\mathbf{d}^2_{n+1} = \left(d^2_{n+1,1}\ d^2_{n+1,2}\ ...\ d^2_{n+1,n}\right)^t.$$

To add the new object, an additional dimension will be necessary in the space $\mathcal{S}$ to adjust the distances. Let $\left(b_1,\ b_2,\ ...\ b_k,\ b_{k+1}\right)^t = (\mathbf{b},\ b_{k+1})^t$ be the sought vector. Also, impose $a_{j_{k+1}} = 0$ for the new coordinates of all vectors in $\mathbf{A}$. We have

$$\begin{aligned} d^2_{n+1,j} &= \sum_{c=1}^{k+1}(b_c - a_{j_c})^2 \\ &= \sum_{c=1}^{k+1} b_c^2 + \sum_{c=1}^{k} a^2_{j_c} - 2\sum_{c=1}^{k} b_c\ a_{j_c} \end{aligned} \tag{2.4}$$

Note that if the points in the configuration $\mathbf{A}$ are centered at the origin, the first and second terms of the equation are the distances from $(\mathbf{b},\ b_{k+1})^t$ and $\mathbf{a}_j$ to the origin, respectively. We can rewrite the first term (see [4]) as

$$\sum_{c=1}^{k+1} b_c^2 = \frac{1}{n}\mathbf{d}^2_{n+1}\mathbf{1} - \frac{1}{2n^2}\mathbf{1}^t\mathbf{D}^{(2)}_A\mathbf{1}, \tag{2.5}$$

and the second term as

$$\sum_{c=1}^{k} a^2_{j_c} = \overline{d^2_{j\bullet}} - \frac{1}{2n^2}\mathbf{1}^t\mathbf{D}^{(2)}_A\mathbf{1}, \tag{2.6}$$

where $\overline{d^2_{j\bullet}}$ is the average squared distances from $\mathbf{aj}$ to the other points in $\mathbf{A}$. Observe that $\frac{1}{n^2}\mathbf{1}^t\mathbf{D}^{(2)}\mathrm{A}\mathbf{1}$ and $\frac{1}{n}\mathbf{d}^2_{n+1}\mathbf{1}$ are scalars: the average of the elements of the matrix $\mathbf{D}^{(2)}_A$ and the elements of $d^2_{n+1,j}$, respectively. Using these expressions, we can write the complete vector $\mathbf{d}^2_{n+1}$ as

$$\mathbf{d}n + 1^2 = \frac{1}{n}\mathbf{1}\mathbf{1}^t\mathbf{d}^2n + 1 + \frac{1}{n}\mathbf{D}^{(2)}_A\mathbf{1} - \frac{1}{n^2}\left(\mathbf{1}^t\mathbf{D}^{(2)}_A\mathbf{1}\right)\mathbf{1} - 2\mathbf{A}\mathbf{b}. \tag{2.7}$$

From this expression, the coordinates $\mathbf{b}$, i.e., the first k coordinates of the new vector, can be obtained as:

$$\mathbf{b} = \frac{1}{2}\left(\mathbf{A}^t\mathbf{A}\right)^{-1}\mathbf{A}^t\left[\frac{1}{n}\mathbf{D}^{(2)}_A\mathbf{1} - \frac{1}{n^2}\left(\mathbf{1}^t\mathbf{D}^{(2)}_A\mathbf{1}\right)\mathbf{1} - \mathbf{H}\mathbf{d}^2_{n+1}\right], \tag{2.8}$$

where $\mathbf{H}$ is the centering matrix of size n × n. From this, the component b_{k+1} can also be found, although it is not used in the MDS application according to the consulted literature (the development can be seen in [4]).

Interpolation of a Block of Points

Now, let's find the configuration $\mathbf{B}$ corresponding to the complete block of m points, which will now have dimension k + m. We start from a distance matrix of size (n+m) × (n+m), in blocks as follows:

$$\mathbf{D}^{(2)} = \begin{bmatrix} \mathbf{D}^{(2)}_{\mathrm{A}} & \mathbf{D}^{(2)}_{\mathrm{AB}} \\ \mathbf{D}^{(2)}_{\mathrm{BA}} & \mathbf{D}^{(2)}_{\mathrm{B}} \end{bmatrix}.$$

Let

$$\mathbf{B} = \big[\mathbf{B}1;\ \mathbf{B}2\big],$$

where $\mathbf{B}_1$ is the matrix of m × k with the coordinates of the new objects in the space $\mathcal{S}$ with the original dimension, and $\mathbf{B}_2$ is the matrix of m × m with the new dimensions. Additionally, a block of n × m with zeros is added to the existing configuration $\mathbf{A}$. With a reasoning analogous to the one already shown for the interpolation of a point, we can obtain

$$\mathbf{B}_1 = \frac{1}{2}\Big[\mathbf{1}\mathrm{m}\mathbf{1}\mathrm{n}^{\mathrm{t}}\Big(\frac{1}{\mathrm{n}}\mathbf{D}^{(2)}\mathrm{A} - \frac{1}{\mathrm{n}^2}\mathbf{1_n}^{\mathrm{t}}\mathbf{D}^{(2)}\mathrm{A}\mathbf{1}\mathrm{n}\Big) - \mathbf{D}^{(2)}\mathrm{BA}\mathbf{H}\Big]\mathbf{A}(\mathbf{A}^{\mathrm{t}}\mathbf{A})^{-1}, \tag{2.9}$$

where $\mathbf{H}$ is the centering matrix of size n × n. This expression is equivalent to what is called the *Gower interpolation formula* in [3]. The coordinates of the block $\mathbf{B}_2$ can be found in the corresponding appendix of [4].

Application in MDS

- To apply the Gower method, MDS will be performed with an initial block of reference points, and the remaining points must be interpolated. This can be done point by point with Equation 2.8 or in blocks with Equation 2.9.
- In [3], it is not mentioned how the additional dimension obtained with interpolation influences the fit of distances.
- As the Gower formulas are derived for Euclidean distances of the elements of the new block, they must be replaced by $\mathbf{\Delta}^2_{\mathrm{BA}}$ for MDS, introducing additional errors. The influence of using particular dissimilarities on the results was not studied.

Algoritmo 1 Interpolation of dissimilarities with the Gower formula

---

Input: Existing configuration $\mathbf{A}$ in $\mathbb{R}^{\mathrm{n}\times\mathrm{k}}$, its Euclidean distances $\mathbf{D}^{(2)}_{\mathrm{A}}$, and dissimilarities of the new group $\mathbf{\Delta}^{(2)}\mathrm{BA}$ of m × n.

1. Apply 2.9 replacing $\mathbf{D}^{(2)}\mathrm{BA}$ with $\mathbf{\Delta}^{(2)}_{\mathrm{BA}}$.

Output: Configuration $\mathbf{B}$ in $\mathbb{R}^{\mathrm{m}\times\mathrm{k}}$.

Ejemplo (Simulation - *Continuation*)**.** We continue the example shown for DCO-MDS but applying interpolation. The same 4 points are taken as the initial block, and the Gower formula is applied for the remaining points, following Algorithm 1. The obtained configuration, overlaid with $\mathbf{Z}_{\text{cmds}}$ is

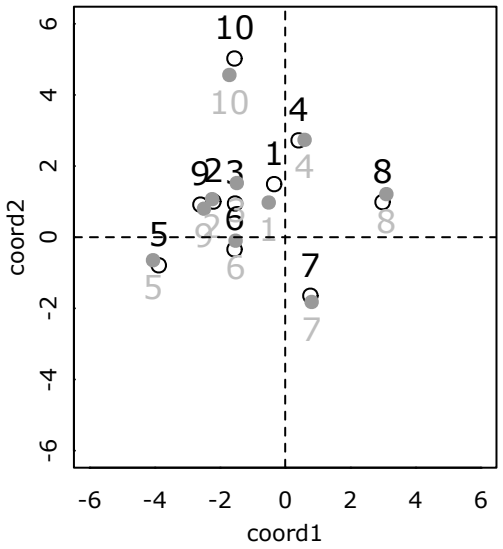


Figura 2.1: INTERP-MDS (no fill) and CMDS (in gray)

with a loss $||\mathbf{Z}\text{interp} - \mathbf{Z}\text{cmds}||_{\text{F}}^2 = 1.140324$.

2.3.2. Minimization of Additional Stress

Consider a group of points with a k-dimensional scaling in a space $\mathcal{Z}$, and new points $\text{O}_{\text{n+1}}, \text{O}_{\text{n+1}}, ..., \text{O}_{\text{n+m}}$ are added with dissimilarities δ_{ij} between them and with the original points. The stress of the original scaling is added to

$$\tau = \tau_1 + \tau_2 = \sum_{\text{i=1}}^{\text{n}} \sum_{\text{j=n+1}}^{\text{n+m}} \left(\hat{\text{f}}(\delta_{\text{ij}}) - \text{d}_{\text{ij}}\right)^2 + \sum_{\text{i=n+1}}^{\text{n+m}} \sum_{\text{j=n+1}}^{\text{n+m}} \left(\hat{\text{f}}(\delta_{\text{ij}}) - \text{d}_{\text{ij}}\right)^2, \tag{2.10}$$

where $\hat{\text{f}}$ is the previously fitted mapping function in the initial scaling, and d_{ij} are the Euclidean distances of the new points in $\mathcal{Z}$. With this, the interpolation problem can be thought of as minimizing τ. Optimization can be done sequentially by adding the new points one at a time, so only its corresponding term in τ_2 will be minimized. In [7], two specific optimization algorithms are used: the PORT routines (Gay, 1990) and the BFGS method (Broyden 1970, Fletcher 1970, Goldfarb 1970, Shanno 1970). In [5], a method based on neural networks is also proposed.

Capítulo 3

Implementation and Simulation

3.1. Introduction

In this chapter, we detail the simulation study conducted using MDS methods for large datasets. The aim is to evaluate processing times for different sample sizes, analyze the quality of solutions, and assess whether the methods are sensitive to the presence of outliers. The following methods are analyzed:

- `proc`, DCO-MDS with Procrustes according to algorithm **??**.
- `qr`, DCO-MDS with QR according to algorithm **??**.
- `gow`, INTERP-MDS with the Gower formula from 2.9.

Performance is compared to classical scaling (CMDS).

3.1.1. Implementation in `R`

Firstly, the fast methods were implemented in `R`. Building upon the code available in [3], adaptation and refinement of the program were performed to include the other methods, among other improvements. The programming work focused on optimizing computation times, yielding results similar to the implementations in the original works. For the hyperparameters of each method, reasonable values were taken according to these works, without an in-depth evaluation of their effect on the final solution. Some results presented in this chapter may vary if these parameters are appropriately optimized. The following parameters were considered:

- The quantity `c` of reference points for the `proc` and `qr` methods. `c` is set to `2k`, where `k` is the dimension of the scaling. This is applied in all cases, aligning with the proposal in [3].
- The size `l` of blocks in the `proc` and `qr` methods. `l` is set to `400`, in line with suggestions in [3].
- The size `m` of the first block of points for interpolation methods. `m` is set to `400`, following the approach in [7].

