Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

Flexible Multidimensional Scaling with the R Package smacofx

Thomas Rusch
WU Vienna University of Economics and Business

Patrick Mair
Harvard University

Jan de Leeuw
UCLA

Kurt Hornik
WU Vienna University of Economics and Business

Abstract

In this article we introduce the R package **smacofx** which offers various implementations of flexible multidimensional scaling (MDS) techniques for two-way data. These techniques allow for flexible MDS via utilizing combinations of optimal scaling and specific parametric transformations of the input dissimilarities and/or the fitted distances, and/or residual weighting. Examples of techniques available in the package include Sammon mapping, power-stress and r-stress MDS, Box-Cox MDS, local MDS, (extended) curvilinear component analysis and curvilinear distance analysis. Various post-fit functionality is available that matches the functionality available in **smacof** (De Leeuw and Mair 2009), including a broad array of plots to visualize and communicate the results. The package further includes tools and functions for goodness-of-fit assessment, local optimum diagnostics and MDS biplots. All the package functionalities are illustrated using two data applications.

Keywords: proximity scaling, dimensionality reduction, multivariate statistics, manifold learning, data visualization, ordination.

1. Introduction

When faced with data in the form of (multivariate) proximities, it is common to represent and visualize them in lower dimensions by means of techniques that are collected under various headings including ordination, nonlinear dimension reduction or manifold learning. Many of these techniques are offshots of multidimensional scaling (MDS) and are especially popular for unsupervised learning, dimensionality reduction, graph drawing and exploratory data

analysis. They have the aim and ability to provide a visualisation of the data, elicit intrinsic dimensions or remove unwanted ones, provide a scaling of the data into a latent space, suggest hypotheses or communicate complex multivariate association.

These MDS techniques apply to a single symmetric matrix of pairwise proximities between objects i and object j (i.e., two-way data). The goal is to find a representation (embedding) of the objects in a lower dimensional target space in such a way that the distances in the lower-dimensional space optimally approximate the proximities. This embedding/representation is also called the configuration of objects and the object positions in the configuration are the quantities of main interest to be estimated (which is done via fitting distances between the objects). MDS techniques differ primarily in how the proximities and fitted distances are treated and also what is considered to be an optimal approximation. Some examples are "classical" Torgerson scaling (Torgerson 1958), metric and non-metric MDS (Kruskal 1964), Sammon mapping (Sammon 1969) or power-stress MDS (POST-MDS, Buja, Swayne, Littman, Dean, Hofmann, and Chen 2008; Groenen and De Leeuw 2010).

Some MDS methods feature hyperparameters that offer extra flexibility to represent and explore the data in manifold ways, yielding what we call "flexible MDS". These techniques are the subject of this article, which introduces the R (R Core Team 2025) package **smacofx** (Rusch, De Leeuw, Chen, and Mair 2025) and its functionality. This package extends the package **smacof** (De Leeuw and Mair 2009; Mair, Groenen, and De Leeuw 2022) which provides the basic software design, infrastructure such as S3 generics and basic functionality for two-way data, including unidimensional scaling (Coombs 1950), Torgerson scaling, metric and non-metric MDS, constrained MDS (De Leeuw and Heiser 1980), spherical MDS (Cox and Cox 2001), unfolding (Coombs 1950) as well as models for three-way data like INDSCAL and IDIOSCAL (Carroll and Chang 1970, for both). Building on top of that, the package **smacofx** extends the basic functionality in **smacof** for two-way data with various flexible MDS methods that apply parametric transformations on the proximities and/or distances and/or weights (e.g., power transformation or Box-Cox transformations). This includes MDS methods such as MULTISCALE (Ramsay 1977), ALSCAL (Takane, Young, and De Leeuw 1977), local MDS (Chen and Buja 2009), elastic scaling (McGee 1966) or r-stress MDS (De Leeuw, Groenen, and Mair 2016); we describe all of them in detail below. The present article describes the available MDS methods for obtaining a configuration given transformation hyperparameters, the infrastructure in **smacofx** and provides a manual how to use the functionality of **smacofx**.

Related work. As mentioned, in the present work there is a strong connection to the R package **smacof** (De Leeuw and Mair 2009; Mair *et al.* 2022). The functionality of the packages **smacofx** is modeled after what is available in **smacof**.

Multidimensional scaling models are implemented in other R packages as well. In **stats**, there is `cmdscale` for principal coordinates analysis (aka Torgerson Scaling). The recommended package **MASS** (Venables and Ripley 2002) allows to fit non-metric MDS (`isoMDS()`). Further contributed packages allow to fit MDS models: Torgerson scaling can also be carried out with `mmds()` from the **DistatisR** package (Beaton and Abdi 2022), which also allows to fit MDS models to three-way data. The package **vegan** (Oksanen, Simpson, Blanchet, Kindt, Legendre, Minchin, O'Hara, Solymos, Stevens, Szoecs, Wagner, Barbour, Bedward, Bolker, Borcard, Carvalho, Chirico, De Caceres, Durand, Evangelista, FitzJohn, Friendly, Furneaux, Hannigan, Hill, Lahti, McGlinn, Ouellette, Ribeiro Cunha, Smith, Stier, Ter Braak, and Weedon 2022) offers weighted principal coordinates analysis (`wcmdsale()`), non-metric

MDS (`monoMDS()`) and Isomap (`isomap()`). Torgerson scaling (`pco()`) and non-metric MDS (`nmds()`) is also implemented in **labdsv** (Roberts 2023) and also with the same function names in **ecodist** (Goslee and Urban 2007). Package **ExPosition** (Beaton, Fatt, and Abdi 2014) implements Torgerson scaling with `coreMDS()`. The package **MLDS** (Knoblauch and Maloney 2008) allows to fit maximum likelihood difference scaling with function `mlds()`. The package **cops** (Rusch and Mair 2024) offers an implementation of cluster optimized proximity scaling (Rusch, Mair, and Hornik 2021). Beyond our focus are asymmetric proximity matries, but the package **asymmetry** (Zielman 2022) offers the functions `asymscal()`, `slidevector()` and `mdsunique()` to analyze them. Three-way and/or asymmetric proximity data can also be analyzed via `semds()` in **semds** (Vera and Mair 2019).

The package **smacofx** offers implementations of MDS techniques that are more flexible than the ones mentioned above, mainly by allowing for weights, explicit power transformations for any combination of proximities, distances and weights and implicit metric (ratio, interval or spline) or non-metric optimal scaling transformations of proximities. For the results obtained, we can utilize a range of post-fitting functions similar to those that are also available in **smacof**.

The rest of this article is organized as follows: It starts with a description of proximity scaling and implicit and explicit transformations in Section 2. In Section 3 we discuss basic infrastructure available for fitted models. In Section 4 we list MDS methods in **smacofx** for obtaining a single configuration from the proximity matrix. After that, we discuss the main functionality of the post-fit infrastructure available (and matching the ones in **smacof**) in Section 5. Throughout the article we already illustrate the use of the functions in various places, but a streamlined fully worked example analysis is featured in Section 6. Concluding remarks can be found in Section 7.

2. Multidimensional scaling

In MDS techniques the starting point are proximities between n data points or objects i and $j; i, j = 1, \ldots, n$. They can either be direct observations (e.g., collected in an experiment) or derived from objects' variables (e.g., distances based on a variable vector). See Cox and Cox (2001) for a discussion on proximities and how to choose them.

The proximities are given as the $n \times n$ matrix Δ containing the pairwise proximities between the objects pairs i, j, with individual entries δ_{ij}. The Δ constitutes the main input in all our fitting functions. We limit ourselves to dissimilarities or distances as proximities; this implies that Δ is symmetric, non-negative and the main diagonal is 0 (i.e., $\delta_{ij} = \delta_{ji} \geq 0$ and $\delta_{ii} = 0$). If the proximities are given as similarities they need to be turned into dissimilarities first (e.g., with the function `smacof::sim2diss()`).

We denote with $\hat{\Delta}$ a disparities matrix, with entries $\hat{\delta}_{ij}$ resulting after transformations were applied to the elements of Δ. We distinguish between transformations that are explicit or implicit and parametric or nonparametric; we'll later talk more about which specific transformations are offered in our implementations. Following Rusch, Mair, and Hornik (2023a) we denote explicit transformation as $T_\Delta(\delta_{ij}|\theta_\Delta)$ with parameter vector θ_Δ; here the main interest lies in transformations that are univariate, parametric and chosen by the analyst. For the software we will also make the small conceptual distinction of the T_Δ being an argument to a MDS method fitting function vs. the T_Δ being manually applied by the analyst beforehand. We may also have implicit optimal scaling transformations, which we denote by f, which

are either parametric or nonparametric. For these transformations we do not specifically designate a part of the parameter vector θ_Δ but we treat them as nuisance parameters that simply get optimized "under the hood". In the end, we thus have two possible transformations applied to our proximities and the disparities are $\hat{\delta}_{ij} := f\left(T_\Delta(\delta_{ij}|\theta_\Delta)\right)$.

The main output of an MDS procedure is the point configuration, a spatial arrangement of the n points so that distances between the points in the configuration optimally approximate the $\hat{\delta}_{ij}$ based on a criterion for what is optimal. This configuration is denoted by X and is a $n \times p$ matrix; the index $s = 1, \ldots, p$ denotes the column dimensions of X in the target space which usually is of much lower dimension than n (typically $p = 2$). This space is almost always the Euclidean space, but in principle can be any metric space. If $p \in \{1, 2, 3\}$ the spatial arrangement of points in the configuration can be conveniently visualized; the positions then represent the coordinates of the points in the target space.

What we primarily estimate in MDS are the points (row vectors) x_i, x_j in X via fitting pairwise distances $d_{ij}(X) = d(x_i, x_j)$ between the points x_i, x_j. The matrix of pairwise distances with elements $d_{ij}(X)$ is denoted by $D(X)$. The fitted distances $d_{ij}(X)$ can be reconstructed from X by applying the distance definition used, typically the Euclidean distance,

$$d_{ij}(X) = ||x_i - x_j||_2 = \left(\sum_{s=1}^{p} |x_{is} - x_{js}|^2\right)^{1/2} \quad i, j = 1, \ldots, n. \tag{1}$$

although there exist MDS versions that use other Minkowski distances (e.g., Groenen, Mathar, and Heiser 1995). In the presented packages, we limit ourselves to the Euclidean distance but we allow flexibility by adding an extra layer of abstraction between X and Δ: we may fit transformed Euclidean distances $\hat{d}_{ij}(X) = T_D(d_{ij}(X)|\theta_D))$. Expressed as a matrix, this is $\hat{D}(X)$ with the elements being $\hat{d}_{ij}(X)$. For the $\hat{d}_{ij}(X)$ we have the explicit parameter vector θ_D. Note that X is still in Euclidean space.

With these building blocks we can say that what we do in MDS and our software is to find X so that we approximate the $\hat{\Delta}$ optimally by the $\hat{D}(X)$ or $\hat{\delta}_{ij} = f(T_\Delta(\delta_{ij}|\theta_\Delta)) \approx \hat{d}_{ij}(X) = T_D(d_{ij}(X)|\theta_D)$. Some MDS methods also allow for explicit weighting of the approximation with an input weight matrix W and finite elements w_{ij}, e.g., by setting them to zero if a dissimilarity is missing. The weights may also be subject to explicit transformations so we can further have $\hat{w}_{ij} = T_W(w_{ij}|\theta_W)$. We will collect all the explicit transformations parameters in a single vector $\theta = (\theta_\Delta, \theta_D, \theta_W)^\top$. What is left is to define what is meant by optimal.

There are many different ideas what constitutes an optimal approximation and these give rise to different versions of MDS we support in our package. What all of them have in common is to define a loss function based on the approximation error of the $\hat{\Delta}$ by the $\hat{D}(X)$, which we will call a MDS badness-of-fit criterion denoted by σ. We use an optional designator specifiying the name of the fit criterion or method as a subscript, e.g., σ_{ALSCAL} which means "badness-of-fit for ALSCAL", with the MDS subscript designating a general MDS badness-of-fit.

In what follows we use MDS badness-of-fit criteria of the following kind,

$$\sigma_{MDS}(X|\theta) = O\left(\hat{\Delta} = \left[f(T_\Delta(\Delta|\theta_\Delta))\right], \hat{D}(X) = \left[T_D(D(X)|\theta_D)\right], \hat{W} = \left[T_W(W|\theta_W)\right]\right) \tag{2}$$

with O denoting an objective function with the arguments $\hat{\Delta}$ and $\hat{D}(X)$ and optionally $\hat{W}$. For a method to be a MDS method we need both Δ and $D(X)$.

Many badness-of-fit proposals have been introduced over the years that Rusch *et al.* (2023a) classify into three main groups based on the type of loss function O used:

First, the *strain* family (dating back to Torgerson 1958) which uses the negative of the doubly centered dissimilarities as the $\hat{\Delta}$ (so, doubly centered similarities) and approximates them with the inner product matrix of X as the $\hat{D}(X)$. The resulting MDS method goes by many names, such as classical MDS, principal coordinates analysis, principal axes scaling or Torgerson scaling. Strain-based MDS methods are appealing because the optimization can be solved with an eigenvalue decomposition. Their drawback is that they are not particularly flexible and only accommodate explicit transformations T_Δ, if any.

Second, the family of least squares methods that use a quadratic loss for O, typically called *stress* (dating back to Kruskal 1964). Stress-based MDS methods are arguably the most common. The majority of methods we offer in **smacofx** are stress-based. These methods can be very flexible and may allow implicit transformations f and explicit transformations T_Δ and T_D. Stresses typically are non-convex, however, and have to be solved iteratively, making them difficult to optimize. Algorithms following the majorization-minimization or minorization-maximization (MM) principle (De Leeuw 1977; Lange 2016) and gradient descent algorithms (Curry 1944) are the most frequently encountered optimization methods used.

Third, the family of *energy models* (Noack 2007) that conceptualize MDS as a weighted combination of a repulsion force proportional to $-\Delta D(X)$ and an attraction force proportional to $D(X)^\mu$ of the objects. This family is popular in graph drawing. Energy models can be quite flexible with respect to allowing explicit transformations T_Δ and T_D but were only envisioned for metric scaling. They are typically optimized with gradient descent algorithms.

In Section 4 we describe which O and T_Δ and T_D we specifically support in our software; the methods available in our software comprise a large array of MDS methods including the most popular MDS models that we are aware of.

To instantiate flexible MDS models specifically, one would plug in the transformations to the respective O. For example, for stress-based MDS this is

$$\sigma_{MDS}(X|\theta) = \sum_{i<j} T_W\left(w_{ij}|\theta_W\right)\left(f\left(T_\Delta(\delta_{ij}|\theta_\Delta)\right) - T_D(d_{ij}(X)|\theta_D)\right)^2 \tag{3}$$

Minimizing the badness-of-fit (2) criterion means finding the optimal configuration X^* out of all possible X as

$$\begin{aligned} X^* &= \arg\min_X \sigma_{MDS}(X|\theta) \\ &= \arg\min_X O\left(\hat{\Delta} = \left[f(T_\Delta(\Delta|\theta_\Delta))\right], \hat{D}(X) = \left[T_D(D(X)|\theta_D)\right], \hat{W} = \left[T_W(W|\theta_W)\right]\right) \end{aligned} \tag{4}$$

This is the task of proximity scaling and can be quite difficult. Note that in our packages we do not optimize Eq. 4 in this generality but use tailored algorithms for specific concrete instances, either as MM or gradient descent algorithms. We refer to the publications that contain the respective technical details throughout this article.

2.1. Implicit transformations: Metric and non-metric MDS

We mentioned that we can have transformations for the dissimilarities Δ that are implicit or explicit. For the latter we have an explicit parameter vector θ_Δ and we will talk more

about this when we discuss the specific methods in Section 4. Implicit transformations are traditionally used in MDS under the term "optimal scaling". They are typically used to accommodate the scale levels of the dissimilarities (Borg and Groenen 2005). The main distinction is between metric MDS (ratio, interval, monotone spline transformations) and non-metric MDS (ordinal transformations). These transformations are admissable under the assumed scale level of the dissimilarities, and are under the constraint that $f(\delta_{ij}) \geq 0$:

$$\begin{aligned} \text{ratio: } & f(\delta_{ij}) = b\delta_{ij} \\ \text{interval: } & f(\delta_{ij}) = a + b\delta_{ij} \\ \text{mspline: } & f(\delta_{ij}) \text{ being a monotone smooth function} \\ \text{ordinal: } & f(\delta_{ij}) \text{ being rank-preserving monotonic} \end{aligned}$$

In the ordinal transformation one uses isotonic regression to find the optimal rank-order preserving monotonic f, which implies that $\hat{d}_{ij}(X) \approx \hat{\delta}_{ij} \leq \hat{d}_{kl}(X) \approx \hat{\delta}_{kl}$ if $\delta_{ij} < \delta_{kl}$. Ties (i.e., $\delta_{ij} = \delta_{kl}$) can also be handled in two other ways (secondary or tertiary approach, Borg and Groenen 2005). For montone splines, an I-spline (Ramsay 1988) is used. Note that these implicit transformations are functions of all δ_{ij}.

