Package ‘stops’

June 28, 2024

Title  Structure Optimized Proximity Scaling
Version 1.6-2
Maintainer Thomas Rusch <thomas.rusch@wu.ac.at>

Description Methods that use flexible variants of multidimensional scaling (MDS) which incorporate parametric nonlinear distance transformations and trade-off the goodness-of-fit fit with structure considerations to find optimal hyperparameters, also known as structure optimized proximity scaling (STOPS) (Rusch, Mair & Hornik, 2023, <doi:10.1007/s11222-022-10197-w>). The package contains various functions, wrappers, methods and classes for fitting, plotting and displaying different 1-way MDS models with ratio, interval, ordinal optimal scaling in a STOPS framework. These cover essentially the functionality of the package smacofx, including Torgerson (classical) scaling with power transformations of dissimilarities, SMACOF MDS with powers of dissimilarities, Sammon mapping with powers of dissimilarities, elastic scaling with powers of dissimilarities, spherical SMACOF with powers of dissimilarities, (ALSCAL) s-stress MDS with powers of dissimilarities, r-stress MDS, MDS with powers of dissimilarities and configuration distances, elastic scaling powers of dissimilarities and configuration distances, Sammon mapping powers of dissimilarities and configuration distances, power stress MDS (POST-MDS), approximate power stress, Box-Cox MDS, local MDS, Isomap, curvilinear component analysis (CLCA), curvilinear distance analysis (CLDA) and sparsified (power) multidimensional scaling and (power) multidimensional distance analysis (experimental models from smacofx influenced by CLCA). All of these models can also be fit by optimizing over hyperparameters based on goodness-of-fit fit only (i.e., no structure considerations). The package further contains functions for optimization, specifically the adaptive Luus-Jaakola algorithm and a wrapper for Bayesian optimization with treed Gaussian process with jumps to linear models, and functions for various c-structuredness indices.

Depends  R (>= 3.5.0), smacofx
Imports  acepack, clue, cmaes, cordillera, dfoptim, DiceOptim, DiceKriging, energy, minerva, nloptr, pomp, pso, scagnostics, smacof, tgp, vegan

Enhances stats
Suggests R.rsp
License GPL-2 | GPL-3
LazyData true
URL: https://r-forge.r-project.org/projects/stops/

VignetteBuilder: R.rsp

Encoding: UTF-8

RoxygenNote: 7.3.1

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: 2024-06-28 07:30:02 UTC

Contents

BankingCrisesDistances .................................................. 3
bootmds.stops .............................................................. 4
coef.stops ................................................................. 5
c_association ............................................................... 6
c_clumpiness ............................................................... 7
c_clusteredness ............................................................ 7
c_complexity ............................................................... 9
c_convexity ............................................................... 10
c_dependence ............................................................. 10
c_faithfulness ............................................................. 11
c_functionality ........................................................... 12
c_hierarchy ............................................................... 13
c_inequality .............................................................. 13
c_linearity ............................................................... 14
c_manifoldness .......................................................... 15
c_mine ................................................................. 16
c_nonmonotonicity ...................................................... 16
c_outlying ............................................................... 17
c_regularity ............................................................. 18
c_shepardness ........................................................... 19
c_skinniness ............................................................. 20
c_sparsity ............................................................... 20
c_striatedness ............................................................ 21
c_stringiness ............................................................. 22
jackmds.stops ............................................................. 22
knn_dist ................................................................. 23
ljoptim ................................................................. 24
Pendigits500 .............................................................. 24
plot.stops ............................................................... 25
print.stops ............................................................. 26
print.summary.stops .................................................... 27
residuals.stops .......................................................... 28
# Banking Crises Distances

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 crisis 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)

bootmds.stops  
MDS Bootstrap for stops 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 (note we cannot as of yet have any dissimilarity matrix).

Usage

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

Arguments

object  Object of class stops or pcops.
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 confidence 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.
Description

S3 coef method for stops objects

Usage

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

Arguments

object object of class stops
... addditional arguments

Value

a vector of hyperparmeters theta
c-associated calculates the c-association based on the maximal information coefficient. We define c-association as the aggregated association between any two columns in \texttt{confs}.

