Multidimensional Scaling in R: SMACOF

Patrick Mair
Harvard University

Jan de Leeuw
University of California, Los Angeles

Patrick J. F. Groenen
Erasmus University Rotterdam

Abstract

This article is an updated version of De Leeuw and Mair (2009b) published in the Journal of Statistical Software. It elaborates on the methodology of multidimensional scaling problems (MDS) solved by means of the majorization algorithm. The objective function to be minimized is known as stress and functions which majorize stress are elaborated. This strategy to solve MDS problems is called SMACOF and it is implemented in an R package of the same name which is presented in this article. We extend the basic SMACOF theory in terms of configuration constraints, three-way data, unfolding models, and projection of the resulting configurations onto spheres and other quadratic surfaces. Various examples are presented to show the possibilities of the SMACOF approach offered by the corresponding package.

Keywords: SMACOF, multidimensional scaling, majorization, R.

1. Introduction

Multidimensional scaling (MDS) is a family of scaling methods for discovering structures in multidimensional data. Based on an proximity matrix, typically derived from variables measured on objects as input entity, these dissimilarities are mapped on a low-dimensional spatial representation. A classical example concerns airline distances between cities in miles as symmetric input matrix. Applying MDS, it results in a two-dimensional graphical representation reflecting the map (see Kruskal and Wish 1978). Depending on the nature of the original data various proximity/dissimilarity measures can be taken into account. For an overview see Cox and Cox (2001, Chapter 1) and an implementation of numerous proximity measures in R (R Core Team 2015) is given by Meyer and Buchta (2007). Typical application areas for MDS are, among others, social and behavioral sciences, marketing, biometrics, and ecology.

For introductory MDS reading we refer to Kruskal and Wish (1978) and more advanced topics can be found in Borg and Groenen (2005) and Cox and Cox (2001).

The smacof package provides a broad variety of MDS implementations. The basic implementation is symmetric SMACOF, i.e. MDS on symmetric input dissimilarity matrices with options for ratio, interval, ordinal, and spline transformations of the proximities. Extensions in terms of confirmatory MDS (internal, external restrictions) are provided as well as individual difference scaling (INDSCAL, IDIOSCAL and friends). In addition the package
SMACOF in R

implements metric unfolding, i.e. SMACOF on rectangular dissimilarity matrices. Some special techniques such as Procrustes, inverse MDS, and unidimensional scaling are available as well.

2. Basic MDS strategies using SMACOF

MDS input data are typically a \( n \times n \) matrix \( \Delta \) of dissimilarities based on observed data. \( \Delta \) is symmetric, non-negative, and hollow (i.e. has zero diagonal). The problem we solve is to locate \( i, j = 1, \ldots, n \) points in low-dimensional Euclidean space in such a way that the distances between the points approximate the given dissimilarities \( \delta_{ij} \). Thus we want to find an \( n \times p \) matrix \( X \) such that \( d_{ij}(X) \approx \delta_{ij} \), where

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

(1)

The index \( s = 1, \ldots, p \) denotes the number of dimensions in the Euclidean space. The elements of \( X \) are called configurations of the objects. Thus, each object is scaled in a \( p \)-dimensional space such that the distances between the points in the space match as well as possible the observed dissimilarities. By representing the results graphically, the configurations represent the coordinates in the configuration plot.

SMACOF stands for Stress Majorization of a COrplicated Function. For MDS, majorization was introduced by De Leeuw (1977a) and further elaborated in De Leeuw and Heiser (1977) and De Leeuw and Heiser (1980). Kruskal’s stress \( \sigma(X) \) is defined by

\[
\sigma(X) = \sum_{i<j} w_{ij}(\delta_{ij} - d_{ij}(X))^2.
\]

(2)

Here, \( W \) is a known \( n \times n \) matrix of weights \( w_{ij} \), also assumed to be symmetric, non-negative, and hollow. We assume, without loss of generality, that

\[
\sum_{i<j} w_{ij}\delta_{ij}^2 = n(n-1)/2
\]

(3)

and that \( W \) is irreducible (De Leeuw 1977a), so that the minimization problem does not separate into a number of independent smaller problems. \( W \) can for instance be used for imposing missing value structures: \( w_{ij} = 1 \) if \( \delta_{ij} \) is known and \( w_{ij} = 0 \) if \( \delta_{ij} \) is missing. However, other kinds of weighting structures are allowed along with the restriction \( w_{ij} \geq 0 \).

Let us start with a very simple example where the input dissimilarities are actually (Euclidean) distances. The data we use are from Michael Friendly’s website (http://www.datavis.ca/gallery/guerry/) and represent distances between French department centroids in 1830. We compute a two-dimensional ratio MDS (input dissimilarities not transformed) which reproduces the map and the resulting configurations correspond to map coordinates. In the left panel of Figure 1 we see that it is not quite the geographical map of France since it is rotated. Note that MDS is blind to geographic directions since it solely operates on dissimilarities/distances. In practice, this does not matter unless we work with geographical data.
Figure 1: Left panel: original MDS solution. Right panel: rotated MDS solution with French map as background.

```r
data(Guerry)
fit.guerry <- mds(Guerry)

op <- par(mfrow = c(1,2))
plot(fit.guerry)
theta <- 82*pi/180  # degrees to radians
rot <- matrix(c(cos(theta), sin(theta), -sin(theta), cos(theta)), ncol = 2)
configs82 <- fit.guerry$conf %*% rot  # rotated configurations
francemap1 <- GetMap(destfile="mypic1.png", zoom = 6, center = c(46.55, 3.05),
                      maptype = "satellite")
PlotOnStaticMap(francemap1)
text(configs82*280, labels = rownames(configs82), col = "white", cex = 0.7)
par(op)
```

The right panel in Figure 1 shows a rotated solution (82 degrees) and the coordinates are multiplied by a dilation factor of 280. Both values are determined by trial-and-error. More sophisticated rotation/translation/dilation configuration transformations will be explained in the section on Procrustes analysis.