These transformations are implicit in the presented MDS methods, which means that the f or its parameters are not of interest by themselves but nuisance parameters. They are found automatically during the scaling process via an inner optimization step that finds the optimal implicit transformation and there is no interference by the user other than specifying the type of optimal scaling. Because of that we do not specifically refer to these as parameters in our models, but call this the "MDS type" as it relates to choosing an appropriate MDS for the nature of the dissimilarities. They can be selected with the `type` argument, which should be the character strings `"ratio"` for ratio, `"interval"` for interval, `"mspline"` for monotone spline and `"ordinal"` for non-metric (so one would use `type="ordinal"` to get a non-metric MDS type). If distances (metrics or pseudo-metrics) are used for δ_{ij}, then ratio, interval or spline transformations can and should be considered. If the δ_{ij} are dissimilarities that lack the properties of a distance (e.g., do not satisfy the triangle inequality) and/or are experimentally determined from human raters where transitivity cannot be expected than the ordinal transformation is appropriate. We will give an overview which implicit transformations are possible in the techniques that we support in the presented packages in Table 1.

2.2. Input dissimilarities

In MDS in general and our software in particular, one can use any dissimilarity matrix that is symmetric and hollow (i.e., non-negative). This includes matrices of any metric norm, distance, pseudo-distance and divergence measure. Dissimilarities can either be directly observed (via an experiment) or derived from a data matrix, by using an appropriate dissimilarity measure to capture the essence of the data.

Within the R ecosystem one can manually create a dissimilarity matrix or use a huge variety of functions that return dissimilarities either as a symmetric matrix, symmetric data frame or '`dist`' object. All of them can be used with the functions in **smacof** and **smacofx**, including all the distances and dissimilarities in the `dist()` function and the functions in **vegan** (Oksanen *et al.* 2022), **proxy** (Meyer and Buchta 2022) and **analogue** (Simpson 2007) or the ϕ-distance in **cops** (`cops::phidistance()`, Rusch, Venturo-Conerly, Baja, and Mair 2023b). We also

offer a distance function in the **smacofx** package to be used with count data or histograms, the blended χ^2-distance in (`bcsdistance()`).

We also single out `vegan::isodist()` which implements geodesic distances, i.e., distances between points as imposed by a weighted neighbourhood graph with either the parameter k for the number of nearest neighbours, or ε as the neighbourhood radius. Geodesic distances approximate (curvilinear) distances along a manifold. They are used for manifold learning, for example in Isomap (Tenenbaum, De Silva, and Langford 2000) which is a strain MDS with the $\hat{\delta}_{ij}$ being the geodesic distance. Note that a $\hat{\Delta}$ obtained from geodesic distances can also be used in other stress or energy badness-of-fit functions and subjected to additional implicit transformations and weighting as we do below. The explicit transformation for geodesic distances is governed by $\theta_\Delta = k$ or $\theta_\Delta = \varepsilon$ as the parameter that defines the neighborhood graph.

The reason for singling out this dissimilarity is that MDS variants applied to geodesic distances are generally useful for manifold learning and often constitute methods in their own right (e.g., curvilinear distance analysis is curvilinear component analysis with gedodesic distances). We stress that all our functions can take geodesic distances as input (or do it automatically) and that way all presented flexible MDS method can be turned into Isomap-type manifold learning techniques as well.

3. Design principles of package functionality

One of the design principles of the **smacofx** package was for it to be largely compatible and comparable with the infrastructure in the **smacof** package. This was not only because conceptually the functionality is related to the functionality in **smacof** (i.e., two-way MDS and built around MM algorithms), but also because **smacof** offers generics to postprocess and extend fitted MDS models, see De Leeuw and Mair (2009); Mair *et al.* (2022) and Section 5.

This is reflected in our package in two ways: First, we made the UI as similar as possible between the packages and **smacof** by using the same arguments whenever we refer to the same aspect of a MDS model. For example, `type` is the argument to choose the implicit optimal scaling transformation both in **smacof** as well as in **smacofx**. The one exception is the desired accuracy at which to stop the iterations; we call the argument `acc` whereas in **smacof** it is called `eps`. Second, we chose an object-oriented approach to various MDS models and made sure that the objects returned by the functions from **smacofx** can be used with the appropriate ideas and generics of **smacof**.

To that end, we use the S3 system and designed a unified output for models with explicit transformation parameters that return an X. These objects are of class '`smacofP`' and inherit the class '`smacofB`' in **smacof**. We implemented S3 methods for many **smacof** generics to make use of the functionality in **smacof** designed for '`smacofB`' objects also with objects of class '`smacofP`'. We also wrote wrappers for `stats::cmdscale()` and `MASS::sammon()` to return objects of class '`cmdscalex`' that are largely similar to '`smacofB`', allowing them to also use specific functionality, especially with respect to the `plot()` method. This way we believe users familiar with the package **smacof** can use the new package with little additional effort.

For the class '`smacofP`' we implemented methods for the standard generics:

`print()`: Prints basic model information, typically including the call, the fitted MDS Model,

the number of objects, and the square root of the badness-of-fit value at convergence (stress-1).

`summary()`: The configuration X and the percentage contribution of each point to the overall badness-of-fit ("stress per point") (only recommended for models with low n).

`coef()`: Returns the explicit transformation parameters used in fitting.

`plot()`: The plot method features a number of plot types that can be selected via the `plot.type` argument. See Section 5 and the help file `?plot.smacofP`.

We make the functionality in R accessible via

```
R> library("smacofx")
```

An overview of the main fitting functions available in **smacofx** are given in Table 1. As far as we know this functionality is not duplicated anywhere else in the R ecosystem.

Throughout this article we make use of a data set as the running example. It is fairly small, suitable for quick illustration and allows to show all the functionality of the package. The data are from Koller, Floh, Zauner, and Rusch (2013) and consist of responses of $n = 1013$ people to items related to consumer susceptibility to interpersonal influence (the 12 item CSII scale, Bearden, Netemeyer, and Teel 1989) and self-concept clarity (the 12 item SCC scale, Campbell, Trapnell, Heine, Katz, Lavallee, and Lehman 1996). The responses were recorded on a 5-point scale for each item, with 5 denoting "fully disagree"; the 10th item of the SCC scale was reversed. Theoretically, the two concepts are distinct but related. Self-concept clarity may be seen as an antecedent to consumer's level of susceptibility to interpersonal influence and can explain its level. People that have a clear and strong self-image are less prone to being interpersonally influenced, whereas people that tend to doubt themselves are more susceptible (Koller *et al.* 2013). We use Euclidean distances between the scale items.

```
R> data("koller")
R> dis <- dist(t(koller))
```

4. Multidimensional scaling methods in smacofx

In this section we describe MDS variants in **smacofx** and illustrate them with the data of Koller *et al.* (2013). To establish a baseline we first fit standard MDS versions from **smacof**.

```
R> mds0 <- mds(dis)
R> mds0i <- mds(dis, type = "interval")
R> mds0o <- mds(dis, type = "ordinal")
```

The configurations[1] for the ratio, interval and ordinal MDS are given in Figure 1. All of the plots in this section will be Procrustes adjusted to the ratio MDS solution with the function `alignplot()`. It takes a list of objects `objectlist`, Procrustes adjusts the configurations in the list objects to a reference configuration (argument `reference`).

[1] In what follows, slight discrepancies between numbers and figures can appear for different operation systems and different low-level linear algebra libraries (e.g., LAPACK). The results in this article were produced with R version 4.5.1, **smacofx** version 1.21-1 on Linux Mint 22.1 with LAPACK 3.12.0.

Method	Scale level	type	R function
Sammon Mapping	Metric	`"ratio"`, `"interval"`	`sammonmap()`
Elastic Scaling	Metric	`"ratio"`, `"interval"`	`elscal()`
ALSCAL	Metric	`"ratio"`, `"interval"`	`alscal()`
MULTISCALE	Metric	`"ratio"`	`multiscale()`
POST-MDS	Metric	`"ratio"`, `"interval"`	`postmds()`
Restricted POST-MDS	Metric	`"ratio"`, `"interval"`	`rpostmds()`
Approximate POST-MDS	Metric	`"ratio"`, `"interval"`	`apostmds()`
R-Stress MDS	Metric	`"ratio"`, `"interval"`, `"mspline"`	`rstressmds()`
	Non-Metric	`"ordinal"`	
Box-Cox MDS	Metric	`"ratio"`	`bcmds()`
Local MDS	Metric	`"ratio"`	`lmds()`
Curvilinear Component Analysis	Metric	`"ratio"`	`clca()`
Curvilinear Distance Analysis	Metric	`"ratio"`	`clda()`
Extended Curvilinear Component Analysis	Metric	`"ratio"`, `"interval"`, `"mspline"`	`eclca()`
Extended Curvilinear Power Component Analysis	Metric	`"ratio"`, `"interval"`, `"mspline"`	`eclda()`
Extended Curvilinear Distance Analysis	Metric	`"ratio"`, `"interval"`, `"mspline"`	`eclpca()`
Extended Curvilinear Power Distance Analysis	Metric	`"ratio"`, `"interval"`, `"mspline"`	`eclpda()`

Table 1: MDS methods that can be fit with functions in **smacofx**.

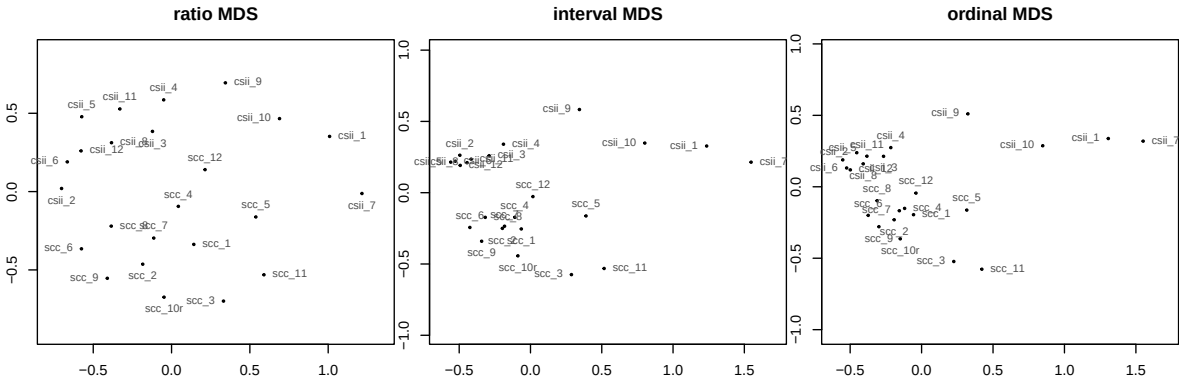

Figure 1: Ratio, interval and ordinal MDS configurations for the Koller data.

```
R> cg <- "grey30"
R> olist <- list(mds0, mds0i, mds0o)
R> par(mai = c(0.27, 0.27, 0.4, 0))
R> alignplot(olist, mds0$conf,
+           mains = c("ratio MDS", "interval MDS", "ordinal MDS"),
+           label.conf = list(pos = 5, col = cg), xlab = "",
+           ylab = "", asp = 1)
```

We see there is little spatial separation between the SCC and the CSII items in the ratio MDS, some in ordinal MDS and most in interval MDS. We also see that the CSII items 1, 7, 9 and 10 are scaled away from the rest, almost like a third scale. We will see how this changes with other MDS types.

4.1. Sammon mapping

Sammon mapping (Sammon 1969) is a nonlinear stress-based MDS method that weights the squared residuals of dissimilarities and fitted distances with the inverse dissimilarities. Originally envisioned for ratio MDS, we extend it to allow for both metric implicit optimal scaling transformations of input dissimilarities δ_{ij} (ratio, interval) and weights.

The badness-of-fit can be formulated as

$$\sigma_{\text{Sammon}}(X) = \sum_{i<j} w_{ij} \frac{\left(\hat{\delta}_{ij} - d_{ij}(X)\right)^2}{\delta_{ij}} = \sum_{i<j} w_{ij} \frac{\left(f(\delta_{ij}) - d_{ij}(X)\right)^2}{\delta_{ij}} \tag{5}$$

with $\hat{\delta}_{ij}$ being the transformed dissimilarities and the w_{ij} being non-negative weights (with 0 if the δ_{ij} is missing). We optimize this with MM (De Leeuw 1977).

The function to get a configuration is called `sammonmap()` and is part of the **smacofx** package. Its basic usage is `sammonmap(delta, type, weightmat)` with `delta` being an input dissimilarity matrix (can be pre-transformed), `type` specifying the MDS type/implicit transformation (one of `"ratio"` or `"interval"`) and `weightmat` being the symmetric matrix of weights $[w_{ij}]$.

For our data example, we can fit an interval Sammon mapping with

```
R> sam <- sammonmap(dis, type = "interval")
```

The configuration is displayed in Figure 2 (corresponding code in Section 4.4).

4.2. Elastic scaling

Elastic scaling (McGee 1966) is similar to Samon mapping but weights the squared residuals of dissimilarities and fitted distances with the square of the inverse dissimilarities, so weights them more heavily. Originally envisioned for ratio MDS, we extend it to allow metric implicit transformation f for the input dissimilarities δ_{ij} and weights. This can be formulated as

$$\sigma_{\text{elastic}}(X) = \sum_{i<j} w_{ij} \frac{\left(\hat{\delta}_{ij} - d_{ij}(X)\right)^2}{\delta_{ij}^2} = \sum_{i<j} w_{ij} \frac{(f(\delta_{ij}) - d_{ij}(X))^2}{\delta_{ij}^2} \tag{6}$$

with $\hat{\delta}_{ij}$ being the transformed dissimilarities and the w_{ij} being non-negative weights (with 0 if the δ_{ij} is missing). We again optimize this with MM.

Standard usage is `elscal(delta, type, weightmat)` with `delta` being an input dissimilarity matrix (can be pre-transformed), `type` specifying the MDS type/implicit transformation (one of `"ratio"` or `"interval"`) and `weightmat` being the symmetric matrix of weights $[w_{ij}]$.

For our data example, we can fit a ratio elastic scaling model as

```
R> els <- elscal(dis, type = "ratio")
```

The configuration is displayed in Figure 2 (corresponding code in Section 4.4).

4.3. ALSCAL

ALSCAL or s-stress MDS (Takane *et al.* 1977) is a metric MDS method that uses an explicit square transformation for input dissimilarities δ_{ij} and the fitted distances $d_{ij}(X)$ as well as the ratio and interval implicit transformations in a stress-type loss. The associated stress is

$$\sigma_{\text{ALSCAL}}(X|\theta) = \sum_{i<j} w_{ij} \left(\hat{\delta}_{ij} - \hat{d}_{ij}(X)\right)^2 = \sum_{i<j} w_{ij} \left(f(\delta_{ij}^2) - d_{ij}(X)^2\right)^2 \tag{7}$$

with $\hat{\delta}_{ij}, \hat{d}_{ij}(X)$ being the transformed dissimilarities and fitted distances respectively. The explicit transformations are $T_\Delta(\delta_{ij}|\theta_\Delta) = \delta_{ij}^2$ and $T_D(d_{ij}(X)|\theta_D) = d_{ij}(X)^2$ with explicit parameter vector $\theta = (\theta_\Delta, \theta_D)^\top = (2, 2)^\top$. The implicit transformations allowed are the ratio and interval transformations. We optimize this with the MM algorithm of De Leeuw *et al.* (2016).

Standard usage is `alscal(delta, type, weightmat)` with `delta` being an input dissimilarity matrix (can be pre-transformed), `type` specifying the implicit transformation (one of `"ratio"` or `"interval"`) and `weightmat` being the symmetric matrix of weights $[w_{ij}]$.