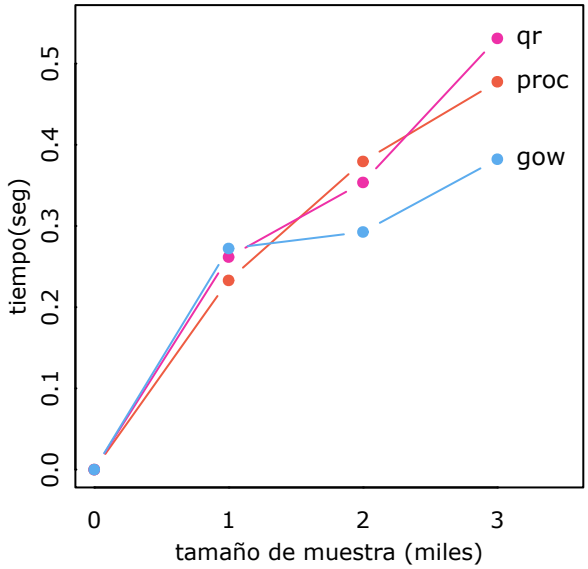
The simulation codes, as well as the examples shown in the previous chapters, are available in the repository `https://github.com/pcosatto/tesis_mds`. A more detailed reference of the repository's file contents is provided in the final section of this chapter.

Among them, the function contained in `cmdscaling.R` is the one that executes fast scaling for

the test simulations. For all simulations, a MacBook Air with an Intel Core i5 processor and 8 GB of RAM, with 4 CPU cores working simultaneously, was used. The parallelization of the scaling of different blocks is supported by the code in `cmdscaling.R`. The user can specify the number of cores to use with the `n_cores` variable, through the `parallel` library. Note that this function cannot be used on Windows.

3.2. Simulation Study

In a preliminary simulation, point clouds with a Normal distribution of dimension `p=2` are generated, and dimension-scaled (`k=2`) based on Euclidean distances. The sample components are independent with mean 0 and variance 5. `Nrep=10` repetitions are performed for each method, and average times are recorded:



	n=1000	n=2000	n=3000
cmds	1.66	12.00	42.67
proc	0.23	0.38	0.48
qr	0.26	0.35	0.53
gow	0.27	0.29	0.38

Tabla 3.1: Average times for `Nrep=10` repetitions.

At first glance, the fast methods show a significant time saving. For 3000 data points, results are obtained in approximately 1 % of the time it would take for classical scaling. For larger samples, this gap could widen even further.

3.2.1. Main Simulation

We aim to assess the ability of fast methods to reproduce the CMDS solution accurately and how they respond to larger samples.

Formulation of the Model

Let $\mathbf{X}$ be the coordinate matrix of a cloud with `n=1000` points in $\mathbb{R}^{\mathtt{d}}$. Suppose that additional coordinates are added to each vector to extend the space to dimension $\mathtt{p} > \mathtt{d}$, resulting in a new vector

$$\mathbf{y} = (x_1, x_2, ..., x_{\mathtt{d}}, \varepsilon_1, \varepsilon_2, ..., \varepsilon_{\mathtt{p-d}})^{\mathrm{t}} = (\boldsymbol{x}\ \mathbf{e})^{\mathrm{t}},$$

where the components of $\boldsymbol{x}$ are the pre-existing coordinates and the components of $\mathbf{e}$ are independent random variables with distribution $\mathcal{N}(0,1)$ and independent of the existing ones. We have

$$\mathbf{Y} = \left[\begin{array}{cc} \mathbf{X} & \mathbf{E} \end{array}\right] \in \mathbb{R}^{\mathtt{n}\times\mathtt{p}} \tag{3.1}$$

as the final coordinate matrix of the $\mathtt{n}$ points, where $\mathbf{E}$ is the block containing the new dimensions. Suppose we want to perform a multidimensional scaling of objects whose *true* coordinates are in $\mathbf{X}$, but starting from the measurement of $\mathbf{D_Y}$, the Euclidean distance matrix of $\mathbf{Y}$, as the initial dissimilarity. For these distances, we will have a positive bias since, by adding dimensions, the points extend into a larger space. Let $\mathrm{d}_{\mathrm{ij}}^2$ be an element of $\mathbf{D}_{\mathbf{Y}}^{(2)}$, under 3.1 we have

$$\mathrm{d}_{\mathrm{ij}}^2 = \sum_{\mathrm{a}=1}^{\mathtt{p}} (\mathrm{Y}_{\mathrm{ai}} - \mathrm{Y}_{\mathrm{aj}})^2 = \sum_{\mathrm{a}=1}^{\mathtt{d}} (x_{\mathrm{ai}} - x_{\mathrm{aj}})^2 + \sum_{\mathrm{a}=\mathtt{d}+1}^{\mathtt{p}} (\varepsilon_{\mathrm{ai}} - \varepsilon_{\mathrm{aj}})^2, \tag{3.2}$$

so that

$$\mathbf{D}_{\mathbf{Y}}^{(2)} = \mathbf{D}_{\mathbf{X}}^{(2)} + \mathbf{D}_{\mathbf{E}}^{(2)}. \tag{3.3}$$

Let's see what happens with the elements of the matrix $\mathbf{B_Y} = \mathbf{Y}\mathbf{Y}^{\mathrm{t}}$, which are the squared norms $\mathrm{b}_{\mathrm{ii}} = \|\mathbf{y}_{\mathrm{i}}\|^2$ on the main diagonal, and the inner products $\mathrm{b}_{\mathrm{ij}} = \mathbf{y}_{\mathrm{i}}^{\mathrm{t}}\mathbf{y}_{\mathrm{j}}$ off the diagonal. In any case, we have

$$\mathrm{b}_{\mathrm{ij}} = \mathbf{y}_{\mathrm{i}}^{\mathrm{t}}\mathbf{y}_{\mathrm{j}} = \sum_{\mathrm{a}=1}^{\mathtt{p}} \mathrm{Y}_{\mathrm{ai}}\mathrm{Y}_{\mathrm{aj}} = \sum_{\mathrm{a}=1}^{\mathtt{d}} x_{\mathrm{ai}}x_{\mathrm{aj}} + \sum_{\mathrm{a}=\mathtt{d}+1}^{\mathtt{p}} \varepsilon_{\mathrm{ai}}\varepsilon_{\mathrm{aj}} = \boldsymbol{x}_{\mathrm{i}}^{\mathrm{t}}\boldsymbol{x}_{\mathrm{j}} + \mathbf{e}_{\mathrm{i}}^{\mathrm{t}}\mathbf{e}_{\mathrm{j}}, \tag{3.4}$$

which, in matrix form, is expressed as

$$\mathbf{B_Y} = \mathbf{Y}\mathbf{Y}^{\mathrm{t}} = \mathbf{X}\mathbf{X}^{\mathrm{t}} + \mathbf{E}\mathbf{E}^{\mathrm{t}} = \mathbf{B_X} + \mathbf{B_E}. \tag{3.5}$$

The component $\mathbf{B_X}$ is fixed, and the component $\mathbf{B_E}$ is random. For the element $\{\mathbf{B_E}\}_{\mathrm{ij}}$, we have

$$\mathsf{E}(\{\mathbf{B_E}\}_{\mathrm{ij}}) = \mathsf{E}(\mathbf{e}_{\mathrm{i}}^{\mathrm{t}}\mathbf{e}_{\mathrm{j}}) = \mathrm{tr}\big(\mathsf{E}(\mathbf{e}_{\mathrm{i}}^{\mathrm{t}}\mathbf{e}_{\mathrm{j}})\big) = \mathsf{E}\big(\mathrm{tr}(\mathbf{e}_{\mathrm{i}}^{\mathrm{t}}\mathbf{e}_{\mathrm{j}})\big) = \mathsf{E}\big(\mathrm{tr}(\mathbf{e}_{\mathrm{i}}\mathbf{e}_{\mathrm{j}}^{\mathrm{t}})\big) = \mathrm{tr}\big(\mathsf{E}(\mathbf{e}_{\mathrm{i}}\mathbf{e}_{\mathrm{j}}^{\mathrm{t}})\big). \tag{3.6}$$

It turns out that $\mathrm{tr}\big(\mathsf{E}(\mathbf{e}_{\mathrm{s}}\mathbf{e}_{\mathrm{r}}^{\mathrm{t}})\big) = 0$ for $\mathrm{s} \neq \mathrm{r}$ since the vectors are independent, and $\mathrm{tr}\big(\mathsf{E}(\mathbf{e}_{\mathrm{s}}\mathbf{e}_{\mathrm{r}}^{\mathrm{t}})\big) = \mathrm{tr}\big(\Sigma_{\mathbf{e}}\big) = \mathtt{p-d}$ for $\mathrm{s} = \mathrm{r}$ since the new components have unit variance.

It can also be shown that $\mathsf{Var}(\mathbf{e}_i^t\mathbf{e}_j)$ is `p-d` for i $\neq$ j and 2(`p-d`) for i = j. All of this suggests that the formulation of 3.1 serves as a model of measurement errors for the inner products between pairs of objects in $\mathbf{X}$, as they are subject to a positive or negative random distortion, while the squared norms increase as dimensions are added. This makes it particularly useful for studying the response of CMDS-based methods since the fit occurs by minimizing the loss in inner products, not distances.

In this work, we fix `p=10` and `d=5`, and generate the matrix $\mathbf{X}$ with a Normal distribution with mean 0 and variance diag$(5, 5, ..., 5)$, with a fixed seed in `R`. As an

example, for `n=1000` vectors:

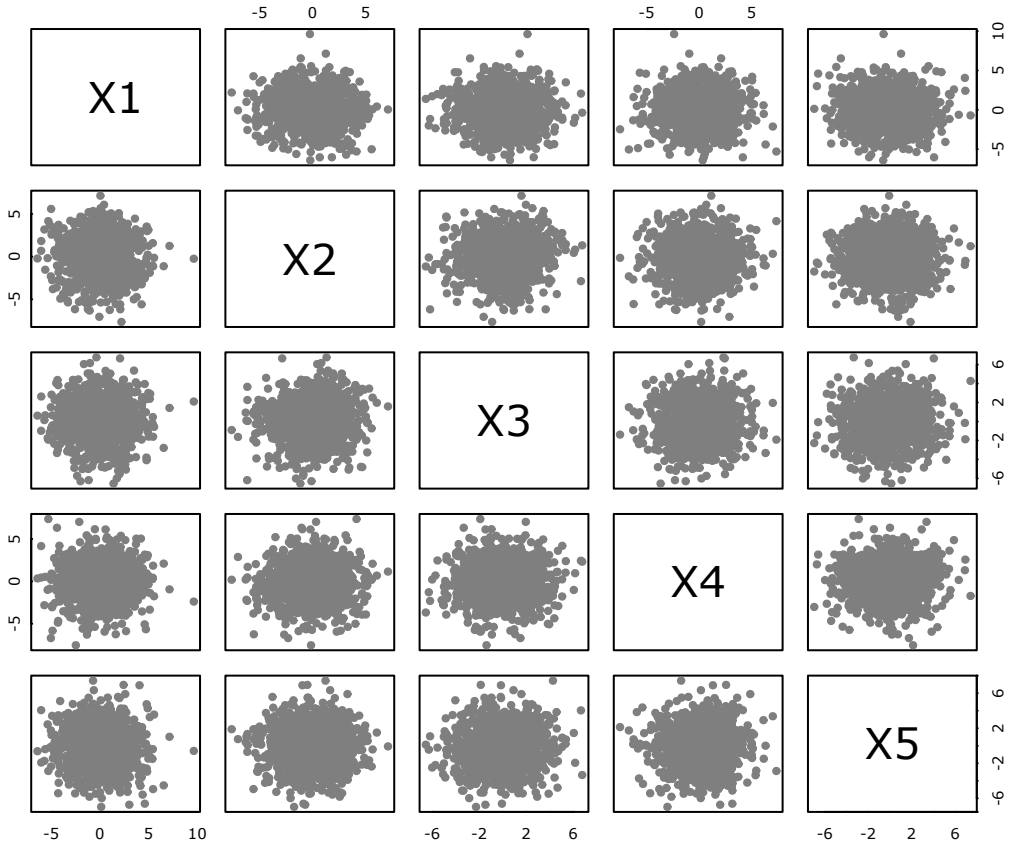


Figura 3.1: Canonical views of a cloud of `n=1000` points without distortion.

