# Package 'smacof'

February 15, 2021

**Type** Package

**Title** Multidimensional Scaling

**Version** 2.1-2

**Date** 2021-02-10

**Description**
Implements the following approaches for multidimensional scaling (MDS) based on stress minimization using majorization (smacof): ratio/interval/ordinal/spline MDS on symmetric dissimilarity matrices, MDS with external constraints on the configuration, individual differences scaling (idioscal, indscal), MDS with spherical restrictions, and ratio/interval/ordinal/spline unfolding (circular restrictions, row-conditional). Various tools and extensions like jackknife MDS, bootstrap MDS, permutation tests, MDS biplots, gravity models, unidimensional scaling, drift vectors (asymmetric MDS), classical scaling, and Procrustes are implemented as well.

**Imports** graphics, stats, polynom, Hmisc, nnls, grDevices, MASS, weights, ellipse, wordcloud, candisc, parallel, foreach, doParallel

**Depends** R (>= 3.5.0), plotrix, colorspace, e1071

**License** GPL-3

**Suggests** knitr, prefmod, MPsychoR, calibrate, ggplot2, rmarkdown

**VignetteBuilder** knitr

**LazyData** yes

**LazyLoad** yes

**ByteCompile** yes

**NeedsCompilation** yes

**Author** Patrick Mair [aut, cre],
Jan De Leeuw [aut],
Patrick J. F. Groenen [aut],
Ingwer Borg [ctb]

**Maintainer** Patrick Mair <mair@fas.harvard.edu>

**Repository** CRAN

**Date/Publication** 2021-02-15 11:03:13 UTC

# R topics documented:

---

`biplotmds` *MDS Biplots*

---

## Description

Regresses external variables on a MDS configuration which results in a MDS biplot.

## Usage

```
## S3 method for class 'smacof'
biplotmds(object, extvar, scale = TRUE)
## S3 method for class 'mdsbi'
plot(x, vecscale = NULL, plot.dim = c(1,2), sphere = TRUE, col = 1,
                   label.conf = list(label = TRUE, pos = 3, col = 1, cex = 0.8),
                     vec.conf = list(col = 1, cex = 0.8, length = 0.1),
                     identify = FALSE, type = "p", pch = 20,
                     asp = 1, main, xlab, ylab, xlim, ylim, ...)
```

## Arguments

| | |
|---|---|
| `object` | Object of class `"smacof"` or `"smacofID"`. |
| `extvar` | Data frame with external variables. |
| `scale` | If `TRUE`, external variables are standardized internally. |
| `x` | Object of class `"mdsbi"`. |
| `vecscale` | Scaling factor for regression coefficients, either a single number or `NULL` (see details). |
| `plot.dim` | Vector with dimensions to be plotted. |
| `main` | Plot title. |

| | |
|---|---|
| xlab | Label of x-axis. |
| ylab | Label of y-axis. |
| xlim | Scale x-axis. |
| ylim | Scale y-axis. |
| pch | Plot symbol. |
| asp | Aspect ratio. |
| col | Point color. |
| type | What type of plot should be drawn. |
| sphere | In case of spherical smacof, whether sphere should be plotted or not. |
| label.conf | List with arguments for plotting the labels of the configurations in a configuration plot (logical value whether to plot labels or not, label position, label color). |
| vec.conf | List with arguments for arrows and arrow labels of the external variables. |
| identify | If TRUE, the identify() function is called internally that allows to add configuration labels by mouse click. |
| ... | Further plot arguments passed: see plot for detailed information. |

## Details

If a model for individual differences is provided, the external variables are regressed on the group stimulus space configurations. In the biplot only the relative length of the vectors and their direction matters. Using the scale argument the user can control for the relative length of the vectors. If vecscale = NULL, the vecscale() function from the **candisc** package is used which tries to automatically calculate the scale factor so that the vectors approximately fill the same space as the configuration.

## Value

Returns an object belonging to classes "mlm" and "mdsbi". See lm for details.

| | |
|---|---|
| R2vec | Vector containing the R2 values. |

## References

Greenacre, M. (2010). Biplots in Practice. Fundacion BBVA, Bilbao, Spain

## See Also

plot.smacof

## Examples

```
## morse code data with external scales
res <- mds(morse)
fitbi <- biplotmds(res, morsescales[,2:3])
plot(fitbi, main = "MDS Biplot", vecscale = 0.5)

## wish data with external economic development factor
```

```
diss <- sim2diss(wish, method = 7)
res <- mds(diss, type = "ordinal")
ecdev <- data.frame(ecdev = c(3,1,3,3,8,3,7,9,4,7,10,6))
fitbi <- biplotmds(res, ecdev)
plot(fitbi, main = "MDS Biplot", vecscale = 1)
plot(fitbi, main = "MDS Biplot", vecscale = 0.5, xlim = c(-1, 1),
vec.conf = list(col = "red", length = 0.05))

## Ekman's color data (by Michael Friendly)
require(colorspace)
wavelengths <- attr(ekman, "Labels")
colors <- c("#2600F0", "#0028FF", "#0092FF", "#00B2FF", "#00FFFF", "#00FF61", "#77FF00", "#B3FF00",
            "#FFF200", "#FFBE00", "#FF9B00", "#FF5700", "#F60000", "#D60000")
ekmanD <- sim2diss(ekman)
res <- mds(ekmanD, type = "ordinal")
RGB <- t(col2rgb(colors)) / 255
HCL <- as(hex2RGB(colors), "polarLUV")
HCL <- slot(HCL, "coords")
fit <- biplotmds(res, cbind(RGB, HCL))
fit
plot(fit, vecscale = 0.5, cex = 6, col = colors,
    label.conf=list(cex = 1, pos = ifelse(wavelengths < 560, 2, 4)),
    vec.conf = list(cex = 1.2), main = "Ekman configuration and color properties" )
```

---

bootmds *SMACOF Bootstrap*

---

## Description

Performs a bootstrap on a SMACOF solution. It works for derived dissimilarities only. The original data matrix needs to be provided, as well as the type of dissimilarity measure used to compute the input dissimilarities.

## Usage

```
## S3 method for class 'smacofB'
bootmds(object, data,  method.dat = "pearson", nrep = 100,
alpha = 0.05, verbose = FALSE, ...)

## S3 method for class 'smacofboot'
plot(x, plot.dim = c(1,2), col = 1,
label.conf = list(label = TRUE, pos = 3, cex = 0.8),
ell = list(lty = 1, lwd = 1, col = "gray"), main, xlab, ylab, xlim, ylim,
asp = 1, type = "p", pch = 20, ...)
```

## Arguments

| | |
|---|---|
| object | Object of class "smacofB", i.e., an MDS solution from mds(). |

| | |
|---|---|
| data | Initial data (before dissimilarity computation). |
| method.dat | Dissimilarity computation used as MDS input. This must be one of "pearson", "spearman", "kendall", "euclidean", "maximum", "manhattan", "canberra", "binary". For unfolding models it is either "full" for full permutations or "rows" for permutations within rows. |
| nrep | Number of bootstrap replications. |
| alpha | Alpha level for confidence ellipsoids. |
| verbose | If TRUE, bootstrap index is printed out. |
| ... | Additional arguments needed for dissimilarity computation as specified in sim2diss(). |
| x | Object of class "smacofboot" |
| plot.dim | Vector with dimensions to be plotted. |
| col | Color for points. |
| label.conf | List with arguments for plotting the labels of the configurations in a configuration plot (logical value whether to plot labels or not, label position). If pos = 5 labels are placed away from the nearest point. |
| ell | List with arguments for plotting ellipses: line type, line width, color. |
| main | Plot title. |
| xlab | Label of x-axis. |
| ylab | Label of y-axis. |
| xlim | Scale x-axis. |
| ylim | Scale y-axis. |
| asp | Aspect ratio. |
| pch | Plotting symbol for object point. |
| type | Type of plot. |

## Details

In order to examine the stability solution of an MDS, a bootstrap on the raw data can be performed. This results in confidence ellipses in the configuration plot. The ellipses are returned as list which allows users to produce (and further customize) the plot by hand.

## Value

| | |
|---|---|
| cov | Covariances for ellipse computation |
| bootconf | Configurations bootstrap samples |
| stressvec | Bootstrap stress values |
| bootci | Stress bootstrap percentile confidence interval |
| stab | Stability coefficient |

## References

Jacoby, W. G., & Armstrong, D. A. (2014). Bootstrap confidence regions for multidimensional scaling solutions. American Journal of Political Science, 58, 264-278.

## See Also

jackmds

## Examples

```
## Example using Euclidean distances
data <- na.omit(PVQ40[,1:5])
diss <- dist(t(data))   ## Euclidean distances
fit <- mds(diss)        ## 2D interval MDS

set.seed(123)
resboot <- bootmds(fit, data, method.dat = "euclidean", nrep = 50)
resboot
plot(resboot)

## Example using Pearson correlations
sim <- cor(data)
diss <- sim2diss(sim, method = 1)  ## subtract from 1 (method needs to be passed to bootmds)
fit <- mds(diss, type = "ratio", ndim = 3)        ## 3D ratio MDS

set.seed(123)
resboot <- bootmds(fit, data, method.dat = "pearson", nrep = 50, alpha = 0.1, method = 1)
resboot
## plot 1st against 3rd dimension
ellipses <- plot(resboot, plot.dim = c(1,3), ell = list(lty = 2, col = "gray", lwd = 0.8))
str(ellipses)  ## list of ellipse coordinates for each object
```

---

# bread *Breakfast preferences*

---

## Description

The data set is described in Bro (1998). The raw data consist of ratings of 10 breads on 11 different attributes carried out by 8 raters. Note that the bread samples are pairwise replications: Each of the 5 different breads, which have a different salt content, was presented twice for rating.

## Usage

```
data(bread)
```

## Format

A list of length 8 with elements of class `"dist"`. The attributes are bread odor, yeast odor, off-flavor, color, moisture, dough, salt taste, sweet taste, yeast taste, other taste, and total taste.

## References

Bro, R. (1998). Multi-way Analysis in the Food Industry: Models, Algorithms, and Applications. Ph.D. thesis, University of Amsterdam (NL) & Royal Veterinary and Agricultural University (DK).

## Examples

```
bread
```

---

`breakfast` *Breakfast preferences*

---

## Description

42 individuals were asked to order 15 breakfast items due to their preference.

## Usage

```
data(breakfast)
```

## Format

Data frame with students in the rows and breakfast items in the columns.

toast: toast pop-up

butoast: buttered toast

engmuff: English muffin and margarine

jdonut: jelly donut

cintoast: cinnamon toast

bluemuff: blueberry muffin and margarine

hrolls: hard rolls and butter

toastmarm: toast and marmalade

butoastj: buttered toast and jelly

toastmarg: toast and margarine

cinbun: cinnamon bun

danpastry: Danish pastry

gdonut: glazed donut

cofcake: coffee cake

cornmuff: corn muffin and butter

## References

Green, P. E. & Rao, V. (1972). Applied multidimensional scaling. Hinsdale, IL: Dryden.

## Examples

```
breakfast
```

## CanadaNews *Canadian Newspapers*

### Description

We took Canadian newspapers that appeared in the time period between June and September 2009 and searched for articles that contained the word "aboriginal". A total of 92 articles was found. In these articles, we determined the frequencies of other meaningful words (e.g., tribal, moose, arctic, and health). The data are organized as word co-occurrence matrix.

### Usage

```
data(CanadaNews)
```

### Format

Matrix with word co-occurrence counts.

### References

Borg, I., Groenen, P. J. F., & Mair, P. (2017). Applied Multidimensional Scaling and Unfolding. New York: Springer.

### Examples

```
str(CanadaNews)
```

## confEllipse *Pseudo Confidence Ellipses*

### Description

Computes pseudo-confidence ellipses for symmetric and individual difference MDS fits.

### Usage

```
## S3 method for class 'smacofID'
confEllipse(object)

## S3 method for class 'confell'
plot(x, eps = 0.05, plot.dim = c(1,2), col = 1,
label.conf = list(label = TRUE, pos = 3, cex = 0.8),
ell = list(lty = 1, lwd = 1, col = 1), main, xlab, ylab, xlim, ylim,
asp = 1, type = "p", pch = 20, ...)
```

## Arguments

| | |
|---|---|
| object | Object of class "smacofB" or "smacofID". |
| x | Object of class "confell" |
| eps | Perturbation region (e.g. 0.05 means that we look at a perturbation region where stress is at most 5% larger than the minimum we have found). |
| plot.dim | Vector with dimensions to be plotted. |
| col | Color for points. |
| label.conf | List with arguments for plotting the labels of the configurations in a configuration plot (logical value whether to plot labels or not, label position). If pos = 5 labels are placed away from the nearest point. |
| ell | List with arguments for plotting ellipses: line type, line width, color. |
| main | Plot title. |
| xlab | Label of x-axis. |
| ylab | Label of y-axis. |
| xlim | Scale x-axis. |
| ylim | Scale y-axis. |
| asp | Aspect ratio. |
| pch | Plotting symbol for object point. |
| type | Type of plot. |
| ... | Additional plotting arguments. |

## Details

The confEllipse function normalizes the dissimilarities and performs a few more iterations to optimize the configuration and the individual diffierence weights. This result is then passed to a function that computes the stress derivatives which are the basis of the ellipses in the plot function. This function works for ratio scaled versions only.

## Value

Returns an object belonging to classes "confell".

| | |
|---|---|
| X | Configuration (group stimulus space for individual difference models) |
| h | Stress derivatives |
| s | Optimized stress (raw value) |

## References

De Leeuw, J. (2019). Individual Differences Multidimensional Scaling.

