# More on Inverse Multidimensional Scaling

Jan de Leeuw, Patrick Groenen, Patrick Mair

First created on July 22, 2016. Last update on November 27, 2021

## Contents

Note: This is a working paper which will be expanded/updated frequently. It is a dynamic and reproducible upgrade of De Leeuw (2012), which itself is an update of "Inverse Multidimensional Scaling" (2007). The directory deleeuwpdx.net/pubfolders/inverse has a pdf copy of this article, the complete Rmd file with all code chunks, the bib file, and the R source code.

# 1 Problem

A *multidimensional scaling* or *MDS* problem, as used in this paper, starts with a matrix of *weights* $W$ and a matrix of *dissimilarities* $\Delta$. Both $W$ and $\Delta$ are non-negative, symmetric, and hollow (i.e. have zero diagonal). The problem is to find an $n \times p$ *configuration* $X$ such that the *stress* loss function, first defined in Kruskal (1964a), Kruskal (1964b),

$$\sigma(X, W, \Delta) := \sum_{1\leq i<j\leq n}\sum w_{ij}(\delta_{ij} - d_{ij}(X))^2 \tag{1}$$

is minimized over $X$. Here the matrix $D(X)$ has *distances*

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

Thus in this paper by MDS we mean *metric least squares MDS using Euclidean distances.*

Our notation emphasizes that stress is a function of configuration, weights, and dissimilarities. In actual metric MDS problems weights and dissimilarities are usually fixed numbers, although researchers may sometimes experiment with different weights and different transformations of the dissimilarities. In non-metric MDS the dissimilarities are additional parameters in the optimization problem, subject to monotonicity constraints.

The function $X(W, \Delta)$ mapping $W$ and $\Delta$ to the MDS solution $X$ is complicated, set-valued, and non-linear. We have $X \in X(W, \Delta)$ if and only if stress has a global minimum at $X$. To evaluate it for a given $W$ and $\Delta$ requires us to find the global minimum of stress, which is very hard and practically impossible for $n$ large. A larger map is defined by $X \in X(W, \Delta)$ if and only if stress has a local minimum at $X$. But again, finding all local minima is a very difficult problem.

It is somewhat easier, and certainly more convenient, to study the function which associates with each $W$ and $\Delta$ the set of solutions to the stationary equations. Using standard MDS notation this function is defined as

$$X(W, \Delta) := \{X \mid B(X)X = VX\} \tag{2}$$

where

$$B(X) := \sum_{1\leq i<j\leq n}\sum \left\{w_{ij}\frac{\delta_{ij}}{d_{ij}(X)}A_{ij} \mid d_{ij}(X) > 0\right\}, \tag{3}$$

$$V := \sum_{1\leq i<j\leq n}\sum w_{ij}A_{ij}, \tag{4}$$

and $A_{ij}$ is defined, using unit vectors $e_i$ and $e_j$, as

$$A_{ij} := (e_i - e_j)(e_i - e_j)'. \tag{5}$$

Again the map $X()$ is non-linear and set-valued, and usually it cannot be computed completely and effectively. Actual MDS applications find a single $X \in X(W, \Delta)$ by some iterative MDS procedure such as `smacof` (De Leeuw and Mair (2009)), or a number of different $X$ by using different starting points for the iterations.

In this paper we study the mapping $X$ by analyzing its partial inverse. More precisely, in *inverse multidimensional scaling*, abbreviated as *iMDS*, we start with $W$ and $X$. We find the set of all $\Delta$ such that $X \in X(W, \Delta)$. In a formula,

$$\Delta(W, X) := \{\Delta \mid B(X)X = VX\}. \tag{6}$$

iMDS is not particularly important in practical MDS applications, but it allows us to study some theoretical aspects of MDS, such as the existence and multiplicity of local minima.

Note that instead of (6) we could have defined

$$\Delta(X) := \{(\Delta, W) \mid B(X)X = VX\}. \tag{7}$$

This is obviously a related map, and many of our results can be applied to this map as well. The map (6) is slightly simpler, and analyzing (7) mostly leads to uninteresting complications in the notation and the formulation of the results.

# 2 Inverse MDS

## 2.1 Basics

In studying the iMDS mapping we limit ourselves, unless explicitly stated otherwise, to configurations $X$ that are *regular*, in the sense that $d_{ij}(X) > 0$ for all $i \neq j$. This can be done without loss of generality. If some of the distances are zero, then the corresponding iMDS problem can be reduced to a regular problem with a smaller number of points (De Leeuw, Groenen, and Mair (2016c)). Also, an $n \times p$ configuration $X$ is *normalized* if it is column-centered and has rank $p$. For such $X$ there exist $n \times (n-p-1)$ centered orthonormal matrix $K$ such that $K'X = 0$. In iMDS we will always assume that $X$ is both regular and normalized. We also assume, unless it is explicitly stated otherwise, that all off-diagonal weights $w_{ij}$ are non-zero.

We use $A \circ B$ for the elementwise or Hadamard product of two matrices or vectors. The elementwise generalized inverse of $A$ is $A^\dagger$, with

$$a_{ij}^\dagger = \begin{cases} a_{ij}^{-1} & \text{if } a_{ij} \neq 0, \\ 0 & \text{if } a_{ij} = 0. \end{cases}$$

Note that all binary matrices satisfy $A^\dagger = A$. $E$ is the elementwise complement of the identity matrix, i.e. a matrix with all elements equal to one, except for the diagonal, which is zero. And $e$ is a vector with all elements equal to one.

Our code also has the two `R` functions `lower_triangle()` and `fill_symmetric()`, which correspond to the two linear operators **low**() and **fill**(). If $A$ is symmetric of order $n$, then **low**$(A)$ are the elements of $A$ below the diagonal ravelled columnwise in a vector with $\frac{1}{2}n(n-1)$ elements, and **fill**() is the inverse of **low**(). Thus if $A$ is

```
##      [,1] [,2] [,3]
## [1,]    0    6   10
## [2,]    6    0   14
## [3,]   10   14    0
```

then **low**$(A)$ is

```
lower_triangle(a)
```

```
## [1]  6 10 14
```

and **fill**(**low**$(A)$) reproduces $A$

```
fill_symmetric(lower_triangle(a))
```

**Lemma 1:** [**Matrix**] Suppose $X$ is an $n \times p$ matrix or rank $p$. Suppose $K$ is an $n \times (n-p)$ orthonormal basis for the null space of $X$. Then the symmetric matrix $A$ satisfies $AX = 0$ if and only if there is a symmetric $S$ such that $A = KSK'$.

**Proof:** Suppose $X = LS$ with $L$ an orthonormal basis for the column space of $X$ and $S$ non-singular. Write $A$ as

$$A = \begin{bmatrix} L & K \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} L' \\ K' \end{bmatrix}.$$

Then $AX = LA_{11}S + KA_{21}S = 0$ if and only if $A_{11} = 0$ and $A_{21} = 0$, and by symmetry $A_{12} = 0$. Thus $A = KA_{22}K'$. **QED**

**Theorem 1:** [**Key**] $\Delta \in \Delta(W, X)$ if and only if there is a symmetric $S$ of order $n - p - 1$ such that for all $i \neq j$

$$\Delta = D(X) \circ (E - W^\dagger \circ KSK'). \tag{8}$$

Thus $\Delta(W, X)$ is a non-empty affine set, closed under linear combinations with coefficients that add up to one.

**Proof:** By lemma 1 we have $(V - B(X))X = 0$ if and only if there is a symmetric $S$ such that $V - B(X) = KSK'$. This can be rearranged to yield (8). Note that always $D(X) \in \Delta(W, X)$. **QED**

Note that theorem 1 implies that if $\Delta_1$ and $\Delta_2$ are in $\Delta(W,X)$, then so is the whole line through $\Delta_1$ and $\Delta_2$. Specifically, if we compute a solution to the stationary equations $X$ with some MDS algorithm such as `smacof`, then the line through the data $\Delta$ and $D(X)$ is in $\Delta(X,W)$. Of course $X \in X(W,\Delta)$ if and only if $\Delta \in \Delta(W,X)$.