For our data example, we can fit ratio ALSCAL as

```
R> als <- alscal(dis, type = "ratio")
```

The configuration is displayed in Figure 2 (corresponding code in Section 4.4).

4.4. MULTISCALE

MULTISCALE (Ramsay 1977) is a metric MDS method that was derived as a maximum likelihood model for lognormal distributed dissimilarities. This leads to explicit logarithmic transformations for input dissimilarities δ_{ij} and the fitted distances $d_{ij}(X)$. We extend this to allow for ratio and interval implicit transformations, so the associated stress is

$$\sigma_{\text{MULTISCALE}}(X) = \sum_{i<j} w_{ij} \left(\hat{\delta}_{ij} - \hat{d}_{ij}(X)\right)^2 = \sum_{i<j} w_{ij} \left(f(\log(\delta_{ij})) - \log(d_{ij}(X))\right)^2 \quad (8)$$

with $\hat{\delta}_{ij}, \hat{d}_{ij}(X)$ being the transformed dissimilarities and fitted distances respectively. The explicit transformations are $T_\Delta(\delta_{ij}) = \log(\delta_{ij})$ and $T_D(d_{ij}(X)) = \log(d_{ij}(X))$ with no explicit parameter vector[2]. The implicit transformations that are allowed are the ratio and interval transformations. We optimize this with an MM algorithm (De Leeuw *et al.* 2016).

We use this as `multiscale(delta, type, weightmat)` with `delta` being an input dissimilarity matrix (can be pre-transformed), `type` specifying the MDS type (one of `"ratio"` or `"interval"`) and `weightmat` being the symmetric matrix of weights $[w_{ij}]$. Note that we internally take the logarithm of the δ_{ij}, so one must ensure that the δ_{ij} are so that $\hat{\delta}_{ij} = \log(\delta_{ij}) \geq 0$ (as dissimilarities must be non-negative). This can often be achieved by adding 1 to the dissimilarities.

A ratio MULTISCALE MDS for our data is then

```
R> ms <- multiscale(dis, type = "ratio")
```

The configuration is displayed in Figure 2 via the following code

```
R> twobytwo <- matrix(1:4, ncol = 2, nrow = 2, byrow = TRUE)
R> olist2 <- list(sam, els, als, ms)
R> par(mai = c(0.3, 0.4, 0.4, 0))
R> alignplot(olist2, mds0$conf,
+           mains = c("interval Sammon mapping", "ratio elastic scaling",
+                     "ratio ALSCAL", "ratio MULTISCALE"),
+           layoutmat = twobytwo,
+           label.conf = list(pos = 5, col = cg),
+           xlab = "", ylab = "", asp = 1)
```

When looking at the configurations in Figure 2 we see that elastic scaling does not help with separating the items well. The interval Sammon result is pretty close to the interval MDS solution. ALSCAL and MULTISCALE separate the SCC and the CSII items into two density-connected clusters; they also single out the CSII items 1, 7, 9, 10 as separated from the other CSII items (but less separated than the SCC items).

[2]Technically, we only approximate this stress by using $T_D(d_{ij}(X)) = d_{ij}(X)^\kappa$ with $\kappa \to 0$; this is based on $\log(z) \approx a z^{\frac{1}{a}} - a$ and thus the difference $\log(\delta_{ij}) - \log(d_{ij}(X)) \approx a(\delta_{ij}^{1/a} - d_{ij}(X)^{1/a})$.

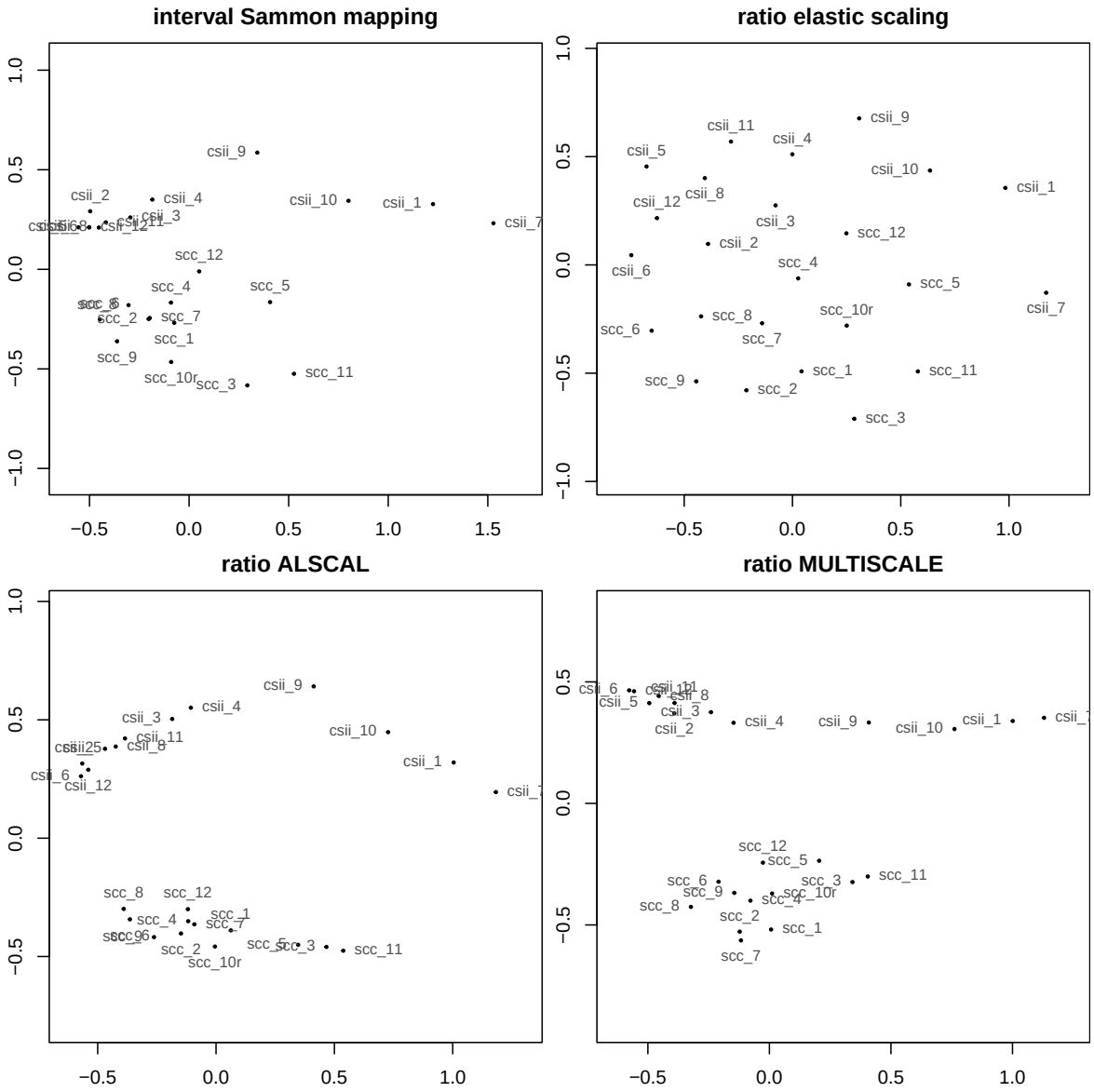

Figure 2: Configurations of interval Sammon, ratio elastic scaling, ratio ALSCAL and ratio MULTISCALE for the Koller data.

4.5. Power-Stress MDS

Power-stress MDS or POST-MDS (Buja *et al.* 2008; Groenen and De Leeuw 2010; Rusch *et al.* 2021) is a metric MDS method that allows for explicit power transformations of the input dissimilarities δ_{ij}, the fitted distances $d_{ij}(X)$ and also of given input weights w_{ij}. It uses a stress-type loss which is called power-stress or p-stress and is

$$\sigma_{\text{POST-MDS}}(X|\theta) = \sum_{i<j} \hat{w}_{ij} \left(\hat{\delta}_{ij} - \hat{d}_{ij}(X)\right)^2 = \sum_{i<j} w_{ij}^{\nu} \left(f(\delta_{ij}^{\lambda}) - d_{ij}(X)^{\kappa}\right)^2 \tag{9}$$

with $\hat{\delta}_{ij}, \hat{d}_{ij}(X), \hat{w}_{ij}$ being the transformed dissimilarities, transformed fitted distances and

transformed weights respectively. The explicit transformations are $T_\Delta(\delta_{ij}|\theta_\Delta) = \delta_{ij}^\lambda$ for input proximities, $T_D(d_{ij}(X)|\theta_D) = d_{ij}(X)^\kappa$ for configuration distances, and $T_W(w_{ij}|\theta_W) = w_{ij}^\nu$ for the weights, so any power transformation with exponents $\kappa \in \mathbb{R}_{\geq 0}$, $\lambda, \nu \in \mathbb{R}$. The explicit parameter vector θ comprises the powers in the exponents, $\theta = (\theta_\Delta, \theta_D, \theta_W)^\top = (\lambda, \kappa, \nu)^\top$. As implicit transformations f we allow the ratio and interval transformation. Setting the powers allows for many different effects, for example it covers convex power functions (for exponents > 1), root functions that are concave (for exponents $\in [0, 1]$) or inverse functions (for exponents < 0 if admissiable). We optimize this with the MM algorithm of De Leeuw *et al.* (2016).

For metric scaling this is a very flexible and versatile MDS badness-of-fit function. First, it encompasses many different metric stresses, including the standard ratio MDS stress results for $\kappa = \lambda = \nu = 1$, ALSCAL for $\kappa = \lambda = 2$, Sammon mapping for $w_{ij} = \delta_{ij}, \nu = -1$, elastic scaling for $w_{ij} = \delta_{ij}, \nu = -2$; it also approximates MULTISCALE with $\hat{\delta}_{ij} = \log(\delta_{ij}), \kappa \to 0$. It also encodes a configuration for constant dissimilarities with $\lambda = 0$. It is closely related to r-stress which is equivalent to p-stress if we use δ_{ij}^λ in r-stress with metric implicit transformations.

It is even more versatile when recognizing that we may use any manual pre-transformation of proximities together with the built-in transformation T_Δ and their θ parameters creatively in a mix-and-match approach; for example, we can create the (hitherto unexisting) "2nd order polynomial MULTISCALE elastic scaling MDS model" when using POST-MDS with $a + b\log(\delta_{ij}) + c\log(\delta_{ij})^2$ as the input proximities, $\lambda = 1, \kappa = 0.01, w_{ij} = a + b\log(\delta_{ij}) + c\log(\delta_{ij})^2, \nu = -2$. Using pre-transformation vs. using built-in transformation makes no difference for the fit of an individual MDS with given θ.

Standard usage is `postmds(delta, kappa, lambda, nu, type, weightmat)` with `delta` being an input dissimilarity matrix (can be pre-transformed), `kappa, lambda, nu` being the elements of θ and `weightmat` being the symmetric matrix of weights $[w_{ij}]$ and `type` specifying the MDS type/implicit transformation (one of `"ratio"` or `"interval"`).

We also offer a version of POST-MDS where κ and λ are restricted to be equal, which is the stress of Groenen, De Leeuw, and Mathar (1996) with power functions. The command is `rpostmds(delta, expo, nu, type, weightmat)` with `delta` being the input dissimilarity matrix (can be pre-transformed), `expo, nu` being the elements of θ (with `expo` being the $\kappa = \lambda$) and the rest as in `postmds()`.

Because POST-MDS can be hard to fit we offer a simplified version called "approximate power-stress" (ap-stress, Rusch *et al.* 2021) that approximates ratio p-stress. With using the proximities as weights, so $w_{ij} = \delta_{ij}$ and allowing a free parameter for a power transformation of the weights (υ), we can approximate p-stress (9) by

$$\sigma_{\text{apstress}}(X|\theta) = \sum_{i<j} \delta_{ij}^\upsilon \left(\delta_{ij}^\tau - d_{ij}(X)\right)^2 \tag{10}$$

with $\theta = (\tau, \upsilon)$. The connection to power-stress is so that $\upsilon = \nu + 2\lambda(1 - 1/\kappa)$ and $\tau = \lambda/\kappa$. This works well in cases when for x_i, x_j for which w_{ij} is large, the error $\delta_{ij}^\lambda - d_{ij}(X)^\kappa$ in p-stress is small, so that $d_{ij}(X)^\kappa$ is approximated reasonably well by δ_{ij}^λ and, equivalently, $d_{ij}(X)^\kappa$ is approximated well by $d_{ij}(X)\delta_{ij}^{(\lambda(\kappa-1)/\kappa)}$. Optimization of ap-stress is more straightforward than of p-stress, e.g., one can use standard SMACOF (De Leeuw 1977).

One can fit this as `apostmds(delta, kappa, lambda, nu, weightmat)` with `delta` being the input dissimilarity matrix (can be pre-transformed), `kappa, lambda, nu` being the elements of θ and `weightmat` being the symmetric matrix of weights $[w_{ij}]$ (which are currently only allowed to be binary here). This only works with implicit ratio transformations.

For our data example, fitting only ratio models this would be

```
R> pst <- postmds(dis, kappa = 0.35, lambda = 1.2, nu = -0.5, weightmat = dis)
R> rpst <- rpostmds(dis, expo = 1.5, nu = -1.5, weightmat = dis)
R> apst <- apostmds(dis, kappa = 3, lambda = 0.8, nu = -0.5)
```

The resulting configurations of all three models can be found in Figure 3 (corresponding code in Section 4.6).

4.6. R-Stress MDS

With r-stress MDS we denote an MDS method that allows for explicit power transformations of the fitted distances $d_{ij}(X)$ and any of the mentioned implicit transformations $f(\delta_{ij})$ (De Leeuw *et al.* 2016). Therefore, r-stress is

$$\sigma_{\text{rstress}}(X|\theta) = \sum_{i<j} w_{ij} \left(\hat{\delta}_{ij} - \hat{d}_{ij}(X)\right)^2 = \sum_{i<j} w_{ij} \left(f(\delta_{ij}) - d_{ij}(X)^{2r}\right)^2 \tag{11}$$

with $\hat{\delta}_{ij}, \hat{d}_{ij}(X)$ being the transformed dissimilarities and transformed fitted distances. The explicit transformation built-in is $T_D(d_{ij}(X)|\theta_D) = d_{ij}(X)^{2r}$ with $r \in \mathbb{R}_{\geq 0}$. The explicit free parameter vector is thus $\theta = r$. The ratio, interval, spline and ordinal implicit optimal scaling transformations are found internally.

This is a very flexible, versatile badness-of-fit measure that can be used in a large variety of settings and encompasses many other stresses. For $r = 0.5$ it yields standard metric or nonmetric MDS. If one inputs δ_{ij}^{λ} or w_{ij}^{ν} in r-stress and sets $r = \kappa/2$ with the ratio transformation for f, then it becomes power-stress. If in the same setup another manual transformation is used we can for example also fit interval POST-MDS, e.g. as $\hat{\delta}_{ij} = a + b\delta_{ij}^{\lambda}$. This latter property is an example of a general approach that can be used in r-stress which is to include any T_Δ and any T_W in combination with any of the implicit transformations by manually pre-transforming the δ_{ij} beforehand and using r-stress with the manually pretransformed δ_{ij}. This way one can include any type of monotonic or nonmonotonic $T_\Delta(\delta_{ij})$ in r-stress in combination with any pre-transformed weights and any implicit transformation f, thus even encompassing the versatility of POST-MDS. To illustrate this flexibility, we can create the (hitherto nonexistant) "interval MULTISCALE MDS model with Sammon weighting" if we manually set up r-stress with $\hat{\delta}_{ij} = a + b\log(\delta_{ij}), r = 0.01, w_{ij} = 1/\log(\delta_{ij})$.

One can fit it via `rstressmds(delta, r, type, weightmat)` with `delta` being an input dissimilarity matrix (can be pre-transformed), `type` specifying the optimal scaling (one of `"ratio", "interval", "mspline", "ordinal"`), `r` being the element of θ, `weightmat` being the symmetric matrix of weights $[w_{ij}]$.

