Package ‘smacof’

March 3, 2020

**Type** Package

**Title** Multidimensional Scaling

**Version** 2.1-0

**Date** 2020-03-02

**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, nls, 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

**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** 2020-03-03 16:34:03 UTC
### R topics documented:

- biplotmds ........................................ 3
- bootmds .......................................... 5
- bread ............................................ 7
- breakfast ....................................... 8
- CanadaNews ..................................... 9
- confEllipse ..................................... 9
- crimes ......................................... 11
- csrranking ..................................... 12
- dissWeights ..................................... 12
- driftVectors .................................... 14
- Duration ....................................... 16
- ekman .......................................... 17
- EW_ger ......................................... 18
- FaceExp ........................................ 19
- fitCircle ........................................ 19
- GOPdtm .......................................... 20
- gravity ......................................... 21
- Guerry .......................................... 22
- Guttman1991 .................................... 22
- helm ............................................ 23
- icExplore ....................................... 24
- indvalues ...................................... 25
- intelligence .................................... 26
- jackmds ......................................... 27
- kinshipdelta .................................... 29
- KIPT ............................................. 30
- LawLer .......................................... 31
- Morse ........................................... 31
- morsescales ..................................... 32
- OCP .............................................. 33
- partypref ....................................... 34
- perception ...................................... 34
- permtest ........................................ 35
- Plato7 ............................................ 37
- plot.smacof ..................................... 38
- Procrustes ...................................... 38
- PVQ40 ........................................... 41
- randomstress .................................... 43
- rectangles ...................................... 44
- residuals.smacof ............................... 45
- RockHard ....................................... 46
- sim2diss ......................................... 47
- smacofConstraint ............................... 48
- smacofIndDiff ................................... 52
- smacofSphere ................................... 54
- smacofSym ....................................... 57
biplotmds

MDS Biplots

Description

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

Usage

```r
## 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


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(3,1,3,3,7,9,4,7,10,6))
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)
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" )

## 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(ly = 1, lwd = 1, col = "gray"), main, xlab, ylab, xlim, ylim,
asp = 1, type = "p", pch = 20, ...)
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

## 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

---

## 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

**Examples**

breakfast

---

<table>
<thead>
<tr>
<th>breakfast</th>
<th>Breakfast preferences</th>
</tr>
</thead>
</table>

**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**


**Examples**

breakfast
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


Examples

str(CanadaNews)

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, ...)

Pseudo Confidence Ellipses
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 difference 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


See Also

plot.smacofboot
Examples

```r
## 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**

```r
data(crimes)
```

**Format**

Crime correlation matrix.

**References**


**Examples**

```r
crimes
```
csrranking | CSR activities
--- | ---

Description
This dataset collects rankings of 100 individual on 5 topics that reflect social responsibilities 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 occur most often will start to dominate the solution. This option de-emphasizes often occurring 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 $\delta^{\text{power}}$. If $\text{power}$ is small (e.g., $\text{power} = -5$) then the smaller dissimilarities will be better fitted. If $\text{power}$ is large (e.g., $\text{power} = 5$) 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

```r
## 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

#### 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

```r
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), ...)
```

---

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

```r
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
### Examples

```r
# 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)
```

---

### Description

The *Duration* 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

Examples

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

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
Examples

ekman

---

**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.


**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


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


Examples

```r
## 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

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document-term matrix based on statements by Republican voters.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>data(GOPdtm)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document-term matrix with statements in the rows and terms (keywords) in the columns</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>This dataset emerges from statements of Republican voters scraped from the official GOP website. They were asked to complete the sentence &quot;I am a Republican because ...&quot;. We have selected the 37 most frequent words and created a document-term matrix.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>References</th>
</tr>
</thead>
</table>
gravity

Examples

data(GOPdtm)
GOPdtm

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-occurrences. Based on the binarized data matrix $Y$ we compute $Y^T Y$ which leads to the co-occurrence 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-occurrences 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-occurrences

Author(s)

Patrick Mair

References


See Also

mds
Examples

```r
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


Examples

Guerry

Guttman1991  Guttman’s Intelligence Facets

Description


Usage

data(Guttman1991)
data(Guttman1965)
**helm**

**Format**

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

**References**


**Examples**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guttman1991[[1]]</td>
<td>## similarity matrix</td>
</tr>
<tr>
<td>Guttman1991[[2]]</td>
<td>## facets</td>
</tr>
<tr>
<td>Guttman1965[[1]]</td>
<td>## similarity matrix</td>
</tr>
<tr>
<td>Guttman1965[[2]]</td>
<td>## facets</td>
</tr>
</tbody>
</table>

**Description**

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

**Usage**

```r
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


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
**indvalues**

**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**


**See Also**

mds

**Examples**

```r
## 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**


**See Also**

PVQ40

**Examples**

str(indvalues)

<table>
<thead>
<tr>
<th>intelligence</th>
<th>Intelligence Tests</th>
</tr>
</thead>
</table>

**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 following 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
Language: numerical, geometrical
Requirement: application, inference
T1-T8: intercorrelations
R1-R8: restrictions
References


Examples

```r
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**

```r
# 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 and connected with their centroid.
- `legend`: If TRUE, the centroid labels are added as legend.
Inset distance from the margins as a fraction of the plot region when legend is placed by keyword.

