Smacof at 50
A Manual

Jan de Leeuw - University of California Los Angeles

Started February 21 2024, Version of March 24, 2024

Abstract

TBD

Contents

Note: This is a working paper which will be expanded/updated frequently. All suggestions for improvement are welcome. All Rmd, tex, html, pdf, R, and C files are in the public domain. Attribution will be appreciated, but is not required. The files can be found at https://github.com/deleeuw/smacofCode.

1 Introduction

In *Multidimensional Scaling (MDS)* the data consists of information about the similarity or dissimilarity between pairs of objects selected from a finite set $\mathcal{O} = \{o_1, \cdots, o_n\}$.

In *metric MDS* we have numerical dissimilarity measures and we want to map the objects o_i into n points x_i of some metric space in such a way that the distances between the points approximate the dissimilarities between the objects. In *smacof*, our framework for MDS theory, algorithms, and computer programs, the metric space is $\mathbb{R}^p$, the space of all p-tuples of real numbers, and in the code documented in this manual we assume the distance is the usual Euclidean distance.

In *non-metric MDS* the information about the dissimilarities is incomplete. It is usually *ordinal*, i.e. it tells us in some way or another that some dissimilarties are larger or smaller than others. Somewhere between metric and non-metric MDS is MDS with *missing data*, in which some dissimilarities are known numbers while others are unknown. MDS with missing data is a form of *distance matrix completion* (Fang and O'Leary (2012)).

1.1 Kruskal's Stress

In the pioneering papers Kruskal (1964a) and Kruskal (1964b) the MDS problem was formulated for the first time as minimization of an explicit loss function, which measures the quality of the approximation of the dissimilarities by the distances. The loss function in least squares metric Euclidean MDS is called *raw stress* or *Kruskal's raw stress* and is defined as

$$\sigma(X) := \frac{1}{2}\sum w_{ij}(\delta_{ij} - d_{ij}(X))^2. \tag{1}$$

The symbol $:=$ is used for definitions. In definition (1) the w_{ij} are known non-negative *weights*, the δ_{ij} are the known non-negative *dissimilarities* between objects o_i and o_j, and the $d_{ij}(X)$ are the *distances* between the corresponding points x_i and x_j. The summation is over all $\binom{n}{2}$ pairs (i, j) with $j > i$, i.e. over elements below the diagonal of the matrices W and Δ. From now on we use "metric MDS" to mean Least Squares Metric Euclidean MDS.

The $n \times p$ matrix X, which has the coordinates x_i of the n points as its rows, is called the *configuration*, where p is the *dimension* of the Euclidean space in which we make the map. Thus

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

The metric MDS problem (of dimension p, for given W and Δ) is the minimization of (1) over the $n \times p$ configurations X.

The weights w_{ij} can be used to quantify information about the precision or importance of the corresponding dissimilarities. Some of the weights may be zero, which can be used to code *missing data*. If all weights are positive we have *complete data*. If we have complete data, and all weights are equal to one, we have *unweighted* metric MDS. Weights were only introduced in MDS in De Leeuw (1977), the pioneering papers by Shepard, Kruskal, and Guttman only consider the unweighted case.

We assume throughout that the weights are *irreducible* (De Leeuw (1977)). This means there is no partitioning of the index set $I_n := \{1, 2, \cdots, n\}$ into subsets for which all between-subset weights are zero. A reducible metric MDS problems decomposes into a number of smaller independent metric MDS problems, so the irreducibility assumption causes no real loss of generality.

The fact that the summation in (1) is over all $j < i$ indicates that the diagonal elements of Δ are not used (they are assumed to be zero) and the elements above the diagonal are not used as well (they are assumed to be equal to the corresponding elements below the diagonal). The somewhat mysterious factor $\frac{1}{2}$ in definition (1) is there because it simplifies some of the formulas in later sections of this paper.

Kruskal was not really interested in metric MDS and the "raw" loss function (1). His papers are really about non-metric MDS, by which we mean least squares non-metric Euclidean MDS. Non-metric MDS differs from metric MDS because we have incomplete information about the dissimilarities. As we have seen, that if some dissimilarities are missing metric MDS can handle this by using zero weights. In some situations, however, we only know the rank order of the non-missing dissimilarities. We do not know, or we refuse to use, their actual numeric values. Or, to put it differently, even if we have numerical dissimilarities we are looking for a *transformation* of the non-missing dissimilarities, where the transformation is chosen from a set of admissible transformations (for instance from all linear or monotone transformations). If the dissimilarities are non-numerical, for example rank orders or partitionings, we choose from the set of admissible *quantifications*.

In non-metric MDS the loss function becomes

$$\sigma(X, \hat{D}) := \frac{1}{2}\sum w_{ij}(\hat{d}_{ij} - d_{ij}(X))^2, \tag{3}$$

where $\hat{D}$ are now the quantified or transformed dissimilarities. In MDS parlance they are also called *pseudo-distances* or *disparities*. Loss function (3) must be minimized over both configurations

and disparities, with the condition that the disparities $\hat{D}$ are an admissible transformation of the dissimilarities Δ. In Kruskal's non-metric MDS this means requiring monotonicity. In this paper we will consider various other choices for the set of admissible transformations. We will use the symbol $\mathfrak{D}$ for the set of admissible transformations

The most familiar sets of admissible transformations (linear, polynomial, monotone) define convex cones with apex at the origin. This means that if $\hat{D} \in \mathfrak{D}$ then so is $\lambda\hat{D}$ for all $\lambda \geq 0$. But consequently minimizing (3) over all $\hat{D} \in \mathfrak{D}$ and over all configurations has the trivial solution $\hat{D} = 0$ and $X = 0$, corresponding with the global minimum $\sigma(X, \hat{D}) = 0$. We need additional constraints to rule out this trivial solution, and in non-metric MDS this is done by choosing a *normalization* that keeps the solution away from zero.

Kruskal's solution is to normalize the raw stress by defining

$$\sigma(X, \Delta) := \frac{\sum w_{ij}(\hat{d}_{ij} - d_{ij}(X))^2}{\sum w_{ij} d_{ij}^2(X)}. \tag{4}$$

In fact in Kruskal's formulation there are no weights, and he actually takes the square root of (4) to define *Kruskal's stress*. The non-metric Euclidean MDS problem is to minimize loss function (4) over all $n \times p$ configurations X and all admissible disparities Δ.

1.2 Normalization

Normalization in non-metric MDS has been discussed in detail in Kruskal and Carroll (1969) and De Leeuw (1975). In the terminology of De Leeuw (1975) there are *explicit* and *implicit* normalizations.

In implicit normalization we minimize either

$$\sigma(X, \hat{D}) := \frac{\sum w_{ij}(\hat{d}_{ij} - d_{ij}(X))^2}{\sum w_{ij}\hat{d}_{ij}^2} \tag{5}$$

or

$$\sigma(X, \hat{D}) := \frac{\sum w_{ij}(\hat{d}_{ij} - d_{ij}(X))^2}{\sum w_{ij} d_{ij}^2(X)} \tag{6}$$

Kruskal (1964a) chooses definition (6) and calls the explicitly normalized loss function *normalized stress*. In fact, he takes the square root, which does not change the minimization problem, and only considers the unweighted case. Note that we overload the symbol σ to denote any one of the least squares loss functions. It will always be clear from the text which σ we are talking about.

In explicit normalization we minimize the raw stress $\sigma(X, \hat{D})$ from (3), but we add the constraint

$$\sum w_{ij} d_{ij}^2(X) = 1, \tag{7}$$

or the constraint

$$\sum w_{ij}\hat{d}_{ij}^2 = 1. \tag{8}$$

Kruskal and Carroll (1969) and De Leeuw (2019) show that these four normalizations all lead to essentially the same solution for X and $\hat{D}$, up to scale factors dictated by the choice of normalization.

It is also possible to normalize both X and $\hat{D}$, either explicitly or implicitly, and again this will give the same solutions, suitably normalized. These invariance results assume the admissible transformations form a closed cone with apex at the origin, i.e. if $\hat{D}$ is admissible and $\lambda \geq 0$ then $\lambda\hat{D}$ is admissible as well. The matrices of Euclidean distances $D(X)$ form a similar closed cone as well. The LSNE-MDS problem is to find an element of the $\hat{D}$ cone and an element of the $D(X)$ cone where the angle between the two is a small as possible.

In the R version of smacof (De Leeuw and Mair (2009), Mair, Groenen, and De Leeuw (2022)) we use explicit normalization (8). This is supported by the result, also due to De Leeuw (1975), that projection on the intersection of the cone of disparities and the sphere defined by (8) is equivalent to first projecting on the cone and then normalizing the projection (see also Bauschke, Bui, and Wang (2018)).

In our version of non-metric MDS we need more flexibility. For algorithmic reasons that will become clear later on, we will go with the other explicit normalization (7) and minimize σ from (3) over normalized X and unnormalized $\hat{D}$. For the final results the choice between (7) and (8) should not make a difference.

1.3 Some thoughts on ALS

I will take this opportunity to clear up some misunderstandings and confusions that have haunted the early development of non-metric MDS.

1.3.1 The Single-Phase approach

In Kruskal (1964a) defines

$$\sigma(X) := \min_{\hat{D}\in\mathfrak{D}} \sigma(\hat{D}, X) = \sigma(X, \hat{D}(X)), \tag{9}$$

where $\sigma(\hat{D}, X)$ is defined by (6). where the minimum is over admissible transformations. In definition (9)

$$\hat{D}(X) := \operatorname*{argmin}_{\hat{D}\in\mathfrak{D}} \sigma(X, \hat{D}). \tag{10}$$

Normalized stress defined by (9) is now a function of X only. Under some conditions, which are true in Kruskal's definition of non-metric MDS,

$$\mathcal{D}\sigma(X) = \mathcal{D}_1\sigma(X, \hat{D}(X)), \tag{11}$$

where $\mathcal{D}\sigma(X)$ are the derivatives of σ from (9) and $\mathcal{D}_1\sigma(X, \hat{D}(X))$ are the partial derivatives of σ from (6) with respect to X. Thus the partials of σ from (9) can be computed by evaluating the partials of σ from (6) with respect to X at $(X, \hat{D}(X))$. This has created much confusion in the past. The non-metric MDS problem is now to minimize σ from (9), which is a function of X alone.