The next result is corollary 6.3 in "Inverse Multidimensional Scaling" (2007).

**Corollary 1:** [**Full**] If $p = n-1$ then $\Delta \in \Delta(W,X)$ if and only if $W \circ \Delta = W \circ D(X)$ if and only if $\sigma(X,W,\Delta) = 0$.

**Proof:** If $p = n-1$ then $S$ in theorem 1 is of order zero. **QED**

For any two elements of $\Delta(X,W)$, one cannot be elementwise larger (or smaller) than the other. This is corollary 3.3 in "Inverse Multidimensional Scaling" (2007).

**Corollary 2:** [**Dominate**] If $\Delta_1$ and $\Delta_2$ are both in $\Delta(X,W)$ and $\Delta_1 \leq \Delta_2$, then $W \circ \Delta_1 = W \circ \Delta_2$.

**Proof:** With obvious notation $\mathbf{tr}\ X'(B_1(X) - B_2(X))X = 0$, which can be written as

$$\sum_{1\leq i<j\leq n}\sum w_{ij}\left(\delta_{1ij} - \delta_{2ij}\right) d_{ij}(X) = 0,$$

which implies $W \circ \Delta_1 = W \circ \Delta_2$. **QED**

We can also compare the various elements of $\Delta(X,W)$ by norm, or, equivalently, by stress value. Define, for hollow symmetric matrices $H$,

$$\|H\|_W := \sqrt{\sum_{1\leq i<j\leq n}\sum w_{ij}h_{ij}^2}.$$

**Theorem 2:** [**Norm**] If $\Delta \in \Delta(X,W)$ then $\sigma(X,W,\Delta) = \|\Delta\|_W^2 - \|D(X)\|_W^2$. Thus $\|\Delta\|_W^2 \geq \|D(X)\|_W^2$.

**Proof:** We have $\sigma(X,W,\Delta) = \|\Delta\|_W^2 + \|D(X)\|_W^2 - 2\mathbf{tr}\ X'B(X)X$. But if $\Delta \in \Delta(X,W)$ we have $\mathbf{tr}\ X'B(X)X = \|D(X)\|_W^2$. **QED**

## 2.2 Non-negative Dissimilarities

There are, of course, affine combinations $\Delta$ with negative elements. We could decide that we are only interested in non-negative dissimilarities. In order to deal with non-negativity we define $\Delta_+$ as the polyhedral convex cone of all symmetric, hollow, and non-negative matrices.

**Theorem 3:** [**Positive**] We have $\Delta \in \Delta(W,X) \cap \Delta_+$ if and only if there is a symmetric $S$ such that (8) holds and such that

$$\mathbf{low}\ (KSK') \leq \mathbf{low}\ (W). \tag{9}$$

Thus $\Delta(W,X) \cap \Delta_+$ is a convex polyhedron, closed under non-negative linear combinations with coefficients that add up to one.

**Proof:** Follows easily from the representation in theorem 1. **QED**

Of course the minimum of $\sigma(X, W, \Delta)$ over $\Delta \in \Delta(W, X) \cap \Delta_+$ is zero, attained at $D(X)$. The maximum of stress, which is a convex quadratic in $\Delta$, is attained at one of the vertices of $\Delta(W, X) \cap \Delta_+$.

**Theorem 4:** **[Bounded]** $\Delta(W, X) \cap \Delta_+$ is bounded, i.e. it is a convex polygon.

**Proof:** This is corollary 3.2 in "Inverse Multidimensional Scaling" (2007), but the proof given there is incorrect. A hopefully correct proof goes as follows. A polyhedron is bounded if and only if its recession cone is the zero vector. If the polyhedron is defined by $Ax \leq b$ then the recession cone is the solution set of $Ax \leq 0$. Thus, in our case, the recession cone consists of all matrices $S$ for which $\textbf{low}\ (KSK') \leq 0$. Since $U := KSK'$ is doubly-centered, and $K$ is orthogonal to $X$, we have

$$0 = \textbf{tr}\ X'UX = 2 \sum_{1 \leq i < j \leq n} \sum u_{ij} d_{ij}^2(X).$$

This implies $U = 0$, and the recession cone is the zero vector. **QED**

We can compute the vertices of $\Delta(W, X) \cap \Delta_+$ using the complete description method of Fukuda (2015), with an `R` implementation in the `rcdd` package by Geyer and Meeden (2015). Alternatively, as a check on our computations, we also use the `lrs` method of Avis (2015), with an `R` implementation in the `vertexenum` package by Robere (2015). Both methods convert the *H-representation* of the polygon, as the solution set of a number of linear inequalities, to the *V-representation*, as the convex combinations of a number of vertices.

There is also a brute-force method of converting H to V that is somewhat wasteful, but still practical for small examples. Start with the H-representation $Ax \leq b$, where $A$ is $n \times m$ with $n \geq m$. Then look at all $\binom{n}{m}$ choices of $m$ rows of $A$. Each choice partitions $A$ into the $m \times m$ matrix $A_1$ and the $(n-m) \times m$ matrix $A_2$ and $b$ into $B - 1$ and $b_2$. If $rank(A_1) < m$ there is no extreme point associated with this partitioning. If $rank(A_1) = m$ we compute $\hat{v} = A_1^{-1} b_1$ and if $A_2 \hat{v} \leq b_2$ then we add $\hat{v}$ to the V representation.

In our `R` implementation we use a basis for the symmetric matrices or order $m := n - p - 1$ consisting of the $m$ matrices $e_i e_i'$ and the $\frac{1}{2}m(m-1)$ matrices $(e_i e_j' + e_j e_i')$. We collect the $m$ vectors $\textbf{low}(k_i k_i')$ and the $\frac{1}{2}m(m-1)$ vectors $\textbf{low}(k_i k_j' + k_j k_i')$ as columns in the $\frac{1}{2}n(n-1)$ by $\frac{1}{2}m(m+1)$ matrix $G$. Then $\textbf{low}(KSK')$ is of the form $Gt$ for some $t$, and we must have $\delta = d(X) \circ (e - w^\dagger \circ Gt)$, where $\delta := \textbf{low}(\Delta)$, $d(X) := \textbf{low}(D(X))$, and $w := \textbf{low}(W)$. The non-negativity constraint is simply $Gt \leq w$.

## 2.3 Zero Weights and/or Distances

If a distance is zero then the corresponding element of $B(X)$ must be zero as well. If a weight is zero, then the corresponding elements of both $V$ and $B(X)$ are zero. It is still true that $(V - B)X = 0$ if and only if there is an $S$ such that $B = V - KSK'$, but it may no longer be possible to find the $\Delta$ corresponding with some $V + KSK'$. In other words, not all solutions $B$ to $(V - B)X = 0$ correspond with a proper $B(X)$. Specifically, zero weights and/or distances imply that one or more elements of $B$ are required to be zero. If these zero requirements are taken into account then not all matrices $S$ are allowed.

If, for example, $X$ is

```
##      [,1]
## [1,] -0.5
## [2,] -0.5
## [3,]  0.5
## [4,]  0.5
```

and the weights are all one, then $V - B$ must be a linear combination of the three matrices, say $P_{11}, P_{22}$ and $P_{12}$,

```
##      [,1] [,2] [,3] [,4]
## [1,]    1   -1    1   -1
## [2,]   -1    1   -1    1
## [3,]    1   -1    1   -1
## [4,]   -1    1   -1    1
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1   -1   -1    1
## [2,]   -1    1    1   -1
## [3,]   -1    1    1   -1
## [4,]    1   -1   -1    1
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   -2    2    0    0
## [2,]    2   -2    0    0
## [3,]    0    0    2   -2
## [4,]    0    0   -2    2
```

and $B$ is $V$ minus the linear combination. For any $B$ computed this way we have $BX = VX$, but we have $b_{12} = b_{34} = 0$ if and only if $B = V - \alpha P_{11} + (1 - \alpha)P_{22}$.