### Usage

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

### Arguments

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

### Value

A numeric value; association (aggregated maximal information coefficient MIC, see \texttt{mine})

### Examples

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

Measures the c-clumpiness structure

Usage

c_clumpiness(conf, aggr = max)

Arguments

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

Value

a numeric value; clumpiness (see scagnostics)

Examples

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

c_clusteredness

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

Description

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

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

Arguments

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

Value

  a numeric value; clusteredness (see cordillera)

Examples

delts<-smacof::kinshipdelta
c_complexity

\[
\text{dis}<-\text{smacofSym(delts)$confdist}
\]
\[
\text{c_clusteredness(dis,minpts=3)}
\]

---

c_complexity

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

Description

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

Usage

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

Arguments

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

Value

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

\begin{verbatim}
x<-seq(-3,3,length.out=200)
y<-sqrt(3^2-x^2)
z<- sin(y-x)
confs<-cbind(x,y,z)
c_complexity(confs)
\end{verbatim}

c_convexity c-convexity

Description

Measures the c-convexity structure

Usage

\begin{verbatim}
c_convexity(conf, aggr = max)
\end{verbatim}

Arguments

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

Value

A numeric value; convexity (see scagnostics)

Examples

\begin{verbatim}
delts<-smacof::kinshipdelta
conf<-smacof::smacofSym(delts)$conf
plot(conf,pch=19,asp=1)
c_convexity(conf)
\end{verbatim}

c_dependence c-dependence

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

Description

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

Usage

\begin{verbatim}
c_dependence(confs, aggr = max, index = 1)
\end{verbatim}
c_faithfulness

Arguments

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

Value

a numeric value; dependence (aggregated distance correlation)

Examples

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

deltas<-smacof::kinshipdelta
diss<-smacofSym(deltas)$confdist
c_faithfulness(diss,deltas,k=3)

c_faithfulness

\textit{c-faithfulness} calculates the c-faithfulness based on the index by Chen and Buja 2013 (M_adj) with equal input neighbourhoods

Description

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

Usage

c_faithfulness(conf, obsdiss, k = 3, ...)

Arguments

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

Value

a numeric value; faithfulness

Examples

deltas<-smacof::kinshipdelta
diss<-smacofSym(deltas)$confdist
c_faithfulness(diss,deltas,k=3)
c_functionality

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

Description

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

Usage

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

Arguments

confs a numeric matrix or data frame

aggr the aggregation function for configurations of more than two dimensions. Defaults to mean

alpha an optional number of cells allowed in the X-by-Y search-grid. Default value is 1

C an optional number determining the starting point of the X-by-Y search-grid. When trying to partition the x-axis into X columns, the algorithm will start with at most C X clumps. Default value is 15.

var.thr minimum value allowed for the variance of the input variables, since mine can not be computed in case of variance close to 0. Default value is 1e-5.

zeta integer in [0,1] (?). If NULL (default) it is set to 1-MIC. It can be set to zero for noiseless functions, but the default choice is the most appropriate parametrization for general cases (as stated in Reshef et al.). It provides robustness.

Value

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

Examples

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

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

c_hierarchy

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

Description

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

Usage

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

Arguments

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

Value

a numeric value; hierarchy (see cl_validity)

Examples

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

c_inequality

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

Description

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

c_inequality(confs)

Arguments

confs a numeric matrix or data frame

Value

a numeric value; inequality (Pearson's coefficient of variation of the fitted distance matrix)

Examples

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

c_linearity

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

Description

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

Usage

c_linearity(confs, aggr = max)

Arguments

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

Value

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

Examples

x<-1:10
y<-2+3*x+rnorm(10)
z<-sin(y-x)
confs<-cbind(z,y,x)
c_linearity(confs)
c_manifoldness calculates c-manifoldness as the aggregated maximal correlation coefficient (i.e., Pearson correlation of the ACE transformed variables) of all pairwise combinations of two different columns in confs. If there is an NA (happens usually when the optimal transformation of any variable is a constant and therefore the covariance is 0 but also one of the sds in the denominator), it gets skipped.

### Usage

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

### Arguments

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

### Value

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

### Examples

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

wrapper for getting the mine coefficients

Description

wrapper for getting the mine coefficients

Usage

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

Arguments

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

c_nonmonotonicity

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

Description

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

Usage

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

Arguments

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

Value

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

Examples

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

c_outlying c-outlying

Description

Measures the c-outlying structure
Usage

c_outlying(conf, aggr = max)

Arguments

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

Value

a numeric value; outlying (see scagnostics)

Examples

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

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

Description

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

Usage

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

Arguments

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

The c_shepardness function calculates the c-shepardness as the correlation between a loess smoother of the transformed distances and the transformed dissimilarities.

#### Description

The c-shepardness calculates the c-shepardness as the correlation between a loess smoother of the transformed distances and the transformed dissimilarities.

