Package ‘smacofx’

July 23, 2025

Title Flexible Multidimensional Scaling and 'smacof' Extensions

Version 1.21-1

Maintainer Thomas Rusch <thomas.rusch@wu.ac.at>

Description Flexible multidimensional scaling (MDS) methods and extensions to the package 'smacof'. This package contains various functions, wrappers, methods and classes for fitting, plotting and displaying a large number of different flexible MDS models. These are: Torgerson scaling (Torgerson, 1958, ISBN:978-0471879459) with powers, Sammon mapping (Sammon, 1969, <doi:10.1109/T-C.1969.222678>) with ratio and interval optimal scaling, Multiscale MDS (Ramsay, 1977, <doi:10.1007/BF02294052>) with ratio and interval optimal scaling, s-
stress MDS (ALSCAL; Takane, Young & De Leeuw, 1977, <doi:10.1007/BF02293745>) with ratio and interval optimal scaling, elastic scaling (McGee, 1966, <doi:10.1111/j.2044-8317.1966.tb00367.x>) with ratio and interval optimal scaling, r-stress MDS (De Leeuw, Groenen & Mair, 2016, <https://rpubs.com/deleeuw/142619>) with ratio, interval, splines and nonmetric optimal scaling, power-stress MDS (POST-MDS; Buja & Swayne, 2002 <doi:10.1007/s00357-001-0031-0>) with ratio and interval optimal scaling, restricted power-stress (Rusch, Mair & Hornik, 2021, <doi:10.1080/10618600.2020.1869027>) with ratio and interval optimal scaling, approximate power-stress with ratio optimal scaling (Rusch, Mair & Hornik, 2021, <doi:10.1080/10618600.2020.1869027>), Box-Cox MDS (Chen & Buja, 2013, <https://jmlr.org/papers/v14/chen13a.html>), local MDS (Chen & Buja, 2009, <doi:10.1198/jasa.2009.0111>), curvilinear component analysis (Demartines & Herault, 1997, <doi:10.1109/72.554199>), curvilinear distance analysis (Lee, Lendasse & Verleysen, 2004, <doi:10.1016/j.neucom.2004.01.007>), nonlinear MDS with optimal dissimilarity powers functions (De Leeuw, 2024, <https://github.com/deleeuw/smacofManual/blob/main/smacofPO(power)/smacofPO.pdf>), sparsified (power) MDS and sparsified multidimensional (power) distance analysis aka extended curvilinear (power) component analysis and extended curvilinear (power) distance analysis (Rusch, 2024, <doi:10.57938/355bf835-ddb7-42f4-8b85-129799fc240e>). Some functions are suitably flexible to allow any other sensible combination of explicit power transformations for weights, distances and input proximities with implicit ratio, interval, splines or nonmetric optimal scaling of the input proximities. Most functions use a Majorization-Minimization algorithm. Currently the methods are only available for one-mode two-way data (symmetric dissimilarity matrices).

Depends R (>= 3.5.0), smacof (>= 1.10-4)

Imports MASS, minqa, plotrix, ProjectionBasedClustering, weights, vegan

License GPL-2 | GPL-3

LazyData true

URL https://r-forge.r-project.org/projects/stops/

BugReports https://r-forge.r-project.org/tracker/?atid=5375&group_id=2037&func=browse

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation no

Author Thomas Rusch [aut, cre] (ORCID: <https://orcid.org/0000-0002-7773-2096>),
Jan de Leeuw [aut],
Lisha Chen [aut],
Patrick Mair [aut] (ORCID: <https://orcid.org/0000-0003-0100-6511>)

Repository CRAN

Date/Publication 2025-07-07 10:30:02 UTC

Contents

`alscal` *ALSCAL - MDS via S-Stress Minimization*

Description

An implementation to minimize s-stress by majorization with ratio and interval optimal scaling.

Usage

```
alscal(
  delta,
  type = "ratio",
  weightmat,
  init = NULL,
  ndim = 2,
  acc = 1e-06,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE
)
```

Arguments

`delta`	dist object or a symmetric, numeric data.frame or matrix of distances
`type`	what type of MDS to fit. Currently one of "ratio" or "interval". Default is "ratio".
`weightmat`	a matrix of finite weights
`init`	starting configuration
`ndim`	dimension of the configuration; defaults to 2
`acc`	numeric accuracy of the iteration. Default is 1e-6.
`itmax`	maximum number of iterations. Default is 10000.
`verbose`	should iteration information been given; if > 0 then yes
`principal`	If 'TRUE', principal axis transformation is applied to the final configuration

Value

a 'smacofP' object (inheriting from 'smacofB', see `smacofSym`). It is a list with the components

- delta: Observed untransformed dissimilarities
- tdelta: Observed explicitly transformed (squared) dissimilarities, normalized
- dhat: Explicitly transformed dissimilarities (dhats), optimally scaled and normalized
- confdist: Transformed configuration distances
- conf: Matrix of fitted configuration
- stress: Default stress (stress 1; sqrt of explicitly normalized stress)
- spp: Stress per point
- ndim: Number of dimensions
- model: Name of smacof model
- niter: Number of iterations
- nobj: Number of objects
- type: Type of MDS model
- weightmat: weighting matrix as supplied
- stress.m: Default stress (stress-1^2)
- tweightmat: transformed weighting matrix (here NULL)

See Also

`rStressMin`

Examples

```
dis<-smacof::kinshipdelta
res<-alscal(as.matrix(dis),type="interval",itmax=1000)
res
summary(res)
plot(res)
```

apStressMin *Approximate Power Stress MDS*

Description

An implementation to minimize approximate power stress by majorization with ratio or interval optimal scaling. This approximates the power stress objective in such a way that it can be fitted with SMACOF without distance transformations. See Rusch et al. (2021) for details.

Usage

```
apStressMin(
  delta,
  kappa = 1,
  lambda = 1,
  nu = 1,
  type = "ratio",
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  ndim = 2,
  acc = 1e-06,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE
)

apowerstressMin(
  delta,
  kappa = 1,
  lambda = 1,
  nu = 1,
  type = "ratio",
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  ndim = 2,
  acc = 1e-06,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE
)

apostmds(
  delta,
  kappa = 1,
  lambda = 1,
  nu = 1,
  type = "ratio",
    weightmat = 1 - diag(nrow(delta)),
    init = NULL,
    ndim = 2,
    acc = 1e-06,
    itmax = 10000,
    verbose = FALSE,
    principal = FALSE
  )

  apstressMin(
    delta,
    kappa = 1,
    lambda = 1,
    nu = 1,
    type = "ratio",
    weightmat = 1 - diag(nrow(delta)),
    init = NULL,
    ndim = 2,
    acc = 1e-06,
    itmax = 10000,
    verbose = FALSE,
    principal = FALSE
  )

  apstressmds(
    delta,
    kappa = 1,
    lambda = 1,
    nu = 1,
    type = "ratio",
    weightmat = 1 - diag(nrow(delta)),
    init = NULL,
    ndim = 2,
    acc = 1e-06,
    itmax = 10000,
    verbose = FALSE,
    principal = FALSE
  )
```

Arguments

`delta`	dist object or a symmetric, numeric data.frame or matrix of distances
`kappa`	power of the transformation of the fitted distances; defaults to 1
`lambda`	the power of the transformation of the proximities; defaults to 1
`nu`	the power of the transformation for weightmat; defaults to 1
`type`	what type of MDS to fit. Only "ratio" currently.
`weightmat`	a binary matrix of finite nonegative weights.

`init`	starting configuration
`ndim`	dimension of the configuration; defaults to 2
`acc`	numeric accuracy of the iteration. Default is 1e-6.
`itmax`	maximum number of iterations. Default is 10000.
`verbose`	should iteration output be printed; if > 1 then yes
`principal`	If 'TRUE', principal axis transformation is applied to the final configuration

Value

a 'smacofP' object (inheriting from 'smacofB', see `smacofSym`). It is a list with the components

- delta: Observed, untransformed dissimilarities
- tdelta: Observed explicitly transformed dissimilarities, normalized
- dhat: Explicitly transformed dissimilarities (dhats), optimally scaled and normalized
- confdist: Tranformed configuration distances
- conf: Matrix of fitted configuration
- stress: Default stress (stress 1; sqrt of explicitly normalized stress)
- spp: Stress per point
- ndim: Number of dimensions
- model: Name of smacof model
- niter: Number of iterations
- nobj: Number of objects
- type: Type of MDS model
- weightmat: weighting matrix as supplied
- stress.m: Default stress (stress-1^2)
- tweightmat: transformed weighting matrix (here weightmat^nu)

Note

Internally we calculate the approximation parameters upsilon=nu+2*lambda*(1-(1/kappa)) and tau=lambda/kappa. They are not output.

References

Rusch, Mair, Hornik (2021). Cluster Optimized Proximity Scaling. JCGS <doi:10.1080/10618600.2020.1869027>

Examples

```
dis<-smacof::kinshipdelta
res<-apStressMin(as.matrix(dis),kappa=2,lambda=1.5,itmax=1000)
res
summary(res)
plot(res)
plot(res,"Shepard")
plot(res,"transplot")
```

BankingCrisesDistances *Banking Crises Distances*

Description

Matrix of Jaccard distances between 70 countries (Hungary and Greece were combined to be the same observation) based on their binary time series of having had a banking crises in a year from 1800 to 2010 or not. See data(bankingCrises) in package Ecdat for more info. The last column is Reinhart & Rogoffs classification as a low (3), middle- (2) or high-income country (1).

Format

A 69 x 70 matrix.

Source

data(bankingCrises) in library(Ecdat)

bcmds *Box-Cox MDS*

Description

This function minimizes the Box-Cox Stress of Chen & Buja (2013) via gradient descent. This is a ratio metric scaling method. The transformations are not straightforward to interpret but mu is associated with fitted distances in the configuration and lambda with the dissimilarities. Concretely for fitted distances (attraction part) it is $BC_{mu+lambda}(d(X))$ and for the repulsion part it is $delta^lambdaBC_{mu}(d(X))$ with BC being the one-parameter Box-Cox transformation.

Usage

```
bcmds(
  delta,
  mu = 1,
  lambda = 1,
  rho = 0,
  type = "ratio",
  ndim = 2,
  weightmat = 1 - diag(nrow(delta)),
  itmax = 2000,
  init = NULL,
  verbose = 0,
  addD0 = 1e-04,
  principal = FALSE,
    normconf = FALSE,
    acc = 1e-05
  )

  bcStressMin(
    delta,
    mu = 1,
    lambda = 1,
    rho = 0,
    type = "ratio",
    ndim = 2,
    weightmat = 1 - diag(nrow(delta)),
    itmax = 2000,
    init = NULL,
    verbose = 0,
    addD0 = 1e-04,
    principal = FALSE,
    normconf = FALSE,
    acc = 1e-05
  )

  bcstressMin(
    delta,
    mu = 1,
    lambda = 1,
    rho = 0,
    type = "ratio",
    ndim = 2,
    weightmat = 1 - diag(nrow(delta)),
    itmax = 2000,
    init = NULL,
    verbose = 0,
    addD0 = 1e-04,
    principal = FALSE,
    normconf = FALSE,
    acc = 1e-05
  )

  boxcoxmds(
    delta,
    mu = 1,
    lambda = 1,
    rho = 0,
    type = "ratio",
    ndim = 2,
    weightmat = 1 - diag(nrow(delta)),
    itmax = 2000,
    init = NULL,
  verbose = 0,
  addD0 = 1e-04,
  principal = FALSE,
  normconf = FALSE,
  acc = 1e-05
)
```

Arguments

`delta`	dissimilarity or distance matrix, dissimilarity or distance data frame or 'dist' object
`mu`	mu parameter. Should be 0 or larger for everything working ok. If mu<0 it works but I find the MDS model is strange and normalized stress tends towards 0 regardless of fit. Use normalized stress at your own risk in that case.
`lambda`	lambda parameter. Must be larger than 0.
`rho`	the rho parameter, power for the weights (called nu in the original article).
`type`	what type of MDS to fit. Only "ratio" currently.
`ndim`	the dimension of the configuration
`weightmat`	a matrix of finite weights. Not implemented.
`itmax`	number of optimizing iterations, defaults to 2000.
`init`	initial configuration. If NULL a classical scaling solution is used.
`verbose`	prints progress if > 3.
`addD0`	a small number that's added for D(X)=0 for numerical evaluation of worst fit (numerical reasons, see details). If addD0=0 the normalized stress for mu!=0 and mu+lambda!=0 is correct, but will give useless normalized stress for mu=0 or mu+lambda!=0.
`principal`	If 'TRUE', principal axis transformation is applied to the final configuration
`normconf`	normalize the configuration to sum(delta^2)=1 (as in the power stresses). Default is FALSE. Note that then the distances in confdist do not match manually calculated ones.
`acc`	Accuracy (lowest stepsize). Defaults to 1e-5.

Details

For numerical reasons with certain parameter combinations, the normalized stress uses a configuration as worst result where every d(X) is 0+addD0. The same number is not added to the delta so there is a small inaccuracy of the normalized stress (but negligible if min(delta)»addD0). Also, for mu<0 or mu+lambda<0 the normalization cannot generally be trusted (in the worst case of D(X)=0 one would have an 0^(-a)).

Value

an object of class 'bcmds' (also inherits from 'smacofP'). It is a list with the components

- delta: Observed, untransformed dissimilarities

- tdelta: Observed explicitly transformed dissimilarities, normalized
- dhat: Explicitly transformed dissimilarities (dhats)
- confdist: Configuration dissimilarities
- conf: Matrix of fitted configuration
- stress: Default stress (stress 1; sqrt of explicitly normalized stress)
- ndim: Number of dimensions
- model: Name of MDS model
- type: Must be "ratio" here.
- niter: Number of iterations
- nobj: Number of objects
- pars: hyperparameter vector theta
- weightmat: 1-diagonal matrix. For compatibility with smacofP classes.
- parameters, pars, theta: The parameters supplied
- call the call

and some additional components

- stress.m: default stress is the explicitly normalized stress on the normalized, transformed dissimilarities
- mu: mu parameter (for attraction)
- lambda: lambda parameter (for repulsion)
- rho: rho parameter (for weights)

Author(s)

Lisha Chen & Thomas Rusch

Examples

```
dis<-smacof::kinshipdelta
res<-bcmds(dis,mu=2,lambda=1.5,rho=0)
res
summary(res)
plot(res)
```

`bcsdistance` *Calculates the blended Chi-square distance matrix between n vectors*

Description

The pairwise blended chi-distance of two vectors x and y is sqrt(sum(((x[i]-y[i])^2)/(2*(ax[i]+by[i])))), with originally a in [0,1] and b=1-a as in Lindsay (1994) (but we allow any non-negative a and b). The function calculates this for all pairs of rows of a matrix or data frame x.

Usage

```
bcsdistance(x, a = 0.5, b = 1 - a)
```

Arguments

x	an n times p numeric matrix or data frame. Note that the valeus of x must be non-negative.
a	first blending weight. Must be non-negative and should be in [0,1] if a blended chi-square distance as in Lindsay (1994) is sought. Defaults to 0.5.
b	second blending weight. Must be non-negative and should be 1-a if a blended chi-square distance as in Lindsay (1994) is sought. Defaults to 1-a.

Value

a symmetric n times n matrix of pairwise blended chi-square distance (between rows of x) with 0 in the main diagonal. It is an object of class distance and matrix with attributes "method", "type" and "par", the latter returning the a and b values.

References

Lindsay (1994). Efficiency versus robustness: the case for minimum Hellinger distance and related methods. Annals of Statistics, 22 (2), 1081-1114. <doi:10.1214/aos/1176325512>

`biplotmds.bcmds` *S3 method for bcmds objects*

Description

S3 method for bcmds objects

Usage

```
## S3 method for class 'bcmds'
biplotmds(object, extvar, scale = TRUE)
```

Arguments

`object`	An object of class smacofP
`extvar`	Data frame with external variables.
`scale`	if 'TRUE' external variables are standardized internally.

Details

If a model for individual differences is provided, the external variables are regressed on the group stimulus space configurations. For objects returned from 'biplotmds' we use the plot method in `biplotmds`. In the biplot called with plot() only the relative length of the vectors and their direction matters. Using the vecscale argument in plot() 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. In this method vecscale should usually be smaller than the one used in smacof by a factor of 0.1. Note that in the biplot object, the configuration is always normalized (which it may not necessarily be in the bcmds object).

Value

Returns an object belonging to classes 'mlm' and 'mdsbi'. See 'lm' for details. R2vec: Vector containing the R2 values. See also `biplotmds` for the plot method.

`biplotmds.lmds` *S3 method for lmds objects*

Description

S3 method for lmds objects

Usage

```
## S3 method for class 'lmds'
biplotmds(object, extvar, scale = TRUE)
```

Arguments

`object`	An object of class smacofP
`extvar`	Data frame with external variables.
`scale`	if 'TRUE' external variables are standardized internally.

Details

If a model for individual differences is provided, the external variables are regressed on the group stimulus space configurations. For objects returned from 'biplotmds' we use the plot method in `biplotmds`. In the biplot called with plot() only the relative length of the vectors and their direction matters. Using the vecscale argument in plot() 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. In this method vecscale should usually be smaller than the one used in smacof by a factor of 0.1. Note that in the biplot object, the configuration is always normalized (which it may not necessarily be in the lmds object).

Value

Returns an object belonging to classes 'mlm' and 'mdsbi'. See 'lm' for details. R2vec: Vector containing the R2 values. See also `biplotmds` for the plot method.

`biplotmds.smacofP` *S3 method for smacofP objects*

Description

S3 method for smacofP objects

Usage

```
## S3 method for class 'smacofP'
biplotmds(object, extvar, scale = TRUE)
```

Arguments

`object`	An object of class smacofP
`extvar`	Data frame with external variables.
`scale`	if 'TRUE' external variables are standardized internally.

Details

If a model for individual differences is provided, the external variables are regressed on the group stimulus space configurations. For objects returned from 'biplotmds' we use the plot method in `biplotmds`. In the biplot called with plot() only the relative length of the vectors and their direction matters. Using the vecscale argument in plot() 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. In this method vecscale should usually be smaller than the one used in smacof by a factor of 0.1.

Value

Returns an object belonging to classes 'mlm' and 'mdsbi'. See 'lm' for details. R2vec: Vector containing the R2 values. See also `biplotmds` for the plot method.

Examples

```
## see smacof::biplotmds for more
res <- powerStressMin(morse,kappa=0.5,lambda=2)
fitbi <- biplotmds(res, morsescales[,2:3])
plot(fitbi, main = "MDS Biplot", vecscale = 0.03)
```

`bootmds.smacofP` *MDS Bootstrap for smacofP objects*

Description

Performs a bootstrap on an MDS solution. It works for derived dissimilarities only, i.e. generated by the call dist(data). The original data matrix needs to be provided, as well as the type of dissimilarity measure used to compute the input dissimilarities.

Usage

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

Arguments

`object`	Object of class smacofP if used as method or another object inheriting from smacofB (needs to be called directly as bootmds.smacofP then).
`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".
`nrep`	Number of bootstrap replications.
`alpha`	Alpha level for condfidence ellipsoids.
`verbose`	If 'TRUE', bootstrap index is printed out.
`...`	Additional arguments needed for dissimilarity computation as specified in `sim2diss`.

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. See `bootmds` for more.

Value

An object of class 'smacofboot', see `bootmds`. With values

- cov: Covariances for ellipse computation
- bootconf: Configurations bootstrap samples
- stressvec: Bootstrap stress values
- bootci: Stress bootstrap percentile confidence interval
- spp: Stress per point (based on stress.en)
- stab: Stability coefficient

Examples

```
##see ?smacof::bootmds for more
data <- na.omit(smacof::PVQ40[,1:5])
diss <- dist(t(data))   ## Euclidean distances
fit <- rStressMin(diss,r=0.5,itmax=1000) ## 2D ratio MDS
set.seed(123)
resboot <- bootmds(fit, data, method.dat = "euclidean", nrep = 10) #run for more nrep
resboot
plot(resboot) #see ?smacof::bootmds for more on the plot method
```

`clca` *Curvilinear Component Analysis (CLCA)*

Description

A wrapper to run curvilinear component analysis via `CCA` and returning a 'smacofP' object. Note this functionality is rather rudimentary.

Usage

```
clca(
  delta,
  Epochs = 20,
  alpha0 = 0.5,
  lambda0,
  ndim = 2,
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
    acc = 1e-06,
    itmax = 10000,
    verbose = 0,
    method = "euclidean",
    principal = FALSE
  )
```

Arguments

`delta`	dist object or a symmetric, numeric data.frame or matrix of distances.
`Epochs`	Scalar; gives the number of passes through the data.
`alpha0`	(scalar) initial step size, 0.5 by default
`lambda0`	the boundary/neighbourhood parameter(s) (called lambda_y in the original paper). It is supposed to be a numeric scalar. It defaults to the 90% quantile of delta.
`ndim`	dimension of the configuration; defaults to 2
`weightmat`	not used
`init`	starting configuration, not used
`acc`	numeric accuracy of the iteration; not used
`itmax`	maximum number of iterations. Not used.
`verbose`	should iteration output be printed; not used
`method`	Distance calculation; currently not used.
`principal`	If 'TRUE', principal axis transformation is applied to the final configuration

Details

This implements CCA as in Desmartines & Herault (1997). A different take on the ideas of curvilinear compomnent analysis is available in the experimental functions `spmds` and `spmds`.

Value

a 'smacofP' object. It is a list with the components

- delta: Observed, untransformed dissimilarities
- tdelta: Observed explicitly transformed dissimilarities, normalized
- dhat: Explicitly transformed dissimilarities (dhats), optimally scaled and normalized
- confdist: Configuration dissimilarities
- conf: Matrix of fitted configuration
- stress: Default stress (stress-1; sqrt of explicitly normalized stress)
- spp: Stress per point
- ndim: Number of dimensions
- model: Name of model
- niter: Number of iterations (training length)

- nobj: Number of objects
- type: Type of MDS model. Only ratio here.
- weightmat: weighting matrix as supplied
- stress.m: Default stress (stress-1^2)
- tweightmat: transformed weighting matrix; it is weightmat here.

Examples

```
dis<-smacof::morse
res<-clca(dis,lambda0=0.4)
res
summary(res)
plot(res)
```

clda *Curvilinear Distance Analysis (CLDA)*

Description

A function to run curvilinear distance analysis via CCA and returning a 'smacofP' object. Note this functionality is rather rudimentary.

Usage

```
clda(
  delta,
  Epochs = 20,
  alpha0 = 0.5,
  lambda0,
  ndim = 2,
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  acc = 1e-06,
  itmax = 10000,
  verbose = 0,
  method = "euclidean",
  principal = FALSE,
  epsilon,
  k,
  path = "shortest",
  fragmentedOK = FALSE
)
```

Arguments

`delta`	dist object or a symmetric, numeric data.frame or matrix of distances. Will be turne dinto geodesci distances.
`Epochs`	Scalar; gives the number of passes through the data.
`alpha0`	(scalar) initial step size, 0.5 by default
`lambda0`	the boundary/neighbourhood parameter(s) (called lambda_y in the original paper). It is supposed to be a numeric scalar. It defaults to the 90% quantile of delta.
`ndim`	dimension of the configuration; defaults to 2
`weightmat`	not used
`init`	starting configuration, not used
`acc`	numeric accuracy of the iteration; not used
`itmax`	maximum number of iterations. Not used.
`verbose`	should iteration output be printed; not used
`method`	Distance calculation; currently not used.
`principal`	If 'TRUE', principal axis transformation is applied to the final configuration
`epsilon`	Shortest dissimilarity retained.
`k`	Number of shortest dissimilarities retained for a point. If both 'epsilon' and 'k' are given, 'epsilon' will be used.
`path`	Method used in 'stepacross' to estimate the shortest path, with alternatives '"shortest"' and '"extended"'.
`fragmentedOK`	What to do if dissimilarity matrix is fragmented. If 'TRUE', analyse the largest connected group, otherwise stop with error.

Details

This implements CLDA as CLCA with geodesic distances. The geodesic distances are calculated via 'vegan::isomapdist', see `isomapdist` for a documentation of what these distances do. 'clda' is just a wrapper for 'clca' applied to the geodesic distances obtained via isomapdist.

Value

a 'smacofP' object. It is a list with the components

- delta: Observed, untransformed dissimilarities
- tdelta: Observed explicitly transformed dissimilarities, normalized
- dhat: Explicitly transformed dissimilarities (dhats), optimally scaled and normalized
- confdist: Configuration dissimilarities
- conf: Matrix of fitted configuration
- stress: Default stress (stress-1; sqrt of explicitly normalized stress)
- spp: Stress per point
- ndim: Number of dimensions

- model: Name of model
- niter: Number of iterations (training length)
- nobj: Number of objects
- type: Type of MDS model. Only ratio here.
- weightmat: weighting matrix as supplied
- stress.m: Default stress (stress-1^2)
- tweightmat: transformed weighting matrix; it is weightmat here.

Examples

```
dis<-smacof::morse
res<-clda(dis,lambda0=0.4,k=4)
res
summary(res)
plot(res)
```

`cmds` *Classical Scaling*

Description

Classical Scaling

Usage

```
cmds(Do)
```

Arguments

`Do`	dissimilarity matrix

`cmdscale` *Wrapper to* `cmdscale` *for S3 class*

Description

Wrapper to `cmdscale` for S3 class

Usage

```
cmdscale(d, k = 2, eig = FALSE, ...)
```

Arguments

`d`	a distance structure such as that returned by 'dist' or a full symmetric matrix containing the dissimilarities
`k`	the maximum dimension of the space which the data are to be represented in
`eig`	indicates whether eigenvalues should be returned. Defaults to TRUE.
`...`	additional parameters passed to cmdscale. See `cmdscale`

Details

overloads stats::cmdscale turns on the liosting and adds slots and class attributes for which there are methods.

Value

Object of class 'cmdscalex' and 'cmdscale' extending `cmdscale`. This wrapper always returns the results of cmdscale as a list, adds column labels to the $points and adds extra elements (conf=points, delta=d, confdist=dist(conf), dhat=d) and the call to the list, and assigns S3 class 'cmdscalex' and 'cmdscale'.

Examples

```
dis<-as.matrix(smacof::kinshipdelta)
res<-cmdscale(dis)
```

`conf_adjust` *conf_adjust: a function to procrustes adjust two matrices*

Description

conf_adjust: a function to procrustes adjust two matrices

Usage

```
conf_adjust(conf1, conf2, verbose = FALSE, eps = 1e-12, itmax = 100)
```

Arguments

`conf1`	reference configuration, a numeric matrix
`conf2`	another configuration, a numeric matrix
`verbose`	should adjustment be output; default to FALSE
`eps`	numerical accuracy
`itmax`	maximum number of iterations

Value

a list with 'ref.conf' being the reference configuration, 'other.conf' the adjusted coniguration and 'comparison.conf' the comparison configuration

`corpsepaint` *Corpse Paint*

Description

A matrix of gray scale images of people in "corpse paint", a black-and-white make-up, plus a surprise.

Format

A 8100 x 32 matrix

Details

The images are gray scale 8 bit, i.e., 0-255 unique gray values scaled to be between 0 and 1. There are 32 total images with pixel size of 90 x 90 that have been vectorized to 32 columns labeled as "F1" through "F32". An image i can be reconstructed with matrix(corpsepaint[,i],ncol=90,nrow=90)

Examples

```
oldpar<-par(no.readonly=TRUE)
par(mfrow=c(4,8))
for(i in 1:ncol(corpsepaint)){
p1<-matrix(corpsepaint[,i],ncol=90,nrow=90,byrow=FALSE)
image(p1,col=gray.colors(256),main=colnames(corpsepaint)[i])
}
par(oldpar)
```

`doubleCenter` *Double centering of a matrix*

Description

Double centering of a matrix

Usage

```
doubleCenter(x)
```

Arguments

`x`	numeric matrix

Value

the double centered matrix

elscal *Elastic Scaling SMACOF*

Description

An implementation to minimize elastic scaling stress by majorization with ratio and interval optimal scaling. Uses a repeat loop.

Usage

```
elscal(
  delta,
  type = c("ratio", "interval"),
  weightmat,
  init = NULL,
  ndim = 2,
  acc = 1e-06,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE
)
```

Arguments

`delta`	dist object or a symmetric, numeric data.frame or matrix of distances
`type`	what type of MDS to fit. Currently one of "ratio" and "interval". Default is "ratio".
`weightmat`	a matrix of finite weights
`init`	starting configuration
`ndim`	dimension of the configuration; defaults to 2
`acc`	numeric accuracy of the iteration. Default is 1e-6.
`itmax`	maximum number of iterations. Default is 10000.
`verbose`	should iteration output be printed; if > 1 then yes
`principal`	If 'TRUE', principal axis transformation is applied to the final configuration

Value

a 'smacofP' object (inheriting from smacofB, see `smacofSym`). It is a list with the components

- delta: Observed untransformed dissimilarities
- tdelta: Observed explicitly transformed dissimilarities, normalized
- dhat: Explicitly transformed dissimilarities (dhats), optimally scaled and normalized
- confdist: Transformation configuration distances
- conf: Matrix of fitted configuration, NOT normalized

- stress: Default stress (stress 1; sqrt of explicitly normalized stress)
- spp: Stress per point (based on stress.en)
- ndim: Number of dimensions
- model: Name of smacof model
- niter: Number of iterations
- nobj: Number of objects
- type: Type of MDS model
- weightmat: weighting matrix as supplied
- tweightmat: transformed weighting matrix (here weightmat/delta^2)
- stress.m: Default stress (stress-1^2)

See Also

`rStressMin`

Examples

```
dis<-smacof::kinshipdelta
res<-elscal(as.matrix(dis),itmax=1000)
res
summary(res)
plot(res)
```

---

enorm *Explicit Normalization Normalizes distances*

Description

Explicit Normalization Normalizes distances

Usage

```
enorm(x, w = 1)
```

Arguments

x	numeric matrix
w	weight

Value

a constant

`icExploreGen` *Exploring initial configurations in an agnostic way*

Description

Allows to user to explore the effect of various starting configurations when fitting an MDS model. This is a bit more general than the icExplore function in smacof, as we allow any PS model to be used as the model is either setup by call or by a prefitted object (for the models in cops and stops we do not have a single UI function which necessitates this). Additionally, one can supply their own configurations and not just random ones.

Usage

```
icExploreGen(
  object,
  mdscall = NULL,
  conflist,
  nrep = 100,
  ndim,
  returnfit = FALSE,
  min = -5,
  max = 5,
  verbose = FALSE
)
```

Arguments

`object`	A fitted object of class 'smacofP', 'smacofB' or 'smacof'. If supplied this takes precedence over the call argument. If given this is added to the output and may be the optimal one.
`mdscall`	Alternatively to a fitted object, one can pass a syntactically valid call for any of the MDS functions cops, stops or smacof that find a configuration (not the ones that do parameter selection like pcops or stops). If object and call is given, object takes precedence.
`conflist`	Optional list of starting configurations.
`nrep`	If conflist is not supplied, how many random starting configurations should be used.
`ndim`	Dimensions of target space.
`returnfit`	Should all fitted MDS be returned. If FALSE (default) none is returned.
`min`	lower bound for the uniform distribution to sample from
`max`	upper bound for the uniform distribution to sample from
`verbose`	If >0 prints the fitting progress.

Details

If no configuration list is supplied, then nrep configurations are simulated. They are drawn from a ndim-dimensional uniform distribution with minimum min and maximum max. We recommend to use the route via supplying a fitted model as these are typically starting from a Torgerson configuration and are likely quite good.

Value

an object of class 'icexplore', see `icExplore` for more. There is a plot method in package 'smacof'.

Examples

```
dis<-kinshipdelta

## Version 1: Using a fitted object (recommended)
res1<-rStressMin(dis,type="ordinal",itmax=100)
resm<-icExploreGen(res1,nrep=5)

## Version 2: Using a call object and supplying conflist
conflist<-list(res1$init,jitter(res1$init,1),jitter(res1$init,1),jitter(res1$init,1))
c1 <- call("smds",delta=dis,tau=0.2,itmax=100)
resm<-icExploreGen(mdscall=c1,conflist=conflist,returnfit=TRUE)

plot(resm)
```

`jackmds.smacofP` *MDS Jackknife for smacofP objects*

Description

These functions perform an MDS Jackknife and plot the corresponding solution.

Usage

```
## S3 method for class 'smacofP'
jackmds(object, eps = 1e-06, itmax = 100, verbose = FALSE)
```

Arguments

`object`	Object of class smacofP if used as method or another object inheriting from smacofB (needs to be called directly as jackmds.smacofP then).
`eps`	Convergence criterion
`itmax`	Maximum number of iterations
`verbose`	If 'TRUE', intermediate stress is printed out.

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 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

An object of class 'smacofJK', see `jackmds`. With values

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

Examples

```
dats <- na.omit(smacof::PVQ40[,1:5])
diss <- dist(t(dats))   ## Euclidean distances
fit <- rStressMin(diss,type="ordinal",r=0.4,itmax=1000) ## 2D ordinal MDS

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)
```

`koller` *Responses to the SCC scale and CSII scale (n=1013).*

Description

1013 respondents answered the items of the consumer susceptibility top interpersonal influence (CSII) and the self-concept clarity (SCC) scale. The answers are on a five-point-Likert scale with 1 meaning "fully agree" and 5 meaning "fully disagree". Note that on scc the 10th item is reversed coded to align with the other items.

Format

A 1013 x 24 data frame.

Source

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

---

lmds *Local MDS*

Description

This function minimizes the Local MDS Stress of Chen & Buja (2006) via gradient descent. This is a ratio metric scaling method.

Usage

```
lmds(
  delta,
  k = 2,
  tau = 1,
  type = "ratio",
  ndim = 2,
  weightmat = 1 - diag(nrow(delta)),
  itmax = 5000,
  acc = 1e-05,
  init = NULL,
  verbose = 0,
  principal = FALSE,
  normconf = FALSE
)
```

Arguments

`delta`	dissimilarity or distance matrix, dissimilarity or distance data frame or 'dist' object
`k`	the k neighbourhood parameter
`tau`	the penalty parameter (suggested to be in [0,1])
`type`	what type of MDS to fit. Only "ratio" currently.
`ndim`	the dimension of the configuration
`weightmat`	a matrix of finite weights. Not implemented.
`itmax`	number of optimizing iterations, defaults to 5000.
`acc`	accuracy (lowest stepsize). Defaults to 1e-5.

`init`	initial configuration. If NULL a classical scaling solution is used.
`verbose`	prints info if > 0 and progress if > 1.
`principal`	If 'TRUE', principal axis transformation is applied to the final configuration
`normconf`	normalize the configuration to sum(delta^2)=1 (as in the power stresses). Note that then the distances in confdist do not match the manually calculated ones.

Details

Note that k and tau are not independent. It is possible for normalized stress to become negative if the tau and k combination is so that the absolute repulsion for the found configuration dominates the local stress substantially less than the repulsion term does for the solution of D(X)=Delta, so that the local stress difference between the found solution and perfect solution is nullified. This can typically be avoided if tau is between 0 and 1. If not, set k and or tau to a smaller value.

Value

an object of class 'lmds' (also inherits from 'smacofP'). See `powerStressMin`. It is a list with the components as in power stress

- delta: Observed, untransformed dissimilarities
- tdelta: Observed explicitly transformed dissimilarities, normalized
- dhat: Explicitly transformed dissimilarities (dhats)
- confdist: Configuration dissimilarities
- conf: Matrix of fitted configuration
- stress: Default stress (stress 1; sqrt of explicitly normalized stress)
- ndim: Number of dimensions
- model: Name of MDS model
- type: Is "ratio" here.
- niter: Number of iterations
- nobj: Number of objects
- pars: explicit transformations hyperparameter vector theta
- weightmat: 1-diagonal matrix (for compatibility with smacof classes)
- parameters, pars, theta: The parameters supplied
- call the call

and some additional components

- stress.m: default stress is the explicitly normalized stress on the normalized, transformed dissimilarities
- tau: tau parameter
- k: k parameter

Author(s)

Lisha Chen & Thomas Rusch

Examples

```
dis<-smacof::kinshipdelta
res<- lmds(dis,k=2,tau=0.1)
res
summary(res)
plot(res)
```

`mkBmat` *Auxfunction1*

Description

only used internally

Usage

```
mkBmat(x)
```

Arguments

`x`	matrix

`mkPower` *Take matrix to a power*

Description

Take matrix to a power

Usage

```
mkPower(x, r)
```

Arguments

`x`	matrix
`r`	numeric (power)

Value

a matrix

multiscale *Multiscale SMACOF*

Description

An implementation for maximum likelihood MDS aka multiscale that minimizes the multiscale stress by majorization with ratio and interval optimal scaling. Uses a repeat loop. Note that since this done via the route of r-sytress, the multiscale stress is approximate and only accuarte for kappa->0.

Usage

```
multiscale(
  delta,
  type = c("ratio", "interval"),
  weightmat,
  init = NULL,
  ndim = 2,
  acc = 1e-06,
  itmax = 10000,
  verbose = FALSE,
  kappa = 0.01,
  principal = FALSE
)
```

Arguments

`delta`	dist object or a symmetric, numeric data.frame or matrix of distances. Warning: these will get transformed to the log scale, so make sure that log(delta)>=0.
`type`	what optimal scaling type of MDS to fit. Currently one of "ratio" or "interval". Default is "ratio".
`weightmat`	a matrix of finite weights
`init`	starting configuration
`ndim`	dimension of the configuration; defaults to 2
`acc`	numeric accuracy of the iteration. Default is 1e-6.
`itmax`	maximum number of iterations. Default is 10000.
`verbose`	should iteration output be printed; if > 0 then yes
`kappa`	As this is not exactly multiscale but an r-stress approximation, we have multiscale only for kappa->0. This argument can therefore be used to make the approximation more accurate by making it smaller. Default is 0.1.
`principal`	If 'TRUE', principal axis transformation is applied to the final configuration

Value

a 'smacofP' object (inheriting from 'smacofB', see smacofSym). It is a list with the components

- delta: Observed dissimilarities
- tdelta: Observed explicitly transformed (log) dissimilarities, normalized
- dhat: Explicitly transformed dissimilarities (dhats), optimally scaled and normalized
- confdist: Transformed configuration distances
- conf: Matrix of fitted configuration
- stress: Default stress (stress 1; sqrt of explicitly normalized stress)
- spp: Stress per point
- ndim: Number of dimensions
- model: Name of smacof model
- niter: Number of iterations
- nobj: Number of objects
- type: Type of MDS model
- weightmat: weighting matrix
- stress.m: Default stress (stress-1^2)

Warning

The input delta will internally get transformed to the log scale, so make sure that log(delta)>=0 otherwise it throws an error. It is often a good idea to use 1+delta in this case.

See Also

rStressMin

Examples

```
dis<-smacof::kinshipdelta
res<-multiscale(as.matrix(dis),type="interval",itmax=1000)
res
summary(res)
plot(res)
```

multistart *Multistart MDS function*

Description

For different starting configurations, this function fits a series of PS models given in object or call and returns the one with the lowest stress overall. The starting configuirations can be supplied or are generated internally.

Usage

```
multistart(
  object,
  mdscall = NULL,
  ndim = 2,
  conflist,
  nstarts = 108,
  return.all = FALSE,
  verbose = TRUE,
  min = -5,
  max = 5
)
```

Arguments

`object`	A fitted object of class 'smacofP', 'smacofB' or 'smacof'. If supplied this takes precedence over the call argument. If given this is added to the output and may be the optimal one.
`mdscall`	Alternatively to a fitted object, one can pass a syntactically valid call for any of the MDS functions cops, stops or smacof that find a configuration (not the ones that do parameter selection like pcops or stops). If object and call is given, object takes precedence.
`ndim`	Dimensions of target space.
`conflist`	Optional list of starting configurations.
`nstarts`	If conflist is not supplied, how many random starting configurations should be used. The default is 108, which implies that at least one of the stress is within the lowest 1 percent of all stresses with probability of 1/3 or within the lowest 5 percent of stresses with probability 0.996
`return.all`	Should all fitted MDS be returned. If FALSE (default) only the optimal one is returned.
`verbose`	If >0 prints the fitting progress.
`min`	lower bound for the uniform distribution to sample from
`max`	upper bound for the uniform distribution to sample from

Details

If no configuration list is supplied, then nstarts configurations are simulated. They are drawn from a ndim-dimesnional uniform distribution with minimum min and maximum max. We recommend to use the route via supplying a fitted model as these are typically starting from a Torgerson configuration and are likely quite good.

One can simply extract $best and save that and work with it right away.

Value

if 'return.all=FALSE', a list with the best fitted model as '$best' (minimal badness-of-fit of all fitted models) and '$stressvec' the stresses of all models. If 'return.all=TRUE' a list with slots

- best: The object resulting from the fit that had the overall lowest objective function value (usually stress)
- stressvec: The vector of objective function values
- models: A list of all the fitted objects.

Examples

```
dis<-smacof::kinshipdelta

## Version 1: Using a fitted object (recommended)
res1<-rStressMin(delta=dis,type="ordinal",itmax=100)
resm<-multistart(res1,nstarts=2)
## best model
res2<-resm$best
#it's starting configuration
res2$init

## Version 2: Using a call object and supplying conflist
conflist<-list(res2$init,jitter(res2$init,1))
c1 <- call("rstressMin",delta=dis,type="ordinal",itmax=100)
resm<-multistart(mdscall=c1,conflist=conflist,return.all=TRUE)
```

`opmds` *Nonlinear ratio MDS with optimal power of dissimilarities*

Description

An implementation to minimize explicitly normalized stress over dissimilarities to a power by majorization with ratio optimal scaling in an alternating minimization algorithm. The optimal power transformation lambda of the dissimilarities is found by an inner optimization step via the Brent-Dekker method.

Usage

```
opmds(
  delta,
  type = "ratio",
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  ndim = 2,
  itmax = 1000,
  acc = 1e-10,
  verbose = FALSE,
  principal = FALSE,
  interval = c(0, 4)
)
```

Arguments

`delta`	dist object or a symmetric, numeric data.frame or matrix of distances
`type`	what type of MDS to fit. Currently only "ratio".
`weightmat`	a matrix of finite weights.
`init`	starting configuration
`ndim`	dimension of the configuration; defaults to 2
`itmax`	maximum number of iterations. Default is 10000.
`acc`	numeric accuracy of the iteration. Default is 1e-6.
`verbose`	should iteration output be printed; defaults to 'FALSE'.
`principal`	If 'TRUE', principal axis transformation is applied to the final configuration.
`interval`	the line constraints c(upper, lower), within which to look for the optimal power transformation lambda. Defaults to c(0,4).

Value

a 'smacofP' object (inheriting from 'smacofB', see `smacofSym`). It is a list with the components

- delta: Observed, untransformed dissimilarities
- tdelta: Observed explicitly transformed dissimilarities, normalized
- dhat: Explicitly transformed dissimilarities (dhats), optimally scaled and normalized
- confdist: Transformed fitted configuration distances
- iord: optimal scaling ordering
- conf: Matrix of fitted configuration
- stress: Default stress (stress 1; sqrt of explicitly normalized stress)
- spp: Stress per point
- ndim: Number of dimensions
- model: Name of smacof model
- niter: Number of iterations

- nobj: Number of objects
- type: Type of MDS model
- weightmat: weighting matrix as supplied
- stress.m: Default stress (stress-1^2)
- tweightmat: transformed weighting matrix (here NULL)
- pars, theta: The optimal transformation parameter lambda

See Also

See `stops` for a similar, more flexible idea.

Examples

```
dis<-smacof::kinshipdelta
res<-opmds(dis,itmax=1000)
res
summary(res)
plot(res)
```

`pdist` *Squared p-distances*

Description

Squared p-distances

Usage

```
pdist(x, p)
```

Arguments

`x`	numeric matrix
`p`	p>0 the Minkoswki distance

Value

squared Minkowski distance matrix

permtest.smacofP — *Permutation test for smacofP objects*

Description

Performs a permutation test on an MDS solution. It works with a smacofP object alone and also for derived dissimilarities, i.e. generated by the call dist(data). The original data matrix needs to be provided, as well as the type of dissimilarity measure used to compute the input dissimilarities.

Usage

```
## S3 method for class 'smacofP'
permtest(
  object,
  data,
  method.dat = "pearson",
  nrep = 100,
  verbose = FALSE,
  ...
)
```

Arguments

`object`	Object of class smacofP if used as method or another object inheriting from smacof (needs to be called directly as permtest.smacofP then).
`data`	Optional: Initial data; if provided permutations are performed on the data matrix (see details)
`method.dat`	Dissimilarity computation used as MDS input. This must be one of "pearson", "spearman", "kendall", "euclidean", "maximum", "manhattan", "canberra", "binary". If data is provided, then this must be provided as well.
`nrep`	Number of permutations.
`verbose`	If TRUE, bootstrap index is printed out.
`...`	Additional arguments needed for dissimilarity computation as specified in `sim2diss`.

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

An object of class 'smacofPerm', see `permtest` for details and methods. It has values

- 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

Examples

```
##see ?smacof::permtest for more
## permuting the dissimilarity matrix (full)
#' data(kinshipdelta)
fitkin <- rStressMin(kinshipdelta, ndim = 2, r=0.5,itmax=10) #use higher itmax
set.seed(222)
res.perm <- permtest(fitkin,nrep=5) #use higher nrep in reality
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 <- alscal(diss1,type="interval",itmax=10) #use higher itmax
fitgop1
set.seed(123)
permtest(fitgop1, GOPdtm[,1:10], nrep = 5, method.dat = "binary")
```

plot.smacofP — *S3 plot method for smacofP objects*

Description

S3 plot method for smacofP objects

Usage

```
## S3 method for class 'smacofP'
plot(
  x,
  plot.type = "confplot",
  plot.dim = c(1, 2),
  bubscale = 1,
  col,
  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",
    cex = 0.5,
    pch = 20,
    asp = 1,
    main,
    xlab,
    ylab,
    xlim,
    ylim,
    col.hist = NULL,
    legend = TRUE,
    legpos,
    loess = TRUE,
    shepard.lin = TRUE,
    ...
  )
```

Arguments

`x`	an object of class smacofP
`plot.type`	String indicating which type of plot to be produced: "confplot", "resplot", "Shepard", "stressplot","transplot", "bubbleplot" (see details)
`plot.dim`	dimensions to be plotted in confplot; defaults to c(1, 2)
`bubscale`	Scaling factor (size) for the bubble plot
`col`	vector of colors for the 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, label color)
`hull.conf`	Option to add convex hulls to a configuration plot. Hull index needs to be provided.
`shepard.x`	Shepard plot only: original data (e.g. correlation matrix) can be provided for plotting on x-axis
`identify`	If 'TRUE', the 'identify()' function is called internally that allows to add configuration labels by mouse click
`type`	What type of plot should be drawn (see also 'plot')
`cex`	Symbol size.
`pch`	Plot symbol
`asp`	Aspect ratio; defaults to 1 so distances between x and y are represented accurately; can lead to slighlty weird looking plots if the variance on one axis is much smaller than on the other axis; use NA if the standard type of R plot is wanted where the ylim and xlim arguments define the aspect ratio - but then the distances seen are no longer accurate
`main`	plot title
`xlab`	label of x axis

`ylab`	label of y axis
`xlim`	scale of x axis
`ylim`	scale of y axis
`col.hist`	Color of the borders of the histogram.
`legend`	Flag whether legends should be drawn for plots that have legends
`legpos`	Position of legend in plots with legends
`loess`	if TRUE a loess fit (by Tukey's rescending M-Estimator) of configuration distances explained by delta is added to the Shepard plot
`shepard.lin`	Shepard plot only: if TRUE the Shepard plot is linearized so d^kappa~delta^lambda. If FALSE d~delta^lambda
`...`	Further plot arguments passed: see 'plot.smacof' and 'plot' for detailed information.

Details

- Configuration plot (plot.type = "confplot"): Plots the MDS configuration.
- Residual plot (plot.type = "resplot"): Plots the dhats f(T(delta)) against the transformed fitted distances T(d(X)).
- (Linearized) Shepard diagram (plot.type = "Shepard"): Is shep.lin=TRUE a diagram with the transformed observed normalized dissimilarities (T(delta) on x) against the transformed fitted distance (T(d(X) on y) as well as a loess curve and a regression line corresponding to type (linear without intercept for ratio, linear for interval and isotonic for ordinal). If shep.lin=FALSE it uses the untransformed delta. Note that the regression line corresponds to the optimal scaling results (dhat) only up to a linear transformation.
- Transformation Plot (plot.type = "transplot"): Diagram with normalized observed dissimilarities (delta, light grey) and the normalized explicitly transformed dissimilarities (T(Delta), darker) against the untransformed fitted distances (d(X)) together with a nonlinear regression curve corresponding to the explicit transformation (fitted power transformation). This is most useful for ratio models with power transformations as the transformations can be read of directly. For other MDS models and stresses, it still gives a quick way to assess how the explicit transformations worked.
- 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 function to percentages (sum is 100). The higher the contribution, the worse the fit.
- Bubble plot (plot.type = "bubbleplot"): 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 (weighted with tweightmat if exists else with weightmat). For optional arguments, see 'wtd.hist'.

Value

no return value; just plots for class 'smacofP' (see details)

Examples

```
dis<-as.matrix(smacof::kinshipdelta)
res<-powerStressMin(dis)
plot(res)
plot(res,"Shepard")
plot(res,"resplot")
plot(res,"transplot")
plot(res,"stressplot")
plot(res,"bubbleplot")
plot(res,"histogram")
```

`powerStressFast` *Power stress minimization by NEWUOA (nloptr)*

Description

An implementation to minimize power stress by a derivative-free trust region optimization algorithm (NEWUOA). Much faster than majorizing as used in powerStressMin but perhaps less accurate.

Usage

```
powerStressFast(
  delta,
  kappa = 1,
  lambda = 1,
  nu = 1,
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  ndim = 2,
  acc = 1e-06,
  itmax = 10000,
  verbose = FALSE
)
```

Arguments

`delta`	dist object or a symmetric, numeric data.frame or matrix of distances
`kappa`	power of the transformation of the fitted distances; defaults to 1
`lambda`	the power of the transformation of the proximities; defaults to 1
`nu`	the power of the transformation for weightmat; defaults to 1
`weightmat`	a matrix of finite weights
`init`	starting configuration
`ndim`	dimension of the configuration; defaults to 2
`acc`	The smallest value of the trust region radius that is allowed. If not defined, then 1e-6 will be used.

`itmax`	maximum number of iterations. Default is 10000.
`verbose`	should iteration output be printed; if > 1 then yes

Value

a 'smacofP' object (inheriting from 'smacofB', see `smacofSym`). It is a list with the components

- delta: Observed dissimilarities, not normalized
- obsdiss: Observed dissimilarities, normalized
- confdist: Configuration dissimilarities, NOT normalized
- conf: Matrix of fitted configuration, NOT normalized
- stress: Default stress (stress 1, square root of the explicitly normalized stress on the normalized, transformed dissimilarities)
- spp: Stress per point (based on stress.en)
- ndim: Number of dimensions
- model: Name of smacof model
- niter: Number of iterations
- nobj: Number of objects
- type: Type of MDS model

and some additional components

- gamma: Empty
- stress.m: default stress for the COPS and STOP. Defaults to the explicitly normalized stress on the normalized, transformed dissimilarities
- stress.en: explicitly stress on the normalized, transformed dissimilarities and normalized transformed distances
- deltaorig: observed, untransformed dissimilarities
- weightmat: weighting matrix

See Also

`smacofSym`

Examples

```
dis<-smacof::kinshipdelta
res<-powerStressFast(as.matrix(dis),kappa=2,lambda=1.5)
res
summary(res)
plot(res)
```

powerStressMin *Power Stress SMACOF*

Description

An implementation to minimize power stress by majorization with ratio or interval optimal scaling. Usually more accurate but slower than powerStressFast. Uses a repeat loop.

Usage

```
powerStressMin(
  delta,
  kappa = 1,
  lambda = 1,
  nu = 1,
  type = "ratio",
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  ndim = 2,
  acc = 1e-06,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE
)

powerstressMin(
  delta,
  kappa = 1,
  lambda = 1,
  nu = 1,
  type = "ratio",
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  ndim = 2,
  acc = 1e-06,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE
)

postmds(
  delta,
  kappa = 1,
  lambda = 1,
  nu = 1,
  type = "ratio",
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  ndim = 2,
  acc = 1e-06,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE
)

pstressMin(
  delta,
  kappa = 1,
  lambda = 1,
  nu = 1,
  type = "ratio",
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  ndim = 2,
  acc = 1e-06,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE
)

pStressMin(
  delta,
  kappa = 1,
  lambda = 1,
  nu = 1,
  type = "ratio",
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  ndim = 2,
  acc = 1e-06,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE
)

pstressmds(
  delta,
  kappa = 1,
  lambda = 1,
  nu = 1,
  type = "ratio",
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  ndim = 2,
  acc = 1e-06,
    itmax = 10000,
    verbose = FALSE,
    principal = FALSE
  )
```

Arguments

`delta`	dist object or a symmetric, numeric data.frame or matrix of distances
`kappa`	power of the transformation of the fitted distances; defaults to 1
`lambda`	the power of the transformation of the proximities; defaults to 1
`nu`	the power of the transformation for weightmat; defaults to 1
`type`	what type of MDS to fit. One of "ratio" or "interval". Default is "ratio".
`weightmat`	a matrix of finite weights or dist object
`init`	starting configuration
`ndim`	dimension of the configuration; defaults to 2
`acc`	numeric accuracy of the iteration. Default is 1e-6.
`itmax`	maximum number of iterations. Default is 10000.
`verbose`	should internal messages be printed; if > 0 then yes (iteration progress with >2)
`principal`	If 'TRUE', principal axis transformation is applied to the final configuration

Value

a 'smacofP' object (inheriting from 'smacofB', see `smacofSym`). It is a list with the components

- delta: Observed, untransformed dissimilarities
- tdelta: Observed explicitly transformed dissimilarities, normalized
- dhat: Explicitly transformed dissimilarities (dhats), optimally scaled and normalized
- confdist: Transformed fitted configuration distances
- conf: Matrix of fitted configuration
- stress: Default stress (stress 1; sqrt of explicitly normalized stress)
- spp: Stress per point
- ndim: Number of dimensions
- model: Name of smacof model
- niter: Number of iterations
- nobj: Number of objects
- type: Type of MDS model
- weightmat: weighting matrix as supplied
- stress.m: Default stress (stress-1^2)
- tweightmat: transformed weighthingmatrix (here weightmat^nu)

See Also

`smacofSym`

Examples

```
dis<-smacof::kinshipdelta
res<-powerStressMin(dis,type="ratio",kappa=2,lambda=1.5,itmax=1000)
res
summary(res)
plot(res)
```

`procruster` *procruster: a procrustes function*

Description

procruster: a procrustes function

Usage

```
procruster(x)
```

Arguments

`x`	numeric matrix

Value

a matrix

`ringdat` *Two interlocking rings*

Description

Artifical data of two interlocking rings in 3D space. The first three columns are the coordinates and the last column is a color designation.

Format

A 500 x 4 matrix.

`rpowerStressMin` *Restricted Power Stress SMACOF*

Description

An implementation to minimize restricted power stress by majorization with ratio or interval optimal scaling. Restricted means that the same power is used for both dissimilarities and fitted distances. Uses a repeat loop.

Usage

```
rpowerStressMin(
  delta,
  expo = 1,
  nu = 1,
  type = "ratio",
  weightmat,
  init = NULL,
  ndim = 2,
  acc = 1e-06,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE
)

rpowerstressMin(
  delta,
  expo = 1,
  nu = 1,
  type = "ratio",
  weightmat,
  init = NULL,
  ndim = 2,
  acc = 1e-06,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE
)

rpostmds(
  delta,
  expo = 1,
  nu = 1,
  type = "ratio",
  weightmat,
  init = NULL,
  ndim = 2,
  acc = 1e-06,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE
)

rpstressMin(
  delta,
  expo = 1,
  nu = 1,
  type = "ratio",
  weightmat,
  init = NULL,
  ndim = 2,
  acc = 1e-06,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE
)

rpStressMin(
  delta,
  expo = 1,
  nu = 1,
  type = "ratio",
  weightmat,
  init = NULL,
  ndim = 2,
  acc = 1e-06,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE
)

rpstressmds(
  delta,
  expo = 1,
  nu = 1,
  type = "ratio",
  weightmat,
  init = NULL,
  ndim = 2,
  acc = 1e-06,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE
)
```

Arguments

`delta`	dist object or a symmetric, numeric data.frame or matrix of distances
`expo`	power of the transformation of the fitted distances and dissimilarities; defaults to 1
`nu`	the power of the transformation for weightmat; defaults to 1
`type`	what type of MDS to fit. One of "ratio" or "interval". Default is "ratio".
`weightmat`	a matrix of finite weights
`init`	starting configuration
`ndim`	dimension of the configuration; defaults to 2
`acc`	numeric accuracy of the iteration. Default is 1e-6.
`itmax`	maximum number of iterations. Default is 10000.
`verbose`	should fitting information be printed; if > 0 then yes
`principal`	If ’TRUE’, principal axis transformation is applied to the final configuration

Value

a ’smacofP’ object (inheriting from ’smacofB’, see `smacofSym`). It is a list with the components

- delta: Observed, untransformed dissimilarities
- tdelta: Observed explicitly transformed dissimilarities, normalized
- dhat: Explicitly transformed dissimilarities (dhats), optimally scaled and normalized
- confdist: Transformed configuration distances
- conf: Matrix of fitted configuration
- stress: Default stress (stress 1; sqrt of explicitly normalized stress)
- spp: Stress per point
- ndim: Number of dimensions
- model: Name of smacof model
- niter: Number of iterations
- nobj: Number of objects
- type: Type of MDS model
- weightmat: weighting matrix as supplied
- stress.m: Default stress (stress-1^2)
- tweightmat: transformed weighthing matrix (here weightmat^nu)
- parameters, pars, theta: The parameter vector of the explicit transformations

Examples

```
dis<-smacof::kinshipdelta
res<-rpowerStressMin(as.matrix(dis),expo=1.7,itmax=1000)
res
summary(res)
plot(res)
```

rStressMin *R stress SMACOF*

Description

An implementation to minimize r-stress by majorization with ratio, interval, monotonic spline and ordinal optimal scaling. Uses a repeat loop.

Usage

```
rStressMin(
  delta,
  r = 0.5,
  type = c("ratio", "interval", "ordinal", "mspline"),
  ties = "primary",
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  ndim = 2,
  acc = 1e-06,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE,
  spline.degree = 2,
  spline.intKnots = 2
)

rstressMin(
  delta,
  r = 0.5,
  type = c("ratio", "interval", "ordinal", "mspline"),
  ties = "primary",
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  ndim = 2,
  acc = 1e-06,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE,
  spline.degree = 2,
  spline.intKnots = 2
)

rstressmds(
  delta,
  r = 0.5,
  type = c("ratio", "interval", "ordinal", "mspline"),
  ties = "primary",
    weightmat = 1 - diag(nrow(delta)),
    init = NULL,
    ndim = 2,
    acc = 1e-06,
    itmax = 10000,
    verbose = FALSE,
    principal = FALSE,
    spline.degree = 2,
    spline.intKnots = 2
  )

  rstress(
    delta,
    r = 0.5,
    type = c("ratio", "interval", "ordinal", "mspline"),
    ties = "primary",
    weightmat = 1 - diag(nrow(delta)),
    init = NULL,
    ndim = 2,
    acc = 1e-06,
    itmax = 10000,
    verbose = FALSE,
    principal = FALSE,
    spline.degree = 2,
    spline.intKnots = 2
  )
```

Arguments

`delta`	dist object or a symmetric, numeric data.frame or matrix of distances
`r`	power of the transformation of the fitted distances (corresponds to kappa/2 in power stress); defaults to 0.5 for standard stress
`type`	what type of MDS to fit. Currently one of "ratio", "interval", "mspline" or "ordinal". Default is "ratio".
`ties`	the handling of ties for ordinal (nonmetric) MDS. Possible are "primary" (default), "secondary" or "tertiary".
`weightmat`	a matrix of finite weights.
`init`	starting configuration
`ndim`	dimension of the configuration; defaults to 2
`acc`	numeric accuracy of the iteration. Default is 1e-6.
`itmax`	maximum number of iterations. Default is 10000.
`verbose`	should fitting information be printed; if > 0 then yes
`principal`	If 'TRUE', principal axis transformation is applied to the final configuration
`spline.degree`	Degree of the spline for 'mspline' MDS type
`spline.intKnots`	Number of interior knots of the spline for 'mspline' MDS type

Value

a 'smacofP' object (inheriting from 'smacofB', see smacofSym). It is a list with the components

- delta: Observed, untransformed dissimilarities
- tdelta: Observed explicitly transformed dissimilarities, normalized
- dhat: Explicitly transformed dissimilarities (dhats), optimally scaled and normalized
- confdist: Transformed fitted configuration distances
- iord: Optimally scaled disparities function
- conf: Matrix of fitted configuration
- stress: Default stress (stress 1; sqrt of explicitly normalized stress)
- spp: Stress per point
- ndim: Number of dimensions
- weightmat: Weighting matrix as supplied
- resmat: Residual matrix
- rss: Sum of residuals
- init: The starting configuration
- model: Name of MDS model
- niter: Number of iterations
- nobj: Number of objects
- type: Type of optimal scaling
- call : the matched call
- stress.m: Default stress (stress-1^2)
- alpha: Alpha matrix
- sigma: Stress
- parameters, pars, theta: Optimal transformation parameter
- tweightmat: Transformed weighting matrix (here NULL)

See Also

smacofSym

Examples

```
dis<-smacof::kinshipdelta

## ordinal MDS
res<-rStressMin(as.matrix(dis), type = "ordinal", r = 1, itmax = 1000)
res
summary(res)
plot(res)

## spline MDS
ress<-rStressMin(as.matrix(dis), type = "mspline", r = 1,
        itmax = 1000)
ress
plot(ress,"Shepard")
```

---

sammon — *Wrapper to* `sammon` *for S3 class*

Description

Wrapper to `sammon` for S3 class

Usage

```
sammon(d, y = NULL, k = 2, ...)
```

Arguments

`d`	a distance structure such as that returned by 'dist' or a full symmetric matrix. Data are assumed to be dissimilarities or relative distances, but must be positive except for self-distance. This can contain missing values.
`y`	An initial configuration. If NULL, `cmdscale` is used to provide the classical solution. (If there are missing values in 'd', an initial configuration must be provided.) This must not have duplicates.
`k`	The dimension of the configuration
`...`	Additional parameters passed to `sammon`, see `sammon`

Details

Overloads MASS::sammon and adds new slots and class attributes for which there are methods.

Value

Object of class 'sammonx' inheriting from `sammon`. This wrapper adds an extra slot to the list with the call, adds column labels to the $points, adds slots conf=points, delta=d, dhat=normalized dissimilarities, confdist=distance between points in conf, stress.m=stress, stress=sqrt(stress.m) and assigns S3 classes 'sammonx', 'sammon' and 'cmdscalex'.

Examples

```
dis<-as.matrix(smacof::kinshipdelta)
res<-sammon(dis)
```

sammonmap *Sammon Mapping SMACOF*

Description

An implementation to minimize Sammon stress by majorization with ratio and interval optimal scaling. Uses a repeat loop.

Usage

```
sammonmap(
  delta,
  type = c("ratio", "interval"),
  weightmat,
  init = NULL,
  ndim = 2,
  acc = 1e-06,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE
)
```

Arguments

`delta`	dist object or a symmetric, numeric data.frame or matrix of distances
`type`	what type of MDS to fit. Currently one of "ratio" and "interval". Default is "ratio".
`weightmat`	a matrix of finite weights
`init`	starting configuration
`ndim`	dimension of the configuration; defaults to 2
`acc`	numeric accuracy of the iteration. Default is 1e-6.
`itmax`	maximum number of iterations. Default is 10000.
`verbose`	should fitting information be printed; if > 0 then yes
`principal`	If 'TRUE', principal axis transformation is applied to the final configuration

Value

a 'smacofP' object (inheriting from smacofB, see `smacofSym`). It is a list with the components

- delta: Observed dissimilarities
- tdelta: Observed explicitly transformed dissimilarities, normalized
- dhat: Observed dissimilarities (dhats), optimally scaled and normalized
- confdist: Transformed configuration distances
- conf: Matrix of fitted configuration

- stress: Default stress (stress 1; sqrt of explicitly normalized stress)
- spp: Stress per point (based on stress.en)
- ndim: Number of dimensions
- model: Name of smacof model
- niter: Number of iterations
- nobj: Number of objects
- type: Type of MDS model
- weightmat: weighting matrix as supplied
- stress.m: default stress (stress-1^2)
- tweightmat: weighting matrix atfer transformation (here weightmat/delta)

See Also

`rStressMin`

Examples

```
dis<-smacof::kinshipdelta
res<-sammonmap(as.matrix(dis),itmax=1000)
res
summary(res)
plot(res)
```

`scale_adjust` *Adjusts a configuration*

Description

Adjusts a configuration

Usage

```
scale_adjust(conf, ref, scale = c("sd", "std", "proc", "none"))
```

Arguments

`conf`	a configuration
`ref`	a reference configuration (only for scale="proc")
`scale`	Scale adjustment. "std" standardizes each column of the configurations to mean=0 and sd=1, "sd" scales the configuration by the maximum standard devation of any column, "proc" adjusts the fitted configuration to the reference

Value

The scale adjusted configuration.

secularEq *Secular Equation*

Description

Secular Equation

Usage

```
secularEq(a, b)
```

Arguments

`a`	matrix
`b`	matrix

smacofxDeleteOne *Helper function to conduct jackknife MDS*

Description

Function deletes every object row and columns once and fits the MDS in object and returns the configuration. The deleted row is set to 0 in the configuration. Is meant for smacofx functions, but should also work for every smacof models.

Usage

```
smacofxDeleteOne(
  object,
  delta,
  weightmat,
  init,
  ndim,
  type,
  verbose = FALSE,
  itmaxi = 10000
)
```

Arguments

`object`	Object of class smacofP if used as method or another object inheriting from smacofB. Note we assume the MDS model was fitted on a symmetric matrix/data frame or dist object
`delta`	the data (symmetric matrix, data frame or dist object)
`weightmat`	weighting matrix

`init`	starting configuration
`ndim`	target dimension of the mds
`type`	type of MDS
`verbose`	print progress
`itmaxi`	maximum iterations of the MDS procedure

Value

An array of size n with n coonfigurations

`spheredat` *Clelia curve on a sphere*

Description

Artifical data of data sampled along a Clelia curve on a sphere. The first three columns are the coordinates and the last column is a color designation (viridis palette).

Format

A 500 x 4 matrix.

`spmdda` *Extended Curvilinear (Power) Distance Analysis (eCLPDA or eCLDA) aka Sparse (POST-)Multidimensional Distance Analysis (SP-MDDA or SMDDA) either as self-organizing or not*

Description

An implementation of a sparsified version of (POST-)MDS by pseudo-majorization with ratio, interval and ordinal optimal scaling for geodesic distances and optional power transformations. This is inspired by curvilinear distance analysis but works differently: It finds an initial weightmatrix where w_ij(X^0)=0 if d_ij(X^0)>tau and fits a POST-MDS with these weights. Then in each successive iteration step, the weightmat is recalculated so that w_ij(X^(n+1))=0 if d_ij(X^(n+1))>tau. Right now the zero weights are not found by the correct optimization, but we're working on that.

Usage

```
spmdda(
  delta,
  lambda = 1,
  kappa = 1,
  nu = 1,
  tau,
  type = "ratio",
  ties = "primary",
  epsilon,
  k,
  path = "shortest",
  fragmentedOK = FALSE,
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  ndim = 2,
  acc = 1e-08,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE,
  spline.degree = 2,
  spline.intKnots = 2,
  traceIt = FALSE
)

smdda(
  delta,
  tau = stats::quantile(delta, 0.9),
  type = "ratio",
  ties = "primary",
  epsilon,
  k,
  path = "shortest",
  fragmentedOK = FALSE,
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  ndim = 2,
  acc = 1e-08,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE,
  traceIt = FALSE,
  spline.degree = 2,
  spline.intKnots = 2
)

so_spmdda(
  delta,
    kappa = 1,
    lambda = 1,
    nu = 1,
    tau = max(delta),
    epochs = 10,
    type = c("ratio"),
    ties = "primary",
    epsilon,
    k,
    path = "shortest",
    fragmentedOK = FALSE,
    weightmat = 1 - diag(nrow(delta)),
    init = NULL,
    ndim = 2,
    acc = 1e-08,
    itmax = 10000,
    verbose = FALSE,
    principal = FALSE,
    spline.degree = 2,
    spline.intKnots = 2
  )

  so_smdda(
    delta,
    tau = max(delta),
    epochs = 10,
    type = c("ratio"),
    ties = "primary",
    epsilon,
    k,
    path = "shortest",
    fragmentedOK = FALSE,
    weightmat = 1 - diag(nrow(delta)),
    init = NULL,
    ndim = 2,
    acc = 1e-08,
    itmax = 10000,
    verbose = FALSE,
    principal = FALSE,
    spline.degree = 2,
    spline.intKnots = 2
  )

  eCLDA(
    delta,
    tau = stats::quantile(delta, 0.9),
    type = "ratio",
    ties = "primary",
  epsilon,
  k,
  path = "shortest",
  fragmentedOK = FALSE,
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  ndim = 2,
  acc = 1e-08,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE,
  traceIt = FALSE,
  spline.degree = 2,
  spline.intKnots = 2
)

eCLPDA(
  delta,
  lambda = 1,
  kappa = 1,
  nu = 1,
  tau,
  type = "ratio",
  ties = "primary",
  epsilon,
  k,
  path = "shortest",
  fragmentedOK = FALSE,
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  ndim = 2,
  acc = 1e-08,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE,
  spline.degree = 2,
  spline.intKnots = 2,
  traceIt = FALSE
)

so_eCLPDA(
  delta,
  kappa = 1,
  lambda = 1,
  nu = 1,
  tau = max(delta),
  epochs = 10,
  type = c("ratio"),
    ties = "primary",
    epsilon,
    k,
    path = "shortest",
    fragmentedOK = FALSE,
    weightmat = 1 - diag(nrow(delta)),
    init = NULL,
    ndim = 2,
    acc = 1e-08,
    itmax = 10000,
    verbose = FALSE,
    principal = FALSE,
    spline.degree = 2,
    spline.intKnots = 2
  )

  so_eCLDA(
    delta,
    tau = max(delta),
    epochs = 10,
    type = c("ratio"),
    ties = "primary",
    epsilon,
    k,
    path = "shortest",
    fragmentedOK = FALSE,
    weightmat = 1 - diag(nrow(delta)),
    init = NULL,
    ndim = 2,
    acc = 1e-08,
    itmax = 10000,
    verbose = FALSE,
    principal = FALSE,
    spline.degree = 2,
    spline.intKnots = 2
  )

  eclpda(
    delta,
    lambda = 1,
    kappa = 1,
    nu = 1,
    tau,
    type = "ratio",
    ties = "primary",
    epsilon,
    k,
    path = "shortest",
  fragmentedOK = FALSE,
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  ndim = 2,
  acc = 1e-08,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE,
  spline.degree = 2,
  spline.intKnots = 2,
  traceIt = FALSE
)

eclda(
  delta,
  tau = stats::quantile(delta, 0.9),
  type = "ratio",
  ties = "primary",
  epsilon,
  k,
  path = "shortest",
  fragmentedOK = FALSE,
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  ndim = 2,
  acc = 1e-08,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE,
  traceIt = FALSE,
  spline.degree = 2,
  spline.intKnots = 2
)

so_eclpda(
  delta,
  kappa = 1,
  lambda = 1,
  nu = 1,
  tau = max(delta),
  epochs = 10,
  type = c("ratio"),
  ties = "primary",
  epsilon,
  k,
  path = "shortest",
  fragmentedOK = FALSE,
  weightmat = 1 - diag(nrow(delta)),
    init = NULL,
    ndim = 2,
    acc = 1e-08,
    itmax = 10000,
    verbose = FALSE,
    principal = FALSE,
    spline.degree = 2,
    spline.intKnots = 2
  )

  so_eclda(
    delta,
    tau = max(delta),
    epochs = 10,
    type = c("ratio"),
    ties = "primary",
    epsilon,
    k,
    path = "shortest",
    fragmentedOK = FALSE,
    weightmat = 1 - diag(nrow(delta)),
    init = NULL,
    ndim = 2,
    acc = 1e-08,
    itmax = 10000,
    verbose = FALSE,
    principal = FALSE,
    spline.degree = 2,
    spline.intKnots = 2
  )
```

Arguments

`delta`	dist object or a symmetric, numeric data.frame or matrix of distances
`lambda`	exponent of the power transformation of the dissimilarities; defaults to 1, which is also the setup of 'smdda'
`kappa`	exponent of the power transformation of the fitted distances; defaults to 1, which is also the setup of 'smdda'.
`nu`	exponent of the power of the weighting matrix; defaults to 1 which is also the setup for 'clca'.
`tau`	the boundary/neighbourhood parameter(s) (called lambda in the original paper). For 'spmdda' and 'smdda' it is supposed to be a numeric scalar (if a sequence is supplied the maximum is taken as tau) and all the transformed fitted distances exceeding tau are set to 0 via the weightmat (assignment can change between iterations). It defaults to the 90% quantile of the enormed (power transformed) geodesic distances of delta. For 'so_pclca' tau is supposed to be either a user supplied decreasing sequence of taus or if a scalar the maxi-

	mum tau from which a decreasing sequence of taus is generated automatically as 'seq(from=tau,to=tau/epochs,length.out=epochs)' and then used in sequence.
`type`	what type of MDS to fit. Currently one of "ratio", "interval","ordinal" or "mspline". Default is "ratio".
`ties`	the handling of ties for ordinal (nonmetric) MDS. Possible are "primary" (default), "secondary" or "tertiary".
`epsilon`	Shortest dissimilarity retained.
`k`	Number of shortest dissimilarities retained for a point. If both 'epsilon' and 'k' are given, 'epsilon' will be used.
`path`	Method used in 'stepacross' to estimate the shortest path, with alternatives '"shortest"' and '"extended"'.
`fragmentedOK`	What to do if dissimilarity matrix is fragmented. If 'TRUE', analyse the largest connected group, otherwise stop with error.
`weightmat`	a matrix of finite weights.
`init`	starting configuration
`ndim`	dimension of the configuration; defaults to 2
`acc`	numeric accuracy of the iteration. Default is 1e-8.
`itmax`	maximum number of iterations. Default is 10000.
`verbose`	should fitting infomation be printed; if > 0 then yes
`principal`	If 'TRUE', principal axis transformation is applied to the final configuration
`spline.degree`	Degree of the spline for 'mspline' MDS type
`spline.intKnots`	Number of interior knots of the spline for 'mspline' MDS type
`traceIt`	save the iteration progress in a vector (stress values)
`epochs`	for 'so_pclca' and tau being scalar, it gives the number of passes through the data. The sequence of taus created is 'seq(tau,tau/epochs,length.out=epochs)'. If tau is of length >1, this argument is ignored.

Details

In 'spmdda' the logic is that we first transform to geodesic distance, then apply the explicit power transformation and then the implicit optimal scaling. There is a wrapper 'smdda', 'eCLDA' where the exponents are 1, which is standard SMDDA or eCLPDA but extend to allow optimal scaling. The neighborhood parameter tau is kept fixed in 'spmdda', 'eCLPDA' and 'smdda', 'eCLDA'. The functions 'so_spmdda', 'so_eCLPDA' and 'so_smdda', 'so_eCLDA' implement a self-organising principle where the is repeatedly fitted for a decreasing sequence of taus.

The solution is found by "quasi-majorization", which mean that the majorization is only working properly after a burn-in of a few iterations when the assignment which distances are ignored no longer changes. Due to that it can be that in the beginning the stress may not decrease monotonically and that there's a chance it might never.

The geodesic distances are calculated via 'vegan::isomapdist', see `isomapdist` for a documentation of what these distances do. The functions of '(p)smdda' are just a wrapper for '(p)clca' applied to the geodesic distances obtained via isomapdist.

If tau is too small it may happen that all distances for one i to all j are zero and then there will be an error, so make sure to set a larger tau.

In the standard functions 'spmdda' and 'smdda' we keep tau fixed throughout. This means that if tau is large enough, then the result is the same as the corresponding MDS. In the orginal publication the idea was that of a self-organizing map which decreased tau over epochs (i.e., passes through the data). This can be achieved with our function 'so_spmdda' 'so_smdda' which creates a vector of decreasing tau values, calls the function 'spmdda' with the first tau, then supplies the optimal configuration obtained as the init for the next call with the next tau and so on.

Value

a 'smacofP' object (inheriting from 'smacofB', see `smacofSym`). It is a list with the components

- delta: Observed, untransformed dissimilarities
- tdelta: Observed explicitly transformed dissimilarities, normalized
- dhat: Explicitly transformed dissimilarities (dhats), optimally scaled and normalized
- confdist: Configuration dissimilarities
- conf: Matrix of fitted configuration
- stress: Default stress (stress 1; sqrt of explicitly normalized stress)
- spp: Stress per point
- ndim: Number of dimensions
- model: Name of smacof model
- niter: Number of iterations
- nobj: Number of objects
- type: Type of MDS model
- weightmat: weighting matrix as supplied
- stress.m: Default stress (stress-1^2)
- tweightmat: transformed weighting matrix; it is weightmat but containing all the 0s for the distances set to 0.
- trace: if 'traceIt=TRUE' a vector with the iteration progress

Examples

```
dis<-smacof::morse
res<-spmdda(dis,kappa=2,lambda=2,tau=0.4,k=5,itmax=500) #use higher itmax
res
#already many parameters
coef(res)

res2<-smdda(dis,type="interval",tau=0.4,epsilon=1,itmax=500) #use higher itmax
#aliases:
resa<-eCLPDA(dis,kappa=2,lambda=2,tau=0.4,k=5,itmax=500) #use higher itmax
res2a<-eCLDA(dis,type="interval",tau=0.4,epsilon=1,itmax=500) #use higher itmax

res2
```

```
summary(res)
oldpar<-par(mfrow=c(1,2))
plot(res)
plot(res2)
par(oldpar)

##which d_{ij}(X) exceeded tau at convergence (i.e., have been set to 0)?
res$tweighmat
res2$tweightmat


## Self-organizing map style (as in the original publication)
#run the som-style (p)smdda
sommod1<-so_spmdda(dis,tau=2,k=5,kappa=0.5,lambda=2,epochs=10,verbose=1)
sommod2<-so_smdda(dis,tau=2.5,epsilon=1,epochs=10,verbose=1)
sommod1
sommod2
```

---

spmds *Extended Curvilinear (Power) Component Analysis aka Sparsified (POST-) Multidimensional Scaling (SPMDS or SMDS) either as self-organizing or not*

Description

An implementation of extended CLPCA which is a sparsified version of (POST-)MDS by quasi-majorization with ratio, interval and ordinal optimal scaling for dissimilarities and optional power transformations. This is inspired by curvilinear component analysis but works differently: It finds an initial weightmatrix where w_ij(X^0)=0 if d_ij(X^0)>tau and fits a POST-MDS with these weights. Then in each successive iteration step, the weightmat is recalculated so that w_ij(X^(n+1))=0 if d_ij(X^(n+1))>tau.

Usage

```
spmds(
  delta,
  lambda = 1,
  kappa = 1,
  nu = 1,
  tau,
  type = "ratio",
  ties = "primary",
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  ndim = 2,
  acc = 1e-08,
    itmax = 10000,
    verbose = FALSE,
    principal = FALSE,
    spline.degree = 2,
    spline.intKnots = 2,
    traceIt = FALSE
  )

  smds(
    delta,
    tau,
    type = "ratio",
    ties = "primary",
    weightmat = 1 - diag(nrow(delta)),
    init = NULL,
    ndim = 2,
    acc = 1e-08,
    itmax = 10000,
    verbose = FALSE,
    principal = FALSE,
    traceIt = FALSE,
    spline.degree = 2,
    spline.intKnots = 2
  )

  so_spmds(
    delta,
    kappa = 1,
    lambda = 1,
    nu = 1,
    tau = max(delta),
    epochs = 10,
    type = "ratio",
    ties = "primary",
    weightmat = 1 - diag(nrow(delta)),
    init = NULL,
    ndim = 2,
    acc = 1e-08,
    itmax = 10000,
    verbose = FALSE,
    principal = FALSE,
    spline.degree = 2,
    spline.intKnots = 2
  )

  so_smds(
    delta,
    tau = max(delta),
  epochs = 10,
  type = "ratio",
  ties = "primary",
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  ndim = 2,
  acc = 1e-08,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE,
  spline.degree = 2,
  spline.intKnots = 2
)

eCLCA(
  delta,
  tau,
  type = "ratio",
  ties = "primary",
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  ndim = 2,
  acc = 1e-08,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE,
  traceIt = FALSE,
  spline.degree = 2,
  spline.intKnots = 2
)

eCLPCA(
  delta,
  lambda = 1,
  kappa = 1,
  nu = 1,
  tau,
  type = "ratio",
  ties = "primary",
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  ndim = 2,
  acc = 1e-08,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE,
  spline.degree = 2,
  spline.intKnots = 2,
    traceIt = FALSE
  )

  so_eCLPCA(
    delta,
    kappa = 1,
    lambda = 1,
    nu = 1,
    tau = max(delta),
    epochs = 10,
    type = "ratio",
    ties = "primary",
    weightmat = 1 - diag(nrow(delta)),
    init = NULL,
    ndim = 2,
    acc = 1e-08,
    itmax = 10000,
    verbose = FALSE,
    principal = FALSE,
    spline.degree = 2,
    spline.intKnots = 2
  )

  so_eCLCA(
    delta,
    tau = max(delta),
    epochs = 10,
    type = "ratio",
    ties = "primary",
    weightmat = 1 - diag(nrow(delta)),
    init = NULL,
    ndim = 2,
    acc = 1e-08,
    itmax = 10000,
    verbose = FALSE,
    principal = FALSE,
    spline.degree = 2,
    spline.intKnots = 2
  )

  eclca(
    delta,
    tau,
    type = "ratio",
    ties = "primary",
    weightmat = 1 - diag(nrow(delta)),
    init = NULL,
    ndim = 2,
  acc = 1e-08,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE,
  traceIt = FALSE,
  spline.degree = 2,
  spline.intKnots = 2
)

eclpca(
  delta,
  lambda = 1,
  kappa = 1,
  nu = 1,
  tau,
  type = "ratio",
  ties = "primary",
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  ndim = 2,
  acc = 1e-08,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE,
  spline.degree = 2,
  spline.intKnots = 2,
  traceIt = FALSE
)

so_eclca(
  delta,
  tau = max(delta),
  epochs = 10,
  type = "ratio",
  ties = "primary",
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  ndim = 2,
  acc = 1e-08,
  itmax = 10000,
  verbose = FALSE,
  principal = FALSE,
  spline.degree = 2,
  spline.intKnots = 2
)

so_eclpca(
  delta,
    kappa = 1,
    lambda = 1,
    nu = 1,
    tau = max(delta),
    epochs = 10,
    type = "ratio",
    ties = "primary",
    weightmat = 1 - diag(nrow(delta)),
    init = NULL,
    ndim = 2,
    acc = 1e-08,
    itmax = 10000,
    verbose = FALSE,
    principal = FALSE,
    spline.degree = 2,
    spline.intKnots = 2
)
```

Arguments

`delta`	dist object or a symmetric, numeric data.frame or matrix of distances
`lambda`	exponent of the power transformation of the dissimilarities; defaults to 1, which is also the setup of 'smds'
`kappa`	exponent of the power transformation of the fitted distances; defaults to 1, which is also the setup of 'smds'.
`nu`	exponent of the power of the weighting matrix; defaults to 1 which is also the setup for 'smds'.
`tau`	the boundary/neighbourhood parameter(s) (called lambda in the original paper). For 'spmds' and 'smds' it is supposed to be a numeric scalar (if a sequence is supplied the maximum is taken as tau) and all the transformed fitted distances exceeding tau are set to 0 via the weightmat (assignment can change between iterations). It defaults to the 1% quantile of delta. For 'so_spmds' tau is supposed to be either a user supplied decreasing sequence of taus or if a scalar the maximum tau from which a decreasing sequence of taus is generated automatically as 'seq(from=tau,to=tau/epochs,length.out=epochs)' and then used in sequence.
`type`	what type of MDS to fit. Currently one of "ratio", "interval", "mspline" or "ordinal". Default is "ratio".
`ties`	the handling of ties for ordinal (nonmetric) MDS. Possible are "primary" (default), "secondary" or "tertiary".
`weightmat`	a matrix of finite weights.
`init`	starting configuration. If NULL (default) we fit a full rstress model.
`ndim`	dimension of the configuration; defaults to 2
`acc`	numeric accuracy of the iteration. Default is 1e-8.
`itmax`	maximum number of iterations. Default is 10000.
`verbose`	should fitting information be printed; if > 0 then yes

`principal`	If 'TRUE', principal axis transformation is applied to the final configuration
`spline.degree`	Degree of the spline for 'mspline' MDS type
`spline.intKnots`	Number of interior knots of the spline for 'mspline' MDS type
`traceIt`	save the iteration progress in a vector (stress values)
`epochs`	for 'so_spmds' and tau being scalar, it gives the number of passes through the data. The sequence of taus created is 'seq(tau,tau/epochs,length.out=epochs)'. If tau is of length >1, this argument is ignored.

Details

There are a wrappers 'smds' and 'eCLCA' where the exponents are 1. The neighborhood parameter tau is kept fixed in 'spmds', 'smds', 'eCLCA' and 'eCLPCA'. The functions 'so_spmds', 'so_eCLPCA' and 'so_smds', 'so_eCLCA' implement a self-organising principle, where the model is repeatedly fitted for a decreasing sequence of taus.

The solution is found by "quasi-majorization", which means that the majorization is only real majorization once the weightmat no longer changes. This typically happens after a few iterations. Due to that it can be that in the beginning the stress may not decrease monotonically and that there's a chance it might never.

If tau is too small it may happen that all distances for one i to all j are zero and then there will be an error, so make sure to set a larger tau.

In the standard functions 'spmds' and 'smds' we keep tau fixed throughout. This means that if tau is large enough, then the result is the same as the corresponding MDS. In the orginal publication the idea was that of a self-organizing map which decreased tau over epochs (i.e., passes through the data). This can be achieved with our function 'so_spmds' 'so_smds' which creates a vector of decreasing tau values, calls the function 'spmds' with the first tau, then supplies the optimal configuration obtained as the init for the next call with the next tau and so on.

Value

a 'smacofP' object (inheriting from 'smacofB', see `smacofSym`). It is a list with the components

- delta: Observed, untransformed dissimilarities
- tdelta: Observed explicitly transformed dissimilarities, normalized
- dhat: Explicitly transformed dissimilarities (dhats), optimally scaled and normalized
- confdist: Transformed configuration distances
- conf: Matrix of fitted configuration
- stress: Default stress (stress 1; sqrt of explicitly normalized stress)
- spp: Stress per point
- ndim: Number of dimensions
- model: Name of smacof model
- niter: Number of iterations
- nobj: Number of objects
- type: Type of MDS model

- weightmat: weighting matrix as supplied
- stress.m: Default stress (stress-1^2)
- tweightmat: transformed weighting matrix; it is weightmat but containing all the 0s for the distances set to 0.
- trace: if 'traceIt=TRUE' a vector with the iteration progress

Examples

```
dis<-smacof::morse
res<-spmds(dis,type="interval",kappa=2,lambda=2,tau=0.3,itmax=100) #use higher itmax
res2<-smds(dis,type="interval",tau=0.3,itmax=500,traceIt=TRUE) #use higher itmax
#Aliases
resa<-eCLPCA(dis,type="interval",kappa=2,lambda=2,tau=0.3,itmax=100) #use higher itmax
res2a<-eCLCA(dis,type="interval",tau=0.3,itmax=500,traceIt=TRUE) #use higher itmax

res
res2
summary(res)
oldpar<-par(mfrow=c(1,2))
plot(res)
plot(res2)
par(oldpar)

##which d_{ij}(X)^kappa exceeded tau at convergence (i.e., have been set to 0)?
res$tweightmat
res2$tweightmat

# We use Quasi-Majorization
res2$trace



## Self-organizing map style (as in the clca publication)
#run the som-style (p)smds
sommod1<-so_spmds(dis,tau=1,kappa=0.5,lambda=2,epochs=10,verbose=1)
sommod2<-so_smds(dis,tau=1,epochs=10,verbose=1)
sommod1
sommod2
```

---

spp *Calculating stress per point*

Description

Calculating stress per point

Usage

```
spp(dhat, confdist, weightmat)
```

Arguments

`dhat`	a dist object or symmetric matrix of dissimilarities
`confdist`	a dist object or symmetric matrix of fitted distances
`weightmat`	dist objetc or symmetric matrix of weights

Value

a list

`sqdist` *Squared distances*

Description

Squared distances

Usage

```
sqdist(x)
```

Arguments

`x`	numeric matrix

Value

squared distance matrix

`udat` *Noisy Data on a U Fold*

Description

Artifical data of data on a 3D U fold with noise that increases towards the edges. The first three columns are the coordinates and the last column is a color designation (viridis palette).

Format

A 500 x 4 matrix.

Index