# 3 Examples

## 3.1 First Example

As our first example we take $X$ equal to four points in the corners of a square. This example is also used in "Inverse Multidimensional Scaling" (2007) and De Leeuw (2012). Here $X$ is

```
##      [,1] [,2]
## [1,] -0.5 -0.5
## [2,] -0.5  0.5
## [3,]  0.5  0.5
## [4,]  0.5 -0.5
```

with distances

```
##          1        2        3
## 2 1.000000
## 3 1.414214 1.000000
## 4 1.000000 1.414214 1.000000
```

and $K$ is the vector

```
## [1] -0.5  0.5 -0.5  0.5
```

For unit weights we have $\Delta \in \Delta(X, W)$ if and only if $\Delta = D(X)\{W - \lambda kk'\}$ for some real $\lambda$. This means that $\Delta \in \Delta(X, W) \cap \Delta_+$ if and only if $-4 \leq \lambda \leq 4$. The endpoints of this interval correspond with the two dissimilarity matrices

$$\Delta_1 := 2\sqrt{2}\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},$$

and

$$\Delta_2 := 2\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}.$$

Thus $\Delta(X, W) \cap \Delta_+$ are the convex combinations

$$\Delta(\alpha) := \alpha\Delta_1 + (1-\alpha)\Delta_2 = \begin{bmatrix} 0 & 2(1-\alpha) & 2\alpha\sqrt{2} & 2(1-\alpha) \\ 2(1-\alpha) & 0 & 2(1-\alpha) & 2\alpha\sqrt{2} \\ 2\alpha\sqrt{2} & 2(1-\alpha) & 0 & 2(1-\alpha) \\ 2(1-\alpha) & 2\alpha\sqrt{2} & 2(1-\alpha) & 0 \end{bmatrix}.$$

This can be thought of as the distances between points on a (generally non-Euclidean) square with sides $2(1-\alpha)$ and diagonal $2\alpha\sqrt{2}$. The triangle inequalities are satisfied if the length of the diagonal is less than twice the length of the sides, i.e. if $\alpha \leq \frac{2}{2+\sqrt{2}} \approx .585786$.

The distances are certainly Euclidean if Pythagoras is satisfied, i.e. if the square of the length of the diagonal is twice the square of the length of the sides. This gives $\alpha = \frac{1}{2}$, for which $\Delta = D(X)$. For a more precise analysis, observe that the two binary matrices, say $E_1$ and $E_2$, in the definition of $\Delta_1$ and $\Delta_2$ commute, and are both diagonalized by

$$L := \begin{bmatrix} \frac{1}{2} & \frac{1}{2}\sqrt{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2}\sqrt{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2}\sqrt{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & -\frac{1}{2}\sqrt{2} & -\frac{1}{2} \end{bmatrix}$$

The diagonal elements of $L'E_1L$ are $1, -1, -1, 1$ and those of $L'E_2L$ are $2, 0, 0, -2$. Because $L$ diagonalizes $E_1$ and $E_2$, it also diagonalizes $\Delta_1$ and $\Delta_2$, as well as the elementwise squares $\Delta_1^2$ and $\Delta_2^2$. And consequently also the Torgerson transform $-\frac{1}{2}J\Delta^2(\alpha)J$, which has eigenvalues $0, 4\alpha^2, 4\alpha^2, 4(1-2\alpha)$. All eigenvalues are non-negative for $\alpha \leq \frac{1}{2}$.

We see that $\Delta(\alpha)$ is two-dimensional Euclidean for $\alpha = \frac{1}{2}$, one-dimensional Euclidean for $\alpha = 0$, and three-dimensional Euclidean for $0 < \alpha < \frac{1}{2}$. In particular for $\alpha = \frac{1}{1+\sqrt{2}}$ all dissimilarities are equal and $\Delta(\alpha)$ is the distance matrix of a regular simplex.

On the unit interval stress is the quadratic $32(\alpha - \frac{1}{2})^2$, which attains its maximum equal to 8 at the endpoints.

## 3.2 Second Example

We next give another small example with four equally spaced points on the line, normalized to sum of squares one, and unit weights. Thus $X$ is

```
##            [,1]
## [1,] -0.6708204
## [2,] -0.2236068
## [3,]  0.2236068
## [4,]  0.6708204
```

and $D(X)$ is

```
##          1        2        3
## 2 3.464102
## 3 4.472136 2.828427
## 4 6.324555 4.472136 3.464102
```

For $S$ we use the basis $\begin{bmatrix}1&0\\0&0\end{bmatrix}$, $\begin{bmatrix}0&0\\0&1\end{bmatrix}$, and $\begin{bmatrix}0&1\\1&0\end{bmatrix}$. Thus $\Delta(X, W)$ are the affine linear combinations of $D(X)$ and the three matrices

```
##          1        2        3
## 2 4.330127
## 3 5.590170 2.121320
## 4 4.743416 5.590170 4.330127
```

```
##          1        2        3
## 2 3.983717
## 3 3.801316 4.101219
## 4 6.640783 3.801316 3.983717
```

```
##          1        2        3
## 2 1.914908
## 3 5.472136 2.828427
## 4 6.324555 3.472136 5.013295
```

Both `scdd()` from `rcdd` and `enumerate.vertices()` from `vertexenum` find the same seven vertices. All non-negative dissimilarity matrices for which $x$ is stationary are convex combinations of these seven matrices.

```
##           1         2         3
## 2 20.784610
## 3  0.000000  5.656854
## 4  0.000000 13.416408  0.000000
##           1         2         3
## 2 13.856406
## 3  4.472136  0.000000
## 4  0.000000 13.416408  0.000000
##          1        2        3
## 2 20.78461
## 3  0.00000 22.62742
## 4  0.00000  0.00000 20.78461
##           1         2         3
## 2  0.000000
## 3 13.416408  5.656854
## 4  0.000000  0.000000 20.784610
##           1         2         3
## 2  0.000000
## 3 13.416408  0.000000
## 4  0.000000  4.472136 13.856406
##          1        2        3
## 2 0.000000
## 3 4.472136 0.000000
## 4 8.432738 4.472136 0.000000
##           1         2         3
## 2  0.000000
## 3  0.000000  5.656854
## 4 12.649111  0.000000  0.000000
```

The stress values for these seven vertices are

```
## [1]  460.00000  248.00000 1072.00000  460.00000  248.00000   36.44444  112.00000
```

and we know that 1072 is the maximum of stress over $\Delta \in \Delta(X, W) \cap \Delta_+$.

In general the vanishing of the stationary equations does not imply that $X$ corresponds with a local minimum. It can also give a local maximum or a saddle point. We know, however, that the only local maximum of stress is the origin (De Leeuw, Groenen, and Mair (2016a)), and that in the one-dimensional case all solutions of the stationary equations that do not have tied coordinates are local minima (De Leeuw (2005)).

## 3.3 Third Example

The number of extreme points of the polytope $\Delta(W, X) \cap \Delta_+$ grows very quickly if the problem becomes larger. In our next example we take six points equally spaced on the unit sphere. Due to the intricacies of floating point comparisons (testing for zero, testing for equality) it can be difficult to determine exactly how many extreme points there are.

"Inverse Multidimensional Scaling" (2007) analyzed this example and found 42 extreme points. We repeat their analysis with our `R` function `bruteForce()` from the code section. We select $\binom{15}{6} = 5005$ sets of six linear equations from our 15 linear inequalities, test them for non-singularity, and solve them to see if they satisfy the remaining nine inequalities. This gives 1394 extreme points of the polytope, but many of them are duplicates. We use our function `cleanUp()` to remove duplicates, which leaves 42 vertices, same number as found by "Inverse Multidimensional Scaling" (2007). The 42 stress values are

```
##  [1] 24.00000 24.00000 24.00000 21.75000 12.00000 13.33333  6.66667 13.33333
##  [9] 17.33333 17.33333 19.68000 13.33333  6.66667 17.33333  6.66667 21.75000
## [17]  6.66667 60.00000 24.00000 21.75000 19.68000 17.33333 21.75000 13.33333
## [25] 19.68000 17.33333 17.33333 21.75000 17.33333 17.33333 13.33333  6.66667
## [33] 21.75000 17.33333 19.68000 13.33333 17.33333 19.68000 19.68000 17.33333
## [41]  6.66667 17.33333
```

and their maximum is 60.