Guttman (1968) calls this the *single-phase approach*. A variation of Kruskal's single-phase approach defines

$$\sigma(X) = \sum w_{ij}(d_{ij}^{\#}(X) - d_{ij}(X))^2$$

where the $d_{ij}^{\#}(X)$ are *Guttman's rank images*, i.e. the permutation of the $d_{ij}(X)$ that makes them monotone with the δ_{ij} (Guttman (1968)). Or, alternatively, define

$$\sigma(X) := \sum w_{ij}(d_{ij}^{\%}(X) - d_{ij}(X))^2$$

where the $\hat{d}_{ij}^{\%}(X)$ are *Shepard's rank images*, i.e. the permutation of the δ_{ij} that makes them monotone with the $d_{ij}(X)$ (Shepard (1962a), Shepard (1962b), De Leeuw (2017b)).

Minimizing the Shepard and Guttman single-phase loss functions is computationally more complicated than Kruskal's *monotone regression* approach, mostly because the rank-image transformations are not differentiable, and there is no analog of (11) and of the equivalence of the different implicit and explicit normalizations.

1.3.2 The Two-Phase Approach

The *two-phase approach* or *alternating least squares (ALS)* approach alternates minimization of $\sigma(\hat{D}, X)$ over X for our current best estimate of $\hat{D}$ with minimization of $\sigma(\hat{D}, X)$ over $\Delta \in \mathfrak{D}$ for our current best value of X. Thus an update from iteration k to iteration $k+1$ looks like

$$\hat{D}^{(k)} = \underset{\hat{D}\in\mathfrak{D}}{\text{argmin}}\, \sigma(\hat{D}, X^{(k)}), \tag{12}$$

$$X^{(k+1)} = \underset{X}{\text{argmin}}\, \sigma(\hat{D}^{(k)}, X). \tag{13}$$

This ALS approach to MDS was in the air since the early (unsuccessful) attempts around 1968 of Young and De Leeuw to combine Torgerson's classic metric MDS method with Kruskal's monotone regression transformation. All previous implementations of non-metric smacof use the two-phase approach, and we will do the same in this paper.

As formulated, however, there are some problems with the ALS algorithm. Step (12) is easy to carry out, using monotone regression. Step (13) means solving a metric scaling problem, which is an iterative proces that requires an infinite number of iterations. Thus, in the usual implementations, step (12) is combined with one of more iterations of a convergent iterative procedure for metric MDS, such as smacof. If we take only one of these *inner iterations* the algorithm becomes indistinguishable from Kruskal's single-phase method. This has also created much confusion in the past.

In the usual implementations of the ALS approach we solve the first subproblem (12) exactly, while we take only a single step towards the solution for given $\hat{D}$ in the second phase (13). If we have an infinite iterative procedure to compute the optimal $\hat{D} \in \mathfrak{D}$ for given X, then a more balanced approach would be to take several inner iterations in the first phase and several inner iterations in the second phase. How many of each, nobody knows. In our current implementation of smacof we take several inner iteration steps in the first phase and a single inner iteration step in the second phase.

2 Smacof Notation and Terminology

We discuss some standard MDS notation, first introduced in De Leeuw (1977). This notation is useful for the second phase of the ALS algorithm, in which solve the metric MDS problem of we

minimizing unnormalized $\sigma(X, \hat{D})$ over X for fixed $\hat{D}$. We will discuss the first ALS phase later in the paper.

Start with the unit vectors e_i of length n. They have a non-zero element equal to one in position i, all other elements are zero. Think of the e_i as the columns of the identity matrix.

Using the e_i we define for all $i \neq j$ the matrices

$$A_{ij} := (e_i - e_j)(e_i - e_j)'. \tag{14}$$

The A_{ij} are of order n, symmetric, doubly-centered, and of rank one. They have four non-zero elements. Elements (i, i) and (j, j) are equal to $+1$, elements (i, j) and (j, i) are -1.

The importance of A_{ij} in MDS comes from the equation

$$d_{ij}^2(X) = \text{tr } X'A_{ij}X. \tag{15}$$

In addition we use the fact that the A_{ij} form a basis for the $binomn2$-dimensional linear space of all doubly-centered symmetric matrices.

Expanding the square in the definition of stress gives

$$\sigma(X) = \frac{1}{2}\{\sum w_k\delta_k^2 - 2\sum w_k\delta_k d_k(X) + \sum w_k d_k^2(X)\}. \tag{16}$$

It is convenient to have notation for the three separate components of stress from equation (16). Define

$$\eta_{\hat{D}}^2 = \sum w_{ij}\hat{d}_{ij}^2, \tag{17}$$

$$\rho(X) = \sum w_{ij}\hat{d}_{ij}d_{ij}(X), \tag{18}$$

$$\eta^2(X) = \sum w_{ij}d_{ij}(X)^2. \tag{19}$$

which lead to

$$\sigma(X) = \frac{1}{2}\left\{\eta_{\hat{D}}^2 - 2\rho(X) + \eta^2(X)\right\}. \tag{20}$$

We also need

$$\lambda(X) = \frac{\rho(X)}{\eta(X)}. \tag{21}$$

Using the A_{ij} makes it possible to give matrix expressions for ρ and η^2. First

$$\eta^2(X) = \text{tr } X'VX, \tag{22}$$

with

$$V := \sum w_{ij}A_{ij}. \tag{23}$$

In the same way

$$\rho(X) = \text{tr } X'B(X)X, \tag{24}$$

with

$$B(X) := \sum w_{ij} r_{ij}(X) A_{ij}, \tag{25}$$

with

$$r_{ij}(X) := \begin{cases} \frac{\delta_{ij}}{d_{ij}(X)} & \text{if } d_{ij}(X) > 0, \\ 0 & \text{if } d_{ij}(X) = 0. \end{cases} \tag{26}$$

Note that B is a function from the set of $n \times p$ configurations into the set of symmetric doubly-dentered matrices of order n. All matrices of the form $\sum x_{ij} A_{ij}$, where summation is over all pairs (i, j) with $j < i$, are symmetric and doubly-centered. They have $-x_{ij}$ as off-diagonal elements while the diagonal elements (i, i) are $\sum_{j=1}^{n} x_{ij}$.

Because $B(X)$ and V are non-negative linear combinations of the A_{ij} they are both positive semi-definite. Because W is assumed to be irreducible the matrix V has rank $n - 1$, with only vectors proportional to the vector e with all elements equal to one in its null-space (De Leeuw (1977)).

Summarizing the results so far we have

$$\sigma(X) = \frac{1}{2}\{\eta_D^2 - \text{tr } X'B(X)X + \text{tr } X'VX\}. \tag{27}$$

Next we define the *Guttman transform* of a configuration X, for given W and Δ, as

$$G(X) = V^+ B(X) X, \tag{28}$$

with V^+ the Moore-Penrose inverse of V. In our computations we use

$$V^+ = (V + \frac{1}{n} ee')^{-1} - \frac{1}{n} ee'$$

Also note that in the unweighted case with complete data $V = nJ$, where J is the centering matrix $I - \frac{1}{n} ee'$, and thus $V^+ = \frac{1}{n} J$. The Guttman transform is then simply $G(X) = n^{-1} B(X) X$.

3 Properties of Smacof Loss

3.1 Derivatives

The Euclidean distance function d_{ij} from … is not differentiable at configurations X with $x_i = x_j$. If $d_{ij}(X) > 0$ then

$$\mathcal{D}\sigma(X) = \frac{1}{d_{ij}(X)} A_{ij} X$$

If $d_{ij}(X) = 0$ then

$$D_+ d_{ij}(X, Y) = \lim_{\epsilon \downarrow 0} \frac{d_{ij}(X + \epsilon Y) - d_{ij}(X)}{\epsilon} = d_{ij}(Y)$$

which is non-linear in Y, showing non-differentiability.

$$D_+\sigma(X,Y) = \text{tr}\, Y'(V - B(X))X + \sum\{w_{ij}\delta_{ij}d_{ij}(Y) \mid d_{ij}(X) = 0\}$$

This form of the directional derivative is used by De Leeuw (1984) to show that two independent necessary conditions for a local minimum are $(V - B(X))X = 0$ and $d_{ij}(X) > 0$ for all (i, j) with $w_{ij}\delta_{ij} > 0$. ### Gradient

$$\mathcal{D}\sigma(X) = (V - B(X))X$$

At a stationary point $B(X)X = VX$ or $V^+B(X)X = X$. Thus a necessary condition for a local minimum is that $V^+B(X)$ has at least p eigenvalues equal to one. De Leeuw (2014) has shown that if $V^+B(X) \lesssim I$ then actually X is a global minimizer of stress.

3.1.1 Hessian

The results on the Hessian of stress are largely unpublished. So we summarize them here in this manual, so they'll be even more unpublished.

$$H_{st}(X) := \sum w_{ij}\frac{\delta_{ij}}{d_{ij}(X)}\left\{\frac{A_{ij}x_s x_t' A_{ij}}{d_{ij}^2(X)}\right\}$$

$$H_{st}(X) = \sum w_{ij}\frac{\delta_{ij}}{d_{ij}^3(X)}(x_{is} - x_{js})(x_{it} - x_{jt})A_{ij}$$

$$\mathcal{D}_{st}\sigma(X) = \begin{cases} H_{st}(X) & \text{if } s \neq t, \\ V - B(X) + H_{st} & \text{if } s = t. \end{cases}$$

If I_p is the identity matrix of order p, and $\otimes$ is the Kronecker product, then

$$\mathcal{D}^2\sigma(X) = I_p \otimes (V - B(X)) + H(X)$$

$$\sum_{s=1}^p\sum_{t=1}^p y_s' H_{st} y_t = \sum w_{ij}\frac{\delta_{ij}}{d_{ij}(X)}\left\{\frac{(\text{tr}\, Y'A_{ij}X)^2}{d_{ij}^2(X)}\right\} \leq \sum w_{ij}\frac{\delta_{ij}}{d_{ij}(X)}\text{tr}\, Y'A_{ij}Y = \text{tr}\, Y'B(X)Y.$$

Thus

$$0 \lesssim H \lesssim I_p \otimes B(X),$$

and

$$I_p \otimes (V - B(X)) \lesssim \mathcal{D}^2\sigma(X) \lesssim I_p \otimes V$$

At a local minimum of σ

$$0 \lesssim \mathcal{D}^2\sigma(X) \lesssim I_p \otimes V$$

In comparing the lower bounds on $\mathcal{D}^2\sigma(X)$ in … and … De Leeuw (2014) shows that $V - B(X) \gtrsim 0$ is sufficient for a *global* minimum of stress (but far from necessary).