## See Also

plot.smacofboot

## Examples

```
## Simple ratio MDS fit
delta <- sim2diss(cor(PVQ40agg))
res <- mds(delta, ndim = 3)
cres <- confEllipse(res)
plot(cres, plot.dim = c(1,2))
plot(cres, plot.dim = c(1,3))
plot(cres, plot.dim = c(2,3))


## INDSCAL on Helm data
fit1 <- indscal(helm)
cfit1 <- confEllipse(fit1)
plot(cfit1, ell = list(col = "gray", lty = 2), ylim = c(-0.04, 0.04))

## IDIOSCAL on Helm data
fit2 <- idioscal(helm)
cfit2 <- confEllipse(fit2)
plot(cfit1, ell = list(col = "gray", lty = 2), ylim = c(-0.04, 0.04))
```

---

# crimes *Crime Correlations*

---

## Description

Correlations of crime rates in 50 US states.

## Usage

```
data(crimes)
```

## Format

Crime correlation matrix.

## References

Borg, I., Groenen, P. J. F., & Mair, P. (2017). Applied Multidimensional Scaling and Unfolding. New York: Springer.

## Examples

```
crimes
```

## csrranking *CSR activities*

### Description

This dataset collects rankings of 100 individual on 5 topics that reflect social responsibilties on corporations.

### Usage

```
data(csrranking)
```

### Format

A data frame where each individual ranked prevention of environmental pollution (Environment), waste prevention (Waste Prevention), selling organic products (Organic Products), participating on charity programs (Charity), and fair treatment of employees (Employee) according to its own preferences. A value of 1 corresponds to highest importance, 5 to lowest importance.

### Examples

```
csrranking
```

## dissWeights *Create Weights for Uniform Weighted Distribution*

### Description

Compute weights as a function of the dissimilarities.

### Usage

```
dissWeights(delta, type = c("unif", "knn", "power", "unifpower"),
           k = NULL, power = 0)
```

### Arguments

| | |
|---|---|
| delta | Either a symmetric dissimilarity matrix or an object of class "dist" |
| type | One of "unif" (default), "knn", "power", "unifpower". See details for a description of the various options. |
| k | The number of smallest dissimilarities per row for which the weights need to be set to 1. The default k = NULL makes k to be set to use the 25% smallest dissimilarities per row. |
| power | power to which the dissimilarities need to be raised as weights. Default is 0, so that all weights are 1. |

## Details

The weights are computed as a function of the dissimilarities depending on `type`.

- `"unif"` Compute weights such that the weighted empirical distribution (histogram) of the dissimilarities is uniform. Particularly if the number of objects is large, the dissimilarities that occuur most often will start to dominate the solution. This option de-emphasizes often occuring dissimilarities such that the weighted empirical distribution (the weighted histogram) becomes approximately uniform.

- `"knn"` Per row of the dissimilarity matrix the `k` smallest dissimilarities obtain a weight of 1 and the others a 0.

- `"power"` The weights are set to the `delta^power`. If `power` is small (e.g., `power = -5`) then the smaller dissimilarities will be better fitted. If `power` is large (e.g., `power = 5`) then the larger dissimilarities will be better fitted.

- `"unifpower"` First weights are determined by the `"unif"` option and then multiplied by the weights obtained by the `"power"` option. If the dissimilarity matrix is large, then this option is needed to see an effect of the `"power"` option on the MDS solution.

## Value

`weightmat` the weight matrix

## Author(s)

Patrick Groenen

## Examples

```
## mds solution for kinship data with uniform weights
res <- mds(kinshipdelta, weightmat = dissWeights(kinshipdelta, type = "unif"))
par(mfrow = c(2,2))
plot(res, main = "uniform weights")
plot(res, plot.type = "Shepard")
plot(res, plot.type = "histogram")

## mds solution for kinship data with knn weights
res <- mds(kinshipdelta, weightmat = dissWeights(kinshipdelta, type = "knn", k = 5))
par(mfrow = c(1,2))
plot(res, main = "knn weights with k=5")
plot(res, plot.type = "Shepard")

## mds solution for kinship data with power weights emphasizing large dissimilarities
res <- mds(kinshipdelta, weightmat = dissWeights(kinshipdelta, type = "power", power = 5))
par(mfrow = c(2,2))
plot(res, main = "Power = 5 weights")
plot(res, plot.type = "Shepard")
plot(res, plot.type = "histogram")

## mds solution for kinship data with power weights emphasizing small dissimilarities
res <- mds(kinshipdelta, weightmat = dissWeights(kinshipdelta, type = "power", power = -5))
```

```
par(mfrow = c(2,2))
plot(res, main = "Power = -5 weights")
plot(res, plot.type = "Shepard")
plot(res, plot.type = "histogram")

## mds solution for kinship data with power weights emphasizing large dissimilarities
## while correcting for nonuniform dissimilarities
res <- mds(kinshipdelta, weightmat = dissWeights(kinshipdelta, type = "unifpower", power = 5))
par(mfrow = c(2,2))
plot(res, main = "Uniform power = 5 weights")
plot(res, plot.type = "Shepard")
plot(res, plot.type = "histogram")

## mds solution for kinship data with power weights emphasizing small dissimilarities
## while correcting for nonuniform dissimilarities
res <- mds(kinshipdelta, weightmat = dissWeights(kinshipdelta, type = "unifpower", power = -5))
par(mfrow = c(2,2))
plot(res, main = "Uniform power = -5 weights")
plot(res, plot.type = "Shepard")
plot(res, plot.type = "histogram")
```

---

`driftVectors` *Asymmetric MDS: Drift Vectors*

---

## Description

Takes an asymmetric dissimilarity matrix and decomposes it into a symmetric and a skew-symmetric part. Fits an MDS on the symmetric part and computes drift vectors for the skew-symmetric portion. This model makes it possible to see how these two components are related to each other. It is limited to two dimensions only.

## Usage

```
driftVectors(data, type = c("ratio", "interval", "ordinal","mspline"),
          weightmat = NULL, init = "torgerson", ties = "primary",  verbose = FALSE,
              relax = FALSE, modulus = 1, itmax = 1000, eps = 1e-6,
              spline.degree = 2, spline.intKnots = 2)

## S3 method for class 'driftvec'
plot(x, adjust = 1, main, xlim, ylim,
                    xlab = "Dimension 1", ylab = "Dimension 2", pch = 20, asp = 1,
                         col.conf = "black", col.drift = "lightgray",
                         label.conf = list(label = TRUE, pos = 3, col = "black",
                         cex = 0.8), ...)
```

## Arguments

| | |
|---|---|
| data | Asymmetric dissimilarity matrix |
| weightmat | Optional matrix with dissimilarity weights |
| init | Either "torgerson" (classical scaling starting solution), "random" (random configuration), or a user-defined matrix |
| type | MDS type: "interval", "ratio", "ordinal" (nonmetric MDS), or "mspline" |
| ties | Tie specification for ordinal MDS only: "primary", "secondary", or "tertiary" |
| verbose | If TRUE, intermediate stress is printed out |
| relax | If TRUE, block relaxation is used for majorization |
| modulus | Number of smacof iterations per monotone regression call |
| itmax | Maximum number of iterations |
| eps | Convergence criterion |
| spline.degree | Degree of the spline for "mspline" MDS type |
| spline.intKnots | Number of interior knots of the spline for "mspline" MDS type |
| x | Object of class "driftvec" |
| adjust | Scaling factor for drift vectors (value larger than 1 increases the length) |
| main | Plot title |
| xlab | Label of x-axis |
| ylab | Label of y-axis |
| xlim | Scale x-axis |
| ylim | Scale y-axis |
| pch | Plot symbol |
| asp | Aspect ratio |
| col.conf | Point color (MDS configurations) |
| col.drift | Color for drift vectors (arrows) |
| label.conf | Settings for plotting labels |
| ... | Additional plotting arguments |

## Details

The skew-symmetric values are embedded into the MDS representation of the symmetrized data by drawing errors (drift vectors) from each point $i$ to each point $j$ in the configuration so that these vectors correspond in length and direction to the values of row $i$ of the skew-symmetric matrix.

## Value

| | |
|---|---|
| fitsym | MDS output for symmetric portion |
| sym | Symmetric matrix |
| skewsym | Skew-symmetric matrix |

| | |
|---|---|
| driftcoor | Drift vector coordinates |
| stress | Stress-1 value |
| niter | Number of iterations |
| nobj | Number of objects |

## Author(s)

Patrick Mair

## References

Borg, I., & Groenen, P. J. F. (2005). Modern Multidimensional Scaling (2nd ed.). Springer.

## See Also

smacofSym

## Examples

```
## simple example
P <- matrix(c(0,  4,  6, 13,
              5,  0, 37, 21,
              4, 38,  0, 16,
              8, 31, 18,  0), nrow = 4, ncol = 4, byrow = TRUE)
D <- sim2diss(P, method = 40)
res <- driftVectors(D, type = "interval")
plot(res)
plot(res, adjust = 0.1)    ## shorten drift vectors

## Morse code data
fit.drift <- driftVectors(morse2, type = "ordinal")
fit.drift
plot(fit.drift)
```

---

# Duration *Facets of the subjective duration of imagined situations*

---

## Description

The DurationRaw dataset contains the duration rating of 76 subjects on 24 situations. Subjects were asked to rate the duration on a 7 point scale (1 ... substantially shorter, 7 ... substantially longer). The Duration data file contains the corresponding correlations between the 24 situations including some information about the facets.

## Usage

```
data(Duration)
data(DurationRaw)
```

## Format

Data frame 24 correlations based on duration ratings:

S1-S24: situation

F1: pleasant (1), neutral (2), unpleasant (3)

F2: variable (1), monotonous (2)

F3: difficult (1), easy (2)

F3: many (1), few (2)

structuple: the facet structure written as a tuple

## References

Borg, I., & Groenen, P. J. F. (2005). Modern Multidimensional Scaling (2nd ed.). Springer.

## Examples

```
ddiss <- sim2diss(Duration[,paste0("S", 1:24)])
fit <- mds(ddiss, type = "ordinal", ndim = 4)
plot(fit)
```

---

`ekman` *Ekman data set*

---

## Description

Ekman dissimilarities

## Usage

```
data(ekman)
```

## Format

Object of class `dist`

## Details

Ekman presents similarities for 14 colors which are based on a rating by 31 subjects where each pair of colors was rated on a 5-point scale (0 = no similarity up to 4 = identical). After averaging, the similarities were divided by 4 such that they are within the unit interval. Similarities of colors with wavelengths from 434 to 674 nm.

## References

Ekman, G. (1954). Dimensions of color vision. Journal of Psychology, 38, 467-474.

## Examples

```
ekman
```

---

`EW_ger` *Work values*

---

## Description

Intercorrelations of 13 working values for former West (first list element) and East Germany.

## Usage

```
data(EW_eng)
```

## Format

Object of class `dist`

## Details

Note that in `EW_ger` the labels are given in German. For smacof, the data must be converted into a dissimilarity matrix by applying the `sim2diss()` function to each list element.

## References

ALLBUS 1991, German General Social Survey.

Borg, I., Groenen, P. J. F., & Mair, P. (2010). Multidimensionale Skalierung. Muenchen: Hampp Verlag.

Borg, I., Groenen, P. J. F., & Mair, P. (2012). Multidimensional Scaling. New York: Springer, forthcoming.

## Examples

```
data(EW_eng)
data(EW_ger)
```

## FaceExp *Facial Expression Data*

### Description

Dissimilarity matrix of 13 facial expressions (Abelson & Sermat, 1962). The external scales are taken from Engen et al. (1958) reflecting the following three perceptual dimensions: pleasant-unpleasant (PU), attention-rejection (AR), and tension-sleep (TS).

### Usage

```
data(FaceExp)
data(FaceScale)
```

### Format

Symmetric dissimilarity matrix and data frame with 3 perceptual dimensions

### References

Abelson, R. P., & Sermat, V. (1962). Multidimensional scaling of facial expressions. Journal of Experimental Psychology, 63, 546-554.

Engen, B., Levy, N., & Schlossberg, H. (1958). The dimensional analysis of a new series of facial expressions. Journal of Experimental Psychology, 55, 454-458.

### Examples

```
str(FaceExp)
str(FaceScale)
```

## fitCircle *Fitting circle into point configuration*

### Description

Utility function for fitting a circle into 2D point configurations.

### Usage

```
fitCircle(x, y)
```

### Arguments

| | |
|---|---|
| x | Vector with x-coordinates |
| y | Vector with y-coordinates |

## Value

| | |
|---|---|
| cx | x-coordinate center |
| cy | y-coordinate center |
| radius | circle radius |

## References

Pratt, V. (1987). Direct least-squares fitting of algebraic surfaces. Computer Graphics, Vol. 21, pages 145-152.

## Examples

```
## Dataset on Schwartz values:
require(plotrix)
valsD <- 1 - cor(indvalues)
fit <- mds(valsD)
plot(fit, main = "MDS Value Circle")
circle <- fitCircle(fit$conf[,1], fit$conf[,2])
draw.circle(circle$cx, circle$cy, radius = circle$radius, border = "gray")
```

---

# GOPdtm *Republican Statements*

---

## Description

Document-term matrix based on statements by Republican voters.

## Usage

```
data(GOPdtm)
```

## Format

Document-term matrix with statements in the rows and terms (keywords) in the columns

## Details

This dataset emerges from statements of Republican voters scraped from the official GOP website. They were asked to complete the sentence "I am a Republican because ...". We have selected the 37 most frequent words and created a document-term matrix.