For our data example, this may be an ordinal r-stress model with a convex relation of dissimilarities and distances

```
R> rst <- rstressmds(dis, r = 0.6, type = "ordinal")
```

The resulting configuration can be created with the following code and found in Figure 3.

```
R> olist3 <- list(pst, rpst, apst, rst)
R> par(mai = c(0.3, 0.4, 0.4, 0))
R> alignplot(olist3, mds0$conf,
+           mains = c("ratio POST-MDS", "ratio restricted POST-MDS",
+                     "ratio approx. POST-MDS", "ordinal r-stress MDS"),
+           layoutmat = twobytwo,
+           label.conf = list(pos = 5, col = cg), xlab = "", ylab = "",
+           asp = 1)
```

When looking at the configurations in Figure 3 we see that the POST-MDS gives a similar configuration than the ordinal MDS perhaps emphasizing the outlier nature of some of the CSII items more strongly. The restricted POST-MDS separates the items into three groups. This is also the case for the ordinal r-stress MDS which seems to be able to combine the results of the ordinal MDS with the POST-MDS, thus allowing to separate the CSII 1, 7, 9, 10 items from the other CSII items and the CSII items from the SCC items. Conversely, the approximative POST-MDS seems to nullify the separation and scales the items in such a way that there is less variability in the distances.

4.7. Local MDS

Local MDS (LMDS, Chen and Buja 2009) is a stress-based metric scaling MDS method for manifold learning that aims at preserving the local neighbourhood around a point. Let N_k define the symmetric set of nearby pairs of points (i, j) so that $(i, j) \in N_k$ if i is among the $k-$nearest neighbours of j or the other way round. The stress function is

$$\sigma_{\text{lmds}}(X|\theta) = \sum_{(i,j)\in N_k} (\delta_{ij} - d_{ij}(X))^2 + \sum_{(i,j)\notin N_k} u\,(\delta_\infty - d_{ij}(X))^2\,. \tag{12}$$

where $\delta_\infty \to \infty$ is a large "imputed" dissimilarity that is constant and u a small weight. See Rusch *et al.* (2023a) for an expression in terms of stress with explicit transformations. The objective (12) can be simplified by taking $u \approx 1/\delta_\infty$ and expanded to the standard LMDS objective of Chen and Buja (2009). In the latter, LMDS uses the tuning parameter $\tau = 2u\delta_\infty$ for given k, so we have the explicit transformation parameter vector $\theta = (k, \tau)^\top$. We optimize this with gradient descent.

The standard usage in R is to use the function `lmds(delta, k, tau)` and for our running data example we may use `k = 5` and `tau = 0.5`, so

```
R> lms <- lmds(dis, k = 5, tau = 0.5)
```

The resulting configuration can be found in Figure 4 (corresponding code in Section 4.9).

Note that per default LMDS configurations are not normalized the same way as the models optimized via MM; if this is not wanted, one can set `normconf = TRUE` to change that (but then the fitted `lms$confdist` no longer correspond to the manually calculated `dist(lms$conf)`).

4.8. Box-Cox MDS

Chen and Buja (2013) propose power transformations on observed proximities and Box-Cox transformations on fitted distances in an energy badness-of-fit formulation. For complete

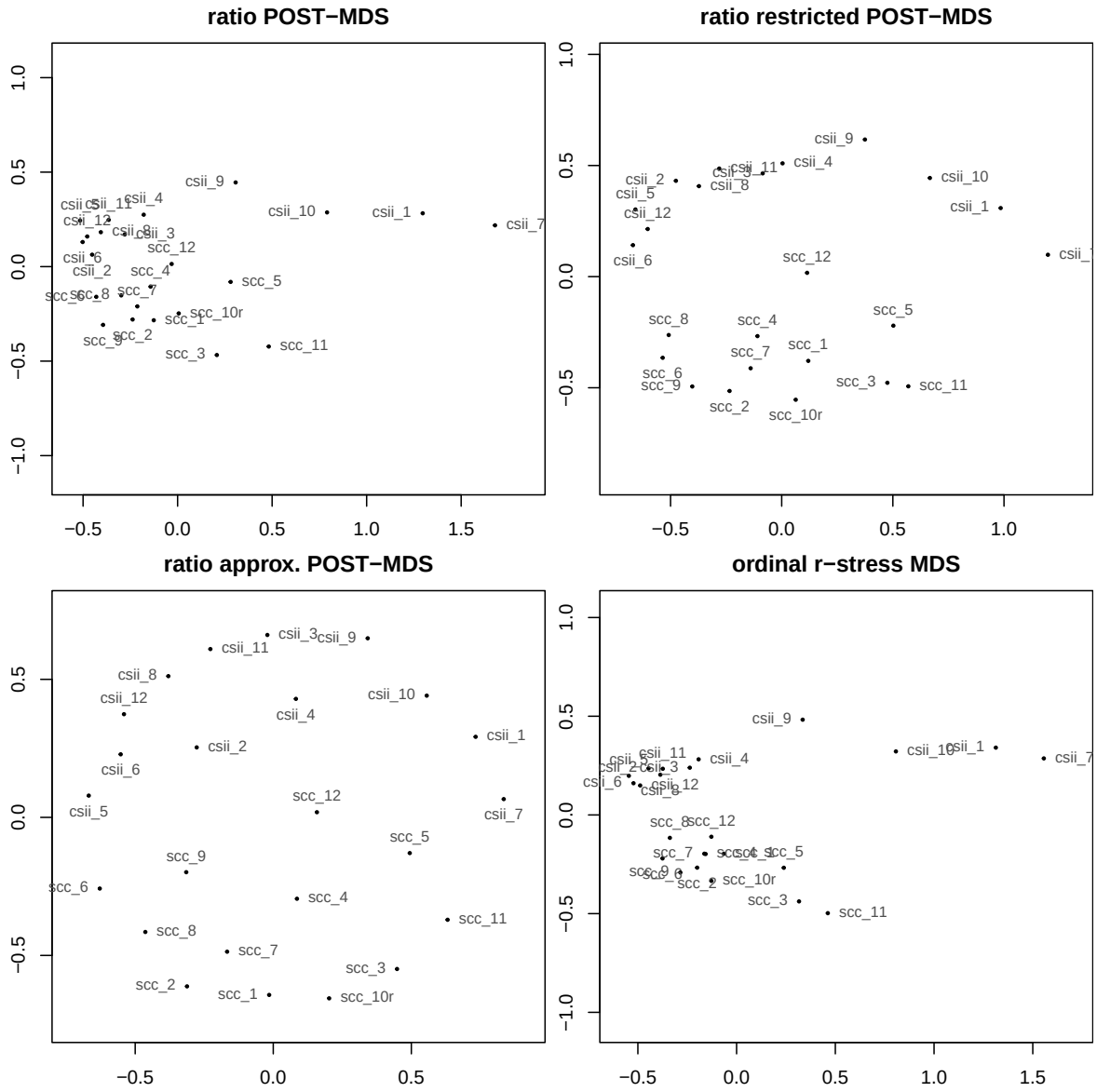

Figure 3: Configurations of ratio POST-MDS, ratio restricted POST-MDS, ratio approx. POST-MDS and ordinal r-stress MDS for the Koller data.

data matrices this yields a three-parameter energy-type MDS family which we coin Box-Cox MDS (BC-MDS). The badness-of-fit function has explicit transformation parameter vector $\theta = (\mu, \lambda, \rho)^\top$ with $\mu, \rho \in \mathbb{R}$ and $\lambda \in \mathbb{R}_+$ and is

$$\sigma_{\text{bcmds}}(X|\theta) = \sum_{i<j} \delta_{ij}^{\rho} \left(BC_{\mu+\lambda}(d_{ij}(X)) - \delta_{ij}^{\lambda} BC_{\mu}(d_{ij}(X)) \right) \tag{13}$$

The transformation BC_α is the one-parameter Box-Cox transformation (Box and Cox 1964) with parameter α,

$$BC_\alpha(d) = \begin{cases} \frac{d^\alpha - 1}{\alpha} \text{ if } \alpha \neq 0 \\ \log(d) \text{ if } \alpha = 0 \end{cases} \tag{14}$$

Note that the explicit transformations used for the distances need not be equal in the attraction and repulsion parts. We optimize this with gradient descent.

The BC-MDS model can be fit via `bcmds(delta, mu, lambda, rho)`. For example for the Koller data, we may use it as

```
R> bcm <- bcmds(dis, mu = 3, lambda = 0.7, rho = -1.5)
```

The resulting configuration can be found in Figure 4 (corresponding code in Section 4.9).

As with LMDS, Box-Cox MDS configurations are not normalized, but one can set `normconf = TRUE` to change that with the side effect that the fitted `bcm$confdist` no longer correspond to the manually calculated `dist(bcm$conf)`.

4.9. Curvilinear component analysis and curvilinear distance analysis

Demartines and Herault (1997) suggest a version of MDS for manifold learning called curvilinear component analysis (CLCA) that gets a local flavour by putting emphasis on smaller fitted configuration distances which is achieved by setting $w_{ij} = 0$ if $d_{ij}(X) > \tau$. This was proposed for ratio MDS and no explicit transformations, thus

$$\sigma_{\text{clca}}(X|\theta) = \sum_{i<j} \left(\hat{\delta}_{ij} - d_{ij}(X)\right)^2 \mathbb{1}\left(d_{ij}(X) \leq \tau\right) = \sum_{i<j} \left(\delta_{ij} - d_{ij}(X)\right)^2 \mathbb{1}\left(d_{ij}(X) \leq \tau\right) \quad (15)$$

with $\theta = \tau$ and $\mathbb{1}$ denoting the indicator function. This stress is optimized via stochastic gradient descent with a positive number of epochs. It was envisioned by the original authors that the τ changes in every epoch in a decreasing sequence starting with τ_0, similar to the self-organizing map (SOM, Kohonen 1982).

If the matrix $\hat{\Delta}$ are geodesic distances as in Isomap, CLCA is called curvilinear distance analysis (CLDA, Lee, Lendasse, and Verleysen 2004).

We include wrapper functions `clca()` and `clda()` in **smacofx** that allows to fit CLCA and CLDA in the way described above. They use `CCA()` from **ProjectionBasedClustering** (Thrun and Ultsch 2021) as their workhorse and return an object of class '`smacofP`'. The usage is one of either `clca(delta, lambda0, alpha0)` or `clda(delta, lambda0, alpha0, k, epsilon)` respectively, with `delta` being an input dissimilarity matrix (can be pre-transformed), `lambda0` being the τ_0, `alpha0` the stepsize of the stochastic gradient descent. In CLDA we use `vegan::isomapdist()` so we have one additional explicit transformation parameter, either `k` or `epsilon` as the neighbourhood parameter for the geodesic distance.

For the running data example, we may use

```
R> set.seed(1)
R> clcares <- clca(dis, lambda0 = 1, alpha0 = 1)
R> cldares <- clda(dis, lambda0 = 1, alpha0 = 1, k = 5)
```

The resulting configurations can be found in Figure 4, obtained by the following code.

```
R> olist4 <- list(lms, bcm, clcares, cldares)
R> par(mai = c(0.43, 0.4, 0.4, 0))
R> alignplot(olist4, mds0$conf,
```

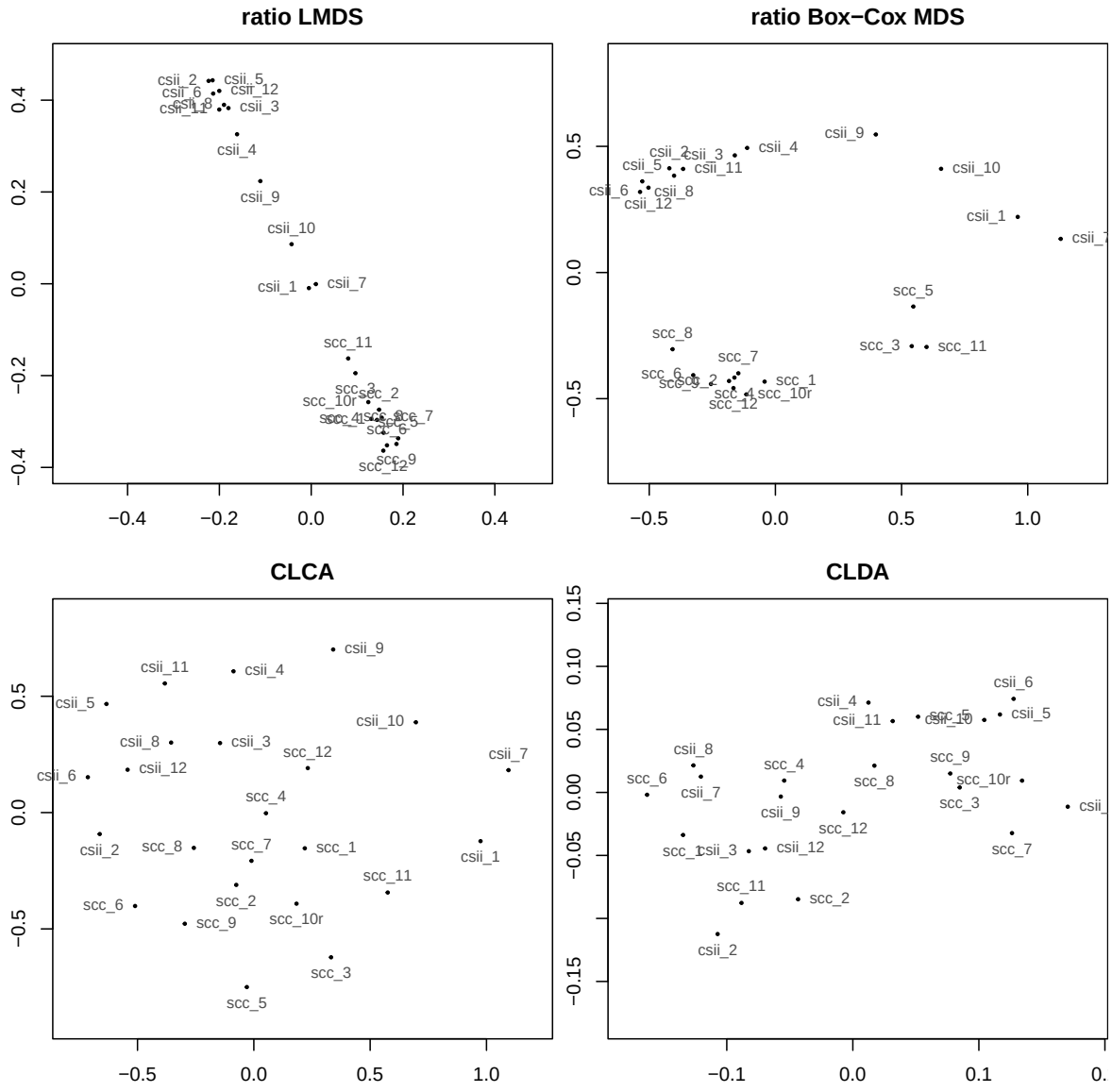

Figure 4: Configurations of ratio LMDS, ratio Box-Cox MDS, CLCA and CLDA for the Koller data.

```
+           mains = c("ratio LMDS", "ratio Box-Cox MDS",
+                     "CLCA", "CLDA"),
+           layoutmat = twobytwo,
+           label.conf = list(pos = 5, col = cg), xlab = "", ylab = "",
+           asp = 1)
```

When looking at the configurations in Figure 4 we see that CLCA and CLDA do not allow separation of the items based on the scales. They arrange items in a relatively regular way and represent the items without obvious crowding; these two methods may be less illuminating for scale items. Local MDS as a method for emphasizing local structure arranges the items

almost linearly in the target space but at least reproduces the cluster nature of the scales we saw before, with SCC items comprising one and the CSII items comprising a second group with CSII 1,7,9,10 bridging the two. The ratio BC-MDS result is similar to the restricted POST-MDS result.

4.10. Extended curvilinear (power) component and distance analysis

Rusch (2025) proposed a generalization of CLCA, coined extended curvilinear power component analysis (eCLPCA). It also uses weighting with the Heaviside function $\mathbb{1}\left(\hat{d}_{ij}(X) \leq \tau\right)$ but additionally allows for given static weights, implicit ratio, interval and spline transformations of dissimiliarities and explicit power transformations of dissimilarities, fitted distances or weights. It essentially marries metric r-stress MDS or POST-MDS to CLCA, thus allowing for manifold learning/local version of r-stress MDS and all its special cases (from Sammon mapping to POST-MDS), and adds flexibility to CLCA.

