Package: cops (via r-universe)

September 23, 2024

Title Cluster Optimized Proximity Scaling

Version 1.12-1

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 explicit power functions and penalizing 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) with given explicit power transformations and implicit ratio, interval and non-metric optimal scaling transformations (Borg & Groenen, 2005, ISBN:978-0-387-28981-6), and one for using the augmented fitting criterion to find optimal hyperparameters for the explicit transformations (P-COPS). The package contains various functions, wrappers, methods and classes for fitting, plotting and displaying a large number of different MDS models (most of the functionality in smacofx) in the COPS framework. 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>) and a functions to calculate the phi-distances for count data or histograms.

Depends R (>= 3.5.0), cordillera (>= 0.7-2), smacofx

Imports smacof, analogue, cmaes, crs, dfoptim, GenSA, minqa, NlcOptim, nloptr, pso, rgenoud, Rsolnp, subplex

Suggests R.rsp, rmarkdown

VignetteBuilder R.rsp

License GPL-2 | GPL-3

LazyData true

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

RoxygenNote 7.3.1

Encoding UTF-8

NeedsCompilation no

Author Thomas Rusch [aut, cre]
(<https://orcid.org/0000-0002-7773-2096>), Patrick Mair [aut]
(<https://orcid.org/0000-0003-0100-6511>), Kurt Hornik [ctb]
(<https://orcid.org/0000-0003-4198-9911>)

Repository CRAN

Date/Publication 2024-09-22 16:50:02 UTC

Contents

BankingCrisesDistances . . . 3
biplotmds.pcops . . . 3
biplotmds.stops . . . 4
bootmds.pcops . . . 5
cops . . . 6
copstress . . . 7
copstressMin . . . 9
cop_apstress . . . 14
cop_cmdscale . . . 16
cop_elastic . . . 17
cop_powerelastic . . . 19
cop_powermds . . . 21
cop_powersammon . . . 22
cop_powerstress . . . 24
cop_rpowerstress . . . 25
cop_rstress . . . 27
cop_sammon . . . 29
cop_sammon2 . . . 30
cop_smacofSphere . . . 32
cop_smacofSym . . . 33
cop_sstress . . . 35
doubleCenter . . . 36
enorm . . . 37
jackmds.pcops . . . 37
ljoptim . . . 38
matchphi . . . 40
mkBmat . . . 40
mkPower . . . 41
pcops . . . 41
pdist . . . 44
phidistance . . . 45
plot.copsc . . . 45
plot.pcops . . . 47
procruster . . . 48

scale_adjust 48
secularEq 49
spp 49
sqdist 50

Index **51**

`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)

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

Description

S3 method for pcops objects

Usage

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

Arguments

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

Details

If a model for individual differences is provided, the external variables are regressed on the group stimulus space configurations. For objects returned from 'biplotmds' we use the plot method in biplotmds. In the biplot called with plot() only the relative length of the vectors and their direction matters. Using the vecscale argument in plot() the user can control for the relative length of the vectors. If 'vecscale = NULL', the 'vecscale()' function from the 'candisc' package is used which tries to automatically calculate the scale factor so that the vectors approximately fill the same space as the configuration. In this method vecscale should usually be smaller than the one used in smacof by a factor of 0.1.

Value

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

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

Description

S3 method for stops objects

Usage

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

Arguments

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

Details

If a model for individual differences is provided, the external variables are regressed on the group stimulus space configurations. For objects returned from 'biplotmds' we use the plot method in biplotmds. In the biplot called with plot() only the relative length of the vectors and their direction matters. Using the vecscale argument in plot() the user can control for the relative length of the vectors. If 'vecscale = NULL', the 'vecscale()' function from the 'candisc' package is used which tries to automatically calculate the scale factor so that the vectors approximately fill the same space as the configuration. In this method vecscale should usually be smaller than the one used in smacof by a factor of 0.1.