Character expansion factor for legend.

Plot title.

Label of x-axis.

Label of y-axis.

Scale x-axis.

Scale y-axis.

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 configurations

An array of n-1 configuration matrices for each Jackknife MDS solution

Centroid Jackknife configurations (comparison matrix)

Stability measure

Cross validity

Dispersion

Value of the loss function

Number of dimensions

Model call

Number of iterations

Number of objects

Author(s)

Jan de Leeuw and Patrick Mair

References

See Also

bootmds

Examples

```r
## 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

<table>
<thead>
<tr>
<th>kinshipdelta</th>
<th>Kinship Terms</th>
</tr>
</thead>
</table>

**Description**

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

**Usage**

```r
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**


**Examples**

```r
kinshipdelta
kinshipscales
```
**Description**

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

**Usage**

\[ \text{data(KIPT)} \]

**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**


**Examples**

\[ \text{KIPT} \]
\[ \text{sim2diss(KIPT)} \]
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


Examples

Lawler

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**


**Examples**

morse
morse2

<table>
<thead>
<tr>
<th>morsescales</th>
<th>Morse Code Confusion Scales</th>
</tr>
</thead>
</table>

**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**

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


Examples

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

Description

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

Usage

data(partypref)

Format

Matrix of party preferences.

References


Examples

partypref

Perception

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


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**


**Examples**

`Plato7`
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, 1 = 50), cex = 0.8),
     label.conf.columns = list(label = TRUE, pos = 3,
     col = hcl(240, 1 = 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).
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pch</td>
<td>Plot symbol.</td>
</tr>
<tr>
<td>cex</td>
<td>Symbol size.</td>
</tr>
<tr>
<td>asp</td>
<td>Aspect ratio.</td>
</tr>
<tr>
<td>col</td>
<td>Point color.</td>
</tr>
<tr>
<td>sphere</td>
<td>In case of spherical smacof, whether sphere should be plotted or not.</td>
</tr>
<tr>
<td>subscale</td>
<td>Scaling factor (size) for the bubble plot.</td>
</tr>
<tr>
<td>label.conf</td>
<td>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.</td>
</tr>
<tr>
<td>hull.conf</td>
<td>Option to add convex hulls to a configuration plot. Hull index needs to be provided.</td>
</tr>
<tr>
<td>shepard.x</td>
<td>Shepard plot only: original data (e.g. correlation matrix) can be provided for plotting on x-axis.</td>
</tr>
<tr>
<td>identify</td>
<td>If TRUE, the identify() function is called internally that allows to add configuration labels by mouse click.</td>
</tr>
<tr>
<td>what</td>
<td>For unfolding only: Whether row coordinates, column coordinates, or both should be plotted.</td>
</tr>
<tr>
<td>col.rows</td>
<td>Row colors in unfolding configuration plot.</td>
</tr>
<tr>
<td>col.columns</td>
<td>Column colors in unfolding configuration plot.</td>
</tr>
<tr>
<td>col.dhat</td>
<td>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.</td>
</tr>
<tr>
<td>label.conf.rows</td>
<td>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).</td>
</tr>
<tr>
<td>label.conf.columns</td>
<td>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).</td>
</tr>
<tr>
<td>col.hist</td>
<td>Color of the borders of the histogram.</td>
</tr>
<tr>
<td>plot.array</td>
<td>Array arrangements of plots for individual difference models (see details).</td>
</tr>
</tbody>
</table>

... 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

```r
## 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(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
ekman0 <- sim2diss(ekman)
fit1 <- mds(kman0, type = "ordinal")
plot(fit1, plot.type = "Shepard")
plot(fit1, plot.type = "Shepard", shepard.x = ekman0) ## original data on x-axis

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

\[ res \leftarrow \text{unfolding(breakfast)} \]
\[ \text{plot(res, plot.type = "confplot")} \]
\[ \text{plot(res, plot.type = "stressplot")} \]

---

**Procrustes**

**Procrustean Similarity Transformations**

**Description**

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

**Usage**

```r
Procrustes(X, Y)
```

## S3 method for class 'procr'

```r
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


Examples

```r
## 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 = "")
```
## 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**

```r
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


See Also

indvalues

Examples

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

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


Examples

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

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 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

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

Format

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


Examples

rectangles
rect_constr
rectangles2

residuals.smacof

Description

Computes the residuals by subtracting the configuration dissimilarities 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**

```r
data(RockHard)
```

**Format**

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

**References**


**Examples**

```r
head(RockHard)
```

---

### sim2diss

*Converts similarities to dissimilarities*

**Description**

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

**Usage**

```r
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", "coocurrence", "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 dissimilarities either as matrix or as dist object.

Examples

```r
## 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

```r
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 ≤ k1 ≤ k2 ≤ 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 \( (n \times q) \) 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 \( (n \times (n + p)) \). 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 \( \text{init} \) (mandatory in this case), matrix \( V \) (weightmat; based on the weight matrix, see package vignette), external scale matrix \( \text{external} \). 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 \( \text{mds()} \) first, and use the resulting configurations as starting configuration for \( \text{smacofConstraint()} \). 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
smacofConstraint

References


See Also

smacofSym, smacofRect, smacofIndDiff, smacofSphere

Examples

```r
## 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")
fitMorse2 <- smacofConstraint(morse, type = "ordinal", constraint = "unrestricted",
                           external = morsescales[,2:3],
                           constraint.type = "ordinal",
                           init = fitMorse1$conf)
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",
```

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.

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

Value

da  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
obj  Number of objects

Author(s)

Jan de Leeuw and Patrick Mair

References

See Also

`smacofConstraint`, `smacofSym`, `smacofRect`, `smacofSphere`

Examples

```r
## 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

```r
delta = 10
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


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
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 dissimilarities 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


See Also

[smacofConstraint, smacofRect, smacofIndDiff, smacofSphere, plot.smacof]
Examples

```r
## 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(PVQ4Qagg))
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

```r
data(stardist)
```

Format

A dist object containing the star distances.
Note
Thanks to Paul Eigenthaler, Department of Astronomy, University of Vienna for calculating the distances.

Examples

stress0

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)
Description

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

Usage

```r
svm_mdsplot(mds_object, svm_object, class, legend1 = TRUE, legend2 = TRUE,
    inset = c(-0.2, 0.5), plot.dim = c(1,2), 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.
- `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`
symdecomp

### Examples

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

### ordinal MDS fit
mds_gut <- mds(Delta, ndim = 2, type = "ordinal")
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$conf
dat <- data.frame(class = class, X)
costvec <- 2^seq(-4, 4)
gamma <- seq(0.01, 0.5, 10)
set.seed(111)
svm_gut <- tune.svm(class ~ D1 + D2, data = dat, kernel = "radial",
cross = 10, cost = costvec)$best.model
preds <- predict(svm_gut, data = dat)
table(obs = class, pred = preds)
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**

```r
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


Examples

\[
P <- \text{matrix}(c(92,5,4,8,4,84,38,62,6,37,87,17,13,31,17,88), \text{ncol} = 4)
\]
\[
symdecomp(P)
\]

\[
torgerson \quad \text{Torgerson Scaling}
\]

Description

Classical MDS aka Torgerson Scaling

Usage

torgerson(delta, p)

Arguments

\( \text{delta} \)  Dissimilarity matrix
\( \text{p} \)  Number of dimensions

Value

Returns an \( n \times p \) matrix of configurations

References


Examples

\[
\text{fit} <- \text{torgerson}(\text{Guerry})
\]
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

unfolding

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 uni-
formly 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.
imax 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 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)
unfolding

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


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**


**See Also**

mds
Examples

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

Description

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

```r
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


Examples

```r
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**

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

```r
## 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**


winedat

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
  tv[i, 1]<- cos(alpha*pi/180)
  tv[i, 2] <- sin(alpha*pi/180)
}
fit_vmu2 <- vmu(PVQ40agg, col.coord = tv)  ## 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 Chardon-
nay, 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 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


Examples

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

*Topic **datasets**
  - bread, 7
  - breakfast, 8
  - CanadaNews, 9
  - crimes, 11
  - csrranking, 12
  - Duration, 16
  - ekman, 17
  - EW.ger, 18
  - FaceExp, 19
  - GOPdtm, 20
  - Guerry, 22
  - Guttman1991, 22
  - helm, 23
  - indvalues, 25
  - intelligence, 26
  - kinshipdelta, 29
  - KIPT, 30
  - LawLer, 31
  - morse, 31
  - morsescales, 32
  - OCP, 33
  - partypref, 34
  - perception, 34
  - Plato7, 37
  - PVQ40, 43
  - rectangles, 45
  - RockHard, 47
  - stardist, 59
  - trading, 65
  - VaziriXu, 71
  - winedat, 73
  - Wish, 74

*Topic **hplot**
  - biplotmds, 3
  - bootmds, 5
  - confEllipse, 9
  - jackmds, 27
  - plot.smacof, 38

  - Procrustes, 41
  - svm_mdsplot, 62

*Topic **methods**
  - residuals.smacof, 46
  - summary.smacofB, 61

*Topic **models**
  - bootmds, 5
  - fitCircle, 19
  - gravity, 21
  - jackmds, 27
  - permtest, 35
  - randomstress, 44
  - sim2diss, 47
  - svm_mdsplot, 62
  - unfolding, 66

*Topic **multivariate**
  - driftVectors, 14
  - Procrustes, 41
  - smacofConstraint, 48
  - smacofIndDiff, 52
  - smacofSphere, 54
  - smacofSym, 57
  - torgerson, 64
  - uniscale, 70
  - vmu, 72

*Topic **utilities**
  - icExplore, 24
  - stress0, 60
  - symdecomp, 63

*Topic **weights**
  - dissWeights, 12

  - biplot, 73
  - biplotmds, 3
  - bootmds, 5, 29, 36
  - bread, 7
  - breakfast, 8

  - CanadaNews, 9
  - confEllipse, 9
crimes, 11
csrranking, 12
dissWeights, 12
driftVectors, 14
Duration, 16
DurationRaw (Duration), 16
ekman, 17
EW_eng (EW_ger), 18
EW_ger, 18
FaceExp, 19
FaceScale (FaceExp), 19
fitCircle, 19
GOPdtm, 20
gavity, 21
Guerry, 22
Guttman1965 (Guttman1991), 22
Guttman1991, 22
helm, 23
icExplore, 24
idoscal (smacofIndDiff), 52
image, 62
indscale (smacofIndDiff), 52
indvalues, 25, 44
intelligence, 26
jackmds, 7, 27, 36
kinshipdelta, 29
kinshipscales (kinshipdelta), 29
KPT, 30
LawLer, 31
Lawler (LawLer), 31
lm, 4
mds, 21, 50, 60, 70
mds (smacofSym), 57
morse, 31
morse2 (morse), 31
morsescales, 32
OCP, 33
partypref, 34
perception, 34
permtest, 35
Plato7, 37
plot, 4, 28, 38, 39
plot.confell (confEllipse), 9
plot.driftvec (driftVectors), 14
plot.icexplore (icExplore), 24
plot.mdsbi (biplotmds), 3
plot.procr, 40
plot.procr (Procrustes), 41
plot.smacof, 4, 38, 58, 69
plot.smacofboot, 10
plot.smacofboot (bootmds), 5
plot.smacofID (plot.smacof), 38
plot.smacofJK (jackmds), 27
plot.smacofPerm (permtest), 35
plot.smacofR (plot.smacof), 38
plot.uniscale (uniscale), 70
plot.vmu (vmu), 72
prefscal (unfolding), 66
print.smacofB (summary.smacofB), 61
print.smacofID (summary.smacofB), 61
print.smacofJK (jackmds), 27
print.smacofPerm (permtest), 35
print.smacofR (summary.smacofB), 61
print.uniscale (uniscale), 70
print.vmu (vmu), 72
Procrustes, 41
PVQ40, 26, 43
PVQ40agg (PVQ40), 43
rainbow_hcl, 27
randomstress, 44
rect_constr (rectangles), 45
rectangles, 35, 45
rectangles2 (rectangles), 45
residuals.smacof, 46
residuals.smacofID (residuals.smacof), 46
residuals.smacofR (residuals.smacof), 46
RockHard, 47
sim2diss, 47
smacofConstraint, 48, 54, 56, 58, 69
smacofIndDiff, 51, 52, 56, 58, 69
smacofRect, 51, 54, 56, 58
smacofRect (unfolding), 66
smacofSphere, 51, 54, 54, 58, 69
smacofSym, 16, 51, 54, 56, 57, 69
stardist, 59
stress0, 60
summary.smacofB, 61
summary.smacofID(summary.smacofB), 61
summary.smacofR(summary.smacofB), 61
svm, 62
svm_mdsplot, 62
symdecomp, 63
torgerson, 64
trading, 65
transform, 66
transPrep(transform), 66
tune.svm, 62
unfolding, 66, 73
uniscale, 70
VaziriXu, 71
vmu, 72
winedat, 73
wish, 74
wtd.hist, 40