## References

air, P., Rusch, T. & Hornik, K. (2014). The Grand Old Party - A party of values? SpringerPlus, 3(697), https://springerplus.springeropen.com/articles/10.1186/2193-1801-3-697

## Examples

```
data(GOPdtm)
GOPdtm
```

---

`gravity` *Gravity dissimilarities*

---

## Description

Computes the dissimilarities using a gravity model based on co-occurrences.

## Usage

```
gravity(X, lambda = 1)
```

## Arguments

| | |
|---|---|
| `X` | numeric matrix |
| `lambda` | tuning parameter |

## Details

The first step in this function is to compute the co-occurences. Based on the binarized data matrix $Y$ we compute $Y'Y$ which leads to the co-occurence matrix. We then use the gravity model to compute the gravity dissimilarities. In order to give more (or less) structure to the MDS solution, the tuning parameter (which defines a power transformation) can be increased (or decreased). In addition, a weight matrix is created that sets cells with no co-occurences to 0. The corresponding weight matrix for blanking out these cells is established automatically in `mds()`.

## Value

| | |
|---|---|
| `gravdiss` | Gravity dissimilarities |
| `weightmat` | Weight matrix for subsequent smacof computation |
| `co.occ` | Matrix with co-occurences |

## Author(s)

Patrick Mair

## References

Mair, P., Rusch, T. & Hornik, K. (2014). The Grand Old Party - A party of values? SpringerPlus, 3(697), https://springerplus.springeropen.com/articles/10.1186/2193-1801-3-697

## See Also

`mds`

## Examples

```
data(GOPdtm)
gravD <- gravity(GOPdtm, lambda = 2)
res <- mds(gravD$gravdiss)
res$weightmat  ## NA's were blanked out when fitting the model
plot(res)
```

---

Guerry *Map Dataset France 1830*

---

## Description

Distances (in km) among French Departments in 1830.

## Usage

```
data(Guerry)
```

## Format

Symmetric matrix with distances.

## References

Friendly, M. (2007). A. M. Guerry's Moral Statistics of France: Challenges for Multivariate Spatial Analysis, Statistical Science, 2007, 22(3), 368-399.

## Examples

```
Guerry
```

---

Guttman1991 *Guttman's Intelligence Facets*

---

## Description

The first dataset (Guttman1991) contains similarities and facets for Guttman's 3D cylindrical intelligence structure as published in Guttman (1991). The second dataset (Guttman1965) contains similarities and structural intelligence facets from Guttman (1965).

## Usage

```
data(Guttman1991)
data(Guttman1965)
```

### Format

List with two elements: The first element contains the similarity matrix, the second element the facets labels.

### References

Guttman, L. & Levy, S. (1991). Two structural laws for intelligence tests. Intelligence, 15, 79-103.

Guttman, L. (1965). The structure of interrelations among intelligence tests. In C. W. Harris (Ed.), Proceedings of the 1964 Invitational Conference on Testing Problems (pp. 23-36). Princeton: ETS.

### Examples

```
Guttman1991[[1]]        ## similarity matrix
Guttman1991[[2]]        ## facets

Guttman1965[[1]]        ## similarity matrix
Guttman1965[[2]]        ## facets
```

---

## helm *Helm's color data*

---

### Description

Contains dissimilarity data for individual difference scaling from an experiment carried out by Helm (1959).

### Usage

```
data(helm)
```

### Format

List containing objects of class `dist`

### Details

A detailed description of the experiment can be found in Borg and Groenen (2005, p. 451) with the corresponding Table 21.1. containing distance estimates for color pairs. There were 14 subjects that rated the similarity of colors, 2 of whom replicated the experiment. 10 subjects have a normal color vision (labelled by N1 to N10 in our list object), 4 of them are red-green deficient in varying degrees. In this dataset we give the dissimilarity matrices for each of the subjects, including the replications. They are organized as a list of length 16 suited for `smacofIndDiff` computations.

The authors thank Michael Friendly and Phil Spector for data preparation.

## References

Helm, C. E. (1959). A multidimensional ratio scaling analysis of color relations. Technical Report, Princeton University and Educational Testing Service. Princeton, NJ.

Borg, I., & Groenen, P. J. F. (2005). Modern Multidimensional Scaling: Theory and Applications (2nd edition). New York: Springer.

## Examples

```
helm
```

---

`icExplore` *Exploring Initial Configurations*

---

## Description

Allows to user to explore the effect of various random starting configurations when fitting an MDS model.

## Usage

```
icExplore(delta, nrep = 100, returnfit = FALSE, ndim = 2,
type = c("ratio", "interval", "ordinal","mspline"), weightmat = NULL, ties = "primary",
verbose = FALSE, relax = FALSE, modulus = 1, itmax = 1000, eps = 1e-6,
spline.degree = 2, spline.intKnots = 2)
```

## Arguments

| | |
|---|---|
| `delta` | Either a symmetric dissimilarity matrix or an object of class `"dist"` |
| `nrep` | Number of initial random configurations |
| `returnfit` | If `TRUE` all fitted models are returned. |
| `ndim` | Number of dimensions |
| `weightmat` | Optional matrix with dissimilarity weights |
| `type` | MDS type: `"interval"`, `"ratio"`, `"ordinal"` (nonmetric MDS), or `"mspline"` |
| `ties` | Tie specification (ordinal MDS only): `"primary"`, `"secondary"`, or `"tertiary"` |
| `verbose` | If `TRUE`, replication number is printed |
| `relax` | If `TRUE`, block relaxation is used for majorization |
| `modulus` | Number of smacof iterations per monotone regression call |
| `itmax` | Maximum number of iterations |
| `eps` | Convergence criterion |
| `spline.degree` | Degree of the spline for `"mspline"` MDS type |
| `spline.intKnots` | Number of interior knots of the spline for `"mspline"` MDS type |

## Details

This function generates a large set of MDS solutions using random initial configurations, matches them all by Procrustean fittings, computes the inter-correlations of their point coordinates, and finally runs an interval MDS of these inter-correlations. It can be used to explore local minima.

In the plot function the number reflects the index of corresponding MDS fit, the size reflects the stress value: the larger the font, the larger the stress (i.e., the worse the solution). The size is associated with a corresponding color shading (the smaller the size the darker the color).

## Value

| | |
|---|---|
| `mdsfit` | Fitted MDS objects (`NULL` if `returnfit = FALSE`) |
| `conf` | Configuration based on multiple random starts |
| `stressvec` | Vector with stress values |

## References

Borg, I. and Mair, P. (2017). The choice of initial configurations in multidimensional scaling: local minima, fit, and interpretability. Austrian Journal of Statistics, 46, 19-32. doi: 10.17713/ajs.v46i2.561

## See Also

`mds`

## Examples

```
## simple example with 20 random starts
diss <- sim2diss(wish, method = 7)
set.seed(123)
res <- icExplore(diss, type = "ordinal", nrep = 20, returnfit = TRUE)
res
plot(res)

res$mdsfit[[14]]  ## bad fitting solution
res$mdsfit[[3]]   ## better fitting solution
```

---

# indvalues *Schwartz Value Survey*

---

## Description

Responses from a sample in Britain were collected varying in value measures of the Schwartz value theory. The instrument used was the Schwartz Value Survey (SVS).

## Usage

```
data(indvalues)
```

## Format

Data frame with 327 persons in the rows and psychological values in the columns.

## Details

The data were centered (row-wise) and converted from preferences into dissimilarities.

## References

Borg, I., Bardi, A., & Schwartz, S. H. (2017). Does the value circle exist within persons or only across persons? Journal of Personality, 85(2), 151-162.

## See Also

PVQ40

## Examples

```
str(indvalues)
```

---

`intelligence` *Intelligence Tests*

---

## Description

Contains intercorrelations of 8 intelligence tests, together with two facets. In addition, a hypothesized restriction matrix for the intercorrelations is provided. The proximities for items with the same structuples, such as p(NA1,NA2) and p(GA1,GA3), all are set to the value 5. The proximities that correspond to the immediate neighborhood relations are set to the value 4, since none of these distances should be larger than any distance between definitionally equivalent items. Finally, the large distances between the groups NI, GA and the groups NA, GI are set to 3. The intelligence tests are coded on the follwing facets: format (N = numerical, G = geometrical) and requirement (A = application, I - inference).

## Usage

```
data(intelligence)
```

## Format

Data frame of 8 intelligence tests: facets, intercorrelations, and restrictions

Test: Test number

Languange: numerical, geometrical

Requirement: application, inference

T1-T8: intercorrelations

R1-R8: restrictions

## References

Borg, I., & Groenen, P. J. F. (2005). Modern Multidimensional Scaling (2nd ed.). Springer.

## Examples

```
idiss <- sim2diss(intelligence[,paste0("T", 1:8)])
fit <- mds(idiss)
plot(fit)
```

---

# jackmds *SMACOF Jackknife*

---

## Description

These methods perform a SMACOF Jackknife and plot the corresponding solution.

## Usage

```
## S3 method for class 'smacofB'
jackmds(object, eps = 1e-6, itmax = 100, verbose = FALSE)

## S3 method for class 'smacofJK'
plot(x, plot.dim = c(1,2), hclpar = list(c = 50, l = 70),
col.p, col.l, plot.lines = TRUE, legend = FALSE, inset = c(-0.2, 0), cex.legend = 0.7,
main, xlab, ylab, xlim, ylim, asp = 1, ...)
```

## Arguments

| | |
|---|---|
| object | Object of class "smacofB", i.e., an MDS solution from smacofSym() |
| itmax | Maximum number of iterations |
| eps | Convergence criterion |
| verbose | If TRUE, intermediate stress is printed out |
| x | Object of class "smacofJK" |
| plot.dim | Vector with dimensions to be plotted. |
| hclpar | Chroma and luminance to be used for HCL colors (further details see rainbow_hcl) |
| col.p | Point color. If omitted, hcl colors will be used; if specified, the corresponding (single) color will be used for plotting. |
| col.l | Line color. If omitted, hcl colors will be used; if specified, the corresponding (single) color will be used for plotting. |
| plot.lines | If TRUE, the Jackknife configurations are plotted are connected with their centroid. |
| legend | If TRUE, the centroid labels are added as legend. |

| | |
|---|---|
| inset | Inset distance from the margins as a fraction of the plot region when legend is placed by keyword. |
| cex.legend | Character expansion factor for legend. |
| main | Plot title. |
| xlab | Label of x-axis. |
| ylab | Label of y-axis. |
| xlim | Scale x-axis. |
| ylim | Scale y-axis. |
| asp | Aspect ratio. |
| ... | Further plot arguments passed: see plot for detailed information. |

## Details

In order to examine the stability solution of an MDS, a Jackknife on the configurations can be performed (see de Leeuw & Meulman, 1986) and plotted. The plot shows the jackknife configurations which are connected to their centroid. In addition, the original smacof configuration (transformed through Procrustes) is plotted. The Jackknife function itself returns also a stability measure (as ratio of between and total variance), a measure for cross validity, and the dispersion around the original smacof solution.

## Value

| | |
|---|---|
| smacof.conf | SMACOF configurations |
| jackknife.conf | An array of n-1 configuration matrices for each Jackknife MDS solution |
| comparison.conf | Centroid Jackknife configurations (comparison matrix) |
| stab | Stability measure |
| cross | Cross validity |
| disp | Dispersion |
| loss | Value of the loss function |
| ndim | Number of dimensions |
| call | Model call |
| niter | Number of iterations |
| nobj | Number of objects |

## Author(s)

Jan de Leeuw and Patrick Mair

## References

De Leeuw, J., & Meulman, J. (1986). A special jackknife for multidimensional scaling. Journal of Classification, 3, 97-112.

## See Also

bootmds

## Examples

```
## symmetric smacof
data <- na.omit(PVQ40[,1:5])
diss <- dist(t(data))   ## Euclidean distances
fit <- mds(diss)
res.jk <- jackmds(fit)

plot(res.jk, col.p = "black", col.l = "gray")
plot(res.jk, hclpar = list(c = 80, l = 40))
plot(res.jk, hclpar = list(c = 80, l = 40), plot.lines = FALSE)
```

---

# kinshipdelta *Kinship Terms*

---

## Description

Percentages of how often 15 kinship terms were not grouped together by college students including three external scales.

## Usage

```
data(kinshipdelta)

data(kinshipscales)
```

## Format

Dissimilarity matrix of 15 kinship terms and data frame with the following external scales:

Gender (1 = male, 2 = female)

Generation (-2 = two back, -1 = one back, 0 = same generation, 1 = one ahead, 2 = two ahead)

Degree (1 = first, 2 = second, 3 = third, 4 = fourth)

## References

Rosenberg, S. & Kim, M. P. (1975). The method of sorting as a data gathering procedure in multivariate research. Multivariate Behavioral Research, 10, 489-502.

## Examples

```
kinshipdelta
kinshipscales
```

KIPT *Kennedy Institute Phonics Test*

## Description

Contains correlations of eight test items of the Kennedy Institute Phonics Test (KIPT), a test for reading skills.

## Usage

```
data(KIPT)
```

## Format

An 8 times 8 correlation matrix. Items:

Nonsense word production: NP

Long vowel production: LVP

Short vowel production: SVP

Consonant cluster production: CCP

Nonsense word recognition: NR

Single letter production: SLP

Consonant cluster recognition: CCR

Initial letter recognition: ILR

## References

Guthrie, J. T. (1973). Models of reading and reading disability. Journal of Educational Psychology, 65, 9-18.

## Examples

```
KIPT
sim2diss(KIPT)
```

## LawLer *Management Performance Data*

### Description

Performance of managers: 3 criteria ("traits") and 3 methods. Traits: T1 = Quality of output, T2 = Ability to generate output, T3 = Demonstrated effort to perform. Methods: M1 = Rating by superior, M2 = Peer rating, M3 = Self-rating.

