# Global Minima in Metric Multidimensional Scaling by Penalized Full-dimensional Scaling

Jan de Leeuw

First created on May 06, 2019. Last update on July 28, 2023

**Abstract**

The full-dimensional (metric, Euclidean, least squares) multidimensional scaling stress loss function is combined with a quadratic external penalty function that penalizes the higher dimensions. The trajectory of minimizers of stress for increasing values of the penalty parameter is then used to find (tentative) global minima for low-dimensional multidimensional scaling. This is illustrated with several one-dimensional and two-dimensional examples. We find solutions that are at least as good as those found with a traditional Torgerson start, and sometimes better, and in some of the one-dimensional examples we actually find the global minimum.

## Contents

**Note:** This is a working paper which will be expanded/updated frequently. All suggestions for improvement are welcome.

# 1 Introduction

Full-dimensional Scaling (FDS) was introduced by De Leeuw (1993). De Leeuw, Groenen, and Mair (2016) discuss it in some detail. In FDS we minimize the usual Multidimensional Scaling (MDS) least squares loss function first used by Kruskal (1964a) and Kruskal (1964b).

$$\sigma(Z) = \frac{1}{2}\sum_{1\leq i<j\leq n}\sum w_{ij}(\delta_{ij} - d_{ij}(Z))^2 \tag{1}$$

over all $n \times n$ *configuration matrices* $Z$. The loss at $Z$ is often called the *stress* of configuration $Z$. More generally we define pMDS as the problem of minimizing (1) over all $n \times p$ matrices. Thus FDS is the same as nMDS. If a configuration $Z$ has $n$ columns (i.e. is square) it is called a *full configuration.*

In (1) the matrices $W = \{w_{ij}\}$ and $\Delta = \{\delta_{ij}\}$ of *weights* and *dissimilarities* are non-negative, symmetric, and hollow. To simplify matters we suppose both $W$ and $\Delta$ have positive off-diagonal elements. The matrix $D(X) = \{d_{ij}(Z)\}$ has the *Euclidean distances* between the rows of the configuration $Z$. Thus

$$d_{ij}(Z) = \sqrt{(z_i - z_j)'(z_i - z_j)}.$$

We now introduce some standard MDS notation, following De Leeuw (1977). Define the matrix $V = \{v_{ij}\}$ by

$$v_{ij} = \begin{cases} -w_{ij} & \text{if } i \neq j, \\ \sum_{j=1}^{n} w_{ij} & \text{if } i = j, \end{cases} \tag{2}$$

and the matrix valued function $B(Z) = \{b_{ij}(Z)\}$ by

$$b_{ij}(Z) = \begin{cases} -w_{ij}e_{ij}(Z) & \text{if } i \neq j, \\ \sum_{j=1}^{n} w_{ij}e_{ij}(Z) & \text{if } i = j, \end{cases} \tag{3}$$

where $E(Z) = \{e_{ij}(Z)\}$ is defined as

$$e_{ij}(Z) = \begin{cases} \frac{\delta_{ij}}{d_{ij}(Z)} & \text{if } d_{ij}(Z) > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Note that $V$ and $B(Z)$ are both positive semi-definite and doubly-centered. Matrix $V$ has rank $n-1$. If all off-diagonal $d_{ij}(Z)$ are positive then $B(Z)$ has rank $n-1$ for all $Z$. Note that De Leeuw (1984) established that near a local minimum of stress all off-diagonal distances are indeed positive. The only vectors in the null-space of both $V$ and $B(Z)$ are the vectors proportional to the vector with all elements equal to one.

We assume in addition, without loss of generality, that

$$\frac{1}{2}\sum\sum_{1\leq i<j\leq n} w_{ij}\delta_{ij}^2 = 1.$$

With these definitions we can rewrite the stress (1) as

$$\sigma(Z) = 1 - \textbf{tr}\ Z'B(Z)Z + \frac{1}{2}\textbf{tr}\ Z'VZ, \tag{5}$$

and we can write the stationary equations as

$$(V - B(Z))Z = 0, \tag{6}$$

or, in fixed point form, $Z = V^{+}B(Z)Z$.

Equation (5) shows, by the way, something which is already obvious from (1). Distances are invariant under translation. This is reflected in $B(Z)$ and $V$ being doubly-centered. As a consequence we usually require, again without loss of generality, that $Z$ is column-centered. And that implies that $Z$ has rank at most $n-1$, which means that FDS is equivalent to minimizing stress over all $n \times (n-1)$ matrices, which we can assume to be column-centered as well. Configurations with $n-1$ columns can be called full configurations as well. In addition, distances are invariant under rotation, and consequently if $Z$ solves the stationary equations with value $\sigma(Z)$ then $ZK$ solves the stationary equations for all rotation matrices $K$, and $\sigma(ZK) = \sigma(K)$. This means there are no isolated local minima in configuration space, each local minimum is actually a continuum of rotated matrices in $\mathbb{R}^{n\times n}$. This is a nuisance in the analysis of FDS and pMDS that is best dealt with by switching to the parametrization outlined in De Leeuw (1993).

## 2 Convex FDS

Instead of defining the loss function (1) on the space of all $n \times n$ configuration matrices $Z$ we can also define it over the space of all positive semidefinite matrices $C$ of order $n$. This gives

$$\sigma(C) = 1 - \mathop{\sum\sum}_{1 \leq i < j \leq n} w_{ij}\delta_{ij}\sqrt{c_{ii} + c_{jj} - 2c_{ij}} + \frac{1}{2}\textbf{tr}\ VC. \tag{7}$$

The Convex Full-dimensional Scaling (CFDS) problem is to minimize loss function (7) over all $C \gtrsim 0$. Obviously if $Z$ minimizes (1) then $C = ZZ'$ minimizes (7). And, conversely, if $C$ minimizes (7) then any $Z$ such that $C = ZZ'$ minimizes (1).

The definition (7) shows that the CFDS loss function is a convex function on the cone of positive semi-definite matrices, because the square root of a non-negative linear function of the elements of $C$ is concave. Positivity of the weights and dissimilarities implies that loss is actually strictly convex. The necessary and sufficient conditions for $C$ to be the unique solution of the CFDS problem are simply the conditions for a proper convex function to attain its minimum at $C$ on a closed convex cone (Rockafellar (1970), theorem 31.4).

$$\begin{aligned} V - B(C) &\gtrsim 0, \\ C &\gtrsim 0, \\ \textbf{tr}\ C(V - B(C)) &= 0. \end{aligned}$$

The conditions say that $C$ and $V - B(C)$ must be positive semi-definite and have complimentary null spaces.

By the same reasoning as in the full configuration case, we also see that CFDS is equivalent to maximizing (7) over all doubly-centered positive semi-definite matrices.

If $C$ is the solution of the CFDS problem then $\textbf{rank}(C)$ is called the *Gower rank* of the MDS problem defined by $W$ and $\Delta$ (De Leeuw (2016)). Although there is a unique Gower rank associated with each CFDS problem, we can also talk about the *approximate Gower rank* by ignoring the small eigenvalues of $C$.

## 3 FDS using SMACOF

The usual SMACOF algorithm can be applied to FDS as well. The iterations start with $Z^{(0)}$ and use the update rule

$$Z^{(k+1)} = V^+B(Z^{(k)})Z^{(k)}, \tag{8}$$

where $V^+$ is the Moore-Penrose inverse of $V$, and is consequently also doubly-centered. This means that all $Z^{(k)}$ in the SMACOF sequence, except possibly $Z^{(0)}$, are column-centered and of rank at most $n - 1$. Equation (8) also shows that if $Z^{(0)}$ is of rank $p < n - 1$ then all $Z^{(k)}$ are of rank $p$ as well.

De Leeuw (1977) shows global convergence of the SMACOF sequence for pMDS, generated by (8), to a stationary point, i.e. a point satisfying $(V - B(Z))Z = 0$. This result also applies, of course, to nMDS, i.e. FDS. If $Z$ is a solution of the stationary equations then with $C = ZZ'$ we have both $(V - B(C))C = 0$ and $C \gtrsim 0$, but since we generally do not have $V - B(Z) \gtrsim 0$, this does not mean that $C$ solves the CFDS problem.

In fact, suppose the unique CMDS solution has Gower rank $r \geq 2$. Start the SMACOF FDS iterations (8) with $Z^{(0)}$ of the form $Z^{(0)} = \begin{bmatrix} X^{(0)} & \mid & 0 \end{bmatrix}$, where $X^{(0)}$ is an $n \times p$ matrix of rank $p < r$. All $Z^{(k)}$ will be of this form and will also be of rank $p$, and all accumulation points $Z$ of the SMACOF sequence will have this form and $\mathbf{rank}(Z) \leq p$. Thus $C = ZZ'$ cannot be the solution of the CMDS problem.

The next result shows that things are allright, after all. Although stress in FDS is certainly not a convex function of $Z$, it remains true that all local minima are global.

**Lemma 1: [Expand]** If FDS stress has a local minimum at $\begin{bmatrix} X & \mid & 0 \end{bmatrix}$, where $X$ is $n \times p$ and the zero block is $n \times q$ with $q > 1$, then

1: $\mathcal{D}\sigma(X) = (V - B(X))X = 0$.

2: $\mathcal{D}^2\sigma(X) \gtrsim 0$.

3: $V - B(X) \gtrsim 0$.

**Proof:** We use the fact that stress is differentiable at a local minimum (De Leeuw (1984)). If $Z = \begin{bmatrix} X & \mid & 0 \end{bmatrix} + \epsilon \begin{bmatrix} P & \mid & Q \end{bmatrix}$ then we must have $\sigma(Z) \geq \sigma(X)$ for all $P$ and $Q$. Now

$$\sigma(Z) = \sigma(X) + \epsilon \text{ tr } P'\mathcal{D}\sigma(X) + \\ + \frac{1}{2}\epsilon^2 \, \mathcal{D}^2\sigma(X)(P, P) + \frac{1}{2}\epsilon^2 \text{ tr } Q'(V - B(X))Q + o(\epsilon^2). \quad (9)$$

The lemma follows from this expansion. ■

**Theorem 1: [FDS Local Minima]** If stationary point $Z$ of FDS is a local minimum, then it also is the global minimum, and $C = ZZ'$ solves the CFDS problem.

**Proof:** We start with a special case. Suppose $Z$ is a doubly-centered solution of the FDS stationary equations with $\mathbf{rank}(Z) = n - 1$. Then $(V - B(Z))Z = 0$ implies $V = B(Z)$, which implies $\delta_{ij} = d_{ij}(Z)$ for all $i, j$. Thus $\sigma(Z) = 0$, which obviously is the global minimum.

Now suppose $Z$ is a doubly-centered local minimum solution of the FDS stationary equations with $\mathbf{rank}(Z) = r < n - 1$. Without loss of generality we assume $Z$ is of the form $Z = \begin{bmatrix} X & \mid & 0 \end{bmatrix}$, with $X$ an $n \times r$ matrix of rank $r$. For $C = ZZ'$ to be a solution of the CFDS problem it is necessary and sufficient that $V - B(Z) \gtrsim 0$. Lemma 1 shows that this is indeed the case at a local minimum. ■

**Corrollary 1: [Saddle]** A pMDS solution of the stationary equations with $Z$ singular is a saddle point.

**Corrollary 2: [Nested]** Solutions of the stationary equations of pMDS are saddle points of qMDS with $q > p$.

The proof of lemma 1 shows that for any $n \times p$ configuration $Z$, not just for solutions of the FDS stationary equations, if $V - B(Z)$ is indefinite we can decrease loss by adding another dimension. If $Z$ is a stationary point and $V - B(Z)$ is positive semi-definite then we actually have found the CFDS solution, the Gower rank, and the global minimum (De Leeuw (2014)).

## 4 Penalizing Dimensions

In Shepard (1962a) and Shepard (1962b) a nonmetric multidimensional scaling technique is developed which minimizes a loss function over configurations in full dimensionality $n-1$. In that sense the technique is similar to FDS. Shepard's iterative process aims to maintain monotonicity between distances and dissimilarities and at the same time concentrate as much of the variation as possible in a small number of dimensions (De Leeuw (2017)).

Let us explore the idea of concentrating variation in $p < n-1$ dimensions, but use an approach which is quite different from the one used by Shepard. We remain in the FDS framework, but we aim for solutions in $p < n-1$ dimensions by penalizing $n-p$ dimensions of the full configuration, using the classical Courant quadratic penalty function.

Partition a full configuration $Z = \begin{bmatrix} X & \mid & Y \end{bmatrix}$, with $X$ of dimension $n \times p$ and $Y$ of dimension $n \times (n-p)$. Then

$$\sigma(Z) = 1 - \mathbf{tr}\ X'B(Z)X - \mathbf{tr}\ Y'B(Z)Y + \frac{1}{2}\mathbf{tr}\ X'VX + \frac{1}{2}\mathbf{tr}\ Y'VY. \tag{10}$$

Also define the *penalty term*

$$\tau(Y) = \frac{1}{2}\mathbf{tr}\ Y'VY, \tag{11}$$

and *penalized stress*

$$\pi(Z,\lambda) = \sigma(Z) + \lambda\ \tau(Y). \tag{12}$$

Our proposed method is to minimize penalized stress over $Z$ for a sequence of values $0 = \lambda_1 < \lambda_2 < \cdots < \lambda_m$. For $\lambda = 0$ this is simply the FDS problem, for which we know we can compute the global minimum. For fixed $0 < \lambda < +\infty$ this is a Penalized FDS or PFDS problem. PFDS problems with increasing values of $\lambda$ generate a *trajectory* $Z(\lambda)$ in configuration space.

The general theory of exterior penalty functions, which we review in appendix A of this paper, shows that increasing $\lambda$ leads to an increasing sequence of stress values $\sigma$ and a decreasing sequence of penalty terms $\tau$. If $\lambda \rightarrow +\infty$ we approximate the global minimum of the FDS problem with $Z$ of the form $Z = \begin{bmatrix} X & \mid & 0 \end{bmatrix}$, i.e. of the pMDS problem. This assumes we do actually compute the global minimum for each value of $\lambda$, which we hope we can do because we start at the FDS global minimum, and we slowly increase $\lambda$. There is also a local version of the exterior penalty result, which implies that $\lambda \rightarrow \infty$ takes us to a local minimum of pMDS, so there is always the possibility of taking the wrong trajectory to a local minimum of pMDS.

## 4.1 Local Minima

The stationary equations of the PFDS problem are solutions to the equations

$$(V - B(Z))X = 0, \tag{13}$$

$$((1+\lambda)V - B(Z))Y = 0. \tag{14}$$

We can easily related stationary points and local minima of the FDS and PFDS problem.

**Theorem 2: [PFDS Local Minima]**

1: If $X$ is a stationary point of the pMDS problem then $Z = [X \mid 0]$ is a stationary point of the PFDS problem, no matter what $\lambda$ is.

2: If $Z = [X \mid 0]$ is a local minimum of the PFDS problem then $X$ is a local minimum of pMDS and $(1+\lambda)V - B(X) \gtrsim 0$, or $\lambda \geq \|V^+B(X)\|_\infty - 1$, with $\| \bullet \|_\infty$ the spectral radius (largest eigenvalue).

**Proof:**

Part 1 follows by simple substitution in the stationary equations.

Part 2 follows from the expansion for $Z = [X + \epsilon P \mid \epsilon Q]$.

$$\pi(Z) = \pi(X) + \epsilon \text{ tr } P'\mathcal{D}\sigma(X) + \\ + \frac{1}{2}\epsilon^2 \mathcal{D}^2\sigma(X)(P,P) + \frac{1}{2}\epsilon^2 \text{ tr } Q'((1+\lambda)V - B(X))Q + o(\epsilon^2). \tag{15}$$

At a local minimum we must have $\mathcal{D}\sigma(X) = 0$ and $\mathcal{D}^2\sigma(X)(P,P) \gtrsim 0$, which are the necessary conditions for a local minimum of pMDS. We also must have $((1+\lambda)V - B(X)) \gtrsim 0$. ■

Note that the conditions in part 2 of theorem 2 are also sufficient for PFDS to have a local minimum at $[X \mid 0]$, provided we eliminate translational and rotational indeterminacy by a suitable reparametrization, as in De Leeuw (1993).

# 5 Algorithm

The SMACOF algorithm for penalized stress is a small modification of the unpenalized FDS algorithm (8). We start our iterations for $\lambda_j$ with the solution for $\lambda_{j-1}$ (the starting solution for $\lambda_1 = 0$ can be completely arbitrary). The update rules for fixed $\lambda$ are

$$X^{(k+1)} = V^+B(Z^{(k)})X^{(k)}, \tag{16}$$

$$Y^{(k+1)} = \frac{1}{1+\lambda}V^+B(Z^{(k)})Y^{(k)}. \tag{17}$$

Thus we compute the FDS update $Z^{(k+1)} = V^+B(Z^{(k)})Z^{(k)}$ and then divide the last $n-p$ columns by $1+\lambda$.

Code is in the appendix. Let us analyze a number of examples.

## 6 Examples

This section has a number of two-dimensional and a number of one-dimensional examples. The one-dimensional examples are of interest, because of the documented large number of local minima of stress in the one-dimensional case, and the fact that for small and medium $n$ exact solutions are available (for example, De Leeuw (2005)). By default we use `seq(0, 1, length = 101)` for $\lambda$ in most examples, but for some of them we dig a bit deeper and use longer sequences with smaller increments.

If for some value of $\lambda$ the penalty term drops below the small cutoff $\gamma$, for example $10^{-10}$, then there is not need to try larger values of $\lambda$, because they will just repeat the same result. We hope that result is the global minimum of the 2MDS problem.

The output for each example is a table in which we give, the minimum value of stress, the value of the penalty term at the minimum, the value of $\lambda$, and the number of iterations needed for convergence. Typically we print for the first three, the last three, and some regularly spaced intermediate values of $\lambda$. Remember that the stress values increase with increasing $\lambda$, and the penalty values decrease.

For two-dimensional examples we plot all two-dimensional configurations, after rotating to optimum match (using the function `matchMe()` from the appendix). We connect corresponding points for different values of $\lambda$. Points corresponding to the highest value of $\lambda$ are labeled and have a different plot symbol. For one-dimensional examples we put `1:n` on the horizontal axes and plot the single dimension on the vertical axis, again connecting corresponding points. We label the points corresponding with the highest value of $\lambda$, and draw horizontal lines through them to more clearly show their order on the dimension.

It should be emphasized that all examples are just meant to study performance and convergence of penalized FDS. There is no interpretation of the MDS results

### 6.1 Two-dimensional Examples

#### 6.1.1 Chi Squares

In this example, of order 10, the $\delta_{ij}$ are square roots of independent draws from a central chi-square distribution with two degrees of freedom.

```r
set.seed(54321)
chi <- matrix(0, 10, 10)
for (i in 2:10) for (j in 1:(i - 1))
  chi[i,j] <- chi[j,i] <- sqrt(rchisq(1, 2))
chi <- 2 * chi / sqrt(sum(chi * chi))
mPrint(chi, digits = 2)
```

```
##        [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]
##  [1,]    +0.00    +0.11    +0.08    +0.17    +0.12    +0.26    +0.14    +0.19
##  [2,]    +0.11    +0.00    +0.11    +0.17    +0.04    +0.24    +0.16    +0.31
##  [3,]    +0.08    +0.11    +0.00    +0.17    +0.07    +0.28    +0.16    +0.19
##  [4,]     +0.17     +0.17     +0.17     +0.00     +0.18     +0.20     +0.25     +0.17
##  [5,]     +0.12     +0.04     +0.07     +0.18     +0.00     +0.38     +0.47     +0.18
##  [6,]     +0.26     +0.24     +0.28     +0.20     +0.38     +0.00     +0.05     +0.29
##  [7,]     +0.14     +0.16     +0.16     +0.25     +0.47     +0.05     +0.00     +0.12
##  [8,]     +0.19     +0.31     +0.19     +0.17     +0.18     +0.29     +0.12     +0.00
##  [9,]     +0.10     +0.25     +0.05     +0.09     +0.36     +0.07     +0.14     +0.32
## [10,]     +0.19     +0.25     +0.07     +0.17     +0.21     +0.16     +0.05     +0.37
##         [,9]      [,10]
##  [1,]     +0.10     +0.19
##  [2,]     +0.25     +0.25
##  [3,]     +0.05     +0.07
##  [4,]     +0.09     +0.17
##  [5,]     +0.36     +0.21
##  [6,]     +0.07     +0.16
##  [7,]     +0.14     +0.05
##  [8,]     +0.32     +0.37
##  [9,]     +0.00     +0.29
## [10,]     +0.29     +0.00
```

If we analyze the data with smacof in two dimensions, using the Torgerson initial configuration, we find a stress of 0.0862287021 after 65 iterations. All smacof runs in this paper have a maximum number of iterations of 10,000 and stop if the difference between successive stress values is less than 1e-15.