Value

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

Examples

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

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

Description

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

Usage

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

Arguments

`object`	Object of class smacofP if used as method or another object inheriting from smacofB (needs to be called directly as bootmds.smacofP then).
`data`	Initial data (before dissimilarity computation).
`method.dat`	Dissimilarity computation used as MDS input. This must be one of "pearson", "spearman", "kendall", "euclidean", "maximum", "manhattan", "canberra", "binary".
`nrep`	Number of bootstrap replications.
`alpha`	Alpha level for condfidence ellipsoids.
`verbose`	If 'TRUE', bootstrap index is printed out.
`...`	Additional arguments needed for dissimilarity computation as specified in `sim2diss`.

Details

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

Value

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

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

Examples

```
dats <- na.omit(PVQ40[,1:5])
diss <- dist(t(dats))   ## Euclidean distances
fit <- pcops(diss,loss="rstress",itmaxi=50) #this is just for illustration: increase itmaxi
set.seed(123)
resboot <- bootmds(fit, dats, method.dat = "euclidean", nrep = 2)
```

`cops` *High Level COPS Function*

Description

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).

Value

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

Examples

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

#COPS-C with equal weight to stress and cordillera
res1<-cops(dis,variant="COPS-C",stressweight=0.75,cordweight=0.25,
          minpts=2,itmax=1000) #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 ratio, interval or ordinal PS configuration with given explicit power transformations theta. The function allows mix-and-match of explicit (via theta) and implicit (via type) transformations by setting the kappa, lambda, nu (or theta) and type arguments.

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 = max(10, max(delta)),
  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", "proc", "none"),
  normed = TRUE,
  accuracy = 1e-07,
  itmax = 10000,
  stresstype = c("stress-1", "stress"),
  principal = FALSE,
  ...
)

copsc(
  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 = max(10, max(delta)),
  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", "proc", "none"),
  normed = TRUE,
  accuracy = 1e-07,
  itmax = 10000,
  stresstype = c("stress-1", "stress"),
  principal = FALSE,
  ...
)

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 = max(10, max(delta)),
  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", "proc", "none"),
  normed = TRUE,
  accuracy = 1e-07,
  itmax = 10000,
  stresstype = c("stress-1", "stress"),
  principal = FALSE,
  ...
)
```

Arguments

`delta`	numeric matrix or dist object of a matrix of proximities
`kappa`	power transformation for fitted distances
`lambda`	power transformation for proximities (only used if type="ratio" or "interval")
`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 and type="ratio" or "interval"), 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, lambda, 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, see `optics` where it is called `minPts`; defaults to ndim+1.
`epsilon`	the epsilon parameter of OPTICS, the neighbourhood that is checked, see `optics`; defaults to 10 (which is plenty for the explicit normalization we use). Note this means we do not expect any noise objects per default. This number will rarely be exceeded if we standardize the configuration as is the default in cops. However if no standardization is applied or there is a procrustes adjustment to a configuration with variance of 10 or more on any of the axes, it can have the effect of being too small. In that case just set a much higher epsilon.
`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' (`newuoa`), 'Nelder-Mead' (see `optim`), 'hjk' (Hooke-Jeeves algorithm from `hjk`), 'solnl' (from `solnl`), 'solnp' (from `solnp`), 'subplex' (from `subplex`), 'SANN' (simulated annealing, `optim`), 'BFGS' (see `optim`), 'snomadr' (from `snomadr`), 'genoud' (from `genoud`), 'gensa' (from `GenSA`), 'cmaes' (from `cma_es`) and 'direct' (from `direct`). See the linked functions 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 everything with 'hjk', 'BFGS', 'Newuoa', 'subplex' and 'solnl' in it 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`	Scale the configuration (in MDS stress is invariant up to a scaling factor). One of "none" (so no extra scaling of the configuration but normalized to sum delta^2=1), "sd" (configuration divided by the highest standard deviation of any the columns), "proc" (procrustes adjustment to the initial fit) and "rmsq" (configuration divided by the maximum root mean square of the columns). Default is "sd" which often gives a nicer spread on the axes. Note that the scaled configuration is returned as $conf and the unscaled as $usconf, so manual calculation of the OC should be done with $conf.
`normed`	should the Cordillera be normed; defaults to TRUE.
`accuracy`	numerical accuracy, defaults to 1e-7.
`itmax`	maximum number of iterations. Defaults to 10000. For the two-step algorithms if itmax is exceeded by the first solver, the second algorithm is run for at least 0.1*itmax (so overall itmax may be exceeded by a factor of 1.1).
`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 is used. Using stress-1 puts more weight on MDS fit.
`principal`	If 'TRUE', principal axis transformation is applied to the final configuration.
`...`	additional arguments to be passed to the optimization procedure

Details

This is an extremely flexible approach to least squares proximity scaling: It supports ratio power stress; ratio, interval and ordinal r stress and ratio, interval and ordinal MDS with or without a COPS penalty. Famous special cases of these models that can be fitted are multiscale MDS if kappa->0 and delta=log(delta), Alscal MDS (sstress) with lambda=kappa=2, sammon type mapping with weightmat=delta and nu=-1, elastic scaling with weightmat=delta and nu=-2. Due to mix-and-match this function also allows to fit models that have not yet been published, such as for example an "elastic scaling ordinal s-stress with cops penalty".

If one wants to fit these models without the cops penalty, we recommend to use `powerStressMin` (for ratio and interval MDS with any power transformation for weights, dissimilarities and distances) or `rStressMin` (for ratio, interval and ordinal MDS with power transformations for distances and weights) as these use majorization.

Some optimizers (including the default hjk-Newuoa) will print a warning if itmax is (too) small or if there was no convergence. Consider increasing itmax then.

For some solvers theresometimes may be an error [NA/NaN/Inf in foreign function call (arg 3)] stemming from smacof::transform(). This happens when the algorithm places two object at exactly the same place so their fitted distance is 0. This is good from an OPTICS Cordillera point of view (as it is more clustered) which is why some solvers like to pick that up, but it can lead to an issue in the optimal scaling in smacof. This can usually be mitigated when specifying the model by either using less cordweight, less itmax, less accuracy or combining the two offending objects into one (so include them as a combined row in the distance matrix).

We might eventually switch to newuoa in nloptr.

Value

A copsc object (inheriting from smacofP). A list with the components

- delta: the original untransformed dissimilarities
- tdelta: the explicitly transformed dissimilarities
- dhat: the explicitly transformed dissimilarities (dhats), optimally scaled and normalized (which are approximated by the fit)
- confdist: Configuration distances, the transformed fitted distances
- conf: the configuration (normed) and scaled as specified in scale.
- usconf: the unscaled configuration (normed to sum delta^2=1). Scaling applied to usconf gives conf.
- parameters, 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
- 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
- copstress: the copstress loss value
- resmat: the matrix of residuals
- weightmat: the matrix of untransformed weights
- tweightmat: the transformed weighting matrix (here weightmat^nu)
- 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)

set.seed(1)
## Copstress with equal weight to stress and cordillera
res1<-copstressMin(dis,stressweight=0.5,cordweight=0.5,
                  itmax=100) #use higher itmax about 10000
res1
summary(res1)
plot(res1)  #super clustered

##Alias name
res1<-copsc(dis,stressweight=0.5,
                  cordweight=0.5,itmax=100)


## Elastic scaling ordinal s-stress with cops penalty
res1<-copsc(dis,type="ordinal",kappa=2,nu=-2,weightmat=dis,
            stressweight=0.5, cordweight=0.5,itmax=100)
```

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

Description

This uses an approximation to power stress that makes use of smacofx as workhorse. Free parameters are kappa, lambda and nu

Usage

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

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 three, with the first the kappa argument, the second the lambda argument and the third the nu argument. One cannot supply upsilon and tau as of yet. Defaults to 1 1 1.
`type`	MDS type.
`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

Value

A list with the components

- stress: the stress-1 of the configuration
- stress.m: default normalized stress (sqrt(stress-1))
- copstress: the weighted loss value
- OC: the OPTICS cordillera value
- parameters: the theta parameters used for fitting (kappa, lambda, nu)
- fit: the returned object of the fitting procedure (typically of class smacofB or smacofP)
- cordillera: the cordillera object

cop_cmdscale *PCOPS version of strain*

Description

The free parameter that pcops optimizes over is lambda for power transformations of the observed proximities.