#### Usage

```r
c_shepardness(object)
```

#### Arguments

- `object` an object of class `smacofP`

#### Value

- a numeric value
**Examples**

```r
delts<-smacof::kinshipdelta
res<-smacofx::postmds(delts)
c_shepardness(res)
```

---

**c_skinniness**  *c-skinniness*

**Description**

Measures the c-skinniness structure

**Usage**

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

**Arguments**

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

**Value**

a numeric value; skinniness (see *scagnostics*)

**Examples**

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

---

**c_sparsity**  *c-sparsity*

**Description**

Measures the c-sparsity structure

**Usage**

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

Arguments

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

Value

a numeric value; sparsity (see scagnostics)

Examples

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

c_striatedness  c-striatedness

Description

Measures the c-striatedness structure

Usage

c_striatedness(conf, aggr = max)

Arguments

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

Value

a numeric value; striatedness (see scagnostics)

Examples

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

**Description**
Measures the c-stringiness structure

**Usage**
c_stringiness(conf, aggr = max)

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

**Value**
a numeric value; stringiness (see scagnostics)

**Examples**
```r
deltas <- smacof::kinshipdelta
c <- smacof::smacofSym(deltas)$conf
plot(c, pch = 19, asp = 1)
c_stringiness(c)
```

---

### jackmds.stops

**MDS Jackknife for stops objects**

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

**Usage**
```r
## S3 method for class 'stops'
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 hyperparameter 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 <- stops(diso,loss="powermds",lower=c(1,1),upper=c(5,5),itmaxps=100)
res.jk <- jackmds(fit)
plot(res.jk)

knn_dist

calculate k nearest neighbours from a distance matrix

Description

calculate k nearest neighbours from a distance matrix

Usage

knn_dist(dis, k)
Arguments

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

---

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

**Description**

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

**Usage**

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

**Arguments**

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

A list with the components (*optim*)

- **par** The position of the optimum 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**

```r
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
```

---

**Description**

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

**Usage**

```r
data(Pendigits500)
```
Format

A data frame with 500 rows and 17 variables

Details

The variables are

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

Source


plot.stops

S3 plot method for stops objects

Description

S3 plot method for stops objects

Usage

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

Arguments

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

Value

no return value, just plots
print.stops

S3 print method for stops objects

Description

S3 print method for stops objects

Usage

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

Arguments

x       stops object
...
...    additional arguments

Value

no return value, just prints

print.summary.stops

S3 print method for summary.stops

Description

S3 print method for summary.stops

Usage

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

Arguments

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

Value

no return value, just prints
residuals.stops  \textit{S3 residuals method for stops}

\section*{Description}
S3 residuals method for stops

\section*{Usage}
\begin{verbatim}
## S3 method for class 'stops'
residuals(object, ...)
\end{verbatim}

\section*{Arguments}
\begin{itemize}
  \item \texttt{object} \hspace{1cm} object of class \texttt{stops}
  \item \ldots \hspace{1cm} additional arguments
\end{itemize}

\section*{Value}
a vector of residuals (observed minus fitted distances)

\section*{stoploss  \textit{Calculate the weighted multiobjective loss function used in STOPS}}

\section*{Description}
Calculate the weighted multiobjective loss function used in STOPS

\section*{Usage}
\begin{verbatim}
stoploss(
  obj,
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
                 "cassociation", "cnonmonotonicity", "cfunctionality", "ccomplexity", "cfaithfulness",
                 "cregularity", "chierarchy", "cconvexity", "cstripedness", "coutlying",
                 "cskinniness", "csparsity", "cstringiness", "cclumpiness", "cinequality"),
  strucweight = rep(-1/length(structures), length(structures)),
  strucpars,
  stoptype = c("additive", "multiplicative"),
  verbose = 0
)
\end{verbatim}
**Arguments**

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

**Value**

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

---

**Description**

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

**Usage**