The stress function that is minimized is:

$$\begin{aligned}\sigma_{\text{eclpca}}(X|\theta) &= \sum_{i<j} \hat{w}_{ij} \left(\hat{\delta}_{ij} - \hat{d}_{ij}(X)\right)^2 \mathbb{1}\left(\hat{d}_{ij}(X) \leq \tau\right) \\ &= \sum_{i<j} w_{ij}^{\nu} \left(f(\delta_{ij}^{\lambda}) - d_{ij}(X)^{\kappa}\right)^2 \mathbb{1}\left(d_{ij}(X)^{\kappa} \leq \tau\right) \end{aligned} \tag{16}$$

with $\theta = (\lambda, \kappa, \nu, \tau)^{\top}$. The Heaviside step function sparsifies the weight matrix as if one were to set $w_{ij} = 0$ if $\hat{d}_{ij}(X) > \tau$, so this may also be also called sparsified (power) MDS. If $\lambda = \kappa = \nu = 1$ and f is the implicit ratio transformation this is extended curvilinear component analysis (eCLCA) with a single, fixed τ. We optimize this with the Quasi-MM algorithm of Rusch (2025). The two versions can be can be fitted with `eclpca(delta, lambda, kappa, tau, nu, type, weightmat)` or `eclca(delta, tau, type, weightmat)` respectively, with `delta` being an input dissimilarity matrix (can be pre-transformed), `lambda, kappa, nu, tau` being the elements of the respective θ and `weightmat` being a symmetric matrix of weights $[w_{ij}]$.

For the Koller data, we may fit an interval eCLCA with $\tau = 0.2$ and a eCLPCA with $\kappa = 3$, $\lambda = 1.5$, $\tau = 0.025$ as

```
R> smdsres <- eclca(dis, tau = 0.2, type = "interval")
R> spmdsres <- eclpca(dis, kappa = 3, lambda = 1.5, tau = 0.025)
```

The configurations are displayed in the top row of Figure 5 (corresponding code below).

If the δ_{ij} used are geodesic distances the methods are called extended curvilinear power distance analysis (eCLPDA) and extended curvilinear distance analysis respectively (eCLDA). We have implemented convenience functions to fit eCLPDA and eCLDA models that use `vegan::isomapdist()` internally. This adds a fifth hyperparameter, either k or ϵ, for the neighbourhood. The usage is one of `eclpda(delta, k, lambda, kappa, tau, nu, type, weightmat)` or `eclda(delta, k, tau, type, weightmat)` for the k version, or `eclpda(delta, eps, lambda, kappa, tau, nu, type, weightmat)` or one can use `eclda(delta, eps, tau, type, weightmat)` for the ϵ version.

For our data example and using the `k` argument

```
R> sddares <- eclda(dis, k = 5, tau = 0.3)
R> spmddares <- eclpda(dis, k = 5, tau = 0.05,
+                      kappa = 2, lambda = 1.5, nu = -1.5,
+                      type = "interval")
```

The configurations are displayed in the bottom row of Figure 5 from this code.

```
R> olist5 <- list(smdsres, spmdsres, sddares, spmddares)
R> par(mai = c(0.3, 0.4, 0.4, 0))
R> alignplot(olist5, mds0$conf,
+           mains = c("interval eCLCA", "ratio eCLPCA",
+                     "ratio eCLDA", "interval eCLPDA"),
+           layoutmat = twobytwo,
+           label.conf = list(pos = 5, col = cg), xlab = "", ylab = "",
+           asp = 1)
```

When looking at the configurations in Figure 5 we see that interval eCLCA indeed looks similar to the interval MDS but with an emphasis on a more regular arrangement of local neighbourhoods. The three clusters are separated. The ratio eCLPCA result is emphasizing the clusters even more, giving an arrangement that is reminiscent of the ordinal r-stress MDS with a more cohesive SCC cluster, again the effect of the sparsification/locality feature. eCLDA and eCLPDA also use geodesic distances and look similar in structure to the local MDS result (especially interval eCLPDA and LMDS look similar). They allow to visually separate the SCC items from the CSII items and arrange the local neighbourhood of the CSII items fairly well (the configuration can be cut with three horizontal line to yield the three clusters); CLDA in its original form was not able to do that, showing that the extended methods can add value.

In the eCLPA functions above we hold τ fixed. In Demartines and Herault (1997) it has been proposed to vary τ over a decreasing sequence of τs as in a SOM (and is also done in our CLCA implementation). We adapted this to eCL(P)CA and eCL(P)DA as described in Rusch (2025), making them into self-organizing variants. The idea is that one starts with a $\tau_s = \tau_0$ and then repeats these steps: fit the model with τ_s, use a new $\tau_t < \tau_s$, set $\tau_s = \tau_t$ fit the model with current τ_s. This gradually refines the solution for increasingly smaller distances and with every pass through the data, called `epochs`, the configuration gets re-arranged based on an increasingly narrower subset of distances. We support this in **smacofx** with functions that prefix `so_`, so

```
R> so_eclpca(delta, lambda, kappa, tau, nu, type, epochs, weightmat)
R> so_eclca(delta, tau, type, epochs, weightmat)
R> so_eclpda(delta, k, epsilon, lambda, kappa, tau, nu, type, epochs,
+           weightmat)
R> so_eclda(delta, k, epsilon, tau, type, epochs, weightmat)
```

The arguments are like in `eclpca()` or `eclpda()`, with the exception of `tau` which now is either a numeric vector containing a sequence of decreasing τ supplied by the user, or if given as a scalar the highest τ (τ_0) of the τ sequence. In case of the latter, a sequence of length

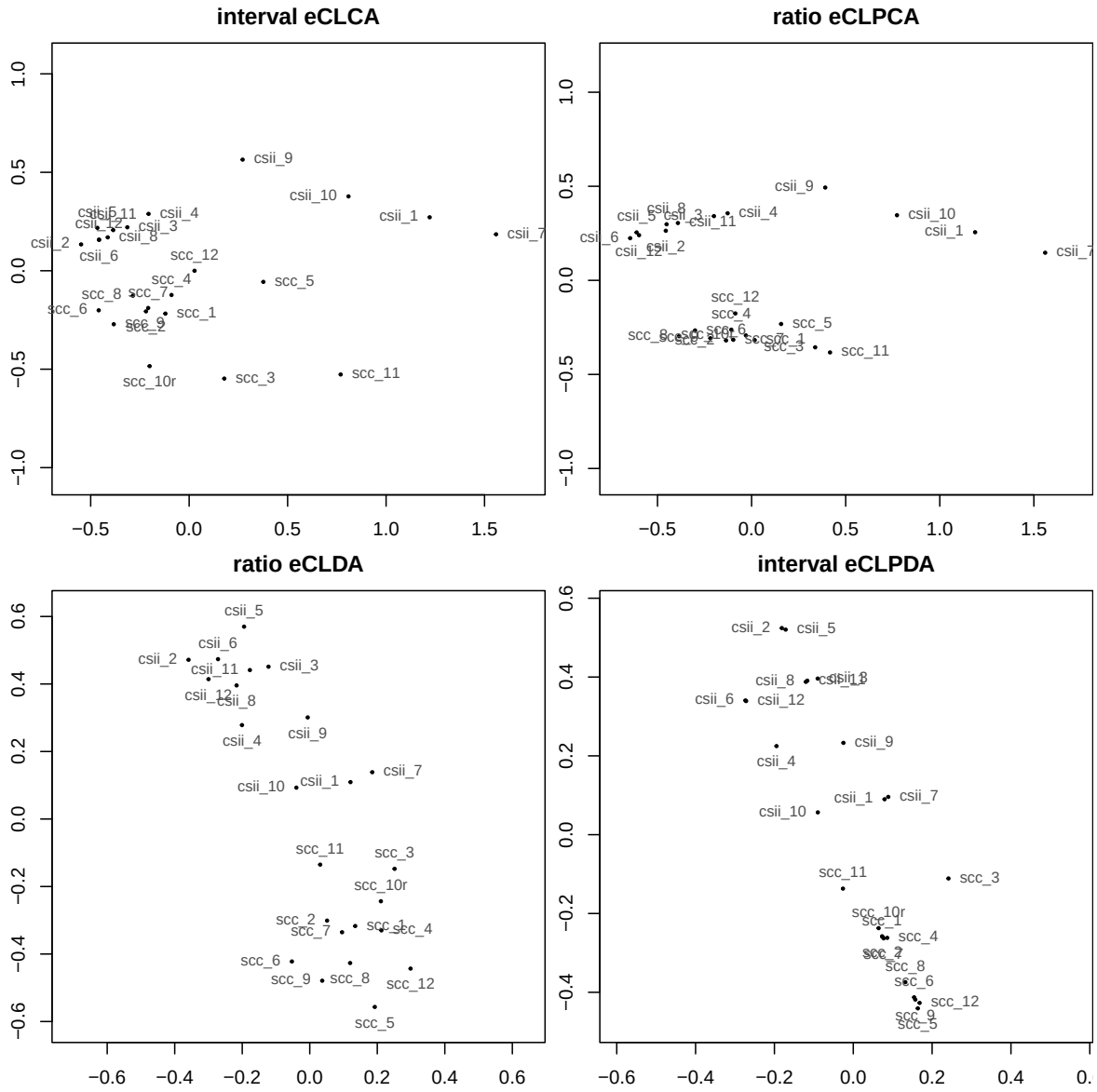

Figure 5: Configurations of interval eCLCA, ratio eCLPCA, ratio eCLDA and interval eCLPDA for the Koller data.

`epochs` is created that runs from `tau` to `tau / epochs`. If `tau` is a sequence, the `epochs` argument is ignored.

4.11. Wrappers for external MDS functions

Our package also feature wrappers for MDS related functions from other packages to make them usable within the **smacofx** framework. Currently we have wrappers for `cmdscale()` in **base** which does Torgerson scaling and the Sammon mapping implementation `Sammon()` in **MASS**.

5. Post-fit infrastructure

Following in the spirit of **smacof** we offer the most important post-fitting infrastructure that **smacof** offers also for objects returned by the functions in **smacofx**. Specifically, we support most of the **smacof** object plots including the configuration plot, bubble plot and Shepard plot. We also support MDS biplots (`biplotmds()`), MDS jackknife (`jackmds()`), MDS bootstrap (`bootmds()`), permutation tests (`permtest()`), included a multistart function (`multistart()`), and a function for the exploration of the effect of initial configurations (`icExploreGen()`). We explain them in turn.

5.1. Plotting

Plot method. The standard `plot()` method for objects of class '`smacofP`' supports the plots listed subsequently. They can be selected with the `plot.type` argument to `plot()`. The plots are the "configuration plot" (default, plots the objects in the configuration with or without convex hulls), the "residual plot" (`plot.type = "resplot"`, a plot of the $\hat{\delta}_{ij}$ vs. the $\hat{d}_{ij}(X)$), the "stress decomposition plot" (`plot.type = "stressplot"`, which plots the badness-of-fit contribution of each observation and the higher the contribution, the worse the fit), the "bubble plot" (`plot.type = "bubbleplot"`, which is a configuration plot with the point size proportional to the badness-of-fit contribution), and a histogram of the $\hat{\delta}_{ij}$ weighted with $\hat{w}_{ij}$ (`plot.type = "histogram"`). These plots correspond directly to the plots available in **smacof** for objects of class '`smacofB`' (see De Leeuw and Mair 2009; Mair *et al.* 2022, for details).

Objects of class '`smacofP`' support a tailored form of the Shepard diagram (`plot.type = "Shepard"`) that takes possible power transformations into account. There are two versions of this Shepard diagram: With the additional argument `shepard.lin = TRUE`, it is a diagram with the transformed observed normalized dissimilarities $T_\Delta(\delta_{ij})$ on the abscissa and the transformed fitted distance $T_D(d_{ij}(X))$ on the ordinate, plus a regression line corresponding to MDS type chosen in the fitting procedure (a linear one without intercept for `"ratio"`, a linear one for `"interval"`, a montone spline for `"mspline"` and an isotonic one for `"ordinal"`), as well as a loess curve. The loess curve can help in gauging how well the power transformation works by checking the congruence of the loess smoother with the regression line; it can be turned off with `loess = FALSE`. We call this "linearized Shepard diagram". If `shepard.lin = FALSE` it is a Shepard plot that uses the untransformed δ_{ij} on the abscissa instead, as well as the fitted power transformation regression line and loess smoother.

A unique plot for models with power transformations is the "transformation plot" (`plot.type = "transplot"`). This is a 2D plot with the ordinate showing the normalized observed dissimilarities (δ_{ij}, light grey) and the normalized explicitly transformed dissimilarities ($T_\Delta(\delta_{ij})$, darker) against the abscissa of untransformed fitted distances ($d_{ij}(X)$) together with the parametric regression curve corresponding to the explicit transformation used by the MDS (so, the fitted power transformation). This plot is most useful for ratio models with power transformations as the transformations can be read off directly. For other MDS models, it still gives a quick way to assess how the explicit transformations worked. If there are no power transformations this would simplify to a residual plot.

We show the latter two plots for the power stress solution of the running data example in

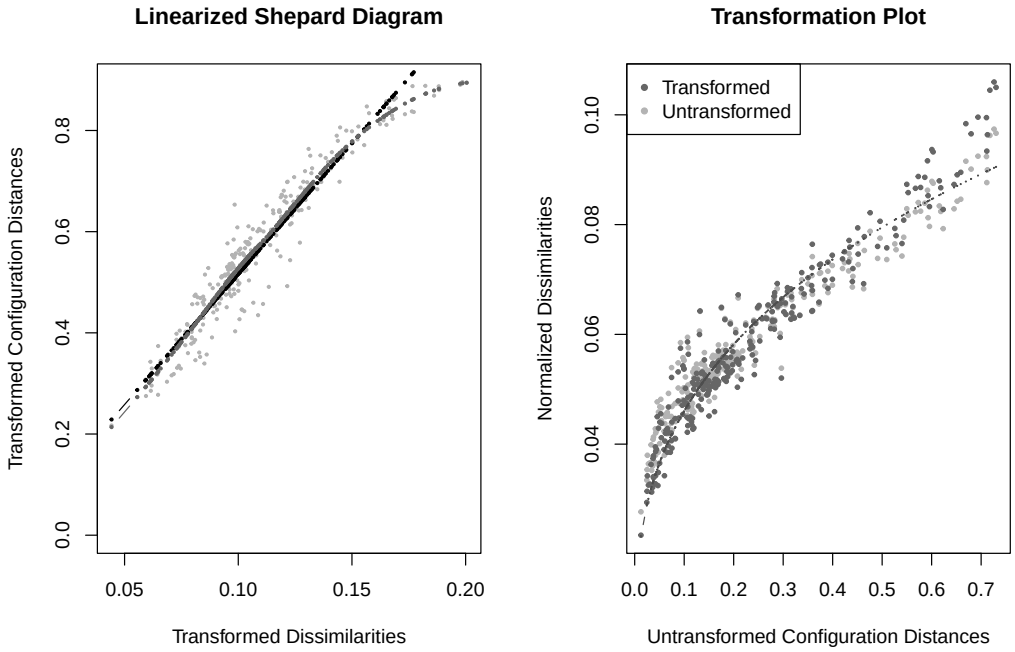

Figure 6: Linearized Shepard plot (left) and transformation plot (right) for '`smacofP`' objects.

Figure 6.

```
R> op <- par(no.readonly = TRUE)
R> par(mfrow = c(1, 2))
R> plot(pst, plot.type = "Shepard")
R> plot(pst, plot.type = "transplot")
R> par(op)
```

We see that the chosen parametrization for the POST-MDS leads to a rather good correspondence of disparities and transformed distances, visible by the regression line and the loess smoother coinciding rather well (with a correlation of 0.992). Only in the region of high $\hat{\delta}_{ij}$ the two curves diverge.

Biplots. We support MDS biplots for all the MDS model classes described previously. These biplots map one or more external variables into the same space where the MDS configuration lies. This can help in interpreting directions in the MDS space, which is often more meaningful then interpreting the axes of the MDS solution. Let's say we have q centered external variables (perhaps also standardized) that comprise the columns of a $n \times q$ matrix Y. The MDS biplot uses the OLS estimates of the $p \times q$ regression coefficient matrix B obtained from the multivariate linear regression $Y = XB + E$ as the coordinates of the external variables in the MDS space. There are two representations that can be chosen, the "vector representation" and the "axis representation". More details can be found in Mair *et al.* (2022). See also the example in Section 6.