If we perform the calculations more efficiently in `rcdd`, using rational arithmetic, we come up with a list of 153 extreme points. Using `cleanUp()` to remove what seem to be duplicates leaves 43. The `vertexenum` package, which uses a different conversion from float to rational and back, finds 147 extreme points, and removing what seem to be duplicates leaves 51.

The fact that we get different numbers of vertices with different methods is somewhat disconcerting. We test the vertices found by `rcdd` and `vertexenum` that are not found with the brute force method by using our function `rankTest()`. This test is based on the fact that a vector $x$ satisfying the $n \times m$ system $Ax \leq b$ is an extreme point if and only if matrix with all rows $a_i$ for which $a_i'x = b_i$ is of rank $m$. It turns out all the additional vertices found by `rcdd` and `vertexenum` do not satisfy this rank test, because the matrix of active constraints (satisfied as equalities) is of rank 5.

## 3.4 Fourth Example

This is an unfolding example with $n = 3 + 3$ points, configuration

```
##             [,1]       [,2]
## [1,]  0.0000000  0.0000000
## [2,]  0.0000000  0.0000000
## [3,]  0.0000000 -0.8164966
## [4,]  0.0000000  0.4082483
## [5,]  0.7071068  0.0000000
## [6,] -0.7071068  0.4082483
```

and weight matrix

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    0    0    0    1    1    1
## [2,]    0    0    0    1    1    1
## [3,]    0    0    0    1    1    1
## [4,]    1    1    1    0    0    0
## [5,]    1    1    1    0    0    0
## [6,]    1    1    1    0    0    0
```

Note that row-points one and two in $X$ are equal, and thus $d_{12}(X) = 0$. The example has both zero weights and zero distances. We now require that $(V - B(X))X = 0$, but also that the elements of $B(X)$ in the two $3 \times 3$ principal submatrices corresponding with the rows and columns are zero. This means that for the elements of $B(X)$ we require $k_i'Sk_j \leq 1$ for $w_{ij} = 1$ and both $k_i'Sk_j \leq 0$ and $-k_i'Sk_j \leq 0$ for $w_{ij} = 0$. We can then solve for edges of the off-diagonal block of dissimilarities. There are no constraints on the dissimilarities in the diagonal blocks, because they are not part of the MDS problem.

Using our brute force method, we find the two edges

```
##           4        5        6
## 1 0.0000000 1.414214 1.632993
## 2 0.8164966 0.000000 0.000000
## 3 1.2247449 1.080123 1.414214
```

```
##           4        5        6
## 1 0.8164966 0.000000 0.000000
## 2 0.0000000 1.414214 1.632993
## 3 1.2247449 1.080123 1.414214
```

and the off-diagonal blocks for which $X$ is an unfolding solution are convex combinations of these two.

# 4 MDS Sensitivity

Suppose $X$ is a solution to the MDS problem with dissimilarities $\Delta$ and weights $W$, found by some iterative algorithm such as `smacof`. We can then compute $\Delta(W, X) \cap \Delta_+$, which is a convex neighborhood of the data $\Delta$, and consists of all non-negative dissimilarity matrices that have $X$ as a solution to the stationary equations. The size of this convex neighborhood can be thought of as a measure of *stability* or *sensitivity*.

For typical MDS examples there is no hope of computing all vertices of $\Delta(X, W) \cap \Delta_+$. Consider the data from De Gruijter (1967), for example, with $n = 9$ objects, to be scaled in $p = 2$ dimensions. We have $\frac{1}{2}n(n-1) = 36$ dissimilarities, and because $m = n - p - 1 = 6$ there are $\frac{1}{2}m(m+1) = 21$ variables. It suffices to consider that there are 5567902560 ways in which we can pick 21 rows from 36 rows to understand the number of potential vertices.

What we can do, however, is to optimize linear (or quadratic functions) over $\Delta(X, W) \cap \Delta_+$, because 36 linear inequalities in 21 variables define an easily manageable LP (or QP) problem. As an example, not necessarily a very sensible one, we solve 36 linear programs to maximize and minimize each of the $\delta_{ij}$ in $\Delta(X, W) \cap \Delta_+$ separately. We use the `lpSolve` package (Berkelaar, M. and others (2015)), and collect the maximum and minimum $\delta_{ij}$ in a matrix. The range from the smallest possible $\delta_{ij}$ to the largest possible $\delta_{ij}$ turns out to be quite large.

The data are

```
##       KVP PvdA  VVD  ARP  CHU  CPN  PSP   BP  D66
## KVP  0.00 5.63 5.27 4.60 4.80 7.54 6.73 7.18 6.17
## PvdA 5.63 0.00 6.72 5.64 6.22 5.12 4.59 7.22 5.47
## VVD  5.27 6.72 0.00 5.46 4.97 8.13 7.55 6.90 4.67
## ARP  4.60 5.64 5.46 0.00 3.20 7.84 6.73 7.28 6.13
## CHU  4.80 6.22 4.97 3.20 0.00 7.80 7.08 6.96 6.04
## CPN  7.54 5.12 8.13 7.84 7.80 0.00 4.08 6.34 7.42
## PSP  6.73 4.59 7.55 6.73 7.08 4.08 0.00 6.88 6.36
## BP   7.18 7.22 6.90 7.28 6.96 6.34 6.88 0.00 7.36
## D66  6.17 5.47 4.67 6.13 6.04 7.42 6.36 7.36 0.00
```

The optimal configuration found by `smacof` is

```
##        [,1]   [,2]
##  [1,]  1.78  3.579
##  [2,] -1.46  2.297
##  [3,]  3.42 -2.776
##  [4,]  3.30  1.837
##  [5,]  3.84  0.308
##  [6,] -5.09 -0.044
##  [7,] -3.79  2.132
##  [8,] -2.86 -4.318
##  [9,]  0.86 -3.015
```

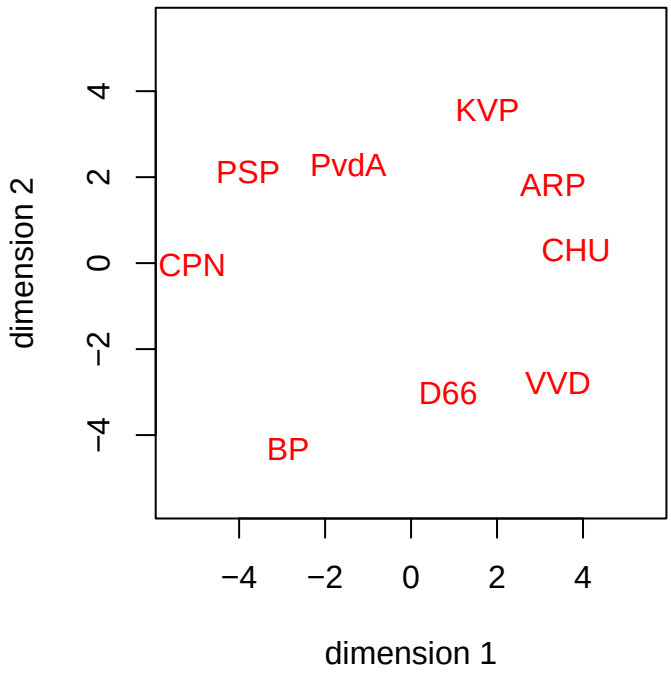


Figure 1: De Gruijter configuration

The maximum and minimum dissimilarities in $\Delta(X, W) \cap \Delta_+$ are