A full-dimensional scaling with $p = 9$ gives stress 0.0730261617 after 198 iterations. The singular values of the full dimensional solution are

```
## [1] +0.322688 +0.207891 +0.163634 +0.096146 +0.045038 +0.000001 +0.000000
## [8] +0.000000 +0.000000
```

and thus the Gower rank of these data is five.

We can also start our smacof iterations with the first two dimensions of the full dimensional solution. We call this the FDS(2) solution, or more generally the FDS(p) solution with $p = 2$. Stress is 0.0862287021 after 62 iterations, the same solution as with the Torgerson start.

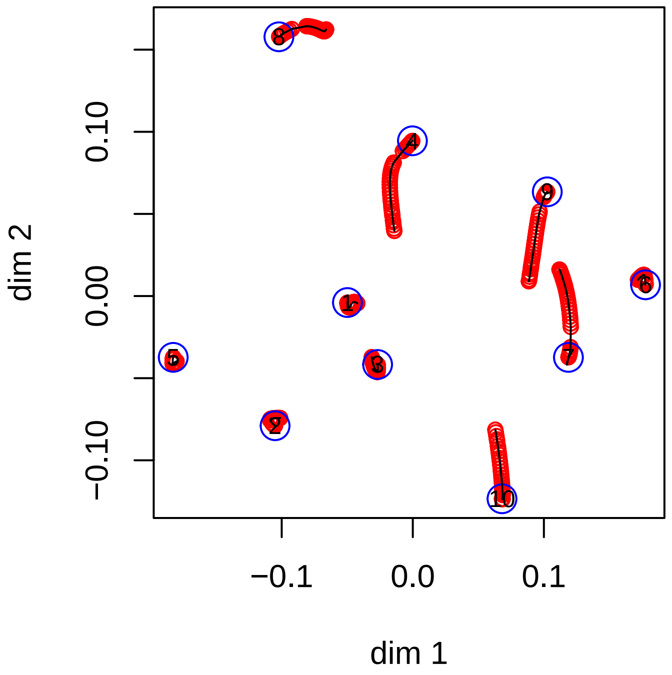
If we run smacof 10,000 times with a random start we find five different local minima, and the one with the smallest stress is found in about 35% of the cases. Again, this is the Torgerson solution.

```
## 
## 0.0862287 0.0955797 0.0990518 0.0995697  0.100125 
##      3657      2593      1670       802      1278
```

We now apply our global MDS method, using 101 values of $\lambda$, equally spaced between zero and one. Again we converge on the Torgerson solution, which by all indications is the global minimum of stress for these data. In the table below itel is the number of smacof iterations for a given value of $\lambda$, and the penalty column has the sum of squares of the last $n - 2$

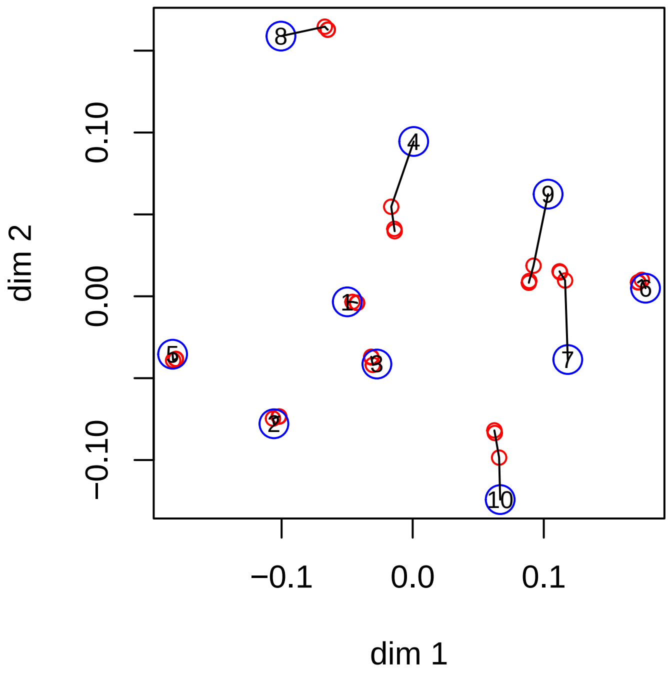
dimensions of the penalized FDS solution. The plot shows the movements of the points as $\lambda$ changes, starting at the FDS solution and ending with the point in the blue circle.

```
## itel  203 lambda   0.000000 stress 0.073026 penalty 0.422435
## itel    4 lambda   0.010000 stress 0.073054 penalty 0.091005
## itel    3 lambda   0.020000 stress 0.073141 penalty 0.086529
## itel    2 lambda   0.030000 stress 0.073269 penalty 0.082545
## itel    1 lambda   0.040000 stress 0.073390 penalty 0.079778
## itel    2 lambda   0.050000 stress 0.073688 penalty 0.074219
## itel    1 lambda   0.060000 stress 0.073899 penalty 0.071053
## itel    1 lambda   0.070000 stress 0.074170 penalty 0.067522
## itel    1 lambda   0.080000 stress 0.074497 penalty 0.063768
## itel    1 lambda   0.090000 stress 0.074874 penalty 0.059891
## itel    1 lambda   0.100000 stress 0.075299 penalty 0.055962
## itel    1 lambda   0.110000 stress 0.075765 penalty 0.052032
## itel    1 lambda   0.120000 stress 0.076269 penalty 0.048144
## itel    1 lambda   0.130000 stress 0.076806 penalty 0.044331
## itel    1 lambda   0.140000 stress 0.077368 penalty 0.040621
## itel    1 lambda   0.150000 stress 0.077951 penalty 0.037038
## itel    1 lambda   0.160000 stress 0.078548 penalty 0.033601
## itel    1 lambda   0.170000 stress 0.079152 penalty 0.030322
## itel    1 lambda   0.180000 stress 0.079756 penalty 0.027213
## itel    1 lambda   0.190000 stress 0.080355 penalty 0.024279
## itel    1 lambda   0.200000 stress 0.080942 penalty 0.021526
## itel    1 lambda   0.210000 stress 0.081511 penalty 0.018955
## itel    1 lambda   0.220000 stress 0.082058 penalty 0.016567
## itel    1 lambda   0.230000 stress 0.082578 penalty 0.014364
## itel    1 lambda   0.240000 stress 0.083067 penalty 0.012345
## itel    9 lambda   0.250000 stress 0.085317 penalty 0.003278
## itel    2 lambda   0.260000 stress 0.085541 penalty 0.002404
## itel    2 lambda   0.270000 stress 0.085724 penalty 0.001723
## itel    3 lambda   0.280000 stress 0.085924 penalty 0.001009
## itel    2 lambda   0.290000 stress 0.086017 penalty 0.000689
## itel    3 lambda   0.300000 stress 0.086111 penalty 0.000374
## itel    4 lambda   0.310000 stress 0.086177 penalty 0.000158
## itel    3 lambda   0.320000 stress 0.086202 penalty 0.000080
## itel    4 lambda   0.330000 stress 0.086218 penalty 0.000031
## itel    6 lambda   0.340000 stress 0.086226 penalty 0.000007
## itel    8 lambda   0.350000 stress 0.086228 penalty 0.000001
```



As the results show, nothing much happens during long stretches of changes in $\lambda$. Thus we try the much shorter sequence $(0, 0.01, 0.1, 1, 10)$.

```
## itel  203 lambda   0.000000 stress 0.073026 penalty 0.422435
## itel    4 lambda   0.010000 stress 0.073054 penalty 0.091005
## itel    6 lambda   0.100000 stress 0.075646 penalty 0.054890
## itel   60 lambda   1.000000 stress 0.086229 penalty 0.000000
```



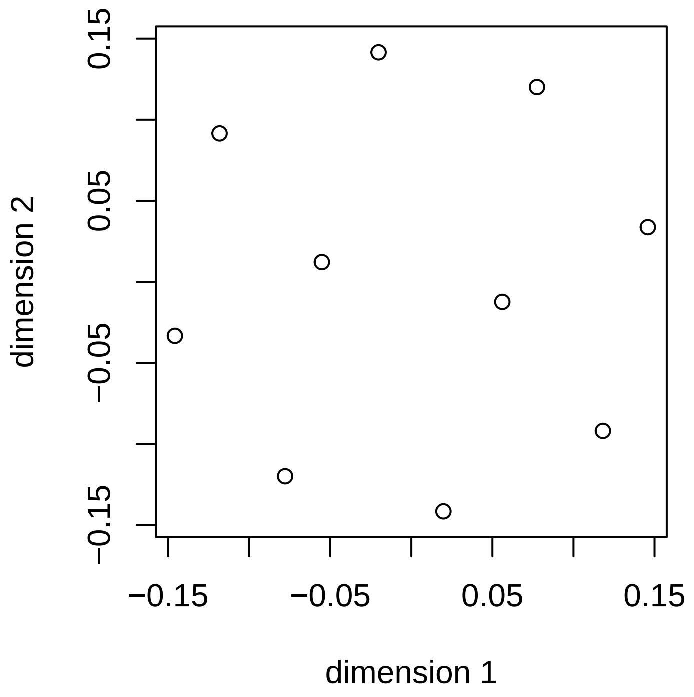
Again, the same result.

### 6.1.2 Regular Simplex

The regular simplex has all dissimilarities equal to one. We use an example with $n = 10$, for which the global minimum (as far as we know) of pMDS with $p = 2$ is a configuration with nine points equally spaced on a circle and one point in the center.

Using the Torgerson initial configuration in two dimensions, smacof finds a stress of 0.1110521776 after 872 iterations. The solution is *not* the presumed global minimum, because it shows points on a circle and two points in the interior.

```
par(pty = "s")
xmin <- min(hsimt$x)
xmax <- max(hsimt$x)
plot(hsimt$x, xlim = c(xmin, xmax), ylim = c(xmin, xmax),
     xlab = "dimension 1", ylab = "dimension 2")
```



A full-dimensional scaling with $p = 9$ trivially gives stress $1.8681520461 \times 10^{-31}$ after 1 iterations. The singular values of the full dimensional solution are

```
## [1] +0.149071 +0.149071 +0.149071 +0.149071 +0.149071 +0.149071 +0.149071
## [8] +0.149071 +0.149071
```

and thus the Gower rank of these data is nine.

In FDS(2) stress is 0.1110521776 after 872 iterations, the same solution as with the Torgerson start.

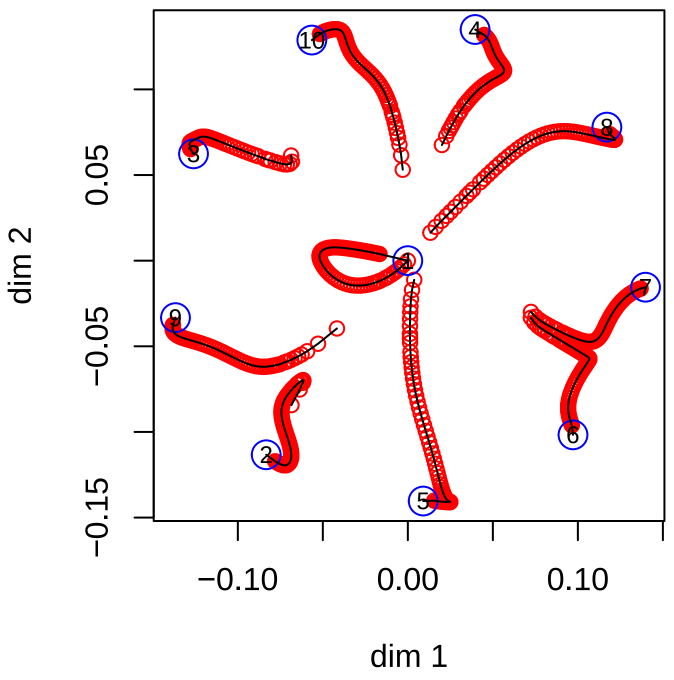
```
## 
##  0.10988 0.111052 0.111058 
##     7160     2829       11
```

If we run smacof 10,000 times with a random start we find three different local minima, and the one with the smallest stress 0.109879978 is found in about 70% of the cases. Again, this the global solution. The solution found by starting with Torgerson or FDS is found in about 30% of the cases.

There are, of course, many more local minima as show in the table, because each permutation of the 10 solution points at a local minimum gives another local minimum with the same stress value.

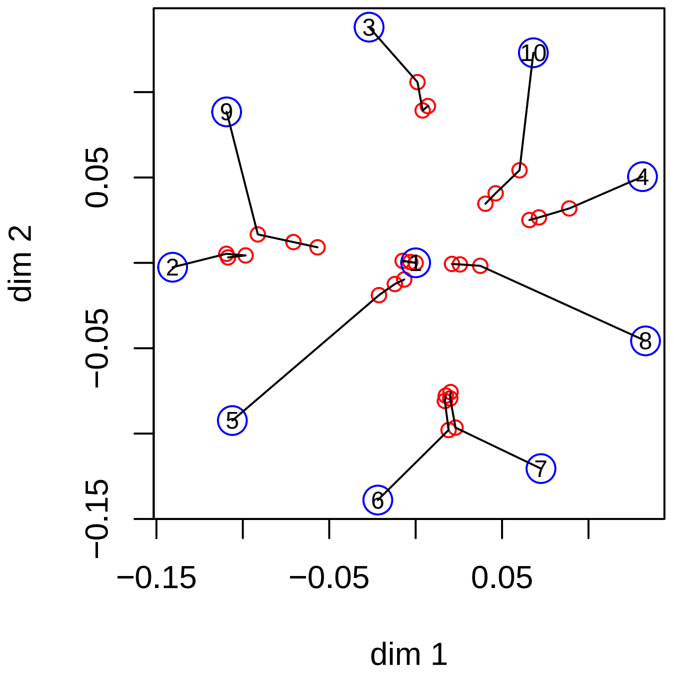
Our global optimization method with the default $\lambda$ sequence also converges on the global solution.

```
## itel    1 lambda   0.000000 stress 0.000000 penalty 0.400000
## itel    7 lambda   0.010000 stress 0.000103 penalty 0.375240
## itel    5 lambda   0.020000 stress 0.000427 penalty 0.360212
## itel    3 lambda   0.030000 stress 0.000916 penalty 0.346937
## itel    2 lambda   0.040000 stress 0.001533 penalty 0.334923
## itel    2 lambda   0.050000 stress 0.002378 penalty 0.321650
## itel    2 lambda   0.060000 stress 0.003468 penalty 0.307513
## itel    2 lambda   0.070000 stress 0.004816 penalty 0.292842
## itel    1 lambda   0.080000 stress 0.005905 penalty 0.283148
## itel    1 lambda   0.090000 stress 0.007178 penalty 0.272814
## itel    2 lambda   0.100000 stress 0.009438 penalty 0.255499
## itel    1 lambda   0.110000 stress 0.011002 penalty 0.245389
## itel    1 lambda   0.120000 stress 0.012752 penalty 0.234937
## itel    1 lambda   0.130000 stress 0.014677 penalty 0.224281
## itel    1 lambda   0.140000 stress 0.016766 penalty 0.213553
## itel    1 lambda   0.150000 stress 0.019004 penalty 0.202867
## itel    1 lambda   0.160000 stress 0.021375 penalty 0.192317
## itel    1 lambda   0.170000 stress 0.023863 penalty 0.181981
## itel    1 lambda   0.180000 stress 0.026446 penalty 0.171921
## itel    1 lambda   0.190000 stress 0.029107 penalty 0.162184
## itel    1 lambda   0.200000 stress 0.031826 penalty 0.152804
## itel    1 lambda   0.210000 stress 0.034585 penalty 0.143802
## itel    1 lambda   0.220000 stress 0.037368 penalty 0.135189
## itel    1 lambda   0.230000 stress 0.040161 penalty 0.126967
## itel    1 lambda   0.240000 stress 0.042951 penalty 0.119133
## itel    1 lambda   0.250000 stress 0.045728 penalty 0.111680
## itel    1 lambda   0.260000 stress 0.048483 penalty 0.104597
## itel    1 lambda   0.270000 stress 0.051208 penalty 0.097873
## itel    1 lambda   0.280000 stress 0.053896 penalty 0.091496
## itel    1 lambda   0.290000 stress 0.056540 penalty 0.085455
## itel    1 lambda   0.300000 stress 0.059136 penalty 0.079739
## itel    1 lambda   0.310000 stress 0.061677 penalty 0.074338
## itel    1 lambda   0.320000 stress 0.064160 penalty 0.069241
## itel    1 lambda   0.330000 stress 0.066580 penalty 0.064438
## itel    1 lambda   0.340000 stress 0.068935 penalty 0.059918
## itel    1 lambda   0.350000 stress 0.071222 penalty 0.055668
## itel    1 lambda   0.360000 stress 0.073439 penalty 0.051676
## itel    1 lambda   0.370000 stress 0.075586 penalty 0.047928
## itel    1 lambda   0.380000 stress 0.077662 penalty 0.044410
## itel    1 lambda   0.390000 stress 0.079669 penalty 0.041106
## itel    1 lambda   0.400000 stress 0.081607 penalty 0.038004
## itel    1 lambda   0.410000 stress 0.083477 penalty 0.035089
## itel    1 lambda   0.420000 stress 0.085281 penalty 0.032347
## itel    1 lambda   0.430000 stress 0.087019 penalty 0.029768
## itel    1 lambda   0.440000 stress 0.088693 penalty 0.027341
## itel    1 lambda   0.450000 stress 0.090303 penalty 0.025055
## itel    1 lambda   0.460000 stress 0.091849 penalty 0.022903
## itel    1 lambda   0.470000 stress 0.093332 penalty 0.020877
## itel    1 lambda   0.480000 stress 0.094752 penalty 0.018972
## itel    1 lambda   0.490000 stress 0.096108 penalty 0.017181
## itel    1 lambda   0.500000 stress 0.097400 penalty 0.015499
## itel    1 lambda   0.510000 stress 0.098629 penalty 0.013924
## itel    1 lambda   0.520000 stress 0.099794 penalty 0.012450
## itel    1 lambda   0.530000 stress 0.100893 penalty 0.011076
## itel    1 lambda   0.540000 stress 0.101928 penalty 0.009798
## itel    1 lambda   0.550000 stress 0.102895 penalty 0.008615
## itel    1 lambda   0.560000 stress 0.103795 penalty 0.007525
## itel    1 lambda   0.570000 stress 0.104626 penalty 0.006527
## itel    1 lambda   0.580000 stress 0.105388 penalty 0.005618
## itel    1 lambda   0.590000 stress 0.106079 penalty 0.004797
## itel    1 lambda   0.600000 stress 0.106700 penalty 0.004060
## itel    1 lambda   0.610000 stress 0.107252 penalty 0.003405
## itel    1 lambda   0.620000 stress 0.107737 penalty 0.002828
## itel    1 lambda   0.630000 stress 0.108158 penalty 0.002324
## itel    1 lambda   0.640000 stress 0.108518 penalty 0.001890
## itel    1 lambda   0.650000 stress 0.108822 penalty 0.001520
## itel    1 lambda   0.660000 stress 0.109075 penalty 0.001208
## itel    1 lambda   0.670000 stress 0.109281 penalty 0.000948
## itel    1 lambda   0.680000 stress 0.109447 penalty 0.000736
## itel    1 lambda   0.690000 stress 0.109578 penalty 0.000564
## itel    1 lambda   0.700000 stress 0.109679 penalty 0.000427
## itel    1 lambda   0.710000 stress 0.109756 penalty 0.000320
## itel   92 lambda   0.720000 stress 0.109880 penalty 0.000000
```



The same result is obtained with the short $\lambda$ sequence.

```
## itel    1 lambda   0.000000 stress 0.000000 penalty 0.400000
## itel    7 lambda   0.010000 stress 0.000103 penalty 0.375240
## itel    6 lambda   0.100000 stress 0.008197 penalty 0.268859
## itel  122 lambda   1.000000 stress 0.109880 penalty 0.000000
```



### 6.1.3 Intelligence

These are correlations between eight intelligence tests, taken from the `smacof` package. We convert to dissimilarities by taking the negative logarithm of the correlations.

```
data(intelligence, package = "smacof")
cor <- as.matrix(intelligence[, c("T1","T2","T3","T4","T5","T6","T7","T8")])
intel <- -log(cor)
w <- matrix(1, 8, 8) - diag(8)
intel <- 2 * intel / sqrt(sum(w * intel * intel))
mPrint(intel, digits = 2)
```

```
##       T1       T2       T3       T4       T5       T6       T7       T8
## [1,]    -0.00    +0.09    +0.20    +0.36    +0.46    +0.30    +0.29    +0.20
## [2,]    +0.09    -0.00    +0.15    +0.29    +0.35    +0.27    +0.29    +0.21
## [3,]    +0.20    +0.15    -0.00    +0.14    +0.20    +0.25    +0.37    +0.31
## [4,]    +0.36    +0.29    +0.14    -0.00    +0.13    +0.15    +0.30    +0.33
## [5,]    +0.46    +0.35    +0.20    +0.13    -0.00    +0.17    +0.27    +0.42
## [6,]    +0.30    +0.27    +0.25    +0.15    +0.17    -0.00    +0.19    +0.21
## [7,]    +0.29    +0.29    +0.37    +0.30    +0.27    +0.19    -0.00    +0.20
## [8,]    +0.20    +0.21    +0.31    +0.33    +0.42    +0.21    +0.20    -0.00
```

Using the Torgerson initial configuration in two dimensions, smacof finds a stress of 0.0075160651 after 28 iterations. This corresponds with the presumed global minimum.

A full-dimensional scaling with $p = 7$ gives stress 0.0049386956 after 1554 iterations. The singular values of the full dimensional solution are

```
## [1] +0.382699 +0.291593 +0.127580 +0.031521 +0.003190 +0.000000 +0.000000
```

and thus the Gower rank of these data is five.

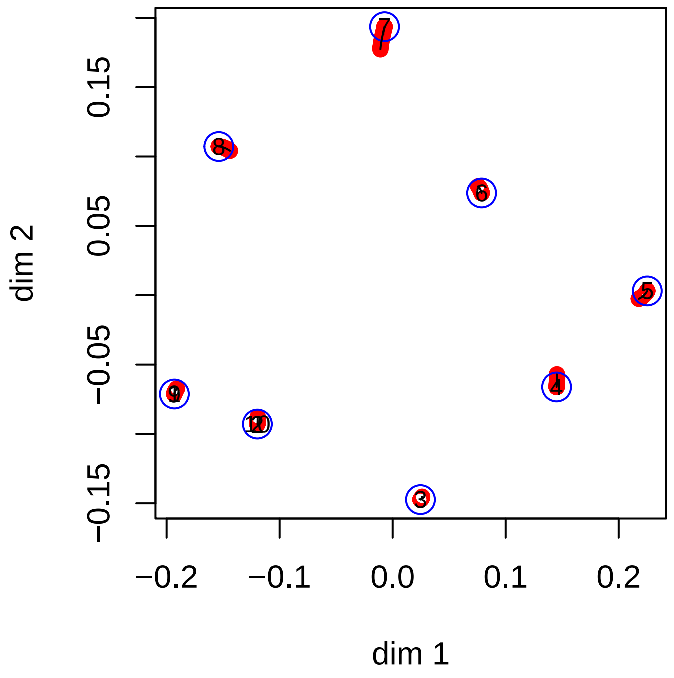
For FDS(2) stress is 0.0075160651 after 26 iterations, the same solution as with the Torgerson start.

If we run smacof 10,000 times with a random start we find two different local minima, and the one with the smallest stress is found in about 90% of the cases. Again, this is the same solution.

```
##
## 0.00751607  0.0573864
##       9086        914
```

Our global optimization method with the default $\lambda$ sequence also converges to the same solution.

```
## itel 1553 lambda   0.000000 stress 0.004939 penalty 0.368081
## itel    7 lambda   0.010000 stress 0.004956 penalty 0.031597
## itel    4 lambda   0.020000 stress 0.005001 penalty 0.028964
## itel    3 lambda   0.030000 stress 0.005070 penalty 0.026414
## itel    3 lambda   0.040000 stress 0.005181 penalty 0.023535
## itel    2 lambda   0.050000 stress 0.005286 penalty 0.021411
## itel    2 lambda   0.060000 stress 0.005420 penalty 0.019125
## itel    2 lambda   0.070000 stress 0.005580 penalty 0.016798
## itel    2 lambda   0.080000 stress 0.005760 penalty 0.014516
## itel    2 lambda   0.090000 stress 0.005954 penalty 0.012341
## itel    2 lambda   0.100000 stress 0.006155 penalty 0.010317
## itel    2 lambda   0.110000 stress 0.006356 penalty 0.008472
## itel    3 lambda   0.120000 stress 0.006631 penalty 0.006162
## itel    2 lambda   0.130000 stress 0.006795 penalty 0.004892
## itel    4 lambda   0.140000 stress 0.007060 penalty 0.002972
## itel    4 lambda   0.150000 stress 0.007241 penalty 0.001739
## itel    9 lambda   0.160000 stress 0.007438 penalty 0.000476
## itel   52 lambda   0.170000 stress 0.007516 penalty 0.000000
## itel    8 lambda   0.180000 stress 0.007516 penalty 0.000000
## itel    4 lambda   0.190000 stress 0.007516 penalty 0.000000
## itel    2 lambda   0.200000 stress 0.007516 penalty 0.000000
## itel    2 lambda   0.210000 stress 0.007516 penalty 0.000000
```



As in the chi-square example, the FDS and the 2MDS solution are very similar and the PMDS trajectories are short.

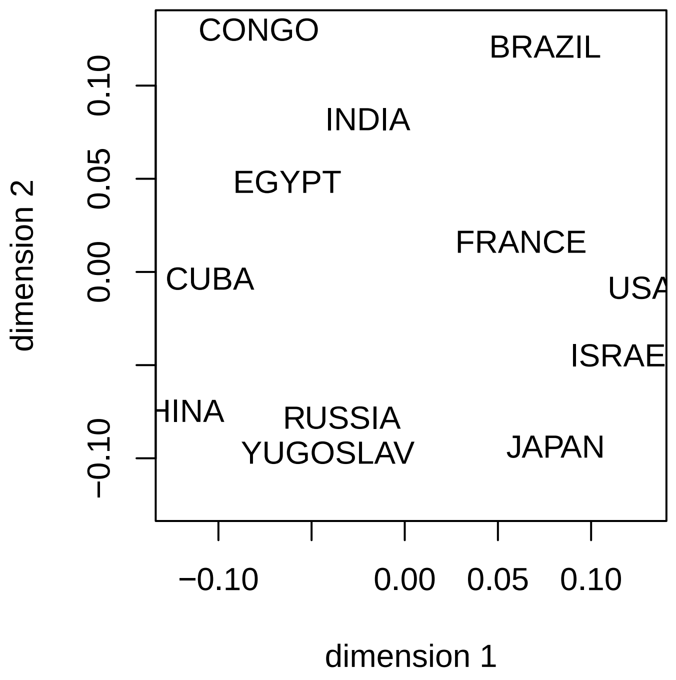
### 6.1.4 Countries

This is the `wish` dataset from the ‘smacof’ package, with similarities between 12 countries. They are converted to dissimilarities by subtracting each of them from seven.

```r
data(wish, package = "smacof")
countries <- as.matrix(wish)
countries <- 7 - countries
diag(countries) <- 0
countries <- 2 * countries / sqrt(sum(countries * countries))
mPrint(countries, digits = 2)
```

```
##            BRAZIL   CONGO    CUBA     EGYPT    FRANCE   INDIA    ISRAEL
## BRAZIL      +0.00    +0.13    +0.10    +0.22    +0.14    +0.15    +0.19
## CONGO       +0.13    +0.00    +0.15    +0.12    +0.18    +0.13    +0.22
## CUBA        +0.10    +0.15    +0.00    +0.11    +0.18    +0.18    +0.21
## EGYPT       +0.22    +0.12    +0.11    +0.00    +0.14    +0.07    +0.14
## FRANCE      +0.14    +0.18    +0.18    +0.14    +0.00    +0.22    +0.18
## INDIA       +0.15    +0.13    +0.18    +0.07    +0.22    +0.00    +0.18
## ISRAEL      +0.19    +0.22    +0.21    +0.14    +0.18    +0.18    +0.00
## JAPAN       +0.21    +0.22    +0.25    +0.19    +0.17    +0.15    +0.13
## CHINA       +0.28    +0.18    +0.09    +0.16    +0.20    +0.18    +0.24
## RUSSIA      +0.24    +0.22    +0.10    +0.16    +0.12    +0.15    +0.17
## USA         +0.10    +0.28    +0.23    +0.22    +0.06    +0.17    +0.06
## YUGOSLAV    +0.23    +0.21    +0.12    +0.17    +0.14    +0.18    +0.16
##            JAPAN    CHINA    RUSSIA   USA      YUGOSLAV
## BRAZIL      +0.21    +0.28    +0.24    +0.10    +0.23
## CONGO       +0.22    +0.18    +0.22    +0.28    +0.21
## CUBA        +0.25    +0.09    +0.10    +0.23    +0.12
## EGYPT       +0.19    +0.16    +0.16    +0.22    +0.17
## FRANCE      +0.17    +0.20    +0.12    +0.06    +0.14
## INDIA       +0.15    +0.18    +0.15    +0.17    +0.18
## ISRAEL      +0.13    +0.24    +0.17    +0.06    +0.16
## JAPAN       +0.00    +0.17    +0.15    +0.06    +0.17
## CHINA       +0.17    +0.00    +0.08    +0.27    +0.12
## RUSSIA      +0.15    +0.08    +0.00    +0.12    +0.02
## USA         +0.06    +0.27    +0.12    +0.00    +0.21
## YUGOSLAV    +0.17    +0.12    +0.02    +0.21    +0.00
```

Using the Torgerson initial configuration in two dimensions, smacof finds a stress of 0.0477490806 after 103 iterations.



A full-dimensional scaling with $p = 11$ gives stress 0.0159699675 after 763 iterations. The singular values of the full dimensional solution are

```
##  [1] +0.256524 +0.226618 +0.163113 +0.109658 +0.081072 +0.040506 +0.000595
##  [8] +0.000000 +0.000000 +0.000000 +0.000000
```

and thus the Gower rank of these data is seven.

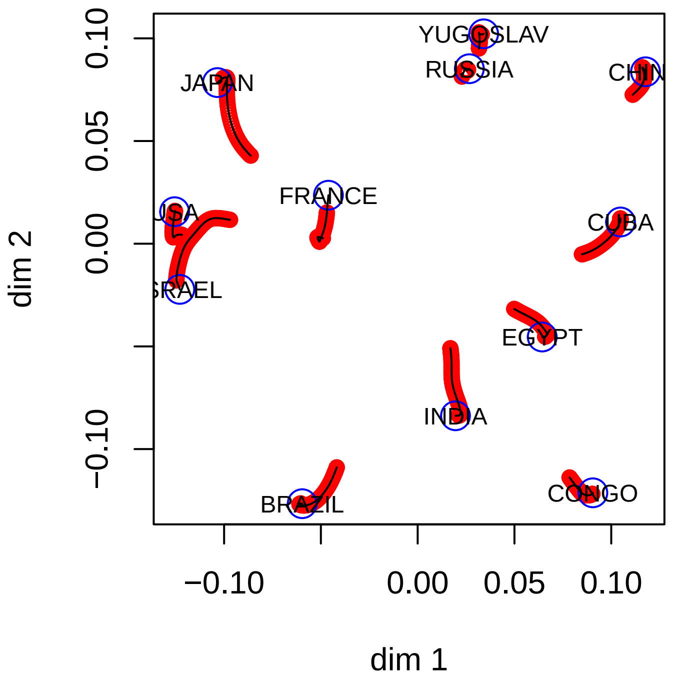
For FDS(2) stress is 0.0477490807 after 95 iterations, and we find the same solution as with the Torgerson start.

```
## 
## 0.0474139 0.0477491 0.0494938 0.0513037 0.0669776 0.0671137 0.0691977 0.0698188 
##      3496      1881      1190      1154       583       672       338       354 
## 0.0898758 0.0909081 0.0911124 0.0913887 0.0914608 0.0917437  0.100849  0.102294 
##        34        50         6        19        38        33        61        81 
##  0.107119  0.125733 
##         4         6
```

If we run smacof 10,000 times with a random start we find eighteen different local minima, and the one with the smallest stress 0.0474139053 is found in about 35% of the cases. Note, however, that the Torgerson and FDS solutions correspond with the second smallest stress, which is consequently certainly does not give the global minimum. It is found in about 20% of the cases. Note that the stress values of the two solutions are very close.

Our global optimization method with the default long $\lambda$ sequence does converge to the solution with minimum stress 0.0474139053, which is our tentative global minimum.

```
## itel  802 lambda   0.000000 stress 0.015970 penalty 0.367070
## itel    4 lambda   0.010000 stress 0.016010 penalty 0.134507
## itel    3 lambda   0.020000 stress 0.016140 penalty 0.128004
## itel    2 lambda   0.030000 stress 0.016331 penalty 0.122297
## itel    1 lambda   0.040000 stress 0.016515 penalty 0.118330
## itel    1 lambda   0.050000 stress 0.016780 penalty 0.113631
## itel    1 lambda   0.060000 stress 0.017127 penalty 0.108490
## itel    1 lambda   0.070000 stress 0.017557 penalty 0.103107
## itel    1 lambda   0.080000 stress 0.018068 penalty 0.097622
## itel    1 lambda   0.090000 stress 0.018654 penalty 0.092138
## itel    1 lambda   0.100000 stress 0.019310 penalty 0.086729
## itel    1 lambda   0.110000 stress 0.020028 penalty 0.081447
## itel    1 lambda   0.120000 stress 0.020801 penalty 0.076329
## itel    1 lambda   0.130000 stress 0.021620 penalty 0.071403
## itel    1 lambda   0.140000 stress 0.022479 penalty 0.066685
## itel    1 lambda   0.150000 stress 0.023369 penalty 0.062184
## itel    1 lambda   0.160000 stress 0.024283 penalty 0.057904
## itel    1 lambda   0.170000 stress 0.025215 penalty 0.053847
## itel    1 lambda   0.180000 stress 0.026159 penalty 0.050009
## itel    1 lambda   0.190000 stress 0.027110 penalty 0.046385
## itel    1 lambda   0.200000 stress 0.028062 penalty 0.042967
## itel    1 lambda   0.210000 stress 0.029011 penalty 0.039749
## itel    1 lambda   0.220000 stress 0.029953 penalty 0.036722
## itel    1 lambda   0.230000 stress 0.030885 penalty 0.033877
## itel    1 lambda   0.240000 stress 0.031802 penalty 0.031208
## itel    1 lambda   0.250000 stress 0.032702 penalty 0.028706
## itel    1 lambda   0.260000 stress 0.033582 penalty 0.026364
## itel    1 lambda   0.270000 stress 0.034440 penalty 0.024176
## itel    1 lambda   0.280000 stress 0.035273 penalty 0.022136
## itel    1 lambda   0.290000 stress 0.036079 penalty 0.020237
## itel    1 lambda   0.300000 stress 0.036857 penalty 0.018474
## itel    1 lambda   0.310000 stress 0.037604 penalty 0.016841
## itel    1 lambda   0.320000 stress 0.038319 penalty 0.015332
## itel    1 lambda   0.330000 stress 0.039003 penalty 0.013940
## itel    1 lambda   0.340000 stress 0.039654 penalty 0.012659
## itel    1 lambda   0.350000 stress 0.040272 penalty 0.011482
## itel    1 lambda   0.360000 stress 0.040858 penalty 0.010403
## itel    1 lambda   0.370000 stress 0.041413 penalty 0.009413
## itel    1 lambda   0.380000 stress 0.041936 penalty 0.008508
## itel    1 lambda   0.390000 stress 0.042428 penalty 0.007679
## itel    1 lambda   0.400000 stress 0.042892 penalty 0.006922
## itel    1 lambda   0.410000 stress 0.043326 penalty 0.006229
## itel    1 lambda   0.420000 stress 0.043734 penalty 0.005595
## itel    1 lambda   0.430000 stress 0.044116 penalty 0.005015
## itel    1 lambda   0.440000 stress 0.044472 penalty 0.004483
## itel    1 lambda   0.450000 stress 0.044803 penalty 0.003995
## itel    1 lambda   0.460000 stress 0.045112 penalty 0.003547
## itel    1 lambda   0.470000 stress 0.045397 penalty 0.003133
## itel    1 lambda   0.480000 stress 0.045660 penalty 0.002751
## itel    1 lambda   0.490000 stress 0.045902 penalty 0.002398
## itel    1 lambda   0.500000 stress 0.046124 penalty 0.002071
## itel    1 lambda   0.510000 stress 0.046325 penalty 0.001769
## itel    1 lambda   0.520000 stress 0.046506 penalty 0.001492
## itel    1 lambda   0.530000 stress 0.046668 penalty 0.001240
## itel    1 lambda   0.540000 stress 0.046811 penalty 0.001016
## itel    1 lambda   0.550000 stress 0.046934 penalty 0.000819
## itel    1 lambda   0.560000 stress 0.047040 penalty 0.000651
## itel    1 lambda   0.570000 stress 0.047128 penalty 0.000509
## itel    1 lambda   0.580000 stress 0.047200 penalty 0.000392
## itel    1 lambda   0.590000 stress 0.047258 penalty 0.000298
## itel    1 lambda   0.600000 stress 0.047303 penalty 0.000224
## itel   68 lambda   0.610000 stress 0.047414 penalty 0.000000
```



### 6.1.5 Dutch Political Parties

In 1967 one hundred psychology students at Leiden University judged the similarity of nine Dutch political parties, using the complete method of triads (De Gruijter (1967)). Data were aggregated and converted to dissimilarities. We first print the matrix of dissimilarities.

```
poldist <-
structure(c(5.63, 5.27, 4.6, 4.8, 7.54, 6.73, 7.18, 6.17, 6.72,
5.64, 6.22, 5.12, 4.59, 7.22, 5.47, 5.46, 4.97, 8.13, 7.55, 6.9,
4.67, 3.2, 7.84, 6.73, 7.28, 6.13, 7.8, 7.08, 6.96, 6.04, 4.08,
6.34, 7.42, 6.88, 6.36, 7.36), Labels = c("KVP", "PvdA", "VVD",
"ARP", "CHU", "CPN", "PSP", "BP", "D66"), Size = 9L,
call = quote(as.dist.default(m = polpar)),
class = "dist", Diag = FALSE, Upper = FALSE)
poldist <- as.matrix(poldist)
poldist <- 2 * poldist / sqrt(sum(poldist * poldist))
poldist <- (poldist - .1) ^ 2
diag(poldist) <- 0
poldist <- 2 * poldist / sqrt(sum(poldist * poldist))
mPrint(poldist, d = 2)
```

```
##       KVP      PvdA     VVD      ARP      CHU      CPN      PSP      BP
## KVP      +0.00    +0.13    +0.10    +0.05    +0.07    +0.35    +0.24    +0.30
## PvdA     +0.13    +0.00    +0.24    +0.13    +0.18    +0.09    +0.05    +0.30
## VVD      +0.10    +0.24    +0.00    +0.11    +0.08    +0.44    +0.35    +0.26
## ARP      +0.05    +0.13    +0.11    +0.00    +0.00    +0.39    +0.24    +0.31
## CHU      +0.07    +0.18    +0.08    +0.00    +0.00    +0.39    +0.29    +0.27
## CPN      +0.35    +0.09    +0.44    +0.39    +0.39    +0.00    +0.03    +0.20
## PSP      +0.24    +0.05    +0.35    +0.24    +0.29    +0.03    +0.00    +0.26
## BP       +0.30    +0.30    +0.26    +0.31    +0.27    +0.20    +0.26    +0.00
## D66      +0.18    +0.11    +0.06    +0.18    +0.17    +0.33    +0.20    +0.32
##       D66
## KVP      +0.18
## PvdA     +0.11
## VVD      +0.06
## ARP      +0.18
## CHU      +0.17
## CPN      +0.33
## PSP      +0.20
## BP       +0.32
## D66      +0.00
```

We subtracted a constant from all off-diagonal dissimilarities. This is mainly because these data, being averages over a heterogeneous group of individuals, regress to the mean and thus have a substantial additive constant.

Using the Torgerson initial configuration in two dimensions, smacof finds a stress of 0.0272187083 after 97 iterations.

A full-dimensional scaling with $p = 8$ gives stress 0.0218562551 after 68 iterations. The singular values of the full dimensional solution are

```
## [1] +0.389414 +0.216045 +0.132915 +0.037145 +0.000000 +0.000000 +0.000000
## [8] +0.000000
```

and thus the Gower rank of these (transformed) data is four.

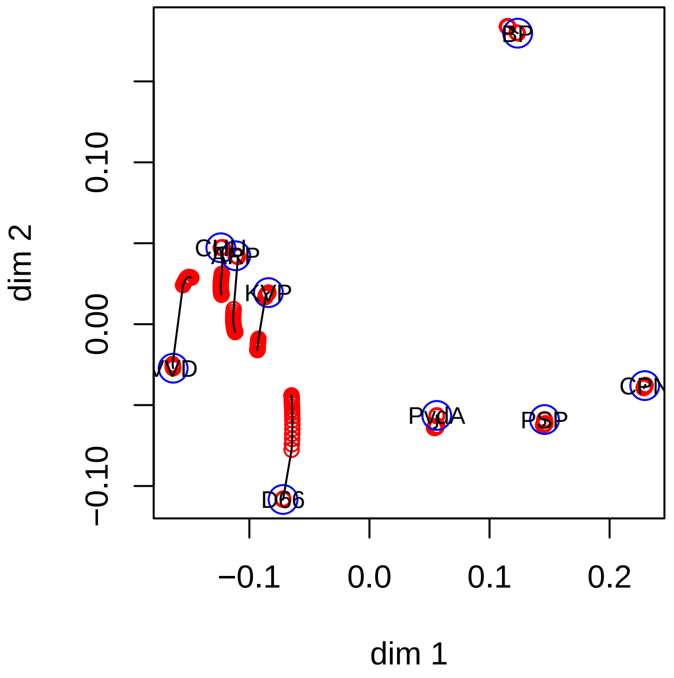
For FDS(2) stress is 0.0272187083 after 78 iterations, the same solution as with the Torgerson start.

```
##
## 0.0272187 0.0301047
##      5557      4443
```

If we run smacof 10,000 times with a random start we find two different local minima, and the one with the smallest stress is found in about 55% of the cases.

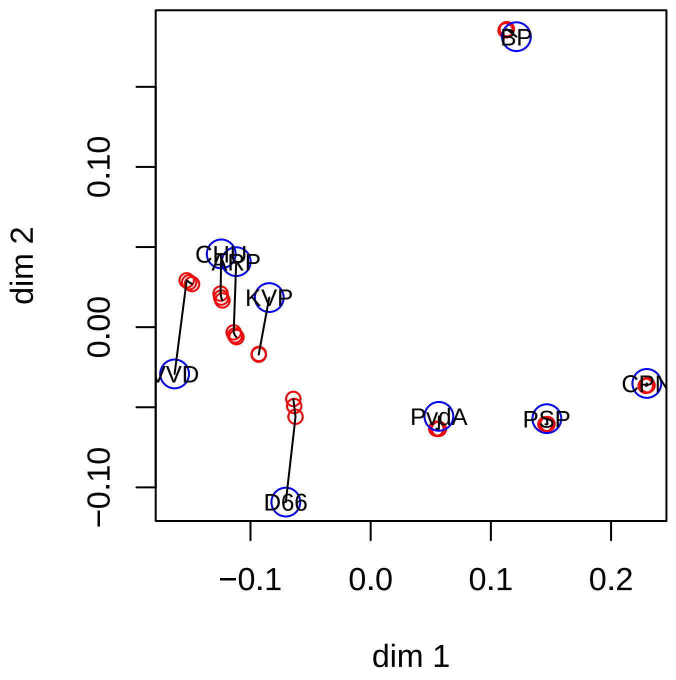
Our global MDS method with the default $\lambda$ sequence also converges to the same smallest local minimum.

```
## itel   80 lambda   0.000000 stress 0.021856 penalty 0.453217
## itel    2 lambda   0.010000 stress 0.021865 penalty 0.041599
## itel    2 lambda   0.020000 stress 0.021899 penalty 0.040014
## itel    1 lambda   0.030000 stress 0.021938 penalty 0.038919
## itel    1 lambda   0.040000 stress 0.022002 penalty 0.037599
## itel    1 lambda   0.050000 stress 0.022091 penalty 0.036178
## itel    1 lambda   0.060000 stress 0.022205 penalty 0.034717
## itel    1 lambda   0.070000 stress 0.022342 penalty 0.033247
## itel    1 lambda   0.080000 stress 0.022499 penalty 0.031782
## itel    1 lambda   0.090000 stress 0.022675 penalty 0.030330
## itel    1 lambda   0.100000 stress 0.022868 penalty 0.028890
## itel    1 lambda   0.110000 stress 0.023077 penalty 0.027462
## itel    1 lambda   0.120000 stress 0.023299 penalty 0.026041
## itel    1 lambda   0.130000 stress 0.023532 penalty 0.024623
## itel    1 lambda   0.140000 stress 0.023776 penalty 0.023204
## itel    1 lambda   0.150000 stress 0.024028 penalty 0.021779
## itel    1 lambda   0.160000 stress 0.024286 penalty 0.020344
## itel    1 lambda   0.170000 stress 0.024548 penalty 0.018899
## itel    1 lambda   0.180000 stress 0.024811 penalty 0.017442
## itel   31 lambda   0.190000 stress 0.026907 penalty 0.001050
## itel    1 lambda   0.200000 stress 0.026926 penalty 0.000956
## itel    1 lambda   0.210000 stress 0.026948 penalty 0.000864
## itel    1 lambda   0.220000 stress 0.026971 penalty 0.000773
## itel    1 lambda   0.230000 stress 0.026995 penalty 0.000685
## itel    1 lambda   0.240000 stress 0.027018 penalty 0.000600
## itel    1 lambda   0.250000 stress 0.027042 penalty 0.000521
## itel    1 lambda   0.260000 stress 0.027064 penalty 0.000448
## itel    1 lambda   0.270000 stress 0.027084 penalty 0.000382
## itel    1 lambda   0.280000 stress 0.027104 penalty 0.000322
## itel    1 lambda   0.290000 stress 0.027121 penalty 0.000269
## itel    1 lambda   0.300000 stress 0.027137 penalty 0.000222
## itel    1 lambda   0.310000 stress 0.027152 penalty 0.000181
## itel    1 lambda   0.320000 stress 0.027164 penalty 0.000147
## itel    1 lambda   0.330000 stress 0.027175 penalty 0.000117
## itel    1 lambda   0.340000 stress 0.027184 penalty 0.000092
## itel    1 lambda   0.350000 stress 0.027191 penalty 0.000072
## itel    1 lambda   0.360000 stress 0.027197 penalty 0.000055
## itel    1 lambda   0.370000 stress 0.027202 penalty 0.000042
## itel    1 lambda   0.380000 stress 0.027206 penalty 0.000032
## itel    6 lambda   0.390000 stress 0.027217 penalty 0.000005
## itel   18 lambda   0.400000 stress 0.027219 penalty 0.000000
## itel    2 lambda   0.410000 stress 0.027219 penalty 0.000000
## itel    2 lambda   0.420000 stress 0.027219 penalty 0.000000
```



We find the same results with a much shorter sequence of $\lambda$ values.

```
## itel   80 lambda   0.000000 stress 0.021856 penalty 0.453217
## itel    3 lambda   0.111111 stress 0.022908 penalty 0.029330
## itel    2 lambda   0.222222 stress 0.024912 penalty 0.020271
## itel   71 lambda   0.333333 stress 0.027219 penalty 0.000000
```



### 6.1.6 Ekman

The next example analyzes dissimilarities between 14 colors, taken from Ekman (1954). The original similarities $s_{ij}$, averaged over 31 subjects, were transformed to dissimilarities by $\delta_{ij} = 1 - s_{ij}$.

```
##      434      445      465      472      490      504      537      555
## 434    +0.00    +0.03    +0.10    +0.10    +0.15    +0.17    +0.17    +0.17
## 445    +0.03    +0.00    +0.09    +0.10    +0.14    +0.16    +0.17    +0.17
## 465    +0.10    +0.09    +0.00    +0.03    +0.10    +0.15    +0.16    +0.17
## 472    +0.10    +0.10    +0.03    +0.00    +0.08    +0.14    +0.16    +0.16
## 490    +0.15    +0.14    +0.10    +0.08    +0.00    +0.07    +0.12    +0.13
## 504    +0.17    +0.16    +0.15    +0.14    +0.07    +0.00    +0.07    +0.10
## 537    +0.17    +0.17    +0.16    +0.16    +0.12    +0.07    +0.00    +0.05
## 555    +0.17    +0.17    +0.17    +0.16    +0.13    +0.10    +0.05    +0.00
## 584    +0.18    +0.18    +0.18    +0.18    +0.17    +0.16    +0.14    +0.12
## 600    +0.17    +0.17    +0.18    +0.18    +0.18    +0.17    +0.16    +0.15
## 610    +0.16    +0.17    +0.18    +0.18    +0.18    +0.18    +0.17    +0.17
## 628    +0.16    +0.16    +0.18    +0.18    +0.18    +0.18    +0.18    +0.18
## 651    +0.16    +0.16    +0.17    +0.18    +0.18    +0.18    +0.18    +0.18
## 674    +0.15    +0.16    +0.18    +0.17    +0.18    +0.18    +0.18    +0.18
##      584      600      610      628      651      674
## 434    +0.18    +0.17    +0.16    +0.16    +0.16    +0.15
## 445    +0.18    +0.17    +0.17    +0.16    +0.16    +0.16
## 465    +0.18    +0.18    +0.18    +0.18    +0.17    +0.18
## 472    +0.18    +0.18    +0.18    +0.18    +0.18    +0.17
## 490    +0.17    +0.18    +0.18    +0.18    +0.18    +0.18
## 504    +0.16    +0.17    +0.18    +0.18    +0.18    +0.18
## 537    +0.14    +0.16    +0.17    +0.18    +0.18    +0.18
## 555    +0.12    +0.15    +0.17    +0.18    +0.18    +0.18
## 584    +0.00    +0.08    +0.11    +0.13    +0.14    +0.14
## 600    +0.08    +0.00    +0.05    +0.09    +0.11    +0.13
## 610    +0.11    +0.05    +0.00    +0.04    +0.07    +0.08
## 628    +0.13    +0.09    +0.04    +0.00    +0.03    +0.06
## 651    +0.14    +0.11    +0.07    +0.03    +0.00    +0.04
## 674    +0.14    +0.13    +0.08    +0.06    +0.04    +0.00
```

Using the Torgerson initial configuration in two dimensions, smacof finds a stress of 0.0172132469 after 25 iterations.

A FDS with $p = 13$ gives stress $8.7569296565 \times 10^{-5}$ after 1204 iterations. The singular values of the full dimensional solution are

```
##  [1] +0.254220 +0.205722 +0.119338 +0.109900 +0.068741 +0.055643 +0.033137
##  [8] +0.022797 +0.011573 +0.001704 +0.000315 +0.000000 +0.000000
```

and thus the Gower rank of these data is eleven.

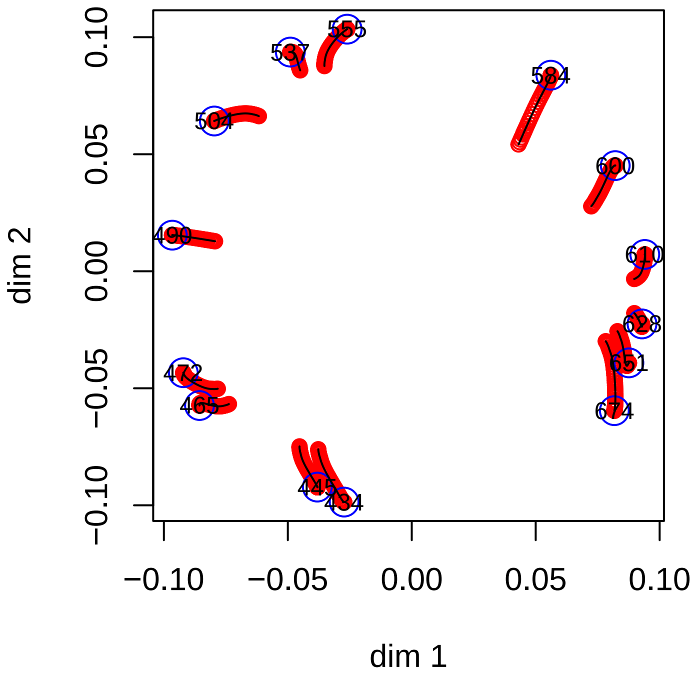
FDS(2) gives stress 0.0172132469 after 25 iterations, the same solution as with the Torgerson start.

```
##
## 0.0172132  0.030068 0.0365429 0.0601969 0.0619843  0.063479 0.0662027 0.0674562
##      8234         1      1353       186        94        43        36        16
## 0.0678714 0.0758563 0.0759964 0.0774167 0.0775059
##        20        12         1         3         1
```

If we run smacof 10,000 times with a random start we find ten different local minima, and the one with the smallest stress 0.0172132468 is found in about 80% of the cases.

Our global optimization method with the default $\lambda$ sequence also converges to the same smallest local minimum.

```
## itel 1482 lambda   0.000000 stress 0.000088 penalty 0.426110
## itel    5 lambda   0.010000 stress 0.000132 penalty 0.118988
## itel    3 lambda   0.020000 stress 0.000253 penalty 0.112777
## itel    2 lambda   0.030000 stress 0.000431 penalty 0.107184
## itel    2 lambda   0.040000 stress 0.000715 penalty 0.100691
## itel    1 lambda   0.050000 stress 0.000942 penalty 0.096656
## itel    1 lambda   0.060000 stress 0.001246 penalty 0.091999
## itel    1 lambda   0.070000 stress 0.001627 penalty 0.086941
## itel    1 lambda   0.080000 stress 0.002082 penalty 0.081645
## itel    1 lambda   0.090000 stress 0.002607 penalty 0.076232
## itel    1 lambda   0.100000 stress 0.003195 penalty 0.070791
## itel    1 lambda   0.110000 stress 0.003840 penalty 0.065388
## itel    1 lambda   0.120000 stress 0.004534 penalty 0.060073
## itel    1 lambda   0.130000 stress 0.005267 penalty 0.054886
## itel    1 lambda   0.140000 stress 0.006033 penalty 0.049859
## itel    1 lambda   0.150000 stress 0.006820 penalty 0.045021
## itel    1 lambda   0.160000 stress 0.007622 penalty 0.040395
## itel    1 lambda   0.170000 stress 0.008427 penalty 0.036003
## itel    1 lambda   0.180000 stress 0.009228 penalty 0.031864
## itel    1 lambda   0.190000 stress 0.010014 penalty 0.027995
## itel    1 lambda   0.200000 stress 0.010778 penalty 0.024407
## itel    1 lambda   0.210000 stress 0.011510 penalty 0.021111
## itel    1 lambda   0.220000 stress 0.012203 penalty 0.018110
## itel    1 lambda   0.230000 stress 0.012852 penalty 0.015406
## itel    1 lambda   0.240000 stress 0.013451 penalty 0.012993
## itel    1 lambda   0.250000 stress 0.013997 penalty 0.010863
## itel    1 lambda   0.260000 stress 0.014489 penalty 0.009003
## itel    1 lambda   0.270000 stress 0.014926 penalty 0.007396
## itel    2 lambda   0.280000 stress 0.015618 penalty 0.004939
## itel    1 lambda   0.290000 stress 0.015890 penalty 0.004012
## itel    1 lambda   0.300000 stress 0.016125 penalty 0.003230
## itel    2 lambda   0.310000 stress 0.016484 penalty 0.002072
## itel    1 lambda   0.320000 stress 0.016620 penalty 0.001651
## itel    1 lambda   0.330000 stress 0.016734 penalty 0.001305
## itel    2 lambda   0.340000 stress 0.016904 penalty 0.000807
## itel    1 lambda   0.350000 stress 0.016966 penalty 0.000632
## itel    1 lambda   0.360000 stress 0.017017 penalty 0.000491
## itel    1 lambda   0.370000 stress 0.017059 penalty 0.000378
## itel    1 lambda   0.380000 stress 0.017093 penalty 0.000289
## itel    1 lambda   0.390000 stress 0.017121 penalty 0.000219
## itel    1 lambda   0.400000 stress 0.017142 penalty 0.000165
## itel    2 lambda   0.410000 stress 0.017172 penalty 0.000092
## itel    1 lambda   0.420000 stress 0.017183 penalty 0.000068
## itel    1 lambda   0.430000 stress 0.017190 penalty 0.000050
## itel    1 lambda   0.440000 stress 0.017196 penalty 0.000037
## itel    2 lambda   0.450000 stress 0.017204 penalty 0.000019
## itel    2 lambda   0.460000 stress 0.017209 penalty 0.000010
## itel    2 lambda   0.470000 stress 0.017211 penalty 0.000005
## itel    3 lambda   0.480000 stress 0.017212 penalty 0.000002
## itel    4 lambda   0.490000 stress 0.017213 penalty 0.000000
## itel    4 lambda   0.500000 stress 0.017213 penalty 0.000000
## itel    4 lambda   0.510000 stress 0.017213 penalty 0.000000
## itel    2 lambda   0.520000 stress 0.017213 penalty 0.000000
```

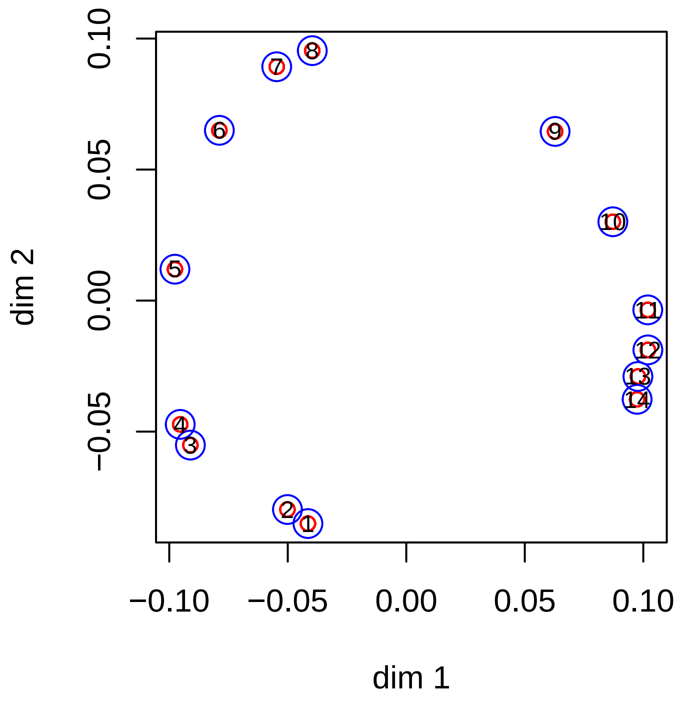


If we transform the Ekman similarities by $\delta_{ij}=(1-s_{ij})^3$ then it is known (De Leeuw (2016)) that the Gower rank of the transformed data is equal to two. Thus the FDS solution has rank 2, and the 2MDS solution is always the global minimum.

```
data(ekman, package = "smacof")
okman <- as.matrix((1 - ekman) ^ 3)
okman <- 2 * okman / sqrt(sum(okman * okman))
lbd <- seq(0, 1, length = 101)
hOkman <- runPenalty(okman, lbd = lbd, cut = 1e-10, write = FALSE)
writeSelected(hOkman, 1:length(hOkman))
```

```
## itel   99 lambda   0.000000 stress 0.011025 penalty 0.433456
## itel    1 lambda   0.010000 stress 0.011025 penalty 0.000000
## itel    1 lambda   0.020000 stress 0.011025 penalty 0.000000
## itel    1 lambda   0.030000 stress 0.011025 penalty 0.000000
## itel    1 lambda   0.040000 stress 0.011025 penalty 0.000000
## itel    1 lambda   0.050000 stress 0.011025 penalty 0.000000
## itel    1 lambda   0.060000 stress 0.011025 penalty 0.000000
## itel    1 lambda   0.070000 stress 0.011025 penalty 0.000000
## itel    1 lambda   0.080000 stress 0.011025 penalty 0.000000
## itel    1 lambda   0.090000 stress 0.011025 penalty 0.000000
## itel    1 lambda   0.100000 stress 0.011025 penalty 0.000000
## itel    1 lambda   0.110000 stress 0.011025 penalty 0.000000
## itel    1 lambda   0.120000 stress 0.011025 penalty 0.000000
## itel    1 lambda   0.130000 stress 0.011025 penalty 0.000000
```

```
plotMe2(hOkman, dimnames(ekman)[[1]])
```



### 6.1.7 Morse in Two

Next, we use dissimilarities between 36 Morse code signals (Rothkopf (1957)). We used the symmetrized version `morse` from the `smacof` package (De Leeuw and Mair (2009)).

Using the Torgerson initial configuration in two dimensions, smacof finds a stress of 0.0899492031 after 238 iterations.

A full-dimensional scaling with $p = 31$ gives stress $7.6345177128 \times 10^{-4}$ after 1347 iterations. The singular values of the full dimensional solution are

```
##  [1] +0.100442 +0.090156 +0.078222 +0.067738 +0.061768 +0.060456 +0.055417
##  [8] +0.049762 +0.047491 +0.045725 +0.044117 +0.039553 +0.036242 +0.033715
## [15] +0.032183 +0.025291 +0.022152 +0.019617 +0.018309 +0.013419 +0.012242
## [22] +0.006957 +0.000575 +0.000071 +0.000014 +0.000000 +0.000000 +0.000000
## [29] +0.000000 +0.000000 +0.000000 +0.000000 +0.000000 +0.000000 +0.000000
```

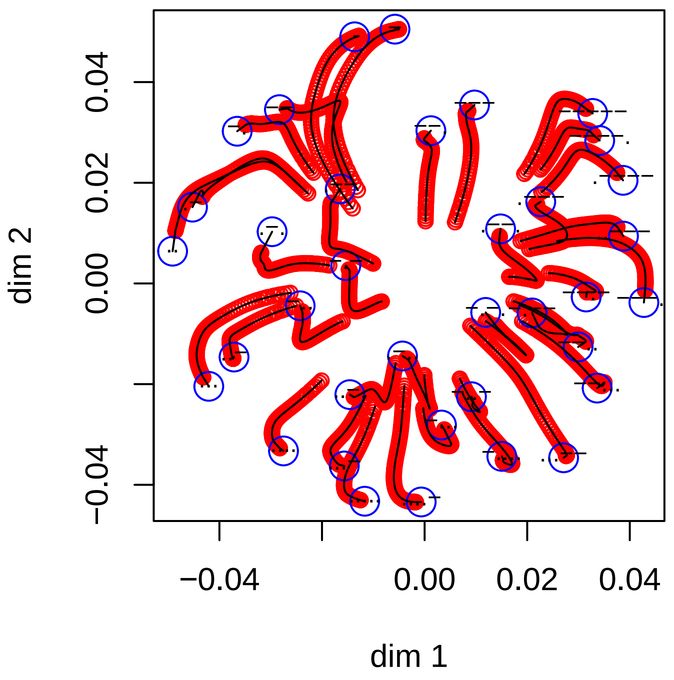
and thus the Gower rank of these (transformed) data is 25.

In FDS(2) stress is 0.0899492031 after 238 iterations, the same solution as with the Torgerson start.

If we run smacof 10,000 times with a random start we find a huge number of local minima. It is difficult to draw the line on which local minima are the same and which are different, but our usual procedure says there are 4070 of them. It would be silly to print them out here. Suffices it to say that the smallest stress is 0.0899492021, which is found in only about 7% of the cases. It is the same as stress for the previous Torgerson and FDS(1) solutions.

Our global optimization method with the default $\lambda$ sequence also converges to the same smallest local minimum.

```
## itel 1461 lambda   0.000000 stress 0.000763 penalty 0.472254
## itel    6 lambda   0.010000 stress 0.000858 penalty 0.322181
## itel    4 lambda   0.020000 stress 0.001147 penalty 0.308335
## itel    3 lambda   0.030000 stress 0.001622 penalty 0.294777
## itel    2 lambda   0.040000 stress 0.002207 penalty 0.283003
## itel    2 lambda   0.050000 stress 0.003018 penalty 0.269965
## itel    1 lambda   0.060000 stress 0.003720 penalty 0.261134
## itel    2 lambda   0.070000 stress 0.005058 penalty 0.245658
## itel    1 lambda   0.080000 stress 0.006058 penalty 0.236320
## itel    1 lambda   0.090000 stress 0.007232 penalty 0.226392
## itel    1 lambda   0.100000 stress 0.008576 penalty 0.216089
## itel    1 lambda   0.110000 stress 0.010085 penalty 0.205587
## itel    1 lambda   0.120000 stress 0.011750 penalty 0.195030
## itel    1 lambda   0.130000 stress 0.013559 penalty 0.184543
## itel    1 lambda   0.140000 stress 0.015500 penalty 0.174225
## itel    1 lambda   0.150000 stress 0.017554 penalty 0.164160
## itel    1 lambda   0.160000 stress 0.019705 penalty 0.154411
## itel    1 lambda   0.170000 stress 0.021934 penalty 0.145029
## itel    1 lambda   0.180000 stress 0.024222 penalty 0.136047
## itel    1 lambda   0.190000 stress 0.026550 penalty 0.127488
## itel    1 lambda   0.200000 stress 0.028903 penalty 0.119364
## itel    1 lambda   0.210000 stress 0.031265 penalty 0.111677
## itel    1 lambda   0.220000 stress 0.033621 penalty 0.104423
## itel    1 lambda   0.230000 stress 0.035962 penalty 0.097591
## itel    1 lambda   0.240000 stress 0.038277 penalty 0.091167
## itel    1 lambda   0.250000 stress 0.040558 penalty 0.085132
## itel    1 lambda   0.260000 stress 0.042799 penalty 0.079467
## itel    1 lambda   0.270000 stress 0.044995 penalty 0.074148
## itel    1 lambda   0.280000 stress 0.047143 penalty 0.069155
## itel    1 lambda   0.290000 stress 0.049240 penalty 0.064466
## itel    1 lambda   0.300000 stress 0.051285 penalty 0.060060
## itel    1 lambda   0.310000 stress 0.053275 penalty 0.055920
## itel    1 lambda   0.320000 stress 0.055211 penalty 0.052027
## itel    1 lambda   0.330000 stress 0.057092 penalty 0.048366
## itel    1 lambda   0.340000 stress 0.058916 penalty 0.044923
## itel    1 lambda   0.350000 stress 0.060683 penalty 0.041685
## itel    1 lambda   0.360000 stress 0.062393 penalty 0.038640
## itel    1 lambda   0.370000 stress 0.064045 penalty 0.035778
## itel    1 lambda   0.380000 stress 0.065639 penalty 0.033088
## itel    1 lambda   0.390000 stress 0.067175 penalty 0.030562
## itel    1 lambda   0.400000 stress 0.068653 penalty 0.028190
## itel    1 lambda   0.410000 stress 0.070072 penalty 0.025963
## itel    1 lambda   0.420000 stress 0.071433 penalty 0.023875
## itel    1 lambda   0.430000 stress 0.072735 penalty 0.021917
## itel    1 lambda   0.440000 stress 0.073980 penalty 0.020083
## itel    1 lambda   0.450000 stress 0.075168 penalty 0.018366
## itel    1 lambda   0.460000 stress 0.076298 penalty 0.016761
## itel    1 lambda   0.470000 stress 0.077371 penalty 0.015263
## itel    1 lambda   0.480000 stress 0.078389 penalty 0.013865
## itel    1 lambda   0.490000 stress 0.079351 penalty 0.012564
## itel    1 lambda   0.500000 stress 0.080258 penalty 0.011356
## itel    1 lambda   0.510000 stress 0.081112 penalty 0.010237
## itel    1 lambda   0.520000 stress 0.081912 penalty 0.009204
## itel    1 lambda   0.530000 stress 0.082661 penalty 0.008252
## itel    1 lambda   0.540000 stress 0.083360 penalty 0.007379
## itel    1 lambda   0.550000 stress 0.084009 penalty 0.006580
## itel    1 lambda   0.560000 stress 0.084610 penalty 0.005853
## itel    1 lambda   0.570000 stress 0.085165 penalty 0.005194
## itel    1 lambda   0.580000 stress 0.085676 penalty 0.004598
## itel    1 lambda   0.590000 stress 0.086144 penalty 0.004061
## itel    1 lambda   0.600000 stress 0.086572 penalty 0.003578
## itel    1 lambda   0.610000 stress 0.086962 penalty 0.003145
## itel    1 lambda   0.620000 stress 0.087316 penalty 0.002757
## itel    1 lambda   0.630000 stress 0.087637 penalty 0.002411
## itel    1 lambda   0.640000 stress 0.087927 penalty 0.002102
## itel    1 lambda   0.650000 stress 0.088188 penalty 0.001827
## itel    1 lambda   0.660000 stress 0.088423 penalty 0.001582
## itel    1 lambda   0.670000 stress 0.088633 penalty 0.001365
## itel    1 lambda   0.680000 stress 0.088822 penalty 0.001173
## itel    1 lambda   0.690000 stress 0.088990 penalty 0.001003
## itel    1 lambda   0.700000 stress 0.089140 penalty 0.000854
## itel    1 lambda   0.710000 stress 0.089273 penalty 0.000723
## itel    1 lambda   0.720000 stress 0.089390 penalty 0.000608
## itel    1 lambda   0.730000 stress 0.089493 penalty 0.000509
## itel    1 lambda   0.740000 stress 0.089583 penalty 0.000422
## itel    1 lambda   0.750000 stress 0.089660 penalty 0.000348
## itel    1 lambda   0.760000 stress 0.089726 penalty 0.000284
## itel    1 lambda   0.770000 stress 0.089782 penalty 0.000230
## itel    1 lambda   0.780000 stress 0.089828 penalty 0.000185
## itel    1 lambda   0.790000 stress 0.089867 penalty 0.000147
## itel    1 lambda   0.800000 stress 0.089898 penalty 0.000116
## itel    1 lambda   0.810000 stress 0.089923 penalty 0.000090
## itel    1 lambda   0.820000 stress 0.089943 penalty 0.000070
## itel    1 lambda   0.830000 stress 0.089958 penalty 0.000053
## itel    1 lambda   0.840000 stress 0.089970 penalty 0.000040
## itel  197 lambda   0.850000 stress 0.089949 penalty 0.000000
```



## 6.2 One-Dimensional Examples

### 6.2.1 Vegetables

Our first one-dimensional example uses paired comparisons of nine vegetables, originating with Guilford (1954) and taken from the `psychTools` package (Revelle (2023)). The proportions are transformed to dissimilarities by using the normal quantile function, i.e. $\delta_{ij} = |\Phi^{-1}(p_{ij})|$. One-dimensional MDS of these transformed data is another way of running a Thurstone Case V paired comparison analysis.

```
data(vegetables, package = "psychTools")
veg <- abs(qnorm(as.matrix(veg)))
veg <- 2 * veg / sqrt(sum(veg * veg))
mPrint(veg, d = 2)
```

```
##           Turn      Cab       Beet      Asp       Car       Spin      S.Beans  Peas
## Turn         +0.00     +0.29     +0.23     +0.28     +0.37     +0.39     +0.40     +0.39
## Cab          +0.29     +0.00     +0.08     +0.19     +0.21     +0.20     +0.28     +0.32
## Beet         +0.23     +0.08     +0.00     +0.05     +0.20     +0.14     +0.32     +0.26
## Asp          +0.28     +0.19     +0.05     +0.00     +0.05     +0.07     +0.14     +0.08
## Car          +0.37     +0.21     +0.20     +0.05     +0.00     +0.01     +0.06     +0.17
## Spin         +0.39     +0.20     +0.14     +0.07     +0.01     +0.00     +0.10     +0.15
## S.Beans      +0.40     +0.28     +0.32     +0.14     +0.06     +0.10     +0.00     +0.02
## Peas         +0.39     +0.32     +0.26     +0.08     +0.17     +0.15     +0.02     +0.00
## Corn         +0.46     +0.34     +0.29     +0.19     +0.23     +0.10     +0.12     +0.10
##           Corn
## Turn         +0.46
## Cab          +0.34
## Beet       +0.29
## Asp        +0.19
## Car        +0.23
## Spin       +0.10
## S.Beans    +0.12
## Peas       +0.10
## Corn       +0.00
```

For the unidimensional scalings we use the uniMDS() function in the appendix, which is just smacof with the simplifications that are possible because of one-dimensionality.

Before we start the global smacof algorithm, we will look in more detail at all local minima. Note that there is a maximum of $3.6288 \times 10^5$ local minima, and we can enumerate them all by investigating all permutations of the 9 vegetables. We use the necessary and sufficient condition for a local minimum, which is that all $x_i$ are different and that

$$x_i = \frac{1}{n}\sum_{j=1}^{n}\delta_{ij}\ \mathbf{sign}(x_i - x_j)$$

for all $i$ (De Leeuw (2023a)).

Using the Torgerson initial configuration in one dimension, smacof finds a stress of 0.0353011713 after 3 iterations.

A full-dimensional scaling with $p = 8$ gives stress 0.0136747248 after 1378 iterations. The singular values of the full dimensional solution are

```
## [1] +0.406637 +0.209179 +0.100334 +0.002692 +0.000000 +0.000000 +0.000000
## [8] +0.000000
```

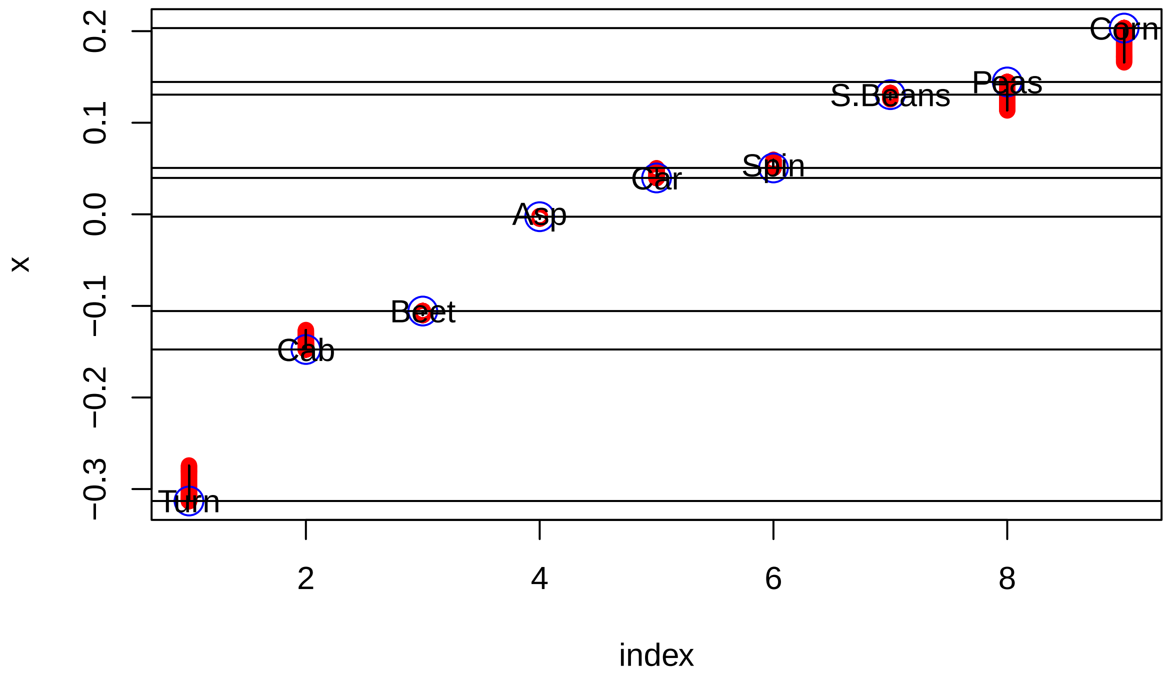
and thus the Gower rank of the (transformed) data is 4.

For FDS(1) stress is 0.0353011713 after 3 iterations, the same solution as with the Torgerson start.

If we run smacof 10,000 times with a random start we find a huge number of local minima. It is difficult to decide which local minima are the same and which are different, but our usual procedure says there are 3283 different ones. It would be silly to print them all out here. Suffices it to say that the smallest stress is 0.0353011713, which is found in 896 of the 10,000 cases. It is the same as stress for the previous Torgerson and FDS(1) solutions.

Our global optimization method with the default $\lambda$ sequence also converges to the same smallest local minimum. Note that the unidimensional plots (function plotMe1()) are different from the two-dimensional ones (function plotMe2()), because they show the movement for different lambda values on the vertical axes.

```
## itel 1412 lambda   0.000000 stress 0.013675 penalty 0.269308
## itel    5 lambda   0.010000 stress 0.013716 penalty 0.114786
## itel    3 lambda   0.020000 stress 0.013831 penalty 0.108848
## itel    2 lambda   0.030000 stress 0.014001 penalty 0.103509
## itel    2 lambda   0.040000 stress 0.014271 penalty 0.097345
## itel    1 lambda   0.050000 stress 0.014489 penalty 0.093513
## itel    1 lambda   0.060000 stress 0.014782 penalty 0.089107
## itel    1 lambda   0.070000 stress 0.015148 penalty 0.084346
## itel    1 lambda   0.080000 stress 0.015587 penalty 0.079388
## itel    1 lambda   0.090000 stress 0.016095 penalty 0.074352
## itel    1 lambda   0.100000 stress 0.016666 penalty 0.069327
## itel    1 lambda   0.110000 stress 0.017295 penalty 0.064380
## itel    1 lambda   0.120000 stress 0.017973 penalty 0.059564
## itel    1 lambda   0.130000 stress 0.018693 penalty 0.054914
## itel    1 lambda   0.140000 stress 0.019446 penalty 0.050458
## itel    1 lambda   0.150000 stress 0.020224 penalty 0.046215
## itel    1 lambda   0.160000 stress 0.021019 penalty 0.042197
## itel    1 lambda   0.170000 stress 0.021824 penalty 0.038412
## itel    1 lambda   0.180000 stress 0.022630 penalty 0.034863
## itel    1 lambda   0.190000 stress 0.023432 penalty 0.031550
## itel    1 lambda   0.200000 stress 0.024224 penalty 0.028472
## itel    1 lambda   0.210000 stress 0.025000 penalty 0.025622
## itel    1 lambda   0.220000 stress 0.025755 penalty 0.022994
## itel    1 lambda   0.230000 stress 0.026487 penalty 0.020581
## itel    1 lambda   0.240000 stress 0.027191 penalty 0.018373
## itel    1 lambda   0.250000 stress 0.027865 penalty 0.016360
## itel    1 lambda   0.260000 stress 0.028506 penalty 0.014530
## itel    1 lambda   0.270000 stress 0.029114 penalty 0.012872
## itel    1 lambda   0.280000 stress 0.029687 penalty 0.011375
## itel    1 lambda   0.290000 stress 0.030225 penalty 0.010027
## itel    1 lambda   0.300000 stress 0.030728 penalty 0.008815
## itel    1 lambda   0.310000 stress 0.031196 penalty 0.007730
## itel    1 lambda   0.320000 stress 0.031630 penalty 0.006761
## itel    1 lambda   0.330000 stress 0.032030 penalty 0.005896
## itel    1 lambda   0.340000 stress 0.032399 penalty 0.005127
## itel    1 lambda   0.350000 stress 0.032736 penalty 0.004445
## itel    1 lambda   0.360000 stress 0.033044 penalty 0.003840
## itel    1 lambda   0.370000 stress 0.033324 penalty 0.003307
## itel    1 lambda   0.380000 stress 0.033578 penalty 0.002836
## itel    1 lambda   0.390000 stress 0.033806 penalty 0.002423
## itel    1 lambda   0.400000 stress 0.034011 penalty 0.002060
## itel    1 lambda   0.410000 stress 0.034195 penalty 0.001744
## itel    1 lambda   0.420000 stress 0.034358 penalty 0.001469
## itel    1 lambda   0.430000 stress 0.034502 penalty 0.001230
## itel    1 lambda   0.440000 stress 0.034629 penalty 0.001024
## itel    1 lambda   0.450000 stress 0.034739 penalty 0.000847
## itel    1 lambda   0.460000 stress 0.034835 penalty 0.000696
## itel    1 lambda   0.470000 stress 0.034918 penalty 0.000567
## itel    1 lambda   0.480000 stress 0.034989 penalty 0.000459
## itel    1 lambda   0.490000 stress 0.035049 penalty 0.000369
## itel    1 lambda   0.500000 stress 0.035099 penalty 0.000293
## itel    1 lambda   0.510000 stress 0.035141 penalty 0.000231
## itel    1 lambda   0.520000 stress 0.035175 penalty 0.000181
## itel    1 lambda   0.530000 stress 0.035204 penalty 0.000140
## itel    1 lambda   0.540000 stress 0.035226 penalty 0.000107
## itel    1 lambda   0.550000 stress 0.035244 penalty 0.000081
## itel    1 lambda   0.560000 stress 0.035258 penalty 0.000061
## itel    1 lambda   0.570000 stress 0.035269 penalty 0.000045
## itel    1 lambda   0.580000 stress 0.035278 penalty 0.000033
## itel    1 lambda   0.590000 stress 0.035284 penalty 0.000024
## itel    1 lambda   0.600000 stress 0.035289 penalty 0.000017
## itel    1 lambda   0.610000 stress 0.035293 penalty 0.000012
## itel    1 lambda   0.620000 stress 0.035295 penalty 0.000009
## itel    1 lambda   0.630000 stress 0.035297 penalty 0.000006
## itel    1 lambda   0.640000 stress 0.035298 penalty 0.000004
## itel    1 lambda   0.650000 stress 0.035299 penalty 0.000003
## itel    1 lambda   0.660000 stress 0.035300 penalty 0.000002
## itel    1 lambda   0.670000 stress 0.035300 penalty 0.000001
## itel    1 lambda   0.680000 stress 0.035301 penalty 0.000001
## itel    1 lambda   0.690000 stress 0.035301 penalty 0.000001
## itel    1 lambda   0.700000 stress 0.035301 penalty 0.000000
## itel    1 lambda   0.710000 stress 0.035301 penalty 0.000000
## itel    4 lambda   0.720000 stress 0.035301 penalty 0.000000
## itel    3 lambda   0.730000 stress 0.035301 penalty 0.000000
```



The standard short sequence gives the same result as before.

```
## itel 1412 lambda   0.000000 stress 0.013675 penalty 0.269308
## itel    5 lambda   0.010000 stress 0.013716 penalty 0.114786
## itel    5 lambda   0.100000 stress 0.016719 penalty 0.069309
## itel   23 lambda   1.000000 stress 0.035301 penalty 0.000000
```

We can also find the exact global minimum, as in De Leeuw (2005), using enumeration of all permutations. The function uniCheck() in the Appendix found 15484 isolated local minima, and a global minimum with stress 0.0353011713, as before.

### 6.2.2 Plato

Mair, Groenen, and De Leeuw (2022) use seriation of the works of Plato, from the data collected by Cox and Brandwood (1959), as an example of unidimensional scaling.

```
data(Plato7, package = "smacof")
plato <- as.matrix(dist(t(Plato7)))
plato <- 2 * plato / sqrt(sum(plato * plato))
mPrint(plato)
```

```
##           Republic  Laws      Critias   Philebus  Politicus Sophist   Timaeus
## Republic  +0.000000 +0.367588 +0.320948 +0.378709 +0.345725 +0.257151 +0.266143
## Laws      +0.367588 +0.000000 +0.353354 +0.192721 +0.257011 +0.316831 +0.389777
## Critias   +0.320948 +0.353354 +0.000000 +0.351879 +0.358141 +0.280729 +0.284928
## Philebus  +0.378709 +0.192721 +0.351879 +0.000000 +0.209307 +0.301852 +0.367112
## Politicus +0.345725 +0.257011 +0.358141 +0.209307 +0.000000 +0.231200 +0.336223
## Sophist   +0.257151 +0.316831 +0.280729 +0.301852 +0.231200 +0.000000 +0.182571
## Timaeus   +0.266143 +0.389777 +0.284928 +0.367112 +0.336223 +0.182571 +0.000000
```

Let's start this time by finding the exact global minimum, using enumeration of the 5040 permutations. We find 5016 isolated local minima and a smallest stress of 0.1287689224. The solution that gives the global minimum is

```
##        Critias       Republic        Timaeus        Sophist      Politicus
## -0.19697758942 -0.13075087684 -0.07319698422 -0.01307457977  0.08131429066
##       Philebus           Laws
##  0.14305164484  0.18963409476
```

Note that Cox and Brandwood (1959) (p. 199) say

> The final ordering is Rep., Tim., Soph., Crit., Pol., Phil., Laws: there is reasonably strong evidence that Tim. is correctly placed before Soph., but the position of Crit. could be anywhere between somewhat before Tim. to before Pol.

Thus the global minimum of stress differs from their computed optimal order (arrived by different data analysis techniques) because it puts Criteas first. Our global minimum is the same as the one found by Mair, Groenen, and De Leeuw (2022), who used basically the same technique as we did.

Using the Torgerson initial configuration in one dimension, smacof finds a stress of

0.1436303539 after 2 iterations. The solution is

```
##        Timaeus       Republic        Critias        Sophist      Politicus
## -0.18453000695 -0.14182330689 -0.07457213666 -0.01307457977  0.08131429066
##           Laws       Philebus
##  0.15069863300  0.18198710660
```

A full-dimensional scaling with $p = 6$ gives stress $1.525336516 \times 10^{-31}$ after 1 iterations. The singular values of the full dimensional solution are

```
## [1] +0.367934 +0.238357 +0.211808 +0.155444 +0.130119 +0.086998
```

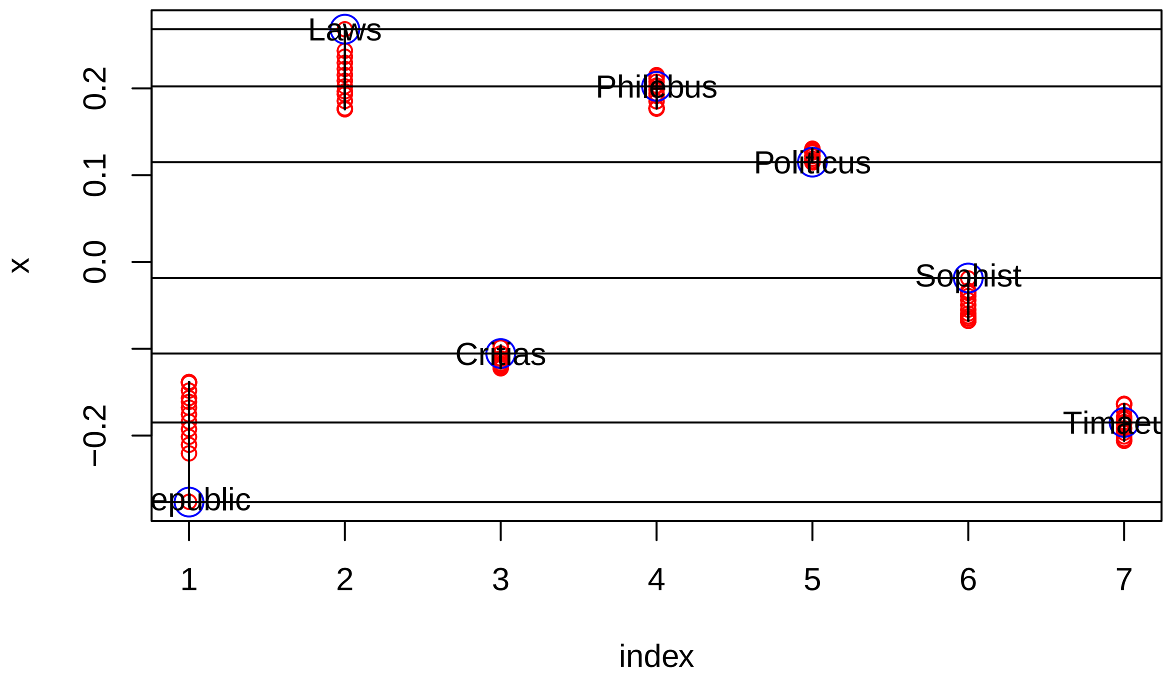
and thus the Gower rank of these data is 6.

For FDS(1) stress is 0.1436303539 after 2 iterations, the same solution as with the Torgerson start.

If we run smacof 10,000 times with a random start we find 2425 local minima. Suffices it to say that the smallest stress is 0.1287689224, which is found in only nine of the 10,000 runs. It is smaller than the stress found by the Torgerson and FDS(1) smacof solutions, and thus those methods have converged to a non-global local minimum. The local minimum found by smacof and FDS, with stress equal to 0.1436303539, is also found in nine of the 10,000 cases, same frequency as the global minimum.

Now start the global method with a relatively short $\lambda$ sequence.

```
## itel  169 lambda   0.000000 stress 0.000000 penalty 0.410927
## itel    3 lambda   0.000100 stress 0.000000 penalty 0.263015
## itel    3 lambda   0.001000 stress 0.000001 penalty 0.262280
## itel    3 lambda   0.010000 stress 0.000064 penalty 0.255078
## itel    3 lambda   0.100000 stress 0.005123 penalty 0.194945
## itel    2 lambda   0.200000 stress 0.016184 penalty 0.147493
## itel    1 lambda   0.300000 stress 0.026997 penalty 0.119323
## itel    1 lambda   0.400000 stress 0.040023 penalty 0.093615
## itel    1 lambda   0.500000 stress 0.053688 penalty 0.072330
## itel    1 lambda   0.600000 stress 0.066833 penalty 0.055452
## itel    1 lambda   0.700000 stress 0.078832 penalty 0.042269
## itel    1 lambda   0.800000 stress 0.089439 penalty 0.032019
## itel    1 lambda   0.900000 stress 0.098557 penalty 0.024079
## itel    1 lambda   1.000000 stress 0.106135 penalty 0.017940
## itel    6 lambda   2.000000 stress 0.130789 penalty 0.000148
## itel   13 lambda   3.000000 stress 0.131135 penalty 0.000000
```



Another non-global local minimum. Stress is 0.1311347419, better than the Torgerson and FDS(1) solutions, but worse than the 10,000 random starts. The order is

```
##       Republic         Timaeus          Critias          Sophist       Politicus
## -0.27660920331 -0.18492407874 -0.10546092719 -0.01849024831  0.11499577263
##       Philebus            Laws
##  0.20230557623  0.26818310868
```

This corresponds, by the way, with the order suggested by Cox and Brandwood (1959).

It may be our sequence of $\lambda$ values was not fine or long enough to do the job of finding the global minimum properly. So we try a longer and finer sequence of length 10,000, going from zero to 10. Again we find stress 0.1311347419, same as in the run with the shorter sequence. And not the global minimum. For now Plato resists our efforts to find the global minimum with our penalty technique. But we do not give up, yet.

If we transform the dissimilarities by raising them to the third power all problems magically disappear. Some may call this cheating, because we do not fitthe original data any more, and thus solve a different problem. But one which is much better behaved (same as in the Ekman example). Compare De Leeuw (2023b) for some discussion of this phenomenon.

Enumeration gives the global minimum of stress for the cubed Plato data as 0.0602806416, with global minimizer

```
##       Republic         Timaeus          Critias          Sophist       Politicus
## -0.19118756839 -0.14017846569 -0.08885229547 -0.02360731990  0.09949313875
##       Philebus            Laws
##  0.15785586601  0.18647664470
```

This is the Cox-Brandwood solution again.

Using the Torgerson initial configuration in one dimension, smacof finds a stress of 0.068458828 after 2 iterations. The solution is the non-global local minimum

```
##        Timaeus       Republic        Critias        Sophist     Politicus 
## -0.17382709262 -0.15753894146 -0.08885229547 -0.02360731990  0.09949313875 
##       Philebus           Laws 
##  0.15785586601  0.18647664470
```

A full-dimensional scaling with $p = 6$ gives stress 0.0150975 after 72 iterations. The singular values of the full dimensional solution are

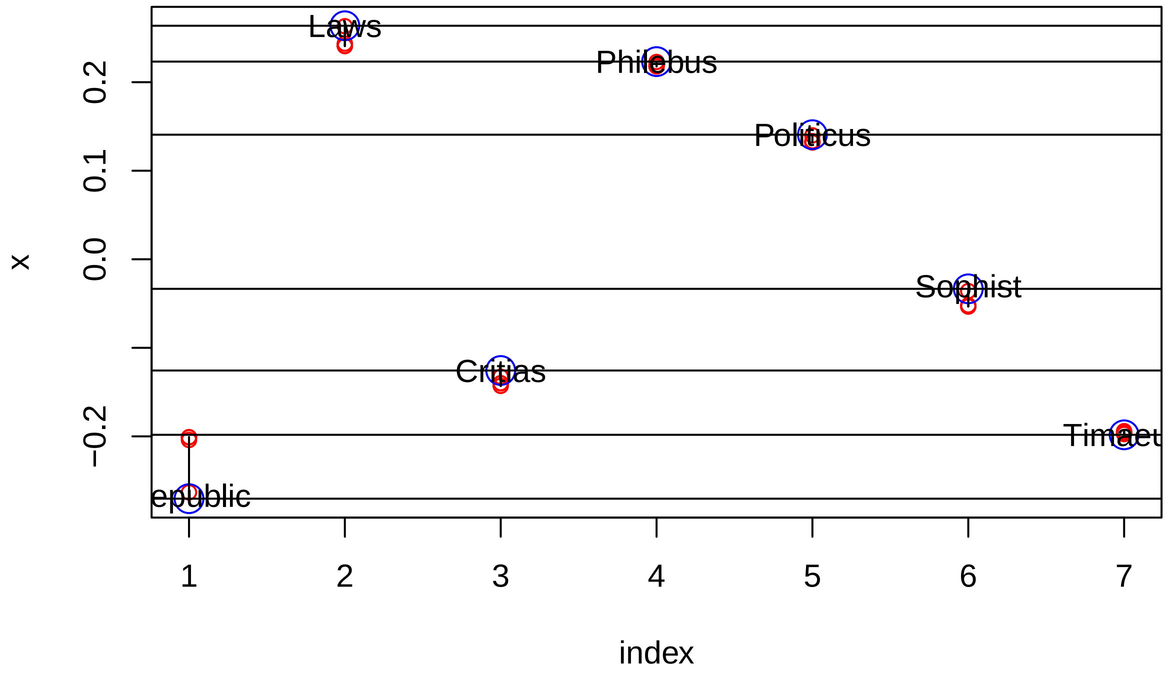
```
## [1] +0.473372 +0.210184 +0.114641 +0.000000 +0.000000 +0.000000
```

and thus the Gower rank is three.

For FDS(1) stress is 0.0602806416 after 2 iterations, which is the global minimum.

For the short sequence $\lambda = (0.000.010.10, 1.00, 10.00)$ our global MDS method quickly converges to the same global minimum.

```
## itel  120 lambda   0.000000 stress 0.015098 penalty 0.393939 
## itel    2 lambda   0.010000 stress 0.015117 penalty 0.097473 
## itel    2 lambda   0.100000 stress 0.016739 penalty 0.075516 
## itel    6 lambda   1.000000 stress 0.056064 penalty 0.003000 
## itel    7 lambda  10.000000 stress 0.060281 penalty 0.000000
```



with the Cox-Brandwood solution.

```
##   Republic    Timaeus    Critias    Sophist Politicus   Philebus       Laws 
##     -0.270     -0.198     -0.126     -0.033    +0.141     +0.223     +0.264
```

### 6.2.3 Morse in One

Now for a more challenging example. The Morse code data have been used to try out exact unidimensional MDS techniques, for example by Palubeckis (2013).

In this case there is no hope of enumerating all 371993326789901177492420297158468206329856 permutations.

Using the Torgerson initial configuration in one dimension, smacof finds a stress of 0.2513310298 after 5 iterations. The solution is

```
##              ...              ..-             ....               .-
## -0.0401975715020 -0.0372931610240 -0.0351446929991 -0.0347468285501
##               ..               -.              -..              .-.
## -0.0342030804697 -0.0308212326528 -0.0247869551756 -0.0242564692435
##             ...-            .....               --                .
## -0.0205430677191 -0.0196412416346 -0.0188322505882 -0.0161135101864
##             ..-.              .--                -              -.-
## -0.0112197774631 -0.0105036214548 -0.0096946304084 -0.0051987621342
##            ....-             -...             .-..            -....
## -0.0032492263339  0.0006365831185  0.0012996905336  0.0045356547191
##              --.             -..-            ...--              ---
##  0.0072146086760  0.0075461623836  0.0116043797639  0.0137130613438
##             -.-.             -.--             .--.            --...
##  0.0157421700339  0.0175458222030  0.0200125817870  0.0226252250024
##             --..            ..---             .---             --.-
##  0.0228506815236  0.0273598119461  0.0277842006918  0.0286595024797
##            .----            ---..            ----.            -----
##  0.0344417991391  0.0346539935120  0.0381154142187  0.0401047364639
```

A full-dimensional scaling with $p = 35$ gives stress $7.6345177128 \times 10^{-4}$ after 1347 iterations. The singular values of the full dimensional solution are

```
##  [1] +0.100442 +0.090156 +0.078222 +0.067738 +0.061768 +0.060456 +0.055417
##  [8] +0.049762 +0.047491 +0.045725 +0.044117 +0.039553 +0.036242 +0.033715
## [15] +0.032183 +0.025291 +0.022152 +0.019617 +0.018309 +0.013419 +0.012242
## [22] +0.006957 +0.000575 +0.000071 +0.000014 +0.000000 +0.000000 +0.000000
## [29] +0.000000 +0.000000 +0.000000 +0.000000 +0.000000 +0.000000 +0.000000
```

and thus the Gower rank of these data is 25.

The FDS(1) solution has stress 0.2515647508 after 5 iterations, a different solution as with the Torgerson start. The solution is

```
##              ...              ..-             ....               .-
## -0.0401975715020 -0.0372931610240 -0.0351446929991 -0.0347468285501
##               ..               -.              -..              .-.
## -0.0342030804697 -0.0308212326528 -0.0247869551756 -0.0242564692435
##             ...-            .....                .               --
## -0.0205430677191 -0.0196412416346 -0.0186067940671 -0.0163389667075
##             ..-.             -.-              .--                -
## -0.0112197774631 -0.0094691738873 -0.0086734449892 -0.0072543951209
##            ....-             -...             .-..            -....
## -0.0032492263339  0.0006365831185  0.0012996905336  0.0045356547191
##             -..-              --.            ...--              ---
##  0.0053181214689  0.0094426495907  0.0116043797639  0.0137130613438
##             -.-.             -.--             .--.            --...
##  0.0157421700339  0.0175458222030  0.0200125817870  0.0226252250024
##             --..            ..---             .---             --.-
##  0.0228506815236  0.0273598119461  0.0277842006918  0.0286595024797
##            .----            ---..            ----.            -----
##  0.0344417991391  0.0346539935120  0.0381154142187  0.0401047364639
```

If we run smacof 10,000 times with a random start we find 8977 local minima. Suffices it to say that the smallest stress is 0.2642351015. It is larger than the stress found by the Torgerson and FDS smacof solutions, and thus gives a non-global local minimum.

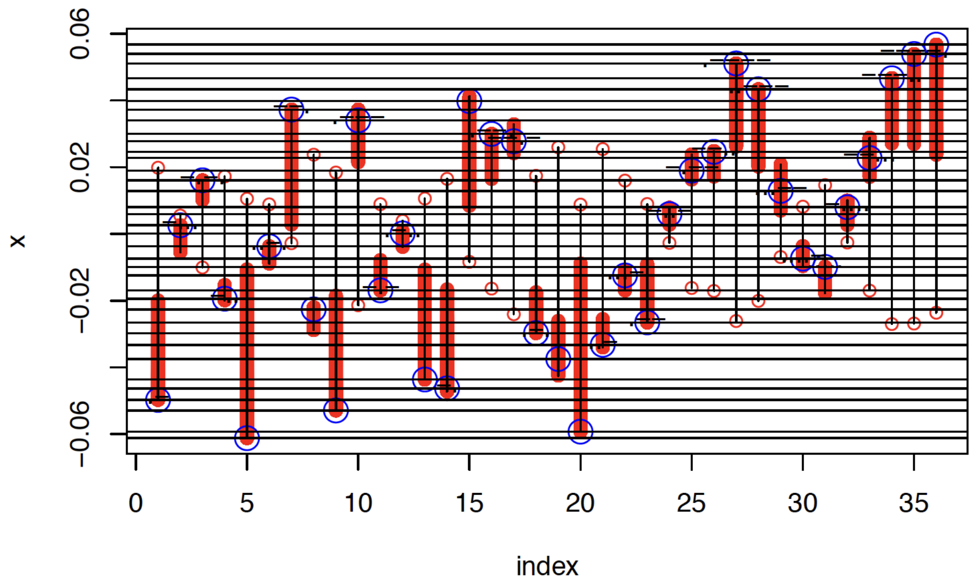
We will seriously enter the global minimum contest by using 10,000 values of $\lambda$, in an equally spaced sequence from 0 to 10. This is not as bad as it sounds. Timing of the 10,000 FDS solutions shows

```
##    user  system elapsed
##   5.825   0.283   6.110
```

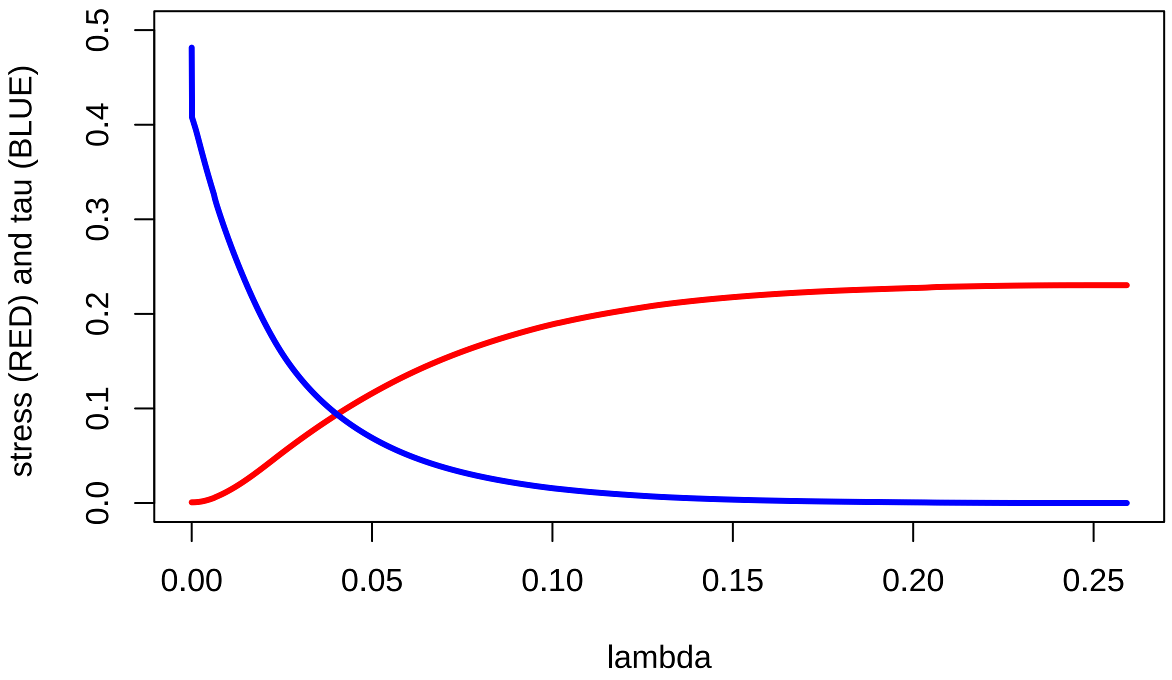
The minimum stress obtained is 0.2303106976, the best solution so far. This solution is

```
##                .                -               ..               .-
## -0.0611992291067 -0.0592486560668 -0.0529092936871 -0.0497396124968
##               -.               --              ...              ..-
## -0.0462885986556 -0.0436253162339 -0.0374922644609 -0.0332347636767
##              .-.              .--             ....              -..
## -0.0297274832756 -0.0265202908781 -0.0226191449432 -0.0194119530244
##              -.-             ...-            .....            ....-
## -0.0169174699638 -0.0124349023936 -0.0098841537903 -0.0074084270706
##             ..-.             .-..             -...             -..-
## -0.0037135912160  0.0001500440045  0.0025882603400  0.0059454966455
##            -....            ...--             -.-.             -.--
##  0.0079898472379  0.0128850353662  0.0160734720446  0.0189618206028
##            --...             --..             --.-             .--.
##  0.0227316781241  0.0245884735682  0.0277769102936  0.0299525494520
##             .---              --.              ---            ..---
##  0.0340037396158  0.0372484428427  0.0397991922038  0.0433439835927
##            ---..            .----            ----.            -----
##  0.0466449533526  0.0510712537118  0.0539033357211  0.0567166622204
```

The one-dimensional plot show quite a bit of movement, with the largest jump at the very first change of $\lambda$ (from 0.000 to 0.001).



We can also plot stress and the penalty term as functions of $\lambda$. Again, note the big change in the penalty term when $\lambda$ goes from zero to 0.001.



After the first 2593 values of $\lambda$ the penalty term is zero and we stop, i.e. we estimate $\lambda_+$ is 2.593. At that point we have run a total of $1.1328 \times 10^4$ FDS iterations, and thus on average about two iterations per $\lambda$ value. Stress has increased from 0.0007634169 to 0.2303106976 and the penalty value has decreased from 0.4815138540 to 0.0000000001.

Our order, and consequently our solution, is the same as the exact global solution given by Palubeckis (2013). See his table 4, reproduced below. The difference is that computing our solution takes 6 seconds, while his takes 494 seconds (different computers, of course). But we would not know that we actually found the global mimimum if his exact method had not analyzed the same data before.

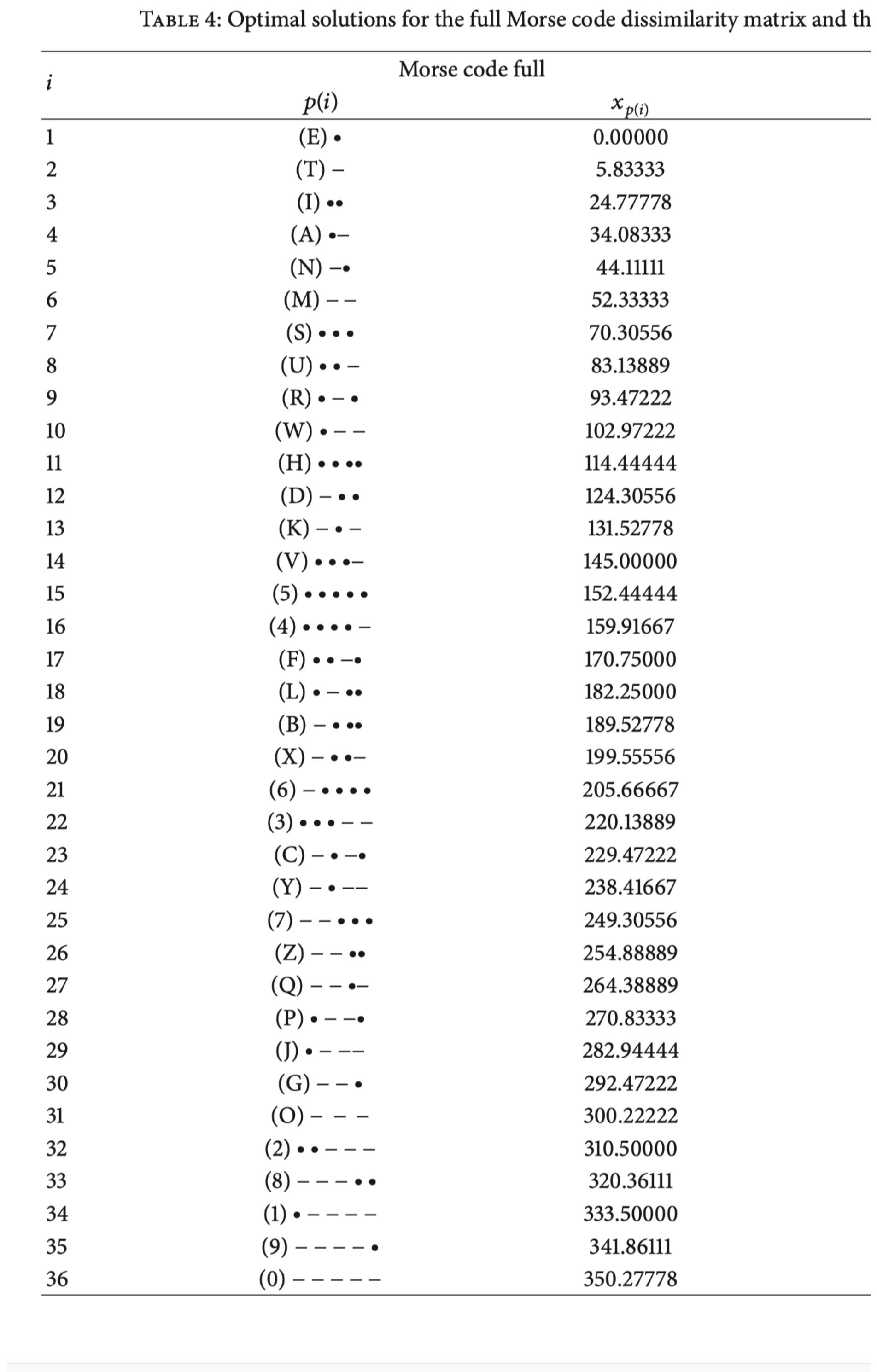


TABLE 4: Optimal solutions for the full Morse code dissimilarity matrix and th

| $i$ | Morse code full | |
|---|---|---|
| | $p(i)$ | $x_{p(i)}$ |
| 1 | (E) • | 0.00000 |
| 2 | (T) – | 5.83333 |
| 3 | (I) •• | 24.77778 |
| 4 | (A) •– | 34.08333 |
| 5 | (N) –• | 44.11111 |
| 6 | (M) – – | 52.33333 |
| 7 | (S) • • • | 70.30556 |
| 8 | (U) • • – | 83.13889 |
| 9 | (R) • – • | 93.47222 |
| 10 | (W) • – – | 102.97222 |
| 11 | (H) • • •• | 114.44444 |
| 12 | (D) – • • | 124.30556 |
| 13 | (K) – • – | 131.52778 |
| 14 | (V) • • •– | 145.00000 |
| 15 | (5) • • • • • | 152.44444 |
| 16 | (4) • • • • – | 159.91667 |
| 17 | (F) • • –• | 170.75000 |
| 18 | (L) • – •• | 182.25000 |
| 19 | (B) – • •• | 189.52778 |
| 20 | (X) – • •– | 199.55556 |
| 21 | (6) – • • • • | 205.66667 |
| 22 | (3) • • • – – | 220.13889 |
| 23 | (C) – • –• | 229.47222 |
| 24 | (Y) – • –– | 238.41667 |
| 25 | (7) – – • • • | 249.30556 |
| 26 | (Z) – – •• | 254.88889 |
| 27 | (Q) – – •– | 264.38889 |
| 28 | (P) • – –• | 270.83333 |
| 29 | (J) • – –– | 282.94444 |
| 30 | (G) – – • | 292.47222 |
| 31 | (O) – – – | 300.22222 |
| 32 | (2) • • – – – | 310.50000 |
| 33 | (8) – – – • • | 320.36111 |
| 34 | (1) • – – – – | 333.50000 |
| 35 | (9) – – – – • | 341.86111 |
| 36 | (0) – – – – – | 350.27778 |

## 7 Discussion

There are two types of examples in which there are many local minima and the task of finding the global minimum is mre difficult. First, the one-dimensional solutions. And second, the solutions where the dissimilarities are in a relatively small interval. This tends to imply a large Gower rank and a bad fit in a small number of dimensions. If the Gower rank is small the number of local minima tends to be small, and the global minimum is easier to find.

Our global MDS method performs rather well. In any serious application we suggest that the user always computes the FDS solution. Then various lambda sequences should be tried to see how small we can make stress, and how the results compare with the more traditional methods (Torgerson initial configuration and/or a large number of random starts). For the Plato data we assume there are too many close local minima, possibly because the dissimilarities do not have enough variation.

In our two dimensional examples we always start our plots with the first two dimensions of the FDS configuration. These two-dimensional configurations are usually small (all points relatively close to the origin), because so much variation is still in the higher dimensions. If $\lambda$ increases the growth of the configurations is one important aspect of configuration change.

In our iteration counts with short sequences of $\lambda$ we see relatively small increases in stress and small decreases in the penalty term, until we get closer to $\lambda_+$, when we suddenly see a sudden change and a larger number of iterations. This is also reflected in the figures, where generally the change to the last solution (with the largest $\lambda$) makes the largest jump. This suggest a finer sequence near $\lambda_+$ and perhaps an adaptive strategy for choosing $\lambda$. Or to use brute force, as in the unidimensional Morse code example. With such longer and finer sequences convergence becomes more smooth.

Another all-important aspect of the method discussed here is that it assumes computation of the global minimum for each $\lambda$. Since we cannot expect a result as nice as the one for FDS (all local minima are global) for $\lambda > 0$ our method remains somewhat heuristic. In the Plato example we have seen that some sequences of $\lambda$ can take us to a non-global local minimum. Of course the fact that we start with a global minimum for $\lambda = 0$ is of some help, but we do not know how far it will take us in general. Jumps near $\lambda_+$ may indicate bifurcations to other local minima.

We have not stressed in the paper that minimizing the penalty function is a *continuation method* (Allgower and George (1979)). This means that probably better methods are available to follow the trajectory of solutions along $\lambda > 0$. There are also possibilities in exploring the fact that the minimum over configurations of the penalty function (12) is a concave function of the single variable $\lambda$.

All or results are for metric scaling, in which we take the numerical values of the dissimilarities seriously. In nonmetric MDS, which is undoubtedly the more popular variant of MDS, the local minimum situation may be more serious. This is easy to see in the one-dimensional case. Each of the $n!$ cones that potentially has a local minimum in its interior is partioned into a number of subcones, defined by the possible blocks in the monotone. Stress is quadratic on each of these subcones, of which there will be many more than $n!$. Also, in the nonmetric case we have to remember that stress is a function of the configuration and of the transformed dissimilarities. Even if we find a global minimum over transformations for a fixed configuration and a global minimum over configurations for a fixed transformation the resulting solution for both can still define a saddle point. regression.

# (APPENDIX) Appendix

# 8 Quadratic External Penalties

Suppose $\mathcal{X} \subseteq \mathbb{R}^n$ and $f : \mathbb{R}^n \Rightarrow \mathbb{R}$ is continuous. Define

$$\mathcal{X}_\star = \underset{x \in \mathcal{X}}{\text{argmin}}\ f(x)$$

Suppose $\mathcal{X}_\star$ is non-empty and that $x_\star$ is any element of $\mathcal{X}_\star$, and

$$f_\star = f(x_\star) = \min_{x \in \mathcal{X}}\ f(x).$$

The following convergence analysis of external linear penalty methods is standard and can be found in many texts (for example, Zangwill (1969), section 12.2).

The penalty term $g : \mathbb{R}^n \Rightarrow \mathbb{R}^+$ is continuous and satisfies $g(x) = 0$ if and only if $x \in \mathcal{X}$. For each $\lambda > 0$ we define the (linear, external) penalty function

$$h(x, \lambda) = f(x) + \lambda g(x). \tag{18}$$

Suppose $\{\lambda_k\}$ is a strictly increasing sequence of positive real numbers. Define

$$\mathcal{X}_k = \underset{x \in \mathcal{X}}{\text{argmin}}\ h(x, \lambda_k). \tag{19}$$

Suppose all $\mathcal{X}_k$ are nonempty and contained in a compact subset of $\mathcal{X}$. Choose $x_k \in \mathcal{X}_k$ arbitrarily.

**Lemma 2: [Basic]**

1: $h(x_k, \lambda_k) \leq h(x_{k+1}, \lambda_{k+1})$.

2: $g(x_k) \geq g(x_{k+1})$.

3: $f(x_k) \leq f(x_{k+1})$.

4: $f_\star \geq h(x_k, \lambda_k) \geq f(x_k)$.

**Proof:**

1: We have the chain

$$h(x_{k+1}, \lambda_{k+1}) = f(x_{k+1}) + \lambda_{k+1} g(x_{k+1}) \geq f(x_{k+1}) + \lambda_k g(x_{k+1}) \geq f(x_k) + \lambda_k g(x_k) = h(x_k, \lambda_k).$$

2: Both

$$f(x_k)+\lambda_k g(x_k)\leq f(x_{k+1})+\lambda_k g(x_{k+1}), \tag{20}$$
$$f(x_{k+1})+\lambda_{k+1} g(x_{k+1})\leq f(x_k)+\lambda_{k+1} g(x_k). \tag{21}$$

Adding inequalities (20) and (21) gives

$$\lambda_k g(x_k)+\lambda_{k+1}g(x_{k+1})\leq \lambda_k g(x_{k+1})+\lambda_{k+1}g(x_k),$$

or

$$(\lambda_k-\lambda_{k+1})g(x_k)\leq(\lambda_k-\lambda_{k+1})g(x_{k+1}),$$

and thus $g(x_k)\geq g(x_{k+1})$.

3: First

$$f(x_{k+1})+\lambda_k g(x_{k+1})\geq f(x_k)+\lambda_k g(x_k). \tag{22}$$

We just proved that $g(x_{k+1})\geq g(x_k)$, and thus

$$f(x_k)+\lambda_k g(x_k)\geq f(x_k)+\lambda_k g(x_{k+1}). \tag{23}$$

Combining inequalities (22) and (23) gives $f(x_{k+1})\geq f(x_k)$.

4: We have the chain

$$f_\star=f(x_\star)+\lambda_k g(x_\star)\geq f(x_k)+\lambda_k g(x_k)\geq f(x_k).$$

■

**Theorem 3:** Suppose the sequence $\{\lambda_k\}_{k\in K}$ diverges to $\infty$ and $x_{\star\star}$ is the limit of any convergent subsequence $\{x_\ell\}_{\ell\in L}$. Then $x_{\star\star}\in\mathcal{X}_\star$, and $f(x_{\star\star})=f_\star$, and $g(x_{\star\star})=0$.

**Proof:** Using part 4 of lemma 2

$$\lim_{\ell\in L}h(x_\ell,\lambda_\ell)=\lim_{\ell\in L}\{f(x_\ell)+\lambda_\ell g(x_\ell)\}=f(x_{\star\star})+\lim_{\ell\in L}\lambda_\ell g(x_\ell)\leq f(x_\star).$$

Thus $\{h(x_\ell,\lambda_\ell)_{\ell\in L}\}$ is a bounded increasing sequence, which consequently converges, and $\lim_{\ell\in L}\lambda_\ell g(x_\ell)$ also converges. Since $\{\lambda_\ell\}_{\ell\in L}\rightarrow\infty$ it follows that $\lim_{\ell\in L}g(x_\ell)=g(x_{\star\star})=0$. Thus $x_{\star\star}\in\mathcal{X}$. Since $f(x_\ell)\leq f_\star$ we see that $f(x_{\star\star})\leq f_\star$, and thus $x_{\star\star}\in\mathcal{X}_\star$ and $f(x_{\star\star})=f_\star$.
■

## 9 Code

### 9.1 penalty.R

```
source("smacofBasics.R")

smacofComplement <- function(y, v) {
  return(sum(v * tcrossprod(y)) / 4)
}

smacofPenalty <-
  function(w,
           delta,
           p = 2,
           lbd = 0,
           zold = columnCenter(diag(nrow(delta))),
           itmax = 10000,
           eps = 1e-10,
           verbose = FALSE) {
    itel <- 1
    n <- nrow(zold)
    vmat <- smacofVmat(w)
    vinv <- solve(vmat + (1 / n)) - (1 / n)
    dold <- as.matrix(dist(zold))
    mold <- sum(w * delta * dold) / sum(w * dold * dold)
    zold <- zold * mold
    dold <- dold * mold
    yold <- zold[, (p + 1):n]
    sold <- smacofLoss(dold, w, delta)
    bold <- smacofBmat(dold, w, delta)
    told <- smacofComplement(yold, vmat)
    uold <- sold + lbd * told
    repeat {
      znew <- smacofGuttman(zold, bold, vinv)
      ynew <- znew[, (p + 1):n] / (1 + lbd)
      znew[, (p + 1):n] <- ynew
      xnew <- znew[, 1:p]
      dnew <- as.matrix(dist(znew))
      bnew <- smacofBmat(dnew, w, delta)
      tnew <- smacofComplement(ynew, vmat)
      snew <- smacofLoss(dnew, w, delta)
      unew <- snew + lbd * tnew
      if (verbose) {
        cat(
          "itel ",
          formatC(itel, width = 4, format = "d"),
          "sold ",
          formatC(
            sold,
            width = 10,
            digits = 6,
            format = "f"
          ),
          "snew ",
          formatC(
            snew,
            width = 10,
            digits = 6,
            format = "f"
          ),
          "told ",
          formatC(
            told,
            width = 10,
            digits = 6,
            format = "f"
          ),
          "tnew ",
          formatC(
            tnew,
            width = 10,
            digits = 6,
            format = "f"
          ),
          "uold ",
          formatC(
            uold,
            width = 10,
            digits = 6,
            format = "f"
          ),
          "unew ",
          formatC(
            unew,
            width = 10,
            digits = 6,
            format = "f"
          ),
          "\n"
        )
      }
      if (((uold - unew) < eps) || (itel == itmax)) {
        break
      }
      itel <- itel + 1
```

```
      zold <- znew
      bold <- bnew
      sold <- snew
      told <- tnew
      uold <- unew
    }
    zpri <- znew %*% svd(znew)$v
    xpri <- zpri[, 1:p]
    return(list(
      x = xpri,
      z = zpri,
      b = bnew,
      l = lbd,
      s = snew,
      t = tnew,
      itel = itel
    ))
  }
```

## 9.2 runPenalty.R

```
runPenalty <-
  function(delta,
           w = array(1, dim(delta)) - diag(nrow(delta)),
           lbd,
           p = 2,
           itmax = 10000,
           eps = 1e-10,
           cut = 1e-8,
           write = TRUE,
           verbose = FALSE) {
    m <- length(lbd)
    hList <- as.list(1:m)
    hList[[1]] <-
      smacofPenalty(
        w,
        delta,
        p,
        lbd = lbd[1],
        itmax = itmax,
        eps = eps,
        verbose = verbose
      )
    for (j in 2:m) {
      hList[[j]] <-
        smacofPenalty(
          w,
          delta,
          p,
          zold = hList[[j - 1]]$z,
          lbd = lbd[j],
          itmax = itmax,
          eps = eps,
          verbose = verbose
        )
    }
    mm <- m
    for (i in 1:m) {
      if (write) {
        cat(
          "itel",
          formatC(hList[[i]]$itel, width = 4, format = "d"),
          "lambda",
          formatC(
            hList[[i]]$l,
            width = 10,
            digits = 6,
            format = "f"
          ),
          "stress",
          formatC(
            hList[[i]]$s,
            width = 8,
            digits = 6,
            format = "f"
          ),
          "penalty",
          formatC(
            hList[[i]]$t,
            width = 8,
            digits = 6,
            format = "f"
          ),
          "\n"
        )
      }
      if (hList[[i]]$t < cut) {
        mm <- i
        break
      }
    }
    return(hList[1:mm])
  }

writeSelected <- function(hList, ind) {
  m <- length(hList)
  n <- length(ind)
  mn <- sort(union(union(1:3, ind), m - (2:0)))
  for (i in mn) {
    if (i > m) {
      next
    }
    cat(
      "itel",
      formatC(hList[[i]]$itel, width = 4, format = "d"),
      "lambda",
      formatC(
        hList[[i]]$l,
        width = 10,
        digits = 6,
        format = "f"
      ),
      "stress",
      formatC(
        hList[[i]]$s,
        width = 8,
        digits = 6,
        format = "f"
      ),
      "penalty",
      formatC(
        hList[[i]]$t,
        width = 8,
        digits = 6,
        format = "f"
      ),
      "\n"
    )
  }
}
```

## 9.3 matchMe.R

```
matchMe <- function (x,
                     itmax = 100,
                     eps = 1e-10,
                     verbose = FALSE) {
  m <- length (x)
  y <- sumList (x) / m
  itel <- 1
  fold <- sum (sapply (x, function (z)
    (z - y) ^ 2))
  repeat {
    for (j in 1:m) {
      u <- crossprod (x[[j]], y)
      s <- svd (u)
      r <- tcrossprod (s$u, s$v)
      x[[j]] <- x[[j]] %*% r
    }
    y <- sumList (x) / m
    fnew <- sum (sapply (x, function (z)
      (z - y) ^ 2))
    if (verbose) {

    }
    if (((fold - fnew) < eps) || (itel == itmax))
      break
    itel <- itel + 1
    fold <- fnew
  }
  return (x)
}

sumList <- function (x) {
  m <- length (x)
  y <- x[[1]]
  for (j in 2:m) {
    y <- y + x[[j]]
  }
  return (y)
}
```

## 9.4 plotMe.R

```r
plotMe2 <- function(hList, labels, s = 1, t = 2) {
  n <- nrow(hList[[1]]$x)
  m <- length (hList)
  par(pty = "s")
  hMatch <- matchMe(lapply (hList, function(r)
    r$x))
  hMat <- matrix(0, 0, 2)
  for (j in 1:m) {
    hMat <- rbind(hMat, hMatch[[j]][, c(s, t)])
  }
  plot(
    hMat,
    xlab = "dim 1",
    ylab = "dim 2",
    col = c(rep("RED", n * (m - 1)), rep("BLUE", n)),
    cex = c(rep(1, n * (m - 1)), rep(2, n))
  )
  for (i in 1:n) {
    hLine <- matrix(0, 0, 2)
    for (j in 1:m) {
      hLine <- rbind(hLine, hMatch[[j]][i, c(s, t)])
    }
    lines(hLine)
  }
  text(hMatch[[m]], labels, cex = .75)
}

plotMe1 <- function(hList, labels) {
  n <- length(hList[[1]]$x)
  m <- length(hList)
  blow <- function(x) {
    n <- length(x)
    return(matrix(c(1:n, x), n, 2))
  }
  hMat <- matrix(0, 0, 2)
  for (j in 1:m) {
    hMat <- rbind(hMat, blow(hList[[j]]$x))
  }
  plot(
    hMat,
    xlab = "index",
    ylab = "x",
    col = c(rep("RED", n * (m - 1)), rep("BLUE", n)),
    cex = c(rep(1, n * (m - 1)), rep(2, n))
  )
  for (i in 1:n) {
    hLine <- matrix(0, 0, 2)
    for (j in 1:m) {
      hLine <- rbind(hLine, blow(hList[[j]]$x)[i,])
      lines(hLine)
    }
  }
  text(blow(hList[[m]]$x), labels, cex = 1.00)
  for (i in 1:n) {
    abline(h = hList[[m]]$x[i])
  }
}
```

## 9.5 uniMDS.R

```
dyn.load("nextperm.so")

uniMDS <-
  function(delta,
           w = array(1, dim(delta)) - diag(nrow(delta)),
           xold = drop(torgerson(delta, 1)),
           itmax = 1000,
           eps = 1e-15,
           verbose = FALSE) {
    delta <- delta / sqrt(sum(w * delta ^ 2) / 2)
    n <- length(xold)
    dold <- as.matrix(dist(xold))
    ssold <- sum(w * (delta - dold) ^ 2) / 2
    itel <- 1
    repeat {
      xnew <- rowSums(w * delta * sign(outer(xold, xold, "-"))) / n
      dnew <- as.matrix(dist(xnew))
      snew <- sum((delta - dnew) ^ 2) / 2
      if (verbose) {
        cat(
          "itel",
          formatC(itel, digits = 4, format = "d"),
          "fold",
          formatC(sold, digits = 15, format = "f"),
          "fnew",
          formatC(snew, digits = 15, format = "f"),
          "\n"
        )
      }
      if ((max(abs(xold - xnew)) < eps) || (itel == itmax)) {
        break
      }
      xold <- xnew
      sold <- snew
      itel <- itel + 1
    }
    gap <- min(diff(sort(xnew)))
    return(list(
      x = xnew,
      s = snew,
      d = dnew,
      itel = itel,
      gap = gap
    ))
  }

uniCheck <- function(delta,
                     w = array(1, dim(delta)) - diag(nrow(delta)),
                     eps = 1e-10) {
  delta <- delta / sqrt(sum(w * delta ^ 2) / 2)
  perm = 1:nrow(delta)
  n <- length(perm)
  m <- factorial(n)
  s <- NULL
  x <- NULL
  for (k in 1:m) {
    d <- sign(outer(perm, perm, "-"))
    t <- rowSums(w * delta * d) / n
    g <- min(diff(sort(t)))
    u <- sign(outer(t, t, "-"))
    r <- sum(u * d)
    if ((r == n * (n - 1)) && (g > eps)) {
      x <- cbind(x, t)
      s <- c(s, sum(w * (delta - abs(outer(t, t, "-"))) ^ 2) / 2)
    }
    perm <-
      .C("nextPermutation", as.integer(perm), as.integer(n))[[1]]
  }
  return(list(x = x, s = s))
```

```
}
```

## 9.6 smacof.R

```
source("smacofBasics.R")


smacof <-
  function(delta,
           w = array(1, dim(delta)) - diag(nrow(delta)),
           p = 2,
           xold = torgerson(delta, p),
           itmax = 10000,
           eps = 1e-10,
           verbose = FALSE) {
    itel <- 1
    n <- nrow(xold)
    vmat <- smacofVmat(w)
    vinv <- solve(vmat + (1 / n)) - (1 / n)
    dold <- as.matrix(dist(xold))
    mold <- sum(w * delta * dold) / sum(w * dold * dold)
    xold <- xold * mold
    dold <- dold * mold
    sold <- smacofLoss(dold, w, delta)
    bold <- smacofBmat(dold, w, delta)
    repeat {
      xnew <- smacofGuttman(xold, bold, vinv)
      dnew <- as.matrix(dist(xnew))
      bnew <- smacofBmat(dnew, w, delta)
      snew <- smacofLoss(dnew, w, delta)
      if (verbose) {
        cat(
          "itel ",
          formatC(itel, width = 4, format = "d"),
          "sold ",
          formatC(
            sold,
            width = 10,
            digits = 6,
            format = "f"
          ),
          "snew ",
          formatC(
            snew,
            width = 10,
            digits = 6,
            format = "f"
          ),
         "\n"
        )
      }
      if (((sold - snew) < eps) || (itel == itmax)) {
        break
      }
      itel <- itel + 1
      xold <- xnew
      bold <- bnew
      sold <- snew
     }
    return(list(
      x = xnew,
      b = bnew,
      s = snew,
      itel = itel
    ))
  }
```

## 9.7 smacofBasics.R

```
smacofLoss <- function(d, w, delta) {
  return(sum(w * (delta - d) ^ 2) / 4)
}

smacofBmat <- function(d, w, delta) {
  dd <- ifelse(d == 0, 0, 1 / d)
  b <- -dd * w * delta
  diag(b) <- -rowSums(b)
  return(b)
}

smacofVmat <- function(w) {
  v <- -w
  diag(v) <- -rowSums(v)
  return(v)
}

smacofGuttman <- function(x, b, vinv) {
  return(vinv %*% b %*% x)
}

doubleCenter <- function(x) {
  rs <- apply(x, 1, mean)
  ss <- mean(x)
  return(x - outer(rs, rs, "+") + ss)
}

columnCenter <- function(x) {
  return(apply(x, 2, function(z)
    z - mean(z)))
}

torgerson <- function(delta, p) {
  e <- eigen(-.5 * doubleCenter(delta ^ 2))
  l <- sqrt(pmax(0, e$values[1:p]))
  if (p == 1) {
    return(as.matrix(e$vectors[, 1] * l))
  } else {
    return(e$vectors[, 1:p] %*% diag(l))
  }
}
```

## 9.8 runSmacof.R

```
source("smacof.R")

runSmacof <-
  function(delta,
           w = array(1, dim(delta)) - diag(nrow(delta)),
           p = 2,
           itmax = 10000,
           eps = 1e-10,
           verbose = FALSE,
           nrand = 1000) {
    str <- NULL
    n <- nrow(delta)
    for (i in 1:nrand) {
      xrnd <- columnCenter(matrix(rnorm(n * p), n, p))
      h <-
        smacof(w = array(1, dim(delta)) - diag(nrow(delta)),
               delta,
               p,
               xold = xrnd,
               itmax = itmax,
               verbose = verbose)
      str <- c(str, h$s)
    }
    return(str)
  }

runUniMDS <- function(
         delta,
         w = array(1, dim(delta)) - diag(nrow(delta)),
         itmax = 10000,
         eps = 1e-10,
         verbose = FALSE,
         nrand = 1000) {
  str <- NULL
  n <- nrow(delta)
  for (i in 1:nrand) {
    xrnd <- rnorm(n)
    h <-
      uniMDS(delta,
             w,
             xold = xrnd,
             itmax = itmax,
             verbose = verbose)
    str <- c(str, h$s)
  }
  return(str)
}
```

## 9.9 janUtil.R

```
mPrint <- function(x,
                   digits = 6,
                   width = 8,
                   format = "f",
                   flag = "+") {
  print(noquote(
    formatC(
      x,
      digits = digits,
      width = width,
      format = format,
      flag = flag
    )
  ))
}

butLast <- function(x, m = 1) {
  return(rev(rev(x)[-(1:m)]))
}

butFirst <- function(x, m = 1) {
  return(x[-(1:m)])
}
```

## References

Allgower, E. L., and K. George. 1979. *Introduction to Numerical Continuation Methods*. Wiley.

Cox, D. R., and L. Brandwood. 1959. "On a Discriminatory Problem Connected with the Works of Plato." *Journal of the Royal Statistical Society, Series B* 21: 195–200.

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. 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.

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

———. 1993. "Fitting Distances by Least Squares." Preprint Series 130. Los Angeles, CA: UCLA Department of Statistics. https://jansweb.netlify.app/publication/deleeuw-r-93-c/deleeuw-r-93-c.pdf.

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

———. 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.

———. 2016. "Gower Rank." 2016. https://jansweb.netlify.app/publication/deleeuw-e-16-k/deleeuw-e-16-k.pdf.

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

———. 2023a. "Least Squares Unidimensional Scaling." 2023. https://doi.org/10.13140/RG.2.2.27920.17927.

———. 2023b. "Powering Dissimilarities in Metric MDS." 2023. https://doi.org/10.13140/RG.2.2.31865.44645.

De Leeuw, J., P. Groenen, and P. Mair. 2016. "Full-Dimensional Scaling." 2016. https://jansweb.netlify.app/publication/deleeuw-groenen-mair-e-16-e/deleeuw-groenen-mair-e-16-e.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.

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

Guilford, J. P. 1954. *Psychometric Methods.* McGraw-Hill.

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.

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.

Palubeckis, G. 2013. "An Improved Exact Algorithm for Least-Squares Unidimensional Scaling." *Journal of Applied Mathematics*, 1–15.

Revelle, W. 2023. *psychTools:Tools to Accompany the 'psych' Package for Psychological Research* . R package version 2.3.4. https://CRAN.R-project.org/package=psychTools.

Rockafellar, R. T. 1970. *Convex Analysis.* Princeton University Press.

Rothkopf, E. Z. 1957. "A Measure of Stimulus Similarity and Errors in some Paired-associate Learning." *Journal of Experimental Psychology* 53: 94–101.

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.

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