In practice, researchers do not have Euclidean distances as data. Having a regular subject x variables data frame, one can compute proximities such as correlations or various kind of distance measures. Having similarities such as correlations, the `sim2diss` helper function offers possibilities for corresponding transformations. Note that the `smacof` package always requires dissimilarities as input.
The next dataset is based on ratings of 576 records by 14 judges in a German Heavy Metal magazine called Rock Hard, collected in 2013. For the moment we are only interested in the judges who rated each record on a scale from 0 (super bad) to 10 (incredibly awesome). Half point ratings were allowed. We use an MDS to explore which judges have a similar taste. We use the Euclidean distance between the judges to compute the input dissimilarities. Note that some input dissimilarities are missing (more on that in Section 2.2). We compute a three-dimensional ratio MDS solution.

```r
ratings <- RockHard[,5:18]
rockdiss <- dist(t(ratings))
fit.rock <- mds(rockdiss, ndim = 3)
```

```r
##
## Call:
## mds(delta = rockdiss, ndim = 3)
##
## Model: Symmetric SMACOF
## Number of objects: 14
## Stress-1 value: 0.177
## Number of iterations: 109
```

Figure 2 shows two configuration plots for the corresponding dimensions. Naturally, one could also produce a three-dimensional configuration plot using either the `scatterplot3d` or the `rgl` package.

### 2.1. Transforming the dissimilarities

An important MDS issue in practice concerns the scale level we want to assign to the dissimilarities. This leads to the basic classical distinction between metric and nonmetric MDS. So far we did not consider transformations on the dissimilarities $\delta_{ij}$. MDS was used to represent the data such that their ratios would correspond to the ratios of the distances in the MDS space. This is called ratio MDS and is a special case of metric MDS.

The most popular approach, especially in the Social Sciences, are transformations that preserve the rank order of the dissimilarities, i.e. we assume that the dissimilarities are on an ordinal scale level. If such a transformation $f$ obeys only the monotonicity constraint $\delta_{ij} < \delta_{i'j'} \Rightarrow f(\delta_{ij}) < f(\delta_{i'j'})$, within an MDS context it is referred to as ordinal MDS or nonmetric MDS (Kruskal 1964b). The resulting $\hat{d}_{ij} = f(\delta_{ij})$ are commonly denoted as disparities which have to be chosen in an optimal way. Straightforwardly, the stress function (for the symmetric case) becomes

$$
\sigma(X) = \sum_{i<j} w_{ij}(\hat{d}_{ij} - d_{ij}(X))^2
$$

(4)
which we have to minimize with respect to the configurations $X$ and, simultaneously, with respect to the disparity matrix $\hat{D}$. Regarding majorization, there is one additional step after each Guttman transform in iteration $t$: The computation of optimal $\hat{d}_{ij}(t)$ (with subsequent normalization) such that the monotonicity constraint is fulfilled. If the order of $d_{ij}(X(t))$ is the same as the order of $\hat{d}_{ij}(t)$, the optimal update is clearly $\hat{d}_{ij}(t) = d_{ij}(X(t))$. If the orders differ, the optimal update is found by monotone regression.

Within this context we have to consider the case of ties in the observed ordinal dissimilarity matrix $\Delta$, i.e., the case of $\delta_{ij} = \delta_{i'j'}$. Having this case, we distinguish between three approaches: the primary approach (“break ties”) does not necessarily require that $\hat{d}_{ij} = \hat{d}_{i'j'}$, whereas the (more restrictive) secondary approach does (“keep ties”). An even less restrictive version is the tertiary approach from De Leeuw (1977b), in which we require that the means of the tie-blocks are in the correct order. More details can be found in Cox and Cox (2001).

When solving the monotone (or isotonic) regression problem in step $t$, one particular tie approach has to be taken into account. In MDS literature this problem is referred to as primary monotone least squares regression and smacof solves it by means of the pooled-adjacent-violators algorithm (PAVA, Ayer, Brunk, Ewing, Reid, and Silverman 1955; Barlow, Bartholomew, Bremner, and Brunk 1972).

Another, even simpler type of transformation we can consider is a linear transformation where $\hat{d}_{ij} = f(\delta_{ij}) = a + b\delta_{ij}$. This strategy is called interval MDS and plays an important role in modern metric MDS applications. In interval MDS, then, the ratio of differences of distances should be equal to the corresponding ratio of differences in the data. Having a linear transformation, we can naturally think of nonlinear transformations as well. This could be a simple polynomial transformation, logarithmic and exponential transformations, or, even more sophisticated, (monotone) spline transformations. Splines are piecewise polynomial

---

**Figure 2:** Left panel: Configuration plot dimension 1 vs. 2. Right panel: Configuration plot dimension 1 vs. 3.
functions; the “pieces” are determined by knots and the spline degree. Within an MDS context, mostly monotone splines are relevant which lead to a smoother transformation of the $\delta_{ij}$’s compared to monotone regression. In the smacof we use $I$-splines (integrated splines), a special type of monotone splines. More technical MDS spline details can be found in Borg and Groenen (2005, Chapter 9.6).

Transformations can be nicely shown in a Shepard diagram (see De Leeuw and Mair 2015, for a general description). A Shepard diagram consists of a scatterplot between the dissimilarities $\delta_{ij}$ and the configuration distances $d_{ij}(X)$. In addition, it shows the disparities $\hat{d}_{ij}$ and from this we see nicely how the $\delta_{ij}$ are transformed. Some examples of various transformations using the kinship dissimilarity data (Rosenberg and Kim 1975) based in 15 kinskip terms are given in Figure 3.

```r
fit.interval <- mds(kinshipdelta, type = "interval")
fit.ordinal1 <- mds(kinshipdelta, type = "ordinal", ties = "primary")
fit.ordinal2 <- mds(kinshipdelta, type = "ordinal", ties = "secondary")
fit.spline <- mds(kinshipdelta, type = "mspline", spline.intKnots = 3,
                 spline.degree = 2)
op <- par(mfrow = c(2,2))
plot(fit.interval, plot.type = "Shepard",
     main = "Shepard Diagram (Interval MDS)", ylim = c(0.1, 1.7))
plot(fit.ordinal1, plot.type = "Shepard",
     main = "Shepard Diagram (Ordinal MDS, Primary)", ylim = c(0.1, 1.7))
plot(fit.ordinal2, plot.type = "Shepard",
     main = "Shepard Diagram (Ordinal MDS, Secondary)", ylim = c(0.1, 1.7))
plot(fit.spline, plot.type = "Shepard",
     main = "Shepard Diagram (Spline MDS)", ylim = c(0.1, 1.7))
par(op)
```