5.2. Uncertainty quantification

The **smacofx** packages features methods for the uncertainty quantification functionality generics in **smacof**, i.e., for the MDS jackknife, the MDS bootstrap and a permutation test.

Jackknife. We implemented the MDS jackknife method of De Leeuw and Meulman (1986) for objects returned by the functions in **smacofx**. It extends the MDS jackknife in **smacof** to work with the flexible MDS methods we described, as well as returns values for the cross-validity, stability and dispersion statistic. See Mair *et al.* (2022) for a thorough discussion.

The usage is straightforward object-oriented, e.g., for the eCLDA

```
R> jk1 <- jackmds(sddares)
R> plot(jk1)
```

See the worked example in Section 6.

The jackknife can also give hints to whether the specific MDS model or optimization at hand is prone to local minima or premature convergence. This can be signalled by some objects being placed very far away from their centroid for some jackknife resamples, often discernable by long rays in the jackknife MDS plot. This then warrants further investigation (but note that not seeing this does not mean there is no problem).

Bootstrap. We also have implemented a bootstrap method `bootmds()` that works for the objects returned by functions in **smacofx**. This is the same bootstrap as used in **smacof**. Note that the bootstrap currently only supports a few distance functions and the original data set needs to be supplied, see Mair *et al.* (2022) for details.

The usage is

```
R> bs1 <- bootmds(sddares, data = koller, method.dat = "euclidean")
R> plot(bs1)
```

and a worked example is given in Section 6.

Permutation test. Mair, Borg, and Rusch (2016) propose a permutation test for testing against whether the configuration may have been obtained from a permuted dissimilarity matrix. If this null is not rejected, any differentiation information that the MDS model used to find the configuration is consistent with idiosyncratic noise. Note that just because we reject the null in this test does not mean we actually established that the MDS result is fitting well—this test only provides some protection against falsely interpreting a MDS result obtained from an uninformative underlying dissimilarity matrix as substantive.

The test can be conducted for '`smacofP`' objects with

```
R> permtest(sddares)
```

See Section 6 for an example.

5.3. Exploring and mitigating local minima

MDS models are notorious for being difficult to optimize, mainly due to most MDS objectives having multiple local optima when p is low. This problem can be even more pronounced for flexible MDS models. The **smacofx** package features functions that can help with exploring and mitigating the problem of local optima with the `jackmds()` (as described previously), `icExploreGen()` and `multistart()` functions.

Exploration of starting configurations. The `icExploreGen()` function in **smacofx** allows to explore how different (random) starting configurations influence the final result. In it, we run a given MDS model from different initial starting configurations up to convergence. The MDS configurations obtained for each different start are then matched via Procrustes analysis. Then the intercorrelations of the point coordinates between any two obtained configurations are calculated and an interval MDS is fitted based on the intercorrelations. By default, the starting configurations are generated randomly from a p-dimensional uniform distribution with given minimum and maximum vectors (defaulting to a p-dimensional vector of -5 and 5 respectively).

The implementation in **smacofx** is a bit more general than `smacof::icExplore()`, as the former is object-oriented and can accommodate various MDS models including those from **smacof**. It can be initialized via a pre-fitted model object, but also setup via a '`call`' object. Additionally, one can supply a list of starting configurations. In the plot method the 2D interval MDS of the configuration similarities is displayed, with the number being the index of the corresponding MDS configuration and the size reflecting the badness-of-fit value: the larger the font, the worse the fit. The size is also associated with a corresponding color shading (the smaller the size the darker the color). See Section 6 for a worked example.

Multiple starts. A way of mitigating the issue of local optima in MDS is to use different starting configurations, letting the algorithm run until convergence from every start and then select the configuration with the overall lowest badness-of-fit value. This is the multistart method (Borg and Mair 2017) and is implemented in the `multistart()` function. It is an object-oriented function and only needs a pre-fitted model object. By default the `multistart()` function creates `nstart` starting configurations from a p-dimensional uniform distribution (with all elements of the minimum vector of -5 and of the maximum vector of 5). One can also supply a list of starting configurations. By default, the function returns the MDS object that had the lowest overall stress over all the configurations obtained.

The usage is

```
R> ms1 <- multistart(bcm)
```

with `ms1$best` giving the best configuration. See Section 6 for a worked version.

If the starting configurations are drawn from a uniform distribution, one can make a probabilistic statement about the obtained fit value being within a specific quantile around the global optimum. This follows from the well-known property of random search that if we evaluate an objective at o random points, we can calculate the probability P that at least one objective value lies within the Q%-quantile band around the global optimum. This is because the probability that none of the o values lies within the Q-th quantile is $(1-Q)^o$ and

that at least one lies within is then $1 - (1 - Q)^o$. From this we can derive that the number of evaluations needed to lie with at least probability P in the Q-th quantile band above the global minimum is $o \geq \frac{\log(1-P)}{\log(1-Q)}$. So for example, if we want that with probability $P = 0.95$ the fit measure lies at least once within the $Q \times 100\% = 1\%$ band of the global optimum, we need to evaluate the badness-of-fit at $o \geq 299$ randomly selected configurations. Since we typically run more than one iteration this is a strict upper bound. In `multistart()`, the default o (`nstart`) we use is 110, which puts us with probability of at least 2/3 inside 1% of the global minimum and with probability of at least 0.996 inside the 5% lowest badness-of-fit values.

6. Worked example

For a fully worked example and tour through the functionality in **smacofx**, we will scale images of "corpse paint". Corpse paint is a type of (usually) black-and-white facial make-up employed by musicians in hard rock, heavy metal, and there especially in Black Metal, meant to shock and/or make the wearer appear and feel "one with darkness", corpse-like, inhuman or demonic (e.g., Phillipov 2012; Venkatesh, Podoshen, Urbaniak, and Wallin 2015). We point out that this is no laughing matter and should only be worn by the devout and certainly not by posers.

We use 32 images of corpse paint that have been downloaded from the internet after a Google image search. We processed these 32 images to a resolution of 90×90 pixel and as 8-bit gray scale images (so allowing for 256 shades of gray) taking intensity values between 0 and 1. The data are available as `corpsepaint` in the package **smacofx**.

```
R> data("corpsepaint")
```

Each column vector represents an image, and each vector element is a pixel, so the vectors are of length $90 \times 90 = 8100$. The images can be plotted with the following code and are displayed in Figure 7.

```
R> par(mfrow = c(8, 4), mai = c(0, 0.01, 0.2, 0.01))
R> for(i in 1:ncol(corpsepaint)) {
+ p1 <- matrix(corpsepaint[, i], ncol = 90, nrow = 90,
+       byrow = FALSE)
+ image(p1, col = gray.colors(256), main =
+       colnames(corpsepaint)[i], axes = FALSE)
+ }
R> par(op)
```

To measure dissimilarity between the images, we use the L_1-distance (Manhattan or city-block distance) between the image vectors based on the corresponding pixels. For two image vectors x_i and x_j this is,

$$d(x_i, x_j) = \sum_{k=1}^{8100} |x_{ik} - x_{jk}|. \tag{17}$$

with index k running over the pixels. This distance is appropriate for images (Gonzalez and Woods 2002) and allows us to illustrate the full functionality of the package.

Figure 7: 32 images of corpse paint. The images are 90×90 pixels in 8-bit gray scale.

```
R> disc <- as.matrix(dist(t(corpsepaint), method = "manhattan"))
```

We fit 15 different MDS models fitted to the data: ratio MDS as reference, CLCA with $\tau_0 = 1$, ratio MULTISCALE, ratio ALSCAL, local MDS with $k = 5, \tau = 0.5$, ratio elastic scaling, interval Sammon mapping, ordinal r-stress MDS with $r = 0.75$, ratio Box-Cox MDS with $\mu = 3, \lambda = 0.7, \rho = -1.5$, restricted ratio POST-MDS with $w_{ij} = \delta_{ij}, \kappa = \lambda = 3, \nu = -1.5$, ratio POST-MDS with $w_{ij} = \delta_{ij}, \kappa = 0.3, \lambda = 0.8, \nu = -0.5$, interval eCLCA with $\tau = 0.15$, ratio eCLDA with $k = 5, \tau = 0.1$, ratio eCLPCA with $\kappa = 2.5, \lambda = 1.5, \tau = 0.05$ and interval eCLPDA with $k = 5, \tau = 0.3, \kappa = 3, \lambda = 1.5, \nu = -1.5$. We set zealous `itmax = 50000` and `acc = 1e-8` for all methods.

```
R> set.seed(0208)
R> mds0c <- mds(disc, eps = 1e-8)
R> clcaresc <- clca(disc, lambda0 = 1, alpha0 = 1)
R> msc <- multiscale(disc, acc = 1e-8)
R> alsc <- alscal(disc, acc = 1e-8, itmax = 50000)
R> lmsc <- lmds(disc, k = 5, tau = 0.5, acc = 1e-8,
+             itmax = 50000)
R> elsc <- elscal(disc, acc = 1e-8, itmax = 50000)
R> samc <- sammonmap(disc, type = "interval", acc = 1e-8,
+                    itmax = 50000)
R> rstc <- rstressMin(disc, r = 0.75, type = "ordinal",
+                     acc = 1e-8, itmax = 50000)
R> bcmc <- bcmds(disc, mu = 3, lambda = 0.7, rho = -1.5,
+              acc = 1e-8, itmax = 50000)
R> rpstc <- rpostmds(disc, expo = 3, nu = -1.5, weightmat = disc,
+                    acc = 1e-8, itmax = 50000)
R> pstc <- postmds(disc, kappa = 0.3, lambda = 0.8, nu = -0.5,
+                  weightmat = disc, acc = 1e-8, itmax = 50000)
R> smdsresc <- eCLCA(disc, tau = 0.15, type = "interval", acc = 1e-8,
+                    itmax = 50000)
R> sddaresc <- eCLDA(disc, k = 5, tau = 0.1, acc = 1e-8,
+                    itmax = 50000)
R> spmdsresc <- eCLPCA(disc, kappa = 2.5, lambda = 1.5, tau = 0.05,
+                      itmax = 50000, acc = 1e-8)
R> spmddaresc <- eCLPDA(disc, k = 5, tau = 0.3, kappa = 3, lambda = 1.5,
+                       nu = -1.5, type = "interval", itmax = 50000,
+                       acc = 1e-8)
```

The subsequent code produces Figure 8 which gives the 2D configurations of the fitted MDS variants in Western reading order.

```
R> lt <- matrix(1:15, nrow = 5, ncol = 3)
R> cplist <- list(mds0c, clcaresc, msc, alsc, lmsc, elsc, samc,
+                 rstc, bcmc, rpstc, pstc, smdsresc, sddaresc,
+                 spmdsresc, spmddaresc)
R> mains <- c("ratio MDS", "CLCA", "ratio MULTISCALE", "ratio ALSCAL",
```

```
+            "ratio LMDS", "ratio elastic scaling",
+            "interval Sammon mapping",
+            "ordinal r-stress MDS", "ratio Box-Cox MDS",
+            "ratio restricted POST-MDS", "ratio POST-MDS",
+            "interval eCLCA", "ratio eCLDA", "ratio eCLPCA",
+            "interval eCLPDA")
R> par(mai = c(0.27, 0.27, 0.4, 0))
R> alignplot(cplist, mds0c$conf, mains = mains, layoutmat = lt,
+           label.conf = list(pos = 5, col = cg), xlab = "", ylab = "",
+           asp = 1)
```

We first notice that the configurations can look quite different in parts, from little discernable structure in the ratio MDS to structured results like for Box-Cox MDS. This is to be expected due to the different implicit and explicit transformations utilized and warrants our chosen term "flexible".

Going forward we use only a subset for further illustration, namely the ordinal r-stress, the ratio POST-MDS, the interval eCLPDA and the ratio Box-Cox MDS. The print output of the models are listed subsequently; it provides information on the model type, explicit transformation parameters, number of objects, badness-of-fit value (square root of stress) and the number of iterations.

```
R> rstc

Call:
rstressMin(delta = disc, r = 0.75, type = "ordinal", acc = 1e-08,
    itmax = 50000)

Model: ordinal r-stress MDS with parameter vector= 0.75
Number of objects: 32
Stress-1 value: 0.257
Number of iterations: 39584

R> pstc

Call:
postmds(delta = disc, kappa = 0.3, lambda = 0.8, nu = -0.5, weightmat = disc,
    acc = 1e-08, itmax = 50000)

Model: ratio power-stress MDS with parameter vector= 0.3 0.8 -0.5
Number of objects: 32
Stress-1 value: 0.087
Number of iterations: 16410

R> spmddaresc

Call:
eCLPDA(delta = disc, lambda = 1.5, kappa = 3, nu = -1.5, tau = 0.3,
```

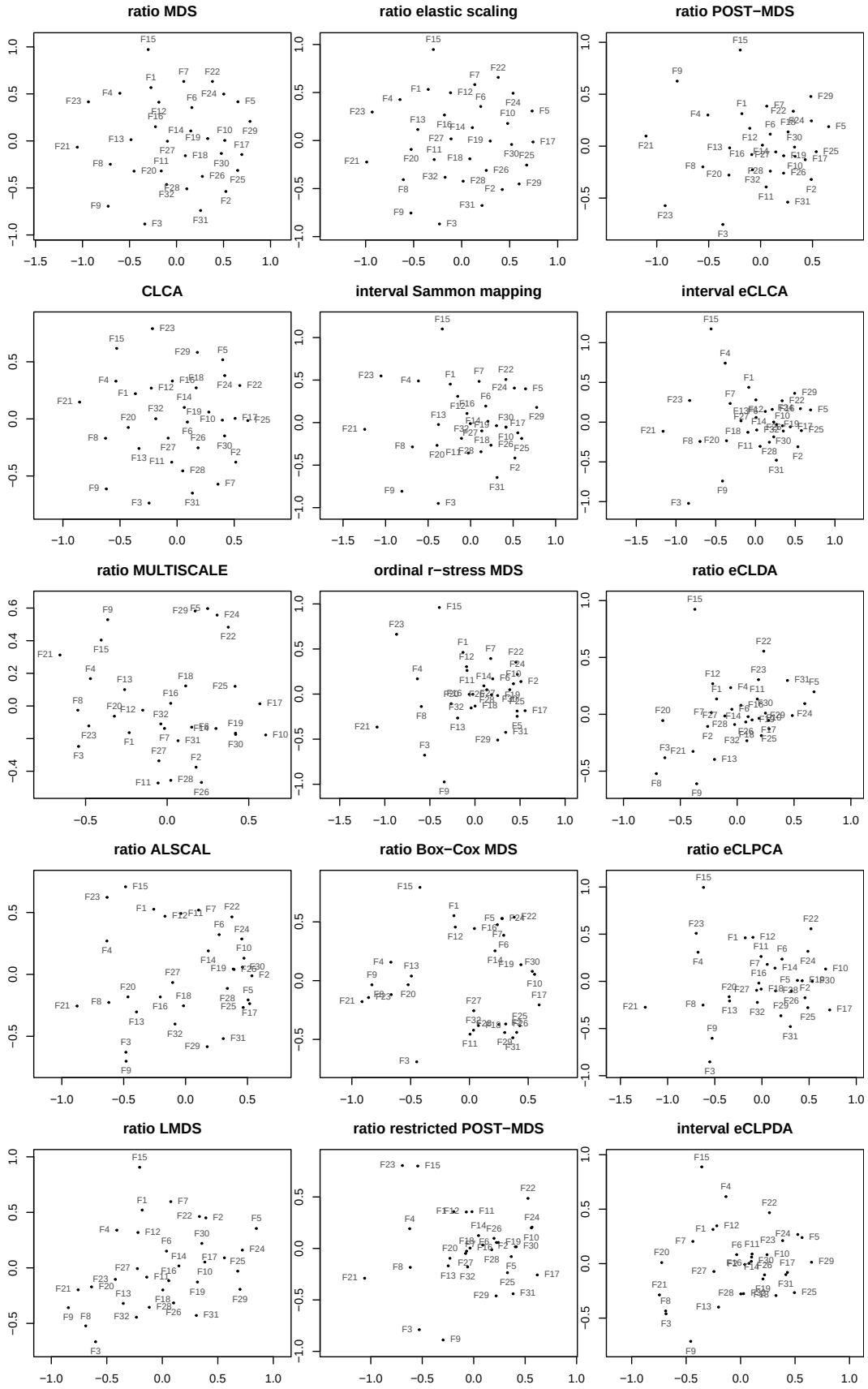

Figure 8: 15 different MDS configurations of the corpse paint data set (Procrustes adjusted to the ratio MDS).

```
    type = "interval", k = 5, acc = 1e-08, itmax = 50000)

Model: interval eCLPDA with parameter vector= 3 1.5 -1.5 0.3 5
Number of objects: 32
Stress-1 value: 0.51
Number of iterations: 50000


R> bcmc


Call:
bcmds(delta = disc, mu = 3, lambda = 0.7, rho = -1.5, itmax = 50000,
    acc = 1e-08)

Model: ratio Box-Cox MDS with parameter vector= 3 0.7 -1.5
Number of objects: 32
Stress-1 value: 0.505
Number of iterations: 520
```