### Usage

```
data(Lawler)
```

### Format

Symmetric matrix (trait-method combinations) with inter-correlations.

### References

Lawler, E. E. (1967). Management performance as seen from above, below, and within. In Evaluation of executive performance. Princeton, New Jersey. Educational Testing Service.

### Examples

```
Lawler
```

## morse *Morse Code Confusion Data*

### Description

Confusion percentages between Morse code signals. The scores are derived from confusion rates on 36 Morse code signals (26 for the alphabet; 10 for the numbers 0,...,9). Each Morse code signal is a sequence of up to five 'beeps'. The beeps can be short (0.05 sec) or long (0.15 sec), and, when there are two or more beeps in a signal, they are separated by periods of silence (0.05 sec).

Rothkopf asked 598 subjects to judge whether two signals, presented acoustically one after another, were the same or not. The values are the average percentages with which the answer 'Same!' was given in each combination of row stimulus i and column stimulus j, where either i or j was the first signal presented. The values are 1 minus the symmetrized confusion rates and are thus dissimilarities.

### Usage

```
data(morse)
data(morse2)
```

## Format

Symmetric and asymmetric dissimilarity matrices of 36 morse codes

## Details

The first dataset (`morse`) contains a symmetric version, the second dataset (`morse2`) the original asymmetric version.

## References

Rothkopf, E. Z. (1957). A measure of stimulus similarity and errors in some paired-associate learning. Journal of Experimental Psychology, 53, 94-101.

## Examples

```
morse
morse2
```

---

`morsescales` *Morse Code Confusion Scales*

---

## Description

Two properties of Morse code signals. Each Morse code signal is a sequence of up to five 'beeps'. The beeps can be short (0.05 sec) or long (0.15 sec), and, when there are two or more beeps in a signal, they are separated by periods of silence (0.05 sec). The two external variables are:

- Signal type:
  - 1: All short beeps
  - 2: More short than long beeps
  - 3: Same short and long beeps
  - 4: More long than short beeps
  - 5: All long beeps
- Signal length (in seconds): 1 = .05, 2 = .15, 3 = .25, 4 = .35, 5 = .45, 6 = .55, 7 = .65, 8 = .85, 9 = .95

## Usage

```
data(morsescales)
```

## Format

Matrix of 36 morse codes by 2 properties. The first column contains the morse code letters.

## References

Rothkopf, E. Z. (1957). A measure of stimulus similarity and errors in some paired-associate learning. Journal of Experimental Psychology, 53, 94-101.

## Examples

```
morsescales
```

---

OCP *Organizational Culture Profile*

---

## Description

Contains similarities (correlations) of 54 OCP (see O'Reilly, Chatman, and Caldwell, 1991) items. The last three columns contain the facet assigned by Bilsky and Jehn (2002) as well as the external variables for regional restrictions.

## Usage

```
data(OCP)
```

## Format

Data frame with OCP item correlations and facet:

i1-i54: OCP item correlations

facet: factor with facets

z1, z2: external constraints

## References

Bilsky, W. & Jehn, K. (2002). Organizational Culture and Individual Values: Evidence for a Common Structure. In M. Myrtek (Ed.), The Person in Biological and Social Context, pp. 211-228. Goettingen, Germany: Hogrefe Press.

## Examples

```
ocpD <- sim2diss(OCP[,1:54])
fit <- mds(ocpD, type = "ordinal")
plot(fit)
```

## partypref *Party preferences*

### Description

Artificial dataset containing the judges in the rows and the parties in the columns.

### Usage

```
data(partypref)
```

### Format

Matrix of party preferences.

### References

Borg, I., Groenen, P. J. F., & Mair, P. (2010). Multidimensionale Skalierung. Muenchen: Hampp Verlag.

### Examples

```
partypref
```

## perception *Rectangle Perception Data*

### Description

42 subjects are assigned to two groups of 21 persons. 120 stimulus pairs of rectangles are presented. For the first group (width-height; WH), the rectangles were constructed according to a design as given in `rect_constr`. For the second group (size-shape; SS) the rectangles were constructed according to a grid design, which is orthogonal in the dimensional system reflecting area (size), and width/height (shape). All subjects had to judge the similarity of the rectangles on a scale from 0 to 9.

### Usage

```
data(perception)
```

### Format

List of subject dissimilarities for WH (first element) and SS group (second element).

## References

Borg, I. & Leutner, D. (1983). Dimensional models for the perception of rectangles. Perception and Psychophysics, 34, 257-269.

## See Also

rectangles

## Examples

```
perception
rect_constr
```

---

# permtest *SMACOF Permutation*

---

## Description

These methods perform a permutation test for a symmetric or an unfolding SMACOF model.

## Usage

```
## S3 method for class 'smacof'
permtest(object, data,  method.dat = "pearson", nrep = 100, verbose = TRUE, ...)
## S3 method for class 'smacofR'
permtest(object, data = NULL,  method.dat = "rows", nrep = 100, verbose = TRUE, ...)
## S3 method for class 'smacofPerm'
plot(x, alpha = 0.05, main, xlab, ylab, ...)
```

## Arguments

| | |
|---|---|
| object | Object of class "smacofB", i.e., an MDS solution from smacofSym() |
| data | Optional argument; if provided permutations are performed on the data matrix (see details; ignored for unfolding models) |
| method.dat | If data are provided, this must be one of "pearson", "spearman", "kendall", "euclidean", "maximum", "manhattan", "canberra", "binary". For unfolding models it is either "full" for full permutations or "rows" for permutations within rows. |
| nrep | Number of permutations |
| verbose | If TRUE, permutation index is printed out |
| x | Object of class "smacofPerm" |
| alpha | Alpha level |
| main | Plot title. |
| xlab | Label of x-axis. |

| | |
|---|---|
| ylab | Label of y-axis. |
| ... | additional plot arguments for plot function; additional arguments to be passed to sim2diss in permutation functions. |

## Details

This routine permutes m dissimilarity values, where m is the number of lower diagonal elements in the corresponding dissimilarity matrix. For each sample a symmetric, nonmetric SMACOF of dimension ndim is computed and the stress values are stored in stressvec. Using the fitted stress value, the p-value is computed. Subsequently, the empirical cumulative distribution function can be plotted using the plot method.

If the MDS fit provided on derived proximities of a data matrix, this matrix can be passed to the permtest function. Consequently, the data matrix is subject to permutations. The proximity measure used for MDS fit has to match the one used for the permutation test. If a correlation similarity is provided, it is converted internally into a dissimilarity using sim2diss with corresponding arguments passed to the ... argument.

## Value

| | |
|---|---|
| stressvec | Vector containing the stress values of the permutation samples |
| stress.obs | Stress (observed sample) |
| pval | Resulting p-value |
| call | Model call |
| nrep | Number of permutations |
| nobj | Number of objects |

## Author(s)

Patrick Mair and Ingwer Borg

## See Also

jackmds, bootmds

## Examples

```
## permuting the dissimilarity matrix (full)
data(kinshipdelta)
fitkin <- mds(kinshipdelta, ndim = 2, type = "interval")
set.seed(222)
res.perm <- permtest(fitkin)
res.perm
plot(res.perm)

## permuting the data matrix
GOPdtm[GOPdtm > 1] <- 1     ## use binary version
diss1 <- dist(t(GOPdtm[,1:10]), method = "binary")  ## Jaccard distance
```

```
fitgop1 <- mds(diss1, type = "ordinal")
fitgop1
set.seed(123)
permtest(fitgop1, GOPdtm[,1:10], nrep = 10, method.dat = "binary")

rmat <- cor(GOPdtm[,1:10], method = "kendall")  ## Kendall correlation
diss2 <- sim2diss(rmat, method = 1)
fitgop2 <- mds(diss2, type = "ordinal")
fitgop2
set.seed(123)
permtest(fitgop2, GOPdtm[,1:10], nrep = 10, method.dat = "kendall", method = 1)

## unfolding permutation
data(breakfast)
res.unfolding <- unfolding(breakfast, ndim = 2)
set.seed(123)
permtest(res.unfolding, nrep = 20, method.dat = "rows")
```

---

Plato7 *Plato's Seven Works*

---

## Description

This dataset contains statistical information about Plato's seven works. The underlying problem to this dataset is the fact that the chronological order of Plato's works is unknown. Scholars only know that Republic was his first work, and Laws his last work. For each work, Cox and Brandwood (1959) extracted the last five syllables of each sentence. Each syllable is classified as long or short which gives 32 types. Consequently, we obtain a percentage distribution across the 32 scenarios for each of the seven works.

## Usage

```
data(Plato7)
```

## Format

Data frame containing syllable percentages of Plato's 7 works.

## References

Cox, D. R. & Brandwood, L. (1959). On a discriminatory problem connected with the work of Plato. Journal of the Royal Statistical Society (Series B), 21, 195-200.

## Examples

```
Plato7
```

plot.smacof *2D SMACOF plots*

## Description

These methods provide various 2D plots for SMACOF models.

## Usage

```
## S3 method for class 'smacof'
plot(x, plot.type = "confplot", plot.dim = c(1,2), sphere = TRUE,
                  bubscale = 1, col = 1, label.conf = list(label = TRUE, pos = 3,
                    col = 1, cex = 0.8), hull.conf = list(hull = FALSE, col = 1,
                      lwd = 1, ind = NULL), shepard.x = NULL, identify = FALSE,
                     type = "p", pch = 20, cex = 0.5, asp = 1, main, xlab, ylab,
                      xlim, ylim, col.hist = NULL, ...)

## S3 method for class 'smacofR'
plot(x, plot.type = "confplot", what = c("both", "columns", "rows"),
                   plot.dim = c(1,2), col.rows = hcl(0), col.columns = hcl(240),
                       label.conf.rows = list(label = TRUE, pos = 3,
                       col = hcl(0, l = 50), cex = 0.8),
                       label.conf.columns = list(label = TRUE, pos = 3,
                       col = hcl(240, l = 50), cex = 0.8),
                       shepard.x = NULL, col.dhat = NULL, type = "p", pch = 20,
                       cex = 0.5, asp = 1, main, xlab, ylab, xlim, ylim, ...)

## S3 method for class 'smacofID'
plot(x, plot.type = "confplot", plot.dim = c(1,2), bubscale = 1,
                      col = 1, label.conf = list(label = TRUE, pos = 3, col = 1,
                   cex = 0.8), identify = FALSE, type = "p", pch = 20,  cex = 0.5,
                        asp = 1, plot.array, main, xlab, ylab, xlim, ylim, ...)
```

## Arguments

| | |
|---|---|
| x | Object of class "smacof", "smacofR", and "smacofID" (see details) |
| plot.type | String indicating which type of plot to be produced: "confplot", "resplot" "Shepard", "stressplot", "bubbleplot" "histogram" (see details) |
| plot.dim | Vector with dimensions to be plotted. |
| main | Plot title. |
| xlab | Label of x-axis. |
| ylab | Label of y-axis. |
| xlim | Scale x-axis. |
| ylim | Scale y-axis. |
| type | What type of plot should be drawn (see also plot). |

| | |
|---|---|
| pch | Plot symbol. |
| cex | Symbol size. |
| asp | Aspect ratio. |
| col | Point color. |
| sphere | In case of spherical smacof, whether sphere should be plotted or not. |
| bubscale | Scaling factor (size) for the bubble plot. |
| label.conf | List with arguments for plotting the labels of the configurations in a configuration plot (logical value whether to plot labels or not, label position, label color). If pos = 5 labels are placed away from the nearest point. |
| hull.conf | Option to add convex hulls to a configuration plot. Hull index needs to be provided. |
| shepard.x | Shepard plot only: original data (e.g. correlation matrix) can be provided for plotting on x-axis. |
| identify | If TRUE, the identify() function is called internally that allows to add configuration labels by mouse click. |
| what | For unfolding only: Whether row coordinates, column coordinates, or both should be plotted. |
| col.rows | Row colors in unfolding configuration plot. |
| col.columns | Column colors in unfolding configuration plot. |
| col.dhat | Shepard plot only: color specification of the dhats. For row conditional transformations in unfolding a vector of the length of the number of rows should be specified. |
| label.conf.rows | List with arguments for plotting the labels of the row configurations in an unfolding configuration plot (logical value whether to plot labels or not, label position, label color). |
| label.conf.columns | List with arguments for plotting the labels of the columns configurations in an unfolding configuration plot (logical value whether to plot labels or not, label position, label color). |
| col.hist | Color of the borders of the histogram. |
| plot.array | Array arrangements of plots for individual difference models (see details). |
| ... | Further plot arguments passed: see plot for detailed information. |

## Details

mds() and smacofSym() create an object of class "smacof", unfolding(), prefscal(), and smacofRect() produce "smacofR", and smacofIndDiff() generates "smacofID".

Plot description:

- Configuration plot (plot.type = "confplot"): Plots the MDS configuration.

- Residual plot (plot.type = "resplot"): Plots the disparities (d-hats) distances against the fitted distances.

- Shepard diagram (`plot.type = "Shepard"`): Diagram with the observed dissimilarities against the fitted distances including (isotonic) regression line.

- Stress decomposition plot (`plot.type = "stressplot"`): Plots the stress contribution in of each observation. Note that it rescales the stress-per-point (SPP) from the corresponding smacof function to percentages (sum is 100). The higher the contribution, the worse the fit.

- Bubble plot (`plot.type = "bubbleplot"`, not available for rectangular SMACOF): Combines the configuration plot with the point stress contribution. The larger the bubbles, the worse the fit.

- Histogram (`plot.type = "histogram"`: gives a weighted histogram of the dissimilarities. For optional arguments, see `wtd.hist`.