```
##       KVP   PvdA  VVD   ARP   CHU   CPN   PSP   BP    D66
## KVP    0.0  32.6   9.7   9.3  24.6  24.3  11.7  18.9  15.2
## PvdA  32.6   0.0  25.7  36.9  20.7  34.1  36.1  29.7  22.2
## VVD    9.7  25.7   0.0  17.4  32.0  34.4  22.3  23.5  20.4
## ARP    9.3  36.9  17.4   0.0  28.0  33.7  25.6  28.9  23.0
## CHU   24.6  20.7  32.0  28.0   0.0  20.8  31.9  33.3  26.3
## CPN   24.3  34.1  34.4  33.7  20.8   0.0  24.9  30.7  31.8
## PSP   11.7  36.1  22.3  25.6  31.9  24.9   0.0  23.7  27.7
## BP    18.9  29.7  23.5  28.9  33.3  30.7  23.7   0.0  35.0
## D66   15.2  22.2  20.4  23.0  26.3  31.8  27.7  35.0   0.0
```

```
##       KVP   PvdA  VVD   ARP   CHU   CPN   PSP   BP    D66
## KVP   0.00  3.48  0.00  0.00  2.34  0.00  0.00  0.00  0.00
## PvdA  3.48  0.00  0.00  0.00  0.00  0.00  0.93  0.00  0.00
## VVD   0.00  0.00  0.00  0.00  0.83  0.00  0.00  0.00  0.00
## ARP   0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
## CHU   2.34  0.00  0.83  0.00  0.00  0.00  0.00  0.00  0.00
## CPN   0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
## PSP   0.00  0.93  0.00  0.00  0.00  0.00  0.00  0.00  0.00
## BP    0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
## D66   0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
```

# 5 Second Order Inverse MDS

As we mentioned before, stationary values are not local minima. It is necessary for a local minimum that the stationary equations are satisfied, but it is also necessary that at a solution of the stationary equations the Hessian is positive semi-definite. Thus it becomes interesting to find the dissimilarities in $\Delta(W,X)\cap\Delta_+$ for which the Hessian is positive semi-definite. Define

$$\Delta_H(W,X):=\{\Delta\mid\mathcal{D}^2\sigma(X)\gtrsim 0\}.$$

We can now study $\Delta(W,X)\cap\Delta_H(W,X)$ or $\Delta(W,X)\cap\Delta_H(W,X)\cap\Delta_+$.

**Theorem 5: [Hessian]** $\Delta_H(W,X)$ is a convex non-polyhedral set.

**Proof:** In MDS the Hessian is $2(V-H(x,W,\Delta))$, where

$$H(x,W,\Delta):=\sum_{1\leq i<j\leq n}\sum w_{ij}\frac{\delta_{ij}}{d_{ij}(x)}\left\{A_{ij}-\frac{A_{ij}xx'A_{ij}}{x'A_{ij}x}\right\}. \tag{10}$$

Here we use $x:=\mathbf{vec}(X)$, and both $A_{ij}$ and $V$ are direct sums of $p$ copies of our previous $A_{ij}$ and $V$ (De Leeuw, Groenen, and Mair (2016b)). The Hessian is linear in $\Delta$, which means that requiring it to be positive semi-definite defines a convex non-polyhedral set. The convex set is defined by the infinite system of linear inequalities $y'H(x,W,\Delta)y\geq 0$. **QED**

In example 3 the smallest eigenvalues of the Hessian at the 42 vertices are

```
##  [1]   0.00000   0.00000   0.00000  -5.86238   0.00000  -2.96688   0.00000
##  [8]  -2.96688  -4.47894  -4.47894  -5.71312  -2.96688   0.00000  -5.00453
## [15]   0.00000  -5.86238   0.00000 -12.00000   0.00000  -5.86238  -5.71312
## [22]  -5.00453  -5.86238  -2.96688  -5.71312  -4.47894  -5.00453  -5.86238
## [29]  -5.00453  -5.00453  -2.96688   0.00000  -5.86238  -4.47894  -5.71312
## [36]  -2.96688  -5.00453  -5.71312  -5.71312  -4.47894   0.00000  -4.47894
```

and thus there are at most 11 vertices corresponding with local minima.

The next step is to refine the polyhedral approximation to $\Delta(W,X)\cap\Delta_H(W,X)\cap\Delta_+$ by using cutting planes. We add linear inequalities to the H-representation by using the eigenvectors corresponding to the smallest eigenvalues of all those vertices for which this smallest eigenvalue is negative. Thus, if the eigenvector is $y$, we would add the inequality

$$\sum_{1\leq i<j\leq n}\sum\zeta_{ij}(w_{ij}-k_i'Sk_j)\geq 0, \tag{11}$$

where

$$\zeta_{ij}:=y'\left(A_{ij}-\frac{A_{ij}xx'A_{ij}}{x'A_{ij}x}\right)y.$$

It is clear, however, that adding a substantial number of linear inequalities will inevitably lead to a very large number of potential extreme points. We proceed conservatively by cutting off only the solution with the smallest negative eigenvalue in computing the new V representation, using our function `bruteForceOne()`.

# 6 Inverse FDS

A configuration $X$ is a *full dimensional scaling* or *FDS* solution (De Leeuw, Groenen, and Mair (2016a)) if $(V - B(X))X = 0$ and $V - B(X)$ is positive semi-definite. In that case $X$ actually provides the global minimum of stress. The *inverse FDS* or *iFDS* problem is to find all $\Delta$ such that a given $X$ is the FDS solution.

**Theorem 6:** **[FDS]** $\Delta \in \Delta_F(W, X)$ if and only if there is a positive semi-definite $S$ such that for all $i \neq j$

$$\delta_{ij} = d_{ij}(X)(1 - \frac{1}{w_{ij}} k_i' S k_j), \tag{12}$$

Thus $\Delta_F(W, X)$ is a non-polyhedral convex set, closed under linear combinations with coefficients that add up to one.

**Proof:** Of course $\Delta_F(W, X) \subseteq \Delta(W, X)$. Thus $V - B(X) = KSK'$ for some $S$, and $XX'$ and $V - B(X)$ must both be positive semi-definite, with complementary null spaces. **QED**

We reanalyze our second example, with the four points equally spaced on the line, requiring a positive semi-definite $S$. We start with the original 7 vertices, for which the minimum eigenvalues of $S$ are

```
## [1] -4.000 -2.899 -1.708 -3.333  8.000 -1.708 -2.899
```

If the minimum eigenvalue is negative, with eigenvector $y$, we add the constraints $y'Sy \geq 0$. This leads to 11 vertices with minimum eigenvalues

```
##  [1]  8.0000 -0.0355  0.0000 -0.7911 -0.3834 -0.3834 -0.0355 -0.0853 -0.0853
## [10] -0.7911  0.0000
```

We see that the negative eigenvalues are getting smaller. Repeating the procedure of adding constraints based on negative eigenvalues three more times gives 19, 37, and 79 vertices, with corresponding minimum eigenvalues

```
##  [1]  8.000000 -0.000021  0.000000 -0.209852 -0.202322 -0.085642 -0.085642
##  [8] -0.000021 -0.106132 -0.018912 -0.008051 -0.023967 -0.008051 -0.023967
## [15] -0.106132 -0.018912 -0.209852 -0.202322  0.000000
```

```
##  [1]  8.000000 -0.000021  0.000000 -0.056872 -0.053562 -0.045307 -0.062028
##  [8] -0.020853 -0.020853 -0.000021 -0.028793 -0.004490 -0.001925 -0.006410
## [15] -0.001925 -0.006410 -0.028793 -0.004490 -0.000005 -0.002104 -0.021934
## [22] -0.024355 -0.021934 -0.024355 -0.000005 -0.002104 -0.004975 -0.005596
## [29] -0.005596 -0.004975 -0.002318 -0.002318 -0.056872 -0.053562 -0.045307
## [36] -0.062028  0.000000
```