Usage

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

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.
`type`	MDS type. Ignored here.
`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.
`add`	should the dissimilarities be made Euclidean? Defaults to TRUE.
`...`	additional arguments to be passed to the fitting procedure smacofx::cmdscale. Note we always use eig=TRUE and that can't be changed (we need the GOF). Also default if nothing is supplied is to use add=TRUE which in my opinion one always should to avoid negative eigenvalues.

`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

Value

A list with the components

- stress: the badness-of-fit value (this isn't stress here but 1-(sum_ndim(max(eigenvalues,0))/sum_n(max(eigenvalues,0)), 1-GOF[2])
- stress.m: default normalized stress (manually calculated)
- copstress: the weighted loss value
- OC: the Optics cordillera value
- parameters: the parameters used for fitting (lambda)
- fit: the returned object of the fitting procedure, which is cmdscalex object with some extra slots for the parameters and stresses
- 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,
  type = "ratio",
  ndim = 2,
  weightmat = 1 - diag(nrow(dis)),
  init = NULL,
  itmaxi = 1000,
  ...,
  stressweight = 1,
  cordweight = 0.5,
  q = 1,
  minpts = ndim + 1,
  epsilon = 10,
  rang = NULL,
  verbose = 0,
  normed = TRUE,
  scale = "sd"
)
```

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.
`type`	MDS type.
`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

Value

A list with the components

- stress: the stress-1
- stress.m: default normalized stress
- copstress: the weighted loss value
- OC: the Optics cordillera value
- parameters: the parameters used for fitting (lambda)
- fit: the returned object of the fitting procedure (plus a slot for the orginal data $deltaorig)
- 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),
  type = "ratio",
  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
)
```

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.
`type`	MDS type. Defaults to "ratio".
`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

Value

A list with the components

- stress: the stress-1 value (sqrt(stress.m))
- stress.m: default normalized stress
- copstress: the weighted loss value
- OC: the OPTICScordillera 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 internally fixed to 1, so no weight transformation.

Usage

```
cop_powermds(
  dis,
  theta = c(1, 1),
  type = "ratio",
  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
)
```

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 1,1.
`type`	MDS type. Defaults to ratio.
`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

Value

A list with the components

- stress: the stress-1 value
- 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),
  type = "ratio",
  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
)
```

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.
`type`	MDS type. defaults to ratio.
`weightmat`	(optional) a matrix of nonnegative weights
`init`	(optional) initial configuration
`ndim`	number of dimensions of the target space
`itmaxi`	number of iterations (of powerstress). 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

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 explicit 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) and ratio and interval optimal scaling.

Usage

```
cop_powerstress(
  dis,
  theta = c(1, 1, 1),
  type = "ratio",
  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
)
```

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.
`type`	MDS type. Default is ratio.
`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

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, nu)
- fit: the returned object of the fitting procedure plus a slot for the original data $deltaorig
- 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),
  type = "ratio",
  weightmat = NULL,
  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
)
```

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 based on theta[1]. 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.
`type`	MDS type. Defaults to "ratio".
`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

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 explicit parameters used for fitting (kappa=lambda, nu)
- fit: the returned object of the fitting procedure
- cordillera: the cordillera object

`cop_rstress` *PCOPS version of rstress*

Description

Free parameter is kappa=2r for the fitted distances.

Usage

```
cop_rstress(
  dis,
  theta = 1,
  type = "ratio",
  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
)
```

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=2*r transformation for the fitted distances proximities. Defaults to 1. Note that what is returned is r, not kappa.
`type`	MDS type. Defaults to "ratio".
`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

Value

A list with the components

- stress: the stress-1 value
- stress.m: default normalized stress
- copstress: the weighted loss value
- OC: the Optics cordillera value
- parameters: the parameters used for fitting (r)
- fit: the returned object of the fitting procedure
- cordillera: the cordillera object

cop_sammon *PCOPS version of Sammon mapping from MASS*

Description

Uses smacofx::sammon wrapper for MASS::sammon. The free parameter is lambda for power transformations of the observed proximities.

Usage

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

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.
`type`	MDS type. Only "ratio" here.
`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`	if TRUE the configuration is scale adjusted
`normed`	should the cordillera be normed; defaults to TRUE

Value

A list with the components

- stress: the stress-1
- stress.m: default normalized stress (stress-1^2)
- copstress: the weighted loss value
- OC: the Optics cordillera value
- parameters: the parameters used for fitting (lambda)
- fit: the returned object of the fitting procedure smacofx::sammon
- cordillera: the cordillera object

`cop_sammon2` *Another COPS versions of Sammon mapping models (via smacofSym)*

Description

Uses smacofSym, so it can deal with a weightmatrix and different types. The free parameter is lambda for power transformations of the observed proximities.

Usage

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

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.
`type`	MDS type. Default is ratio.
`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

Value

A list with the components

- stress: the stress-1 value
- stress.m: default normalized stress
- copstress: the weighted loss value
- OC: the Optics cordillera value
- parameters: the parameters used for fitting (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,
  type = "ratio",
  ndim = 2,
  weightmat = NULL,
  init = NULL,
  itmaxi = 5000,
  ...,
  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.
`type`	MDS type
`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-1 value
- stress.m: default normalized stress
- copstress: the weighted loss value
- OC: the Optics cordillera value
- parameters: the parameters used for fitting (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,
  type = "ratio",
  ndim = 2,
  weightmat = 1 - diag(nrow(dis)),
  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; should be a scalar for the lambda (proximity) transformation. Defaults to 1.
`type`	MDS type.
`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 (lambda)
- fit: the returned object of the fitting procedure
- 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 = 1,
  type = "ratio",
  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
)
```

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)).
`type`	MDS type, defaults to "ratio".

`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

Value

A list with the components

- stress: the stress-1 value
- stress.m: default normalized stress
- copstress: the weighted loss value
- OC: the Optics cordillera value
- parameters: the parameters used for fitting (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

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

Description

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

Usage

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

Arguments

`object`	Object of class pcops.
`eps`	Convergence criterion
`itmax`	Maximum number of iterations
`verbose`	If ’TRUE’, intermediate stress is printed out.

Details

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

Note that this jackknife only resamples the configuration given the selected hyperparameters, so uncertainty with respect to the hyperparameetr selection is not incorporated.

Value

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

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

Examples

```
diso<-kinshipdelta
fit <- pcops(diso,loss="stress")
res.jk <- jackmds(fit)
plot(res.jk)
```

`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
```

`matchphi` *Distances of MATCH-ADTC modules*

Description

A symmetric distance matrix of 32 MATCH-ADT modules based on their usage by clinicians. The raw data were counts of how often each module has been used with each of 449 youths, resulting in a count profile of each module. Based on that the phi-distance between the modules has been calculated.

Format

A 32 x 32 distance matrix.

Details

This is an object that inherits from class distance (see package analogue) and matrix.

`mkBmat` *Auxfunction1*

Description

only used internally

Usage

```
mkBmat(x)
```

Arguments

x	matrix

mkPower *Take matrix to a power*

Description

Take matrix to a power

Usage

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

Arguments

x	matrix
r	numeric (power)

Value

a matrix

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 of explicit transformations (currently power transformations). It is a special case of a STOPS model and predated it; stops has more functionality and can be seen as the successor. pcops uses explicitly normalized stress for copstress (not stress-1).