Also

$$\sum_{t=1}^{p} H_{st} y_t = \sum w_{ij} \frac{\delta_{ij}}{d_{ij}(X)} \left\{ \frac{\text{tr } Y' A_{ij} X}{d_{ij}^2(X)} \right\} A_{ij} x_s$$

If $Y = X$ then $H(X)y = (I_p \otimes B(X))x$ and thus

$$\mathcal{D}^2\sigma(X)x = (I_p \otimes V)x.$$

In the unweighted case this means that X is an eigenvector of $\mathcal{D}^2\sigma(X)$ with eigenvalue n. Inequalities ... show that this is actually the largest eigenvalue. Or $(I_p \otimes V)^+ \mathcal{D}^2\sigma(X) \lesssim I$.

If $Y = XT$ with T anti-symmetric then tr $Y' A_{ij} X = 0$ then thus $H(X)y = 0$. Thus

$$\sum_{t=1}^{p} \mathcal{D}_{st}\sigma(X) y_t = (V - B(X))y_t$$

which is zero if $\mathcal{D}\sigma(X)$ is zero. Thus at a stationary point of stress $\mathcal{D}^\sigma(X)$ has $\frac{1}{2}p(p-1)$ zero eigenvalues.

There are several ways to think of the Hessian. The simplest one (perhaps) is as an $np \times np$ symmetric matrix (corresponding to

column-major R vector of length $\frac{1}{2}np(np+1)$). This is what we would use for a straightforward version of Newton-Raphson.

It is more elegant, however, to think of H as a symmetric super-matrix of order p, with as elements $n \times n$ matrices. And, for some purposes, such as the pseudo-confidence ellipsoids in De Leeuw (2017a), as a super-matrix of order n with as elements $p \times p$ matrices. Both the super-matrix interpretations lead to four-dimensional arrays, the first a $p \times p \times n \times n$ array, the second an $n \times n \times p \times p$ array. The different interpretations lead to different ways to store the Hessian in memory, and to different ways to retrieve its elements. Of course we can write routines to transform from one interpretation to another.

3.2 Lagrangian

In our implementation of the smacof algorithm we minimize stress over configurations with $\eta(X) = 1$, or, equivalently, $\sum w_{ij} d_{ij}^2(X) = 1$. This means we do not look for X with $\mathcal{D}\sigma(X) = (V - B(X))X = 0$, but we look for solutions of

$$(V - B(X))X - \lambda VX = 0, \text{tr } X'VX = 1.$$

At the solution

$$\lambda = 1 - \rho(X)$$

and

$$X = \frac{\Gamma(X)}{\eta(\Gamma(X))}$$

Also it is necessary for a local minimum that

$$\Gamma(X) = \rho(X)X$$

Because the Guttman transform is homogeneous of degree zero this implies

$$\Gamma(\Gamma(X) = \Gamma(X),$$

so although X is not a fixed point of the Guttman transform, $\Gamma(X)$ is.

The second order necessary condition is that

$$H(X) \gtrsim I_p \otimes (\rho(X)V - B(X))$$

is positive

3.2.1 Kuhn-Tucker Points

4 Smacof Algorithm

4.1 First Phase: Update Configuration

4.1.1 Introduction to Majorization

Majorization, more recently better known as MM (Lange (2016)), is a general approach for the construction of minimization algorithms. There is also minorization, which leads to maximization algorithms, which explains the MM acronym: minorization for maximization and majorization for minimization.

Before the MM principle was formulated as a general approach to algorithm construction there were some important predecessors. Major classes of MM algorithms avant la lettre were the *EM Algorithm* for maximum likelihood estimation of Dempster, Laird, and Rubin (1977), the *Smacof Algorithm* for MDS of De Leeuw (1977), the *Generalized Weiszfeldt Method* of Vosz and Eckhardt (1980), and the *Quadratic Approximation Method* of Böhning and Lindsay (1988). The first formulation of the general majorization principle seems to be De Leeuw (1994).

Let's start with a brief introduction to majorization. Minimize a real valued function σ over $x \in \mathbb{S}$, where $\mathbb{S}$ is some subset of $\mathbb{R}^n$. There are obvious extensions of majorization to functions defined on more general spaces, with values in any partially ordered set, but we do not need that level of generality in this manual. Also majorization applied to σ is minorization applied to $-\sigma$, so concentrating on majorization-minimization and ignoring minorization-maximization causes no loss of generality

Suppose there is a real-valued function η on $\mathbb{S} \otimes \mathbb{S}$ such that

$$\sigma(x) \leq \eta(x, y) \qquad \forall x, y \in \mathbb{S}, \tag{29}$$
$$\sigma(x) = \eta(x, x) \qquad \forall x \in \mathbb{S}. \tag{30}$$

The function η is called a *majorization scheme* for σ on S. A majorization scheme is *strict* if $\sigma(x) < \eta(x, y)$ for all $x, y \in S$ withj $x \neq y$.

Define

$$x^{(k+1)} \in \operatorname*{argmin}_{x \in \mathbb{S}} \eta(x, x^{(k)}), \tag{31}$$

assuming that $\eta(\bullet, y)$ attains its (not necessarily unique) minimum over $x \in \mathbb{S}$ for each y. If $x^{(k)} \in \operatorname{argmin}_{x \in \mathbb{S}} \eta(x, x^{(k)})$ we stop.

If we do not stop, then by update rule (31)

$$\sigma(x^{(k+1)}) \leq \eta(x^{(k+1)}, x^{(k)}), \tag{32}$$

by majorization property (29)

$$\eta(x^{(k+1)}, x^{(k)}) \leq \eta(x^{(k)}, x^{(k)}). \tag{33}$$

and by majorization property (29)

$$\eta(x^{(k)}, x^{(k)}) = \sigma(x^{(k)}). \tag{34}$$

If the minimum in (31) is attained for a unique x then $\eta(x^{(k+1)}, x^{(k)}) < \eta(x^{(k)}, x^{(k)})$. If the majorization scheme is strict then $\sigma(x^{(k+1)}) < \eta(x^{(k+1)}, x^{(k)})$. Under either of these two additional conditions $\sigma(x^{(k+1)}) < \sigma(x^{(k)})$, which means that the majorization algorithm is a monotone descent algorithm, and if σ is bounded below on $\mathbb{S}$ then the sequence $\sigma(x^{(k)})$ converges.

Note that we only use the order relation to prove convergence of the sequence of function values. To prove convergence of the $x^{(k)}$ we need stronger compactness and continuity assumptions to apply the general theory of Zangwill (1969). For such a proof the argmin in update formula (31) can be generalized to $x^{(k+1)} = \phi(x^{(k)})$, where ϕ maps $\mathbb{S}$ into $\mathbb{S}$ such that $\eta(\phi(x), x) \leq \sigma(x)$ for all x.

We give a small illustration in which we minimize σ with $\sigma(x) = \sqrt{x} - \log x$ over $x > 0$. Obviously we do not need majorization here, because solving $\mathcal{D}\sigma(x) = 0$ immediately gives $x = 4$ as the solution we are looking for.

To arrive at this solution using majorization we start with

$$\sqrt{x} \leq \sqrt{y} + \frac{1}{2}\frac{x - y}{\sqrt{y}}, \tag{35}$$

which is true because a differentiable concave function such as the square root is majorized by its tangent everywhere. Inequality (35) implies

$$\sigma(x) \leq \eta(x, y) := \sqrt{y} + \frac{1}{2}\frac{x - y}{\sqrt{y}} - \log x. \tag{36}$$

Note that $\eta(\bullet, y)$ is convex in its first argument for each y. We have $\mathcal{D}_1\eta(x, y) = 0$ if and only if $x = 2\sqrt{y}$ and thus the majorization algorithm is

$$x^{(k+1)} = 2\sqrt{x^{(k)}} \tag{37}$$

The sequence $x^{(k)}$ converges monotonically to the fixed point $x = 2\sqrt{x}$, i.e. to $x = 4$. If $x^{(0)} < 4$ the sequence is increasing, if $x^{(0)} < 4$ it is decreasing. Also, by l'Hôpital,

$$\lim_{x\to 4}\frac{2\sqrt{x}-4}{x-4}=\frac{1}{2} \tag{38}$$

and thus convergence to the minimizer is linear with asymptotic convergence rate $\frac{1}{2}$. By another application of l'Hôpital

$$\lim_{x\to 4}\frac{\sigma(2\sqrt{x}))-\sigma(4)}{\sigma(x)-\sigma(4)}=\frac{1}{4}, \tag{39}$$