```
##  [1]  8.000000 -0.000021  0.000000 -0.015026 -0.013698 -0.010861 -0.017838
##  [8] -0.013830 -0.013426 -0.012084 -0.013998 -0.018140 -0.005180 -0.005180
## [15] -0.000021 -0.007568 -0.001096 -0.000471 -0.001662 -0.000471 -0.001662
## [22] -0.007568 -0.001096 -0.000005 -0.000001 -0.000538 -0.000488 -0.005588
## [29] -0.005886 -0.005588 -0.005886 -0.000005 -0.000001 -0.000538 -0.000488
## [36] -0.001278 -0.001355 -0.001355 -0.001278 -0.000001 -0.000649 -0.000594
## [43] -0.005244 -0.005378 -0.005244 -0.005378 -0.000001 -0.000649 -0.000594
## [50] -0.006838 -0.006293 -0.001150 -0.001210 -0.000492 -0.000514 -0.000566
## [57] -0.001545 -0.001444 -0.000492 -0.000514 -0.000566 -0.001545 -0.001444
## [64] -0.006838 -0.006293 -0.001150 -0.001210 -0.000001 -0.000001 -0.015026
## [71] -0.013698 -0.010861 -0.017838 -0.013830 -0.013426 -0.012084 -0.013998
## [78] -0.018140  0.000000
```

It should be noted that the last step already takes an uncomfortable number of minutes to compute. Although the number of vertices goes up quickly, the diameter of the polygon (the maximum distance between two vertices) slowly goes down and will eventually converge to the diameter of $\Delta_F(W, X) \cap \Delta_+$. Diameters in subsequent steps are

```
## [1] 5.657 5.086 5.065 5.061 5.060
```

# 7 Multiple Solutions

If $X_1, \cdots, X_s$ are configurations with the same number of points, then the intersection $\{\bigcap_{r=1}^s \Delta(W, X_r)\} \cap \Delta_+$ is again a polygon, i.e. a closed and bounded convex polyhedron (which may be empty). If $\Delta$ is in this intersection then $X_1, \cdots, X_s$ are all solutions of the stationary equations for this $\Delta$ and $W$.

Let's look at the case of two configurations $X_1$ and $X_2$. We must find vectors $t_1$ and $t_2$ such that

$$d_1 \circ (e - w^\dagger \circ G_1 t_1) = d_2 \circ (e - w^\dagger \circ G_2 t_2).$$

If $H_1 := \mathbf{diag}(d_1 \circ w^\dagger) G_1$ and $H_2 := \mathbf{diag}(d_2 \circ w^\dagger) G_2$ then this can be written as

$$\begin{bmatrix} H_1 & -H_2 \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = d_1 - d_2$$

This is a system of linear equations in $t_1$ and $t_2$. If it is solvable we can intersect it with the convex sets $G_1 t_1 \leq w$ and $G_2 t_2 \leq w$ to find the non-negative dissimilarity matrices $\Delta$ for which both $X_1$ and $X_2$ are stationary.

```
## $delta1
##          [,1]
## [1,] 1.464102
## [2,] 1.464102
```

```
## [3,] 1.464102
## [4,] 1.464102
## [5,] 1.464102
## [6,] 1.464102
##
## $delta2
##          [,1]
## [1,] 1.464102
## [2,] 1.464102
## [3,] 1.464102
## [4,] 1.464102
## [5,] 1.464102
## [6,] 1.464102
##
## $res
## [1] 1.024137e-15
##
## $rank
## [1] 2
```

As a real simple example, suppose $X$ and $Y$ are four by two. They both have three points in the corners of an equilateral triangle, and one point in the centroid of the other three. In $X$ the fourth point is in the middle, in $Y$ the first point. The only solution to the linear equations is the matrix with all dissimilarities equal.

For a much more complicated example we can choose the De Gruijter data. We use `smacof` to find two stationary points in two dimensions. The matrices $G_1$ and $G_2$ are $36 \times 21$, and thus $H := (H_1 \mid -H_2)$ is $36 \times 42$. The solutions of the linear system $Ht = d_1 - d_2$ are of the form $t_0 - Lt$, with $t_0$ an arbitrary solution and $L$ a $42 \times 6$ basis for the space of $Ht = 0$. To find the non-negative solutions we can use the H representation $Lv \leq t_0$, and then compute the V representation, realizing of course that we can choose 6 rows from 42 rows in 5245786 ways.

# 8 Minimizing iStress

The iMDS approach can also be used to construct an alternative MDS loss function. We call it *iStress*, defined as

$$\sigma_\iota(X, W, \Delta) := \min_{\tilde{\Delta} \in \Delta(W,X) \cap \Delta_+} \sum\sum_{1 \leq i < j \leq n} w_{ij}(\delta_{ij} - \tilde{\delta}_{ij})^2.$$

Minimizing iStress means minimizing the projection distance between the observed dissimilarity matrix and the moving convex set of non-negative dissimilarity matrices for which $X$ satisfies the stationary equations. The convex set is moving, because it depends on $X$. For

each $X$ we have to solve the iMDS problem of finding $\Delta(X,W)\cap\Delta_+$, and then solve the quadratic programming problem that computes the projection.

**Theorem 7:** **[iStress]** $\min_X \sigma_\iota(X,W,\Delta)=0$ and the minimum is attained at all $X$ with $(V-B(X))X=0$.

**Proof:** If $X$ is a stationary point of stress then $\Delta\in\Delta(X,W)\cap\Delta_+$ and thus iStress is zero. Conversely, if iStress is zero then $\Delta\in\Delta(X,W)\cap\Delta_+$ and $X$ is a stationary point of stress. **QED**

Minimizing iStress may not be a actual practical MDS method, but it has some conceptual interest, because it provides another way of looking at the relationship of MDS and iMDS.

We use the De Gruijter data for an example of iStress minimization. We use `optim` from base `R`, and the `quadprog` package of Turlach and Weingessel (2013). Two different solutions are presented, the first with iterations starting at the classical MDS solution, the second starting at the `smacof` solution. In both cases iStress converges to zero, i.e. the configurations converge to a solution of the stationary equations for stress, and in the `smacof` case the initial solution already has stress equal to zero.

```
## initial  value 106.624678
## iter  10 value 0.205734
## iter  20 value 0.025163
## iter  30 value 0.018427
## iter  40 value 0.000116
## iter  50 value 0.000007
## iter  60 value 0.000000
## final  value 0.000000
## converged
```

```
## initial  value 0.000000
## iter  10 value 0.000000
## iter  20 value 0.000000
## final  value 0.000000
## converged
```

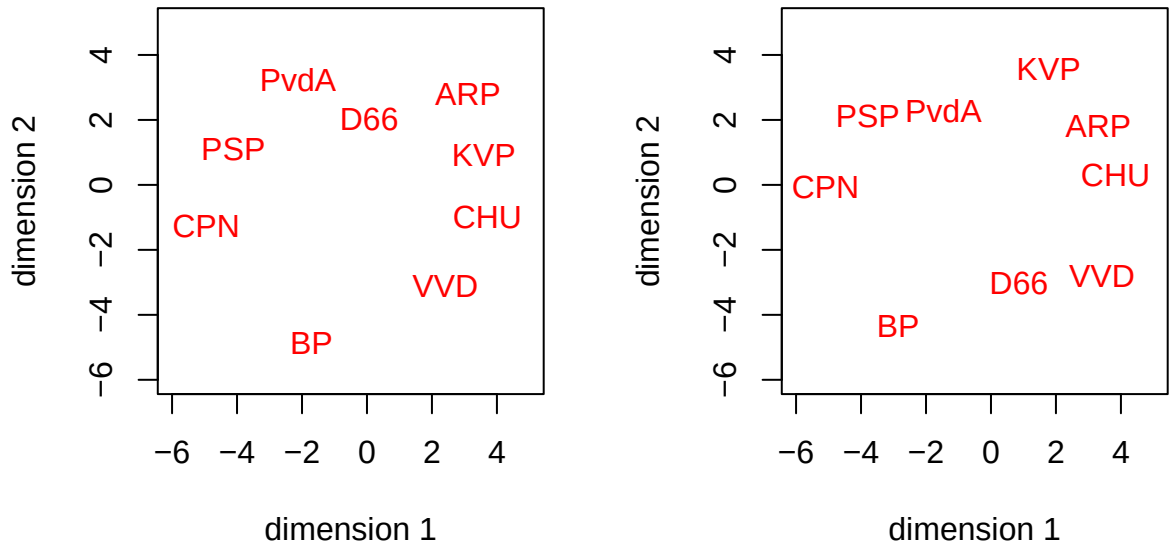


Figure 2: De Gruijter minimum iStress configuration

# 9 Appendix: Code