2.2. Missing Values

Sometimes we have the case where some input dissimilarities are missing. As usual in R, they should be coded as NA. In order to perform an MDS computation, a proper specification of the weight matrix $W$ does the trick: $w_{ij} = 0$ if $\delta_{ij}$ is missing; $w_{ij} = 1$ (or any other known weight), if $\delta_{ij}$ non-missing. The smacof package does this $W$ specification automatically if missing dissimilarities are found, the user does not have to worry about. We have already seen a corresponding example using the RockHard data.

2.3. Starting configurations

MDS often ends up in a local minimum, especially for low-dimensional solutions. By default, smacof performs classical scaling to determine (hopefully good) starting configurations. A common strategy to check whether the algorithm ended up in a global minimum is to try several random starting configurations and pick the one with the lowest stress value (of course, it can not be guaranteed that this solution is actually a global minimum). For an example we use the Lawler dataset (Lawler 1967) examining the performance of managers. There are three criteria or traits ($T1 =$ quality of output, $T2 =$ ability to generate output, $T3 =$
Figure 3: Shepard diagrams for various dissimilarity transformations. Top left: Linear transformation. Top right: Monotone regression, break ties. Bottom left: Monotone regression, keep ties. Bottom right: Spline transformation (3 interior knots, cubic splines.)
demonstrated effort to perform) and three methods (M1 = rating by superior, M2 = peer rating, M3 = self-rating). We look at the stress values of the default classical scaling starting solution and 20 random starts.

```r
LawlerD <- sim2diss(Lawler)
fitclas <- mds(LawlerD)
fitclas$stress
## [1] 0.2414665

stressvec <- NULL
set.seed(123)
for(i in 1:20) {
  fitran <- mds(LawlerD, init = "random")
  stressvec[i] <- fitran$stress
}
stressvec
## stress values
## [1] 0.2446885 0.2668114 0.2446878 0.2675292 0.2620006 0.2623503 0.2401496
## [8] 0.2423066 0.2609520 0.2431139 0.2443404 0.2560029 0.2588567 0.2467979
## [15] 0.2436688 0.2467940 0.2554348 0.2515141 0.2529192 0.2526784
```

We see that the 7th solution is the best one of the random starts; and it is even better than the one based on classical scaling starting values.

### 2.4. MDS goodness-of-fit

MDS goodness-of-fit should be judged by the normalized stress value, the stress-per-point, the Shepard diagram, and some tests. The core output value in MDS the stress value. The raw stress value itself is not very informative. A large value does not necessarily indicate bad fit. Several normalizations have been proposed in the literature in order to make the stress value not dependent on the measurement units used in $\Delta$ and $X$, respectively. A popular normalization is Kruskal’s stress-1 which is given by

$$
\sigma(X) = \sqrt{\frac{\sum_{i<j} w_{ij} (\hat{d}_{ij} - d_{ij}(X))^2}{n(n-1)/2}}.
$$

The scaling factor in the denominator comes from the normalization given in (3). We see that the stress is based on an additive decomposition: each object (point) “contributes” to the stress. The higher a point’s contribution, the more “responsible” this point is with respect to lack of fit. These individual stress contributions are called “stress-per-point” and smacof returns the corresponding contributions as percentages. Thus, this concept is somewhat similar to influential points in regression.

The lower bound of the stress value is 0 (perfect fit), the upper bound is nontrivial (see De Leeuw and Stoop 1984). What is a “good” stress value then? Kruskal (1964a) gave some stress-1 benchmarks for ordinal MDS: 0.20 = poor, 0.10 = fair, 0.05 = good, 0.025 = excellent, and 0.00 = perfect. As always, such general rules of thumb are problematic since there are
many aspects that need to be considered when judging stress (see Borg, Groenen, and Mair 2012, for details). Early approaches (e.g. Spence and Ogilvie 1973) suggest to use the average stress value based on random dissimilarity MDS fits as the upper benchmark. It turned out, however, that for most applications it is not too difficult to achieve a stress value that is considerably lower than this benchmark, since there is always some sort of structure in the data. Nonetheless, the smacof package provides the following utility function to compute random stress values dependent on the number of objects $n$, the number of dimensions $p$, and the type of MDS. Let us look at the average ratio MDS stress value for $n = 9$ and $p = 2$ (500 replications), as we had in the Lawler example above:

```r
stressvec <- randomstress(n = 9, ndim = 2, nrep = 500)
mean(stressvec)
## [1] 0.3118535
fit <- mds(LawlerD)
fit$stress
## [1] 0.2414665
```

Not surprisingly, the stress value of 0.24 in the Lawler example is clearly smaller than the average random stress. However, it can be doubted that this is really a good solution.

More modern approaches focus on permutations of the dissimilarity matrix rather than on random dissimilarities. A simple implementation is given by means of the permtest() function. Let us perform a permutation test for the Lawler example (1000 permutations):

```r
set.seed(1234)
res.perm <- permtest(fit, nrep = 1000, verbose = FALSE)
res.perm
##
## Call: permtest.smacof(object = fit, nrep = 1000, verbose = FALSE)
##
## SMACOF Permutation Test
## Number of objects: 9
## Number of replications (permutations): 1000
##
## Observed stress value: 0.241
## p-value: 0.303
```

It results that our MDS fit is not significantly better than the null solutions based on the permutations ($p = 0.303$). We see that permutation tests provide a more useful null distribution than the random dissimilarity approach. Figure 4 shows the results.
Figure 4: Left panel: Histogram of permutation stress values (gray lines show the 5% rejection region, red line the observed stress value). Right panel: Empirical cumulative distribution function of the permuted stress values (dotted horizontal line denotes the .05 significance threshold).