For these points, we have a *total variability* given by $\text{tr}(\mathbf{X}^t\mathbf{X}) = 24841.93$, while for $\text{tr}(\mathbf{E}^t\mathbf{E})$ the expected value is

$$\mathsf{E}(\text{tr}(\mathbf{E}^t\mathbf{E})) = \text{tr}(\mathsf{E}(\mathbf{E}\mathbf{E}^t)) = \text{tr}(\text{diag}(5, 5, ..., 5)) = 5000, \tag{3.7}$$

which is approximately one-fifth.

Identification of the True Dimension

A first set of simulations is conducted with a sample size (`n`) equal to 1000, as shown in Figure 4.1, including errors. In each simulation, a multidimensional scaling ($\mathbf{Z}$) is obtained, starting from the information of $\mathbf{D_Y}$, and the following metrics are calculated:

$$\sigma_{B_Z} = \frac{||\mathbf{B_X} - \mathbf{Z}\mathbf{Z}^T||_F}{||\mathbf{B_X}||_F}, \tag{3.8}$$

$$\sigma_{D_Z} = \frac{||\mathbf{D_X} - \mathbf{D_Z}||_F}{||\mathbf{D_Z}||_F}, \tag{3.9}$$

$$\sigma_Z = \frac{||\mathbf{Z}_{cmds} - \mathbf{Z}||_F}{||\mathbf{Z}_{cmds}||_F}. \tag{3.10}$$

A total of `Nrep=100` repetitions are performed for each configuration, where $\mathbf{Z}_{cmds}$ is the matrix obtained with CMDS. The first coefficient is the error in the Strain loss, the second is the Stress-1 of Kruskal, and the third is an error coefficient of the direct fit between point configurations.

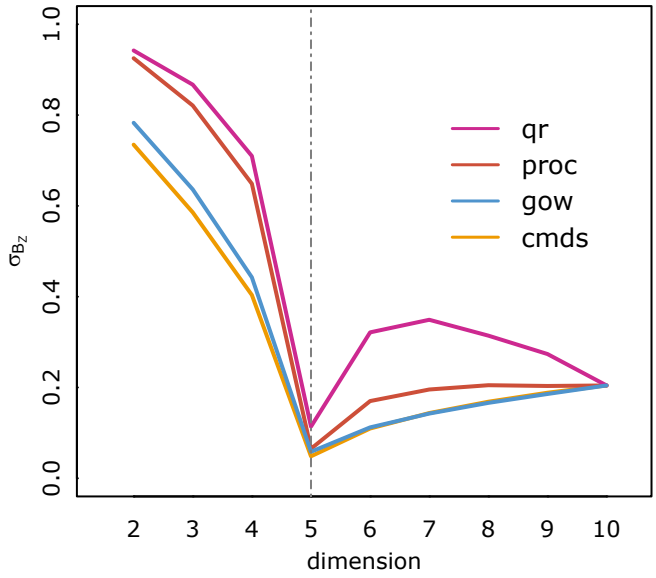


Figura 3.2: Averages of the Strain loss coefficient for scalings of different dimensions.

	σ_{B_Z}	σ_{D_Z}	σ_Z
`cmds`	0.04845001	0.00789342	0
`proc`	0.06503457	0.01512929	0.03017452
`qr`	0.11382362	0.02963782	0.07386895
`gow`	0.05843108	0.01279837	0.02260295

Tabla 3.2: Averages of the coefficients for `k=5`, the true dimension of $\mathbf{X}$.

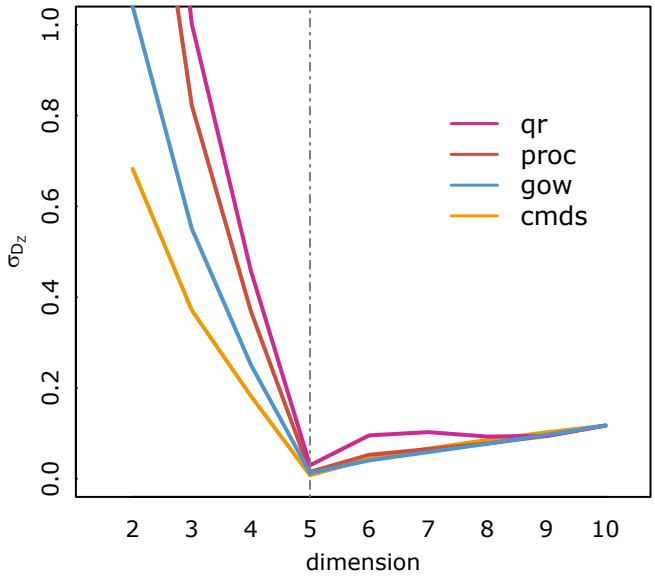


Figura 3.3: Averages of Stress-1 for scalings of different dimensions.

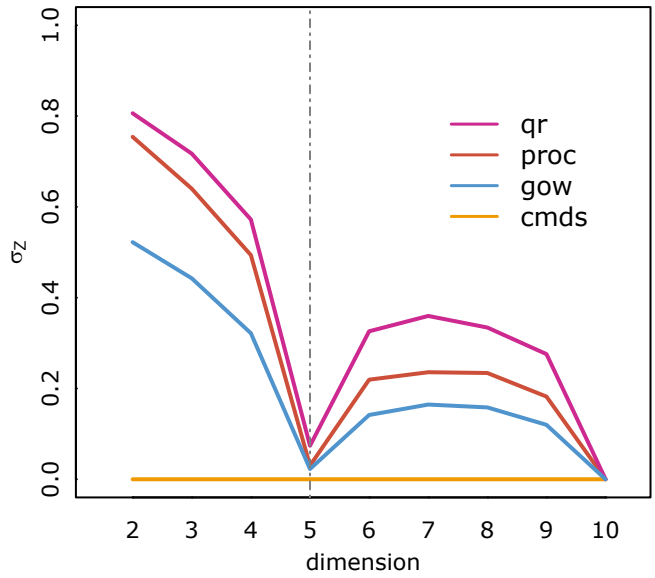


Figura 3.4: Averages of the coefficient $\sigma_{\mathbf{Z}}$ for scalings of different dimensions.

These plots show that the method that best reproduces the CMDS solution is the interpolation method with the Gower formula. Alignment-based methods have lower performance, and among them, the QR decomposition-based method seems to have a worse fit.

Larger Sample Sizes

Secondly, we aim to study the performance of fast methods for larger sample sizes. The general scheme is as follows:

- Increasing sample sizes (`n`): 10000, 50000, and 100000.
- Number of repetitions (`Nrep`): 100.
- Scenario 1: The distribution of $\mathbf{y}$ remains the same.
- Scenario 2: 90 % of observations follow the Scenario 1 scheme, but 10 % are simulated with variance 25 for the first 2 error components, i.e., variables ε_1 and ε_2. This introduces outliers.

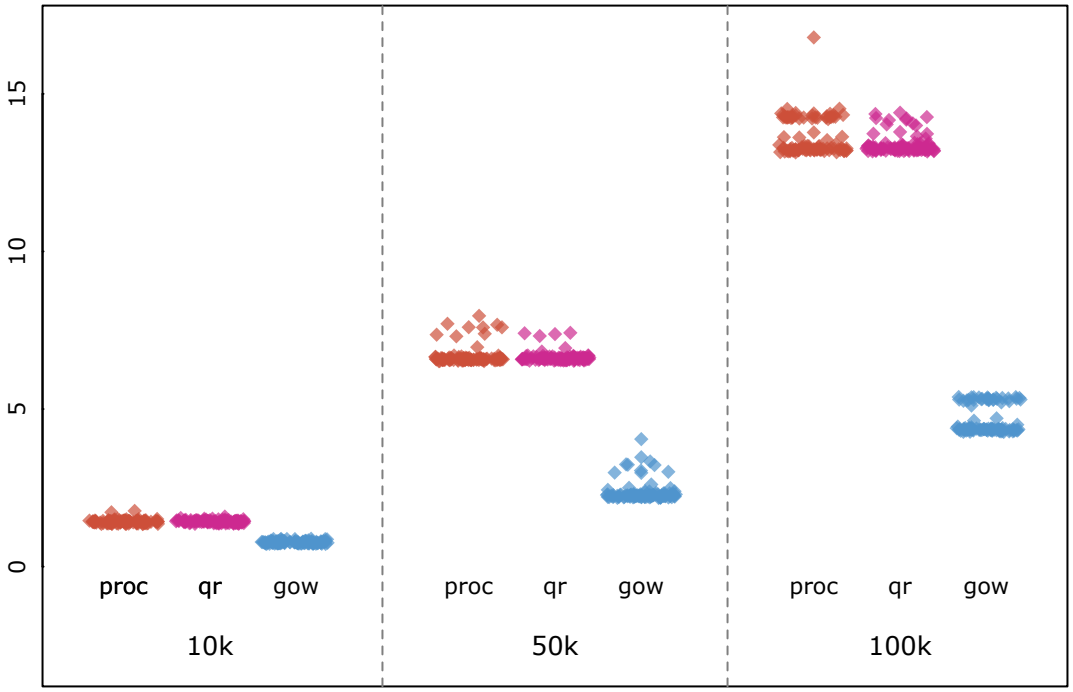


Figura 3.5: Processing times (in seconds) for larger sample sizes.