For `smacofIndDiff()` the residual plot, Shepard diagram, and stress plot are based on the sum of the residuals across individuals/ways. The configuration plot represents the group stimulus space (i.e., joint configuration). If `plot.array` is not specified, it produces a Shepard plot of the distances summed across subjects, if `plot.array = 0` it produces a sqrt(nsubjects) times sqrt(nsubjects) array of graph panels, if `plot.array = 3` it produces 3x3 arrays of graph panels, if `plot.array = c(2,3)` it produces 2x3 arrays of graph panels, and if `plot.array = c(3,2,5)` produces 3x2 arrays of panels (only the first two values are used).

## See Also

`plot.procr`

## Examples

```
## 2D plots for simple MDS
data(trading)
res <- mds(trading)
plot(res, plot.type = "confplot")
plot(res, plot.type = "confplot", label.conf = list(pos = 5)) ## avoid overlapping labels
plot(res, plot.type = "Shepard")
plot(res, plot.type = "stressplot")
plot(res, plot.type = "resplot")
plot(res, plot.type = "bubbleplot")
plot(res, plot.type = "histogram")

## Add convex hulls to configuration plot
r <- cor(PVQ40, use = "pairwise.complete.obs")
diss <- sim2diss(r, method = "corr")
res <- mds(delta = diss, type = "ordinal")
codes <- substring(colnames(PVQ40), 1, 2)  ## supplementary variable
plot(res, hull.conf = list(hull = TRUE, ind = codes, col = "coral1", lwd = 2))

## Shepard plots
ekmanD <- sim2diss(ekman)
fit1 <- mds(ekmanD, type = "ordinal")
plot(fit1, plot.type = "Shepard")
plot(fit1, plot.type = "Shepard", shepard.x = ekman)  ## original data on x-axis

## Joint configuration plot and row/column stressplots for unfolding
data(breakfast)
```

```
res <- unfolding(breakfast)
plot(res, plot.type = "confplot")
plot(res, plot.type = "stressplot")
```

---

`Procrustes` *Procrustean Similarity Transformations*

---

## Description

Solves the Procrustean problem of fitting one (MDS) configuration (testee) to another (target) MDS configuration.

## Usage

```
Procrustes(X, Y)

## S3 method for class 'procr'
plot(x, plot.type = "jointplot", plot.dim = c(1,2), main, xlab, ylab,
                     xlim, ylim, asp = 1, pch = 20, col.X = "cadetblue",
                     col.Y = "gray", col.Yhat = "coral1",
                     label.conf = list(label = TRUE, pos = 3, cex = 0.8),
                     arrows = TRUE, length = 0.10,
                     legend = list(plot = TRUE, labels = c("Target", "Testee"),
                                   pos = "bottomright"), ...)
```

## Arguments

| | |
|---|---|
| `X` | Target configuration |
| `Y` | Testee configuration |
| `x` | Object of class `procr` |
| `plot.type` | Either `"jointplot"` or `"transplot"` |
| `plot.dim` | Vector with dimensions to be plotted. |
| `main` | Plot title. |
| `xlab` | Label of x-axis. |
| `ylab` | Label of y-axis. |
| `xlim` | Scale x-axis. |
| `ylim` | Scale y-axis. |
| `pch` | Plot symbol. |
| `asp` | Aspect ratio. |
| `col.X` | Color target configuration. |
| `col.Y` | Color testee configuration. |
| `col.Yhat` | Color transformed configuration. |

| | |
|---|---|
| label.conf | List with arguments for plotting the labels of the configurations in a configuration plot (logical value whether to plot labels or not, label position, label color). |
| length | length of the edges of the arrow head (in inches). |
| arrows | For "transplot" only, whether arrows should be plotted or not. |
| legend | List with arguments for plotting the legend. |
| ... | Additional plot arguments. |

## Details

Y is going to be modified by finding an optimal dilation factor, an optimal translation and rotation for Y such that it is a similar as possible to X. X remains untouched.

## Value

Returns an object of class procr with:

| | |
|---|---|
| X | Input target configuration |
| Y | Input testee configuration |
| Yhat | Procrustes transformed (fitted) configuration |
| translation | Translation vector |
| dilation | Dilation factor |
| rotation | Rotation-reflection matrix |
| confdistX | Configuration distances X |
| confdistY | Configuration distances Y |
| confdistYhat | Configuration distances of fitted configuration |
| congcoef | Congruence coefficient |
| aliencoef | Alienation coefficient |
| pairdist | Pairwise object distances (sorted) |

## References

Borg, I., & Groenen, P. J. F. (2005). Modern Multidimensional Scaling (2nd ed.). Springer.

## Examples

```
## artificial example:
X <- matrix(c(1, -1, -1, 1, 2, 2, -2, -2), ncol = 2)
Y <- matrix(c(0.07, 0.93, 1.93, 1.07, 2.62, 3.12, 1.38, 0.88), ncol = 2)
op <- par(mfrow = c(1,2))
plot(X[,1], X[,2], xlim = c(-3, 3), ylim = c(-2, 3.5), asp = 1, xlab = "", ylab = "")
rect(-1, -2, 1, 2)
points(Y[,1], Y[,2], xlim = c(-3, 3), col = "gray")
polygon(Y[,1], Y[,2], border = "gray")
fitp <- Procrustes(X, Y)
plot(fitp$Yhat[,1], fitp$Yhat[,2], col = "red", xlim = c(-3, 3), ylim = c(-2, 3.5),
     asp = 1, xlab = "", ylab = "")
```

```
polygon(fitp$Yhat[,1], fitp$Yhat[,2], border = "red")
par(op)

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

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

## Configuration plots; Procrustes plots.
plot(fit.east, main = "MDS East Germany")   ## MDS plot East Germany
plot(fit.west, main = "MDS West Germany")   ## MDS plot West Germany

## Procrustes configurations (X and Yhat)
plot(fit.proc, ylim = c(-1, 1),  col.X = "cadetblue", col.Yhat = "brown", pch = 19,
    legend = list(pos = "topleft", labels = c("East Germany", "West Germany")))

## Procrustes transformations (Y and Yhat)
plot(fit.proc, plot.type = "transplot", length = 0.05, ylim = c(-1,1),
     legend = list(pos = "bottomright",
     labels = c("West Germany (untransformed)", "West Germany (transformed)")))
```

---

PVQ40 *Portrait Value Questionnaire*

---

## Description

The PVQ40 (Schwartz et al., 1999) consists of 40 items, each a short portrait of one person. For example, to measure power, the PVQ includes two portraits (male/female versions): It is important to him to be rich. He wants to have a lot of money and expensive things. It is important to him to get respect from others. He wants people to do what he says. Respondents indicate on 6-point bipolar rating scale (1 ... not at all like me, 6 ... very much like me) the degree to which the description also fits himself/herself. Gender and age of the participants are added as attributes.

## Usage

```
data(PVQ40)
```

## Format

PVQ40 data of 151 adults from various states in the USA:

sd1-sd4: self-direction

po1-po3: power

un1-un6: universalism

ac1-ac4: achievement

se1-se5: security

st1-st3: stimulation

co1-co4: conformity

tr1-tr4: tradition

he1-he3: hedonism

be1-be4: benevolence

Age and Gender are added as attributes.

## Details

`PVQ40agg` is an aggregated version of `PVQ40` where the item scores belonging to the same value are averaged. Abbreviations: power (PO), achievement (AC), hedonism (HE), stimulation (ST), self-direction (SD), universalism (UN), benevolence (BE), tradition (TR), conformity (CO), security (SE).

## References

Borg, I., Bardi, A., & Schwartz, S. H. (2017). Does the value circle exist within persons or only across persons? Journal of Personality, 85(2), 151-162.

## See Also

`indvalues`

## Examples

```
str(PVQ40)
head(PVQ40)
attr(PVQ40, "Gender")
attr(PVQ40, "Age")
str(PVQ40agg)
```

---

`randomstress` *Stress Calculation for Random Dissimilarities*

---

## Description

Creates random dissimilarity matrices (n objects), fits an MDS, and returns the stress values of each MDS fit.

## Usage

```
randomstress(n, ndim, nrep = 100, type = c("ratio", "interval", "ordinal", "mspline"))
```

### Arguments

| | |
|---|---|
| n | Number of objects |
| ndim | Number of dimensions for MDS |
| nrep | Number of random samples |
| type | MDS type |

### Details

The random dissimilarities are drawn from a U(0,1) distribution.

### Value

Returns a vector with stress values.

### References

Spence I., Ogilvie, J.C. (1973). A table of expected stress values for random rankings in nonmetric multidimensional scaling. Multivariate Behavioral Research, 8, 511-517.

### Examples

```
## 8 objects, 2 dimensions, interval MDS (50 replications)
stressvec <- randomstress(n = 8, ndim = 2, nrep = 50, type = "interval")
mean(stressvec)
```

---

## rectangles *Rectangles*

---

### Description

These data are based on an experiment by Borg and Leutner (1983). They constructed rectangles on the basis of the grid design (see `rect_constr`). Each point in this grid defines a rectangle. Rectangle 16, for example, had a width of 4.25 cm and a height of 1.25 cm; rectangle 4 was 3.00 cm wide and 2.75 cm tall. A total of 21 persons rated (twice) the similarity of each pair of these 16 rectangles (on a 10-point scale ranging from from 0 = equal/identical to 9 = very different) The means of these ratings over persons and replications are given in `rectangles`. A second dataset (`rectangles2`) is constructed based on area and shape of the rectangles.

### Usage

```
data(rectangles)
data(rect_constr)
data(rectangles2)
```

### Format

The rectangles are object of class `dist`, the constraints are given as matrix

## References

Borg, I., & Leutner, D. (1983). Dimensional models for the perception of rectangles. Perception and Psychophysics, 34, 257-269.

Borg, I., Groenen, P. J. F., & Mair, P. (2017). Applied Multidimensional Scaling and Unfolding. New York, Springer.

## Examples

```
rectangles
rect_constr
rectangles2
```

---

`residuals.smacof` *Residuals*

---

## Description

Computes the residuals by subtracting the configuration dissimilarites from the observed dissimilarities.

## Usage

```
## S3 method for class 'smacof'
residuals(object, ...)
## S3 method for class 'smacofR'
residuals(object, ...)
## S3 method for class 'smacofID'
residuals(object, ...)
```

## Arguments

| | |
|---|---|
| `object` | Object of class `smacof`, `smacofR` (rectangular), or `smacofID` (individual differences) |
| `...` | Ignored |

## Examples

```
res <- mds(kinshipdelta)
residuals(res)
```

## RockHard *RockHard Ratings*

### Description

Data from RockHard Magazine: In this German Heavy Metal Magazine around 50 records are rated by the writers on a scale from (0 ... worst to 10 ... best) each month. The dataset contains all ratings from 2013.

### Usage

```
data(RockHard)
```

### Format

Data frame with raters in the colums, bands/albums in the rows.

### References

Mair, P., de Leeuw, J., & Wurzer, M. (2015). Multidimensional Unfolding. Wiley StatsRef: Statistics Reference Online. New York: Wiley.

### Examples

```
head(RockHard)
```

## sim2diss *Converts similarites to dissimilarities*

### Description

Utility function for converting similarities into dissimilarities. Different methods are provided.

### Usage

```
sim2diss(s, method = "corr", to.dist = FALSE)
```

### Arguments

| | |
|---|---|
| s | Similarity matrix (not necessarily symmetric, nor square) |
| method | Various methods for converting similarities into dissimilarities: "corr", "reverse", "reciprocal", "ranks", "exp", "Gaussian", "cooccurrence", "gravity", "confusion", "transition", "membership", "probability", or an integer value from which the similarity is subtracted |
| to.dist | If TRUE, object of class dist is produced |

## Details

The conversion formulas for the various methods can be found in the package vignette.

## Value

Returns dissimiarities either as matrix or as dist object.

## Examples

```
## Convert crimes data (correlations)
data(crimes)
crimeD <- sim2diss(crimes, method = "corr", to.dist = TRUE)

## Convert Wish data (similarities) by subtracting from 7
data(wish)
wishD <- sim2diss(wish, method = 7, to.dist = TRUE)

## Convert Ekman data (similarities) into dissimilarities
data(ekman)
ekmanD <- sim2diss(ekman, method = "confusion", to.dist = TRUE)

## Convert album ratings (rectangular similarities) by reversing the ratings
data(RockHard)
rockD1 <- sim2diss(RockHard[,5:18], method = "reverse")
```

---

# smacofConstraint *SMACOF Constraint*

---

## Description

SMACOF with internal constraints on the configurations.