We are first interested in looking at the configuration with the images plotted. For this we defined a function `plot_cpmds()` that allows us to plot the image into the configuration.

```
R> library("TeachingDemos")
R> plot_cpmds <- function(mdsc, inches = 0.3, main = "MDS", xlab = "",
+                         ylab = "")
+     {
+     plot(mdsc, type = "n", main = main, asp = 1, xlab = xlab, ylab = ylab)
+     for(i in 1:dim(mdsc)[1])
+     {
+         r <- array(matrix(rev(corpsepaint[, i]), 90, 90, byrow = TRUE),
+                    c(90, 90, 3))
+         my.symbols(mdsc[i, 1], mdsc[i, 2], symb = ms.image, MoreArgs =
+                    list(img = r), inches = inches, symb.plots = TRUE,
+                    add = TRUE)
+      }
+    }
```

We use Procrustes adjusted configuration (adjusted to the POST-MDS) configuration to make them align visually.

```
R> pstc2 <- pstc$conf
R> bcmc2 <- Procrustes(pstc$conf, bcmc$conf)$Yhat
R> rstc2 <- Procrustes(pstc$conf, rstc$conf)$Yhat
R> spmddaresc2 <- Procrustes(pstc$conf, spmddaresc$conf)$Yhat
```

The four configurations of the 32 images are displayed in Figure 9.

```
R> par(mfrow = c(2, 2), mai = c(0.3, 0.3, 0.4, 0))
R> plot_cpmds(pstc2, inches = 0.3, main = "ratio POST-MDS")
R> plot_cpmds(rstc2, inches = 0.3, main = "ordinal r-Stress MDS")
R> plot_cpmds(spmddaresc2, inches = 0.3, main = "interval eCLPDA")
R> plot_cpmds(bcmc2, inches = 0.3, main = "ratio Box-Cox MDS")
R> par(op)
```

Luckily the great panda is generally scaled towards the edge. For the arrangement in the MDS configuration it seems quite important that some images feature high levels of middling greyness and others show a strong black and white contrast. Dimension 1 (abscissa) apparently captures this distinction quite well (negative values on Dimension 1 seem to be associated with higher greyness, positive values with stronger contrasts). The ordinate is harder to interpret; perhaps the images placed in the upper half look less human and more animalistic.

The eCLPDA cuts off a substantial number of proximities, so we check with a permutation test whether the result can be distinguished from a situation with uninformative dissimilarities, which would mean that the explicit transformations in eCLPDA (so, the τ and the geodesic distance transformation) have the effect of nullifying the information content in the L_1-distance matrix.

```
R> set.seed(666)
R> pte <- permtest(spmddaresc, nrep = 100)

R> pte

Call: permtest.smacofP(object = spmddaresc, nrep = 100)

SMACOF Permutation Test
Number of objects: 32
Number of replications (permutations): 100

Observed stress value: 0.51
p-value: 0.02
```

We reject the null at the 5% level ($p = 0.02$) pointing towards this configuration indeed being consistent with an informative mapping (the τ does not nullify the information content after the Heaviside function was applied, see `spmddares$tweightmat`).

We now calculate a jackknife stability for the POST-MDS and a bootstrap for the ratio Box-Cox MDS and display the plots in Figure 10.

```
R> jk <- jackmds(pstc)

R> jk

Call: jackmds.smacofP(object = pstc)
```

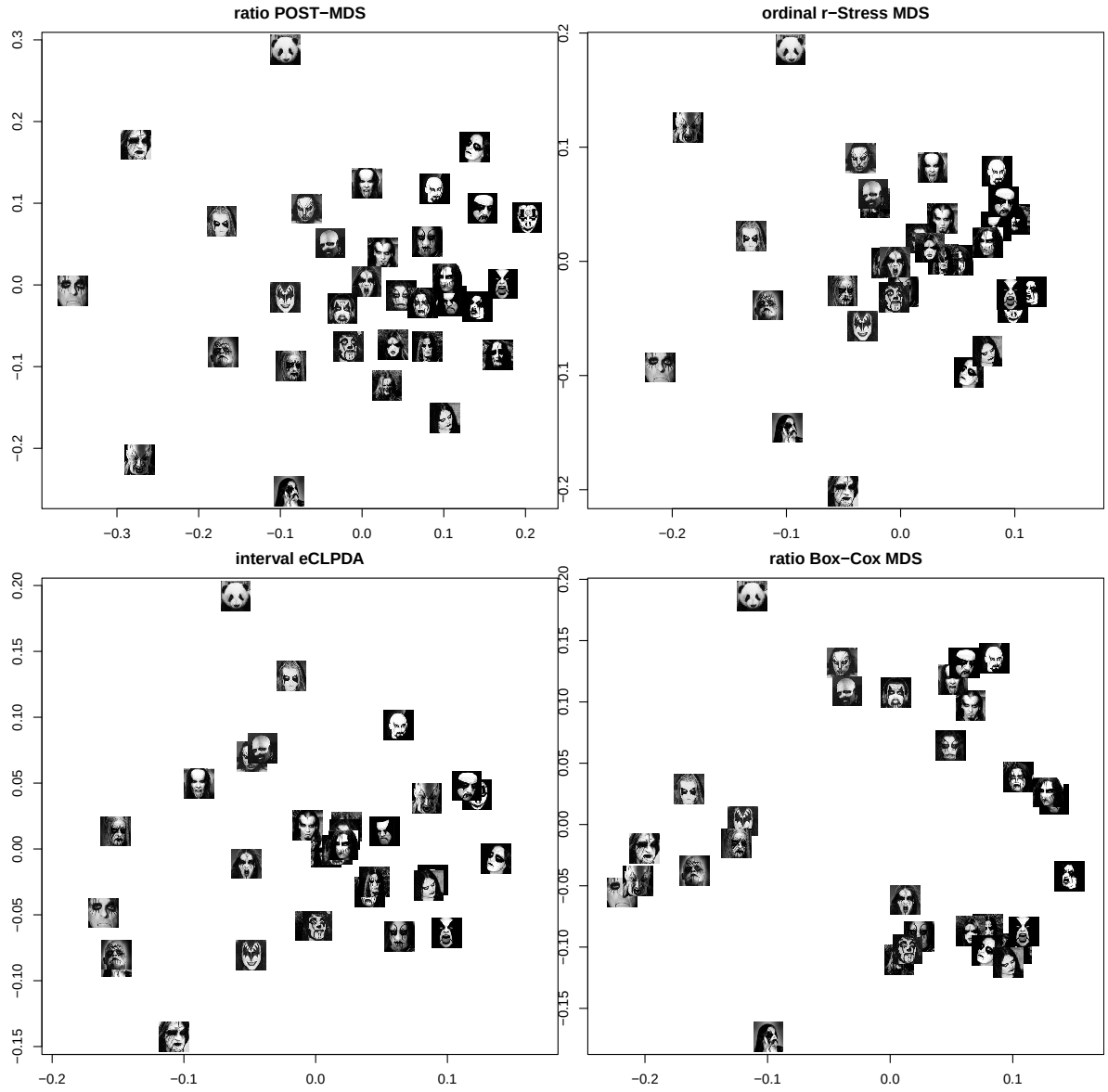

Figure 9: Four configurations with the 32 images overplotted (after Procrustes adjustment to the ratio POST-MDS result).

```
SMACOF Jackknife
Number of objects: 32
Value loss function: 0.4361
Number of iterations: 3

Stability measure: 0.991
Cross validity: 0.9987
Dispersion: 0.0103

R> bs <- bootmds(bcmc, corpsepaint, "manhattan")

R> bs

Call: bootmds.smacofP(object = bcmc, data = corpsepaint, method.dat =
  "manhattan")

SMACOF Bootstrap:
Number of objects: 32
Number of replications: 100

Mean bootstrap stress:  0.5007
Stress percentile CI:
  2.5%  97.5%
0.4959 0.5070

Stability coefficient: 0.7201

R> par(mfrow = c(1, 2))
R> plot(jk)
R> plot(bs)
R> par(op)
```

We see that the jackknife stability of the POST-MDS is relatively high, whereas the bootstrap for the Box-Cox MDS shows large uncertainty ellipses for many points. This is likely due to the difference in transformation parameters: The POST-MDS parameters are close to the ratio MDS and the differences in dissimilarities between runs does not change much. In the Box-Cox MDS, however, we used rather high parameter values which emphasize the dissimilarity differences very much, and nonlinearly at that. We also map to a space that is quite curved ($\mu = 3$) and then look at it as if it were Euclidean. These two effects together make the spatial arrangement quite variable for a few of the points.

We saw interpretation of the axes is not straightforward in the selected examples. An MDS biplot can help by adding an external covariate into the configuration plot. To that end, we had five Black Metal experts rate the corpse paint images on a "kvltness" scale from 1 to 7, with 7 meaning "trve kvlt". We averaged the ratings over the raters and scaled the rating into the MDS result with an MDS biplot, in this case for the ordinal r-stress MDS result. Note that the scaling itself did not take the rating into account.

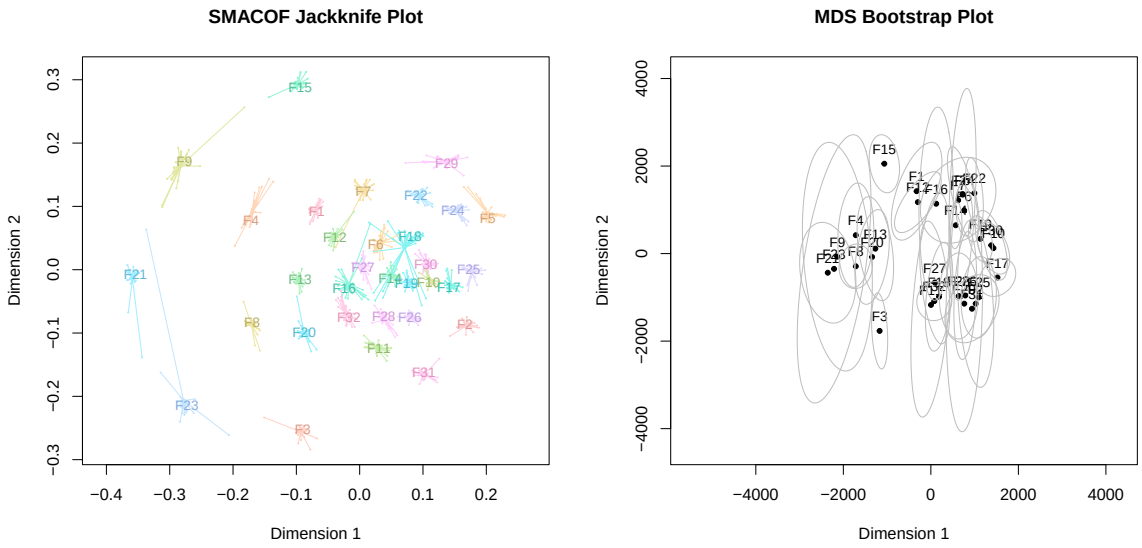

Figure 10: MDS jackknife plot (left) for the ratio POST-MDS and MDS bootstrap plot with 95%-confidence ellipses (right) for the Box-Cox MDS.

```
R> prt.mean <- data.frame(kvltness = c(4.0, 4.8, 5.8, 5.8, 2.6, 3.0, 3.6,
+                                      3.2, 4.6, 5.4, 4.6, 1.2, 1.6, 3.2, 4.0,
+                                      3.4, 6.0, 2.8, 2.0, 5.8, 2.2, 2.4, 1.8,
+                                      3.2, 5.8, 5.4, 4.8, 4.6, 3.4, 2.0, 5.4,
+                                      0.4),
+                          row.names = paste0("F", 1:32))
R> bi1 <- biplotmds(rstc, extvar = prt.mean)
```

In the left panel of Figure 11 we see that the external variable points towards positive x axis values/negative y axis values corresponding to most kvlt. The kvltness direction is quite highly correlated to the y axis (Dimension 2). In this direction of more kvltness lie famous Norwegian Black Metal musicians, e.g. Euronymous of Mayhem (F31), Abbath (F25) of Immortal, Shagrath of Dimmu Borgir (F26), Nocturno Culto (F28) and Fenriz (F29) of Darkthrone. On the opposite side lie the panda bear (F15), F1, F12 (untypical example) or F23, musicians of Behemoth (F6, F7) but also the originator of Black Metal's use of this make-up, Dead (F4). Images almost perpendicular to the kvltness axis show Kiss's Demon (F13), Alice Cooper (F21), over-the-top (F5, F9) and untypical examples (F8, F22). It also seems that the raters have a tendency to rate the high-contrast images as more kvlt.

When looking at the restricted POST-MDS (with $\kappa = \lambda = 3, \nu = -1.5, w_{ij} = \delta_{ij}$) and the Box-Cox MDS result ($\mu = 3, \lambda = 0.7, \rho = -1.5$), we see that the configuration is unexpectedly different although they have similar explicit transformations and hyperparameters. When using the Box-Cox MDS configuration as the initial start to the restricted POST-MDS and running just one iteration, we see that the stress with the Box-Cox MDS result as the start is lower compared to the restricted POST-MDS that started from the Torgerson solution and was run until convergence.

```
R> rpst2 <- rpostmds(disc, expo = 3, nu = -1.5, weightmat = disc,
+                    init = bcmc$conf, itmax = 1)
R> round(c("Torgerson Start" = rpstc$stress, "BC-MDS Start" = rpst2$stress),
+       3)

Torgerson Start    BC-MDS Start
          0.590           0.502
```

This suggests that the restricted POST-MDS starting from the Torgerson solution was trapped in a local minimum. We can explore this more systematically with the `icExploreGen()` function. We generate a list of 47 random starting configurations manually, and then append them to the Box-Cox MDS configuration and the restricted POST-MDS configuration which started from the Torgerson solution. The latter two are on positions 1 and 2 of the list respectively. We then fit the restricted POST-MDS again with each of these 49 starting configurations.

```
R> set.seed(666)
R> cl1 <- replicate(47, matrix(runif(2 * 32, min = -5, max = 5), ncol = 2),
+                   simplify = FALSE)
R> cl1 <- c(list(bcmc$conf), list(rpstc$conf), cl1)
R> ic1 <- icExploreGen(rpstc, conflist = cl1, verbose = FALSE)

R> par(mfrow = c(1, 2))
R> plot(bi1, vecscale = 0.2, label.conf = list(pos = 5))
R> plot(ic1, cex.scale = 3)
R> par(op)
```

In the right panel of Figure 11 we see that the configuration 1 (the one we obtained from using the Box-Cox MDS configuration as the starting configuration) has the lowest stress (smallest size) and that configuration 2 (the orginal restricted power stress solution from a Torgerson start) is closest to it and has fairly low stress too.