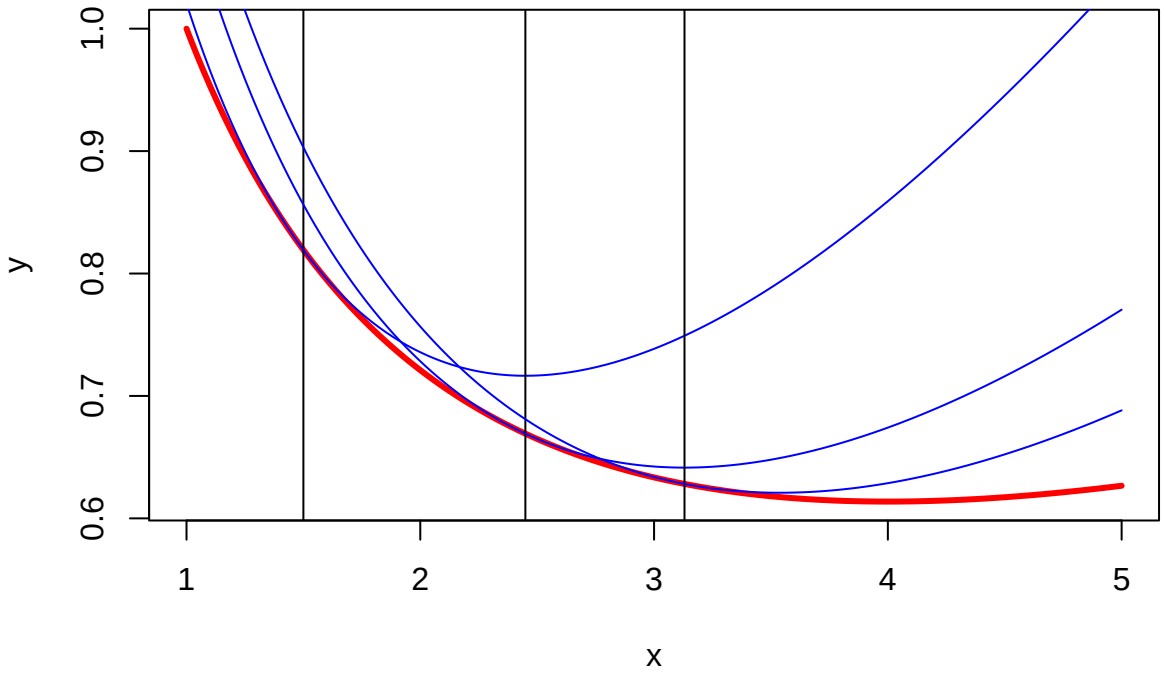
and convergence to the minimum is linear with asymptotic convergence rate $\frac{1}{4}$. Linear convergence to the minimizer is typical for majorization algorithms, as is the twice-as-fast linear convergence to the minimum value.

This small example is also of interest, because we minimize a *DC function*, the difference of two convex functions. In our example the convex functions are minus the square root and minus the logarithm. Algorithms for minimizing DC functions define other important subclasses of MM algorithms, the *DC Algorithm* of Tao Pham Dinh (see Le Thi and Tao (2018) for a recent overview), the *Concave-Convex Procedure* of Yuille and Rangarajan (2003), and the *Half-Quadratic Method* of Donald Geman (see Niikolova and Ng (2005) for a recent overview). For each of these methods there is a huge literature, with surprisingly little non-overlapping literatures. The first phase of the smacof algorithm, in which we improve the configuration for given disparities, is DC, concave-convex, and half-quadratic.

In the table below we show convergence of (37) starting at $x = 1.5$. The first column show how far $x^{(k)}$ deviates from the minimizer (i.e. from 4), the second shows how far$\sigma(x^{(k)})$ deviates from the minimum (i.e. from $2 - \log 4$). We clearly see the convergence rates $\frac{1}{2}$ and $\frac{1}{4}$ in action.

```
## itel   1 2.5000000000 0.2055741244
## itel   2 1.5505102572 0.0554992066
## itel   3 0.8698308399 0.0144357214
## itel   4 0.4615431837 0.0036822877
## itel   5 0.2378427379 0.0009299530
## itel   6 0.1207437506 0.0002336744
## itel   7 0.0608344795 0.0000585677
## itel   8 0.0305337787 0.0000146606
## itel   9 0.0152961358 0.0000036675
## itel  10 0.0076553935 0.0000009172
## itel  11 0.0038295299 0.0000002293
## itel  12 0.0019152235 0.0000000573
## itel  13 0.0009577264 0.0000000143
## itel  14 0.0004788919 0.0000000036
## itel  15 0.0002394531 0.0000000009
```

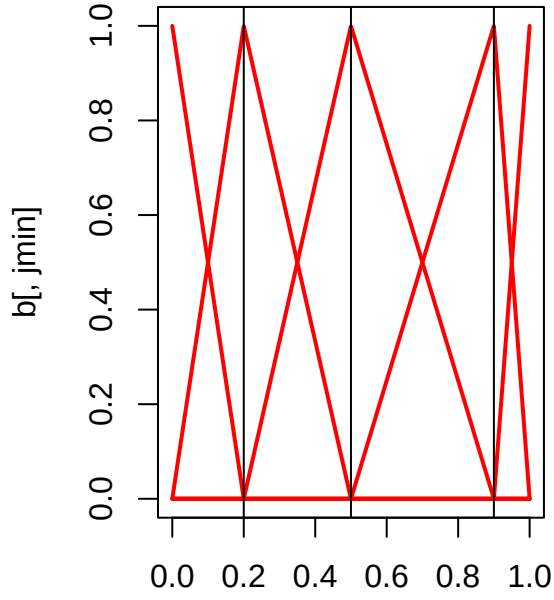
The first three iterations are shown in the figure below. The vertical lines indicate the value of x, function is in red, and the first three majorizations are in blue.

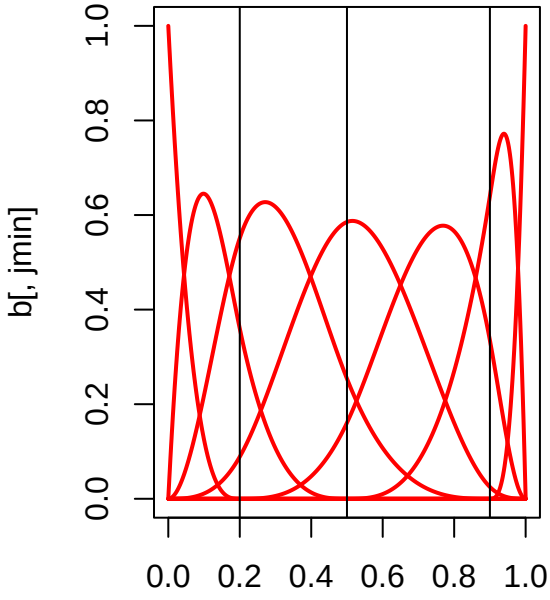


4.1.2 Majorizing Stress

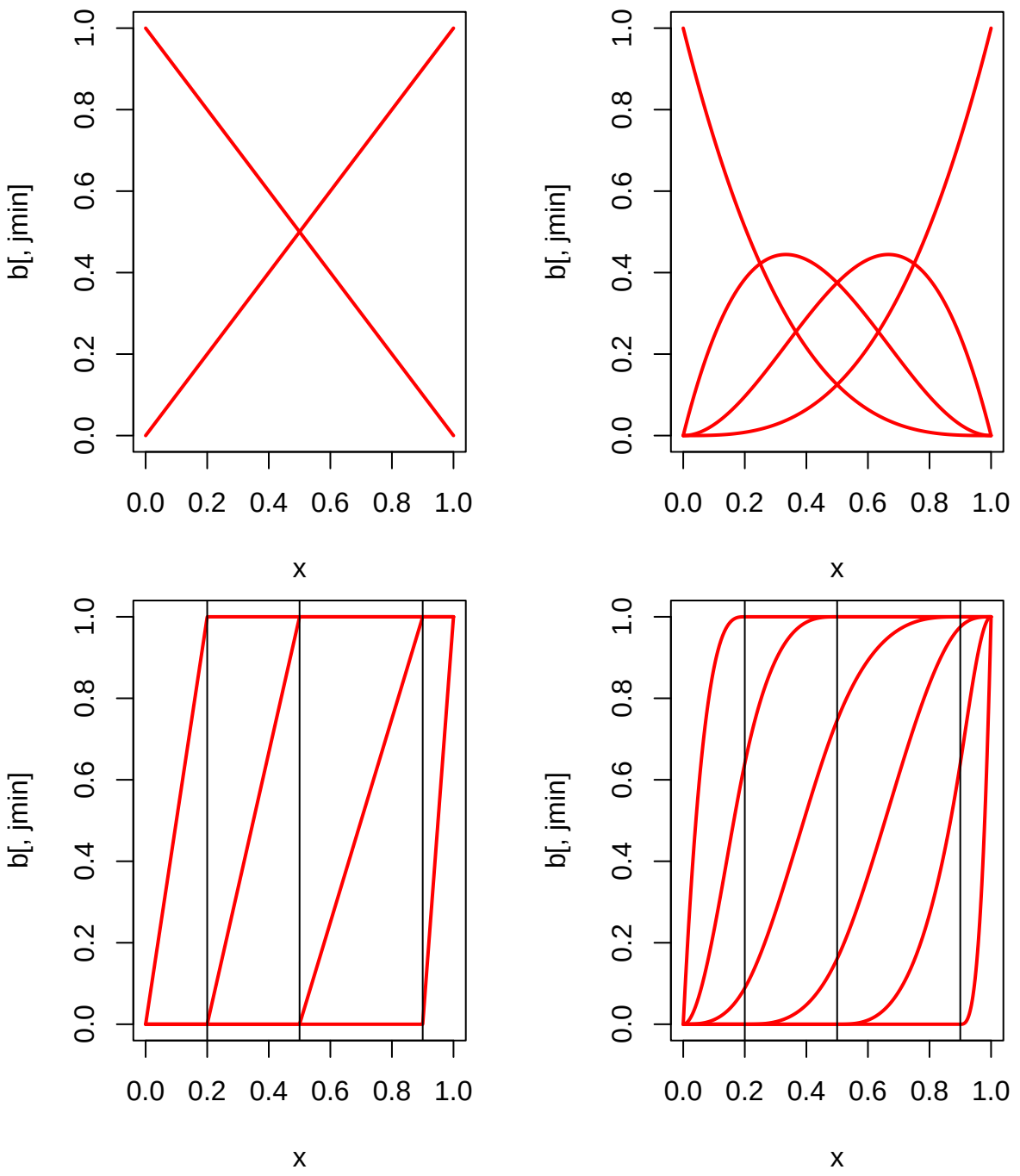
4.2 Second Phase: Update Transformation

4.2.1 Spline Basis Details





4.2.1.1 B-splines



ninner m, degree k, order $d = k + 1$, nknots $m + 2d$, span $p = d + m$

B-splines

$B_{i,k}(x)$ is zero outside $[t_i, t_{i+k+1}]$

```
inner = c(.1, .5, .55, .9)
x = 0:10/10
print(x)
```

```
##  [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