For very large sample sizes, as in this case, it is not feasible to perform the complete classical scaling or obtain all the eigenvalues of $\mathbf{B}_{\mathrm{X}}$. Therefore, we work with `k=5` eigenvalues of the covariance $\hat{\mathbf{\Sigma}}_{\mathrm{Z}} = \frac{1}{\mathrm{n}}\mathbf{Z}^{\mathrm{T}}\mathbf{Z}$ of each specific scaling, denoted as $\hat{\lambda}_{\mathrm{a}}$. If the representation is good, these eigenvalues should not deviate significantly from 5, which are the variances of the *true* coordinates of the points. An initial inspection of the results involves calculating the average of the eigenvalues for each solution:

$$\frac{1}{5}\sum_{\mathrm{a}=1}^{\mathtt{d}}\hat{\lambda}_{\mathrm{at}}, \quad \text{with} \quad \mathrm{t} = \{1, 2, ..., \mathtt{Nrep}\}. \tag{3.11}$$

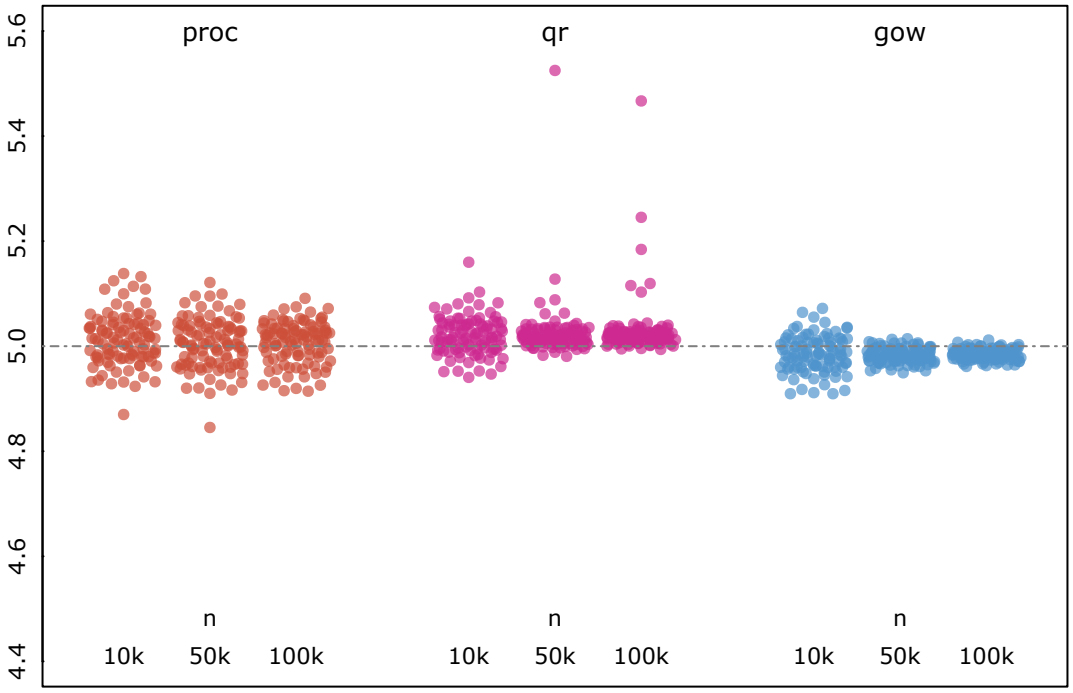


Figura 3.6: Average eigenvalues in Scenario 1 (no outliers).

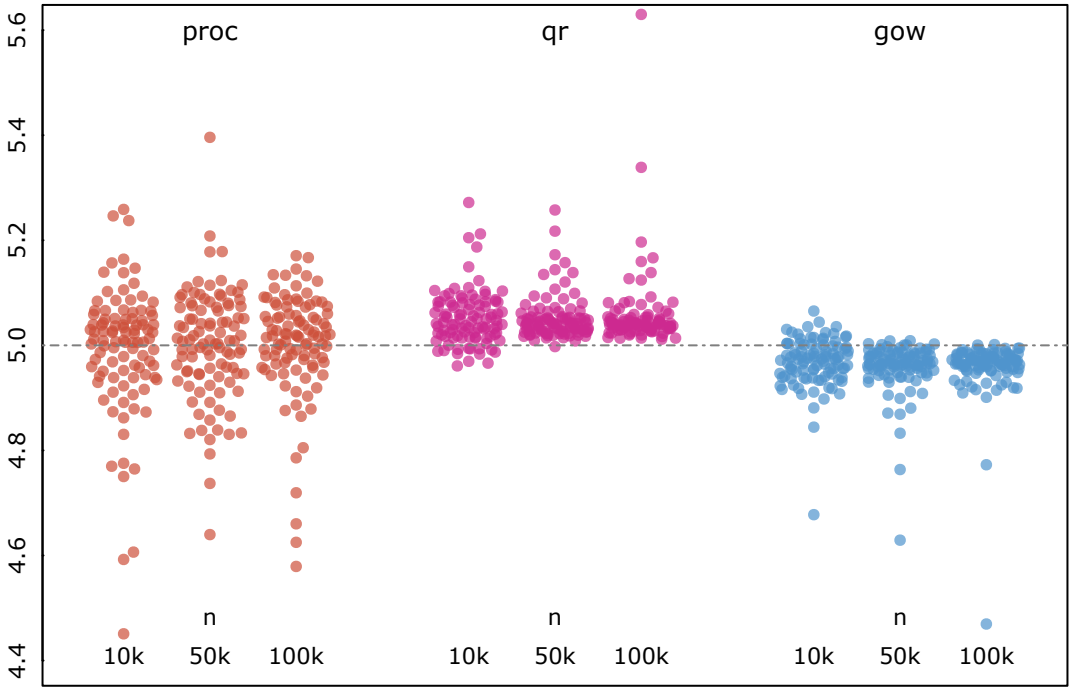


Figura 3.7: Average eigenvalues in Scenario 2 (with outliers).

These graphs show the degree of dispersion for each method and the bias in relation to the true eigenvalues. The `qr` method produced some solutions, in both scenarios, that are outside the scale of the graph.

Let $\boldsymbol{\lambda} = (5, 5, 5, 5, 5)^{\mathrm{T}}$ be the vector of population eigenvalues and $\hat{\boldsymbol{\lambda}}$ be the vector of sample

eigenvalues for any given solution. We have:

$$\mathsf{B}(\hat{\boldsymbol{\lambda}}) = \mathsf{E}\big(\hat{\boldsymbol{\lambda}} - \boldsymbol{\lambda}\big) \quad \text{and} \quad \mathsf{Cov}(\hat{\boldsymbol{\lambda}}) = \mathsf{E}$$

$$\Big\{ \big(\ \text{-} \ \mathrm{E}() \ \big) \ \big(\ \text{-} \ \mathrm{E}() \ \big)^{\mathrm{T}} \Big\}, \quad (3.12)$$

representing the bias and the covariance matrix, respectively. We can calculate a scalar version of the mean squared error as:

$$\begin{aligned}
\mathsf{MSE}(\hat{\boldsymbol{\lambda}}) &= \mathsf{E}\Big\{ \big(\hat{\boldsymbol{\lambda}} - \boldsymbol{\lambda}\big)^{\mathrm{T}} \big(\hat{\boldsymbol{\lambda}} - \boldsymbol{\lambda}\big) \Big\} \\
&= \mathrm{tr}\ \mathsf{E}\Big\{ \big(\hat{\boldsymbol{\lambda}} - \boldsymbol{\lambda}\big) \big(\hat{\boldsymbol{\lambda}} - \boldsymbol{\lambda}\big)^{\mathrm{T}} \Big\} \\
&= \mathrm{tr}\ \mathsf{E}\Big\{ \big(\hat{\boldsymbol{\lambda}} - \mathsf{E}(\hat{\boldsymbol{\lambda}}) + \mathsf{B}(\hat{\boldsymbol{\lambda}})\big) \big(\hat{\boldsymbol{\lambda}} - \mathsf{E}(\hat{\boldsymbol{\lambda}}) + \mathsf{B}(\hat{\boldsymbol{\lambda}})\big)^{\mathrm{T}} \Big\} \\
&= \mathrm{tr}\ \Big\{ \mathsf{Cov}(\hat{\boldsymbol{\lambda}}) + \mathsf{B}(\hat{\boldsymbol{\lambda}})^{\mathrm{T}} \mathsf{B}(\hat{\boldsymbol{\lambda}}) + \mathsf{B}(\hat{\boldsymbol{\lambda}}) \mathsf{E}\{\hat{\boldsymbol{\lambda}} - \mathsf{E}(\hat{\boldsymbol{\lambda}})\} + \mathsf{E}\{\hat{\boldsymbol{\lambda}} - \mathsf{E}(\hat{\boldsymbol{\lambda}})\} \mathsf{B}(\hat{\boldsymbol{\lambda}}) \Big\} \\
&= \mathrm{tr}\ \mathsf{Cov}(\hat{\boldsymbol{\lambda}}) + \mathsf{B}(\hat{\boldsymbol{\lambda}})^{\mathrm{T}} \mathsf{B}(\hat{\boldsymbol{\lambda}})
\end{aligned} \quad (3.13)$$

This expression separates the bias from the net variability of each method. Let $\hat{\boldsymbol{\lambda}}$ be the vector of eigenvalues of the t-th scaling of the simulation, where $\mathrm{t} = 1, 2, ...,$ `Nrep`. We estimate the mean squared error as:

$$\hat{\mathsf{MSE}}(\hat{\boldsymbol{\lambda}}_{\mathrm{t}}) = \mathrm{tr} \left(\frac{1}{\texttt{Nrep} - 1} \sum_{\mathrm{t}=1}^{\texttt{Nrep}} \big(\hat{\boldsymbol{\lambda}}_{\mathrm{t}} - \overline{\boldsymbol{\lambda}}\big) \big(\hat{\boldsymbol{\lambda}}_{\mathrm{t}} - \overline{\boldsymbol{\lambda}}\big)^{\mathrm{T}} \right) + \|\overline{\boldsymbol{\lambda}} - \boldsymbol{\lambda}\|. \quad (3.14)$$

Where $\overline{\boldsymbol{\lambda}} = \frac{1}{\texttt{Nrep}} \sum_{\mathrm{t}=1}^{\texttt{Nrep}} \hat{\boldsymbol{\lambda}}_{\mathrm{t}}$ is the sample mean of the eigenvalue vectors.