For the restricted POST-MDS we also run `multistart()` with 110 random configurations and hope we improve upon the "best" configuration we found so far.

```
R> set.seed(666)
R> rpst3 <- rpostmds(disc, expo = 3, nu = -1.5, weightmat = disc,
+                    init = bcmc$conf)
R> m2 <- multistart(rpst3,verbose=FALSE)

R> m2$best

Call:
rpostmds(delta = disc, expo = 3, nu = -1.5, weightmat = disc,
    init = bcmc$conf)

Model: ratio power-stress MDS with parameter vector= 3 3 -1.5
Number of objects: 32
Stress-1 value: 0.502
Number of iterations: 1
```

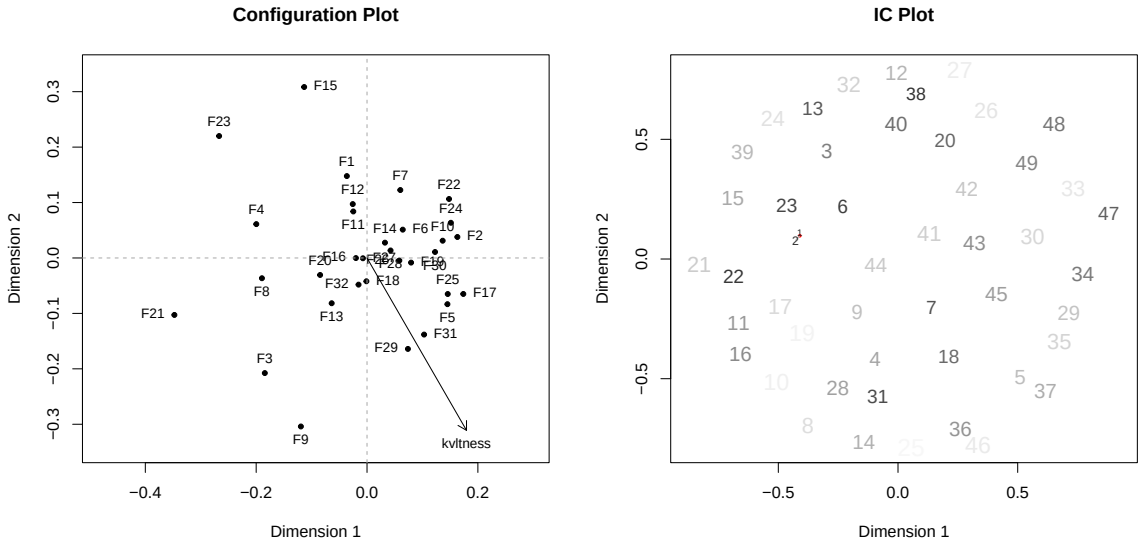

Figure 11: Left panel: MDS biplot for the ordinal r-stress MDS configuration and the average "kvltness" rating. Right panel: Interval MDS configuration of solutions starting from different configurations. This is the plot for the result of `icExploreGen()`.

We see that the best stress we found is the one from starting with the Box-Cox MDS solution, so we can say that with at least a probability of 0.996, it is within 5% of the global optimum.

7. Conclusion

We presented the package **smacofx** which features flexible multidimensional scaling methods and extensions to the package **smacof**. We illustrated various functions for fitting, plotting and displaying a large number of different flexible MDS models including Sammon mapping with ratio and interval optimal scaling, MULTISCALE with ratio and interval optimal scaling, ALSCAL with ratio and interval optimal scaling, elastic scaling with ratio and interval optimal scaling, r-stress MDS with ratio, interval, spline and non-metric optimal scaling, POST-MDS with ratio and interval optimal scaling, restricted POST-MDS with ratio and interval optimal scaling, approximate POST-MDS with ratio optimal scaling, Box-Cox MDS, local MDS, curvilinear component analysis and curvilinear distance analysis, extended curvilinear component analysis and extended curvilinear distance analysis with or without powers. These MDS variants and the accompanying functions are highly flexible and also allow for other sensible combination of explicit power transformations for weights, distances and input proximities with implicit ratio, interval, spline or non-metric optimal scaling of the input proximities. Most functions use a variation of the MM principle for optimization. Currently the methods are only available for two-way data (symmetric dissimilarity matrices). Together with the packages **smacof** this package comprises the most comprehensive multidimensional scaling suite that we are aware of.

For future research we plan to improve individual aspects of the software such as support for parallelization for jackknife, bootstrap, and the functions to explore starting configurations.

We also plan to strengthen the object-oriented approach, for example allowing the MDS bootstrap to work with more distance functions. While we already feature a large number of MDS models in this package, there still are developments that have not found their way into R as of yet and would fit very well into the package. Extending this to unfolding and three-way data is also planned.

References

Bearden WO, Netemeyer RG, Teel JE (1989). "Measurement of Consumer Susceptibility to Interpersonal Influence." *Journal of Consumer Research*, **15**(4), 473–481. doi:10.1086/209186.

Beaton D, Abdi H (2022). ***DistatisR****: DiSTATIS Three Way Metric Multidimensional Scaling*. R package version 1.1.1, URL https://CRAN.R-project.org/package=DistatisR.

Beaton D, Fatt CRC, Abdi H (2014). "An ExPosition of Multivariate Analysis with the Singular Value Decomposition in R." *Computational Statistics & Data Analysis*, **72**(0), 176 – 189. doi:10.1016/j.csda.2013.11.006.

Borg I, Groenen P (2005). *Modern Multidimensional Scaling: Theory and Applications*. 2nd edition. Springer-Verlag, New York.

Borg I, Mair P (2017). "The Choice of Initial Configurations in Multidimensional Scaling: Local Minima, Fit, and Interpretability." *Austrian Journal of Statistics*, **46**(2), 19–32. doi:10.17713/ajs.v46i2.561.

Box GE, Cox DR (1964). "An Analysis of Transformations." *Journal of the Royal Statistical Society B*, **26**(2), 211–243. doi:10.1111/j.2517-6161.1964.tb00553.x.

Buja A, Swayne DF, Littman ML, Dean N, Hofmann H, Chen L (2008). "Data Visualization with Multidimensional Scaling." *Journal of Computational and Graphical Statistics*, **17**(2), 444–472. doi:10.1198/106186008X318440.

Campbell JD, Trapnell PD, Heine SJ, Katz IM, Lavallee LF, Lehman DR (1996). "Self-Concept Clarity: Measurement, Personality Correlates, and Cultural Boundaries." *Journal of Personality and Social Psychology*, **70**(1), 141.

Carroll JD, Chang JJ (1970). "Analysis of Individual Differences in Multidimensional Scaling via an N-way Generalization of "Eckart-Young" Decomposition." *Psychometrika*, **35**(3), 283–319. doi:10.1007/BF02310791.

Chen L, Buja A (2009). "Local Multidimensional Scaling for Nonlinear Dimension Reduction, Graph Drawing, and Proximity Analysis." *Journal of the American Statistical Association*, **104**(485), 209–219. doi:10.1198/jasa.2009.0111.

Chen L, Buja A (2013). "Stress Functions for Nonlinear Dimension Reduction, Proximity Analysis, and Graph Drawing." *Journal of Machine Learning Research*, **14**, 1145–1173. URL https://jmlr.org/papers/v14/chen13a.html.

Coombs CH (1950). "Psychological Scaling without a Unit of Measurement." *Psychological Review*, **57**(3), 145.

Cox TF, Cox MA (2001). *Multidimensional Scaling*. CRC Press, Boca Raton, FL.

Curry HB (1944). "The Method of Steepest Descent for Non-linear Minimization Problems." *Quarterly of Applied Mathematics*, **2**(3), 258–261. doi:10.1090/qam/10667.

De Leeuw J (1977). "Applications of Convex Analysis to Multidimensional Scaling." In JR Barra, F Brodeau, G Romier, BV Cutsem (eds.), *Recent Developments in Statistics*, pp. 133–145. North Holland Publishing Company, Amsterdam. URL https://escholarship.org/uc/item/4ps3b5mj.

De Leeuw J, Groenen P, Mair P (2016). "Minimizing rStress Using Majorization." *Technical report*, UCLA Statistics Preprint Series. URL https://jansweb.netlify.app/publication/deleeuw-groenen-mair-e-16-a/deleeuw-groenen-mair-e-16-a.pdf.

De Leeuw J, Heiser WJ (1980). "Multidimensional scaling with restrictions on the configuration." *Multivariate Analysis*, **5**(1), 501–522.

De Leeuw J, Mair P (2009). "Multidimensional Scaling Using Majorization: SMACOF in R." *Journal of Statistical Software*, **31**(3), 1–30. doi:10.18637/jss.v031.i03.

De Leeuw J, Meulman J (1986). "A Special Jackknife for Multidimensional Scaling." *Journal of Classification*, **3**, 97–112. doi:10.1007/BF01896814.

Demartines P, Herault J (1997). "Curvilinear Component Analysis: A Self-Organizing Neural Network for Nonlinear Mapping of Data Sets." *IEEE Transactions on Neural Networks*, **8**(1), 148–154. doi:10.1109/72.554199.

Gonzalez R, Woods R (2002). *Digital Image Processing*. 2nd edition. Prentice & Hall, New Jersey.

Goslee SC, Urban DL (2007). "The **ecodist** Package for Dissimilarity-Based Analysis of Ecological Data." *Journal of Statistical Software*, **22**, 1–19. doi:10.18637/jss.v022.i07.

Groenen P, De Leeuw J (2010). "Power-Stress for Multidimensional Scaling." *Technical report*, UCLA, Los Angeles, USA. URL https://jansweb.netlify.app/publication/groenen-deleeuw-u-10/groenen-deleeuw-u-10.pdf.

Groenen P, De Leeuw J, Mathar R (1996). "Least Squares Multidimensional Scaling with Transformed Distances." In W Gaul, D Pfeifer (eds.), *From Data to Knowledge: Theoretical Perspectives and Practical Aspects of Classification*, pp. 177–185. Springer-Verlag, Berlin. doi:10.1007/978-3-642-79999-0_17.

Groenen P, Mathar R, Heiser W (1995). "The Majorization Approach to Multidimensional Scaling for Minkowski Distances." *Journal of Classification*, **12**, 3–19. doi:10.1007/BF01202265.

Knoblauch K, Maloney LT (2008). "**MLDS** : Maximum Likelihood Difference Scaling in R." *Journal of Statistical Software*, **25**, 1–26. doi:10.18637/jss.v025.i02.

Kohonen T (1982). "Self-Organized Formation of Topologically Correct Feature Maps." *Biological Cybernetics*, **43**, 59–69. doi:10.1007/BF00337288.

Koller M, Floh A, Zauner A, Rusch T (2013). "Persuasibility and the Self – Investigating Heterogeneity among Consumers." *Australasian Marketing Journal*, **21**(2), 94–104.

Kruskal JB (1964). "Multidimensional Scaling by Optimizing Goodness of Fit to a Nonmetric Hypothesis." *Psychometrika*, **29**(1), 1–27. doi:10.1007/BF02289565.

Lange K (2016). *MM Optimization Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA. doi:10.1137/1.9781611974409.

Lee JA, Lendasse A, Verleysen M (2004). "Nonlinear Projection with Curvilinear Distances: Isomap versus Curvilinear Distance Analysis." *Neurocomputing*, **57**, 49–76. doi:10.1016/j.neucom.2004.01.007.

Mair P, Borg I, Rusch T (2016). "Goodness-of-Fit Assessment in Multidimensional Scaling and Unfolding." *Multivariate Behavioral Research*, **51**, 772–789.

Mair P, Groenen P, De Leeuw J (2022). "More on Multidimensional Scaling and Unfolding in R: **smacof** Version 2." *Journal of Statistical Software*, **102**(10), 1–47. doi:10.18637/jss.v102.i10.

McGee VE (1966). "The Multidimensional Analysis of 'Elastic' Distances." *British Journal of Mathematical and Statistical Psychology*, **19**(2), 181–196. doi:10.1111/j.2044-8317.1966.tb00367.x.

Meyer D, Buchta C (2022). ***proxy**: Distance and Similarity Measures*. R package version 0.4-27, URL https://CRAN.R-project.org/package=proxy.

Noack A (2007). "Energy Models for Graph Clustering." *Journal of Graph Algorithms and Applications*, **11**(2), 453–480.

Oksanen J, Simpson GL, Blanchet FG, Kindt R, Legendre P, Minchin PR, O'Hara R, Solymos P, Stevens MHH, Szoecs E, Wagner H, Barbour M, Bedward M, Bolker B, Borcard D, Carvalho G, Chirico M, De Caceres M, Durand S, Evangelista HBA, FitzJohn R, Friendly M, Furneaux B, Hannigan G, Hill MO, Lahti L, McGlinn D, Ouellette MH, Ribeiro Cunha E, Smith T, Stier A, Ter Braak CJ, Weedon J (2022). ***vegan**: Community Ecology Package*. R package version 2.6-4, URL https://CRAN.R-project.org/package=vegan.

Phillipov M (2012). "Extreme Music for Extreme People? Norwegian Black Metal and Transcendent Violence." *Popular Music History*, **6**(1-2), 150–163. doi:10.1558/pomh.v6i1.150.

R Core Team (2025). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Ramsay JO (1977). "Maximum Likelihood Estimation in Multidimensional Scaling." *Psychometrika*, **42**(2), 241–266. doi:10.1007/BF02294052.

Ramsay JO (1988). "Monotone Regression Splines in Action." *Statistical Science*, **3**(4), 425–441. URL http://www.jstor.org/stable/2245395.

Roberts DW (2023). ***labdsv**: Ordination and Multivariate Analysis for Ecology*. R package version 2.1-0, URL https://CRAN.R-project.org/package=labdsv.

Rusch T (2025). "Multidimensional Scaling with Heaviside Weighting: Extensions to Curvilinear Component Analysis and Curvilinear Distance Analysis." *Stat*, **14**(3), e70086. doi:10.1002/sta4.70086.

Rusch T, De Leeuw J, Chen L, Mair P (2025). ***smacofx**: Flexible Multidimensional Scaling and 'smacof' Extensions*. R package version 1.21-1, URL https://CRAN.R-project.org/package=smacofx.

Rusch T, Mair P (2024). ***cops**: Cluster Optimized Proximity Scaling*. R package version 1.12-1, URL https://CRAN.R-project.org/package=cops.

Rusch T, Mair P, Hornik K (2021). "Cluster Optimized Proximity Scaling." *Journal of Computational and Graphical Statistics*, **30**(4), 1156–1167. doi:10.1080/10618600.2020.1869027.

Rusch T, Mair P, Hornik K (2023a). "Structure-Based Hyperparameter Selection with Bayesian Optimization in Multidimensional Scaling." *Statistics and Computing*, **33**(28), 1–18. doi:10.1007/s11222-022-10197-w.

Rusch T, Venturo-Conerly K, Baja G, Mair P (2023b). "COPS in Action: Exploring Structure in the Usage of the Youth Psychotherapy MATCH." *Psych*, **5**(2), 274–302. doi:10.3390/psych5020020.

Sammon JW (1969). "A Nonlinear Mapping for Data Structure Analysis." *IEEE Transactions on Computers*, **C-18**(5), 401–409. doi:10.1109/T-C.1969.222678.

Simpson GL (2007). "Analogue Methods in Palaeoecology: Using the **analogue** Package." *Journal of Statistical Software*, **22**(2), 1–29.

Takane Y, Young F, De Leeuw J (1977). "Nonmetric Individual Differences Multidimensional Scaling: An Alternating Least Squares Method with Optimal Scaling Features." *Psychometrika*, **42**(1), 7–67. doi:10.1007/BF02293745.

Tenenbaum JB, De Silva V, Langford JC (2000). "A Global Geometric Framework for Nonlinear Dimensionality Reduction." *Science*, **290**(5500), 2319–2323. doi:10.1126/science.290.5500.2319.

Thrun MC, Ultsch A (2021). "Using Projection-Based Clustering to Find Distance- and Density-Based Clusters in High-Dimensional Data." *Journal of Classification*, **38**, 280–312. doi:10.1007/s00357-020-09373-2.

Torgerson WS (1958). *Theory and Methods of Scaling*. John Wiley & Sons, New York.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. Fourth edition. Springer-Verlag, New York.

Venkatesh V, Podoshen JS, Urbaniak K, Wallin JJ (2015). "Eschewing Community: Black Metal." *Journal of Community & Applied Social Psychology*, **25**(1), 66–81. doi:10.1002/casp.2197.

Vera JF, Mair P (2019). "**SEMDS**: An R Package for Structural Equation Multidimensional Scaling." *Structural Equation Modeling: A Multidisciplinary Journal*, **26**(5), 803–818. doi:10.1080/10705511.2018.1561292.

Zielman B (2022). ***asymmetry**: Multidimensional Scaling of Asymmetric Proximities*. R package version 2.0.4, URL https://CRAN.R-project.org/package=asymmetry.

Affiliation:

Thomas Rusch
Competence Center for Empirical Research Methods
WU (Wirtschaftsuniversität Wien)
Welthandelsplatz 1, D5
1020 Wien, Austria
E-mail: Thomas.Rusch@wu.ac.at

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd