Package ‘cops’

January 19, 2023

Title Cluster Optimized Proximity Scaling

Version 1.3-1

Date 2023-01-19

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

Description Multidimensional scaling (MDS) methods that aim at pronouncing the clustered appearance of the configuration (Rusch, Mair & Hornik, 2021, <doi:10.1080/10618600.2020.1869027>). They achieve this by transforming proximities/distances with power functions and augment the fitting criterion with a clusteredness index, the OPTICS Cordillera (Rusch, Hornik & Mair, 2018, <doi:10.1080/10618600.2017.1349664>). There are two variants: One for finding the configuration directly (COPS-C) for ratio, power, interval and non-metric MDS (Borg & Groenen, 2005, ISBN:978-0-387-28981-6), and one for using the augmented fitting criterion to find optimal parameters (P-COPS). The package contains various functions, wrappers, methods and classes for fitting, plotting and displaying different MDS models in a COPS framework like ratio, interval and non-metric MDS for COPS-C and P-COPS with Torgerson scaling (Torgerson, 1958, ISBN:978-0471879459), scaling by majorizing a complex function (SMACOF; de Leeuw, 1977, <https://escholarship.org/uc/item/4ps3b5mj>), Sammon mapping (Sammon, 1969, <doi:10.1109/T-C.1969.222678>), elastic scaling (McGee, 1966, <doi:10.1111/j.2044-8317.1966.tb00367.x>), s-stress (Takane, Young & de Leeuw, 1977, <doi:10.1007/BF02293745>), r-stress (de Leeuw, Groenen & Mair, 2016, <https://rpubs.com/deleeuw/142619>), power stress (Buja & Swayne, 2002 <doi:10.1007/s00357-001-0031-0>), restricted power stress, approximate power stress, power elastic scaling, power Sammon mapping (for all Rusch, Mair & Hornik, 2021, <doi:10.1080/10618600.2020.1869027>). All of these models can also solely be fit as MDS with power transformations. The package further contains a function for pattern search optimization, the ``Adaptive Luus-Jaakola Algorithm'' (Rusch, Mair & Hornik, 2021,<doi:10.1080/10618600.2020.1869027>).

Depends R (>= 3.1.2), cordillera (>= 0.7-2), smacof (>= 1.10-4)

Imports MASS, minqa, pso, scatterplot3d, NlcOptim, Rsolnp, dfoptim, subplex, cmaes, crs, nloptr, rgenoud, GenSA

Suggests R.rsp, rmarkdown

VignetteBuilder R.rsp

License GPL-2 | GPL-3

LazyData true

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

RoxygenNote 7.2.3

Encoding UTF-8

NeedsCompilation no

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

Repository CRAN

Date/Publication 2023-01-19 15:50:02 UTC

R topics documented:

`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)

`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 stops package.

Value

Object of class "cmdscaleE' and 'cmdscale' extending `cmdscale`. This wrapper only adds an extra slot to the list with the call, adds column labels to the $points and assigns S3 class 'cmdscaleE' and 'cmdscale'.

Examples

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

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

Description

conf_adjust: a function to procrustes adjust two matrices

Usage

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

Arguments

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

Value

a list with ref.conf being the reference configuration, other.conf the adjusted coniguration and comparison.conf the comparison configuration cops *cops: cluster optimized proximity scaling*

Description

About the package cops: Cluster optimized proximity scaling (COPS) refers to multidimensional scaling methods that aim at pronouncing the clustered appearance of the configuration. They achieve this by transforming proximities/distances with power functions and augment the fitting criterion with a clusteredness index, the OPTICS Cordillera (Rusch, Hornik & Mair 2018). There are two variants: One for finding the configuration directly for given parameters (COPS-C), and one for using the augmented fitting criterion to find optimal parameters for the power transformations (P-COPS). The package contains various functions, wrappers, methods and classes for fitting, plotting and displaying different MDS models in a COPS framework like Torgerson scaling, SMACOF, Sammon mapping, elastic scaling, symmetric SMACOF, spherical SMACOF, sstress, rstress, powermds, power elastic scaling, power sammon mapping, powerstress. All of these models can also solely be fit as MDS with power transformations. The package further contains functions for optimization (Adaptive LJ Algorithmus).

About the function cops: The high level function allows for minimizing copstress for a clustered MDS configuration. Allows to choose COPS-C (finding a configuration from copstress with cordillera penalty) and profile COPS (finding hyperparameters for MDS models with power transformations). It is a wrapper for copstressMin and pcops.

Usage

```
cops(
  dis,
  variant = c("1", "2", "Variant1", "Variant2", "v1", "v2", "COPS-C", "P-COPS",
   "configuration-c", "profile", "copstress-c", "p-copstress", "COPS-P", "copstress-p",
     "cops-c", "p-cops", "copsc", "pcops"),
  ...
)
```

Arguments

`dis`	a dissimilarity matrix or a dist object
`variant`	a character string specifying which variant of COPS to fit. Allowed is any of the following "1","2","Variant1","Variant2","v1","v2","COPS-C","P-COPS","configuration-c","profile","copstress-c","p-copstress". Defaults to "COPS-C".
`...`	arguments to be passed to `copstressMin` (for Variant 1) or `pcops` (for Variant 2).

Details

The cops package provides five categories of important functions:

Models & Algorithms:

- cops() ... high level interface to fit COPS models as described in Rusch et al. (2021). By setting cordweight to zero they can also be used to fit metric MDS for many different models, see below.
- copstressMin()... The workhorse for fitting a COPS-C model. Can also be called directly.
- pcops()... The workhorse for fitting a P-COPS model. Can also be called directly.
- powerStressMin()... a workhorse for fitting s-stress, r-stress (de Leeuw, 2014), p-stress (e.g., Rusch et al., 2021), Sammon mapping with power transformations (powersammon) and elastic scaling with power transformation (powerelastic). They can conveniently also be fitted via the cops functions and setting stressweight=1 and cordweight or by the dedicated functions starting with cops_XXX where XXX is the method and setting stressweight=1 and cordweight=0. It uses the nested majorization algorithm for r-stress of De Leeuw (2014).

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 to be used with cops
- copstress() ... a function to calculate copstress (Rusch et al., 2021)
- cop_smacofSym(), cop_sammon(), cop_cmdscale(), cop_rstress(), cop_powerstress(), cop_smacofSphere(), cop_sammon2(), cop_elastic(), cop_sstress(), cop_powerelastic(), cop_powersammon(): cop versions of these MDS models.

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, plot3dstatic.

References:

- Rusch, T., Hornik, K. & Mair, P. (2018) Assessing and quantifying clusteredness: The OPTICS Cordillera. Journal of Computational and Graphical Statistics, 27 (1), 220-233. doi:10.1080/10618600.2017.1349664
- Rusch, T., Mair, P. & Hornik, K. (2021) Cluster optimized proximity scaling. Journal of Computational and Graphical Statistics. doi:10.1080/10618600.2020.1869027

Authors: Thomas Rusch, Jan de Leeuw, Patrick Mair

Maintainer: Thomas Rusch

Value

For COPS-C Variant 1 see `copstressMin`, for P-COPS Variant 2 see `pcops`

Examples

```
data(BankingCrisesDistances)


# shorthand function for COPS-C (finding configuration with copstress)
```

```
res<-cops(BankingCrisesDistances[,1:69],variant="COPS-C",
          stressweight=0.98,cordweight=0.02,itmax=1000)
# Note: itmax is very small here for illustration; will give a non-convergence
# warning of the optimizer which disappears at itmax=275000

res
summary(res)
plot(res)
plot(res,"reachplot")
plot(res,"transplot")
plot(res,"Shepard")
#shorthand function for P-COPS (hyperparameter search for powerstress)
res<-cops(BankingCrisesDistances[,1:69],variant="P-COPS")
res
summary(res)
plot(res)
plot(res,"reachplot")
plot(res,"transplot")
plot(res,"Shepard")

dis<-as.matrix(smacof::kinshipdelta)

#COPS-C with equal weight to stress and cordillera
res1<-cops(dis,variant="COPS-C",stressweight=0.5,cordweight=0.5,
          minpts=2,itmax=500) #use higher itmax in real
res1
summary(res1)
plot(res1)
plot(res1,"reachplot")



#s-stress type copstress (i.e. kappa=2, lambda=2)
res3<-cops(dis,variant="COPS-C",kappa=2,lambda=2,stressweight=0.5,cordweight=0.5)
res3
summary(res3)
plot(res3)


# power-stress type profile copstress
# search for optimal kappa and lambda between
# kappa=0.5,lambda=0.5 and kappa=2,lambda=5
# nu is fixed on -1
ws<-1/dis
diag(ws)<-1
res5<-cops(dis,variant="P-COPS",loss="powerstress",
          theta=c(1.4,3,-1), lower=c(1,0.5,-1),upper=c(3,5,-1),
          weightmat=ws, stressweight=0.9,cordweight=0.1)
res5
summary(res5)
plot(res5)
```

copstress *Calculates copstress for given MDS object*

Description

Calculates copstress for given MDS object

Usage

```
copstress(
  obj,
  stressweight = 1,
  cordweight = 5,
  q = 1,
  minpts = 2,
  epsilon = 10,
  rang = NULL,
  verbose = 0,
  normed = TRUE,
  scale = c("std", "sd", "proc", "none"),
  init,
  ...
)
```

Arguments

`obj`	MDS object (supported are sammon, cmdscale, smacof, rstress, powermds)
`stressweight`	weight to be used for the fit measure; defaults to 1
`cordweight`	weight to be used for the cordillera; defaults to 0.5
`q`	the norm of the cordillera; defaults to 1
`minpts`	the minimum points to make up a cluster in OPTICS; defaults to 2
`epsilon`	the epsilon parameter of OPTICS, the neighbourhood that is checked; defaults to 10
`rang`	range of the distances (min distance minus max distance). If NULL (default) the cordillera will be normed to each configuration's maximum distance, so an absolute value of goodness-of-clusteredness.
`verbose`	numeric value hat prints information on the fitting process; >2 is very verbose (copstress level), >3 is extremely (up to MDS optimization level)
`normed`	should the cordillera be normed; defaults to TRUE
`scale`	should the configuration be scale adjusted.
`init`	a reference configuration when doing procrustes adjustment
`...`	additional arguments to be passed to the cordillera function

Value

A list with the components

- copstress: the weighted loss value
- OC: the Optics cordillera value
- parameters: the parameters used for fitting (kappa, lambda)
- cordillera: the cordillera object

`copstressMin` *Fitting a COPS-C Model (COPS Variant 1).*

Description

Minimizing Copstress to obtain a clustered MDS configuration with given hyperparameters theta.

Usage

```
copstressMin(
  delta,
  kappa = 1,
  lambda = 1,
  nu = 1,
  theta = c(kappa, lambda, nu),
  type = c("ratio", "interval", "ordinal"),
  ties = "primary",
  weightmat = 1 - diag(nrow(delta)),
  ndim = 2,
  init = NULL,
  stressweight = 0.975,
  cordweight = 0.025,
  q = 1,
  minpts = ndim + 1,
  epsilon = 10,
  dmax = NULL,
  rang,
 optimmethod = c("NelderMead", "Newuoa", "BFGS", "SANN", "hjk", "solnl", "solnp",
   "subplex", "snomadr", "hjk-Newuoa", "hjk-BFGS", "BFGS-hjk", "Newuoa-hjk", "cmaes",
    "direct", "direct-Newuoa", "direct-BFGS", "genoud", "gensa"),
  verbose = 0,
  scale = c("sd", "rmsq", "std", "proc", "none"),
  normed = TRUE,
  accuracy = 1e-07,
  itmax = 5000,
  stresstype = c("stress-1", "stress"),
  ...
)
```

Arguments

`delta`	numeric matrix or dist object of a matrix of proximities
`kappa`	power transformation for fitted distances
`lambda`	power transformation for proximities
`nu`	power transformation for weights
`theta`	the theta vector of powers; the first is kappa (for the fitted distances if it exists), the second lambda (for the observed proximities if it exist), the third is nu (for the weights if it exists) . If less than three elements are is given as argument, it will be recycled. Defaults to 1 1 1. Will override any kappa, lmabda, nu parameters if they are given and do not match
`type`	what type of MDS to fit. Currently one of "ratio", "interval" or "ordinal". Default is "ratio".
`ties`	the handling of ties for ordinal (nonmetric) MDS. Possible are "primary" (default), "secondary" or "tertiary".
`weightmat`	(optional) a matrix of nonnegative weights; defaults to 1 for all off diagonals
`ndim`	number of dimensions of the target space
`init`	(optional) initial configuration
`stressweight`	weight to be used for the fit measure; defaults to 0.975
`cordweight`	weight to be used for the cordillera; defaults to 0.025
`q`	the norm of the cordillera; defaults to 1
`minpts`	the minimum points to make up a cluster in OPTICS; defaults to ndim+1
`epsilon`	the epsilon parameter of OPTICS, the neighbourhood that is checked; defaults to 10
`dmax`	The winsorization limit of reachability distances in the OPTICS Cordillera. If supplied, it should be either a numeric value that matches max(rang) or NULL; if NULL it is found as 1.5 times (for kappa >1) or 1 times (for kappa <=1) the maximum reachbility value of the power torgerson model with the same lambda. If dmax and rang are supplied and dmax is not max(rang), a warning is given and rang takes precedence.
`rang`	range of the reachabilities to be considered. If missing it is found from the initial configuration by taking 0 as the lower boundary and dmax (see above) as upper boundary. See also `cordillera`
`optimmethod`	What optimizer to use? Choose one string of 'Newuoa' (from package minqa), 'NelderMead', 'hjk' (Hooke-Jeeves algorithm from dfoptim), 'solnl' (from nlcOptim), 'solnp' (from Rsolnp), 'subplex' (from subplex), 'SANN' (simulated annealing), 'BFGS', 'snomadr' (from crs), 'genoud' (from rgenoud), 'gensa' (from GenSA), 'cmaes' (from cmaes) and 'direct' (from nloptr). See the according R packages for details on these solvers. There are also combinations that proved to work well good, like 'hjk-Newuoa', 'hjk-BFGS', 'BFGS-hjk', 'Newuoa-hjk', 'direct-Newuoa' and 'direct-BFGS' . Usually hjk, BFGS, newuoa, subplex and solnl work rather well in an acceptable time frame (depending on the smoothness of copstress). Default is 'hjk-Newuoa'.
`verbose`	numeric value hat prints information on the fitting process; >2 is very verbose

`scale`	Allows to scale the configuration for the OC (the scaled configuration is also returned as $conf). One of "none" (so no scaling), "sd" (configuration divided by the highest standard deviation of the columns), "std" (standardize all columns !NOTE: This does not preserve the relative distances of the optimal config), "proc" (procrustes adjustment to the initial fit) and "rmsq" (configuration divided by the maximum root mean square of the columns). Default is "sd".
`normed`	should the cordillera be normed; defaults to TRUE
`accuracy`	numerical accuracy, defaults to 1e-7
`itmax`	maximum number of iterations. Defaults to 5000. If itmax is (too) small, some optimizers will print warnings. For example, for optimizers using NEWUOA, an iteration number of 10*length(par)^2 is recommended. The number of parameters to optimize over for the COPS problem is number of objects * target space dimensions and can grow large very quickly, so being able to live with these warnings is probably a good idea.
`stresstype`	which stress to use in the copstress. Defaults to stress-1. If anything else is set, explicitly normed stress which is (stress-1)^2. Using stress-1 puts more weight on MDS fit.
`...`	additional arguments to be passed to the optimization procedure

Value

A list with the components

- delta: the original transformed dissimilarities
- obsdiss: the explicitly normed transformed dissimilarities (which are approximated by the fit)
- confdist: the fitted distances
- conf: the configuration to which the scaling of argument scale was applied
- confo: the unscaled but explicitly normed configuration returned from the fitting procedure. Scaling applied to confo gives conf.
- par, pars : the theta vector of powers tranformations (kappa,lambda,nu)
- niter: number of iterations of the optimizer.
- stress: the square root of explicitly normalized stress (calculated for confo).
- spp: stress per point
- ndim: number of dimensions
- model: Fitted model name with optimizer
- call: the call
- nobj: the number of objects
- type, loss, losstype: stresstype
- stress.m: The stress used for copstress. If stresstype="stress-1" this is like $stress else it is stress^2
- stress.en: another ways to calculate the stress
- deltaorig: the original untransformed dissimilarities

- copstress: the copstress loss value
- resmat: the matrix of residuals
- weightmat: the matrix of untransformed weights
- OC: the (normed) OPTICS Cordillera object (calculated for scaled conf)
- OCv: the (normed) OPTICS Cordillera value alone (calculated for scaled conf)
- optim: the object returned from the optimization procedure
- stressweight, cordweight: the weights of the stress and OC respectively (v_1 and v_2)
- optimmethod: The solver used
- type: the type of MDS fitted

Examples

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

#Copstress with equal weight to stress and cordillera
res1<-copstressMin(dis,stressweight=0.5,cordweight=0.5,
                   itmax=1000) #use higher itmax about 10000
res1
summary(res1)
plot(res1)  #super clustered
```

`cop_apstress` *PCOPS version of approximated power stress model.*

Description

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

Usage

```
cop_apstress(
  dis,
  theta = c(1, 1, 1),
  ndim = 2,
  weightmat = NULL,
  init = NULL,
  itmaxi = 1000,
  ...,
  stressweight = 1,
  cordweight = 0.5,
  q = 1,
  minpts = ndim + 1,
  epsilon = 10,
  rang = NULL,
  verbose = 0,
  normed = TRUE,
  scale = "sd",
  stresstype = "default"
)
```

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
`itmaxi`	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
`cordweight`	weight to be used for the cordillera; defaults to 0.5
`q`	the norm of the cordillera; defaults to 1
`minpts`	the minimum points to make up a cluster in OPTICS; defaults to ndim+1
`epsilon`	the epsilon parameter of OPTICS, the neighbourhood that is checked; defaults to 10
`rang`	range of the distances (min distance minus max distance). If NULL (default) the cordillera will be normed to each configuration's maximum distance, so an absolute value of goodness-of-clusteredness.
`verbose`	numeric value hat prints information on the fitting process; >2 is extremely verbose
`normed`	should the cordillera be normed; defaults to TRUE
`scale`	should the configuration be scale adjusted
`stresstype`	which stress to report. Only takes smacofs default stress currrently.

Value

A list with the components

- stress: the stress
- stress.m: default normalized stress
- copstress: the weighted loss value
- OC: the Optics cordillera value
- parameters: the parameters used for fitting (kappa, lambda)

- fit: the returned object of the fitting procedure (which has all smacofB elements and some more
- cordillera: the cordillera object

`cop_cmdscale` *PCOPS version of strain*

Description

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

Usage

```
cop_cmdscale(
  dis,
  theta = c(1, 1, 1),
  weightmat = NULL,
  ndim = 2,
  init = NULL,
  itmaxi = 1000,
  ...,
  stressweight = 1,
  cordweight = 0.5,
  q = 1,
  minpts = ndim + 1,
  epsilon = 10,
  rang = NULL,
  verbose = 0,
  scale = "sd",
  normed = TRUE,
  stresstype = "default"
)
```

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
`ndim`	number of dimensions of the target space
`init`	(optional) initial configuration
`itmaxi`	number of iterations. No effect here.
`...`	additional arguments to be passed to the fitting procedure
`stressweight`	weight to be used for the fit measure; defaults to 1

`cordweight`	weight to be used for the cordillera; defaults to 0.5
`q`	the norm of the corrdillera; defaults to 1
`minpts`	the minimum points to make up a cluster in OPTICS; defaults to ndim+1
`epsilon`	the epsilon parameter of OPTICS, the neighbourhood that is checked; defaults to 10
`rang`	range of the distances (min distance minus max distance). If NULL (default) the cordillera will be normed to each configuration's maximum distance, so an absolute value of goodness-of-clusteredness.
`verbose`	numeric value hat prints information on the fitting process; >2 is extremely verbose
`scale`	should the configuration be scale adjusted
`normed`	should the cordillera be normed; defaults to TRUE
`stresstype`	which stress to report. Only takes cmdscales default stress currrently.

Value

A list with the components

- stress: the stress
- stress.m: default normalized stress
- copstress: the weighted loss value
- OC: the Optics cordillera value
- parameters: the parameters used for fitting (kappa, lambda)
- fit: the returned object of the fitting procedure
- cordillera: the cordillera object

`cop_elastic` *PCOPS 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

```
cop_elastic(
  dis,
  theta = 1,
  ndim = 2,
  weightmat = 1,
  init = NULL,
  itmaxi = 1000,
  ...,
  stressweight = 1,
  cordweight = 0.5,
  q = 1,
  minpts = ndim + 1,
  epsilon = 10,
  rang = NULL,
  verbose = 0,
  normed = TRUE,
  scale = "sd",
  stresstype = "default"
)
```

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
`itmaxi`	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
`cordweight`	weight to be used for the cordillera; defaults to 0.5
`q`	the norm of the corrdillera; defaults to 1
`minpts`	the minimum points to make up a cluster in OPTICS; defaults to ndim+1
`epsilon`	the epsilon parameter of OPTICS, the neighbourhood that is checked; defaults to 10
`rang`	range of the distances (min distance minus max distance). If NULL (default) the cordillera will be normed to each configuration's maximum distance, so an absolute value of goodness-of-clusteredness.
`verbose`	numeric value hat prints information on the fitting process; >2 is extremely verbose
`normed`	should the cordillera be normed; defaults to TRUE
`scale`	should the configuration be scale adjusted
`stresstype`	which stress to report. Only takes smacofs default stress currrently.

Value

A list with the components

- stress: the stress

- stress.m: default normalized stress
- copstress: the weighted loss value
- OC: the Optics cordillera value
- parameters: the parameters used for fitting (kappa, lambda)
- fit: the returned object of the fitting procedure
- cordillera: the cordillera object

cop_powerelastic *PCOPS version of elastic scaling with powers*

Description

PCOPS version of elastic scaling with powers

Usage

```
cop_powerelastic(
  dis,
  theta = c(1, 1),
  weightmat = 1 - diag(nrow(dis)),
  init = NULL,
  ndim = 2,
  itmaxi = 10000,
  ...,
  stressweight = 1,
  cordweight = 0.5,
  q = 1,
  minpts = ndim + 1,
  epsilon = 10,
  rang = NULL,
  verbose = 0,
  scale = "sd",
  normed = TRUE,
  stresstype = c("default", "stress1", "rawstress", "normstress", "enormstress",
    "enormstress1")
)
```

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
`itmaxi`	number of iterations. default is 10000.
`...`	additional arguments to be passed to the fitting procedure
`stressweight`	weight to be used for the fit measure; defaults to 1
`cordweight`	weight to be used for the cordillera; defaults to 0.5
`q`	the norm of the cordillera; defaults to 1
`minpts`	the minimum points to make up a cluster in OPTICS; defaults to ndim+1
`epsilon`	the epsilon parameter of OPTICS, the neighbourhood that is checked; defaults to 10
`rang`	range of the distances (min distance minus max distance). If NULL (default) the cordillera will be normed to each configuration's maximum distance, so an absolute value of goodness-of-clusteredness.
`verbose`	numeric value hat prints information on the fitting process; >2 is extremely verbose
`scale`	should the configuration be scale adjusted
`normed`	should the cordillera be normed; defaults to TRUE
`stresstype`	which stress to report? Defaults to explicitly normed stress

Value

A list with the components

- stress: the stress
- stress.m: default normalized stress
- copstress: the weighted loss value
- OC: the Optics cordillera value
- parameters: the parameters used for fitting (kappa, lambda)
- fit: the returned object of the fitting procedure
- cordillera: the cordillera object

`cop_powermds` *PCOPS version of powermds*

Description

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

Usage

```
cop_powermds(
  dis,
  theta = c(1, 1, 1),
  weightmat = 1 - diag(nrow(dis)),
  init = NULL,
  ndim = 2,
  itmaxi = itmaxi,
  ...,
  stressweight = 1,
  cordweight = 0.5,
  q = 1,
  minpts = ndim + 1,
  epsilon = 10,
  rang = NULL,
  verbose = 0,
  scale = "sd",
  normed = TRUE,
  stresstype = c("default", "stress1", "rawstress", "normstress", "enormstress",
    "enormstress1")
)
```

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
`itmaxi`	number of iterations. default is 10000.
`...`	additional arguments to be passed to the fitting procedure
`stressweight`	weight to be used for the fit measure; defaults to 1
`cordweight`	weight to be used for the cordillera; defaults to 0.5
`q`	the norm of the cordillera; defaults to 1
`minpts`	the minimum points to make up a cluster in OPTICS; defaults to ndim+1
`epsilon`	the epsilon parameter of OPTICS, the neighbourhood that is checked; defaults to 10
`rang`	range of the distances (min distance minus max distance). If NULL (default) the cordillera will be normed to each configuration's maximum distance, so an absolute value of goodness-of-clusteredness.
`verbose`	numeric value hat prints information on the fitting process; >2 is extremely verbose

`scale`	should the configuration be scale adjusted
`normed`	should the cordillera be normed; defaults to TRUE
`stresstype`	which stress to report? Defaults to whatever whim is my default (currently explicitly normed stress)

Value

A list with the components

- stress: the stress
- stress.m: default normalized stress
- copstress: the weighted loss value
- OC: the Optics cordillera value
- parameters: the parameters used for fitting (kappa, lambda)
- fit: the returned object of the fitting procedure
- cordillera: the cordillera object

`cop_powersammon` *PCOPS version of sammon with powers*

Description

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

Usage

```
cop_powersammon(
  dis,
  theta = c(1, 1),
  weightmat = 1 - diag(nrow(dis)),
  init = NULL,
  ndim = 2,
  itmaxi = 10000,
  ...,
  stressweight = 1,
  cordweight = 0.5,
  q = 1,
  minpts = ndim + 1,
  epsilon = 10,
  rang = NULL,
  verbose = 0,
  scale = "sd",
  normed = TRUE,
  stresstype = c("default", "stress1", "rawstress", "normstress", "enormstress",
    "enormstress1")
)
```

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
`itmaxi`	number of iterations. default is 10000.
`...`	additional arguments to be passed to the fitting procedure
`stressweight`	weight to be used for the fit measure; defaults to 1
`cordweight`	weight to be used for the cordillera; defaults to 0.5
`q`	the norm of the cordillera; defaults to 1
`minpts`	the minimum points to make up a cluster in OPTICS; defaults to ndim+1
`epsilon`	the epsilon parameter of OPTICS, the neighbourhood that is checked; defaults to 10
`rang`	range of the distances (min distance minus max distance). If NULL (default) the cordillera will be normed to each configuration's maximum distance, so an absolute value of goodness-of-clusteredness.
`verbose`	numeric value hat prints information on the fitting process; >2 is extremely verbose
`scale`	should the configuration be scale adjusted
`normed`	should the cordillera be normed; defaults to TRUE
`stresstype`	which stress to report? Defaults to explicitly normed stress

Value

A list with the components

- stress: the stress
- stress.m: default normalized stress
- copstress: the weighted loss value
- OC: the Optics cordillera value
- parameters: the parameters used for fitting (kappa, lambda)
- fit: the returned object of the fitting procedure
- cordillera: the cordillera object

cop_powerstress *COPS version of powerstress*

Description

Power stress with free kappa and lambda and rho (the theta argument).

Usage

```
cop_powerstress(
  dis,
  theta = c(1, 1, 1),
  weightmat = 1 - diag(nrow(dis)),
  init = NULL,
  ndim = 2,
  itmaxi = 10000,
  ...,
  stressweight = 1,
  cordweight = 0.5,
  q = 1,
  minpts = ndim + 1,
  epsilon = 10,
  rang = NULL,
  verbose = 0,
  scale = "sd",
  normed = TRUE,
  stresstype = c("default", "stress1", "rawstress", "normstress", "enormstress",
    "enormstress1")
)
```

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
`itmaxi`	number of iterations. default is 10000.
`...`	additional arguments to be passed to the fitting procedure
`stressweight`	weight to be used for the fit measure; defaults to 1
`cordweight`	weight to be used for the cordillera; defaults to 0.5
`q`	the norm of the cordillera; defaults to 1

`minpts`	the minimum points to make up a cluster in OPTICS; defaults to ndim+1
`epsilon`	the epsilon parameter of OPTICS, the neighbourhood that is checked; defaults to 10
`rang`	range of the distances (min distance minus max distance). If NULL (default) the cordillera will be normed to each configuration's maximum distance, so an absolute value of goodness-of-clusteredness.
`verbose`	numeric value hat prints information on the fitting process; >2 is extremely verbose
`scale`	should the configuration be scale adjusted
`normed`	should the cordillera be normed; defaults to TRUE
`stresstype`	which stress to report? Defaults to explicitly normed stress

Value

A list with the components

- stress: the stress
- stress.m: default normalized stress
- copstress: the weighted loss value
- OC: the Optics cordillera value
- parameters: the parameters used for fitting (kappa, lambda)
- fit: the returned object of the fitting procedure
- cordillera: the cordillera object

`cop_rpowerstress` *PCOPS version of restricted powerstress.*

Description

This is a power stress where kappa and lambda are free to vary but restricted to be equal, so the same exponent will be used for distances and dissimilarities. nu (for the weights) is also free.

Usage

```
cop_rpowerstress(
  dis,
  theta = c(1, 1, 1),
  weightmat = 1 - diag(nrow(dis)),
  init = NULL,
  ndim = 2,
  itmaxi = 10000,
  ...,
  stressweight = 1,
  cordweight = 0.5,
  q = 1,
  minpts = ndim + 1,
  epsilon = 10,
  rang = NULL,
  verbose = 0,
  scale = "sd",
  normed = TRUE,
  stresstype = c("default", "stress1", "rawstress", "normstress", "enormstress",
    "enormstress1")
)
```

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
`itmaxi`	number of iterations. default is 10000.
`...`	additional arguments to be passed to the fitting procedure
`stressweight`	weight to be used for the fit measure; defaults to 1
`cordweight`	weight to be used for the cordillera; defaults to 0.5
`q`	the norm of the cordillera; defaults to 1
`minpts`	the minimum points to make up a cluster in OPTICS; defaults to ndim+1
`epsilon`	the epsilon parameter of OPTICS, the neighbourhood that is checked; defaults to 10
`rang`	range of the distances (min distance minus max distance). If NULL (default) the cordillera will be normed to each configuration's maximum distance, so an absolute value of goodness-of-clusteredness.
`verbose`	numeric value hat prints information on the fitting process; >2 is extremely verbose
`scale`	should the configuration be scale adjusted
`normed`	should the cordillera be normed; defaults to TRUE
`stresstype`	which stress to report? Defaults to explicitly normed stress

Value

A list with the components

- stress: the stress1 value (sqrt(stress.m))

- stress.m: default normalized stress
- copstress: the weighted loss value
- OC: the Optics cordillera value
- parameters: the parameters used for fitting (kappa, lambda)
- fit: the returned object of the fitting procedure
- cordillera: the cordillera object

`cop_rstress` *PCOPS version of rstress*

Description

Free parameter is kappa for the fitted distances.

Usage

```
cop_rstress(
  dis,
  theta = c(1, 1, 1),
  weightmat = 1 - diag(nrow(dis)),
  init = NULL,
  ndim = 2,
  itmaxi = 10000,
  ...,
  stressweight = 1,
  cordweight = 0.5,
  q = 1,
  minpts = ndim + 1,
  epsilon = 10,
  rang = NULL,
  verbose = 0,
  scale = "sd",
  normed = TRUE,
  stresstype = c("default", "stress1", "rawstress", "normstress", "enormstress",
    "enormstress1")
)
```

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
`itmaxi`	number of iterations. default is 10000.
`...`	additional arguments to be passed to the fitting procedure
`stressweight`	weight to be used for the fit measure; defaults to 1
`cordweight`	weight to be used for the cordillera; defaults to 0.5
`q`	the norm of the cordillera; defaults to 1
`minpts`	the minimum points to make up a cluster in OPTICS; defaults to ndim+1
`epsilon`	the epsilon parameter of OPTICS, the neighbourhood that is checked; defaults to 10
`rang`	range of the distances (min distance minus max distance). If NULL (default) the cordillera will be normed to each configuration's maximum distance, so an absolute value of goodness-of-clusteredness.
`verbose`	numeric value hat prints information on the fitting process; >2 is extremely verbose
`scale`	should the configuration be scale adjusted
`normed`	should the cordillera be normed; defaults to TRUE
`stresstype`	which stress to report? Defaults to explicitly normed stress

Value

A list with the components

- stress: the stress
- stress.m: default normalized stress
- copstress: the weighted loss value
- OC: the Optics cordillera value
- parameters: the parameters used for fitting (kappa, lambda)
- fit: the returned object of the fitting procedure
- cordillera: the cordillera object

`cop_sammon` *PCOPS 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

```
cop_sammon(
  dis,
  theta = 1,
  ndim = 2,
  init = NULL,
  weightmat = NULL,
  itmaxi = 100,
  ...,
  stressweight = 1,
  cordweight = 0.5,
  q = 1,
  minpts = ndim + 1,
  epsilon = 10,
  rang = NULL,
  verbose = 0,
  scale = "sd",
  normed = TRUE,
  stresstype = "default"
)
```

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`	(optional) a matrix of nonnegative weights
`itmaxi`	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
`cordweight`	weight to be used for the cordillera; defaults to 0.5
`q`	the norm of the corrdillera; defaults to 1
`minpts`	the minimum points to make up a cluster in OPTICS; defaults to ndim+1
`epsilon`	the epsilon parameter of OPTICS, the neighbourhood that is checked; defaults to 10
`rang`	range of the distances (min distance minus max distance). If NULL (default) the cordillera will be normed to each configuration's maximum distance, so an absolute value of goodness-of-clusteredness.
`verbose`	numeric value hat prints information on the fitting process; >2 is extremely verbose
`scale`	should the configuration be scale adjusted
`normed`	should the cordillera be normed; defaults to TRUE
`stresstype`	which stress to report. Only takes smacofs default stress currrently.

Value

A list with the components

- stress: the stress
- stress.m: default normalized stress
- copstress: the weighted loss value
- OC: the Optics cordillera value
- parameters: the parameters used for fitting (kappa, lambda)
- fit: the returned object of the fitting procedure
- cordillera: the cordillera object

`cop_sammon2` *Another COPS versions 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

```
cop_sammon2(
  dis,
  theta = 1,
  ndim = 2,
  weightmat = NULL,
  init = NULL,
  itmaxi = 1000,
  ...,
  stressweight = 1,
  cordweight = 0.5,
  q = 1,
  minpts = ndim + 1,
  epsilon = 10,
  rang = NULL,
  verbose = 0,
  normed = TRUE,
  scale = "sd",
  stresstype = "default"
)
```

Arguments

`dis`	numeric matrix or dist object of a matrix of proximities
`theta`	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 sammon weights)
`init`	(optional) initial configuration
`itmaxi`	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
`cordweight`	weight to be used for the cordillera; defaults to 0.5
`q`	the norm of the corrdillera; defaults to 1
`minpts`	the minimum points to make up a cluster in OPTICS; defaults to ndim+1
`epsilon`	the epsilon parameter of OPTICS, the neighbourhood that is checked; defaults to 10
`rang`	range of the distances (min distance minus max distance). If NULL (default) the cordillera will be normed to each configuration's maximum distance, so an absolute value of goodness-of-clusteredness.
`verbose`	numeric value hat prints information on the fitting process; >2 is extremely verbose
`normed`	should the cordillera be normed; defaults to TRUE
`scale`	should the configuration be scale adjusted
`stresstype`	which stress to report. Only takes smacofs default stress currrently.

Value

A list with the components

- stress: the stress
- stress.m: default normalized stress
- copstress: the weighted loss value
- OC: the Optics cordillera value
- parameters: the parameters used for fitting (kappa, lambda)
- fit: the returned object of the fitting procedure
- cordillera: the cordillera object

cop_smacofSphere *PCOPS 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

```
cop_smacofSphere(
  dis,
  theta = 1,
  ndim = 2,
  weightmat = NULL,
  init = NULL,
  itmaxi = 1000,
  ...,
  stressweight = 1,
  cordweight = 0.5,
  q = 1,
  minpts = ndim + 1,
  epsilon = 10,
  rang = NULL,
  verbose = 0,
  normed = TRUE,
  scale = "sd",
  stresstype = "default"
)
```

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
`itmaxi`	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
`cordweight`	weight to be used for the cordillera; defaults to 0.5
`q`	the norm of the corrdillera; defaults to 1
`minpts`	the minimum points to make up a cluster in OPTICS; defaults to ndim+1

`epsilon`	the epsilon parameter of OPTICS, the neighbourhood that is checked; defaults to 10
`rang`	range of the distances (min distance minus max distance). If NULL (default) the cordillera will be normed to each configuration's maximum distance, so an absolute value of goodness-of-clusteredness.
`verbose`	numeric value hat prints information on the fitting process; >2 is extremely verbose
`normed`	should the cordillera be normed; defaults to TRUE
`scale`	should the configuration be scale adjusted
`stresstype`	which stress to report. Only takes smacofs default stress currrently.

Value

A list with the components

- stress: the stress
- stress.m: default normalized stress
- copstress: the weighted loss value
- OC: the Optics cordillera value
- parameters: the parameters used for fitting (kappa, lambda)
- fit: the returned object of the fitting procedure
- cordillera: the cordillera object

`cop_smacofSym` *PCOPS versions 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

```
cop_smacofSym(
  dis,
  theta = 1,
  ndim = 2,
  weightmat = NULL,
  init = NULL,
  itmaxi = 1000,
  ...,
  stressweight = 1,
  cordweight = 0.5,
  q = 1,
  minpts = ndim + 1,
  epsilon = 10,
  rang = NULL,
  verbose = 0,
  normed = TRUE,
  scale = "sd",
  stresstype = "default"
)
```

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
`itmaxi`	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
`cordweight`	weight to be used for the cordillera; defaults to 0.5
`q`	the norm of the cordillera; defaults to 1
`minpts`	the minimum points to make up a cluster in OPTICS; defaults to ndim+1
`epsilon`	the epsilon parameter of OPTICS, the neighbourhood that is checked; defaults to 10
`rang`	range of the distances (min distance minus max distance). If NULL (default) the cordillera will be normed to each configuration's maximum distance, so an absolute value of goodness-of-clusteredness.
`verbose`	numeric value hat prints information on the fitting process; >2 is extremely verbose
`normed`	should the cordillera be normed; defaults to TRUE
`scale`	should the configuration be scale adjusted
`stresstype`	which stress to report. Only takes smacofs default stress currrently.

Value

A list with the components

- stress: the stress
- stress.m: default normalized stress
- copstress: the weighted loss value
- OC: the Optics cordillera value
- parameters: the parameters used for fitting (kappa, lambda)

- fit: the returned object of the fitting procedure (which has all smacofB elements and some more)
- cordillera: the cordillera object

`cop_sstress` *PCOPS 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

```
cop_sstress(
  dis,
  theta = c(2, 1, 1),
  weightmat = 1 - diag(nrow(dis)),
  init = NULL,
  ndim = 2,
  itmaxi = 10000,
  ...,
  stressweight = 1,
  cordweight = 0.5,
  q = 1,
  minpts = ndim + 1,
  epsilon = 10,
  rang = NULL,
  verbose = 0,
  scale = "sd",
  normed = TRUE,
  stresstype = c("default", "stress1", "rawstress", "normstress", "enormstress",
    "enormstress1")
)
```

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`	number of dimensions of the target space

`itmaxi`	number of iterations. default is 10000.
`...`	additional arguments to be passed to the fitting procedure
`stressweight`	weight to be used for the fit measure; defaults to 1
`cordweight`	weight to be used for the cordillera; defaults to 0.5
`q`	the norm of the cordillera; defaults to 1
`minpts`	the minimum points to make up a cluster in OPTICS; defaults to ndim+1
`epsilon`	the epsilon parameter of OPTICS, the neighbourhood that is checked; defaults to 10
`rang`	range of the distances (min distance minus max distance). If NULL (default) the cordillera will be normed to each configuration's maximum distance, so an absolute value of goodness-of-clusteredness.
`verbose`	numeric value hat prints information on the fitting process; >2 is extremely verbose
`scale`	should the configuration be scale adjusted
`normed`	should the cordillera be normed; defaults to TRUE
`stresstype`	which stress to report? Defaults to explicitly normed stress

Value

A list with the components

- stress: the stress
- stress.m: default normalized stress
- copstress: the weighted loss value
- OC: the Optics cordillera value
- parameters: the parameters used for fitting (kappa, lambda)
- fit: the returned object of the fitting procedure
- cordillera: the cordillera object

`doubleCenter` *Double centering of a matrix*

Description

Double centering of a matrix

Usage

```
doubleCenter(x)
```

Arguments

`x`	numeric matrix

Value

the double centered matrix

`enorm` *Explicit Normalization Normalizes distances*

Description

Explicit Normalization Normalizes distances

Usage

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

Arguments

`x`	numeric matrix
`w`	weight

Value

a constant

`ljoptim` *(Adaptive) Version of Luus-Jakola 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 (see also `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
```

`mkBmat` *Auxfunction1*

Description

only used internally

Usage

```
mkBmat(x)
```

Arguments

`x`	matrix

Value

a matrix

`mkPower` *Take matrix to a power*

Description

Take matrix to a power

Usage

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

Arguments

`x`	matrix
`r`	numeric (power)

Value

a matrix

pcops *Profile COPS Function (aka COPS Variant 2)*

Description

Metaparameter selection for MDS models baseed on the Profile COPS approach (COPS Variant 2). It uses copstress for hyperparameter selection. It is a special case of a STOPS model.

Usage

```
pcops(
  dis,
  loss = c("stress", "smacofSym", "smacofSphere", "strain", "sammon", "rstress",
   "powermds", "sstress", "elastic", "powersammon", "powerelastic", "powerstress",
    "sammon2", "powerstrain", "apstress", "rpowerstress"),
  weightmat = NULL,
  ndim = 2,
  init = NULL,
  theta = 1,
  stressweight = 1,
  cordweight,
  q = 2,
  minpts = ndim + 1,
  epsilon = 100,
  rang,
 optimmethod = c("ALJ", "pso", "SANN", "DIRECT", "DIRECTL", "stogo", "MADS", "hjk"),
  lower = 0.5,
  upper = 5,
  verbose = 0,
  scale = c("proc", "sd", "none", "std"),
  normed = TRUE,
  s = 4,
  stresstype = "default",
  acc = 1e-07,
  itmaxo = 200,
  itmaxi = 10000,
  ...
)
```

Arguments

`dis`	numeric matrix or dist object of a matrix of proximities
`loss`	which loss function to be used for fitting, defaults to strain. Currently allows for the following models:

- Power transformations of observed proximities only (theta must be scalar): Strain loss or classical scaling (`strain`, workhorse is cmdscale), Kruskall's stress for symmetric matrices (`smacofSym` or `stress` and `smacofSphere` for scaling onto a sphere; workhorse is smacof), Sammon mapping (`sammon` or `sammon2`; for the earlier the workhorse is sammon from MASS for the latter it is smacof), elastic scaling (`elastic`, the workhorse is smacof), Takane et al's s-Stress `sstress` (workhorse is powerstressMin)

- Power transformations of fitted distances only (theta must be scalar): De Leeuw's r-stress `rstress` (workhorse is powerstressMin)
- Power transformations of fitted distances and observed proximities (theta must be scalar or of length 2): Power MDS (`powermds`), Sammon mapping/elastic scaling with powers (`powersammon`, `powerelastic`)
- Power transfomations of fitted distances, observed proximities and weights (theta must be of length 3 at most): powerstress (POST-MDS, `powerstress`), restricted powerstress with equal transformations for distances and proximities (`rpowerstress`); workhorse is powerstressMin
- Approximation to power stress (theta must be of length 2): Approximated power stress (`apstress`; workhorse is smacof)

`weightmat`	(optional) a matrix of nonnegative weights; defaults to 1 for all off diagonals
`ndim`	number of dimensions of the target space
`init`	(optional) initial configuration. If not supplied, the Torgerson scaling result of the dissimilarity matrix dis^theta[2]/enorm(dis^theta[2],weightmat) is used.
`theta`	the theta vector of powers; see the corresponding cop_XXX function for which theta are allowed. If a scalar is given as argument, it will be recycled. Defaults to 1.
`stressweight`	weight to be used for the fit measure; defaults to 1
`cordweight`	weight to be used for the cordillera; if missing gets estimated from the initial configuration so that copstress = 0 for theta=c(1,1)
`q`	the norm of the cordillera; defaults to 1
`minpts`	the minimum points to make up a cluster in OPTICS; defaults to ndim+1
`epsilon`	the epsilon parameter of OPTICS, the neighbourhood that is checked; defaults to 10
`rang`	range of the minimum reachabilities to be considered. If missing it is found from the initial configuration by taking 1.5 times the maximal minimum reachability of the model with theta=c(1,1). If NULL it will be normed to each configuration's minimum and maximum distance, so an absolute value of goodness-of-clusteredness. Note that the latter is not necessarily desirable when comparing configurations for their relative clusteredness. See also `cordillera`.
`optimmethod`	What general purpose optimizer to use? Defaults to our adaptive LJ version (ALJ). Also allows particle swarm optimization with s particles ("pso") and simulated annealing ("SANN"), "DIRECT" and "DIRECTL", Hooke-Jeeves ("hjk"), StoGo ("stogo"), and "MADS". We recommend not using SANN and pso with the rstress, sstress and the power stress models. We made good experiences with ALJ, stogo, DIRECT and DIRECTL and also MADS.
`lower`	A vector of the lower box contraints of the search region. Its length must match the length of theta.

`upper`	A vector of the upper box contraints of the search region. Its length must match the length of theta.
`verbose`	numeric value hat prints information on the fitting process; >2 is extremely verbose. Note that for models with some parameters fixed, the iteration progress of the optimizer shows different values also for the fixed parameters because due to the modular setup we always optimize over a three parameter vector. These values are inconsequential however as internally they will be fixed.
`scale`	should the configuration be scaled and/or centered for calculating the cordillera? "std" standardizes each column of the configurations to mean=0 and sd=1 (typically not a good idea), "sd" scales the configuration by the maximum standard devation of any column (default), "proc" adjusts the fitted configuration to the init configuration (or the Togerson scaling solution if init=NULL). This parameter only has an effect for calculating the cordillera, the fitted and returned configuration is NOT scaled.
`normed`	should the cordillera be normed; defaults to TRUE
`s`	number of particles if pso is used
`stresstype`	what stress to be used for comparisons between solutions. Currently not implemented and pcops uses explicitly normalized stress for copstress (not stress-1). Stress-1 is reported by the print function though.
`acc`	termination threshold difference of two successive outer minimization steps.
`itmaxo`	iterations of the outer step (optimization over the hyperparmeters; if solver allows it). Defaults to 200.
`itmaxi`	iterations of the inner step (optimization of the MDS). Defaults to 10000 (whichis huge).
`...`	additional arguments to be passed to the optimization procedure

Value

A list with the components

- copstress: the weighted loss value
- OC: the OPTICS cordillera for the scaled configuration (as defined by scale)
- optim: the object returned from the optimization procedure
- stress: the stress (square root of stress.m)
- stress.m: default normalized stress
- parameters: the parameters used for fitting (kappa, lambda)
- fit: the returned object of the fitting procedure
- cordillera: the cordillera object

Examples

```
dis<-as.matrix(smacof::kinshipdelta)
set.seed(210485)
#configuration is scaled with highest column sd for calculating cordilera
res1<-pcops(dis,loss="strain",lower=0.1,upper=5,minpts=2)
```

```
res1
summary(res1)
plot(res1)
```

`pdist` *Squared p-distances*

Description

Squared p-distances

Usage

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

Arguments

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

Value

squared Minkowski distance matrix

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

Description

S3 plot method for cops objects

Usage

```
## S3 method for class 'cops'
plot(x, plot.type = c("confplot"), main, asp = 1, ...)
```

Arguments

`x`	an object of class cops
`plot.type`	String indicating which type of plot to be produced: "confplot", "reachplot", "resplot","transplot", "Shepard", "stressplot" (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.
- Reachability plot (plot.type = "confplot"): Plots the OPTICS reachability plot and the OPTICS cordillera
- 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.
- Transformation Plot (plot.type = "transplot"): Diagram with the observed dissimilarities (lighter) and the transformed observed dissimilarities (darker) against the fitted distances together with loess smoothing lines
- 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.

Examples

```
dis<-as.matrix(smacof::kinshipdelta)
resl<-copstressMin(dis,itmax=20)
plot(resl)
```

`plot.pcops` *S3 plot method for p-cops objects*

Description

S3 plot method for p-cops objects

Usage

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

Arguments

`x`	an object of class cops
`plot.type`	String indicating which type of plot to be produced: "confplot", "reachplot", "resplot","transplot", "Shepard", "stressplot" (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.
- Reachability plot (plot.type = "confplot"): Plots the OPTICS reachability plot and the OPTICS cordillera
- 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.
- Transformation Plot (plot.type = "transplot"): Diagram with the observed dissimilarities (lighter) and the transformed observed dissimilarities (darker) against the fitted distances together with loess smoothing lines
- 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.

Examples

```
dis<-as.matrix(smacof::kinshipdelta)
resl<-pcops(dis,loss="strain",lower=0.1,upper=5,minpts=2)
plot(resl)
plot(resl,plot.type="Shepard")
```

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

Examples

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

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

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

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 cmdscale
`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.

`powerStressFast` *Power stress minimization by NEWUOA*

Description

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

Usage

```
powerStressFast(
  delta,
  kappa = 1,
  lambda = 1,
  nu = 1,
  weightmat = 1 - diag(nrow(delta)),
  init = NULL,
  ndim = 2,
  acc = 1e-12,
  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`	The smallest value of the trust region radius that is allowed. If not defined, then 1e-10 will be used.
`itmax`	maximum number of iterations. Default is 50000.
`verbose`	should iteration output be printed; if > 1 then yes

Value

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

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

and some additional components

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

See Also

`smacofSym`

Examples

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

`powerStressMin` *Power Stress SMACOF*

Description

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

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. Default is 50000.
`verbose`	should iteration output be printed; if > 1 then yes

Value

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

- delta: Observed dissimilarities, not normalized
- obsdiss: Observed dissimilarities, normalized
- confdist: Configuration dissimilarities, NOT normalized
- conf: Matrix of fitted configuration, NOT normalized

- stress: Default stress (stress 1; 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

and some additional components

- stress.m: default stress for the COPS and STOP defaults to the explicitly normalized stress on the normalized, transformed dissimilarities
- stress.en: a manually calculated stress on the normalized, transformed dissimilarities and normalized transformed distances which is not correct
- deltaorig: observed, untransformed dissimilarities
- weightmat: weighting matrix

See Also

`smacofSym`

Examples

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

`procruster` *procruster: a procrustes function*

Description

procruster: a procrustes function

Usage

```
procruster(x)
```

Arguments

`x`	numeric matrix

Value

a matrix

`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 stops package.

Value

Object of class sammonE' inheriting from `sammon`. This wrapper only adds an extra slot to the list with the call, adds column labels to the $points and assigns S3 classes 'sammonE', 'sammon' and 'cmdscale'. It also adds a slot obsdiss with normalized dissimilarities.

Examples

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

`scale_adjust` *Adjusts a configuration*

Description

Adjusts a configuration

Usage

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

Arguments

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

Value

The scale adjusted configuration.

`secularEq` *Secular Equation*

Description

Secular Equation

Usage

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

Arguments

`a`	matrix
`b`	matrix

Value

a matrix

`sqdist` *Squared distances*

Description

Squared distances

Usage

```
sqdist(x)
```

Arguments

`x`	numeric matrix

Value

squared distance matrix

`torgerson` *Torgerson scaling*

Description

Torgerson scaling

Usage

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

Arguments

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

Value

a n x p matrix (the configuration)

Examples

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

Index