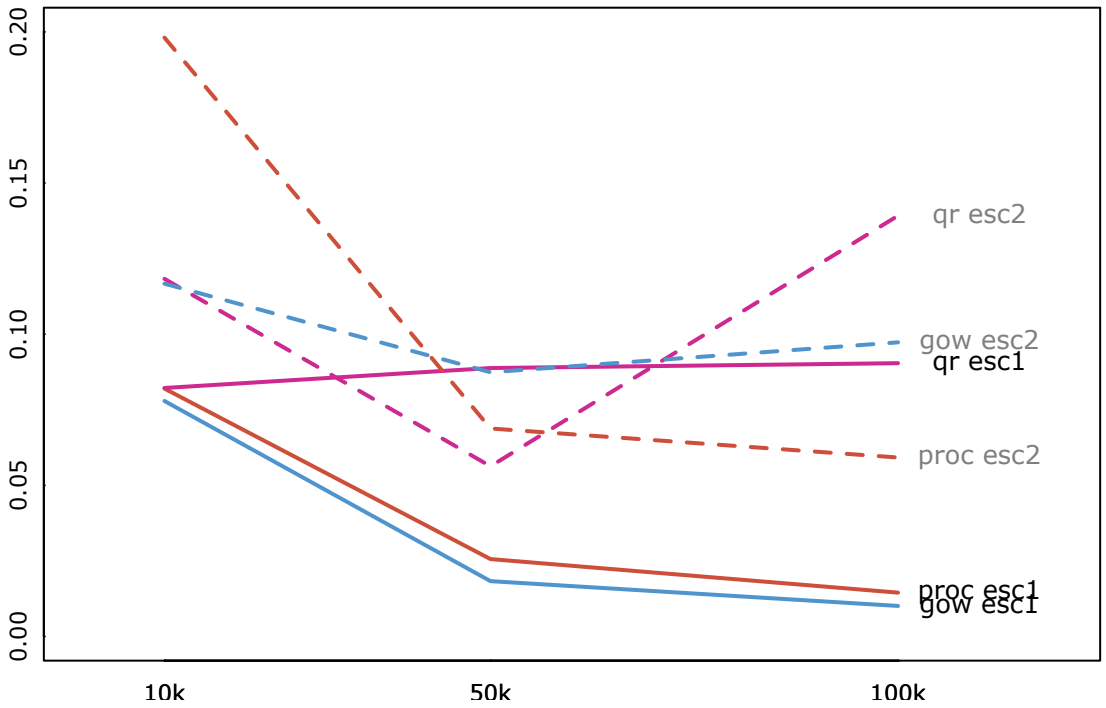


Figura 3.8: Mean squared error for different sample sizes in Scenarios 1 and 2.

		$\hat{\mathrm{MSE}}(\hat{\boldsymbol{\lambda}})$	tr $\mathrm{Cov}(\hat{\boldsymbol{\lambda}})$	$\mathrm{B}(\hat{\boldsymbol{\lambda}})^{\mathrm{T}}\mathrm{B}(\hat{\boldsymbol{\lambda}})$
proc	n=10k	0.0820	0.0170	0.0650
	n=50k	0.0256	0.0123	0.0133
	n=100k	0.0145	0.0081	0.0064
qr	n=10k	0.0822	0.0131	0.0691
	n=50k	0.0888	0.0661	0.0227
	n=100k	0.0904	0.0714	0.0190
gow	n=10k	0.0779	0.0109	0.0670
	n=50k	0.0183	0.0021	0.0162
	n=100k	0.0101	0.0013	0.0088

Tabla 3.3: Details of Scenario 1 (no outliers).

		$\hat{\mathrm{MSE}}(\hat{\boldsymbol{\lambda}})$	tr $\mathrm{Cov}(\hat{\boldsymbol{\lambda}})$	$\mathrm{B}(\hat{\boldsymbol{\lambda}})^{\mathrm{T}}\mathrm{B}(\hat{\boldsymbol{\lambda}})$
proc	n=10k	0.1981	0.1346	0.0635
	n=50k	0.0688	0.0559	0.0129
	n=100k	0.0592	0.053	0.0062
qr	n=10k	0.1183	0.0307	0.0876
	n=50k	0.0563	0.0226	0.0337
	n=100k	0.1392	0.1055	0.0337
gow	n=10k	0.1167	0.0301	0.0866
	n=50k	0.0874	0.0465	0.0409
	n=100k	0.0973	0.066	0.0313

Tabla 3.4: Details of Scenario 2 (with outliers).

Some General Observations

The results presented are only indicative, as the variability is influenced in all cases by numerical errors in the algorithm and the choice of hyperparameters. These issues deserve a more detailed analysis.

- The Gower interpolation method is faster than the others and appears to produce results most similar to CMDS.
- The alignment method with QR decomposition produces a poorer fit due to the lack of a scale-changing transformation, which the Procrustes method does have.
- The Procrustes method has more variance than the others but, at the same time, less bias.
- The Procrustes method seems to be the most sensitive to outliers. The other two methods show biases towards one side.
- There is an outlier solution far from the others for the QR method, not visible in Figures 3.6 and 3.7.
- In the time graph, as the sample size increases, the presence of two groups of solutions becomes more pronounced: some faster and others slower.
- For the Gower method, projecting the new coordinates into the space of the original coordinates is pending, which could potentially improve the method.

3.2.2. Traceability of Solutions

Each point cloud is simulated with seed `n+i-1`, where `n` is the sample size and `i` is the repetition or cycle number. So, `set.seed(10001)` is used in the second repetition of the sample of size 10000. Within each method, randomization is used to choose points: In the `proc` and `qr` methods, observations are randomly permuted with a seed, and the reference points will be the first `c`. In the `gow` method, a random block of points of size `m` must be chosen with a seed. To the seed of the original dataset, $w10^6$ is added, where `w` is the method number: `w=1` for `proc`, `w=2` for `qr`, and `w=3` for `interp`. For example, `set.seed(2010002)` is the seed used to choose the partitions for the `qr` method in the second run with sample size 10000.

3.2.3. Repository

We list the files contained in the repository:

- `aux_functions.R` - Auxiliary functions used throughout the work.
- `Examples Chap1.R` - Development of examples and cases from Chapter 1.
- `Examples Chap2.R` - Development of examples and cases from Chapter 2.
- `cmdscaling.R` - Main MDS function used in the simulation.
- `Simulation Main.R` - Code used for the simulation.
- `Simulation Main Results.RData` - Results of the simulation.
- `Simulation Analysis.R` - Calculations and graphics for Chapter 3.
- `Chap4.R` - Development of the case in Chapter 4 (upcoming).
- `Chap4 Data.RData` - Results of Chapter 4 (upcoming).

Capítulo 4

Application Case

4.1. Introduction

In this chapter, we work with an application case of the methods seen for MDS (Multidimensional Scaling) with large datasets. Through a specific problem, we apply MDS as a dimensionality reduction technique prior to data clustering. To achieve this, we use a custom implementation function in `R`. The goal is to demonstrate the application of MDS as a way to overcome the çurse of dimensionalityïn a clustering problem with a large dataset. In cases like this, conventional MDS methods would be impractical, and alternative techniques such as Principal Component Analysis or t-SNE might be necessary.

4.1.1. Installation of the `mvtools` Library

In the `mvtools` library, developed in-house, the `procrustes_mdscal` function is available and will be used in this chapter. The code can be found at `https://github.com/pcosatto/mvtools/blob/main/R/procrustes_mdscal.R`. To install the library in R-Studio, execute the following command in the console, assuming you have already installed the `devtools` package:

```
devtools::install_github("pcosatto/mvtools")
```

Then, load the library as usual with `library(mvtools)`. The help page for the `procrustes_mdscal` function can be opened with `?procrustes_mdscal`. This function implements DCO-MDS with the Procrustes method, using distance scaling. Stress adjustment is performed with the SMACOF algorithm, and splines are used as the mapping of dissimilarities.

To obtain dissimilarities between observations, Minkowski metrics or the Gower metric for mixed data types can be used. The use of these metrics may result in a slower implementation compared to the `cmdscaling_test` function used in the previous chapter. The latter was specifically designed for simulation studies and lacks the necessary functionality for practical use.

4.2. Application Case

Data for `n=8621` bank customers were collected, recording the following variables regarding their credit card usage. The objective is to find specific segments among customers through cluster analysis. The original data can be found at [https://www.kaggle.com/datasets/arjunbhasin2013/ccdata](https://www.k
The following variables will be considered:

- `oneoff_purchases` - Maximum amount of one-off purchases.
- `install_purchases` - Amount of purchases made in installments.
- `cash_adv` - Cash advances granted to the customer.
- `oneoff_purchases_freq` - Frequency index of one-off purchases (1 = frequent, 0 = not frequent).
- `install_purchases_freq` - Frequency index of purchases in installments (1 = frequent, 0 = not frequent).
- `cash_adv_freq` - Frequency index of cash advances (1 = frequent, 0 = not frequent).
- `purchases_trx` - Number of transactions in the last period.
- `credit_limit` - Credit card limit for the customer.
- `prc_full_payment` - Proportion of times the customer makes full payment of the balance.

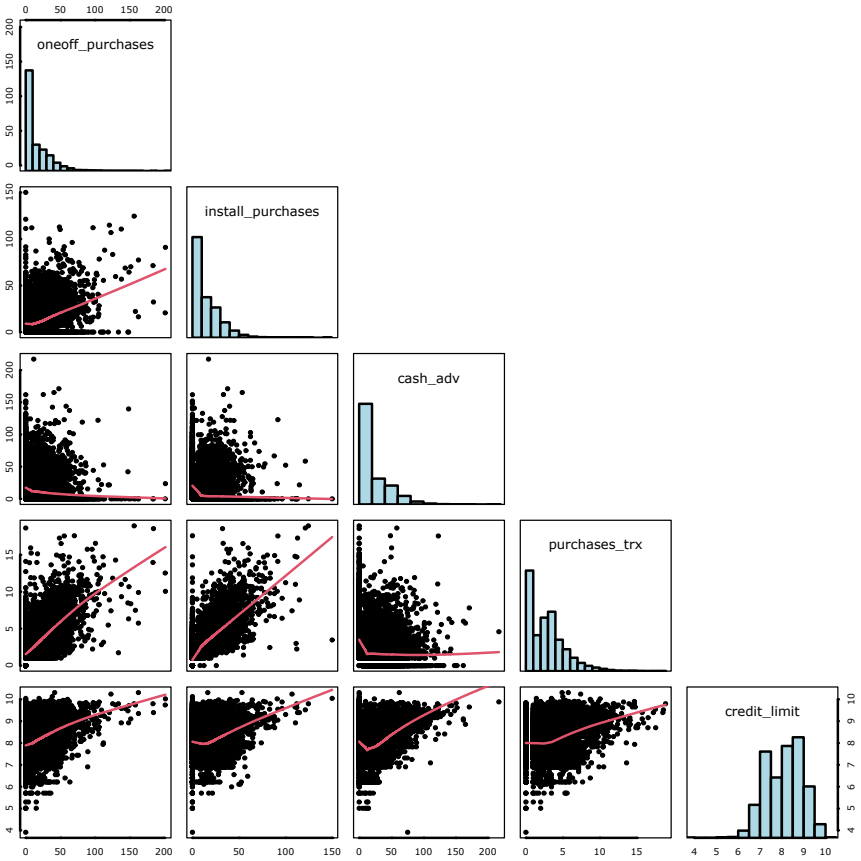


Figura 4.1: Pair plots of four variables with transformed data

As dissimilarity for performing MDS in this problem, we use the Gower distance for quantitative variables. This distance takes values close to 1 for maximum dissimilarity and 0 for maximum similarity. Given two data vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ composed of p quantitative variables, the Gower distance between them is calculated as:

$$d_{ij} = 1 - \left\{ \frac{1}{p} \sum_{k=1}^{p} \left(1 - \frac{|X_{1k} - X_{2k}|}{R_k} \right) \right\}, \tag{4.1}$$

where R_k is the range of the k-th variable. As the distance directly depends on the range, the absence of significant outliers in the sample was previously verified. To work on smaller scales, the square root was taken for `oneoff_purchases`, `install_purchases`, `purchases_trx`, and `cash_adv`; and the logarithm for the variable `credit_limit`, as shown in the graph 4.1. All these variables exhibited a highly significant positive skewness in the original sample, which can significantly affect the metric.