```r
for (degree in 0:4) {
  ord <- degree + 1
  knots <- c(rep(0, ord), inner, rep(1, ord))
  a <- splineDesign(knots = knots, ord = ord, x = x)
  for (i in 1:11) {
    cat(formatC(x[i], digits = 2, format = "f"), " *** ",
        formatC(a[i, ], digits = 4, format = "f"), "\n")
  }
  cat("\n\n")
}
```

```
## 0.00  ***  1.0000 0.0000 0.0000 0.0000 0.0000
## 0.10  ***  0.0000 1.0000 0.0000 0.0000 0.0000
## 0.20  ***  0.0000 1.0000 0.0000 0.0000 0.0000
## 0.30  ***  0.0000 1.0000 0.0000 0.0000 0.0000
## 0.40  ***  0.0000 1.0000 0.0000 0.0000 0.0000
## 0.50  ***  0.0000 0.0000 1.0000 0.0000 0.0000
## 0.60  ***  0.0000 0.0000 0.0000 1.0000 0.0000
## 0.70  ***  0.0000 0.0000 0.0000 1.0000 0.0000
## 0.80  ***  0.0000 0.0000 0.0000 1.0000 0.0000
## 0.90  ***  0.0000 0.0000 0.0000 0.0000 1.0000
## 1.00  ***  0.0000 0.0000 0.0000 0.0000 1.0000
##
##
## 0.00  ***  1.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## 0.10  ***  0.0000 1.0000 0.0000 0.0000 0.0000 0.0000
## 0.20  ***  0.0000 0.7500 0.2500 0.0000 0.0000 0.0000
## 0.30  ***  0.0000 0.5000 0.5000 0.0000 0.0000 0.0000
## 0.40  ***  0.0000 0.2500 0.7500 0.0000 0.0000 0.0000
## 0.50  ***  0.0000 0.0000 1.0000 0.0000 0.0000 0.0000
## 0.60  ***  0.0000 0.0000 0.0000 0.8571 0.1429 0.0000
## 0.70  ***  0.0000 0.0000 0.0000 0.5714 0.4286 0.0000
## 0.80  ***  0.0000 0.0000 0.0000 0.2857 0.7143 0.0000
## 0.90  ***  0.0000 0.0000 0.0000 0.0000 1.0000 0.0000
## 1.00  ***  0.0000 0.0000 0.0000 0.0000 0.0000 1.0000
##
##
## 0.00  ***  1.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## 0.10  ***  0.0000 0.8000 0.2000 0.0000 0.0000 0.0000 0.0000
## 0.20  ***  0.0000 0.4500 0.4944 0.0556 0.0000 0.0000 0.0000
## 0.30  ***  0.0000 0.2000 0.5778 0.2222 0.0000 0.0000 0.0000
## 0.40  ***  0.0000 0.0500 0.4500 0.5000 0.0000 0.0000 0.0000
## 0.50  ***  0.0000 0.0000 0.1111 0.8889 0.0000 0.0000 0.0000
## 0.60  ***  0.0000 0.0000 0.0000 0.6429 0.3413 0.0159 0.0000
```

```
## 0.70  ***  0.0000 0.0000 0.0000 0.2857 0.5714 0.1429 0.0000
## 0.80  ***  0.0000 0.0000 0.0000 0.0714 0.5317 0.3968 0.0000
## 0.90  ***  0.0000 0.0000 0.0000 0.0000 0.2222 0.7778 0.0000
## 1.00  ***  0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 1.0000
##
##
## 0.00  ***  1.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## 0.10  ***  0.0000 0.6400 0.3236 0.0364 0.0000 0.0000 0.0000 0.0000
## 0.20  ***  0.0000 0.2700 0.4946 0.2284 0.0069 0.0000 0.0000 0.0000
## 0.30  ***  0.0000 0.0800 0.3826 0.4818 0.0556 0.0000 0.0000 0.0000
## 0.40  ***  0.0000 0.0100 0.1627 0.6398 0.1875 0.0000 0.0000 0.0000
## 0.50  ***  0.0000 0.0000 0.0101 0.5455 0.4444 0.0000 0.0000 0.0000
## 0.60  ***  0.0000 0.0000 0.0000 0.2411 0.6748 0.0824 0.0018 0.0000
## 0.70  ***  0.0000 0.0000 0.0000 0.0714 0.5571 0.3238 0.0476 0.0000
## 0.80  ***  0.0000 0.0000 0.0000 0.0089 0.2752 0.4954 0.2205 0.0000
## 0.90  ***  0.0000 0.0000 0.0000 0.0000 0.0444 0.3506 0.6049 0.0000
## 1.00  ***  0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 1.0000
##
##
## 0.00  ***  1.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## 0.10  ***  0.0000 0.5120 0.3928 0.0912 0.0040 0.0000 0.0000 0.0000 0.0000
## 0.20  ***  0.0000 0.1620 0.4228 0.3575 0.0569 0.0008 0.0000 0.0000 0.0000
## 0.30  ***  0.0000 0.0320 0.2219 0.5299 0.2038 0.0123 0.0000 0.0000 0.0000
## 0.40  ***  0.0000 0.0020 0.0524 0.4738 0.4093 0.0625 0.0000 0.0000 0.0000
## 0.50  ***  0.0000 0.0000 0.0009 0.2516 0.5499 0.1975 0.0000 0.0000 0.0000
## 0.60  ***  0.0000 0.0000 0.0000 0.0804 0.4606 0.4408 0.0180 0.0002 0.0000
## 0.70  ***  0.0000 0.0000 0.0000 0.0159 0.2413 0.5657 0.1613 0.0159 0.0000
## 0.80  ***  0.0000 0.0000 0.0000 0.0010 0.0691 0.4122 0.3952 0.1225 0.0000
## 0.90  ***  0.0000 0.0000 0.0000 0.0000 0.0049 0.1096 0.4149 0.4705 0.0000
## 1.00  ***  0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 1.0000
```

degree	order	ninner	nknots	span
0	1	4	6	5
1	2	4	8	6
2	3	4	10	7
3	4	4	12	8
4	5	4	14	9

$$\sum_i B_{i,k}(x) = 1$$

M-splines

$$M_{i,k}(x) = \frac{k+1}{t_{i+k+1} - t_i} B_{i,k}(X)$$

then

$$\int M_{i,k}(x)dx = 1$$

I-splines

$$I_{i,k+1}(z) = \int_{-\infty}^{z} M_{i,k}(x)dx$$

When is a B-spline increasing ?

$$\mathcal{D}B_{i,k}(x) =$$

Thus if

$$\mathcal{D}\sum_{i=1}^{d+m} \alpha_i B_{i,k}(x) =$$

It is sufficient that $\alpha_i \leq \alpha_{i+1}$

Integral, I-splines

4.2.2 Ordinal MDS

4.2.3 Interval and Ratio MDS

4.2.4 Cyclic Coordinate Decent

In the non-linear least squares (NNLS) problem the data are an $n \times p$ matrix X, a vector y with n elements, and a positive semi-definite diagonal matrix W. We want to minimize

$$\sigma(\beta) := \frac{1}{2}(X\beta - y)'W(X\beta - y)$$

over $\beta \geq 0$. In data analysis and statistics the problem is often solved by *active set methods*, implemented in R for example by NNLS (Mullen and van Stokkum (2023)) and FNNLS (Bro and De Jong (1997)). Active set methods are finitely convergent dual methods. While iterating the intermediate solutions are not feasible (i.e. non-negative). In fact in dual methods we reach feasibility and optimality at the same time. Also the number of iterations, although theoretically finite, can be very large.

In each smacof iteration we need an NNLS solution. Especially in the early iterations the solution does not have to be very precise. Also the solution from the previous NNLS problem will generally provide a very good starting value for the next iteration (each NNLS problem has a "hot start"). And finally, we would like all internediate solutions to be feasible. These considerations have lead us to using *cyclic coordinate descent* (CCD).

Suppose the current best feasible solution in CCD iteration k is $\beta^{(k)}$. The next CCD iteration changes each of the p coordinates of $\beta^{(k)}$ in turn, maintaining feasibility, while keeping the other

$p-1$ coordinates fixed at their current values. Thus within a CCD iteration k we create intermediate solutions $\beta^{(k,1)}, \cdots, \beta^{(k,p)}$, where each of the intermediate solutions $\beta^{(k,r)}$ differs from the previous one $\beta^{(k,r-1)}$ in a single coordinate. For consistency we define $\beta^{(k,0)} := \beta^{(k)}$. After the iteration is finished we set $\beta^{(k+1)} = \beta^{(k,p)}$.

Note that in smacof each iteration modifies the coordinates in the order $1, \cdots, p$, which explains why the method is called "cyclic". There are variations of CCD in which the order within an iteration is random or greedy (choose the coordinate which gives the largest improvement) or zig-zag $1, \cdots, p, p-1, \cdots, 1$. We have not tried out these alternatives in smacf, but we may in the future.

The effect of changing a single coordinate on the loss function is

$$\sigma(\beta + \epsilon e_j) = \sigma(\beta) + \epsilon\, g_j(\beta) + \frac{1}{2}\epsilon^2 s_{jj},$$

where e_j is the unit vector corresponding with the coordinate we are changing, $g(\beta) := \mathcal{D}\sigma(\beta) = X'Wr(\beta)$ is the gradient at β, and $r(\beta) := X\beta - y$ is the residual. Also $S := X'WX$. Note that if $s_{jj} = 0$ then also $g_j(\beta) = 0$ and thus $\sigma(\beta + \epsilon e_j) = \sigma(\beta)$. In each CCD cycle we simply skip updating coordinate j.

If $s_{jj} > 0$ then $\sigma(\beta + \epsilon e_j)$ is a strictly convex quadratic in ϵ, which we must minimize under the constraint $\beta_j + \epsilon \geq 0$ or $\epsilon \geq -\beta_j$. Define $\hat{\epsilon}$ to be the solution of this constrained minimization problem.

The quadratic … has its minimum at

$$\tilde{\epsilon} = -\frac{g_j(\beta)}{s_{jj}}$$

If $\beta + \tilde{\epsilon}$ is feasible then it is the update we are looking for. Thus $\hat{\epsilon} = \tilde{\epsilon}$. If $\beta + \tilde{\epsilon} < 0$ then the contrained minimum is attained at the boundary, i.e. $\hat{\epsilon} = -\beta_j$ and the updated β_j is zero. Thus, in summary, $\hat{\epsilon} = \max(\tilde{\epsilon}, -\beta_j)$.

One of the nice things about CCD is that

$$r(\hat{\beta}) = r(\beta) + \hat{\epsilon} x_j$$

$$g(\hat{\beta}) = g(\beta) + \hat{\epsilon} s_j$$

It follows that $\hat{\epsilon} = 0$ if and only if either $\beta_j = 0$ and $g_j(\beta) \geq 0$ or if $g_j(beta) = 0$ and $beta_j > 0$.

If $g_j(\beta) < 0$ then $\tilde{\epsilon} > 0$, and thus $\hat{\epsilon} > 0$ and $\sigma(\hat{\beta}) < \sigma(\beta)$. Thus we must have $g_j(\beta) \geq 0$.

If $\beta_j > 0$ and $g_j(\beta) \neq 0$ then there is an ϵ such that $\sigma(\beta + \epsilon e_j) < \sigma(\beta)$. Thus if $\beta_j > 0$ we must have $g_j(\beta) = 0$.

In summary at the minimum of σ over $\beta \geq 0$ we must have $\beta_j \geq 0$, $g_j(\beta) \geq 0$, and $\beta_j g_j(\beta) = 0$ for all j (*complementary slackness*).

$$\sigma(\beta + \epsilon e_j) = \sigma(\beta) + \epsilon\, g_j(\beta) + \frac{1}{2}\epsilon^2 s_{jj},$$

where $S := X'WX$.

Now suppose we minimize σ over $\beta \geq 0$.

Our best solution so far is $\beta^{(k)} \geq 0$. Minimize $\sigma(\beta^{(k)} + \epsilon e_1)$ over ϵ on the condition that $\beta_1^{(k)} + \epsilon \geq 0$ or $\epsilon \leq -\beta_1^{(k)}$. If $s_{11} = 0$ then also $g_1(\beta) = 0$ and we set $\beta^{(k+1,1)} = \beta^{(k,1)}$. If $s_{11} > 0$ we compute

$$\tilde{\epsilon} = -g_1(\beta)/s_{11}$$

If

$$\beta_1^{(k)} + \tilde{\epsilon} \geq 0$$

then

$$\beta^{(k+1,1)} = \beta_1^{(k)} + \tilde{\epsilon}$$

If

$$\beta_1^{(k)} + \tilde{\epsilon} < 0$$

we set

$$\beta^{(k+1,1)} = 0.$$

5 Smacof Program

5.0.1 Front-end

The front-end for both smacofRR and smacofRC is written in R. The analysis is started in the user's working directory by the command smacofRR(foo) or smacofRC(foo), where foo is a user-chosen name (without quotes).

Two text files need to be present in the working directory. The first is fooParameters.txt, where of course you substitute the user-chosen name for foo. The second file is fooDelta.txt, which has the dissimilarities below the diagonal in row-major order.

The parameter file has key-value format. Here is an example.