## Usage

```
smacofConstraint(delta, constraint = "unrestricted", external, ndim = 2,
            type = c("ratio", "interval", "ordinal", "mspline"), weightmat = NULL,
                 init = NULL, ties = "primary", verbose = FALSE, modulus = 1,
               itmax = 1000, eps = 1e-6, spline.intKnots = 4, spline.degree = 2,
                 constraint.type = c("ratio", "interval", "ordinal", "spline",
                 "mspline"), constraint.ties = "primary",
                 constraint.spline.intKnots = 2, constraint.spline.degree = 2)
```

## Arguments

| | |
|---|---|
| delta | Either a symmetric dissimilarity matrix or an object of class "dist" |
| constraint | Type of constraint: "unrestricted", "unique", "diagonal", or a user-specified function (see details) |
| external | Data frame or matrix with external covariates, or list for simplex and circumplex (see details) |
| ndim | Number of dimensions |
| type | MDS type: "interval", "ratio", "ordinal" (nonmetric MDS), or "mspline" |
| weightmat | Optional matrix with dissimilarity weights |
| init | Optional matrix with starting values for configurations. If NULL random starts are used (see details). |
| ties | Tie specification for non-metric MDS only: "primary", "secondary", or "tertiary" |
| verbose | If TRUE, intermediate stress is printed out |
| modulus | Number of smacof iterations per monotone regression call |
| itmax | Maximum number of iterations |
| eps | Convergence criterion |
| spline.degree | Degree of the spline for "mspline" MDS type |
| spline.intKnots | Number of interior knots of the spline for "mspline" MDS type |
| constraint.type | Transformation for external covariates: "ratio", "interval", "ordinal", "spline", or "mspline") |
| constraint.ties | Tie specification for external covariates with constraint.type = "ordinal": "primary", "secondary", or "tertiary" |
| constraint.spline.intKnots | Number of interior knots for external covariates with constraint.type = "spline" or "mspline" |
| constraint.spline.degree | Degree of the spline for external covariates with constraint.type = "spline" or "mspline" |

## Details

The argument external is mandatory to specify and requires a data frame (or matrix) of dimension (n x q). Alternatively, for simplex fitting the user can specify a list of the following structure: external = list("simplex",dim2) with dim2 denoting the dimension of the simplex with dim2 < n. For a circumplex fitting, the list has to be of the following form: external = list("circumplex",dim2,k1,k2) with $1 \leq k1 \leq k2 \leq n$ (see also examples section). k1 and k2 denote the circumplex width.

In constraint smacof, the configuration matrix $X$ is subject to a constraint based on the external scales (predictors $Z$ specified using external) of the following linear form: $X = ZC$. The type of constraint in $C$ can be specified using the constraint argument. We provide the following standard setting:

For `constraint = "unrestricted"`, $C$ is unrestricted. Note that `"linear"` still works as well for backward compatibility.

The same for `constraint = "diagonal"` where $X$ needs to be of dimension $(nxq)$ where $q$ is the number of columns of the external scale matrix (and thus number of dimensions). Here, $C$ is restricted to be diagonal.

For `constraint = "unrestricted"` or `"diagonal"`, the external covariates $Z$ can be optimally transformed as specified by `constraint.type`. Choosing the number of covariates equal to the number of dimensions together with `constraint.type = "ordinal"`, `constraint.ties = "primary"` will effectively restrict the configuration to parallel regions defined by the categories of the covariates. Note that missing values of the covariates are estimated by the model.

For `constraint = "unique"` we get the Bentler-Weeks uniqueness model. Hence $X$ is of dimension $(nx(n+p))$. This implies that we fit a certain number of dimensions p and, in addition we extract n additional dimensions where each object is scored on a separate dimension. More technical details can be found in the corresponding JSS article (reference see below).

In addition, the user can specify his own constraint function with the following arguments: configuration matrix with starting values (`init`) (mandatory in this case), matrix $V$ (`weightmat`; based on the weight matrix, see package vignette), external scale matrix (`external`). The function must return a matrix of resulting configurations.

If no starting configuration is provided, a random starting solution is used. In most applications, this is not a good idea in order to find a well fitting model. The user can fit an exploratory MDS using `mds()` first, and use the resulting configurations as starting configuration for `smacofConstraint()`. Alternatively, if the user has starting configurations determined by some underlying theory, they can be used as well.

## Value

| | |
|---|---|
| `delta` | Observed dissimilarities |
| `obsdiss` | Observed dissimilarities, normalized |
| `confdist` | Configuration dissimilarities |
| `conf` | Matrix of final configurations |
| `C` | Matrix with restrictions |
| `stress` | Stress-1 value |
| `spp` | Stress per point |
| `resmat` | Matrix with squared residuals |
| `rss` | Residual sum-of-squares |
| `weightmat` | Weight matrix |
| `ndim` | Number of dimensions |
| `extvars` | List for each external covariate with a list of class `"optscal"` |
| `init` | Starting configuration |
| `model` | Type of smacof model |
| `niter` | Number of iterations |
| `nobj` | Number of objects |

## References

De Leeuw, J. & Mair, P. (2009). Multidimensional scaling using majorization: The R package smacof. Journal of Statistical Software, 31(3), 1-30, https://www.jstatsoft.org/v31/i03/

Mair, P., Groenen, P. J. F., & De Leeuw, J. (2020). More on multidimensional scaling and unfolding in R: smacof version 2. Journal of Statistical Software, Forthcoming.

De Leeuw, J., & Heiser, W. (1980). Multidimensional scaling with restrictions on the configurations. In P. R. Krishnaiah (eds.), Multivariate Analysis V, pp. 501-522. North-Holland.

Borg, I., & Lingoes, J. C. (1980). A model and algorithm for multidimensional scaling with external constraints on the distances. Psychometrika, 45, 25-38.

## See Also

smacofSym, smacofRect, smacofIndDiff, smacofSphere

## Examples

```
## theoretical grid restrictions (rectangles; keep covariate ties tied)
fit.rect1 <- mds(rectangles, type = "ordinal", init = rect_constr)
fit.rect2 <- smacofConstraint(rectangles, type = "ordinal", ties = "secondary",
                      constraint = "diagonal", init = fit.rect1$conf,
                      external = rect_constr, constraint.type = "ordinal")
plot(fit.rect2)

## regional restrictions morse code data (signal length, strength)
fitMorse1 <- mds(morse, type = "ordinal")
fitMorse1
fitMorse2 <- smacofConstraint(morse, type = "ordinal", constraint = "unrestricted",
                              external = morsescales[,2:3],
                              constraint.type = "ordinal",
                              init = fitMorse1$conf)
fitMorse2
plot(fitMorse2)

## facial expression data I (axial restriction, C diagonal)
Delta <- FaceExp
attr(Delta, "Labels") <- NULL
fitFace <- mds(Delta, type = "ordinal")   ## starting solution
Z <- FaceScale[, c(1,3)]                  ## external variables
fitFaceC1 <- smacofConstraint(Delta, type = "ordinal",
  constraint = "diagonal", external = Z, constraint.type = "ordinal",
  init = fitFace$conf)
fitFaceC1$C
plot(fitFaceC1, xlab = "Pleasant-Unpleasant", ylab = "Tension-Sleep",
  main = "Face Expression (Diagonal Restriction)")

## facial expression data II (C unrestricted)
fitFaceC3 <- smacofConstraint(Delta, type = "ordinal",
  constraint = "unrestricted", external = Z, constraint.type = "ordinal",
```

```
  init = fitFace$conf)
fitFaceC3$C
plot(fitFaceC3, main = "Face Expression (C Unrestricted, Ordinal Transformation)")
```

---

`smacofIndDiff` *SMACOF for Individual Differences*

---

## Description

Performs smacof for individual differences also known as Three-Way smacof on a list of dissimilarity matrices. Various restrictions decompositions and restrictions on the weight matrix are provided. The most prominent models are INDSCAL and IDIOSCAL.

## Usage

```
smacofIndDiff(delta, ndim = 2, type = c("ratio", "interval", "ordinal", "mspline"),
              constraint = c("indscal", "idioscal", "identity"),
              weightmat = NULL, init = "torgerson", ties = "primary",
              verbose = FALSE, modulus = 1, itmax = 1000, eps = 1e-6,
              spline.degree = 2, spline.intKnots = 2)

indscal(delta, ndim = 2, type = c("ratio", "interval", "ordinal", "mspline"),
        weightmat = NULL, init = "torgerson", ties = "primary",
        verbose = FALSE, modulus = 1, itmax = 1000, eps = 1e-6,
        spline.degree = 2, spline.intKnots = 2)

idioscal(delta, ndim = 2, type = c("ratio", "interval", "ordinal", "mspline"),
         weightmat = NULL, init = "torgerson", ties = "primary",
         verbose = FALSE, modulus = 1, itmax = 1000, eps = 1e-6,
         spline.degree = 2, spline.intKnots = 2)
```

## Arguments

| | |
|---|---|
| `delta` | A list of dissimilarity matrices or a list objects of class `dist` |
| `ndim` | Number of dimensions |
| `type` | MDS type: `"interval"`, `"ratio"`, `"ordinal"` (nonmetric MDS), or `"mspline"` |
| `weightmat` | Optional matrix with dissimilarity weights |
| `init` | Matrix with starting values for configurations (optional) |
| `ties` | Tie specification for non-metric MDS |
| `constraint` | Either `"indscal"`, `"idioscal"`, or `"identity"` (see details) |
| `verbose` | If `TRUE`, intermediate stress is printed out |
| `modulus` | Number of smacof iterations per monotone regression call |
| `itmax` | Maximum number of iterations |

| | |
|---|---|
| eps | Convergence criterion |
| spline.degree | Degree of the spline for "mspline" MDS type |
| spline.intKnots | Number of interior knots of the spline for "mspline" MDS type |

## Details

If the constraint is "indscal", INDSCAL is performed with configuration weight matrices restricted to be diagonal. indscal() is a corresponding wrapper function that can be used instead of smacofIndDiff() with "indscal" constraints.

IDIOSCAL can be computed using the "idioscal" argument. The weight matrices are then unconstrained. idioscal() is a corresponding wrapper function that can be used instead of smacofIndDiff() with "idioscal" constraints.

Addtional weight restrictions can be imposed with "identity" which restricts the configurations across individuals/replications/ways to be equal.

## Value

| | |
|---|---|
| delta | Observed dissimilarities |
| obsdiss | List of observed dissimilarities, normalized |
| confdist | List of configuration dissimilarities |
| conf | List of matrices of final configurations |
| gspace | Joint configuration aka group stimulus space |
| cweights | Individual weights |
| stress | Stress-1 value |
| resmat | Matrix with squared residuals |
| rss | Residual sum-of-squares |
| spp | Stress per point (in percent) |
| spps | Stress per point per subject (in percent, conditional on subject) |
| sps | Stress per subject (in percent) |
| ndim | Number of dimensions |
| model | Type of smacof model |
| niter | Number of iterations |
| nobj | Number of objects |

## Author(s)

Jan de Leeuw and Patrick Mair

## References

De Leeuw, J., & Mair, P. (2009). Multidimensional scaling using majorization: The R package smacof. Journal of Statistical Software, 31(3), 1-30, https://www.jstatsoft.org/v31/i03/

## See Also

smacofConstraint, smacofSym, smacofRect, smacofSphere

## Examples

```
## Example 1: rectangle perception data
res.diag <- indscal(perception, type = "ordinal")        ## INDSCAL
res.diag$cweights
plot(res.diag)
plot(res.diag, type = "p", pch = 25, col = 4, label.conf = list(label = TRUE, pos = 3, col = 4))

res.idio <- idioscal(perception, type = "ordinal")  ## IDIOSCAL
Wk <- res.idio$cweights
G <- res.idio$gspace
G
G

## identity restricted weights
res.id <- smacofIndDiff(perception, type = "ordinal", constraint = "identity")
summary(res.id)
res.id$cweights
plot(res.id)
plot(res.id, type = "p", pch = 25, col = 4, label.conf = list(label = TRUE, pos = 3, col = 4))


## Example 2: Helm's color data
res.helm <- indscal(helm, type = "interval")
plot(res.helm, plot.type = "confplot")
barplot(sort(res.helm$sps, decreasing = TRUE), main = "Stress per Subject", cex.names = 0.8)
plot(res.helm, plot.type = "bubbleplot")
plot(res.helm, plot.type = "stressplot")
plot(res.helm, plot.type = "Shepard")

## idioscal and indscal with random starting configuration:
set.seed(123)
startconf <- matrix(rnorm(20), 10, 2)
idioscal(helm, init = startconf, type = "interval")
indscal(helm, init = startconf, type = "interval")
```

---

smacofSphere *Spherical SMACOF*

---

## Description

Dual and primal approach for spherical SMACOF.

## Usage

```
smacofSphere(delta, ndim = 2, type = c("ratio", "interval", "ordinal","mspline"),
             algorithm = c("dual", "primal"), weightmat = NULL,
           init = "torgerson", ties = "primary", verbose = FALSE, penalty = 100,
             relax = FALSE, modulus = 1, itmax = 1000, eps = 1e-6,
             spline.degree = 2, spline.intKnots = 2)
```

## Arguments

| | |
|---|---|
| delta | Either a symmetric dissimilarity matrix or an object of class dist |
| ndim | Number of dimensions |
| type | MDS type: "interval", "ratio", or "ordinal" (nonmetric MDS) |
| algorithm | Algorithm type (see details) |
| weightmat | Optional matrix with dissimilarity weights |
| init | Either "torgerson" (classical scaling starting solution), "random" (random configuration), or a user-defined matrix |
| ties | Tie specification for non-metric MDS only |
| verbose | If TRUE, intermediate stress is printed out |
| penalty | Penalty parameter for dual algorithm (larger 0), see details |
| relax | If TRUE, block relaxation is used for majorization (dual algorithm) |
| modulus | Number of smacof iterations per monotone regression call |
| itmax | Maximum number of iterations |
| eps | Convergence criterion |
| spline.degree | Degree of the spline for "mspline" MDS type |
| spline.intKnots | Number of interior knots of the spline for "mspline" MDS type |

## Details

For large scale problems it is suggested to use the dual algorithm. Using the penalty parameter (dual algorithm), the user allow for slight point deviations from the circle (the higher the penalty, the stricter the algorithm is in terms of placing points in the sphere, see examples section below).