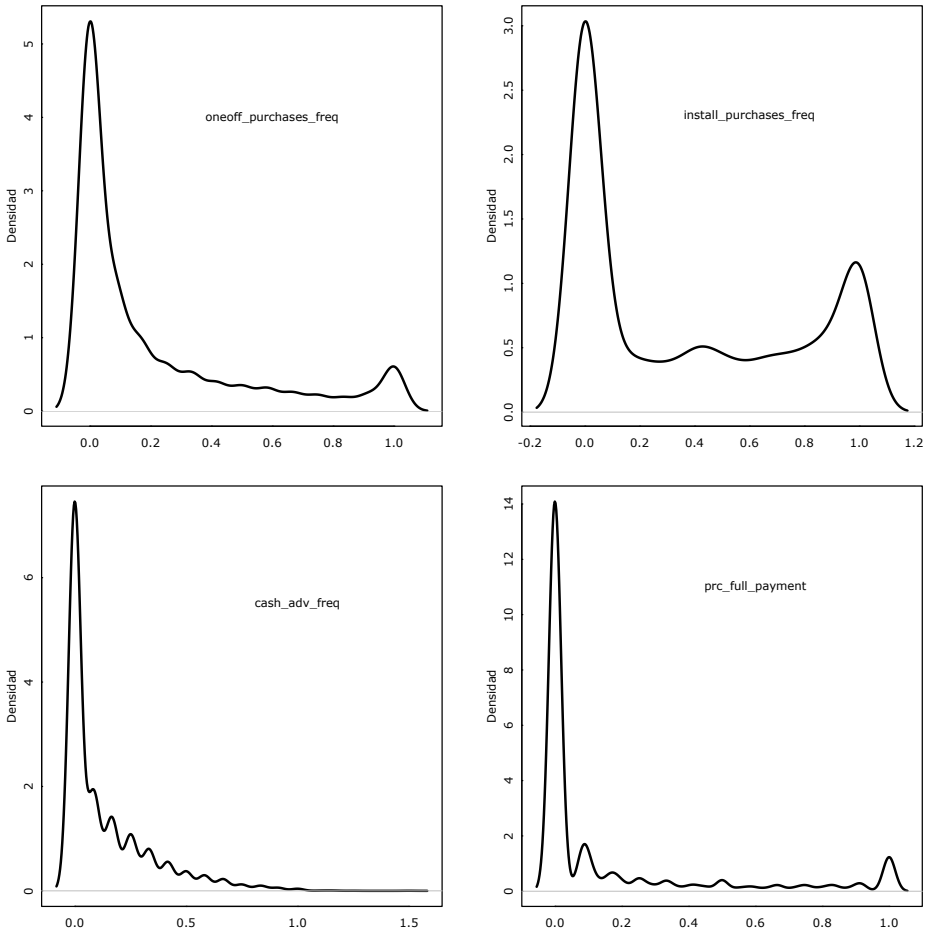


Figura 4.2: Density plots of the remaining variables, without transformation.

! latex Copy code

4.2.1. Dimensionality Reduction

A first approach to the problem is to reduce the dimension to a two-dimensional space using the `procrustes_mdscal` function. The obtained cloud is shown below:

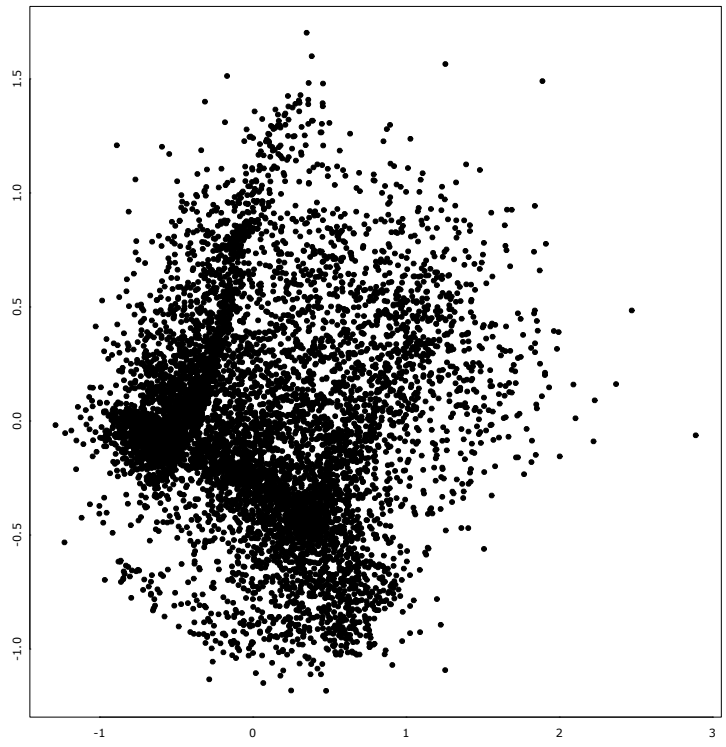


Figura 4.3: Two-dimensional distance scaling (points in black)

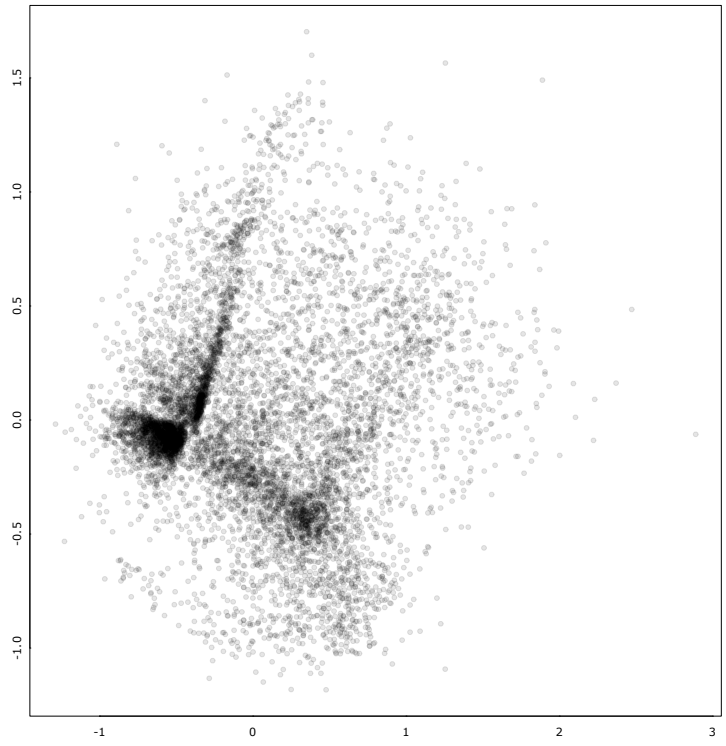


Figura 4.4: Two-dimensional distance scaling (points with transparency)

Due to the dense cloud of points, it is challenging to analyze the result at a glance. Situations like this are expected in reductions with a large amount of data. In the 4.4 graph, transparency is added to the points, allowing identification of some areas of higher concentration where clusters might be found. Also, some isolated points with very high density compared to the rest are observed.

4.2.2. Group Identification

We apply a density-based clustering method called DBSCAN (*Density-based spatial clustering of applications with noise*), using the `dbscan` library. This method is suitable for clustering in point clouds with high density as it detects compact areas and separates observations as *noise*. It not only does not require the prior specification of the number of groups but also does not form groups with all observations.

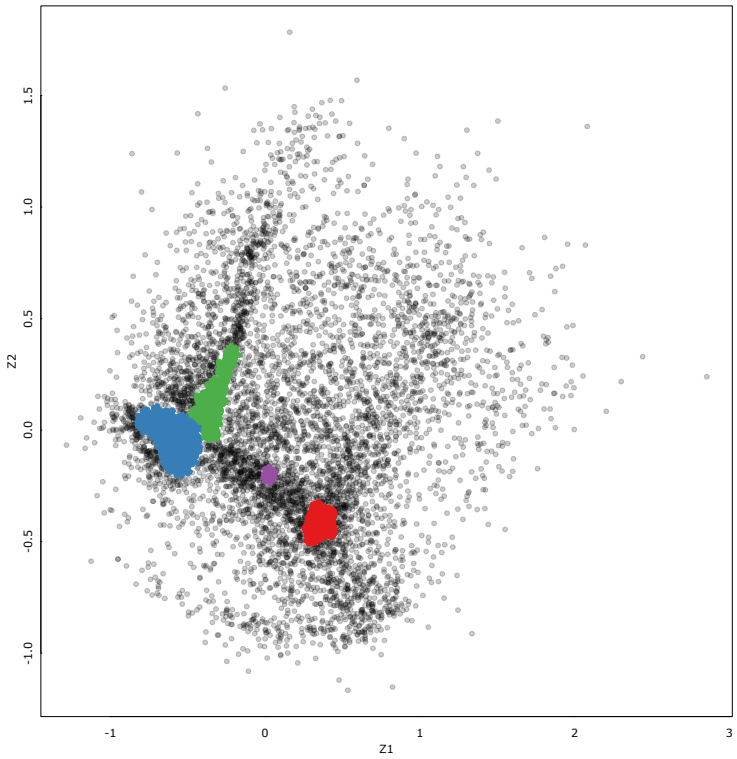


Figura 4.5: Clustering with DBSCAN, using data scaled to $\mathbb{R}^2$

The graph shows four distinct groups that can be found in the two-dimensional scaling. As the DBSCAN method has two hyperparameters, it was necessary to determine them with a grid search. The configuration of at least 4 groups that maximizes the index

$$\frac{\sum_{k=1}^{K} n_k ||\bar{\mathbf{z}}_k - \bar{\mathbf{z}}||^2}{\sum_{k=1}^{K}\sum_{i=1}^{n} ||\mathbf{z}_{ik} - \bar{\mathbf{z}}_k||^2} \frac{n-k}{K-1}, \tag{4.2}$$

where $k \in \{1, 2, ..., K\}$ is the group, n_k is the number of observations per group, $\bar{\mathbf{z}}$ is the mean of all observations in the reduced space, and $\bar{\mathbf{z}}_k$ is the mean of the observations of the k-th group. This index, known as the Calinski-Harabasz metric, increases when there are compact and well-separated

groups. The advantage for this application is that it does not require the calculation of all pairs of distances between points but only distances to centroids. In Figure 4.5, an optimal grouping with the restriction K $\geq$ 4 is shown. The `optimize_dbscan` function used for this search is available in the `aux_functions.R` in the repository, and the values used to set up the grid are in `Cap4.R`.