```
nobj    9
ndim    3
init    2
width    10
precision    6
haveweights    0
itmax   1000
epsi   10
verbose   1
ditmax   5
depsi   6
dverbose   0
kitmax   5
kepsi   6
```

```
kverbose  0
degree  3
haveknots  3
ninner  5
ordinal  1
origin  1
```

The parameter file is read first, using the R function read.table(). There is one key-value pair at the start of each line. The order of the lines does not matter. There can be additional comments or other text on each line after the value field, as long as the text is space-separated from the value field. Additional key-value lines with non-existing parameters can be added at will.

Values of the parameters are put the local environment using R function assign(), which means they are available to R throughout the smacof run. Of course if we choose smacofRC the front-end needs to pass them to C using C(), but they will be available again for the back-end.

The Delta file, and any subsequent optional input files, are read with the R function scan(). Values are separated by spaces. They can be on a single line, or laid out as a lower-triangular matrix, or whatever. The function scan() only stops reading if it reaches the end-of-file.

We'll now discuss the parameters one by one. Note that all parameters are integers. The first two are obvious: *nobj* is the number of objects and *ndim* the number of dimensions. These two parameters have no default or recommended values, because they deyermined by the data. All other parameters in our example parameter file are set to reasonable values in our example parameter file. But the whole idea is to experiment with various combinations of parameter values, so "reasonable" is weaker than "recommended" and "recommended" is weaker than "default".

The *init* parameter can have values 1, 2, or 3. If *init* equals 1 the program reads an initial configuration from the file fooXinit.txt in the working directory. The file has *nobj* * *ndim* numbers, the initial configuration, in row-major format. If *init* = 2 then the classical Torgerson initial estimate will be computed. If *init* = 3 a random initial estimate will be used.

width and *precision* are parameters for the output of the values of stress during iterations.

haveweights is either zero or one. If zero there are no weights, which is equivalent to all weights equal to one. If one then we will read a file fooWeights.txt, which has the lower-diagonal $\frac{1}{2}n(n-1)$ weights in row-major order.

As explained in previous sections there are three iterative running in smacof. There are two inner iterations: one for the configuration for fixed disparities, and one for the disparities for fixed configuration. The two inner iteration loops are nested in one outer iteration loop. Each of the iterations has three parameters: one for the maximum number of iterations, one for the stopping criterion, and one for the verbosity of the iteration output. For the outer loop the parameters are *itmax*, *ieps*, and *verbose*. For the inner configuration loop they are *kitmax*, *keps*, and *kverbose*. And the inner transformation loop they are *ditmax*, *deps*, and *dverbose*. If the verbose parameter is one, then each iteration prints out the stress before and the stress after update. If verbose is zero, nothing is printed. The stopping parameters check if the change in stress in an iteration is less than epsilon, where epsilon is 10^-ieps, 10^-keps, or 10^-deps.

The final five parameters are used to define the nature of the spline space for the transformations. *degree* is the degree of the piecewise polynomials. The *haveknots* parameter can be 0, 1, 2, or 3. If it is zero, there are no inner knots and we use the Bernstein polynomial basis. If *haveknots* is one, the inner knots are read in from fooKnots.txt in the usual way. If *haveknots* is two the knots are equally spaced between zero and one, and if it is three the knots are equally spaced on the precentile scale (so that the number of data points between knots is approximately the same). The *ninner* parameter determines the number of knots in the case that *haveknots* is either two or three. If *haveknots* is zero, then *ninner* should be zero, if *haveknots* is one it should be equal to the number of knots in fooKnots.txt.

The two final spline parameters are *ordinal* and *origin*. If *ordinal* is one the fitted spline will be monotone, if *origin* is one it is constrained to go through the origin.

The computations in the frontend are straightforward. We first transform the dissimilarities linearly so that the smallest becomes zero and the largest becomes one. This is not strictly necessary but it makes the spline computations slightly easier.

Initial Estimates for X Spline Basis

5.0.2 Engine

ALS First Phase Second Phase

5.0.3 Back-end