```
library (rcdd)
library (numDeriv)
library (vertexenum)
library (MASS)
library (lpSolve)
library (quadprog)

dyn.load("nextCombination.so")
dyn.load("cleanup.so")

inverseMDS <- function (x) {
  n <- nrow (x)
  m <- ncol (x)
  x <- apply (x, 2, function (y)
    y - mean (y))
  nm <- n - (m + 1)
  kk <- cbind (1, x, matrix (rnorm (n * nm), n , nm))
  kperp <- as.matrix (qr.Q (qr (kk))[, -(1:(m + 1))])
  dd <- Euclid (x)
  k <- 1
  base <- matrix (0, n * (n - 1) / 2, nm * (nm + 1) / 2)
  for (i in 1:nm) {
    for (j in 1:i) {
```

```
      oo <- outer (kperp[, i], kperp[, j])
      if (j != i) {
        oo <- oo + t(oo)
      }
      base[, k] <- lower_triangle (dd + (1 - oo))
      k <- k + 1
      print (c(i, j, k))
    }
  }
  return (base = cbind (lower_triangle (dd), base))
}

inversePlus <- function (base, affine = TRUE) {
  if (affine) {
    hrep <- makeH (
      a1 = d2q (-base),
      b1 = d2q (rep (0, nrow (base))),
      a2 = d2q (rep (1, ncol (base))),
      b2 = d2q (1)
    )
  } else {
    hrep <- makeH (a1 = d2q (-base), b1 = d2q (rep (0, nrow (base))))
  }
  vrep <- scdd (hrep)
  hrep <- q2d (hrep)
  vrep <- q2d (vrep$output)
  pr <- tcrossprod (hrep[,-c(1, 2)], vrep[,-c(1, 2)])[-1,]
  return (list (
    base = base,
    hrep = hrep,
    vrep = vrep,
    pr = pr
  ))
}

twoPoints <- function (x, y, w = 1 - diag (nrow (x))) {
  dx <- lower_triangle (as.matrix (dist (x)))
  dy <- lower_triangle (as.matrix (dist (y)))
  w <- lower_triangle (w)
  gx <- makeG (x)
  gy <- makeG (y)
  hx <- (dx / w) * gx
  hy <- (dy / w) * gy
  lxy <- lm.fit (cbind (hx, -hy), dx - dy)
```

```
  lxx <- lxy$coefficients[1:ncol(hx)]
  lyy <- lxy$coefficients[-(1:ncol(hx))]
  return (list(delta1 = dx - hx %*% lxx, delta2 = dy - hy %*% lyy, res = sum (abs(lxy$re
}

second_partials_stress <-
  function (x, delta, w = nonDiag (nrow (x))) {
    n <- nrow (x)
    p <- ncol (x)
    d <- Euclid (x)
    fac <- (w * delta) / (d + diag (n))
    dd <- d * d
    v <- vmat (w)
    deri <- direct_sum (repList (v, p))
    xx <- as.vector (x)
    for (i in 1:(n - 1)) {
      for (j in (i + 1):n) {
        aa <- direct_sum (repList (aijn (i, j, n), p))
        ax <- drop (aa %*% xx)
        deri <- deri - fac[i, j] * (aa - outer (ax, ax) / dd[i, j])
      }
    }
    return (deri)
  }

second_partials_numerical <-
  function (x, delta, w = nonDiag (nrow (x))) {
    stress <- function (x, delta, w) {
      n <- nrow (delta)
      p <- length (x) / n
      d <- Euclid (matrix (x, n, p))
      res <- delta - d
      return (sum (w * res * res) / 2)
    }
    return (hessian (stress, x, delta = delta, w = w))
  }

lower_triangle <- function (x) {
  n <- nrow (x)
  return (x[outer(1:n, 1:n, ">")])
}

fill_symmetric <- function (x) {
  m <- length (x)
```

```
  n <- (1 + sqrt (1 + 8 * m)) / 2
  d <- matrix (0, n, n)
  d[outer(1:n, 1:n, ">")] <- x
  return (d + t(d))
}

Euclid <- function (x) {
  c <- tcrossprod (x)
  d <- diag (c)
  return (sqrt (abs (outer (d, d, "+") - 2 * c)))
}

circular <- function (n) {
  x <- seq (0, 2 * pi, length = n + 1)
  z <- matrix (0, n + 1, 2)
  z[, 1] <- sin (x)
  z[, 2] <- cos (x)
  return (z[-1,])
}

direct_sum <- function (x) {
  n <- length (x)
  nr <- sapply (x, nrow)
  nc <- sapply (x, ncol)
  s <- matrix (0, sum (nr), sum (nc))
  k <- 0
  l <- 0
  for (j in 1:n) {
    s[k + (1:nr[j]), l + (1:nc [j])] <- x[[j]]
    k <- k + nr[j]
    l <- l + nc[j]
  }
  return (s)
}

ein <- function (i, n) {
  return (ifelse (i == 1:n, 1, 0))
}

aijn <- function (i, j, n) {
  d <- ein (i, n) - ein (j, n)
  return (outer (d, d))
}
```

```
nextCombination <- function (x, n) {
  m <- length (x)
  if (all (x == ((n - m) + 1:m))) return (NULL)
  z <- .C("nextCombination", as.integer(n), as.integer (m), as.integer(x))
  return (z[[3]])
}

repList <- function (x, n) {
  z <- list ()
  for (i in 1:n) {
    z <- c (z, list (x))
  }
  return (z)
}

nonDiag <- function (n) {
  return (matrix (1, n, n) - diag (n))
}

bmat <- function (x, delta, w = nonDiag (nrow (x))) {
  d <- Euclid (x)
  bmat <- -(w * delta) / (d + diag (nrow (x)))
  diag (bmat) <- -rowSums (bmat)
  return (bmat)
}

vmat <- function (w) {
  vmat <- -w
  diag (vmat) <- -rowSums (vmat)
  return (vmat)
}

torgerson <- function (delta, p = 2) {
  n <- nrow (delta)
  dd <- delta * delta
  sd <- rowSums (dd) / n
  ed <- sum (dd) / (n * n)
  cc <- -(dd - outer (sd, sd, "+") + ed) / 2
  ee <- eigen (cc)
  ea <- ee$values
  eb <- ifelse (ea > 0, sqrt (abs (ea)), 0)
  return (ee$vectors[, 1:p] %*% diag (eb[1:p]))
}
```

```
cleanUp <- function (a, eps = 1e-3) {
  nv <- nrow (a)
  ind <- rep (TRUE, nv)
  for (i in 1:(nv - 1)) {
    xx <- a[i,]
    for (j in (i + 1):nv) {
      if (!ind[j])
        next
      yy <- a[j,]
      mm <- max (abs (xx - yy))
      if (mm < eps)
        ind[j] <- FALSE
    }
  }
  return (ind)
}

bruteForce <- function (a, b, eps = 1e-3) {
  n <- nrow (a)
  m <- ncol (a)
  cb <- combn (n, m)
  n1 <- ncol (cb)
  ind <- rep(TRUE, n1)
  ht <- numeric()
  for (i in 1:n1) {
    gg <- a[cb[, i], ]
    bg <- b[cb[, i]]
    qg <- qr(gg)
    if (qg$rank < m) {
      ind[i] <- FALSE
      next
    }
    hh <- solve (qg, bg)
    hg <- drop (a %*% hh)
    if (min (b - hg) < -eps) {
      ind[i] <- FALSE
      next
    }
    ht <- c(ht, hh)
  }
  n2 <- sum (ind)
  ht <- matrix (ht, m, n2)
  ind <- .C ("cleanup", as.double(ht), as.integer(n2), as.integer(m), as.integer(rep(1,
  n3 <- sum (ind)
```