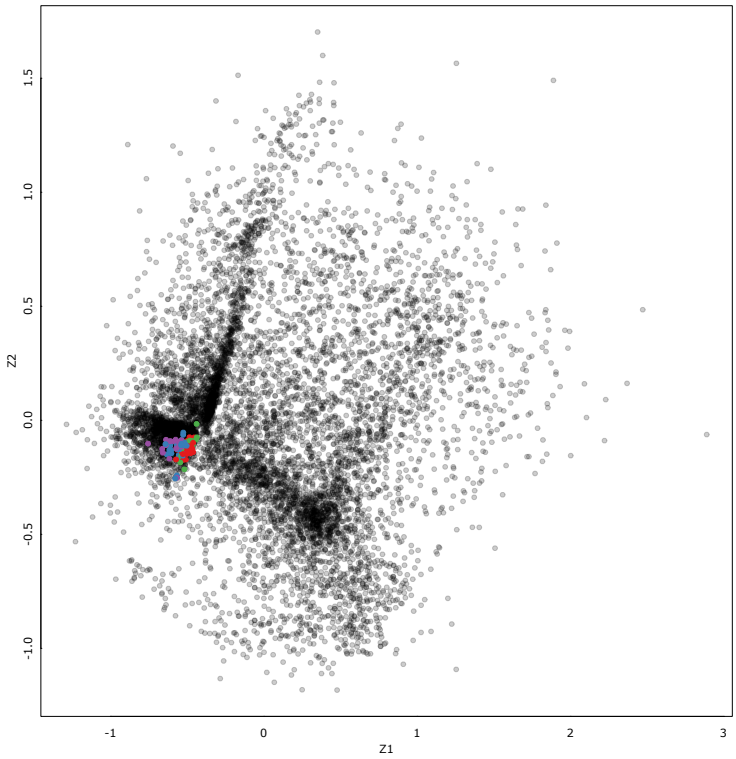


Figura 4.6: Clustering with DBSCAN, using original data

The same procedure was carried out for the original data without reducing the dimension. In Figure 4.6, the points are represented with reduced coordinates but colored according to the grouping done considering all dimensions, for an optimal combination of hyperparameters.

	Noise	**1**	**2**	**3**	**4**
Noise	6252	1	0	0	0
1	1540	11	16	12	12
2	202	0	0	0	0
3	528	0	0	0	0
4	50	0	0	0	0

Tabla 4.1: Groups with MDS (rows) versus groups with original dimensions (columns)

The poor quality of the grouping may be due to the high dimensionality of the data, making it challenging to achieve areas of sufficient density. This is known as the çurse of dimensionality."To overcome this problem, dimensionality reduction methods are usually used beforehand. Multidimensional scaling would have been excluded from the options for this problem due to the large number of observations.

A Three-dimensional Representation

We can extend the representation to three dimensions, increasing the space of scaling. The figure shows canonical views of the point cloud, including the groupings made with DBSCAN using the same hyperparameter optimization procedure as for two dimensions.

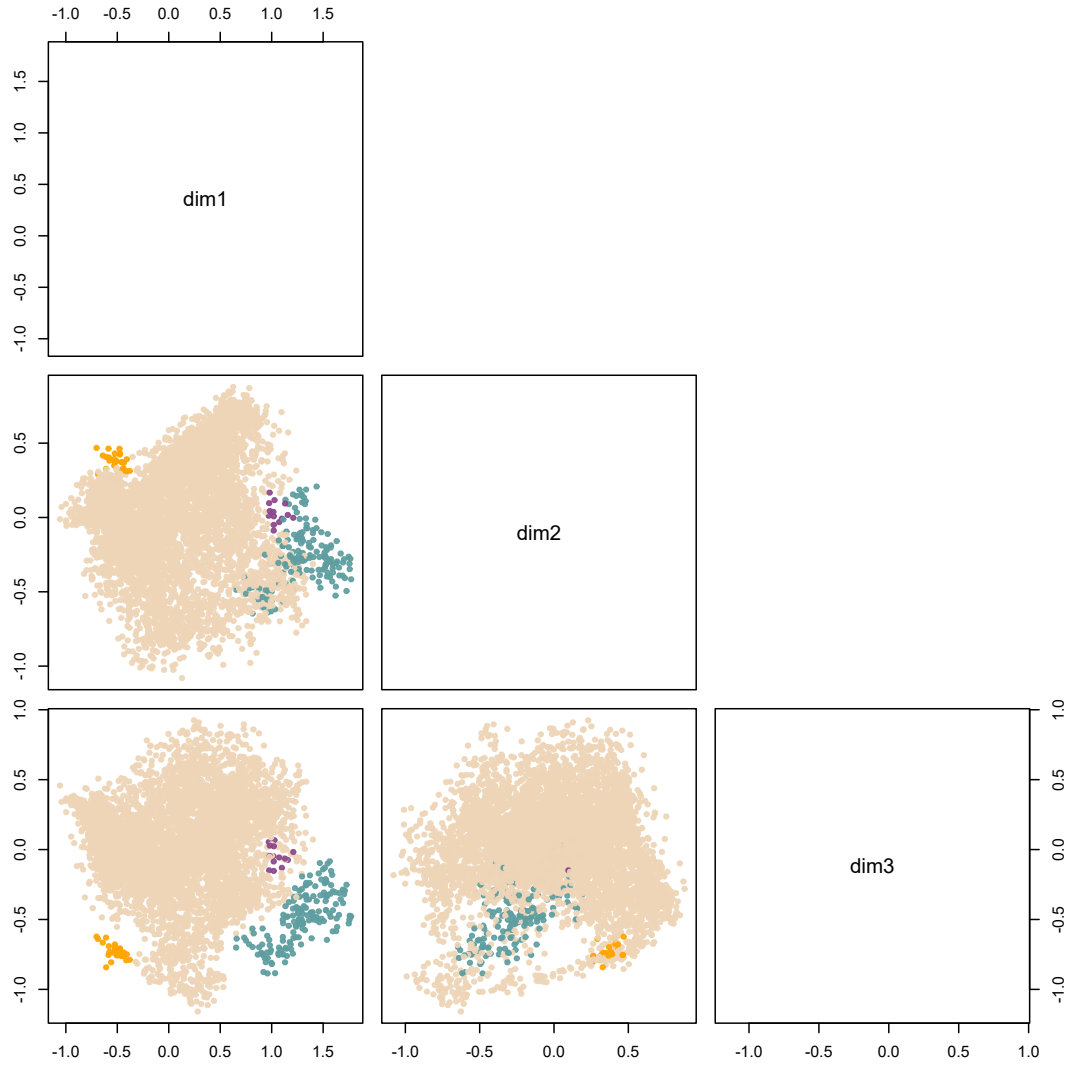


Figura 4.7: Groupings in three dimensions, excluding points classified as noise.

We analyze the composition of the groups by calculating

$$100\ \hat{\mathrm{F}}_{\mathrm{j}}\Big(\mathrm{M}_{\mathrm{e}}(\mathrm{Z}_{\mathrm{jk}})\Big), \tag{4.3}$$

where $\mathrm{j} \in \{1, 2, ..., \mathrm{p}\}$ is the variable, $\hat{\mathrm{F}}_{\mathrm{j}}$ is the empirical distribution function of the j-th variable obtained with all observations, and $\mathrm{M}_{\mathrm{e}}(\mathrm{Z}_{\mathrm{jk}})$ is the median of the observations of the j-th variable in the k-th group. This measure allows us to understand how the group is positioned relative to all observations, including those classified as noise. Values close to 100 indicate that the group has high values for the j-th variable, the opposite if they are close to 0. For values close to 50, it means that they do not deviate from the center of the cloud for that coordinate.

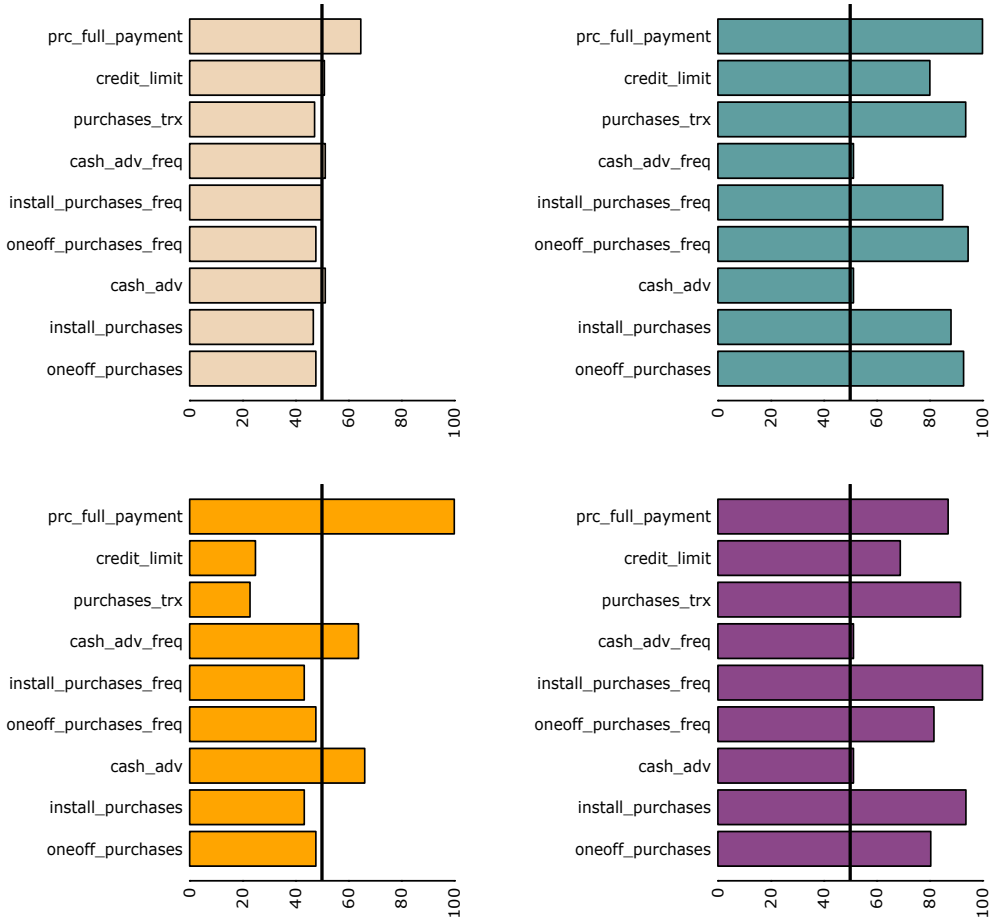


Figura 4.8: Values of 100 $\hat{\mathrm{F}}_{\mathrm{j}}\Big(\mathrm{M}_{\mathrm{e}}(\mathrm{Z}_{\mathrm{jk}})\Big)$ for the four groups.

In the four groups found, the first one (upper left, color peach) represents the majority of the bank's customers, positioned near the center of the cloud with no distinctive behavior. The second most representative group (upper right, color **blue**) consists of customers who make very frequent purchases, mostly in one payment. The next group (lower left, color **orange**) is composed of customers who almost exclusively use their card to request cash advances. They make very few purchases, have a low credit limit, and are characterized by having the full payment percentage near 100 %. The last group found (lower right, **violet**) is similar to the second, but they stand out for making installment purchases instead of one-time purchases.