The back-end consists of a number of R functions that have the list returned by smacofRR or smacofRC as an argument. They can be used to make plots, compute derivatives, convert matrices to an easily printable format, do sensitivity analysis, and so on. The philosophy is that in the backend the main computing is finished and we just create different representations of the results.

5.0.3.1 Plotting There are two main plot functions in the backend: smacofShepardPlot() and smacofConfigurationPlot(). A smacofShepardPlot has the dissimilarities (un-normalized) on the horizontal axes and it has the distances and the disparities on the vertical axis. It draws the spline, and shows where the fitted disparities are on the spline. It also plots the (delta, dist) pairs as points, to show how far they deviate from the spline. Optionally smacofShepardPlot() can draw vertical lines at the inner knots (argument knotlines = TRUE), and optionally it can connect the (delta, dhat) points on the spline to the (delta, dist) points with lines (argument fitlines = TRUE).

It must be emphasized that smacofShepardPlot() draws the spline over the whole interval, which is either (deltamin, 1) if origin = FALSE and (0, 1) if origin = TRUE. It does this by recomputing the spline at a large number of uniformly spaced points in the interval, where the number of points is given by the smacofShepardPlot() parameter resolution. Thus we do not use only the data points (delta,dhat) and then let the R plot function interpolate linearly. That can be misleading. It is especially misleading if degree is zero (step function) or if there are consecutive inner knots with no data values between the knots. Degree zero is handled by the special purpose step function plotting routine smacofPlotStepFunction(), which makes sure the spline is drawn as a horizontal

segment from one knot to the next knot. In addition smacofShepardPlot() can set some base R plot parameters such as col, cex, lwd, and pch (see the R documentation).

The smacofConfigurationPlot() function is much simpler than smacofShepardPlot(). It sets pch, col, and cex. It uses the smacofRR/RC labels parameter to decide how to label the points in the configuration. If labels = 1 it reads a character vector of labels from fooLabels.txt, where foo is of course the name of the run. If labels = 2 the points are numbered, if labels = 3 plotting uses the pch symbol for all points. If the dimension p is larger than two, smacofConfigurationPlot() uses the parameters dim1 and dim2 to select the dimensions to plot.

5.0.3.2 Writing

5.0.3.3 Checking

5.0.3.4 Derivatives

5.0.3.5 Sensitivity Perturbation regions

Parametric Bootstrap

Jacknife

6 Examples

6.1 dcity

To construct the first example we use 9 points equally spaced on a circle in the plane. The dissimilarities in dcityDelta.txt are the city block distances between these nine points. We did five analyses of these data, with parameters and results in table 2. In all analyses the transformation is required to be monotone, and the eps and itmax parameters are the same for all.

Table 2: Analyses of the dcity example

ninner	degree	haveknots	origin	stress	itel
0	1	0	1	0.026343760	13
0	1	0	0	0.004798412	27
0	3	0	1	0.009020231	30
5	3	3	1	0.004491316	416
25	0	3	1	0.002466866	359

The transformation in the first analysis is a line through the origin, in the second analysis it also is a line, but there is an intercept. Thus the first analysis is what is usually called “ratio”, the second is “interval”. Shepard plots are in figure 1. Note that the red points are (delta,dhat) points, which

are on the spline, while the blue points are (delta,dist) points. The fit is the sum of squares of the line-lengths between corresponding red and blue points.

The results plotted in figure 2 show the monotone cubic polynomial fit, and a cubic spline with five interior knots. Finally, figure 3 shows a zero-degree spline with 25 interior knots, which emulates a classical Kruskal non-metric MDS solution. The figure on the left, with knotslines = TRUE, is a but crowded, so we also show the alternative with knotsline = FALSE and fitlines = TRUE. Note that in the upper regions there are some pieces of the splines where there are no data points (some empty steps).

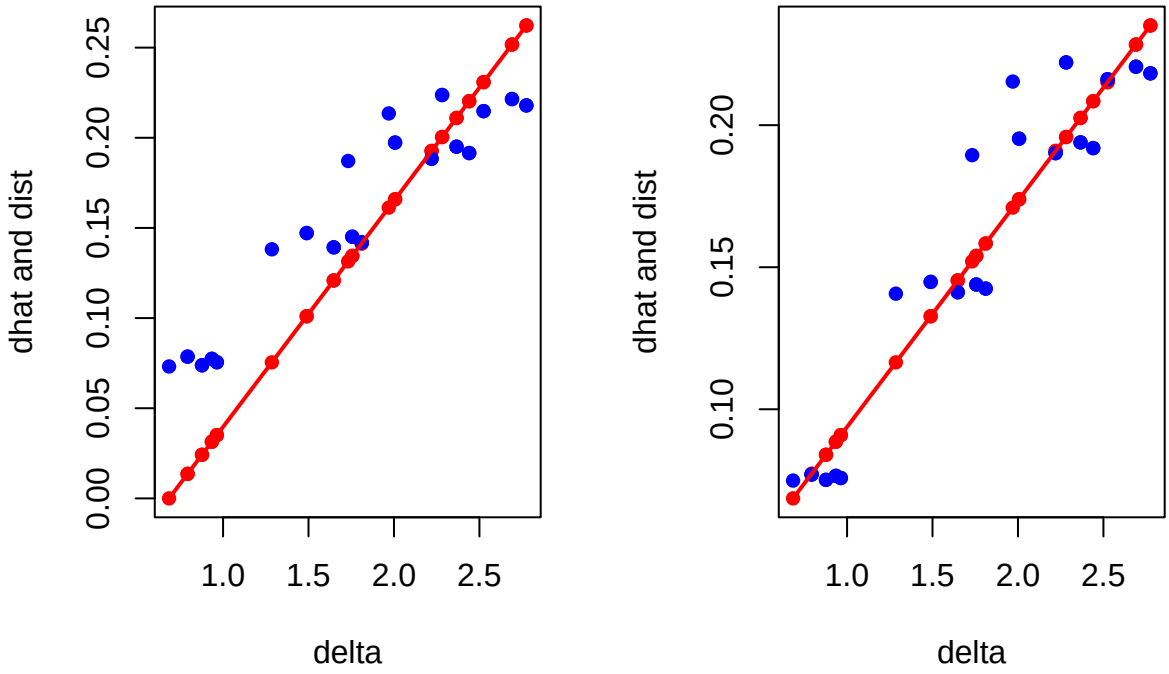


Figure 1: dcity example, linear analysis

We do not show the configuration plots for this example, but all five are basically the same: nine points approximately equally spaced on a circle.

6.2 Ekman

The Ekman (1954) color circle example has been used in many, if not most, multidimensional scaling textbooks and review articles. This is due, no doubt, to its astonishing good fit and its easy interpretability.

Four analyses this time, all monotone through the origin. In this case, because the minimum dissimilarity is zero anyway, the origin parameter has almost no effect.