Usage

```
pcops(
  dis,
  loss = c("stress", "smacofSym", "smacofSphere", "strain", "sammon", "rstress",
   "powermds", "sstress", "elastic", "powersammon", "powerelastic", "powerstress",
    "sammon2", "powerstrain", "apstress", "rpowerstress"),
  type = "ratio",
  weightmat = NULL,
  ndim = 2,
  init = NULL,
  theta = c(1, 1, 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,
  acc = 1e-05,
  itmaxo = 200,
  itmaxi = 5000,
  ...
)
```

Arguments

`dis`	numeric matrix or dist object of a matrix of proximities
`loss`	which loss function to be used for fitting, defaults to strain. See Details.
`type`	MDS type which may be one of "ratio", interval", "ordinal". Defaults to "ratio". Note not all loss arguments support all types; if not there will be an error and infor which types are supported. In that case choose another type.
`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 free parameters; see details for the number of free parameters for each loss function. Defaults to 1 for all free parameters. Make sure to supply a theta of the correct length as the mechanisms in place to automatically choose theta/upper/lower are dependent on the optimizer and ad hoc: If this is a vector with more elements than necessary, it is either cut (so for a vector of length 3 and a function with 2 free parameters, the first two elements of the vector are used) or there will be an error. If a scalar is given as argument and the number of free parameters is larger than 1, the scalar will be recycled and this may also make the optimizers equate all free parameters.
`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=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=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", `psoptim`) and simulated annealing ("SANN", `optim`), "directT" or "directL" (see `direct`), Hooke-Jeeves ("hjk", `hjk`), StoGo ("stogo", `stogo`), and "snomadr" (`snomadr`). 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 snomadr.
`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
`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 5000.
`...`	additional arguments to be passed to the optimization procedure

Details

Currently allows for the following models:

- Power transformations applied to observed proximities only (theta, upper, lower should be numeric scalar): Strain loss/Torgerson scaling (`strain`, workhorse: smacofx::cmdscale), Stress for symmetric matrices (`smacofSym`, `stress`,`smacofSphere` for scaling onto a sphere; workhorse: smacof::smacofSym), Sammon mapping (`sammon`, workhorse is smacofx::sammon or `sammon2`, workhorse: smacof::smacofSym), elastic scaling (`elastic`, workhorse smacof::smacofSym), Alscal or S-Stress `sstress` (workhorse: smacofx::powerStressMin)

- Power transformations of fitted distances only (theta, upper, lower should be numeric scalar): r-stress `rstress` (workhorse: smacofx:rStressMin)
- Power transformations applied to fitted distances and observed proximities (theta, upper, lower should be numeric of length 2): Power MDS (`powermds`, workhorse: smacofx::powerStressMin), Sammon Mapping/elastic scaling with powers (`powersammon`, `powerelastic`, workhorse: smacofx::powerStressMin)
- Power transformations applied to fitted distances, observed proximities and weights (theta, upper, lower should be numeric of length 3): power stress (POST-MDS, `powerstress`, workhorse: smacofx::powerStressMin), restricted power stress with equal transformations for distances and proximities (`rpowerstress`); workhorse: smacofx::powerStressMin), approximated power stress (`apstress`; workhorse: smacof::smacofSym)

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

`phidistance` *Calculating the pairwise phi distance matrix between n vectors*

Description

The pairwise phi-distance of two vectors x and y is sqrt(sum(((x[i]-y[i])^2)/((x[i]+y[i])*(sum(x)+sum(y))))). The function calculates this for all pairs of rows of a matrix or data frame X.

Usage

```
phidistance(X)
```

Arguments

`X`	an n times p numeric matrix or data frame

Value

a symmetric n times n matrix of pairwise phi distance (between rows of X) with 0 in the main diagonal. Is an object of class distance and matrix.

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

Description

S3 plot method for cops objects

Usage

```
## S3 method for class 'copsc'
plot(x, plot.type, 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", "bubblepot", "histogram" (see details)
`main`	the main title of the plot
`asp`	aspect ratio of x/y axis; defaults to 1; setting to 1 will lead to an accurate representation of the fitted distances.
`...`	Further plot arguments passed: see 'plot.smacofP' 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.

#@importFrom smacofx plot

Examples

```
dis<-as.matrix(smacof::kinshipdelta)
set.seed(1)
resl<-copstressMin(dis,itmax=500)
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, main, asp = 1, ...)
```

Arguments

x	an object of class cops
plot.type	String indicating which type of plot to be produced: can be one of "confplot", "reachplot", "resplot", "transplot", "Shepard", "stressplot", "bubbleplot", "histogram" *see plot.smacofP, plot.smacofB and plot.copsc). Note that not all plots might be available for all losses.
main	the main title of the plot
asp	aspect ratio of x/y axis; defaults to 1; setting to 1 will lead to an accurate represenation of the fitted distances.
...	Further plot arguments passed: see 'plot.smacofP', 'plot.smacofB' and 'plot' for detailed information.

Details

See plot.smacofP

#@importFrom smacofx plot

Examples

```
dis<-as.matrix(smacof::kinshipdelta)
resl<-pcops(dis,loss="stress",lower=0.1,upper=5,minpts=2)
plot(resl,plot.type="confplot")
plot(resl,plot.type="reachplot")
plot(resl,plot.type="Shepard")
plot(resl,plot.type="transplot")
plot(resl,plot.type="stressplot")
plot(resl,plot.type="bubbleplot")
plot(resl,plot.type="histogram")
```

`procruster` *procruster: a procrustes function*

Description

procruster: a procrustes function

Usage

```
procruster(x)
```

Arguments

`x`	numeric matrix

Value

a matrix

`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

`spp` *Calculating stress per point*

Description

Calculating stress per point

Usage

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

Arguments

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

Value

a list

`sqdist` *Squared distances*

Description

Squared distances

Usage

```
sqdist(x)
```

Arguments

`x`	numeric matrix

Value

squared distance matrix

Index

∗ **clustering**
    cops, 6
    copstressMin, 9
    pcops, 41
∗ **multivariate**
    cop_apstress, 14
    cop_cmdscale, 16
    cop_elastic, 17
    cop_powerelastic, 19
    cop_powermds, 21
    cop_powersammon, 22
    cop_powerstress, 24
    cop_rpowerstress, 25
    cop_rstress, 27
    cop_sammon, 29
    cop_sammon2, 30
    cop_smacofSphere, 32
    cop_smacofSym, 33
    cop_sstress, 35
    cops, 6
    copstress, 7
    copstressMin, 9
    pcops, 41

BankingCrisesDistances, 3
biplotmds, *4*
biplotmds.pcops, 3
biplotmds.stops, 4
bootmds, *5*, *6*
bootmds.pcops, 5

cma_es, *12*
cop_apstress, 14
cop_cmdscale, 16
cop_elastic, 17
cop_powerelastic, 19
cop_powermds, 21
cop_powersammon, 22
cop_powerstress, 24
cop_rpowerstress, 25
cop_rstress, 27
cop_sammon, 29
cop_sammon2, 30
cop_smacofSphere, 32
cop_smacofSym, 33
cop_sstress, 35
cops, 6
copsc *(*copstressMin*)*, 9
copstress, 7
copStressMin *(*copstressMin*)*, 9
copstressMin, *6*, *7*, 9
cordillera, *12*, *43*

direct, *12*, *43*
doubleCenter, 36

enorm, 37

genoud, *12*
GenSA, *12*

hjk, *12*, *43*

jackmds, *38*
jackmds.pcops, 37

ljoptim, 38

matchphi, 40
mkBmat, 40
mkPower, 41

newuoa, *12*

optics, *11*
optim, *12*, *39*, *43*

pcops, *6*, *7*, 41
pdist, 44
phidistance, 45
plot.copsc, 45

plot.pcops, 47
powerStressMin, *12*
procruster, 48
psoptim, *43*

rStressMin, *12*

scale_adjust, 48
secularEq, 49
sim2diss, *5*
snomadr, *12*, *43*
solnl, *12*
solnp, *12*
spp, 49
sqdist, 50
stogo, *43*
stops, *41*
subplex, *12*