A more detailed analysis of this composition could include comparing the correlation structures between different groups. The visual representation in Figure 4.7 allows us to see the weight of each group in the composition of customers and the proximity between different groups. An additional problem could be classifying new customers into these groups to offer products or promotions according to their consumption type.

4.2.3. Goodness of Fit

Fast MDS methods have the disadvantage that it is not possible to obtain the exact value of metrics such as *Stress-1* or the loss coefficient *Strain*, due to the number of distances and dissimilarities in large-sized samples. In this case, new ways of studying goodness of fit must be defined.

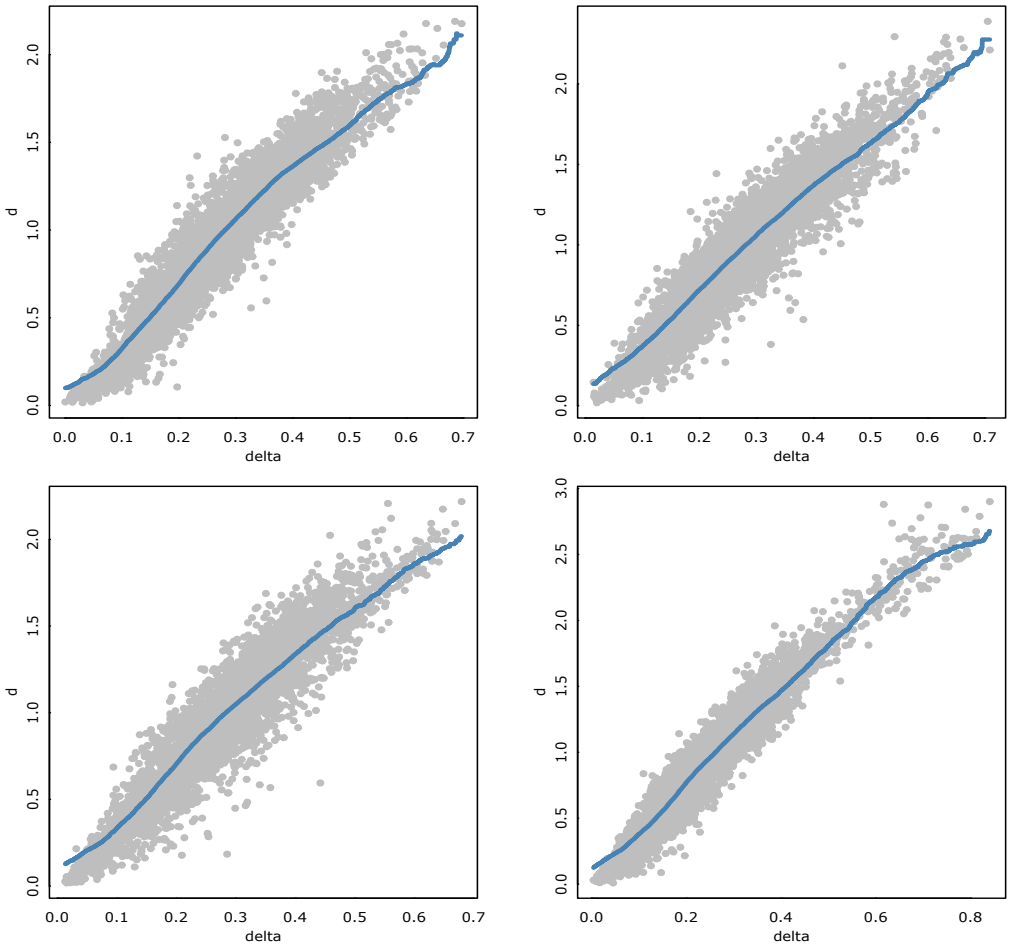


Figura 4.9: Dissimilarities versus Euclidean distances in $\mathcal{Z}$ for four samples of 100 clients.

Note that, by using distance scaling with mapping functions adjusted within each group of DCO-MDS, an additional cost is added if one wants to collect the exact disparities $\hat{d}_{ij}$ for each distance. Figure 4.9 shows the plots of dissimilarity δ_{ij} (in this case calculated with the Gower metric) versus Euclidean distance d_{ij} in the reduced space for four random samples of 100 observations. The plots overlay the Nadaraya-Watson adjusted regression function. A simple option to analyze the overall goodness of fit is to calculate the residuals of successive regressions like these and obtain the sums of squared residuals. With these sums, estimates of *Stress-1* can be obtained, and a non-parametric *bootstrap* can be used to construct, for example, a confidence interval.

Another method, similar to the one implemented in [3], involves calculating *Stress-1* within each block of DCO-MDS and then taking a weighted average of the different values according to the size of the groups. This approach loses information about cross-distances between different blocks.

4.2.4. Comparison with t-SNE

A widely used method for dimensionality reduction focused on clustering is t-SNE (t-distributed stochastic neighbor embedding), which is based on adjusting a similarity measure based on the probability that an observation chooses others as neighbors:

$$p_{i|j} = \frac{e^{-|x_i - x_j|^2/2\sigma_j^2}}{\sum_{k \neq j} e^{-|x_k - x_j|^2/2\sigma_j^2}} \tag{4.4}$$

Here, σ is a parameter determined based on the user-chosen perplexity. Perplexity is a coefficient typically between 5 and 50, allowing the formation of more or less dense neighborhoods of points. The method produces coordinates in reduced dimension so that the similarity $p_{i|j}$ calculated with these points fits as closely as possible to that calculated with the high-dimensional points.

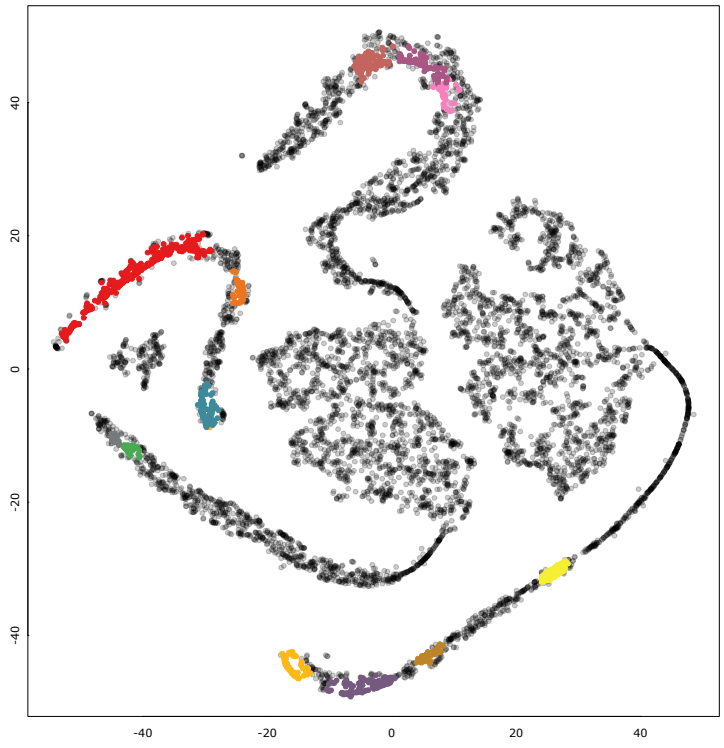


Figura 4.10: Customer clusters with t-SNE, with perplexity equal to 50.

The graph shows the result of clustering customers after reducing with t-SNE. Given a perplexity value, the same optimal group search was performed according to the Calinski-Harabasz index. Results were tested with different perplexity values, and it was found that configurations with values less than 50 formed a large number of small groups with highly distorted shapes. Apparently, this method tends to exaggerate non-linear structures, making the interpretation and spatial visualization of groups challenging.

Topics for Further Exploration

As a conclusion to the work, we enumerate some topics that we believe are worth exploring in depth:

Regarding the rapid methods themselves

- Improve the DCO-MDS method with QR decomposition to incorporate a change of scale. It was shown how this method has less capability than others to reproduce the classical solution. In line with this, seek a unified method of DCO-MDS that combines the two strategies seen in this work.
- Investigate the response of the DCO-INTERP method with the Gower formula using other dissimilarities, as it was only analyzed with Euclidean distances.
- We demonstrated that outliers significantly influence the solution. It may be interesting to propose robust variants of rapid methods.
- In the methods discussed, reference points or landmarks must be selected. In [7], a criterion for selecting these points is proposed instead of choosing them at random. This could reduce the influence of outliers or improve the quality of the final representation.
- Study in more detail the influence of hyperparameters—maximum block size, number of landmarks—on the solution's quality. In this work, we set the hyperparameters according to the general recommendations of the authors, but we noticed that the solution's quality varies when modifying them. A thorough sensitivity analysis was not performed.

Regarding the application of the methods

- Propose goodness-of-fit measures that require little computation time and adequately represent the solution's quality.
- Study the construction of nonlinear biplots using rapid methods to help understand the low-dimensional representation.
- Analyze in more detail the appropriateness of using different metrics for MDS in dimensionality reduction problems.
- Look for variable regularization methods in dimensionality reduction problems. For example, given a metric weighted by variables, in the form of

$$\mathrm{d_{ij}} = \sum_{\mathrm{k}=1}^{\mathrm{p}} \mathrm{w_k}\ \mathrm{d_{ijk}},$$

  where $\mathrm{d_{ijk}}$ is the distance term between $\mathrm{O_i}$ and $\mathrm{O_j}$ corresponding to the k-th variable, and $\mathrm{w_k}$ is a weight coefficient, analyze how to calibrate the weight coefficients to select variables

in multidimensional scaling. In other words, detect which variables help more in representing distances in low dimensions and to what extent. This may require implementing some optimization or search method.

- Investigate alternative applications, such as the simplified representation of graphs or networks with a large number of nodes.

Referencias

[1] Ingwer Borg y Patrick JF Groenen. *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2005.

[2] Michael AA Cox y Trevor F Cox. *Multidimensional scaling*. Chapman y Hall/CRC, 2001.

[3] Pedro Delicado y Cristian Pachon-Garcia. "Multidimensional Scaling for Big Data". En: *arXiv preprint arXiv:2007.11919* (2020).

[4] John C Gower y David J Hand. *Biplots*. Vol. 54. CRC Press, 1995.

[5] Samudra Herath, Matthew Roughan y Gary Glonek. "High Performance Out-of-sample Embedding Techniques for Multidimensional Scaling". En: *arXiv preprint arXiv:2111.04067* (2021).

[6] KV Mardia, JT Kent y JM Bibby. *Multivariate analysis, 1979*. Academic Press Inc, 1979.

[7] Emmanuel Paradis. "Multidimensional scaling with very large datasets". En: *Journal of Computational and Graphical Statistics* 27.4 (2018), págs. 935-939.

[8] Nasir Saeed et al. "A survey on multidimensional scaling". En: *ACM Computing Surveys (CSUR)* 51.3 (2018), págs. 1-25.

[9] Jengnan Tzeng, Henry Horng-Shing Lu y Wen-Hsiung Li. "Multidimensional scaling for large genomic data sets". En: *BMC bioinformatics* 9.1 (2008), págs. 1-17.