Finally, for the same example, let us have a look at the stress-per-point contributions.

These values represent stress contributions in percentage and we see that T2M1 (trait: ability to generate output; method: rating by superior) is responsible for approximately 14% of the stress. A stress decomposition chart plots these values in descending order and the bubble
plot combines the configuration plot with stress contributions (the larger the bubble, the smaller the contribution, and, consequently, the better the fit; see Figure 5).

```r
op <- par(mfrow = c(1,2))
plot(fit, plot.type = "stressplot")
plot(fit, plot.type = "bubbleplot")
par(op)
```

### 3. Confirmatory MDS I: circular restrictions

The fact that quadratic surfaces frequently show up empirically leads to some interesting technical and methodological problems. In some cases it may be appropriate to require that the points computed by MDS are indeed located exactly on some parametric surface.

Here we are interested in the case in which the points in the configuration are constrained to lie on a quadratic surface (Cox and Cox 1991) in $\mathbb{R}^p$. In $\mathbb{R}^2$, important special cases are a circle, ellipse, hyperbola, and parabola; in $\mathbb{R}^3$, corresponding special cases are a sphere and an ellipsoid.

We call the technique of placing the points on the MDS with quadratic constraints MDS-Q. Borg and Groenen (2005) call this type of MDS *weakly constrained MDS* since the external quadratic restrictions are not necessarily forced. In the most general form of MDS-Q the vector of configurations $x_i$, each of length $p$, must satisfy

$$x_i'\Lambda x_i + 2x_i'\beta + \gamma = 0,$$

(6)
for some $p \times p$ matrix $\Lambda$, some $p$-element vector $\beta$, and some constant $\gamma$. Because of the invariance of the distance function under translations we can put the center of the surface in the origin. And because distance is invariant under rotation, we can also require, without loss of generality, that $\Lambda$ is diagonal. This covers conics (ellipse, hyperbola, parabola) in $\mathbb{R}^2$, and the various kinds of ellipsoids, hyperboloids, paraboloids, and cylinders in $\mathbb{R}^3$. In the case of ellipsoids and hyperboloids we can choose $\beta = 0$ and $\gamma = -1$, such that the constraints become $x_i' \Lambda x_i = 1$. For ellipsoids, the matrix $\Lambda$ is positive semi-definite which means that we can also write

$$x_i = \Lambda^{1/2} z_i, \quad \|z_i\| = 1 \text{ for all } i.$$  

(7)

And, of course, spheres are ellipses in which the matrix $\Lambda$ is scalar, i.e. $\Lambda = \lambda I$.

There are several strategies for fitting MDS-Q. The package smacof allows for the following approaches: Primal methods, in which the constraints are incorporated in parametric form directly into the loss function, and dual methods, where constraints are imposed at convergence by using penalty or Lagrangian terms. The dual method is known as CMDA (Constrained/Confirmatory Monotone Distance Analysis). It was proposed by Borg and Lingoes (1979, 1980) and discussed in Borg and Groenen (2005, Section 10.4). The idea is to impose the restrictions directly on the distances and not on the configurations. This makes the method more specific to MDS.

As an example, we use Ekman’s color data (Ekman 1954). The dataset, as provided in the package represents similarities for 14 colors. These need to be converted into dissimilarities (by simply subtracting from 1). Fitting a basic ordinal MDS on the data we see that the colors are almost aligned in a circle (left panel of Figure 6). Using spherical SMACOF, we can actually restrict these configurations to be on a circle.

```r
ekmanD <- sim2diss(ekman, method = 1)
fit.basic <- mds(ekmanD, type = "ordinal")
fit.circ <- smacofSphere(ekmanD, type = "ordinal", verbose = FALSE)
```

By looking at the stress values we see that the restricted solution (stress-1: 0.0322) is not much worse than the unrestricted solution (stress-1: 0.0233).

### 4. Confirmatory MDS II: external restrictions

#### 4.1. Linear restrictions

De Leeuw and Heiser (1980) introduced a SMACOF version with restrictions on the configuration matrix $X$ which Borg and Groenen (2005, Chapter 10) call confirmatory MDS with external constraints (see also Heiser and Meulman 1983). The basic idea behind this approach is that the researcher has some substantive underlying theory regarding a decomposition of the dissimilarities. We start with the simplest restriction in terms of a linear combination, show the majorization solution and then present some additional possibilities for constraints. The linear restriction in its basic form is

$$X = ZC$$  

(8)
where $Z$ is a known predictor matrix of dimension $n \times q$ ($q \geq p$). The predictors can be numeric in terms of external covariates or one can specify an ANOVA-like design matrix. $C$ is a $q \times p$ matrix of regression weights to be estimated.

### 4.2. Additional restrictions

Basically, the smacof package allows the user to implement arbitrary configuration restrictions by specifying a corresponding update function for $X$. Nevertheless, we provide additional restriction possibilities which are commonly used. Besides the classical linear restriction described above, for the special case of number of predictors equal number of dimensions, i.e. $q = p$, the square matrix $C$ can be restricted to be diagonal: $C = \text{diag}(c_{11}, c_{22}, \ldots, c_{ss}, \ldots, c_{qq})$.

Combining unrestricted, linearly restricted and the diagonally restricted models leads to a framework of a partially restricted $X$. De Leeuw and Heiser (1980) use the block notation

$$X = \begin{bmatrix} X_1 & ZC_1 & C_2 \end{bmatrix}$$

in which $X_1$ is the unrestricted part and of dimension $n \times q_1$. $ZC_1$ is the linearly restricted part of dimension $n \times q_2$ and $C_2$ is a diagonal matrix of order $n$ which can be either present or absent. The corresponding models are commonly coded as triples $(q_1, q_2, q_3)$ denoting the number of dimensions contributed by each component: $q_1$ is the number of unrestricted dimensions, $q_2$ the number of linearly restricted dimensions, and $q_3$ is either zero or one, depending on presence or absence of the diagonal matrix $C_2$. An important special case and the one which is also implemented in smacof is $(q, 0, 1)$ which is a $q$-dimensional MDS model with uniquenesses (Bentler and Weeks 1978).
Further specifications of this partially restricted framework can be found in De Leeuw and Heiser (1980) and additional elaborations in Borg and Groenen (2005, Chapter 10).