```r
stops(
  dis,
  loss = "stress",
  theta = 1,
  type = "ratio",
  structures,
  ndim = 2,
  weightmat = NULL,
  init = NULL,
  stressweight = 1,
  strucweight,
  strucpars,
  optimmethod = c("SANN", "ALJ", "pso", "Kriging", "tgp", "direct", "stogo", "cobyla",
                  "crs2lm", "isres", "mlsl", "neldermead", "sbplx", "hjk", "cmaes"),
  lower,
  upper,
  verbose = 0,
```
stotype = c("additive", "multiplicative"),
initpoints = 10,
itmax = 50,
itmaxps = 10000,
model,
control,
... )

Arguments

- **dis**
  - numeric matrix or dist object of a matrix of proximities

- **loss**
  - which loss function to be used for fitting, defaults to stress.

- **theta**
  - hyperparameter vector starting values for the transformation functions. If the length is smaller than the number of hyperparameters for the MDS version the vector gets recycled (see the corresponding stop_XXX function or the vignette for how theta must look like exactly for each loss). If larger than the number of hyperparameters for the MDS method, an error is thrown. If completely missing theta is set to 1 and recycled.

- **type**
  - type of MDS optimal scaling (implicit transformation). One of "ratio", "interval" or "ordinal". Default is "ratio". Not every type can be used with every loss, only ratio works with all.

- **structures**
  - character vector of which c-structuredness indices should be considered; if missing no structure is considered.

- **ndim**
  - number of dimensions of the target space

- **weightmat**
  - (optional) a matrix of nonnegative weights; defaults to 1 for all off diagonals

- **init**
  - (optional) initial configuration

- **stressweight**
  - weight to be used for the fit measure; defaults to 1

- **strucweight**
  - vector of weights to be used for the c-structuredness indices (in the same order as in structures); defaults to -1/length(structures) for each index

- **strucpars**
  - (possibly named with the structure). Metaparameters for the structuredness indices (gamma in the article). It's safest for it be a list of lists with the named arguments for the structuredness indices and the order of the lists must be like the order of structures. So something like this list(list(par1Struc1=par1Struc1,par2Struc1=par2Struc1),list(par1Struc2=par1Struc2,par2Struc2=par2Struc2),...) where parYStrucX are the named arguments for the metaparameter Y of the structure X the list elements corresponds to. For a structure without parameters, set NULL. Parameters in different list elements parYStrucX can have the same name. For example, say we want to use cclusteredness with metaparameters epsilon=10 and k=4 (and the default for the other parameters), cdependence with no metaparameters and cfaithfulness with metaparameter k=7 one would list(list(epsilon=10,k=4),list(NULL),list(dis=obdiss,k=6)) for structures vector ("cclusteredness","cdependence","cfaithfulness"). The parameter lists must be in the same ordering as the indices in structures. If missing it is set to NULL and defaults are used. It is also possible to supply a structure's metaparameters as a list of vectors with named elements if the metaparameters are scalars, so like list(c(par1Struc1=par1Struc1,par2Struc1=par2Struc1,...),c(par1Struc2=p...)}
That can have unintended consequences if the metaparameter is a vector or matrix.

**optimmethod**

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

**lower**

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

**upper**

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

**verbose**

Numeric value that prints information on the fitting process; >2 is very verbose.

**stotype**

Which aggregation for the multi objective target function? Either 'additive' (default) or 'multiplicative'.

**initpoints**

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

**itmax**

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

**itmaxps**

Maximum number of iterations of the inner optimization (to obtain the PS configuration).

**model**

A character specifying the surrogate model to use. For Kriging it specifies the covariance kernel for the GP prior; see `covTensorProduct-class` defaults to "powerexp". For tgp it specifies the non stationary process used see `bgp`, defaults to "btgpllm".

**control**

A control argument passed to the outer optimization procedure. Will override any other control arguments passed, especially verbose and itmax. For the effect of control, see the functions `pomp::sannbox` for SANN and `pso::psoptim` for pso, `cmaes::cma_es` for cmaes, `dfoptim::hjkb` for hjk and the `nloptr` docs for the algorithms direct, stogo, cobyla, crs2lm, isres, mlsl, neldermead, sbplx.

... additional arguments passed to the outer optimization procedures (not fully tested).
Details

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

Value

A list with the components

• stoploss: the stoploss value
• optim: the object returned from the optimization procedure
• stressweight: the stressweight
• strucweight: the vector of structure weights
• call: the call
• optimmethod: The solver selected
• loss: The PS badness-of-fit function
• nobj: the number of objects in the configuration
• type: The type of stoploss scalarisation (additive or multiplicative)
• fit: The fitted PS object (most importantly $fit$conf the fitted configuration)
• stoptype: Type of stoploss combinatio

Examples

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

#use higher itmax in general, we use 5 just to shorten the tests
data(BankingCrisesDistances)
strucpar<-list(c(epsilon=10,minpts=2),NULL) #parameters for indices
res1<-stops(BankingCrisesDistances[,1:69],loss="stress",verbose=0,
structures=c("cclusteredness","clinearity"),strucpars=strucpar,
lower=0,upper=10,itmax=5)
res1

strucpar<-list(list(alpha=0.6,C=15,var.thr=1e-5,zeta=NULL),
list(alpha=0.6,C=15,var.thr=1e-5,zeta=NULL))
res1<-stops(BankingCrisesDistances[,1:69],loss="stress",verbose=0,
structures=c("cfunctionality","ccomplexity"),strucpars=strucpar,
lower=0,upper=10,itmax=5)
res1
Description

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

Usage

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

Arguments

dis numeric matrix or dist object of a matrix of proximities
theta the theta vector of parameters to optimize over. Must be of length 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
structures a character vector listing the structure indices to use. They always are called "cfoo" with foo being the structure.
strucweight  
weight to be used for the structures; defaults to 1/number of structures

strucpars  
a list of list of parameters for the structuredness indices; each list element corresponds to one index in the order of the appearance in structures vector. See examples.

verbose  
numeric value that prints information on the fitting process; >2 is extremely verbose

stoptype  
which weighting to be used in the multi-objective optimization? Either 'additive' (default) or 'multiplicative'.

Value

A list with the components

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

stop_bcmds

STOPS version of Box Cox Stress

Description

STOPS version of Box Cox Stress

Usage

```
stop_bcmds(
  dis,
  theta = c(1, 1, 0),
  type = "ratio",
  weightmat = NULL,
  init = NULL,
  ndim = 2,
  itmaxi = 5000,
  ...,
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
                 "cassociation", "cmonotonicity", "cfunctionality", "ccomplexity", "cfaithfulness",
                 "cregularity", "chierarchy", "cconvexity", "cstriatedness", "coutlying",
                 "cskinniness", "csparsity", "cstringiness", "cclumpiness", "cinequality"),
  strucweight = rep(1/length(structures), length(structures)),
)```
stop_bcmds

strucpars,
verbose = 0,
stoptype = c("additive", "multiplicative")
)

Arguments

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

Value

A list with the components

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

STOPS version of CLCA.