```
  return (list (
    x = t(ht)[which(ind == 1), ],
    n1 = n1,
    n2 = n2,
    n3 = n3
  ))
}

bruteForceOne <- function (a, b, p, q, v, eps = 1e-3) {
  n <- nrow (a)
  m <- ncol (a)
  ind <- which ((q - v %*% p) > -eps )
  v <- v[ind, ]
  cb <- combn (n, m - 1)
  n1 <- ncol (cb)
  ind <- rep(TRUE, n1)
  ht <- numeric()
  for (i in 1:n1) {
    gg <- rbind (a[cb[, i], ], p)
    bg <- c (b[cb[, i]], q)
    qg <- qr(gg)
    if (qg$rank < m) {
      ind[i] <- FALSE
      next
    }
    hh <- solve (qg, bg)
    hg <- drop (a %*% hh)
    if (min (b - hg) < -eps) {
      ind[i] <- FALSE
      next
    }
    ht <- c(ht, hh)
  }
  n2 <- sum (ind)
  ht <- t (matrix (ht, m, n2))
  ht <- rbind (v, ht)
  ind <- cleanUp (ht, eps)
  print (ind)
  n3 <- sum (ind)
  return (list (
    x = ht[ind, ],
    n1 = n1,
    n2 = n2,
    n3 = n3
```

```
  ))
}

rankTest <- function (x, a, b, eps = 1e-3) {
  h <- drop (a %*% x)
  ind <- which (abs (h - b) < eps)
  r <- qr (a[ind,])$rank
  f <- min (b - h) > -eps
  return (list (rank = r, feasibility = f))
}

makeDC <- function (x) {
  y <- -x
  diag(y) <- -rowSums (y)
  return (y)
}

bmat <- function (delta, w, d) {
  n <- nrow (w)
  dd <- ifelse (d == 0, 0, 1 / d)
  return (makeDC (w * delta * dd))
}

smacof <-
  function (delta,
            w,
            xini,
            eps = 1e-6,
            itmax = 100,
            verbose = TRUE) {
    n <- nrow (xini)
    xold <- xini
    dold <- as.matrix (dist (xold))
    sold <- sum (w * (delta - dold) ^ 2) / 2
    itel <- 1
    v <- ginv (makeDC (w))
    repeat {
      b <- bmat (delta, w, dold)
      xnew <- v %*% b %*% xold
      dnew <- as.matrix (dist (xnew))
      snew <- sum (w * (delta - dnew) ^ 2) / 2
      if (verbose) {
        cat (
          formatC (itel, width = 4, format = "d"),
```

```
          formatC (
            sold,
            digits = 10,
            width = 13,
            format = "f"
          ),
          formatC (
            snew,
            digits = 10,
            width = 13,
            format = "f"
          ),
          "\n"
        )
      }
      if ((itel == itmax) || (sold - snew) < eps)
        break ()
      itel <- itel + 1
      sold <- snew
      dold <- dnew
      xold <- xnew
    }
    return (list (
      x = xnew,
      d = dnew,
      s = snew,
      itel = itel
    ))
  }

oneMore <- function (g, u) {
  v <- bruteForce (g, u)$x
  nv <- nrow (v)
  s <- matrix (0, 2, 2)
  ev <- rep (0, nv)
  for (i in 1:nv) {
    s[1, 1] <- v[i, 1]
    s[2, 2] <- v[i, 2]
    s[1, 2] <- s[2, 1] <- v[i, 3]
    ee <- eigen (s)
    ev[i] <- min (ee$values)
    if (ev[i] < 0) {
      yy <- ee$vectors[, 2]
      hh <- c (yy[1] ^ 2, yy[2] ^ 2, 2 * yy[1] * yy [2])
```

```
      g <- rbind (g, -hh)
      u <- c (u, 0)
    }
  }
  return (list (v = v, g = g, u = u, e = ev))
}

makeG <- function (x) {
  n <- nrow (x)
  p <- ncol (x)
  m <- n - p - 1
  k <- qr.Q(qr(cbind(1, x, diag (n))))[, -c(1:(p + 1))]
  g <- matrix (0, n * (n - 1) / 2, m * (m + 1) / 2)
  l <- 1
  if (m == 1) {
    g[, 1] <- lower_triangle (outer (k, k))
  }
  else {
    for (i in 1:m) {
      g[, l] <- lower_triangle (outer(k[, i], k[, i]))
      l <- l + 1
    }
    for (i in 1:(m - 1))
      for (j in (i + 1):m) {
        g[, l] <- lower_triangle (outer(k[, i], k[, j]) + outer(k[, j], k[, i]))
        l <- l + 1
      }
  }
  return (g)
}

iStress <- function (x, delta, w = rep (1, length (delta)), only = TRUE) {
  m <- length (delta)
  n <- (1 + sqrt (1 + 8 * m)) / 2
  x <- matrix (x, n, length (x) / n)
  d <- lower_triangle (as.matrix (dist (x)))
  g <- makeG (x)
  h <- (d / w) * makeG (x)
  u <- - colSums(w * (delta - d) * h)
  v <- crossprod (h, w * h)
  s <- solve.QP (Dmat = v, dvec = u, Amat = -t(h), bvec = -d)
  ds <- d  - h %*% s$solution
  is <- sum (w * (delta - ds) ^ 2)
  if (only)
```

```
    return (is)
  else
    return (list (istress = is, delta = fill_symmetric (ds)))
}
```

# References


Avis, D. 2015. *User's Guide for lrs - Version 6.1.* The Computational Geometry Lab at McGill. http://cgm.cs.mcgill.ca/~avis/C/lrslib/USERGUIDE.html.

Berkelaar, M. and others. 2015. *lpSolve: Interface to 'Lp_solve' v. 5.5 to Solve Linear/Integer Programs.* %7Bhttps://CRAN.R-project.org/package=lpSolve%7D.

De Gruijter, D. N. M. 1967. "The Cognitive Structure of Dutch Political Parties in 1966." Report E019-67. Psychological Institute, University of Leiden.

De Leeuw, J. 2005. "Unidimensional Scaling." In *The Encyclopedia of Statistics in Behavioral Science*, edited by B. S. Everitt and D. C, 4:2095–97. New York, N.Y.: Wiley.

———. 2012. "Inverse Multidimensional Scaling." UCLA Department of Statistics.

De Leeuw, J., P. Groenen, and P. Mair. 2016a. "Full-Dimensional Scaling." 2016.

———. 2016b. "Second Derivatives of rStress, with Applications." 2016.

———. 2016c. "Singularities and Zero Distances in Multidimensional Scaling." 2016.

De Leeuw, J., and P. Mair. 2009. "Multidimensional Scaling Using Majorization: SMACOF in R." *Journal of Statistical Software* 31 (3): 1–30.

Fukuda, K. 2015. *cdd and cddplus Homepage.* https://www.inf.ethz.ch/personal/fukudak/cdd_home/.

Geyer, C. J., and G. D. Meeden. 2015. "rcdd: Computational Geometry." 2015. https://CRAN.R-project.org/package=rcdd.

"Inverse Multidimensional Scaling." 2007. *Journal of Classification* 14: 3–21.

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.

Robere, R. 2015. *vertexenum: Vertex Enumeration of Polytopes.* https://CRAN.R-project.org/package=vertexenum.

Turlach, B. A., and A. Weingessel. 2013. *quadprog: Functions to solve Quadratic Programming Problems.* https://CRAN.R-project.org/package=quadprog.