Package 'stops'

January 20, 2023

Title Structure Optimized Proximity Scaling

Version 1.0-1

Date 2023-01-18

Author Thomas Rusch [aut, cre],
Jan de Leeuw [aut],
Lisha Chen [aut],
Patrick Mair [aut],
Kurt Hornik [ctb]

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

Description A collection of methods that fit nonlinear distance transformations in multidimensional scaling (MDS) and trade-off the fit with structure considerations to find optimal parameters also known as structure optimized proximity scaling (STOPS) (Rusch, Mair & Hornik, 2023,<doi:10.1007/s11222-022-10197-w>). The package contains various functions, wrappers, methods and classes for fitting, plotting and displaying different MDS models in a STOPS framework like Torgerson (classical) scaling, scaling by majorizing a complex function (SMACOF), Sammon mapping, elastic scaling, symmetric SMACOF, spherical SMACOF, s-stress, r-stress, power MDS, power elastic scaling, power Sammon mapping, power stress MDS (POST-MDS), approximate power stress, Box-Cox MDS, local MDS and Isomap. All of these models can also be fit individually with given hyperparameters or by optimizing over hyperparameters based on fit only (i.e., no structure considerations). The package further contains functions for optimization, specifically the adaptive Luus-Jaakola algorithm and a wrapper for Bayesian optimization with treed Gaussian process with jumps to linear models, and functions for various c-structuredness indices.

Depends R (>= 3.5.0), smacof, rgl

Imports cordillera, MASS, pso, scatterplot3d, acepack, minerva, energy, DiceOptim, DiceKriging, tgp, pomp, vegan, scagnostics, clue, cmaes, dfoptim, nloptr

Enhances stats

Suggests sp, R.rsp

License GPL-2 | GPL-3

LazyData true

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

VignetteBuilder R.rsp

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

Repository CRAN

Date/Publication 2023-01-20 10:10:02 UTC

R topics documented:

`apStressMin` *Approximate Power Stress SMACOF*

Description

Minimize approximate power stress by minimization-majorization.

Usage

```
apStressMin(
  delta,
  tau = 1,
  ups = 1,
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  ndim = 2,
  eps = 1e-06,
  itmax = 1000,
  verbose = FALSE
)
```

Arguments

`delta`	dist object or a symmetric, numeric data.frame or matrix of distances
`tau`	the power of the transformation of the proximities; defaults to 1
`ups`	the power of the transformation for weightmat; defaults to 1
`weightmat`	a square, symmetric matrix of finite weights (same dimensions as delta)
`init`	starting configuration
`ndim`	dimension of the configuration; defaults to 2
`eps`	numeric accuracy of the iteration
`itmax`	maximum number of iterations
`verbose`	should iteration output be printed; if TRUE then yes

Value

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

- delta: Observed dissimilarities, not normalized
- obsdiss: Observed transformed dissimilarities
- dhats: Observed transformed dissimilarities, normalized
- confdist: Configuration dissimilarities, NOT normalized
- 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 MDS model
- niter: Number of iterations
- nobj: Number of objects
- type: Type of MDS model
- weightmat: weighting matrix
- pars: hyperparameter vector theta

and some additional components

- stress.m: default stress for the COPS and STOP defaults to the explicitly normalized stress on the normalized, transformed dissimilarities. The square of stress-1 in stress.
- deltaorig: observed, untransformed dissimilarities
- tau: tau parameter
- ups: upsilon parameter

Author(s)

Thomas Rusch

See Also

smacofSym

Examples

```
dis<-smacof::kinshipdelta
res<-apStressMin(as.matrix(dis),tau=2,ups=0.7)
res
summary(res)
plot(res)
```

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)

`bcStressMin` *An MDS version for minimizing BoxCox Stress (Chen & Buja 2013)*

Description

An MDS version for minimizing BoxCox Stress (Chen & Buja 2013)

Usage

```
bcStressMin(
  delta,
  init = NULL,
  verbose = 0,
  ndim = 2,
  mu = 1,
  lambda = 1,
  rho = 0,
  itmax = 2000,
  addD0 = 1e-04
)
```

Arguments

`delta`	dissimilarity or distance matrix
`init`	initial configuration. If NULL a classical scaling solution is used.
`verbose`	prints progress if > 3.
`ndim`	the dimension of the configuration
`mu`	mu parameter. Should be 0 or larger for everything working ok. If mu<0 it works but the 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.
`itmax`	number of optimizing iterations, defaults to 2000.
`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.

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 dissimilarities, not normalized
- obsdiss: Observed transformed dissimilarities, not normalized
- confdist: Configuration dissimilarities, NOT normalized
- conf: Matrix of fitted configuration, NOT normalized
- stress: Default stress (stress 1; sqrt of explicitly normalized stress)
- ndim: Number of dimensions
- model: Name of MDS model
- niter: Number of iterations
- nobj: Number of objects
- pars: hyperparameter vector theta

and some additional components

- stress.m: default stress is the explicitly normalized stress on the normalized, transformed dissimilarities
- deltaorig: observed, untransformed 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<-bcStressMin(as.matrix(dis),mu=2,lambda=1.5,rho=0)
res
summary(res)
plot(res)
```

cmds *normalization function Classical Scaling*

Description

normalization function 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 = TRUE, ...)
```

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.
...	additional parameters passed to cmdscale. See cmdscale

Details

overloads base::cmdscale and adds class attributes for which there are methods. The functionality is duplicated in the cops package.

Value

An object of class 'cmdscaleE' and inheriting from cmdscale. This function just adds an extra slot to the list with the call, adds column labels to the $points.

coef.stops *S3 coef method for stops objects*

Description

S3 coef method for stops objects

Usage

```
## S3 method for class 'stops'
coef(object, ...)
```

Arguments

object	object of class stops
...	addditional arguments

Value

a vector of hyperparmeters theta

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 to be adjusted, a numeric matrix
verbose	should adjustment be output; default to FALSE
eps	numerical accuracy
itmax	maximum number of iterations

Value

A list of configuration matrices. The 'ref.conf' is the reference configuration, the 'other.conf' is the Procrustes adjusted configuration and the 'comparison.conf' is the one that was adjusted.

c_association *c-association calculates the c-association based on the maximal information coefficient We define c-association as the aggregated association between any two columns in confs*

Description

c-association calculates the c-association based on the maximal information coefficient We define c-association as the aggregated association between any two columns in confs

Usage

```
c_association(
  confs,
  aggr = max,
  alpha = 0.6,
  C = 15,
  var.thr = 1e-05,
  zeta = NULL
)
```

Arguments

confs	a numeric matrix or data frame
aggr	the aggregation function for configurations of more than two dimensions. Defaults to max.
alpha	an optional number of cells allowed in the X-by-Y search-grid. Default value is 0.6
C	an optional number determining the starting point of the X-by-Y search-grid. When trying to partition the x-axis into X columns, the algorithm will start with at most C X clumps. Default value is 15.
var.thr	minimum value allowed for the variance of the input variables, since mine can not be computed in case of variance close to 0. Default value is 1e-5.
zeta	integer in [0,1] (?). If NULL (default) it is set to 1-MIC. It can be set to zero for noiseless functions, but the default choice is the most appropriate parametrization for general cases (as stated in Reshef et al). It provides robustness.

Value

a numeric value; association (aggregated maximal information coefficient MIC, see mine)

Examples

```
x<-seq(-3,3,length.out=200)
y<-sqrt(3^2-x^2)
z<- sin(y-x)
```

```
confs<-cbind(x,y,z)
c_association(confs)
```

`c_clumpiness` *c-clumpiness*

Description

Measures the c-clumpiness structure

Usage

```
c_clumpiness(conf, aggr = max)
```

Arguments

`conf`	A numeric matrix.
`aggr`	the aggregation function for configurations of more than two dimensions. Defaults to max.

Value

a numeric value; clumpiness (see `scagnostics`)

Examples

```
delts<-smacof::kinshipdelta
conf<-smacof::smacofSym(delts)$conf
plot(conf,pch=19,asp=1)
c_clumpiness(conf)
```

`c_clusteredness` *c-clusteredness calculates c-clusteredness as the OPTICS cordillera. The higher the more clustered.*

Description

c-clusteredness calculates c-clusteredness as the OPTICS cordillera. The higher the more clustered.

Usage

```
c_clusteredness(
  confs,
  minpts = 2,
  q = 2,
  epsilon = 2 * max(dist(confs)),
  distmeth = "euclidean",
  dmax = NULL,
  digits = 10,
  scale = 0,
  ...
)
```

Arguments

confs	a numeric matrix or a dist object
minpts	The minimum number of points that must make up a cluster in OPTICS (corresponds to k in the paper). It is passed to optics where it is called minPts. Defaults to 2.
q	The norm used for the Cordillera. Defaults to 2.
epsilon	The epsilon parameter for OPTICS (called epsilon_max in the paper). Defaults to 2 times the maximum distance between any two points.
distmeth	The distance to be computed if X is not a symmetric matrix or a dist object (otherwise ignored). Defaults to Euclidean distance.
dmax	The winsorization value for the highest allowed reachability. If used for comparisons between different configurations this should be supplied. If no value is supplied, it is NULL (default); then dmax is taken from the data as the either epsilon or the largest reachability, whatever is smaller.
digits	The precision to round the raw Cordillera and the norm factor. Defaults to 10.
scale	Should X be scaled if it is an asymmetric matrix or data frame? Can take values TRUE or FALSE or a numeric value. If TRUE or 1, standardisation is to mean=0 and sd=1. If 2, no centering is applied and scaling of each column is done with the root mean square of each column. If 3, no centering is applied and scaling of all columns is done as X/max(standard deviation(allcolumns)). If 4, no centering is applied and scaling of all columns is done as X/max(rmsq(allcolumns)). If FALSE, 0 or any other numeric value, no standardisation is applied. Defaults to 0.
...	Additional arguments to be passed to cordillera::cordillera

Value

a numeric value; clusteredness (see cordillera)

Examples

```
delts<-smacof::kinshipdelta
```

```
dis<-smacofSym(delts)$confdist
c_clusteredness(dis,minpts=3)
```

`c_complexity` *c-complexity Calculates the c-complexity based on the minimum cell number We define c-complexity as the aggregated minimum cell number between any two columns in confs This is one of few c-structuredness indices not between 0 and 1, but can be between 0 and (theoretically) infinity*

Description

c-complexity Calculates the c-complexity based on the minimum cell number We define c-complexity as the aggregated minimum cell number between any two columns in confs This is one of few c-structuredness indices not between 0 and 1, but can be between 0 and (theoretically) infinity

Usage

```
c_complexity(
  confs,
  aggr = min,
  alpha = 1,
  C = 15,
  var.thr = 1e-05,
  zeta = NULL
)
```

Arguments

`confs`	a numeric matrix or data frame
`aggr`	the aggregation function for configurations of more than two dimensions. Defaults to min.
`alpha`	an optional number of cells allowed in the X-by-Y search-grid. Default value is 1
`C`	an optional number determining the starting point of the X-by-Y search-grid. When trying to partition the x-axis into X columns, the algorithm will start with at most C X clumps. Default value is 15.
`var.thr`	minimum value allowed for the variance of the input variables, since mine can not be computed in case of variance close to 0. Default value is 1e-5.
`zeta`	integer in [0,1] (?). If NULL (default) it is set to 1-MIC. It can be set to zero for noiseless functions, but the default choice is the most appropriate parametrization for general cases (as stated in Reshef et al.). It provides robustness.

Value

a numeric value; complexity (aggregated minimum cell number MCN, see `mine`)

Examples

```
x<-seq(-3,3,length.out=200)
y<-sqrt(3^2-x^2)
z<- sin(y-x)
confs<-cbind(x,y,z)
c_complexity(confs)
```

`c_convexity` *c-convexity*

Description

Measures the c-convexity structure

Usage

```
c_convexity(conf, aggr = max)
```

Arguments

`conf`	A numeric matrix.
`aggr`	the aggregation function for configurations of more than two dimensions. Defaults to max.

Value

a numeric value; convexity (see `scagnostics`)

Examples

```
delts<-smacof::kinshipdelta
conf<-smacof::smacofSym(delts)$conf
plot(conf,pch=19,asp=1)
c_convexity(conf)
```

`c_dependence` *c-dependence calculates c-dependence as the aggregated distance correlation of each pair if nonidentical columns*

Description

c-dependence calculates c-dependence as the aggregated distance correlation of each pair if nonidentical columns

Usage

```
c_dependence(confs, aggr = max, index = 1)
```

Arguments

confs	a numeric matrix or data frame
aggr	the aggregation function for configurations of more than two dimensions. Defaults to max.
index	exponent on Euclidean distance, in (0,2]

Value

a numeric value; dependence (aggregated distance correlation)

Examples

```
x<-1:10
y<-2+3*x+rnorm(10)
confs<-cbind(x,y)
c_dependence(confs,1.5)
```

---

c_faithfulness *c-faithfulness calculates the c-faithfulness based on the index by Chen and Buja 2013 (M_adj) with equal input neigbourhoods*

Description

c-faithfulness calculates the c-faithfulness based on the index by Chen and Buja 2013 (M_adj) with equal input neigbourhoods

Usage

```
c_faithfulness(confs, obsdiss, k = 3, ...)
```

Arguments

confs	a numeric matrix or a dist object
obsdiss	a symmetric numeric matrix or a dist object
k	the number of nearest neighbours to be looked at
...	additional arguments passed to dist()

Value

a numeric value; faithfulness

Examples

```
delts<-smacof::kinshipdelta
dis<-smacofSym(delts)$confdist
c_faithfulness(dis,delts,k=3)
```

c_functionality *c-functionality calculates the c-functionality based on the maximum edge value We define c-functionality as the aggregated functionality between any two columns of confs*

Description

c-functionality calculates the c-functionality based on the maximum edge value We define c-functionality as the aggregated functionality between any two columns of confs

Usage

```
c_functionality(
  confs,
  aggr = max,
  alpha = 1,
  C = 15,
  var.thr = 1e-05,
  zeta = NULL
)
```

Arguments

confs	a numeric matrix or data frame
aggr	the aggregation function for configurations of more than two dimensions. Defaults to mean
alpha	an optional number of cells allowed in the X-by-Y search-grid. Default value is 1
C	an optional number determining the starting point of the X-by-Y search-grid. When trying to partition the x-axis into X columns, the algorithm will start with at most C X clumps. Default value is 15.
var.thr	minimum value allowed for the variance of the input variables, since mine can not be computed in case of variance close to 0. Default value is 1e-5.
zeta	integer in [0,1] (?). If NULL (default) it is set to 1-MIC. It can be set to zero for noiseless functions, but the default choice is the most appropriate parametrization for general cases (as stated in Reshef et al.). It provides robustness.

Value

a numeric value; functionality (aggregated maximaum edge value MEV, see mine)

Examples

```
x<-seq(-3,3,length.out=200)
y<-sqrt(3^2-x^2)
z<- sin(y-x)
```

```
confs<-cbind(x,y,z)
c_functionality(confs)
```

`c_hierarchy` *c-hierarchy captures how well a partition/ultrametric (obtained by hclust) explains the configuration distances. Uses variance explained for euclidean distances and deviance explained for everything else.*

Description

c-hierarchy captures how well a partition/ultrametric (obtained by hclust) explains the configuration distances. Uses variance explained for euclidean distances and deviance explained for everything else.

Usage

```
c_hierarchy(confs, p = 2, agglmethod = "complete")
```

Arguments

`confs`	a numeric matrix
`p`	the parameter of the Minokwski distances (p=2 euclidean and p=1 is manhattan)
`agglmethod`	the method used for creating the clustering, see `hclust`.

Value

a numeric value; hierarchy (see `cl_validity`)

Examples

```
delts<-smacof::kinshipdelta
conf<-smacofSym(delts)$conf
c_hierarchy(conf,p=2,agglmethod="single")
```

`c_inequality` *c-inequality Calculates c-inequality (as in an economic measure of inequality) as Pearsons coefficient of variation of the fitted distance matrix. This can help with avoiding degenerate solutions. This is one of few c-structuredness indices not between 0 and 1, but 0 and infinity.*

Description

c-inequality Calculates c-inequality (as in an economic measure of inequality) as Pearsons coefficient of variation of the fitted distance matrix. This can help with avoiding degenerate solutions. This is one of few c-structuredness indices not between 0 and 1, but 0 and infinity.

Usage

```
c_inequality(confs)
```

Arguments

confs	a numeric matrix or data frame

Value

a numeric value; inequality (Pearsons coefficient of variation of the fitted distance matrix)

Examples

```
x<-1:10
y<-2+3*x+rnorm(10)
z<- sin(y-x)
confs<-cbind(z,y,x)
c_inequality(confs)
```

---

c_linearity *c-linearity calculates c-linearity as the aggregated multiple correlation of all columns of the configuration.*

Description

c-linearity calculates c-linearity as the aggregated multiple correlation of all columns of the configuration.

Usage

```
c_linearity(confs, aggr = max)
```

Arguments

confs	a numeric matrix or data frame
aggr	the aggregation function for configurations of more than two dimensions. Defaults to max.

Value

a numeric value; linearity (aggregated multiple correlation of all columns of the configuration)

Examples

```
x<-1:10
y<-2+3*x+rnorm(10)
z<- sin(y-x)
confs<-cbind(z,y,x)
c_linearity(confs)
```

c_manifoldness	*c-manifoldness calculates c-manifoldness as the aggregated maximal correlation coefficient (i.e., Pearson correlation of the ACE transformed variables) of all pairwise combinations of two different columns in confs. If there is an NA (happens usually when the optimal transformation of any variable is a constant and therefore the covariance is 0 but also one of the sds in the denominator), it gets skipped.*

Description

c-manifoldness calculates c-manifoldness as the aggregated maximal correlation coefficient (i.e., Pearson correlation of the ACE transformed variables) of all pairwise combinations of two different columns in confs. If there is an NA (happens usually when the optimal transformation of any variable is a constant and therefore the covariance is 0 but also one of the sds in the denominator), it gets skipped.

Usage

```
c_manifoldness(confs, aggr = max)
```

Arguments

confs	a numeric matrix or data frame
aggr	the aggregation function for configurations of more than two dimensions. Defaults to max.

Value

a numeric value; manifoldness (aggregated maximal correlation, correlation of ACE tranformed x and y, see ace)

Examples

```
x<--100:100
y<-sqrt(100^2-x^2)
confs<-cbind(x,y)
c_manifoldness(confs)
```

`c_mine` *wrapper for getting the mine coefficients*

Description

wrapper for getting the mine coefficients

Usage

```
c_mine(confs, master = NULL, alpha = 0.6, C = 15, var.thr = 1e-05, zeta = NULL)
```

Arguments

`confs`	a numeric matrix or data frame with two columns
`master`	the master column
`alpha`	an optional number of cells allowed in the X-by-Y search-grid. Default value is 0.6
`C`	an optional number determining the starting point of the X-by-Y search-grid. When trying to partition the x-axis into X columns, the algorithm will start with at most C X clumps. Default value is 15.
`var.thr`	minimum value allowed for the variance of the input variables, since mine can not be computed in case of variance close to 0. Default value is 1e-5.
`zeta`	integer in [0,1] (?). If NULL (default) it is set to 1-MIC. It can be set to zero for noiseless functions, but the default choice is the most appropriate parametrization for general cases (as stated in Reshef et al. SOM; they call it epsilon in the paper). It provides robustness.

`c_nonmonotonicity` *c-nonmonotonicity calculates the c-nonmonotonicity based on the maximum asymmetric score We define c-nonmonotonicity as the aggregated nonmonotonicity between any two columns in confs this is one of few c-structuredness indices not between 0 and 1*

Description

c-nonmonotonicity calculates the c-nonmonotonicity based on the maximum asymmetric score We define c-nonmonotonicity as the aggregated nonmonotonicity between any two columns in confs this is one of few c-structuredness indices not between 0 and 1

Usage

```
c_nonmonotonicity(
  confs,
  aggr = max,
  alpha = 1,
  C = 15,
  var.thr = 1e-05,
  zeta = NULL
)
```

Arguments

confs	a numeric matrix or data frame
aggr	the aggregation function for configurations of more than two dimensions. Defaults to max.
alpha	an optional number of cells allowed in the X-by-Y search-grid. Default value is 1
C	an optional number determining the starting point of the X-by-Y search-grid. When trying to partition the x-axis into X columns, the algorithm will start with at most C X clumps. Default value is 15.
var.thr	minimum value allowed for the variance of the input variables, since mine can not be computed in case of variance close to 0. Default value is 1e-5.
zeta	integer in [0,1] (?). If NULL (default) it is set to 1-MIC. It can be set to zero for noiseless functions, but the default choice is the most appropriate parametrization for general cases (as stated in Reshef et al. SOM). It provides robustness.

Value

a numeric value; nonmonotonicity (aggregated maximal asymmetric score MAS, see mine)

Examples

```
x<-seq(-3,3,length.out=200)
y<-sqrt(3^2-x^2)
z<- sin(y-x)
confs<-cbind(x,y,z)
c_nonmonotonicity(confs)
```

---

c_outlying *c-outlying*

Description

Measures the c-outlying structure

Usage

```
c_outlying(conf, aggr = max)
```

Arguments

conf	A numeric matrix.
aggr	the aggregation function for configurations of more than two dimensions. Defaults to max.

Value

a numeric value; outlying (see scagnostics)

Examples

```
delts<-smacof::kinshipdelta
conf3<-smacof::smacofSym(delts,ndim=3)$conf
c_outlying(conf3)
```

---

c_regularity *c-regularity calculates c-regularity as 1 - OPTICS cordillera for k=2. The higher the more regular.*

Description

c-regularity calculates c-regularity as 1 - OPTICS cordillera for k=2. The higher the more regular.

Usage

```
c_regularity(
  confs,
  q = 1,
  epsilon = 2 * max(dist(confs)),
  distmeth = "euclidean",
  dmax = NULL,
  digits = 10,
  scale = 0,
  ...
)
```

Arguments

confs	a numeric matrix or a dist object
q	The norm used for the Cordillera. Defaults to 1 (and should always be 1 imo).
epsilon	The epsilon parameter for OPTICS (called epsilon_max in the paper). Defaults to 2 times the maximum distance between any two points.

distmeth	The distance to be computed if X is not a symmetric matrix or a dist object (otherwise ignored). Defaults to Euclidean distance.
dmax	The winsorization value for the highest allowed reachability. If used for comparisons this should be supplied. If no value is supplied, it is NULL (default), then dmax is taken from the data as minimum of epsilon or the largest reachability.
digits	The precision to round the raw Cordillera and the norm factor. Defaults to 10.
scale	Should X be scaled if it is an asymmetric matrix or data frame? Can take values TRUE or FALSE or a numeric value. If TRUE or 1, standardisation is to mean=0 and sd=1. If 2, no centering is applied and scaling of each column is done with the root mean square of each column. If 3, no centering is applied and scaling of all columns is done as X/max(standard deviation(allcolumns)). If 4, no centering is applied and scaling of all columns is done as X/max(rmsq(allcolumns)). If FALSE, 0 or any other numeric value, no standardisation is applied. Defaults to 0.
...	Additional arguments to be passed to cordillera

Value

a numeric value; regularity

Examples

```
hpts<-expand.grid(seq(-5,5),seq(-5,5))
c_regularity(hpts)
hpts2<-cbind(jitter(hpts[,1]),jitter(hpts[,2]))
c_regularity(hpts2)
```

---

c_skinniness *c-skinniness*

Description

Measures the c-skinniness structure

Usage

```
c_skinniness(conf, aggr = max)
```

Arguments

conf	A numeric matrix.
aggr	the aggregation function for configurations of more than two dimensions. Defaults to max.

Value

a numeric value; skininess (see scagnostics)

Examples

```
delts<-smacof::kinshipdelta
conf<-smacof::smacofSym(delts)$conf
plot(conf,pch=19,asp=1)
c_skinniness(conf)
```

`c_sparsity` *c-sparsity*

Description

Measures the c-sparsity structure

Usage

```
c_sparsity(conf, aggr = max)
```

Arguments

`conf`	A numeric matrix.
`aggr`	the aggregation function for configurations of more than two dimensions. Defaults to max.

Value

a numeric value; sparsity (see `scagnostics`)

Examples

```
delts<-smacof::kinshipdelta
conf<-smacof::smacofSym(delts)$conf
plot(conf,pch=19,asp=1)
c_sparsity(conf)
```

`c_striatedness` *c-striatedness*

Description

Measures the c-striatedness structure

Usage

```
c_striatedness(conf, aggr = max)
```

Arguments

conf	A numeric matrix.
aggr	the aggregation function for configurations of more than two dimensions. Defaults to max.

Value

a numeric value; striatedness (see scagnostics)

Examples

```
delts<-smacof::kinshipdelta
conf<-smacof::smacofSym(delts)$conf
plot(conf,pch=19,asp=1)
c_striatedness(conf)
```

---

c_stringiness *c-stringiness*

Description

Measures the c-stringiness structure

Usage

```
c_stringiness(conf, aggr = max)
```

Arguments

conf	A numeric matrix.
aggr	the aggregation function for configurations of more than two dimensions. Defaults to max.

Value

a numeric value; stringiness (see scagnostics)

Examples

```
delts<-smacof::kinshipdelta
conf<-smacof::smacofSym(delts)$conf
plot(conf,pch=19,asp=1)
c_stringiness(conf)
```

doubleCenter *double centering*

Description

double centering

Usage

```
doubleCenter(x)
```

Arguments

x	numeric matrix

enorm *Explicit Norm*

Description

Explicit Norm

Usage

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

Arguments

x	numeric matrix
w	weight

Value

a numeric scalar; the sum(w*x^2)

knn_dist *calculate k nearest neighbours from a distance matrix*

Description

calculate k nearest neighbours from a distance matrix

Usage

```
knn_dist(dis, k)
```

Arguments

dis	distance matrix
k	number of nearest neighbours (Note that with a tie, the function returns the alphanumerically first one!)

ljoptim *(Adaptive) Version of Luus-Jaakola Optimization*

Description

Adaptive means that the search space reduction factors in the number of iterations; makes convergence faster at about 100 iterations

Usage

```
ljoptim(
  x,
  fun,
  ...,
  red = ifelse(adaptive, 0.99, 0.95),
  lower,
  upper,
  acc = 1e-06,
  accd = 1e-04,
  itmax = 1000,
  verbose = 0,
  adaptive = TRUE
)
```

Arguments

x	optional starting values
fun	function to minimize
...	additional arguments to be passed to the function to be optimized
red	value of the reduction of the search region
lower	The lower contraints of the search region
upper	The upper contraints of the search region
acc	if the numerical accuracy of two successive target function values is below this, stop the optimization; defaults to 1e-6
accd	if the width of the search space is below this, stop the optimization; defaults to 1e-4
itmax	maximum number of iterations
verbose	numeric value hat prints information on the fitting process; >2 is extremely verbose
adaptive	should the adaptive version be used? defaults to TRUE.

Value

A list with the components (optim)

- par The position of the optimimum in the search space (parameters that minimize the function; argmin fun)
- value The value of the objective function at the optimum (min fun)
- counts The number of iterations performed at convergence with entries fnction for the number of iterations and gradient which is always NA at the moment
- convergence 0 successful completion by the accd or acc criterion, 1 indicate iteration limit was reached, 99 is a problem
- message is NULL (only for compatibility or future use)

Examples

```
fbana <- function(x) {
x1 <- x[1]
x2 <- x[2]
100 * (x2 - x1 * x1)^2 + (1 - x1)^2
}
res1<-ljoptim(c(-1.2,1),fbana,lower=-5,upper=5,accd=1e-16,acc=1e-16)
res1

set.seed(210485)
fwild <- function (x) 10*sin(0.3*x)*sin(1.3*x^2) + 0.00001*x^4 + 0.2*x+80
plot(fwild, -50, 50, n = 1000, main = "ljoptim() minimising 'wild function'")
res2<-ljoptim(50, fwild,lower=-50,upper=50,adaptive=FALSE,accd=1e-16,acc=1e-16)
points(res2$par,res2$value,col="red",pch=19)
res2
```

`lmds` *An function for local MDS (Chen & Buja 2006)*

Description

An function for local MDS (Chen & Buja 2006)

Usage

```
lmds(delta, init = NULL, ndim = 3, k = 2, tau = 1, itmax = 5000, verbose = 0)
```

Arguments

`delta`	dissimilarity or distance matrix
`init`	initial configuration. If NULL a classical scaling solution is used.
`ndim`	the dimension of the configuration
`k`	the k neighbourhood parameter
`tau`	the penalty parameter (suggested to be in [0,1])
`itmax`	number of optimizing iterations, defaults to 5000.
`verbose`	prints progress if > 4.

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 dissimilarities, not normalized
- obsdiss: Observed transformed dissimilarities, not normalized
- confdist: Configuration dissimilarities, NOT normalized
- conf: Matrix of fitted configuration, NOT normalized
- stress: Default stress (stress 1; sqrt of explicitly normalized stress)
- ndim: Number of dimensions
- model: Name of MDS model
- niter: Number of iterations
- nobj: Number of objects

- pars: hyperparameter vector theta

and some additional components

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

Author(s)

Lisha Chen & Thomas Rusch

Examples

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

`mkBmat` *MkBmat function (internal)*

Description

MkBmat function (internal)

Usage

```
mkBmat(x)
```

Arguments

`x`	matrix

mkPower *MakePower Old*

Description

MakePower Old

Usage

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

Arguments

x	matrix
r	numeric (power)

Value

the matrix to a power

mkPower2 *MakePower*

Description

MakePower

Usage

```
mkPower2(x, theta)
```

Arguments

x	matrix
theta	numeric (power)

Pendigits500 *Pen digits*

Description

These data are a random sample of 500 of the 10992 pendigits data from Alimoglu (1996). The original data were from 44 writers who handwrote 250 times the digits 0,...,9. The digits were written inside a rectangular box with a resolution of 500 x 500 pixels and the first 10 per writer were ignored for further analysis. This led to 10992 digits. They were recorded in small time intervals by following the trajectory of the pen on the 500 x 500 grid and then normalized. From the normalized trajectory 8 points (x and y axis position) were randomly selected for each handwritten digit, leading to 16 predictors variables. We extarcted a random sample of 500 of them.

Usage

```
data(Pendigits500)
```

Format

A data frame with 500 rows and 17 variables

Details

The variables are

- The rownames of Pendigits500 refer to the data point of the 10992 original data
- V1-V16: trajectory points (x, y coordinate) of the grid
- digits: The digit actually written (the label)

Source

From A. Izenman (2010) Modern multivariate statistical techniques. Springer.

plot.cmdscaleE *S3 plot method for cmdscaleE*

Description

S3 plot method for cmdscaleE

Usage

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

Arguments

x	cmdscaleE object
plot.type	type of plot
plot.dim	dimensions used for plotting
col	color
label.conf	list of label options
identify	boolean flag for interactively identify points
type	type of plot
pch	plotting character
asp	aspect ratio (defaults to 1)
main	main title
xlab	label of x axis
ylab	label of y axis
xlim	limits of x axis
ylim	limits of y axis
legpos	position of legend
...	additional arguments passed to plot

Details

This function duplicates the plot method for smacof so it can be used with cmdscaleE objects. See plot.smacof for the arguments.

Value

No return value, just plots a 'cmdscaleE' object.

`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 = 5,
  col,
  label.conf = list(label = TRUE, pos = 3, col = 1, cex = 0.8),
  identify = FALSE,
  type = "p",
  pch = 20,
  asp = 1,
  main,
  xlab,
  ylab,
  xlim,
  ylim,
  legend = TRUE,
  legpos,
  loess = 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)

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')
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
legend	Flag whether legends should be drawn for plots that have legends
legpos	Position of legend in plots with legends
loess	should loess fit be added to Shepard plot
...	Further plot arguments passed: see 'plot.smacof' and 'plot' for detailed information.

Details

- Configuration plot (plot.type = "confplot"): Plots the MDS configurations.
- Residual plot (plot.type = "resplot"): Plots the dissimilarities against the fitted distances with a linear regression line (without an intercept as in ratio MDS).
- Linearized Shepard diagram (plot.type = "Shepard"): Diagram with the transformed observed dissimilarities against the transformed fitted distance as well as loess curve and a least squares line. The fitted lines do not have an intercept.
- Transformation Plot (plot.type = "transplot"): Diagram with the observed dissimilarities (lighter) and the transformed observed dissimilarities (darker) against the fitted distances together with the nonlinear regression curve (no intercept). Works for lmds or bcStress models too, but is somewhat nonsensical due to them being energy models.
- Stress decomposition plot (plot.type = "stressplot"): Plots the stress contribution in of each observation. Note that it rescales the stress-per-point (SPP) from the corresponding smacof function to percentages (sum is 100). The higher the contribution, the worse the fit. Only implemented for models from the classical stress world, not for bcmds or lmds (throws an error).
- Bubble plot (plot.type = "bubbleplot"): Combines the configuration plot with the point stress contribution. The larger the bubbles, the better the fit.Only implemented for models from the classical stress world, bcmds or lmds (throws an error).

Value

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

`plot.stops` *S3 plot method for stops objects*

Description

S3 plot method for stops objects

Usage

```
## S3 method for class 'stops'
plot(x, plot.type = c("confplot"), main, asp = NA, ...)
```

Arguments

`x`	an object of class stops
`plot.type`	String indicating which type of plot to be produced: "confplot", "resplot", "Shepard", "stressplot", "bubbleplot" (see details)
`main`	the main title of the plot
`asp`	aspect ratio of x/y axis; defaults to NA; setting to 1 will lead to an accurate represenation of the fitted distances.
`...`	Further plot arguments passed: see 'plot.smacof' and 'plot' for detailed information.

Details:

- Configuration plot (plot.type = "confplot"): Plots the MDS configurations.
- Residual plot (plot.type = "resplot"): Plots the dissimilarities against the fitted distances.
- Linearized Shepard diagram (plot.type = "Shepard"): Diagram with the transformed observed dissimilarities against the transformed fitted distance as well as loess smooth and a least squares line.
- Stress decomposition plot (plot.type = "stressplot", only for SMACOF objects in $fit): Plots the stress contribution in of each observation. Note that it rescales the stress-per-point (SPP) from the corresponding smacof function to percentages (sum is 100). The higher the contribution, the worse the fit.
- Bubble plot (plot.type = "bubbleplot", only available for SMACOF objects $fit): Combines the configuration plot with the point stress contribution. The larger the bubbles, the better the fit.

Value

no return value, just plots

`plot3d.cmdscaleE` *S3 plot3d method for class cmdscaleE*

Description

This methods produces a dynamic 3D configuration plot.

Usage

```
## S3 method for class 'cmdscaleE'
plot3d(
  x,
  plot.dim = c(1, 2, 3),
  xlab,
  ylab,
  zlab,
  col,
  main,
  bgpng = NULL,
  ax.grid = TRUE,
  sphere.rgl = FALSE,
  ...
)
```

Arguments

`x`	object of class cmdscaleE
`plot.dim`	vector of length 3 with dimensions to be plotted
`xlab`	label of x axis
`ylab`	label of y axis
`zlab`	label of z axis
`col`	color of the text labels
`main`	plot title
`bgpng`	Background image from rgl library; 'NULL' for white background
`ax.grid`	If 'TRUE', axes grid is plotted.
`sphere.rgl`	If 'TRUE', rgl sphere (background) is plotted.
`...`	Further plot arguments passed: see 'plot3d' in package 'rgl' for detailed information.

Value

No return value, just plots a 'cmdscale' object.

plot3d.stops *S3 plot3d method for class stops*

Description

This methods produces a dynamic 3D configuration plot.

Usage

```
## S3 method for class 'stops'
plot3d(x, ...)
```

Arguments

x	object of class stops
...	Further plot arguments to the method of the class of slot $fit, see plot.smacof or plot3d.cmdscaleE . Also see 'rgl' in package 'rgl'

Value

no return value, just plots

plot3dstatic *plot3dstatic: static 3D plots*

Description

A static 3d plot S3 generic

Usage

```
plot3dstatic(x, plot.dim = c(1, 2, 3), main, xlab, ylab, zlab, col, ...)
```

Arguments

x	object
plot.dim	dimensions to plot
main	main title
xlab	label for x axis
ylab	label for y axis
zlab	label for z axis
col	color
...	other arguments

Details

A static 3d plot

Value

No return value, just plots.

`plot3dstatic.cmdscaleE` *3D plots: plot3dstatic method for class cmdscaleE*

Description

This methods produces a static 3D configuration plot.

Usage

```
## S3 method for class 'cmdscaleE'
plot3dstatic(x, plot.dim = c(1, 2, 3), main, xlab, ylab, zlab, col, ...)
```

Arguments

`x`	object of class cmdscaleE
`plot.dim`	vector of length 3 with dimensions to be plotted
`main`	plot title
`xlab`	label of x axis
`ylab`	label of y axis
`zlab`	label of z axis
`col`	color of the text labels
`...`	Further plot arguments passed: see 'scatterplot3d' in package 'scatterplot3d' for detailed information.

Value

No return value, just plots a 'cmdscaleE' object.

plot3dstatic.stops *S3 plot3dstatic method for class stops*

Description

This methods produces a static 3D configuration plot.

Usage

```
## S3 method for class 'stops'
plot3dstatic(x, ...)
```

Arguments

x	object of class stops
...	Further plot arguments to the method of the class of slot fit, see plot3dstatic or plot3dstatic.cmdscaleE . Also see 'scatterplot3d' in package 'scatterplot3d'.

Value

no return value, just plots

powerStressMin *Power Stress SMACOF*

Description

An implementation to minimize power stress by minimization-majorization. Usually more accurate but slower than powerStressFast.

Usage

```
powerStressMin(
  delta,
  kappa = 1,
  lambda = 1,
  nu = 1,
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  ndim = 2,
  acc = 1e-10,
  itmax = 50000,
  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	numeric accuracy of the iteration
itmax	maximum number of iterations. Defaults to 50000.
verbose	should iteration output be printed; if > 1 then yes

Value

an object of class 'smacofP' (inheriting form 'smacofB', see smacofSym). It is a list with the components

- delta: Observed dissimilarities, not normalized
- obsdiss: Observed transformed dissimilarities, not normalized
- confdist: Configuration dissimilarities, NOT normalized
- 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
- pars: hyperparameter vector theta

and some additional components

- stress.m: default stress is the explicitly normalized stress on the normalized, transformed dissimilarities
- deltaorig: observed, untransformed dissimilarities
- kappa: kappa parameter
- lambda: lambda parameter
- nu: nu parameter (aka rho)

Note

The functionality related to power stress and the 'smacofP' class is also available in the 'cops' package. Expect masking when both are loaded.

Author(s)

Jan de Leeuw & Thomas Rusch

See Also

smacofSym

Examples

```
dis<-smacof::kinshipdelta
res<-powerStressMin(as.matrix(dis),kappa=2,lambda=1.5,nu=2,
                    weightmat=as.matrix(dis/2),itmax=1000)
res
summary(res)
plot(res)
```

---

print.cmdscale *S3 print method for cmdscale*

Description

S3 print method for cmdscale

Usage

```
## S3 method for class 'cmdscale'
print(x, ...)
```

Arguments

x	cmdscale object
...	additional arguments

Value

No return value, just prints.

print.sammon *S3 print method for sammon objects*

Description

S3 print method for sammon objects

Usage

```
## S3 method for class 'sammon'
print(x, ...)
```

Arguments

x	cmdscale object
...	additional arguments

Value

No return value, just prints.

print.stops *S3 print method for stops objects*

Description

S3 print method for stops objects

Usage

```
## S3 method for class 'stops'
print(x, ...)
```

Arguments

x	stops object
...	additional arguments

Value

no return value, just prints

print.summary.smacofP *S3 print method for summary.smacofP*

Description

S3 print method for summary.smacofP

Usage

```
## S3 method for class 'summary.smacofP'
print(x, ...)
```

Arguments

x	object of class summary.smacofP
...	additional arguments

Value

No return value, just prints a 'summary.smacofP'

print.summary.stops *S3 print method for summary.stops*

Description

S3 print method for summary.stops

Usage

```
## S3 method for class 'summary.stops'
print(x, ...)
```

Arguments

x	object of class summary.stops
...	additional arguments

Value

no return value, just prints

`procruster` *procruster: a procrustes function*

Description

procruster: a procrustes function

Usage

```
procruster(x)
```

Arguments

`x`	mumeric matrix

Value

A double or complex matrix.

`residuals.stops` *S3 residuals method for stops*

Description

S3 residuals method for stops

Usage

```
## S3 method for class 'stops'
residuals(object, ...)
```

Arguments

`object`	object of class stops
`...`	addditional arguments

Value

a vector of residuals (observed minus fitted distances)

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 class attributes for which there are methods. The functionality is duplicated in the cops package.

Value

An object of class 'sammonE' that inherits from sammon. This function only adds an extra slot to the list with the call, adds column labels to the $points and assigns S3 classes 'sammonE', 'cmdscale'. It also adds a slot obsdiss with normalized dissimilarities.

secularEq *Secular Equation*

Description

Secular Equation

Usage

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

Arguments

a	matrix
b	matrix

---

sqdist *Squared distances*

Description

Squared distances

Usage

```
sqdist(x)
```

Arguments

x	numeric matrix

Value

a matrix of squared distances

---

stoploss *Calculate the weighted multiobjective loss function used in STOPS*

Description

Calculate the weighted multiobjective loss function used in STOPS

Usage

```
stoploss(
  obj,
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
   "cassociation", "cnonmonotonicity", "cfunctionality", "ccomplexity", "cfaithfulness",
    "cregularity", "chierarchy", "cconvexity", "cstriatedness", "coutlying",
    "cskinniness", "csparsity", "cstringiness", "cclumpiness", "cinequality"),
  strucweight = rep(-1/length(structures), length(structures)),
  strucpars,
  type = c("additive", "multiplicative"),
  verbose = 0
)
```

Arguments

obj	object returned inside a stop_* function. Uses the stress.m slot for getting the stress.
stressweight	weight to be used for the fit measure; defaults to 1
structures	which c-structuredness indices to be included in the loss
strucweight	the weights of the structuredness indices; defaults to -1/#number of structures
strucpars	a list of parameters to be passed to the c-structuredness indices in the same order as the values in structures. If the index has no parameters or you want to use the defaults, supply NULL. (alternatively a named list that has the structure name as the element name).
type	what type of weighted combination should be used? Can be 'additive' or 'multiplicative'.
verbose	verbose output

Value

a list with calculated stoploss ($stoploss), structuredness indices ($strucinidices) and hyperparameters ($parameters and $theta)

---

stops *stops: structure optimized proximity scaling*

Description

A package for "structure optimized proximity scaling" (STOPS), a collection of methods that fit nonlinear distance transformations in multidimensional scaling (MDS) and trade-off the fit with structure considerations to find optimal parameters or optimal configurations. The package contains various functions, wrappers, methods and classes for fitting, plotting and displaying different MDS models in a STOPS framework like Torgerson scaling, SMACOF, Sammon mapping, elastic scaling, symmetric SMACOF, spherical SMACOF, sstress, rstress, powermds, power elastic scaling, power sammon mapping, power stress, Isomap, approximate power stress, restricted power stress. All of these models can also be fit as MDS variants (i.e., no structuredness). The package further contains functions for optimization (Adaptive Luus-Jaakola and for Bayesian optimization with treed Gaussian process with jump to linear models) and functions for various structuredness indices

This allows to fit STOPS models as described in Rusch, Mair, Hornik (2023).

Usage

```
stops(
  dis,
 loss = c("strain", "stress", "smacofSym", "powerstress", "powermds", "powerelastic",
   "powerstrain", "elastic", "sammon", "sammon2", "smacofSphere", "powersammon",
   "rstress", "sstress", "isomap", "isomapeps", "bcstress", "lmds", "apstress",
   "rpowerstress"),
```

```
  theta = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
   "cassociation", "cnonmonotonicity", "cfunctionality", "ccomplexity", "cfaithfulness",
    "cregularity", "chierarchy", "cconvexity", "cstriatedness", "coutlying",
    "cskinniness", "csparsity", "cstringiness", "cclumpiness", "cinequality"),
  ndim = 2,
  weightmat = NULL,
  init = NULL,
  stressweight = 1,
  strucweight,
  strucpars,
  optimmethod = c("SANN", "ALJ", "pso", "Kriging", "tgp", "DIRECT", "stogo", "cobyla",
    "crs2lm", "isres", "mlsl", "neldermead", "sbplx", "hjk", "cmaes"),
  lower,
  upper,
  verbose = 0,
  type = c("additive", "multiplicative"),
  initpoints = 10,
  itmax = 50,
  itmaxps = 10000,
  model,
  control,
  ...
)
```

Arguments

dis	numeric matrix or dist object of a matrix of proximities
loss	which loss function to be used for fitting, defaults to stress.
theta	hyperparameter vector starting values for the transformation functions. If the length is smaller than the number of hyperparameters for the MDS version the vector gets recycled (see the corresponding stop_XXX function or the vignette for how theta must look like exactly for each loss). If larger than the number of hyperparameters for the MDS method, an error is thrown. If completely missing theta is set to 1 and recycled.
structures	character vector of which c-structuredness indices should be considered; if missing no structure is considered.
ndim	number of dimensions of the target space
weightmat	(optional) a matrix of nonnegative weights; defaults to 1 for all off diagonals
init	(optional) initial configuration
stressweight	weight to be used for the fit measure; defaults to 1
strucweight	vector of weights to be used for the c-structuredness indices (in the same order as in structures); defaults to -1/length(structures) for each index
strucpars	(possibly named with the structure). Metaparameters for the structuredness indices (gamma in the article). It's safest for it be a list of lists with the named arguments for the structuredness indices and the order of the lists must be like the

order of structures. So something like this `list(list(par1Struc1=par1Struc1,par2Struc1=par2Stru` where parYStrucX are the named arguments for the metaparameter Y of the structure X the list elements corresponds to. For a structure without parameters, set NULL. Parameters in different list elements parYStrucX can have the same name. For example, say we want to use cclusteredness with metaparameters epsilon=10 and k=4 (and the default for the other parameters), cdependence with no metaparameters and cfaithfulness with metaparameter k=7 one would `list(list(epsilon=10,k=4),list(NULL),list(dis=obdiss,k=6))` for structures vector ("cclusteredness","cdependence","cfaithfulness"). The parameter lists must be in the same ordering as the indices in structures. If missing it is set to NULL and defaults are used. It is also possible to supply a structure's metaparameters as a list of vectors with named elements if the metaparameters are scalars, so like `list(c(par1Struc1=parStruc1,par2Struc1=par1Struc1,...),c(par1Struc2=p` That can have unintended consequences if the metaparameter is a vector or matrix.

`optimmethod` What solver to use. Currently supported are Bayesian optimization with Gaussian Process priors and Kriging ("Kriging"), Bayesian optimization with treed Gaussian processes with jump to linear models ("tgp"), Adaptive LJ Search ("ALJ"), Particle Swarm optimization ("pso"), simulated annealing ("SANN"), "DIRECT", Stochastic Global Optimization ("stogo"), COBYLA ("cobyla"), Controlled Random Search 2 with local mutation ("crs2lm"), Improved Stochastic Ranking Evolution Strategy ("isres"), Multi-Level Single-Linkage ("mlsl"), Nelder-Mead ("neldermead"), Subplex ("sbplx"), Hooke-Jeeves Pattern Search ("hjk"), CMA-ES ("cmaes"). Defaults to "ALJ" version. tgp, ALJ, Kriging and pso usually work well for relatively low values of itmax.

`lower` The lower contraints of the search region. Needs to be a numeric vector of the same length as the parameter vector theta.

`upper` The upper contraints of the search region. Needs to be a numeric vector of the same length as the parameter vector theta.

`verbose` numeric value hat prints information on the fitting process; >2 is very verbose.

`type` which aggregation for the multi objective target function? Either 'additive' (default) or 'multiplicative'

`initpoints` number of initial points to fit the surrogate model for Bayesian optimization; default is 10.

`itmax` maximum number of iterations of the outer optimization (for theta) or number of steps of Bayesian optimization; default is 50. We recommend a higher number for ALJ (around 150). Note that due to the inner workings of some solvers, this may or may not correspond to the actual number of function evaluations performed (or PS models fitted). E.g., with tgp the actual number of function evaluation of the PS method is between itmax and 6*itmax as tgp samples 1-6 candidates from the posterior and uses the best candidate. For pso it is the number of particles s times itmax. For cmaes it is usually a bit higher than itmax. This currently may get overruled by a control argument if it is used (and then set to either ewhat is supplie dby control or to the default of the method).

`itmaxps` maximum number of iterations of the inner optimization (to obtain the PS configuration)

model	a character specifying the surrogate model to use. For Kriging it specifies the covariance kernel for the GP prior; see covTensorProduct-class defaults to "powerexp". For tgp it specifies the non stationary process used see bgp, defaults to "btgpllm"
control	a control argument passed to the outer optimization procedure. Will override any other control arguents passed, especially verbose and itmax. For the efect of control, see the functions pomp::sannbox for SANN and pso::psoptim for pso, cmaes::cma_es for cmaes, dfoptim::hjkb for hjk and the nloptr docs for the algorithms DIRECT, stogo, cobyla, crs2lm, isres, mlsl, neldermead, sbplx.
...	additional arguments passed to the outer optimization procedures (not fully tested).

Details

The stops package provides five categories of important functions:

Models & Algorithms:

- stops() ... which fits STOPS models as described in Rusch et al. (2023). By setting cordweight or strucweight to zero they can also be used to fit metric MDS for many different models, see below.
- powerStressMin()... a workhorse for fitting many stresses, including s-stress, r-stress (De Leeuw, 2014), Sammon mapping with power transformations (powersammon), elastic scaling with power transformation (powerelastic), power stress. They can most conveniently be accessed via the stops functions and setting stressweight=1 and cordweight or strucweight=0 or by the dedicated functions starting with stop_foo where foo is the method and setting stressweight=1 and strucweight=0. It uses the nested majorization algorithm for r-stress of De Leeuw(2014).
- bcStressMin()... a workhorse for fitting Box-Cox stress (Chen & Buja, 2013).
- lmds()... a workhorse for the local MDS of Chen & Buja (2008).

Structuredness Indices: Various c-structuredness as c_foo(), where foo is the name of the structuredness. See Rusch et al. (2023).

Optimization functions:

- ljoptim() ... An (adaptive) version of the Luus-Jakola random search

Wrappers and convenience functions:

- conf_adjust(): procrustes adjustment of configurations
- cmdscale(), sammon(): wrappers that return S3 objects
- stop_smacofSym(), stop_sammon(), stop_cmdscale(), stop_rstress(), stop_powerstress(),stop_smacofSphere(), stop_sammon2(), stop_elastic(), stop_sstress(), stop_powerelastic(), stop_powersammon(), stop_powermds(), stop_isomap(), stop_isomapeps(), stop_bcstress(), stop_lmds(), stop_apstress(),stops_rpowerstress(): stop versions of these MDS models.
- stoploss() ... a function to calculate stoploss (Rusch et al., 2023)

Methods: For most of the objects returned by the high-level functions S3 classes and methods for standard generics were implemented, including print, summary, plot, plot3d, plot3dstatic.

References:

- Rusch, T., Mair, P., & Hornik, K. (2023). Structure-based hyperparameter selection with Bayesian optimization in multidimensional scaling. Statistics & Computing, 33, [28]. https://doi.org/10.1007/s11222-022-10197-w

Authors: Thomas Rusch, Lisha Chen, Jan de Leeuw, Patrick Mair, Kurt Hornik

Maintainer: Thomas Rusch

The combination of c-structurednes indices and stress uses the stress.m values, which are the explictly normalized stresses. Reported however is the stress-1 value which is sqrt(stress.m).

Value

A list with the components

- stoploss: the stoploss value
- optim: the object returned from the optimization procedure
- stressweight: the stressweight
- strucweight: the vector of structure weights
- call: the call
- optimmethod: The solver selected
- losstype: The PS badness-of-fit function
- nobj: the number of objects in the configuration
- type: The type of stoploss scalacrisation (additive or multiplicative)
- fit: The fitted PS object (most importantly fitconf the fitted configuration)

Examples

```
data(kinshipdelta,package="smacof")

strucpars<-list(list(epsilon=10,minpts=2,scale=3),list(NULL))
dissm<-as.matrix(kinshipdelta)

#STOPS with strain
resstrain<-stops(dissm,loss="strain",theta=1,structures=c("cclusteredness","cdependence"),
strucpars=strucpars,optimmethod="ALJ",lower=0,upper=10,itmax=10)
resstrain
summary(resstrain)
plot(resstrain)


#STOPS with stress
strucpars<-list(list(epsilon=10,minpts=2,scale=3),NULL)
resstress<-stops(dissm,loss="stress",
structures=c("cclusteredness","cdependence"),
strucpars=strucpars,optimmethod="ALJ",lower=0,upper=10)
resstress
summary(resstress)
plot(resstress)
plot(resstress,"Shepard")
```

```
#STOPS with powerstress
respstress<-stops(dissm,loss="powerstress",
structures=c("cclusteredness","cdependence"),
strucpars=strucpars,weightmat=dissm,
itmaxps=1000,optimmethod="ALJ",lower=c(0,0,1),upper=c(10,10,10))
respstress
summary(respstress)
plot(respstress)

#STOPS with bcstress
resbcstress<-stops(dissm,loss="bcstress",
structures=c("cclusteredness","cdependence"),
strucpars=strucpars,optimmethod="ALJ",lower=c(0,1,0),upper=c(10,10,10))
resbcstress
summary(resbcstress)
plot(resbcstress)

#STOPS with lmds
reslmds<-stops(dissm,loss="lmds",
structures=c("cclusteredness","clinearity"),
strucpars=strucpars,optimmethod="ALJ",lower=c(2,0),upper=c(10,2))
reslmds
summary(reslmds)
plot(reslmds)

#STOPS with Isomap (the epsilon version)
resiso<-stops(dissm,loss="isomapeps",
structures=c("cclusteredness","clinearity"),
strucpars=strucpars,optimmethod="ALJ",lower=70,upper=120)
resiso
summary(resiso)
plot(resiso)


data(kinshipdelta,package="smacof")
strucpar<-list(NULL,NULL) #parameters for indices
res1<-stops(kinshipdelta,loss="stress",
structures=c("cclumpiness","cassociation"),strucpars=strucpar,
lower=0,upper=10,itmax=10)
res1


data(BankingCrisesDistances)
strucpar<-list(c(epsilon=10,minpts=2),NULL) #parameters for indices
res1<-stops(BankingCrisesDistances[,1:69],loss="stress",verbose=0,
structures=c("cclusteredness","clinearity"),strucpars=strucpar,
lower=0,upper=10)
res1

strucpar<-list(list(alpha=0.6,C=15,var.thr=1e-5,zeta=NULL),
list(alpha=0.6,C=15,var.thr=1e-5,zeta=NULL))
res1<-stops(BankingCrisesDistances[,1:69],loss="stress",verbose=0,
```

```
structures=c("cfunctionality","ccomplexity"),strucpars=strucpar,
lower=0,upper=10)
res1
```

---

stop_apstress *STOPS version of approximated power stress models.*

Description

This uses an approximation to power stress that can make use of smacof as workhorse. Free parameters are tau and upsilon.

Usage

```
stop_apstress(
  dis,
  theta = c(1, 1),
  ndim = 2,
  weightmat = NULL,
  init = NULL,
  itmax = 1000,
  ...,
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
   "cassociation", "cnonmonotonicity", "cfunctionality", "ccomplexity", "cfaithfulness",
    "cregularity", "chierarchy", "cconvexity", "cstriatedness", "coutlying",
    "cskinniness", "csparsity", "cstringiness", "cclumpiness", "cinequality"),
  strucweight = rep(1/length(structures), length(structures)),
  strucpars,
  verbose = 0,
  type = c("additive", "multiplicative")
)
```

Arguments

dis	numeric matrix or dist object of a matrix of proximities
theta	the theta vector of parameters to optimize over. Must be of length two, with the first the tau argument and the second the upsilon argument. It can also be a scalar of the tau and upsilon transformation for the observed proximities and gets recycled for both ups and tau (so they are equal). Defaults to 1 1.
ndim	number of dimensions of the target space
weightmat	(optional) a binary matrix of nonnegative weights
init	(optional) initial configuration
itmax	number of iterations. default is 1000.

...	additional arguments to be passed to the fitting procedure
stressweight	weight to be used for the fit measure; defaults to 1
structures	a character vector listing the structure indices to use. They always are called "cfoo" with foo being the structure.
strucweight	weight to be used for the structures; defaults to 1/number of structures
strucpars	a list of list of parameters for the structuredness indices; each list element corresponds to one index in the order of the appearance in structures vector. See examples.
verbose	numeric value hat prints information on the fitting process; >2 is extremely verbose
type	which weighting to be used in the multi-objective optimization? Either 'additive' (default) or 'multiplicative'.

Value

A list with the components

- stress: the stress 1 (sqrt stress.m)
- stress.m: default normalized stress
- stoploss: the weighted loss value
- struc: the structuredness indices
- parameters: the parameters used for fitting (kappa=1, tau, ups)
- fit: the returned object of the fitting procedure
- stopobj: the stopobj object

---

stop_bcstress *STOPS version of Box Cox Stress*

Description

STOPS version of Box Cox Stress

Usage

```
stop_bcstress(
  dis,
  theta = c(1, 1, 0),
  weightmat = NULL,
  init = NULL,
  ndim = 2,
  itmax = 5000,
  ...,
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
```

```
   "cassociation", "cnonmonotonicity", "cfunctionality", "ccomplexity", "cfaithfulness",
    "cregularity", "chierarchy", "cconvexity", "cstriatedness", "coutlying",
    "cskinniness", "csparsity", "cstringiness", "cclumpiness", "cinequality"),
  strucweight = rep(1/length(structures), length(structures)),
  strucpars,
  verbose = 0,
  type = c("additive", "multiplicative")
)
```

Arguments

dis	numeric matrix or dist object of a matrix of proximities
theta	the theta vector of powers; the first is mu (for the fitted distances), the second lambda (for the proximities), the third nu (for the weights). If a scalar is given it is recycled. Defaults to 1 1 0.
weightmat	(not used)
init	(optional) initial configuration
ndim	number of dimensions of the target space
itmax	number of iterations
...	additional arguments to be passed to the fitting procedure
stressweight	weight to be used for the fit measure; defaults to 1
structures	which structures to look for
strucweight	weight to be used for the structures; defaults to 0.5
strucpars	a list of parameters for the structuredness indices; each list element corresponds to one index in the order of the appeacrance in structures
verbose	numeric value hat prints information on the fitting process; >2 is extremely verbose
type	which weighting to be used in the multi-objective optimization? Either 'additive' (default) or 'multiplicative'.

Value

A list with the components

- stress: the stress
- stress.m: default normalized stress
- stoploss: the weighted loss value
- struc: the structuredness indices
- parameters: the parameters used for fitting (kappa, lambda)
- fit: the returned object of the fitting procedure
- stopobj: the stopobj object

`stop_cmdscale` *STOPS version of strain*

Description

The free parameter is lambda for power transformations of the observed proximities.

Usage

```
stop_cmdscale(
  dis,
  theta = 1,
  weightmat = NULL,
  ndim = 2,
  init = NULL,
  ...,
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
   "cassociation", "cnonmonotonicity", "cfunctionality", "ccomplexity", "cfaithfulness",
    "cregularity", "chierarchy", "cconvexity", "cstriatedness", "coutlying",
    "cskinniness", "csparsity", "cstringiness", "cclumpiness", "cinequality"),
  strucweight = rep(1/length(structures), length(structures)),
  strucpars,
  verbose = 0,
  type = c("additive", "multiplicative"),
  itmax = NULL
)
```

Arguments

`dis`	numeric matrix or dist object of a matrix of proximities
`theta`	the theta vector of powers; this must be a scalar of the lambda transformation for the observed proximities.
`weightmat`	(optional) a matrix of nonnegative weights. Not used.
`ndim`	number of dimensions of the target space
`init`	(optional) initial configuration
`...`	additional arguments to be passed to the fitting procedure
`stressweight`	weight to be used for the fit measure; defaults to 1
`structures`	which structuredness indices to be included in the loss
`strucweight`	weight to be used for the structuredness indices; ; defaults to 1/#number of structures
`strucpars`	the parameters for the structuredness indices
`verbose`	numeric value hat prints information on the fitting process; >2 is extremely verbose

type	How to construct the target function for the multi objective optimization? Either 'additive' (default) or 'multiplicative'
itmax	placeholder for compatibility in stops call; not used

Value

A list with the components

- stress: Sqrt of explicitly normalized stress.
- stress.m: explictly normalized stress
- stoploss: the weighted loss value
- indices: the values of the structuredness indices
- parameters: the parameters used for fitting
- fit: the returned object of the fitting procedure
- stopobj: the stopobj object

---

stop_elastic *STOPS versions of elastic scaling models (via smacofSym)*

Description

The free parameter is lambda for power transformations the observed proximities. The fitted distances power is internally fixed to 1 and the power for the weights=delta is -2. Allows for a weight matrix because of smacof.

Usage

```
stop_elastic(
  dis,
  theta = 1,
  ndim = 2,
  weightmat = NULL,
  init = NULL,
  itmax = 1000,
  ...,
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
   "cassociation", "cnonmonotonicity", "cfunctionality", "ccomplexity", "cfaithfulness",
    "cregularity", "chierarchy", "cconvexity", "cstriatedness", "coutlying",
    "cskinniness", "csparsity", "cstringiness", "cclumpiness", "cinequality"),
  strucweight = rep(1/length(structures), length(structures)),
  strucpars,
  verbose = 0,
  type = c("additive", "multiplicative")
)
```

Arguments

dis	numeric matrix or dist object of a matrix of proximities
theta	the theta vector of powers; this must be a scalar of the lambda transformation for the observed proximities. Defaults to 1.
ndim	number of dimensions of the target space
weightmat	(optional) a matrix of nonnegative weights (NOT the elscal weights)
init	(optional) initial configuration
itmax	number of iterations
...	additional arguments to be passed to the fitting procedure
stressweight	weight to be used for the fit measure; defaults to 1
structures	which structuredness indices to be included in the loss
strucweight	weight to be used for the structuredness indices; ; defaults to 1/#number of structures
strucpars	the parameters for the structuredness indices
verbose	numeric value hat prints information on the fitting process; >2 is extremely verbose
type	How to construct the target function for the multi objective optimization? Either 'additive' (default) or 'multiplicative'

Value

A list with the components

- stress: the stress-1 (sqrt(stress.m))
- stress.m: default normalized stress (used for STOPS)
- stoploss: the weighted loss value
- indices: the values of the structuredness indices
- parameters: the parameters used for fitting
- fit: the returned object of the fitting procedure
- stopobj: the stopobj objects

---

stop_isomap1 *STOPS version of isomap to optimize over integer k.*

Description

Free parameter is k.

Usage

```
stop_isomap1(
  dis,
  theta = 3,
  weightmat = NULL,
  ndim = 2,
  init = NULL,
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
   "cassociation", "cnonmonotonicity", "cfunctionality", "ccomplexity", "cfaithfulness",
    "cregularity", "chierarchy", "cconvexity", "cstriatedness", "coutlying",
    "cskinniness", "csparsity", "cstringiness", "cclumpiness", "cinequality"),
  strucweight = rep(1/length(structures), length(structures)),
  strucpars,
  verbose = 0,
  type = c("additive", "multiplicative"),
  itmax = NULL
)
```

Arguments

dis	numeric matrix or dist object of a matrix of proximities
theta	the number of shortest dissimilarities retained for a point (nearest neighbours), the isomap parameter. Must be a numeric scalar. Defaults to 3.
weightmat	(optional) a matrix of nonnegative weights
ndim	number of dimensions of the target space
init	(optional) initial configuration
stressweight	weight to be used for the fit measure; defaults to 1
structures	which structuredness indices to be included in the loss
strucweight	weight to be used for the structuredness indices; ; defaults to 1/#number of structures
strucpars	the parameters for the structuredness indices
verbose	numeric value hat prints information on the fitting process; >2 is extremely verbose
type	How to construct the target function for the multi objective optimization? Either 'additive' (default) or 'multiplicative'
itmax	placeholder for compatibility in stops call; not used

Details

Currently this version is a bit less flexible than the vegan one, as the only allowed parameter for isomap is the theta (k in isomap, no epsilon) and the shortest path is always estimated with argument "shortest". Also note that fragmentedOK is always set to TRUE which means that for theta that is too small only the largest conected group will be analyzed. If that's not wanted just set the theta higher.

Value

A list with the components

- stress: Not really stress but 1-GOF where GOF is the first element returned from cmdscale (the sum of the first ndim absolute eigenvalues divided by the sum of all absolute eigenvalues).
- stress.m: default normalized stress (sqrt explicitly normalized stress; really the stress this time)
- stoploss: the weighted loss value
- indices: the values of the structuredness indices
- parameters: the parameters used for fitting
- fit: the returned object of the fitting procedure
- stopobj: the stopobj object

---

stop_isomap2 *STOPS version of isomap over real epsilon.*

Description

Free parameter is eps.

Usage

```
stop_isomap2(
  dis,
  theta = stats::quantile(dis, 0.1),
  weightmat = NULL,
  ndim = 2,
  init = NULL,
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
   "cassociation", "cnonmonotonicity", "cfunctionality", "ccomplexity", "cfaithfulness",
    "cregularity", "chierarchy", "cconvexity", "cstriatedness", "coutlying",
    "cskinniness", "csparsity", "cstringiness", "cclumpiness", "cinequality"),
  strucweight = rep(1/length(structures), length(structures)),
  strucpars,
  verbose = 0,
  type = c("additive", "multiplicative"),
  itmax = NULL
)
```

Arguments

dis	numeric matrix or dist object of a matrix of proximities
theta	the number of shortest dissimilarities retained for a point (neighbourhood region), the isomap parameter. Defaults to the 0.1 quantile of the empirical distribution of dis.
weightmat	(optional) a matrix of nonnegative weights
ndim	number of dimensions of the target space
init	(optional) initial configuration
stressweight	weight to be used for the fit measure; defaults to 1
structures	which structuredness indices to be included in the loss
strucweight	weight to be used for the structuredness indices; ; defaults to 1/#number of structures
strucpars	the parameters for the structuredness indices
verbose	numeric value hat prints information on the fitting process; >2 is extremely verbose
type	How to construct the target function for the multi objective optimization? Either 'additive' (default) or 'multiplicative'
itmax	placeholder for compatibility in stops call; not used

Details

Currently this version is a bit less flexible than the vegan one, as the only allowed parameter for isomap is the theta (epsilon in isomap) and the shortest path is always estimated with argument "shortest". Also note that fragmentedOK is always set to TRUE which means that for theta that is too small only the largest conected group will be analyzed. If that's not wanted just set the theta higher.

Value

A list with the components

- stress: Not really stress but 1-GOF where GOF is the first element returned from cmdscale (the sum of the first ndim absolute eigenvalues divided by the sum of all absolute eigenvalues).
- stress.m: default normalized stress (sqrt explicitly normalized stress; really the stress this time)
- stoploss: the weighted loss value
- indices: the values of the structuredness indices
- parameters: the parameters used for fitting
- fit: the returned object of the fitting procedure
- stopobj: the stopobj object

stop_lmds *STOPS version of lMDS*

Description

STOPS version of lMDS

Usage

```
stop_lmds(
  dis,
  theta = c(2, 0.5),
  weightmat = NULL,
  init = NULL,
  ndim = 2,
  itmax = 5000,
  ...,
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
   "cassociation", "cnonmonotonicity", "cfunctionality", "ccomplexity", "cfaithfulness",
    "cregularity", "chierarchy", "cconvexity", "cstriatedness", "coutlying",
    "cskinniness", "csparsity", "cstringiness", "cclumpiness", "cinequality"),
  strucweight = rep(1/length(structures), length(structures)),
  strucpars,
  verbose = 0,
  type = c("additive", "multiplicative")
)
```

Arguments

dis	numeric matrix or dist object of a matrix of proximities
theta	the theta vector of powers; the first is k (for the neighbourhood), the second tau (for the penalty) . If a scalar is given it is recycled. Defaults to 2 and 0.5.
weightmat	(not used)
init	(optional) initial configuration
ndim	number of dimensions of the target space
itmax	number of iterations
...	additional arguments to be passed to the fitting procedure
stressweight	weight to be used for the fit measure; defaults to 1
structures	which structures to look for
strucweight	weight to be used for the structures; defaults to 0.5
strucpars	a list of parameters for the structuredness indices; each list element corresponds to one index in the order of the appeacrance in structures

verbose	numeric value hat prints information on the fitting process; >2 is extremely verbose
type	which weighting to be used in the multi-objective optimization? Either 'additive' (default) or 'multiplicative'.

Value

A list with the components

- stress: the stress
- stress.m: default normalized stress
- stoploss: the weighted loss value
- struc: the structuredness indices
- parameters: the parameters used for fitting (kappa, lambda)
- fit: the returned object of the fitting procedure
- stopobj: the stopobj object

`stop_powerelastic` *STOPS version of elastic scaling with powers for proximities and distances*

Description

This is power stress with free kappa and lambda but rho is fixed to -2 and the weights are delta.

Usage

```
stop_powerelastic(
  dis,
  theta = c(1, 1, -2),
  weightmat = NULL,
  init = NULL,
  ndim = 2,
  itmax = 1e+05,
  ...,
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
   "cassociation", "cnonmonotonicity", "cfunctionality", "ccomplexity", "cfaithfulness",
    "cregularity", "chierarchy", "cconvexity", "cstriatedness", "coutlying",
    "cskinniness", "csparsity", "cstringiness", "cclumpiness", "cinequality"),
  strucweight = rep(1/length(structures), length(structures)),
  strucpars,
  verbose = 0,
  type = c("additive", "multiplicative")
)
```

Arguments

dis	numeric matrix or dist object of a matrix of proximities
theta	the theta vector of powers; a vector of length two where the first element is kappa (for the fitted distances), the second lambda (for the observed proximities). If a scalar for the free parameters is given it is recycled. Defaults to 1 1.
weightmat	(optional) a matrix of nonnegative weights
init	(optional) initial configuration
ndim	number of dimensions of the target space
itmax	number of iterations
...	additional arguments to be passed to the fitting procedure
stressweight	weight to be used for the fit measure; defaults to 1
structures	which strcutures to look for
strucweight	weight to be used for the structures; defaults to 0.5
strucpars	a list of parameters for the structuredness indices; each list element corresponds to one index in the order of the appeacrance in structures
verbose	numeric value hat prints information on the fitting process; >2 is extremely verbose
type	which weighting to be used in the multi-objective optimization? Either 'additive' (default) or 'multiplicative'.

Value

A list with the components

- stress: the stress
- stress.m: default normalized stress
- stoploss: the weighted loss value
- struc: the structuredness indices
- parameters: the parameters used for fitting (kappa, lambda)
- fit: the returned object of the fitting procedure
- stopobj: the stopobj object

---

stop_powermds *STOPS version of powermds*

Description

This is power stress with free kappa and lambda but rho is fixed to 1, so no weight transformation.

Usage

```
stop_powermds(
  dis,
  theta = c(1, 1),
  weightmat = NULL,
  init = NULL,
  ndim = 2,
  itmax = 1e+05,
  ...,
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
   "cassociation", "cnonmonotonicity", "cfunctionality", "ccomplexity", "cfaithfulness",
    "cregularity", "chierarchy", "cconvexity", "cstriatedness", "coutlying",
    "cskinniness", "csparsity", "cstringiness", "cclumpiness", "cinequality"),
  strucweight = rep(1/length(structures), length(structures)),
  strucpars,
  verbose = 0,
  type = c("additive", "multiplicative")
)
```

Arguments

dis	numeric matrix or dist object of a matrix of proximities
theta	the theta vector of powers; a vector of length 2 where the first element is kappa (for the fitted distances), the second lambda (for the observed proximities). If a scalar is given it is recycled. Defaults to 1.
weightmat	(optional) a matrix of nonnegative weights
init	(optional) initial configuration
ndim	number of dimensions of the target space
itmax	number of iterations
...	additional arguments to be passed to the fitting procedure
stressweight	weight to be used for the fit measure; defaults to 1
structures	which structures to look for
strucweight	weight to be used for the structures; defaults to 0.5
strucpars	a list of parameters for the structuredness indices; each list element corresponds to one index in the order of the appeacrance in structures
verbose	numeric value hat prints information on the fitting process; >2 is extremely verbose
type	which weighting to be used in the multi-objective optimization? Either 'additive' (default) or 'multiplicative'.

Value

A list with the components

- stress: the stress
- stress.m: default normalized stress
- stoploss: the weighted loss value
- struc: the structuredness indices
- parameters: the parameters used for fitting (kappa, lambda)
- fit: the returned object of the fitting procedure
- stopobj: the stopobj object

`stop_powersammon` *STOPS version of sammon with powers*

Description

This is power stress with free kappa and lambda but rho is fixed to -1 and the weights are delta.

Usage

```
stop_powersammon(
  dis,
  theta = c(1, 1),
  weightmat = NULL,
  init = NULL,
  ndim = 2,
  itmax = 1e+05,
  ...,
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
   "cassociation", "cnonmonotonicity", "cfunctionality", "ccomplexity", "cfaithfulness",
    "cregularity", "chierarchy", "cconvexity", "cstriatedness", "coutlying",
    "cskinniness", "csparsity", "cstringiness", "cclumpiness", "cinequality"),
  strucweight = rep(1/length(structures), length(structures)),
  strucpars,
  verbose = 0,
  type = c("additive", "multiplicative")
)
```

Arguments

`dis`	numeric matrix or dist object of a matrix of proximities
`theta`	the theta vector of powers; a vector of length two where the first element is kappa (for the fitted distances), the second lambda (for the observed proximities). If a scalar is given it is recycled for the free parameters. Defaults to 1 1.
`weightmat`	(optional) a matrix of nonnegative weights
`init`	(optional) initial configuration

ndim	number of dimensions of the target space
itmax	number of iterations
...	additional arguments to be passed to the fitting procedure
stressweight	weight to be used for the fit measure; defaults to 1
structures	which structures to look for
strucweight	weight to be used for the structures; defaults to 0.5
strucpars	a list of parameters for the structuredness indices; each list element corresponds to one index in the order of the appeacrance in structures
verbose	numeric value hat prints information on the fitting process; >2 is extremely verbose
type	which weighting to be used in the multi-objective optimization? Either 'additive' (default) or 'multiplicative'.

Value

A list with the components

- stress: the stress
- stress.m: default normalized stress
- stoploss: the weighted loss value
- struc: the structuredness indices
- parameters: the parameters used for fitting (kappa, lambda)
- fit: the returned object of the fitting procedure
- stopobj: the stopobj object

---

stop_powerstress *STOPS version of powerstress*

Description

Power stress with free kappa and lambda and rho.

Usage

```
stop_powerstress(
  dis,
  theta = c(1, 1, 1),
  weightmat = NULL,
  init = NULL,
  ndim = 2,
  itmax = 10000,
  ...,
  stressweight = 1,
```

```
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
   "cassociation", "cnonmonotonicity", "cfunctionality", "ccomplexity", "cfaithfulness",
    "cregularity", "chierarchy", "cconvexity", "cstriatedness", "coutlying",
    "cskinniness", "csparsity", "cstringiness", "cclumpiness", "cinequality"),
  strucweight = rep(1/length(structures), length(structures)),
  strucpars,
  verbose = 0,
  type = c("additive", "multiplicative")
)
```

Arguments

dis	numeric matrix or dist object of a matrix of proximities
theta	the theta vector of powers; the first is kappa (for the fitted distances), the second lambda (for the observed proximities), the third nu (for the weights). If a scalar is given it is recycled. Defaults to 1 1 1.
weightmat	(optional) a matrix of nonnegative weights
init	(optional) initial configuration
ndim	number of dimensions of the target space
itmax	number of iterations
...	additional arguments to be passed to the fitting procedure
stressweight	weight to be used for the fit measure; defaults to 1
structures	a character vector listing the structure indices to use. They always are called "cfoo" with foo being the structure.
strucweight	weight to be used for the structures; defaults to 1/number of structures
strucpars	a list of parameters for the structuredness indices; each list element corresponds to one index in the order of the appeacrance in structures
verbose	numeric value hat prints information on the fitting process; >2 is extremely verbose
type	which weighting to be used in the multi-objective optimization? Either 'additive' (default) or 'multiplicative'.

Value

A list with the components

- stress: the stress
- stress.m: default normalized stress
- stoploss: the weighted loss value
- struc: the structuredness indices
- parameters: the parameters used for fitting (kappa, lambda, nu)
- fit: the returned object of the fitting procedure
- stopobj: the stopobj object

`stop_rpowerstress` *STOPS version of restricted powerstress*

Description

STOPS version of restricted powerstress

Usage

```
stop_rpowerstress(
  dis,
  theta = c(1, 1, 1),
  weightmat = NULL,
  init = NULL,
  ndim = 2,
  itmax = 10000,
  ...,
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
   "cassociation", "cnonmonotonicity", "cfunctionality", "ccomplexity", "cfaithfulness",
    "cregularity", "chierarchy", "cconvexity", "cstriatedness", "coutlying",
    "cskinniness", "csparsity", "cstringiness", "cclumpiness", "cinequality"),
  strucweight = rep(1/length(structures), length(structures)),
  strucpars,
  verbose = 0,
  type = c("additive", "multiplicative")
)
```

Arguments

`dis`	numeric matrix or dist object of a matrix of proximities
`theta`	the theta vector of powers; the first two arguments are for kappa and lambda and should be equal (for the fitted distances and observed proximities), the third nu (for the weights). Internally the kappa and lambda are equated. If a scalar is given it is recycled (so all elements of theta are equal); if a vector of length 2 is given, it gets expanded to c(theta[1],theta[1],theta[2]). Defaults to 1 1 1.
`weightmat`	(optional) a matrix of nonnegative weights
`init`	(optional) initial configuration
`ndim`	number of dimensions of the target space
`itmax`	number of iterations. default is 10000.
`...`	additional arguments to be passed to the fitting procedure powerStressMin
`stressweight`	weight to be used for the fit measure; defaults to 1
`structures`	a character vector listing the structure indices to use. They always are called "cfoo" with foo being the structure.

strucweight	weight to be used for the structures; defaults to 1/number of structures
strucpars	a list of list of parameters for the structuredness indices; each list element corresponds to one index in the order of the appearance in structures vector. See examples.
verbose	numeric value hat prints information on the fitting process; >2 is extremely verbose
type	which weighting to be used in the multi-objective optimization? Either 'additive' (default) or 'multiplicative'.

Value

A list with the components

- stress: the stress
- stress.m: default normalized stress
- stoploss: the weighted loss value
- struc: the structuredness indices
- parameters: the parameters used for fitting (kappa=lambda, nu)
- fit: the returned object of the fitting procedure
- stopobj: the stopobj object

---

stop_rstress *STOPS version of rstress*

Description

Free parameter is kappa for the fitted distances.

Usage

```
stop_rstress(
  dis,
  theta = 1,
  weightmat = NULL,
  init = NULL,
  ndim = 2,
  itmax = 1e+05,
  ...,
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
   "cassociation", "cnonmonotonicity", "cfunctionality", "ccomplexity", "cfaithfulness",
    "cregularity", "chierarchy", "cconvexity", "cstriatedness", "coutlying",
    "cskinniness", "csparsity", "cstringiness", "cclumpiness", "cinequality"),
  strucweight = rep(1/length(structures), length(structures)),
  strucpars,
```

```
  verbose = 0,
  type = c("additive", "multiplicative")
)
```

Arguments

dis	numeric matrix or dist object of a matrix of proximities
theta	the theta vector of powers; this must be a scalar of the kappa transformation for the fitted distances proximities. Defaults to 1. Note the kappa here differs from Jan's version where the parameter was called r and the relationship is r=kappa/2 or kappa=2r.
weightmat	(optional) a matrix of nonnegative weights
init	(optional) initial configuration
ndim	number of dimensions of the target space
itmax	number of iterations
...	additional arguments to be passed to the fitting procedure
stressweight	weight to be used for the fit measure; defaults to 1
structures	which structuredness indices to be included in the loss
strucweight	weight to be used for the structuredness indices; ; defaults to 1/#number of structures
strucpars	the parameters for the structuredness indices
verbose	numeric value hat prints information on the fitting process; >2 is extremely verbose
type	How to construct the target function for the multi objective optimization? Either 'additive' (default) or 'multiplicative'

Value

A list with the components

- stress: the stress
- stress.m: default normalized stress
- stoploss: the weighted loss value
- indices: the values of the structuredness indices
- parameters: the parameters used for fitting
- fit: the returned object of the fitting procedure
- stopobj: the stopobj object

stop_sammon *STOPS version of Sammon mapping*

Description

Uses MASS::sammon. The free parameter is lambda for power transformations of the observed proximities. The fitted distances power is internally fixed to 1 and the power for the weights=delta is -1.

Usage

```
stop_sammon(
  dis,
  theta = 1,
  ndim = 2,
  init = NULL,
  weightmat = NULL,
  itmax = 1000,
  ...,
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
   "cassociation", "cnonmonotonicity", "cfunctionality", "ccomplexity", "cfaithfulness",
   "chierarchy", "cconvexity", "cstriatedness", "coutlying", "cskinniness", "csparsity",
    "cstringiness", "cclumpiness", "cinequality"),
  strucweight = rep(1/length(structures), length(structures)),
  strucpars,
  verbose = 0,
  type = c("additive", "multiplicative")
)
```

Arguments

dis	numeric matrix or dist object of a matrix of proximities
theta	the theta vector of powers; this must be a scalar of the lambda transformation for the observed proximities. Defaults to 1.
ndim	number of dimensions of the target space
init	(optional) initial configuration
weightmat	a matrix of nonnegative weights. Has no effect here.
itmax	number of iterations
...	additional arguments to be passed to the fitting procedure
stressweight	weight to be used for the fit measure; defaults to 1
structures	which structuredness indices to be included in the loss
strucweight	weight to be used for the structuredness indices; ; defaults to 1/#number of structures

strucpars	the parameters for the structuredness indices
verbose	numeric value hat prints information on the fitting process; >2 is extremely verbose
type	How to construct the target function for the multi objective optimization? Either 'additive' (default) or 'multiplicative'

Value

A list with the components

- stress: the stress
- stress.m: default normalized stress
- stoploss: the weighted loss value
- indices: the values of the structuredness indices
- parameters: the parameters used for fitting
- fit: the returned object of the fitting procedure
- stopobj: the stopobj object

---

stop_sammon2 *Another STOPS version of Sammon mapping models (via smacofSym)*

Description

Uses Smacof, so it can deal with a weight matrix too. The free parameter is lambda for power transformations of the observed proximities. The fitted distances power is internally fixed to 1 and the power for the weights=delta is -1.

Usage

```
stop_sammon2(
  dis,
  theta = 1,
  ndim = 2,
  weightmat = NULL,
  init = NULL,
  itmax = 1000,
  ...,
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
   "cassociation", "cnonmonotonicity", "cfunctionality", "ccomplexity", "cfaithfulness",
    "cregularity", "chierarchy", "cconvexity", "cstriatedness", "coutlying",
    "cskinniness", "csparsity", "cstringiness", "cclumpiness", "cinequality"),
  strucweight = rep(1/length(structures), length(structures)),
  strucpars,
  verbose = 0,
  type = c("additive", "multiplicative")
)
```

Arguments

dis	numeric matrix or dist object of a matrix of proximities
theta	the theta vector of powers; this must be a scalar of the lambda transformation for the observed proximities. Defaults to 1.
ndim	number of dimensions of the target space
weightmat	(optional) a matrix of nonnegative weights
init	(optional) initial configuration
itmax	number of iterations
...	additional arguments to be passed to the fitting procedure
stressweight	weight to be used for the fit measure; defaults to 1
structures	which structuredness indices to be included in the loss
strucweight	weight to be used for the structuredness indices; ; defaults to 1/#number of structures
strucpars	the parameters for the structuredness indices
verbose	numeric value hat prints information on the fitting process; >2 is extremely verbose
type	How to construct the target function for the multi objective optimization? Either 'additive' (default) or 'multiplicative'.

Value

A list with the components

- stress: the stress-1 (sqrt(stress.m))
- stress.m: default normalized stress (used for STOPS)
- stoploss: the weighted loss value
- indices: the values of the structuredness indices
- parameters: the parameters used for fitting
- fit: the returned object of the fitting procedure
- stopobj: the stopobj object

---

stop_smacofSphere *STOPS versions of smacofSphere models*

Description

The free parameter is lambda for power transformations the observed proximities. The fitted distances power is internally fixed to 1 and the power for the weights is 1.

Usage

```
stop_smacofSphere(
  dis,
  theta = 1,
  ndim = 2,
  weightmat = NULL,
  init = NULL,
  itmax = 1000,
  ...,
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
   "cassociation", "cnonmonotonicity", "cfunctionality", "ccomplexity", "cfaithfulness",
    "cregularity", "chierarchy", "cconvexity", "cstriatedness", "coutlying",
    "cskinniness", "csparsity", "cstringiness", "cclumpiness", "cinequality"),
  strucweight = rep(1/length(structures), length(structures)),
  strucpars,
  verbose = 0,
  type = c("additive", "multiplicative")
)
```

Arguments

dis	numeric matrix or dist object of a matrix of proximities
theta	the theta vector of powers; this must be a scalar of the lambda transformation for the observed proximities. Defaults to 1.
ndim	number of dimensions of the target space
weightmat	(optional) a matrix of nonnegative weights
init	(optional) initial configuration
itmax	number of iterations
...	additional arguments to be passed to the fitting procedure
stressweight	weight to be used for the fit measure; defaults to 1
structures	which structuredness indices to be included in the loss
strucweight	weight to be used for the structuredness indices; ; defaults to 1/#number of structures
strucpars	the parameters for the structuredness indices
verbose	numeric value hat prints information on the fitting process; >2 is extremely verbose
type	How to construct the target function for the multi objective optimization? Either 'additive' (default) or 'multiplicative'

Value

A list with the components

- stress: the stress

- stress.m: default normalized stress
- stoploss: the weighted loss value
- indices: the values of the structuredness indices
- parameters: the parameters used for fitting
- fit: the returned object of the fitting procedure
- stopobj: the stopobj object

`stop_smacofSym` *STOPS version of smacofSym models*

Description

The free parameter is lambda for power transformations the observed proximities. The fitted distances power is internally fixed to 1 and the power for the weights is 1.

Usage

```
stop_smacofSym(
  dis,
  theta = 1,
  ndim = 2,
  weightmat = NULL,
  init = NULL,
  itmax = 1000,
  ...,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
   "cassociation", "cnonmonotonicity", "cfunctionality", "ccomplexity", "cfaithfulness",
   "chierarchy", "cconvexity", "cstriatedness", "coutlying", "cskinniness", "csparsity",
    "cstringiness", "cclumpiness", "cinequality"),
  stressweight = 1,
  strucweight = rep(1/length(structures), length(structures)),
  strucpars,
  verbose = 0,
  type = c("additive", "multiplicative")
)
```

Arguments

`dis`	numeric matrix or dist object of a matrix of proximities
`theta`	the theta vector; must be a scalar for the lambda (proximity) transformation. Defaults to 1.
`ndim`	number of dimensions of the target space
`weightmat`	(optional) a matrix of nonnegative weights
`init`	(optional) initial configuration

itmax	number of iterations
...	additional arguments to be passed to the fitting
structures	which structuredness indices to be included in the loss
stressweight	weight to be used for the fit measure; defaults to 1
strucweight	weight to be used for the structuredness indices; ; defaults to 1/#number of structures
strucpars	the parameters for the structuredness indices
verbose	numeric value hat prints information on the fitting process; >2 is extremely verbose
type	How to construct the target function for the multi objective optimization? Either 'additive' (default) or 'multiplicative'

Value

A list with the components

- stress: the stress-1 (sqrt(stress.m))
- stress.m: default normalized stress (used for STOPS)
- stoploss: the weighted loss value
- indices: the values of the structuredness indices
- parameters: the parameters used for fitting
- fit: the returned object of the fitting procedure
- stopobj: the stops object

---

stop_sstress *STOPS version of sstress*

Description

Free parameter is lambda for the observed proximities. Fitted distances are transformed with power 2, weights have exponent of 1. Note that the lambda here works as a multiplicator of 2 (as sstress has f(delta^2)).

Usage

```
stop_sstress(
  dis,
  theta = 1,
  weightmat = NULL,
  init = NULL,
  ndim = 2,
  itmax = 1e+05,
  ...,
```

```
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
   "cassociation", "cnonmonotonicity", "cfunctionality", "ccomplexity", "cfaithfulness",
    "cregularity", "chierarchy", "cconvexity", "cstriatedness", "coutlying",
    "cskinniness", "csparsity", "cstringiness", "cclumpiness", "cinequality"),
  strucweight = rep(1/length(structures), length(structures)),
  strucpars,
  verbose = 0,
  type = c("additive", "multiplicative")
)
```

Arguments

dis	numeric matrix or dist object of a matrix of proximities
theta	the theta vector of powers; this must be a scalar of the lambda transformation for the observed proximities. Defaults to 1. Note that the lambda here works as a multiplicator of 2 (as sstress has f(delta^2)).
weightmat	(optional) a matrix of nonnegative weights
init	(optional) initial configuration
ndim	the number of dimensions of the target space
itmax	number of iterations
...	additional arguments to be passed to the fitting procedure
stressweight	weight to be used for the fit measure; defaults to 1
structures	which structuredness indices to be included in the loss
strucweight	weight to be used for the structuredness indices; ; defaults to 1/#number of structures
strucpars	the parameters for the structuredness indices
verbose	numeric value hat prints information on the fitting process; >2 is extremely verbose
type	How to construct the target function for the multi objective optimization? Either 'additive' (default) or 'multiplicative'

Value

A list with the components

- stress: the stress
- stress.m: default normalized stress
- stoploss: the weighted loss value
- indices: the values of the structuredness indices
- parameters: the parameters used for fitting
- fit: the returned object of the fitting procedure
- stopobj: the stopobj object

summary.cmdscale *S3 summary method for cmdscale*

Description

S3 summary method for cmdscale

Usage

```
## S3 method for class 'cmdscale'
summary(object, ...)
```

Arguments

object	object of class cmdscale
...	additional arguments

Value

No return value, just prints.

summary.sammon *S3 summary method for sammon*

Description

S3 summary method for sammon

Usage

```
## S3 method for class 'sammon'
summary(object, ...)
```

Arguments

object	object of class sammon
...	additional arguments

Value

No return value, just prints.

summary.smacofP *S3 summary method for smacofP*

Description

S3 summary method for smacofP

Usage

```
## S3 method for class 'smacofP'
summary(object, ...)
```

Arguments

object	object of class smacofP
...	additional arguments

Value

an object of class summary.smacofP

summary.stops *S3 summary method for stops*

Description

S3 summary method for stops

Usage

```
## S3 method for class 'stops'
summary(object, ...)
```

Arguments

object	object of class stops
...	addditional arguments

Value

object of class 'summary.stops'

Swissroll *Swiss roll*

Description

A swiss roll data example where 150 data points are arranged on a swiss roll embedded in a 3D space.

Usage

```
data(Swissroll)
```

Format

A data frame with 150 rows and 4 columns

Details

A data frame with the variables (columns)

- x The x axis coordinate for each point
- y The y axis coordinate for each point
- z The z axis coordinate for each point
- col a color code for each point with points along the y axis having the same color (based on the viridis palette)

tgpoptim *Bayesian Optimization by a (treed) Bayesian Gaussian Process Prior (with jumps to linear models) surrogate model Essentially a wrapper for the functionality in tgp that has the same slots as optim with defaults for STOPS models.*

Description

Bayesian Optimization by a (treed) Bayesian Gaussian Process Prior (with jumps to linear models) surrogate model Essentially a wrapper for the functionality in tgp that has the same slots as optim with defaults for STOPS models.

Usage

```
tgpoptim(
  x,
  fun,
  ...,
  initpoints = 10,
  lower,
  upper,
  acc = 1e-08,
  itmax = 10,
  verbose = 0,
  model = "bgp"
)
```

Arguments

x	optional starting values
fun	function to minimize
...	additional arguments to be passed to the function to be optimized
initpoints	the number of points to sample initially to fit the surrogate model
lower	The lower contraints of the search region
upper	The upper contraints of the search region
acc	if the numerical accuracy of two successive target function values is below this, stop the optimization; defaults to 1e-8
itmax	maximum number of iterations
verbose	numeric value hat prints information on the fitting process; >2 is extremely verbose
model	which surrogate model class to use (currently uses defaults only, will extend this to tweak the model)

Value

A list with the components (for compatiility with optim)

- par The position of the optimum in the search space (parameters that minimize the function; argmin fun).
- value The value of the objective function at the optimum (min fun). Note we do not use the last value in the candidate list but the best candidate (which can but need not coincide).
- svalue The value of the surrogate objective function at the optimal parameters
- counts The number of iterations performed at convergence with entries fnction for the number of iterations and gradient which is always NA at the moment
- convergence 0 successful completion by the accd or acc criterion, 1 indicate iteration limit was reached, 99 is a problem
- message is NULL (only for compatibility or future use)
- history the improvement history
- tgpout the output of the tgp model

Examples

```
fbana <- function(x) {
x1 <- x[1]
x2 <- x[2]
100 * (x2 - x1 * x1)^2 + (1 - x1)^2
}
res1<-tgpoptim(c(-1.2,1),fbana,lower=c(-5,-5),upper=c(5,5),acc=1e-16,itmax=20)
res1

fwild <- function (x) 10*sin(0.3*x)*sin(1.3*x^2) + 0.00001*x^4 + 0.2*x+80
plot(fwild, -50, 50, n = 1000, main = "Bayesian GP Optimization minimizing 'wild function'")
set.seed(210485)
res2<-tgpoptim(50, fwild,lower=-50,upper=50,acc=1e-16,itmax=20,model="btgpllm")
points(res2$par,res2$value,col="red",pch=19)
res2
```

`torgerson` *Torgerson scaling*

Description

Torgerson scaling

Usage

```
torgerson(delta, p = 2)
```

Arguments

`delta`	symmetric, numeric matrix of distances
`p`	target space dimensions

Value

a matrix (a Torgerson scaling configuration)

Index