Description

CLCA with free lambda0 and 20 epochs. Should we add alpha0?

Usage

stop_clca(
  dis,
  theta = 3 * max(sd(dis)),
  type = "ratio",
  weightmat = 1 - diag(nrow(dis)),
  init = NULL,
  ndim = 2,
  itmaxi = 10000,
  ...
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
                 "cassociation", "ccomonotonicity", "cfrequency", "ccomplexity", "cfaithfulness",
                 "cregularity", "chierarchy", "cconvexity", "cstriatedness", "coutlying",
                 "csparseness", "cssparsity", "cstringiness", "cclassification",
                 "cinequality"),
  strucweight = rep(1/length(structures), length(structures)),
  strucpars,
  verbose = 0,
  stoptype = c("additive", "multiplicative")
)

Arguments

dis numeric matrix or dist object of a matrix of proximities
theta the theta vector of explicit parameters; lambda0 for the maximal neighbourhood. Defaults to 100.

Arguments

type MDS type.
weightmat (optional) a matrix of nonnegative weights
init (optional) initial configuration
ndim number of dimensions of the target space
itmaxi number of iterations
... additional arguments to be passed to the fitting procedure
stressweight weight to be used for the fit measure; defaults to 1
structures a character vector listing the structure indices to use. They always are called "cfoo" with foo being the structure.
strucweight weight to be used for the structures; defaults to 1/number of structures
**Value**

A list with the components

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

---

**Description**

CLDA with free lambda0 and epsilon and 20 epochs. Should we add alpha0?

**Usage**

```r
stop_cldae(
  dis,
  theta = rep(3 * max(sd(dis)), 2),
  type = "ratio",
  weightmat = 1 - diag(nrow(dis)),
  init = NULL,
  ndim = 2,
  itmaxi = 10000,
  ...
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
                 "cassociation", "cnonmonotonicity", "cfunctionality", "ccomplexity",
                 "cfaithfulness", "cregularity", "chierarchy", "cconvexity", "cstiatedness",
                 "coutlying", "cskinniness", "csparsity", "cstringiness", "cclumpiness", "cinequality"),
  strucwight = rep(1/length(structures), length(structures)),
  strucpars,
  verbose = 0,
  stoptype = c("additive", "multiplicative")
)
```
**Arguments**

- **dis**: numeric matrix or dist object of a matrix of proximities
- **theta**: the theta vector of explicit parameters; first is lambda0 for the maximal neighbourhood and second is k for the number of neighbours for the geodesic distance.
- **type**: MDS type.
- **weightmat**: (optional) a matrix of nonnegative weights
- **init**: (optional) initial configuration
- **ndim**: number of dimensions of the target space
- **itmaxi**: number of iterations
- **stressweight**: weight to be used for the fit measure; defaults to 1
- **structures**: a character vector listing the structure indices to use. They always are called "cfoo" with foo being the structure.
- **strucweight**: weight to be used for the structures; defaults to 1/number of structures
- **strucpars**: a list of parameters for the structuredness indices; each list element corresponds to one index in the order of the appearance in structures
- **verbose**: numeric value that prints information on the fitting process; >2 is extremely verbose
- **stoptype**: which weighting to be used in the multi-objective optimization? Either 'additive' (default) or 'multiplicative'.

**Value**

A list with the components

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

CLDA with free lambda0 and k and 20 epochs. Should we add alpha0?

Usage

```r
stop_cldak(
  dis,
  theta = c(3 * max(sd(dis)), nrow(dis)/4),
  type = "ratio",
  weightmat = 1 - diag(nrow(dis)),
  init = NULL,
  ndim = 2,
  itmaxi = 10000,
  ..., 
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
                "cassociation", "cnonmonotonicity", "cffunctionality", "ccomplexity", "cfaithfulness",
                "cregularity", "chierarchy", "cconvexity", "cstrializedness", "coutlying",
                "cskinniness", "csparseity", "cstringiness", "cclumpiness", "cinequality"),
  strucweight = rep(1/length(structures), length(structures)),
  strucpars,
  verbose = 0,
  stoptype = c("additive", "multiplicative")
)
```

Arguments

dis numeric matrix or dist object of a matrix of proximities
theta the theta vector of explicit parameters; first is lambda0 for the maximal neighbourhood and second is k for the number of neighbours for the geodesic distance.
type MDS type.
weightmat (optional) a matrix of nonnegative weights
init (optional) initial configuration
ndim number of dimensions of the target space
itmaxi number of iterations
... additional arguments to be passed to the fitting procedure
stressweight weight to be used for the fit measure; defaults to 1
structures a character vector listing the structure indices to use. They always are called "cfoo" with foo being the structure.
strucweight: weight to be used for the structures; defaults to 1/number of structures
strucpars: a list of parameters for the structuredness indices; each list element corresponds to one index in the order of the appearance in structures
verbose: numeric value that prints information on the fitting process; >2 is extremely verbose
stoptype: which weighting to be used in the multi-objective optimization? Either 'additive' (default) or 'multiplicative'.

Value

A list with the components
- stress: the stress-1 value
- stress.m: default normalized stress
- stoploss: the weighted loss value
- struc: the structuredness indices
- parameters: the parameters used for fitting (tau)
- fit: the returned object of the fitting procedure
- stopobj: the stopobj object

stop_cmdscale

STOPS version of strain

Description

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

Usage

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

```r
stoptype = c("additive", "multiplicative"),
itmaxi = 1000,
add = 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. Not used.
- **ndim**: number of dimensions of the target space
- **init**: (optional) initial configuration
- **...**: additional arguments to be passed to the fitting procedure
- **stressweight**: weight to be used for the fit measure; defaults to 1
- **structures**: which structuredness indices to be included in the loss
- **strucweight**: weight to be used for the structuredness indices; defaults to 1/#number of structures
- **strucpars**: the parameters for the structuredness indices
- **verbose**: numeric value that prints information on the fitting process; >2 is extremely verbose
- **stoptype**: How to construct the target function for the multi objective optimization? Either 'additive' (default) or 'multiplicative'
- **itmaxi**: number of iterations. No effect here.
- **add**: if TRUE dis is made to Euclidean distances

**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**: explicitly normalized stress (manually calculated)
- **stoploss**: the weighted loss value
- **indices**: the values of the structuredness indices
- **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
- **stopobj**: the stopobj object
stop_elastic

STOPS versions of elastic scaling models (via smacofSym)

Description

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

Usage

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

Arguments

- `dis` numeric matrix or dist object of a matrix of proximities
- `theta` the theta vector of powers; this must be a scalar of the lambda transformation for the observed proximities. Defaults to 1.
- `type` MDS type. Defaults to 'ratio'.
- `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
- `...` additional arguments to be passed to the fitting procedure
- `stressweight` weight to be used for the fit measure; defaults to 1
- `structures` which structuredness indices to be included in the loss
stop_isomap1

strucweight  weight to be used for the structuredness indices; ; defaults to 1/#number of structures
strucpars  the parameters for the structuredness indices
verbose  numeric value that prints information on the fitting process; >2 is extremely verbose
stoptype  How to construct the target function for the multi objective optimization? Either 'additive' (default) or 'multiplicative'

Value
A list with the components
- stress: the stress-1 (sqrt(stress.m))
- stress.m: default normalized stress (used for STOPs)
- stoploss: the weighted loss value
- indices: the values of the structuredness indices
- parameters: the parameters used for fitting (lambda)
- fit: the returned object of the fitting procedure
- stopobj: the stopobj objects

Description
Free parameter is k.

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

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

Details

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

Value

A list with the components

- **stress**: Not really stress but 1-GOF[2] where GOF is the second element returned from smacofx::cmdscale (the sum of the first ndim eigenvalues divided by the sum of all absolute eigenvalues).
- **stress.m**: default normalized stress (sqrt explicitly normalized stress; really the stress this time)
- **stoploss**: the weighted loss value
- **indices**: the values of the structuredness indices
- **parameters**: the parameters used for fitting
- **fit**: the returned object of the fitting procedure
- **stopobj**: the stopobj object
**stop_isomap2**

STOPS version of isomap over real epsilon.

**Description**

Free parameter is eps.

**Usage**

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

**Arguments**

- **dis**: numeric matrix or dist object of a matrix of proximities
- **theta**: the number of shortest dissimilarities retained for a point (neighbourhood region), the isomap parameter. Defaults to the 0.1 quantile of the empirical distribution of dis.
- **type**: MDS type. Is "ratio".
- **weightmat**: (optional) a matrix of nonnegative weights
- **ndim**: number of dimensions of the target space
- **init**: (optional) initial configuration
- **stressweight**: weight to be used for the fit measure; defaults to 1
- **structures**: which structuredness indices to be included in the loss
- **strucweight**: weight to be used for the structuredness indices; defaults to 1/#number of structures
- **strucpars**: the parameters for the structuredness indices
stop_lmds

verbose numeric value hat prints information on the fitting process; >2 is extremely verbose
stoptype How to construct the target function for the multi objective optimization? Either 'additive' (default) or 'multiplicative'
itmaxi placeholder for compatibility in stops call; not used

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

Value
A list with the components

• stress: Not really stress but 1-GOF[2] where GOF is the second element returned from cmdscale (the sum of the first ndim absolute eigenvalues divided by the sum of all absolute eigenvalues).
• stress.m: default normalized stress (sqrt explicitly normalized stress; really the stress this time)
• stoploss: the weighted loss value
• indices: the values of the structuredness indices
• parameters: the parameters used for fitting
• fit: the returned object of the fitting procedure
• stopobj: the stopobj object

stop_lmds

STOPS version of lMDS

Description
STOPS version of lMDS

Usage
stop_lmds(
  dis,
  theta = c(2, 0.5),
  type = "ratio",
  weightmat = NULL,
  init = NULL,
  ndim = 2,
itmaxi = 5000,
..., 
stressweight = 1,
strucweight = rep(1/length(structures), length(structures)),
strucpars,
verbose = 0,
stopty = c("additive", "multiplicative")
)

Arguments

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

Value

A list with the components

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

**STOPs version of elastic scaling with powers for proximities and distances**

**Description**

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

**Usage**

```r
stop_powerelastic(
  dis,
  theta = c(1, 1),
  type = "ratio",
  weightmat = 1 - diag(nrow(dis)),
  init = NULL,
  ndim = 2,
  itmaxi = 1e+05,
  ...
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
                 "ccassociation", "ccnonmonotonicity", "cffunctionality", "cccomplexity", "cfaithfulness",
                 "cregularity", "chierarchy", "cconvexity", "cstriatedness", "coutlying",
                 "cskinniness", "csparsity", "cstringiness", "ccomplexity", "cfaithfulness",
                 "cregularity", "chierarchy", "ccconvexity", "cstriatedness", "coutlying"),
  strucweight = rep(1/length(structures), length(structures)),
  strucpars,
  verbose = 0,
  stoptype = c("additive", "multiplicative")
)
```

**Arguments**

- **dis**: numeric matrix or dist object of a matrix of proximities
- **theta**: the theta vector of powers; a vector of length two where the first element is kappa (for the fitted distances), the second lambda (for the observed proximities). If a scalar for the free parameters is given it is recycled. Defaults to 1 1.
- **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
- **...**: additional arguments to be passed to the fitting procedure
- **stressweight**: weight to be used for the fit measure; defaults to 1
- **structures**: which structures to look for
stop_powermds

   strucweight  weight to be used for the structures; defaults to 0.5
   strucpars    a list of parameters for the structuredness indices; each list element corresponds
to one index in the order of the appearance in structures
   verbose     numeric value that prints information on the fitting process; >2 is extremely ver-
              bose
   stoptype    which weighting to be used in the multi-objective optimization? Either 'addi-
              tive' (default) or 'multiplicative'.

Value

A list with the components

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

stop_powermds

STOPS version of powermds

Description

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

Usage

stop_powermds(
  dis,
  theta = c(1, 1),
  type = "ratio",
  weightmat = 1 - diag(nrow(dis)),
  init = NULL,
  ndim = 2,
  itmaxi = 10000,
  ...
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
                   "cassociation", "cnonmonotonicity", "cfunctionality", "ccomplexity", "cfaultiness",
                   "cregularity", "chierarchy", "cconvexity", "cstriatedness", "coutlying",
                   "csparsity", "cstringiness", "cclassiness", "cclassiness", "cinequality"),
  strucweight = rep(1/length(structures), length(structures)),
  strucpars,
stop_powermds

verbose = 0,
stoptype = c("additive", "multiplicative")
)

Arguments

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

Value

A list with the components

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

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

**Usage**

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

**Arguments**

- `dis`: numeric matrix or dist object of a matrix of proximities
- `theta`: the theta vector of powers; a vector of length two where the first element is kappa (for the fitted distances), the second lambda (for the observed proximities). If a scalar is given it is recycled for the free parameters. Defaults to 1 1.
- `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
- `...`: additional arguments to be passed to the fitting procedure
- `stressweight`: weight to be used for the fit measure; defaults to 1
- `structures`: which structures to look for
- `strucweight`: weight to be used for the structures; defaults to 0.5
stop_powerstress

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>strucpars</td>
<td>a list of parameters for the structuredness indices; each list element corresponds to one index in the order of the appearance in structures</td>
</tr>
<tr>
<td>verbose</td>
<td>numeric value that prints information on the fitting process; &gt;2 is extremely verbose</td>
</tr>
<tr>
<td>stoptype</td>
<td>which weighting to be used in the multi-objective optimization? Either 'additive' (default) or 'multiplicative'.</td>
</tr>
</tbody>
</table>

Value

A list with the components

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

---

**Description**

Power stress with free kappa and lambda and rho.

**Usage**

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

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

Value

A list with the components

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

STOPS version of restricted powerstress

Usage

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

Arguments

dis numeric matrix or dist object of a matrix of proximities

theta the theta vector of powers; the first two arguments are for kappa and lambda and should be equal (for the fitted distances and observed proximities), the third nu (for the weights). Internally the kappa and lambda are equated. If a scalar is given it is recycled (so all elements of theta are equal); if a vector of length 2 is given, it gets expanded to c(theta[1],theta[1],theta[2]). Defaults to 1 1 1.

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.

stressweight weight to be used for the fit measure; defaults to 1
stop_rstress

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

**Value**

A list with the components

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

---

**Description**

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

**Usage**

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

Arguments

dis numeric matrix or dist object of a matrix of proximities
theta the theta vector of powers; this must be a scalar of the kappa=2r transformation for the fitted distances proximities. Defaults to 1. Note that what is returned is r, not kappa.
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.
... additional arguments to be passed to the fitting procedure
stressweight weight to be used for the fit measure; defaults to 1
structures which structuredness indices to be included in the loss
strucweight weight to be used for the structuredness indices; ; defaults to 1/#number of structures
strucpars the parameters for the structuredness indices
verbose numeric value hat prints information on the fitting process; >2 is extremely verbose
stoptype How to construct the target function for the multi objective optimization? Either 'additive' (default) or 'multiplicative'

Value

A list with the components

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

STOPs version of Sammon mapping

Description

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

Usage