4.3. Optimal scaling on external constraints

Meulman (1992) incorporates the MDS approach into Gifi’s optimal scaling model family (Gifi 1990; De Leeuw and Mair 2009a). In each MDS majorization iteration there is one more optimal scaling step in the external constraints. Consequently, Equation (8) changes to

\[ X = \hat{Z}C. \]  

(10)

Each predictor variable \( z_1, \ldots, z_q \) is subject to an optimal scaling transformation. The classical case is to scale in an ordinal way (i.e. monotone regression) but we can think of additional transformations which we have already applied to the dissimilarities such as interval and monotone splines. All of these transformations (plus a spline without monotonicity constraints) are implemented in \texttt{smacof}. They are not only applicable to the linearly constrained configurations but also to all kinds of configuration restrictions presented above. Especially if we think of the general De Leeuw-Heiser framework as given in (9) and which changes to

\[ X = \left[ X_1 \right. \hat{Z}C \left. \right| C_2 \right], \]  

(11)

the possibilities for specifying constrained MDS models is MDS.

Let’s look at an example using the classical morse code data (Rothkopf 1957). Let us fit and unconstrained solution (i.e. a regular, ordinal MDS) and an ordinal solution with external constraints, subject to optimal scaling (see Borg and Groenen 2005, p. 234). The external data refer to the signal type (all short beeps, more short than long beeps, same short and long beeps, more long than short beeps, all long beeps) and the signal length (9 categories). In this case the analysis leads to an MDS with regional constraints.

```r
res.unc <- smacofSym(morse, type = "ordinal")
res.parreg <- smacofConstraint(morse, type = "ordinal", ties = "primary",
                               constraint = "linear",
                               external = morsescales[,2:3],
                               constraint.type = "ordinal",
                               init = res.unc$conf)
```

For the unconstrained solution we get a stress-1 value of 0.181. For the theory-consistent solution the stress is 0.246 which is only slightly worse than the first fit. The corresponding configuration plots are given in Figure 7.

```r
op <- par(mfrow = c(1,2))
plot(res.unc, main = "Unconditional MDS")
plot(res.parreg, main = "Regional MDS")
par(op)
```
Figure 7: Left panel: ordinal MDS (unrestricted). Right panel: ordinal MDS with external configuration restrictions, optimally scaled.
5. SMACOF for individual differences

Individual difference scaling models are an extension of the standard MDS setting in terms of $k = 1, \ldots, K$ separate $n \times n$ symmetric dissimilarity matrices $\Delta_k$. A typical situation is, e.g., that we have $K$ judges and each of them produces a dissimilarity matrix or that we have $K$ replications on some MDS data. The very classical approach for MDS computation on such structures is INDSCAL (INdividual Differences SCALing; Carroll and Chang 1970). An elaborated overview of additional algorithms is given in Borg and Groenen (2005, Chapter 22).

We will focus on the majorization solution and collect the $\Delta_k$ matrices in a block-diagonal structure

$$\Delta^* = \begin{bmatrix}
\Delta_1 \\
\Delta_2 \\
\vdots \\
\Delta_K
\end{bmatrix}.$$ 

The corresponding observed distances are denoted by $\delta_{ij,k}$. Similarly, we merge the resulting configurations $X_k$ into the configuration supermatrix

$$X^* = \begin{bmatrix}
X_1 \\
X_2 \\
\vdots \\
X_K
\end{bmatrix}.$$ 

Allowing for weight matrices $W_k$ with elements $w_{ij,k}$, the total stress to be minimized, consisting of the single $\sigma(X_k)$’s, can be written as

$$\sigma(X^*) = \sum_{k=1}^{K} \sum_{i<j} w_{ij,k} (\delta_{ij,k} - d_{ij}(X_k))^2. \quad (12)$$

In individual difference models there is an additional issue regarding the distance computations. We compute a configuration matrix $X_k$ for each individual, but we constrain the $X_k$ by only allowing differential weighting of each dimension by each individual. If we think of a linear decomposition of $X_k$, as described in the former section, we have

$$X_k = ZC_k \quad (13)$$

with the $C_k$ diagonal matrices of order $p$. The weighted Euclidean distance can be expressed as

$$d_{ij}(ZC_k) = \sqrt{\sum_{s=1}^{p} (c_{ss,k}z_{is} - c_{ss,k}z_{js})^2} = \sqrt{\sum_{s=1}^{p} c_{ss,k}^2 (z_{is} - z_{js})^2}. \quad (14)$$

$Z$ is the $n \times p$ matrix of coordinates of the so called group stimulus space or common space. If $C_k = I$ for all $k$ we get the so called identity model.

In brief, we present three extensions of the classical INDSCAL approach above. Carroll and Chang (1970) extend differential weighting by means of the generalized Euclidean distance,
allowing the $C_k$ in (13) to be general, and not necessarily diagonal. This means
\[ d_{ij}(X_k) = \sqrt{\sum_{s=1}^{p} \sum_{s'=1}^{p} (x_{is} - x_{js'}) h_{ss',k}(x_{is'} - x_{js'})}, \] (15)
with $H_k = C_k C_k'$. This is known as the IDIOSCAL (Individual Differences in Orientation SCALing) model. For identification purposes $H_k$ can be decomposed in various ways. The spectral decomposition (Carroll-Chang decomposition) leads to
\[ H_k = U_k \Lambda_k U_k', \] (16)
where $U_k U_k' = I$ and $\Lambda_k = \text{diag}(\lambda_{ij})$. The Tucker-Harshman decomposition implies
\[ H_k = D_k R_k D_k, \] (17)
where $D_k$ is a diagonal matrix of standard deviations and $R_k$ a correlation matrix. This is often combined with the normalization
\[ \frac{1}{K} \sum_{k=1}^{K} H_k = I \] (18)
proposed by Schönemann (1972). The models currently implemented in smacof are IDIOSCAL, INDSCAL with $C_k$ restricted to be diagonal, and the identity model with $C_k = I$. Additional models can be found in Cox and Cox (2001, Chapter 10).