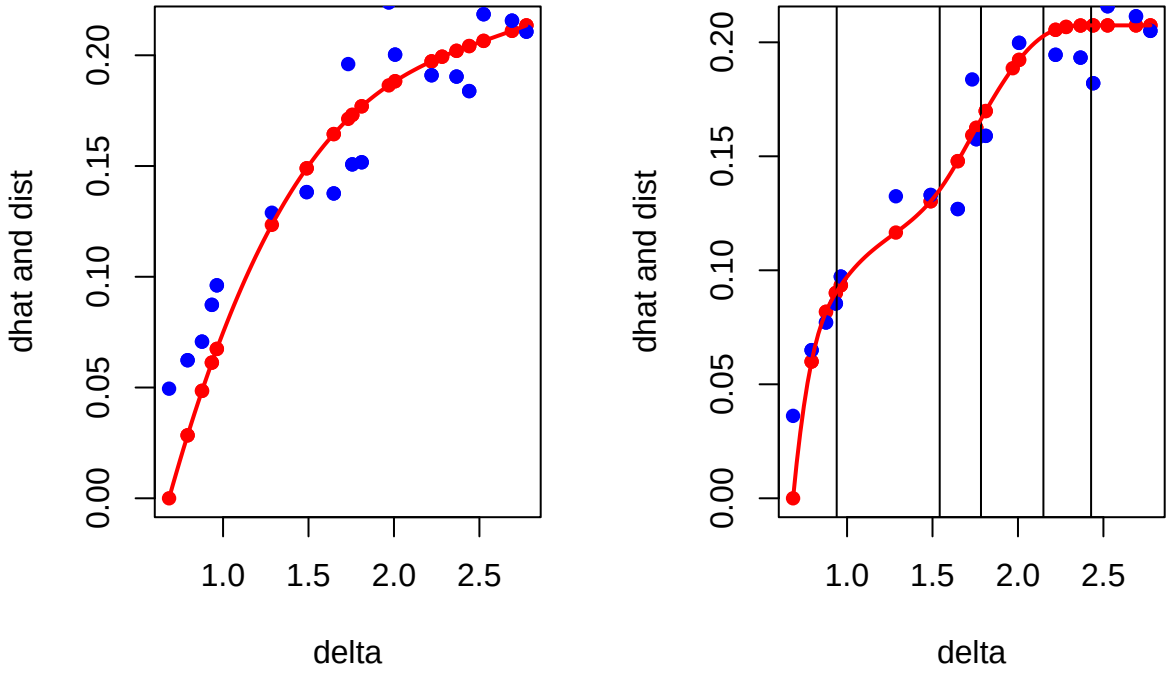


Figure 2: dcity example, cubic analysis

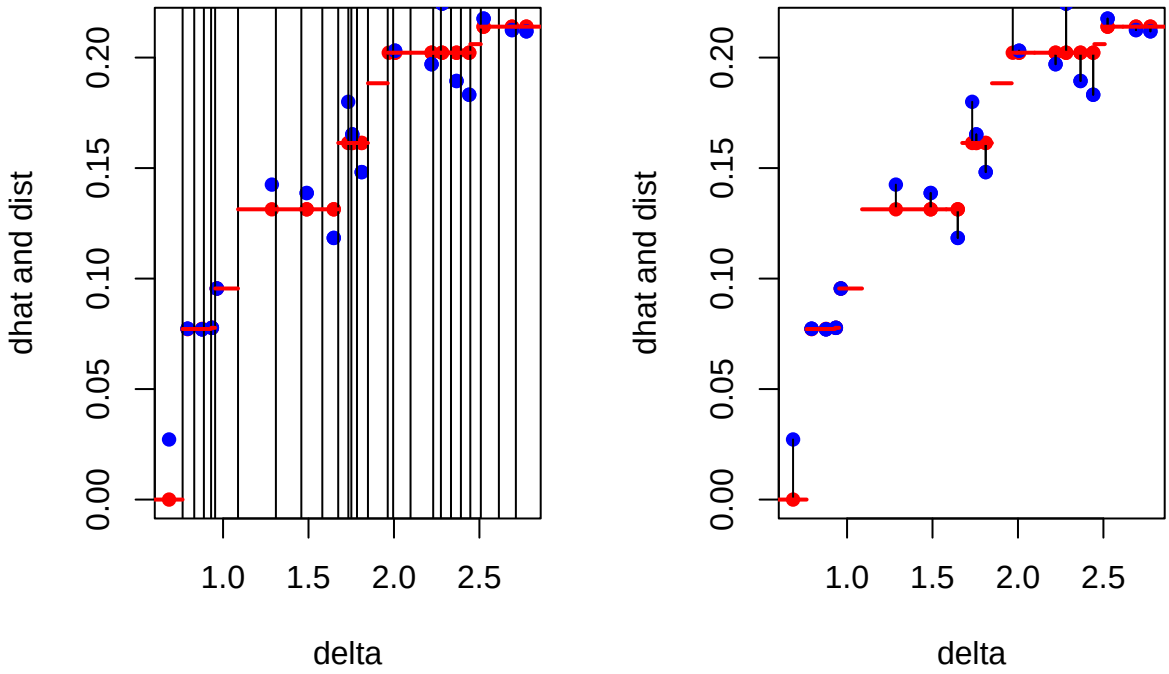


Figure 3: dcity example, nonmetric analysis

Table 3: Analyses of the Ekman example

ninner	degree	haveknots	origin	stress	itel
0	3	0	1	0.0022444459	38
5	3	3	1	0.0009260466	1268
50	0	3	1	0.0005138551	877
50	0	2	1	0.0011423393	810

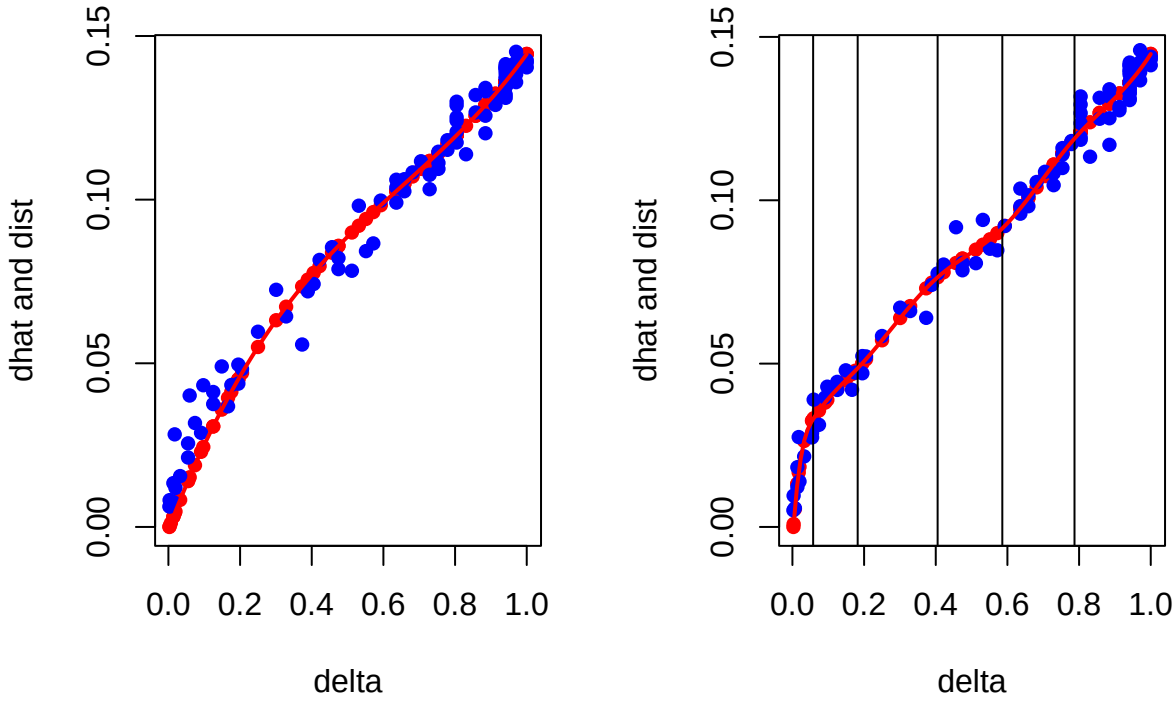


Figure 4: Ekman example, cubic analysis

References

Bauschke, H. H., M. N. Bui, and X. Wang. 2018. "Projecting onto the Intersection of a Cone and a Sphere." *SIAM Journal on Optimization* 28: 2158–88.

Böhning, D., and B. G. Lindsay. 1988. "Monotonicity of Quadratic-approximation Algorithms." *Annals of the Institute of Statistical Mathematics* 40 (4): 641–63.

Bro, R., and S. De Jong. 1997. "A Fast Non-Negatively-Constrained Least Squares Algorithm." *Journal of Chemometrics* 11: 393–401.

De Leeuw, J. 1975. "A Normalized Cone Regression Approach to Alternating Least Squares Algorithms." Department of Data Theory FSW/RUL.

———. 1977. "Applications of Convex Analysis to Multidimensional Scaling." In *Recent Developments in Statistics*, edited by J. R. Barra, F. Brodeau, G. Romier, and B. Van Cutsem, 133–45. Amsterdam, The Netherlands: North Holland Publishing Company.

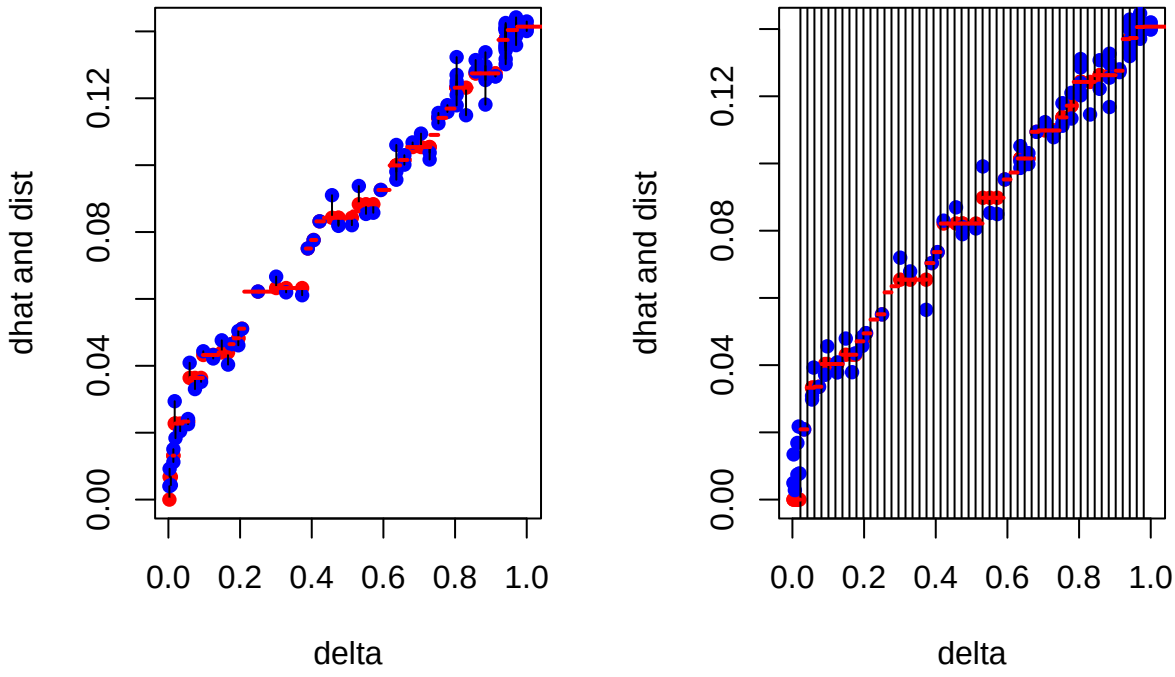


Figure 5: Ekman example, nonmetric analysis

———. 1984. "Differentiability of Kruskal's Stress at a Local Minimum." *Psychometrika* 49: 111–13.

———. 1994. "Block Relaxation Algorithms in Statistics." In *Information Systems and Data Analysis*, edited by H. H. Bock, W. Lenski, and M. M. Richter, 308–24. Berlin: Springer Verlag. https://jansweb.netlify.app/publication/deleeuw-c-94-c/deleeuw-c-94-c.pdf.

———. 2014. "Bounding, and Sometimes Finding, the Global Minimum in Multidimensional Scaling." UCLA Department of Statistics. https://jansweb.netlify.app/publication/deleeuw-u-14-b/deleeuw-u-14-b.pdf.

———. 2017a. "Pseudo Confidence Regions for MDS." 2017.

———. 2017b. "Shepard Non-metric Multidimensional Scaling." 2017.

———. 2019. "Normalized Cone Regression." 2019. https://jansweb.netlify.app/publication/deleeuw-e-19-d/deleeuw-e-19-d.pdf.

De Leeuw, J., and P. Mair. 2009. "Multidimensional Scaling Using Majorization: SMACOF in R." *Journal of Statistical Software* 31 (3): 1–30. https://www.jstatsoft.org/article/view/v031i03.

Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. "Maximum Likelihood for Incomplete Data via the EM Algorithm." *Journal of the Royal Statistical Society* B39: 1–38.

Ekman, G. 1954. "Dimensions of Color Vision." *Journal of Psychology* 38: 467–74.

Fang, H., and D. P. O'Leary. 2012. "Euclidean Distance Matrix Completion Problems Euclidean Distance Matrix Completion Problems Euclidean Distance Matrix Completion Problems." *Oprtimization Methods and Software* 27 (4-5): 695–717.

Guttman, L. 1968. "A General Nonmetric Technique for Fitting the Smallest Coordinate Space for a Configuration of Points." *Psychometrika* 33: 469–506.

Kruskal, J. B. 1964a. "Multidimensional Scaling by Optimizing Goodness of Fit to a Nonmetric Hypothesis." *Psychometrika* 29: 1–27.

———. 1964b. “Nonmetric Multidimensional Scaling: a Numerical Method.” *Psychometrika* 29: 115–29.

Kruskal, J. B., and J. D. Carroll. 1969. “Geometrical Models and Badness of Fit Functions.” In *Multivariate Analysis, Volume II*, edited by P. R. Krishnaiah, 639–71. North Holland Publishing Company.

Lange, K. 2016. *MM Optimization Algorithms*. SIAM.

Le Thi, H. A., and P. D. Tao. 2018. “DC Programming and DCA: Thirty Years of Developments.” *Mathematical Programming, Series B*.

Mair, P., P. J. F. Groenen, and J. De Leeuw. 2022. “More on Multidimensional Scaling in R: smacof Version 2.” *Journal of Statistical Software* 102 (10): 1–47. https://www.jstatsoft.org/article/view/v102i10.

Mullen, K. M., and I. H. M. van Stokkum. 2023. *nnls: The Lawson-Hanson algorithm for non-negative least squares (NNLS)*. %7Bhttps://CRAN.R-project.org/package=nnls%7D.

Niikolova, M., and M. Ng. 2005. “Analysis of Half-Quadratic Minimization Methods for Signal and Image Recovery.” *SIAM Journal Scientific Computing* 27 (3): 937–66.

Shepard, R. N. 1962a. “The Analysis of Proximities: Multidimensional Scaling with an Unknown Distance Function. I.” *Psychometrika* 27: 125–40.

———. 1962b. “The Analysis of Proximities: Multidimensional Scaling with an Unknown Distance Function. II.” *Psychometrika* 27: 219–46.

Vosz, H., and U. Eckhardt. 1980. “Linear Convergence of Generalized Weiszfeld’s Method.” *Computing* 25: 243–51.

Yuille, A. L., and A. Rangarajan. 2003. “The Concave-Convex Procedure.” *Neural Computation* 15: 915–36.

Zangwill, W. I. 1969. *Nonlinear Programming: a Unified Approach*. Englewood-Cliffs, N.J.: Prentice-Hall.