## Value

| | |
|---|---|
| delta | Observed dissimilarities |
| obsdiss | Observed dissimilarities, normalized |
| obsdiss1 | Dual SMACOF: Observed dissimilarities |
| obsdiss2 | Dual SMACOF: Restriction matrix |
| confdist | Configuration dissimilarities |
| conf | Matrix with fitted configurations |
| spp | Stress per point |

| | |
|---|---|
| resmat | Matrix with squared residuals |
| rss | Residual sum-of-squares |
| stress | Stress-1 value |
| init | Starting configurations |
| ndim | Number of dimensions |
| dummyvec | Dummy vector of restriction matrix |
| model | Type of smacof model |
| niter | Number of iterations |
| nobj | Number of objects |

## Author(s)

Jan de Leeuw and Patrick Mair

## References

De Leeuw, J. & Mair, P. (2009). Multidimensional scaling using majorization: The R package smacof. Journal of Statistical Software, 31(3), 1-30, https://www.jstatsoft.org/v31/i03/

## See Also

smacofRect, smacofIndDiff, smacofSym,smacofConstraint

## Examples

```
## spherical SMACOF solution for trading data
## dual algorithm
res <- smacofSphere(trading, type = "ordinal")
res
plot(res)

## lower penalty
res <- smacofSphere(trading, penalty = 20, type = "ordinal")
res
plot(res)

## primal algorithm, interval
res <- smacofSphere(trading, type = "interval", algorithm = "primal")
res
```

smacofSym *Symmetric smacof*

## Description

Multidimensional scaling on a symmetric dissimilarity matrix using SMACOF.

## Usage

```
smacofSym(delta, ndim = 2, type = c("ratio", "interval", "ordinal", "mspline"),
         weightmat = NULL, init = "torgerson", ties = "primary", verbose = FALSE,
          relax = FALSE, modulus = 1, itmax = 1000, eps = 1e-06,
          spline.degree = 2, spline.intKnots = 2)

mds(delta, ndim = 2, type = c("ratio", "interval", "ordinal", "mspline"),
    weightmat = NULL, init = "torgerson", ties = "primary", verbose = FALSE,
    relax = FALSE, modulus = 1, itmax = 1000, eps = 1e-06,
    spline.degree = 2, spline.intKnots = 2)
```

## Arguments

| | |
|---|---|
| delta | Either a symmetric dissimilarity matrix or an object of class "dist" |
| ndim | Number of dimensions |
| weightmat | Optional matrix with dissimilarity weights |
| init | Either "torgerson" (classical scaling starting solution), "random" (random configuration), or a user-defined matrix |
| type | MDS type: "interval", "ratio", "ordinal" (nonmetric MDS), or "mspline" |
| ties | Tie specification (ordinal MDS only): "primary", "secondary", or "tertiary" |
| verbose | If TRUE, intermediate stress is printed out |
| relax | If TRUE, block relaxation is used for majorization |
| modulus | Number of smacof iterations per monotone regression call |
| itmax | Maximum number of iterations |
| eps | Convergence criterion |
| spline.degree | Degree of the spline for "mspline" MDS type |
| spline.intKnots | Number of interior knots of the spline for "mspline" MDS type |

## Details

This is the simplest MDS-SMACOF version of the package. It solves the stress target function for symmetric dissimiliarities by means of the majorization approach (SMACOF) and reports the Stress-1 value (normalized). The main output are the coordinates in the low-dimensional space (configurations; conf).

The function `mds()` is a wrapper function and can be used instead of `smacofSym()`

This function allows for fitting three basic types of MDS: ratio MDS (default), interval MDS (polynomial transformation), and ordinal MDS (aka nonmetric MDS). It also returns the point stress, i.e. the larger the contribution of a point to the total stress, the worse the fit (see also `plot.smacof`.

## Value

| | |
|---|---|
| `delta` | Observed dissimilarities, not normalized |
| `dhat` | Disparities (transformed proximities, approximated distances, d-hats) |
| `confdist` | Configuration distances |
| `conf` | Matrix of fitted configurations |
| `stress` | Stress-1 value |
| `spp` | Stress per point (stress contribution in percentages) |
| `resmat` | Matrix with squared residuals |
| `rss` | Residual sum-of-squares |
| `weightmat` | Weight matrix |
| `ndim` | Number of dimensions |
| `init` | Starting configuration |
| `model` | Name of smacof model |
| `niter` | Number of iterations |
| `nobj` | Number of objects |
| `type` | Type of MDS model |

## Author(s)

Jan de Leeuw and Patrick Mair

## References

De Leeuw, J. & Mair, P. (2009). Multidimensional scaling using majorization: The R package smacof. Journal of Statistical Software, 31(3), 1-30, https://www.jstatsoft.org/v31/i03/

Borg, I., & Groenen, P. J. F. (2005). Modern Multidimensional Scaling (2nd ed.). Springer.

Borg, I., Groenen, P. J. F., & Mair, P. (2013). Applied Multidimensional Scaling. Springer.

## See Also

`smacofConstraint`, `smacofRect`, `smacofIndDiff`, `smacofSphere`, `plot.smacof`

## Examples

```
## simple SMACOF solution (interval MDS) for kinship data
res <- mds(kinshipdelta, type = "interval")
res
summary(res)
plot(res)
plot(res, type = "p", label.conf = list(label = TRUE, col = "darkgray"), pch = 25, col = "red")

## ratio MDS, random starts
set.seed(123)
res <- mds(kinshipdelta, init = "random")
res

## 3D ordinal SMACOF solution for trading data (secondary approach to ties)
data(trading)
res <- mds(trading, ndim = 3, type = "ordinal", ties = "secondary")
res

## spline MDS
delta <- sim2diss(cor(PVQ40agg))
res <- mds(delta, type = "mspline", spline.degree = 3, spline.intKnots = 4)
res
plot(res, "Shepard")
```

---

`stardist` *Distances among stars in zodiac signs*

---

## Description

A distance matrix for the 10 brightest stars in each of the 12 zodiac signs was computed. Astronomers measure the projected positions of objects on the celestial sphere in two angles, i.e. right ascension $\alpha$ and declination $\delta$. For every zodiac sign, the projected distances on the sky between individual stars $S_i$ and $S_j$ have been calculated in decimal degrees by means of the Pythagorean theorem

$$d_{i,j} = \sqrt{(\alpha_i - \alpha_j)\,2 + (\delta_i - \delta_j)\,2}$$

assuming planar geometry. Since the zodiac signs are relatively small compared to the whole celestial sphere and the computation is only done for illustrative purposes, such a simplified assumption is appropriate.

## Usage

```
data(stardist)
```

## Format

A dist object contaning the star distances.

## Note

Thanks to Paul Eigenthaler, Department of Astronomy, University of Vienna for calculating the distances.

## Examples

```
stardist
```

---

`stress0` *Zero-Iterations Stress*

---

## Description

Computes the stress for 0 iterations based on a starting configuration provided by the user.

## Usage

```
stress0(delta, init, type = c("interval", "ratio", "ordinal", "mspline"),
      weightmat = NULL, ties = "primary", spline.degree = 2, spline.intKnots = 2)
```

## Arguments

| | |
|---|---|
| `delta` | Either a symmetric dissimilarity matrix or an object of class `"dist"` |
| `init` | An initial configuration provided by the user |
| `weightmat` | Optional matrix with dissimilarity weights |
| `type` | MDS type: `"interval"`, `"ratio"`, `"ordinal"` (nonmetric MDS), or `"mspline"` |
| `ties` | Tie specification (ordinal MDS only): `"primary"`, `"secondary"`, or `"tertiary"` |
| `spline.degree` | Degree of the spline for `"mspline"` MDS type |
| `spline.intKnots` | Number of interior knots of the spline for `"mspline"` MDS type |

## Details

Computes stress-1 for a particular starting configuration the user needs to provide. It can also be helpful if the user wants to move some points in a particular configuration such that it fits some theoretical expectations.

## Value

Stress-1 value

## See Also

`mds`

## Examples

```
## rectangle starting solution
rect_constr
stress0(rectangles, init = rect_constr)

## torgerson starting solution
tstart <- torgerson(rectangles)
stress0(rectangles, init = tstart)
```

---

summary.smacofB *S3 methods for smacof*

---

## Description

Print and summary methods for objects of class smacofB, smacofR (rectangular), and smacofID (individual differences).

## Usage

```
## S3 method for class 'smacofB'
summary(object, ...)
## S3 method for class 'smacofB'
print(x, ...)
## S3 method for class 'smacofR'
summary(object, ...)
## S3 method for class 'smacofR'
print(x, ...)
## S3 method for class 'smacofID'
summary(object, ...)
## S3 method for class 'smacofID'
print(x, ...)
```

## Arguments

| | |
|---|---|
| object | Object of class smacofB, smacofR, smacofID |
| x | Object of class smacofB, smacofR, smacofID |
| ... | Ignored |

## Examples

```
data(kinshipdelta)
res <- smacofSym(kinshipdelta)
res
summary(res)
```

svm_mdsplot *Support Vector Machine MDS*

## Description

Plots 2D MDS configuration including facets as determined by an SVM.

## Usage

```
svm_mdsplot(mds_object, svm_object, class, legend1 = TRUE, legend2 = TRUE,
            inset = c(-0.2, 0.5), plot.dim = c(1,2), by = 0.01,
            main, xlab, ylab, xlim, ylim, ...)
```

## Arguments

| | |
|---|---|
| mds_object | Object of class "smacofB", i.e., an MDS solution from smacofSym() or smacofConstraint. |
| svm_object | Object of class "svm", i.e., an SVM solution from svm or tune.svm |
| class | Vector of class assignments (facets) for each object. |
| legend1 | If TRUE, facet legend is added. |
| legend2 | If TRUE, class legend is added. |
| inset | Inset distance from the margins for both legends as a fraction of the plot region when legend is placed by keyword. |
| plot.dim | Vector with dimensions to be plotted. |
| by | Scaling factor for resolution (the smaller, the higher the resolution). |
| main | Plot title. |
| xlab | Label of x-axis. |
| ylab | Label of y-axis. |
| xlim | Scale x-axis. |
| ylim | Scale y-axis. |
| ... | Further plot arguments passed: see image for detailed information. |

## Details

Using the SVM implementation of e1071 one can determine facets in an MDS configuration based on an SVM fit. This function plots the resulting facets on top of the 2D MDS configuration. Note that this function is work in progress.

## See Also

svm, tune.svm

## Examples

```
## Guttman intelligence data
Delta <- sim2diss(Guttman1965[[1]])
class <- Guttman1965[[2]]

## ordinal MDS fit
mds_gut <- mds(Delta, ndim = 2, type = "ordinal")
mds_gut
cols <- rainbow_hcl(4)[as.numeric(class)]
plot(mds_gut, col = cols, label.conf = list(col = cols))
legend("bottomright", legend = levels(class), cex = 0.7, col = rainbow_hcl(4), pch = 19)

## radial SVM fit
X <- mds_gut$conf                    ## extract configuration
dat <- data.frame(class = class, X)  ## merge with class vector
costvec <- 2^seq(-4, 4)              ## tuning parameter grid
gamma <- seq(0.01, 0.5, 10)

set.seed(111)
svm_gut <- tune.svm(class ~ D1 + D2, data = dat, kernel = "radial",
                    cross = 10, cost = costvec)$best.model
svm_gut
preds <- predict(svm_gut, data = dat)  ## predicted classes
table(obs = class, pred = preds)       ## confusion matrix

svm_mdsplot(mds_gut, svm_gut, dat$class, inset = c(-0.3, 0.5))
```

---

# symdecomp *Proximity Matrix Decomposition*

---

## Description

Additive decomposition of an asymmetric, square proximity matrix into a symmetric matrix and an skew-symmetric matrix

## Usage

```
symdecomp(P)
```

## Arguments

| | |
|---|---|
| P | Square proximity matrix |

## Details

Performs the decompositon $P = M + N$ (M and N are orthogonal).

## Value

Returns the following matrices:

| | |
|---|---|
| M | symmetric component |
| N | skew-symmetric component |

## References

Borg, I., & Groenen, P. J. F. (2005). Modern Multidimensional Scaling (2nd ed.). Springer.

## Examples

```
P <- matrix(c(92,5,4,8,4,84,38,62,6,37,87,17,13,31,17,88), ncol = 4)
symdecomp(P)
```

---

# torgerson *Torgerson Scaling*

---

## Description

Classical MDS aka Torgerson Scaling

## Usage

```
torgerson(delta, p)
```

## Arguments

| | |
|---|---|
| delta | Dissimilarity matrix |
| p | Number of dimensions |

## Value

Returns an $n \times p$ matrix of configurations

## References

Borg, I., & Groenen, P. J. F. (2005). Modern Multidimensional Scaling (2nd ed.). Springer.

## Examples

```
fit <- torgerson(Guerry)
```

`trading` *Trading data*

## Description

Data from the New Geographical Digest (1986) analyzed in Cox and Cox (2001). For 20 countries their main trading partners were dichotomously scored (1 = trade performed, 0 = trade not performed). Based on this dichotomous matrix the dissimilarities were computed using the Jaccard coefficient.

## Usage

```
data(trading)
```

## Format

Object of class `"dist"` with dissimilarities of the following countries:

Arge: Argentina

Aust: Australia

Braz: Brazil

Cana: Canada

Chin: China

Czec: Czechoslovakia

Egyp: Egypt

E.Ge: East Germany

Fran: France

Hung: Hungary

Indi: India

Ital: Italy

Japa: Japan

N.Ze: New Zealand

Pola: Poland

Swed: Sweden

USA

USSR: Soviet Union

U.K.: United Kingdom

W.Ge: West Germany

## References

Cox, T.F., Cox, M.A.A. (1991). Multidimensional scaling on a sphere. Communications in Statistics: Theory and Methods, 20, 2943-2953.