6. Unfolding Models

Unfolding models are somewhat different from the models presented so far. The input matrix is not a symmetric matrix anymore. The prototypical case for such an input matrix is that we have $n_1$ individuals or judges which rate $n_2$ objects or stimuli. Therefore, MDS becomes a model for preferential choice which is commonly referred to as an unfolding model. The basic idea is that the ratings and the judges are represented on the same scale and for each judge, the corresponding line can be folded together at the judge’s point and his original rankings are observed (Cox and Cox 2001, p.165). This principle of scaling is sometimes denoted as Coombs scaling (Coombs 1950). Detailed explanations on various unfolding techniques can be found in Borg and Groenen (2005, Chapters 14-16). We will limit our explanations to the SMACOF version of metric unfolding.

Let us assume an observed dissimilarity (preference) matrix $\Delta$ of dimension $n_1 \times n_2$ with elements $\delta_{ij}$. For rectangular SMACOF the resulting configuration matrix $X$ is partitioned into two matrices: $X_1$ of dimension $n_1 \times p$ as the individual’s or judge’s configuration, and $X_2$ of dimension $n_2 \times p$ as the object’s configuration matrix. Consequently, stress can be represented as
\[ \sigma(X_1, X_2) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} w_{ij} (\delta_{ij} - d_{ij}(X_1, X_2))^2 \] (19)
with
\[ d_{ij}(X_1, X_2) = \sqrt{\sum_{s=1}^{p} (x_{1is} - x_{2js})^2}. \] (20)
So far, only the metric version of unfolding is implemented in \texttt{smacof}. Nonmetric (ordinal) unfolding is slightly more complicated. The original techniques proposed by Coombs (1964) were purely nonmetric and did not even lead to metric representations. The ranking information is row-conditional, which means we cannot compare the ranks given by individual $i$ to the ranks given by individual $k$. The order is defined only within rows. Metric data are generally unconditional, because we can compare numbers both within and between rows. Because of the paucity of information (only rank order, only row-conditional, only off-diagonal) the usual Kruskal approach to nonmetric unfolding often leads to degenerate solutions, even after clever renormalization and partitioning of the loss function. In nonmetric unfolding the \textit{Stress} becomes

$$\sigma(X_1, X_2) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} w_{ij}(\hat{d}_{ij} - d_{ij}(X_1, X_2))^2$$

with $\hat{d}_{ij} = f(\delta_{ij})$ reflecting a monotone regression on the dissimilarities. Degenerate solutions are characterized by constant disparities. Busing, Groenen, and Heiser (2005) identify constant d-hats using the coefficient of variation and, subsequently, penalize nonmetric transformations of the dissimilarities with small variation. They present a majorization approach for minimizing the adjusted loss function which is a topic of future implementation.

As an example, let us use the RockHard data once more. This time we will not collapse the bands by means of distance computations between the judges (as in the introductory MDS section). Rather, we try to compute and plot configurations for the rows (bands/albums) and columns (judges) in the same space. Note that the input data need to represent dissimilarities: that is, if a judge rated an album very high, the dissimilarity value should be low and vice versa. Therefore, we reverse the coding of the input data.

```r
ratings <- 11-RockHard[,5:18]  # reverse ratings
rownames(ratings) <- RockHard[,"Band"]
fit.rock <- unfolding(ratings)  # 2D metric unfolding solution
fit.rock
```

We see that the stress value is considerably high. We could consider a three-dimensional solution but for illustration purposes the 2D solution is just fine. In Figure 8 we give the joint configuration plot. We label all the judges and the 10 best and 10 worst rated bands.
As we have seen in this example, the data can have missing values. The `smacof` package also provides a permutation test implementation for unfolding.

### 7. Additional methods and implementations

#### 7.1. Unidimensional scaling

Unidimensional scaling is applied in situations where we have a strong reason to believe that there is only one interesting underlying dimension, such as time, ability, or preference.

Unidimensional scaling is often considered as a special one-dimensional case of MDS. However, it is often discussed separately, because the unidimensional case is quite different from the general multidimensional case. It has been shown that the minimization of the Stress target function with equal weights leads to a combinatorial problem when the number of dimensions of the target space is one (De Leeuw and Heiser 1977). The `smacof` package provides a simple implementation where all possible dissimilarity permutations are considered and the one which leads to a minimal stress is returned. Obviously, this strategy is feasible for small problems only (Mair and De Leeuw 2015).

In the following example we examine seven works by Plato. The chronological order of Plato’s works is unknown. Scholars only know that “Republic” was his first work, and “Laws” his last work. Unidimensional scaling can be used to map the works on a unidimensional continuum. The input dissimilarities are based on data from Cox and Brandwood (1959). They extracted the last five syllables of each sentence. Each syllable is classified as long or short which gives 32 types. Consequently, we obtain a percentage distribution across the 32 scenarios for each of the seven works.

```r
PlatoD <- dist(t(Plato7))
fit.uni <- uniscale(PlatoD)
```
Figure 8: Joint configuration plot for unfolding solution of album ratings.
Permutation 11 of 5040
Permutation 12 of 5040
Permutation 13 of 5040
Permutation 14 of 5040
Permutation 15 of 5040
Permutation 16 of 5040
Permutation 17 of 5040
Permutation 18 of 5040
Permutation 19 of 5040
Permutation 20 of 5040
Permutation 21 of 5040
Permutation 22 of 5040
Permutation 23 of 5040
Permutation 24 of 5040
Permutation 25 of 5040
Permutation 26 of 5040
Permutation 27 of 5040
Permutation 28 of 5040
Permutation 29 of 5040
Permutation 30 of 5040
Permutation 31 of 5040
Permutation 32 of 5040
Permutation 33 of 5040
Permutation 34 of 5040
Permutation 35 of 5040
Permutation 36 of 5040
Permutation 37 of 5040
Permutation 38 of 5040
Permutation 39 of 5040
Permutation 40 of 5040
Permutation 41 of 5040
Permutation 42 of 5040
Permutation 43 of 5040
Permutation 44 of 5040
Permutation 45 of 5040
Permutation 46 of 5040
Permutation 47 of 5040
Permutation 48 of 5040
Permutation 49 of 5040
Permutation 50 of 5040
Permutation 51 of 5040
Permutation 52 of 5040
Permutation 53 of 5040
Permutation 54 of 5040
Permutation 55 of 5040
Permutation 56 of 5040
Permutation 57 of 5040
Permutation 58 of 5040
Permutation 59 of 5040
Permutation 60 of 5040
Permutation 61 of 5040
Permutation 62 of 5040
Permutation 63 of 5040
Permutation 64 of 5040
Permutation 65 of 5040
Permutation 66 of 5040
Permutation 67 of 5040
Permutation 68 of 5040
Permutation 69 of 5040
Permutation 70 of 5040
Permutation 71 of 5040
Permutation 72 of 5040
Permutation 73 of 5040
Permutation 74 of 5040
Permutation 75 of 5040
Permutation 76 of 5040
Permutation 77 of 5040
Permutation 78 of 5040
Permutation 79 of 5040
Permutation 80 of 5040
Permutation 81 of 5040
Permutation 82 of 5040
Permutation 83 of 5040
Permutation 84 of 5040
Permutation 85 of 5040
Permutation 86 of 5040
Permutation 87 of 5040
Permutation 88 of 5040
Permutation 89 of 5040
Permutation 90 of 5040
Permutation 91 of 5040
Permutation 92 of 5040
Permutation 93 of 5040
Permutation 94 of 5040
Permutation 95 of 5040
Permutation 96 of 5040
Permutation 97 of 5040
Permutation 98 of 5040
Permutation 99 of 5040
Permutation 100 of 5040
Permutation 101 of 5040
Permutation 102 of 5040
Permutation 149 of 5040
Permutation 150 of 5040
Permutation 151 of 5040
Permutation 152 of 5040
Permutation 153 of 5040
Permutation 154 of 5040
Permutation 155 of 5040
Permutation 156 of 5040
Permutation 157 of 5040
Permutation 158 of 5040
Permutation 159 of 5040
Permutation 160 of 5040
Permutation 161 of 5040
Permutation 162 of 5040
Permutation 163 of 5040
Permutation 164 of 5040
Permutation 165 of 5040
Permutation 166 of 5040
Permutation 167 of 5040
Permutation 168 of 5040
Permutation 169 of 5040
Permutation 170 of 5040
Permutation 171 of 5040
Permutation 172 of 5040
Permutation 173 of 5040
Permutation 174 of 5040
Permutation 175 of 5040
Permutation 176 of 5040
Permutation 177 of 5040
Permutation 178 of 5040
Permutation 179 of 5040
Permutation 180 of 5040
Permutation 181 of 5040
Permutation 182 of 5040
Permutation 183 of 5040
Permutation 184 of 5040
Permutation 185 of 5040
Permutation 186 of 5040
Permutation 187 of 5040
Permutation 188 of 5040
Permutation 189 of 5040
Permutation 190 of 5040
Permutation 191 of 5040
Permutation 192 of 5040
Permutation 193 of 5040
Permutation 194 of 5040
The results of our multidimensional scaling are shown in Figure 9. Of course, we could perform multidimensional scaling through a regular MDS fit as well but it is pretty much guaranteed that we end up in a local minimum. For instance, using the default classical scaling starting configurations, we get a larger stress value (stress-1: 0.379) than with the permutation approach, which is not really surprising.
7.2. Inverse MDS

The basic problem of inverse MDS is to compute a dissimilarity matrix $\Delta$ from a given configuration matrix $X$. The corresponding theory can be found in De Leeuw and Groenen (1997) and De Leeuw (2012). Here, we just give a simple example using a subset of the kinship data. First, we fit an MDS on the kinship dissimilarities and perform an inverse MDS on the corresponding configuration matrix. This gives us seven dissimilarity matrices which also includes the trivial one with the Euclidean distances based on the fitted MDS configurations.

```r
D <- as.matrix(kinshipdelta)[1:6, 1:6]
fit <- mds(D)  ## MDS D --> conf
ifit <- inverseMDS(fit$conf)  ## inverse MDS conf --> D
```

Now we fit again seven MDS on the resulting inverse MDS dissimilarity output matrices and look at the configurations (see Figure 10)

```r
op <- par(mfrow = c(3,3))
plot(fit, main = "Original MDS")
for (i in 1:length(ifit)) {
  fit.i <- mds(ifit[[i]])
  plot(fit.i, main = paste0("Inverse MDS (", i, ")"))
}
par(op)
```

Note that so far smacof provides a very basic implementation only and it can happen that some of the output dissimilarities are negative. Future extensions will implement the more sophisticated theory from De Leeuw (2012).

7.3. Procrustes

The Procrustes problem is the following: We have two known $n \times p$ matrices $X$ and $Y$ (MDS
Figure 10: Configuration plots resulting from MDS fit on 7 dissimilarity matrices computed with inverse MDS. The first panel (top left) gives the original solution.
SMACOF in R

configurations). \(X\) is the target matrix. We want to transform \(Y\) such that the configurations in the resulting matrix \(\hat{Y}\) are as close as possible to the ones given in \(X\). \(Y\) is subject to three transformations: rotation (\(T\) as rotation matrix), dilation (\(s\) as dilation factor), and translation (\(t\) as translation vector). Technical details can be found in Borg and Groenen (2005), Chapter 20; here we just summarize the basic computations. We need the restriction \(T'T = I\) and \(J = I - n^{-1}11'\) as the centering matrix.

1. Compute \(C = XJY\).
2. Compute the SVD of \(C\); that is, \(C = P\Phi Q'\).
3. The optimal rotation matrix is \(T = QP'\).
4. The optimal dilation factor is \(s = (\text{tr}X'JYT)/(\text{tr}Y'JY)\).
5. The optimal translation vector is \(t = n^{-1}(X - sYT)'1\).

If we just want to quantify the configurational similarity between the two configurations \(X\) and \(Y\), we can compute a congruence coefficient based on the configuration distances:

\[
c(X, Y) = \frac{\sum_{i<j} d_{ij}(X)d_{ij}(Y)}{\sqrt{\sum_{i<j} d_{ij}^2(X)}\sqrt{\sum_{i<j} d_{ij}^2(Y)}}
\]

(21)

As an example we use a dataset which contains correlations between 13 work values. We have data for (back then) East and West Germany. First we fit two separate MDS: one for East and one for West Germany. We abbreviate the object labels for better plotting. For the second MDS we use a classical scaling as starting solution; otherwise the signs would be switched. By doing this the Procrustes transformation is easier to see in the plots.

```r
eastD <- sim2diss(EW_eng$east)
attr(eastD, "Labels") <- abbreviate(attr(eastD, "Labels"))
fit.east <- mds(eastD, type = "ordinal")
westD <- sim2diss(EW_eng$west)
attr(westD, "Labels") <- abbreviate(attr(westD, "Labels"))
fit.west <- mds(westD, type = "ordinal", init = torgerson(eastD))

Now we perform a Procrustes transformation with the East Germany configurations as target \(X\), the West Germany configurations as testee \(Y\). We also get the congruence coefficient.

```r
fit.proc <- Procrustes(fit.east$conf, fit.west$conf)
fit.proc
```

```
## Call: Procrustes(X = fit.east$conf, Y = fit.west$conf)
## ## Congruence coefficient: 0.965
## ```
## Rotation matrix:
## D1  D2
## D1  1.00 -0.01
## D2  0.01  1.00
##
## Translation vector: 0 0
## Dilation factor: 0.923

Finally, Figure 11 shows the configurations $X$ and $Y$ from the separate MDS fits and the Procrustes fit. For Procrustes we have two plots: one plots $X$ and $\hat{Y}$, the other one plots $Y$ and $\hat{Y}$ showing the change due to Procrustes.

```r
op <- par(mfrow = c(2,2))
plot(fit.east, main = "MDS East Germany")
plot(fit.west, main = "MDS West Germany")
plot(fit.proc)
plot(fit.proc, plot.type = "transplot", length = 0.05)
par(op)
```

### 7.4. MDS Jackknife

De Leeuw and Meulman (1986) proposed a jackknife strategy for MDS in order to examine the stability of a solution. The MDS jackknife approach, as implemented via the `jackknife()` function in `smacof` computes $i = 1, \ldots, n$ additional solutions with configurations $X_i^*$. Note that $X_i^*$ denotes the solution where object $i$ is left out. Each of these configurations is subject to a Procrustes transformation with target configuration $X$, i.e. the original solution. In addition, the average (centroid) jackknife solution $\bar{X}^*$ can be computed. It total, we have $n + 2$ comparable configurations which can be represented in a single plot.

A stability measure (Heiser and Meulman 1983) can be computed as follows:

$$ ST = 1 - \frac{\sum_{i=1}^{n} \|X_i^* - \bar{X}^*\|^2}{\sum_{i=1}^{n} \|X_i^*\|^2}.$$  

It can be interpreted as the ratio of between and total variance. To measure the cross-validity, i.e. comparing the “predicted” configuration of object $i$ as the $i$-th row in $\bar{X}^*$ with the actual configuration ($i$-th row in $X$), we can compute

$$ CV = 1 - \frac{n\|X - \bar{X}^*\|^2}{\sum_{i=1}^{n} \|X_i^*\|^2}.$$  

Finally, the dispersion around the original solution $X$ can be expressed as
Figure 11: Top panels: configuration plots for separate MDS fits. Bottom left panel: Original East German configuration (blue) with Procrustes transformed West German configuration (red). Bottom right panel: Original West German configuration (gray) with Procrustes transformation (red).
\[ DI = \frac{1}{n} \sum_{i=1}^{n} \| X_i^* - X \|^2 \]

\[ = \frac{1}{n} \sum_{i=1}^{n} \| X_i^* - \bar{X}^* \|^2 + \| X - \bar{X}^* \|^2 \]

\[ = 2 - (ST + CV) \]

As an example let’s just look at the Lawler dataset once more. First we fit a two-dimensional interval MDS on the data and then we perform the leave-one-out jackknife. The resulting plot is given in Figure 12. Note that the labels denote the original solution \( X \), the small dots the individual configurations \( X_i^* \), and the bigger dots the corresponding centroid \( \bar{X}^* \), connected to each \( X_i^* \).

```r
fit.lawler <- mds(LawlerD, type = "interval")
jackfit <- jackknife(fit.lawler)
jackfit

# Call: jackknife.smacofB(object = fit.lawler)
#
# SMACOF Jackknife
# Number of objects: 9
# Value loss function: 0.6583261
# Number of iterations: 13
#
# Stability measure: 0.9854694
# Cross validity: 0.9944329
# Dispersion: 0.02009773
```

plot(jackfit)

8. Discussion and Outlook

The \texttt{smacof} package provides the most comprehensive framework to perform MDS R. Future implementations will include nonmetric unfolding (Busing \textit{et al.} 2005) and modeling asymmetric data as given in Borg and Groenen (2005), Chapter 23.

References

Figure 12: Jackknife MDS plot: The labels denote the original MDS configuration. The small dots the configurations resulting from leave-one-out procedure. The big dots, connected to the small dots, denote their centroids.


**Affiliation:**

Patrick Mair  
Department of Psychology  
Harvard University  
E-mail: mair@fas.harvard.edu  
URL: [http://scholar.harvard.edu/mair](http://scholar.harvard.edu/mair)