## Examples

```
data(trading)
```

---

`transform` *Internal Dissimilarity Transformation*

---

## Description

Utility functions for optimal scaling calls (used internally)

## Usage

```
transPrep(x, trans = "ordinals", spline.intKnots = 4, spline.degree = 2, missing = "none")
transform(Target, x, w = rep(1,length(x$x)), normq = 0)
```

## Arguments

| | |
|---|---|
| `Target` | unconstrained vector of target values |
| `x` | object of type optScal |
| `w` | vector non-negative weights |
| `normq` | sum of squares normalization |
| `trans` | type of transformation |
| `spline.intKnots` | interior spline knots |
| `spline.degree` | spline degree |
| `missing` | missing treatment |

---

`unfolding` *Nonmetric unfolding*

---

## Description

Variant of smacof for rectangular matrices (typically ratings, preferences) that allows for nonmetric transformations. Also known as nonmetric unfolding.

## Usage

```
unfolding(delta, ndim = 2, type = c("ratio", "interval", "ordinal", "mspline"),
         conditionality = "unconditional", lambda = 0.5, omega = 1,
         circle = c("none", "row", "column"), weightmat = NULL, init = NULL,
         fixed = c("none", "row", "column"), fixed.coord = NULL,
         ties = c("primary", "secondary"), verbose = FALSE, relax = TRUE,
         itmax = 10000, eps = 1e-6, spline.degree = 2, spline.intKnots = 2,
         parallelize = FALSE)

smacofRect(delta, ndim = 2, type = c("ratio", "interval", "ordinal", "mspline"),
         conditionality = "unconditional", lambda = 0.5, omega = 1,
         circle = c("none", "row", "column"), weightmat = NULL, init = NULL,
         fixed = c("none", "row", "column"), fixed.coord = NULL,
         ties = c("primary", "secondary"), verbose = FALSE, relax = TRUE,
         itmax = 10000, eps = 1e-6, spline.degree = 2, spline.intKnots = 2,
         parallelize = FALSE)

prefscal(delta, ndim = 2, type = c("ratio", "interval", "ordinal", "mspline"),
         conditionality = "unconditional", lambda = 0.5, omega = 1,
         circle = c("none", "row", "column"), weightmat = NULL, init = NULL,
         fixed = c("none", "row", "column"), fixed.coord = NULL,
         ties = c("primary", "secondary"), verbose = FALSE, relax = TRUE,
         itmax = 10000, eps = 1e-6, spline.degree = 2, spline.intKnots = 2,
         parallelize = FALSE)
```

## Arguments

| | |
|---|---|
| `delta` | Data frame or matrix of preferences, ratings, dissimilarities. |
| `ndim` | Number of dimensions. |
| `type` | MDS type: `"interval"`, `"ratio"`, `"ordinal"`, or `"mspline"`. |
| `conditionality` | A single transformations are applied for the entire matrix `"unconditional"`, or for each row separately `"row"`. |
| `lambda` | Penalty strength balancing the loss contribution of stress and the penalty (see details). |
| `omega` | Penalty width determines for what values of the variation coefficient the penalty should become active (see details). |
| `circle` | If `"column"`, the column configurations are restricted to be on a circle, if `"row"`, row configurations are on a circle, if `"none"`, there are no restrictions on row and column configurations |
| `weightmat` | Optional matrix with dissimilarity weights. |
| `init` | Optional list of length two with starting values for the row coordinates (first element) and column coordinates (second element). |
| `fixed` | Do external unfolding by fixing the `row` coordinates, `column` coordinate, or choose `none` (default) to do normal unfolding. Even fixed coordinates are uniformly scaled by a constant to fit the loss function. |

| | |
|---|---|
| fixed.coord | Matrix with fixed coordinates of the appropriate size. |
| ties | Tie specification for ordinal transformations: primary unties the ties and secondary keeps the ties tied. |
| verbose | If TRUE, intermediate stress is printed out. |
| relax | If TRUE, block relaxation is used for majorization after 100 iterations. It tends to reduce the number of iterations by a factor 2. |
| itmax | Maximum number of iterations. |
| eps | Convergence criterion. |
| spline.degree | Degree of the spline for an "mspline" transformation. |
| spline.intKnots | Number of interior knots of the spline for a "mspline" transformation. |
| parallelize | Tries to parallelize the computations when conditionality = "row". |

## Details

Unfolding tries to match a rectangular matrix delta of dissimilarities between row and column objects by Euclidean distances between row and column points. Badness of fit is measured by raw Stress as the sum of squared differences between delta and the Euclidean distances. Instead of dissimilarities optimal transformations (dhats) can be found. The dhats should be a function of the original delta restricted to be "ratio", "interval", "ordinal", or "mspline". These transformations can be the same for the entire matrix (conditionality = "unconditional") of data, or different per row (conditionality = "row"). To avoid a degenerate solution with all dhats and distances equal to 1, the prefscal penalty is is used.

A penalty is added based on the variation coefficient of the dhats (mean dhat divided by the standard deviation of the dhats). The penalty width (omega) weights the penalty and determines from what value of the variation coefficient of the dhats the penalty should become active. The penalty strength (lambda) is needed to ensure that the penalty can be strong enough. Busing et al. (2005) suggest to use $\lambda = 0.5$ and $\omega = 1.0$ (for unconditional solutions $\omega$ can be lowered to a value as low as 0.1).

External unfolding can be done by specifying fixed = "row" or "column" and providing the fixed coordinates in fixed.coord. Then, either the rows or columns are fixed up to a uniform constant.

Creates an object of class smacofR.

## Value

| | |
|---|---|
| obsdiss | Observed dissimilarities, corresponds to delta |
| confdist | Configuration dissimilarities |
| dhat | Matrix with optimal transformation of size delta |
| iord | List of size 1 for matrix conditional and size nrow(delta) for row conditional with the index that orders the dhats. Needed for the Shepard plot |
| conf.row | Matrix of final row configurations |
| conf.col | Matrix of final column configurations |
| stress | Final, normalized stress value |
| pstress | Penalized stress value (the criterion that is minimized) |

| | |
|---|---|
| spp.row | Stress per point, rows |
| spp.col | Stress per point, columns |
| congvec | Vector of congruency coefficients |
| ndim | Number of dimensions |
| model | Type of smacof model |
| niter | Number of iterations |
| nind | Number of individuals (rows) |
| trans | Transformation |
| conditionality | Conditionality of the transformation |
| nobj | Number of objects (columns) |

## Author(s)

Patrick Groenen, Jan de Leeuw and Patrick Mair

## References

de Leeuw, J. & Mair, P. (2009). Multidimensional scaling using majorization: The R package smacof. Journal of Statistical Software, 31(3), 1-30, https://www.jstatsoft.org/v31/i03/

Busing, F. M. T. A., Groenen, P. J. F., & Heiser, W. J. (2005). Avoiding degeneracy in multidimensional unfolding by penalizing on the coefficient of variation. Psychometrika, 70, 71-98.

## See Also

plot.smacof, smacofConstraint, smacofSym, smacofIndDiff, smacofSphere

## Examples

```
## Ratio unfolding
res <- unfolding(breakfast)
res

## various configuration plots
plot(res)
plot(res, type = "p", pch = 25)
plot(res, type = "p", pch = 25, col.columns = 3,
     label.conf.columns = list(label = TRUE, pos = 3, col = 3),
     col.rows = 8, label.conf.rows = list(label = TRUE, pos = 3, col = 8))

## Shepard plot
plot(res, "Shepard")

## Stress decomposition chart
plot(res, "stressplot")

## Not run:
## Ordinal unfolding, row-conditional
## Note that ordinal unfolding may need many iterations (several thousands)
```

```
res <- unfolding(breakfast, type = "ordinal", conditionality = "row", omega = 0.1, itmax = 3000)
res
plot(res, "Shepard")      ## Shepard plot
plot(res)

## End(Not run)
```

---

uniscale *Unidimensional Scaling*

---

## Description

Simple implementation where all dissimilarity permutations are subject to a 1D MDS fit and the one which leads to a minimal stress values is returned.

## Usage

```
uniscale(delta, weightmat = NULL, verbose = TRUE)
```

## Arguments

| | |
|---|---|
| delta | Either a symmetric dissimilarity matrix or an object of class "dist" |
| weightmat | Optional matrix with dissimilarity weights |
| verbose | Permutation printout |

## Value

| | |
|---|---|
| delta | Observed dissimilarities, not normalized |
| confdist | Configuration distances |
| conf | Vector with fitted configurations |
| stress | Stress-1 value |
| weightmat | Weight matrix |
| nobj | Number of objects |
| npermtot | Total number of permutations (factorial) |
| npermscale | Number of accepted permutations (monotonicity check) |

## References

Mair P., De Leeuw J. (2015). Unidimensional scaling. In Wiley StatsRef: Statistics Reference Online, Wiley, New York.

## See Also

mds

## Examples

```
## unidimensional scaling of Plato's 7 works
PlatoD <- dist(t(Plato7))
fit.uni <- uniscale(PlatoD)
fit.uni
plot(fit.uni)
```

---

VaziriXu *Visual Object Representations*

---

## Description

Contains two similarity matrices related to an experiment on visual object represenations.

similarities (correlations) of 54 OCP (see O'Reilly, Chatman, and Caldwell, 1991) items. The last three columns contain the facet assigned by Bilsky and Jehn (2002) as well as the external variables for regional restrictions.

## Usage

```
data(VaziriXu)
```

## Format

List of two similarity matrices from two experimental conditions: artificial and real object categories.

V1-V4: early visual areas

LO: lateral occipital regions

VOT: ventral occipitotemporal regions

V3A, V3B, IPS0-IPS4: topographic regions along the intraparietal sulcus

Inferior IPS: inferior intraparietal sulcus

Superior IPS: superior intraparietal sulcus

## References

Vaziri-Pashkam M, Xu Y (2018). An information-driven two-pathway characterization of occipito-temporal and posterior parietal visual object representations. Cerebral Cortex, forthcoming.

## Examples

```
vx1 <- sim2diss(VaziriXu[[1]])
vx2 <- sim2diss(VaziriXu[[2]])
```

vmu *Vector Model of Unfolding*

## Description

Computes the metric vector model of unfolding (VMU) on rectangular input data (preferences, ratings) with the individuals (rows) represented as vectors in the biplot. There is also the option to fix the column coordinates.

## Usage

```
vmu(delta, ndim = 2, center = TRUE, scale = FALSE, col.coord = NULL)

## S3 method for class 'vmu'
plot(x, ...)
```

## Arguments

| | |
|---|---|
| delta | Data frame or matrix of preferences, ratings, dissimilarities |
| ndim | Number of dimensions |
| center | If TRUE input data are centered row-wise. |
| scale | If TRUE input data are scaled row-wise. |
| col.coord | Optional fixed coordinates for the column objects in delta. |
| x | Object of class "vmu". |
| ... | Additional arguments passed to biplot in stats. |

## Value

| | |
|---|---|
| conf.row | Row coordinates |
| conf.col | Column coordinates |
| VAF | variance accounted for |

## Author(s)

Ingwer Borg and Patrick Mair

## References

Borg, I., & Groenen, P. J. F. (2005). Modern Multidimensional Scaling (2nd ed.). Springer.

Borg, I., Groenen, P. J. F., & Mair, P. (2018). Applied Multidimensional Scaling and Unfolding (2nd ed.). Springer.

Tucker, L. R. (1960). Intra-individual and inter-individual multidimensionality. In H. Gulliksen & S. Messick (Eds.), Psychological scaling: Theory and applications (pp. 155-167). Wiley.

## See Also

biplot, unfolding

## Examples

```
## VMU on portrait value questionnaire ratings
fit_vmu <- vmu(PVQ40agg)         ## fit 2D VMU
fit_vmu
plot(fit_vmu, cex = c(1, 0.7))   ## call biplot from stats

## VMU with fixed column coordinates (circular)
tuv <- matrix(0, nrow = 10, ncol = 2)
alpha <- -360/10
for (i in 1:10){
  alpha <- alpha+360/10
  tuv[i, 1]<- cos(alpha*pi/180)
  tuv[i, 2] <- sin(alpha*pi/180)
}
fit_vmu2 <- vmu(PVQ40agg, col.coord = tuv)  ## fit 2D circular VMU
fit_vmu2
plot(fit_vmu2, cex = c(1, 0.7))
```

---

# winedat *Wine tasting*

---

## Description

This dataset collects dissimilarity matrices of 10 raters of 6 different wines.

## Usage

```
data(winedat)
```

## Format

A list of dissimilarity matrices reflecting the rating of 10 judges on 6 different wines (Ziniel Chardonnay, Markowitsch Chardonnay, Krems Chardonnay, Castel Nova Chardonnay, Ritinitis Noble Retsina, RetsinaCriteria). The attributes color, smell, taste, fun, and overall impression were rated on a scale from 1 (very good) to 5. Based on these ratings the distances were computed.

## Examples

```
winedat
```

wish *Wish dataset*

## Description

Similarity ratings for 12 countries. There were no instructions concerning the characteristics on which these similarity judgements were to be made, this was information to discover rather than to impose.

## Usage

```
data(wish)
```

## Format

Object of class `dist`

## Details

For smacof, the data must be converted into a dissimilarity matrix (see examples).

## References

Borg, I., Groenen, P. J. F., & Mair, P. (2010). Multidimensionale Skalierung. Muenchen: Hampp Verlag.

Borg, I., Groenen, P. J. F., & Mair, P. (2012). Multidimensional Scaling. New York: Springer, forthcoming.

Wish, M. (1971). Individual differences in perceptions and preferences among nations. In C. W. King and D. Tigert (Eds.), Attitude research reaches new heights, pp. 312-328. Chicago: American Marketing Association.

## Examples

```
data(wish)
sim2diss(wish, method = max(wish))
```

# Index