```r
stop_sammon(
  dis,
  theta = 1,
  type = "ratio",
  ndim = 2,
  init = NULL,
  weightmat = NULL,
  itmaxi = 1000,
  ...
)
```

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. Ignored here.
- `ndim`: number of dimensions of the target space
- `init`: (optional) initial configuration
- `weightmat`: a matrix of nonnegative weights. Has no effect here.
- `itmaxi`: number of iterations
- `...`: additional arguments to be passed to the fitting procedure
- `stressweight`: weight to be used for the fit measure; defaults to 1
- `structures`: which structuredness indices to be included in the loss
strucweight  weight to be used for the structuredness indices; ; defaults to 1/#number of structures
strucpars  the parameters for the structuredness indices
verbose  numeric value that prints information on the fitting process; >2 is extremely verbose
stoptype  How to construct the target function for the multi objective optimization? Either 'additive' (default) or 'multiplicative'

Value

A list with the components

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

stop_sammon2

Another STOPS version of Sammon mapping models (via smacofSym)

Description

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

Usage

stop_sammon2(
  dis,
  theta = 1,
  type = "ratio",
  ndim = 2,
  weightmat = NULL,
  init = NULL,
  itmaxi = 1000,
  ...
  stressweight = 1,

...
stop_sammon2

```r
strucweight = rep(1/length(structures), length(structures)),
strucpars,
verbose = 0,
stoptype = c("additive", "multiplicative")
)
```

**Arguments**

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

**Value**

A list with the components

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

```r
stop_smacofSphere(
    dis,
    theta = 1,
    type = "ratio",
    ndim = 2,
    weightmat = NULL,
    init = NULL,
    itmaxi = 1000,
    ...
    stressweight = 1,
    strucweight = rep(1/length(structures), length(structures)),
    strucpars,
    verbose = 0,
    stoptype = c("additive", "multiplicative")
)
```

Arguments

dis numeric matrix or dist object of a matrix of proximities

theta the theta vector of powers; this must be a scalar of the lambda transformation for the observed proximities. Defaults to 1.

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

... additional arguments to be passed to the fitting procedure

stressweight weight to be used for the fit measure; defaults to 1

structures which structuredness indices to be included in the loss
stop_smacofSym

strucweight weight to be used for the structuredness indices; defaults to 1/#number of structures
strucpars the parameters for the structuredness indices
verbose numeric value that prints information on the fitting process; >2 is extremely verbose
stop type How to construct the target function for the multi objective optimization? Either 'additive' (default) or 'multiplicative'

Value

A list with the components

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

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

```r
stop_smacofSym(
  dis,
  theta = 1,
  type = "ratio",
  ndim = 2,
  weightmat = 1 - diag(nrow(dis)),
  init = NULL,
  itmaxi = 1000,
  ...
)
```

stressweight = 1,
strucweight = rep(1/length(structures), length(structures)),
```
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.
type MDS type. Defaults to 'ratio'.
ndim number of dimensions of the target space
weightmat (optional) a matrix of nonnegative weights
init (optional) initial configuration
itmaxi number of iterations
... additional arguments to be passed to the fitting
structures which structuredness indices to be included in the loss
stressweight weight to be used for the fit measure; defaults to 1
strucweight weight to be used for the structuredness indices; defaults to 1/#number of structures
strucpars the parameters for the structuredness indices
verbose numeric value that prints information on the fitting process; >2 is extremely verbose
stoptype How to construct the target function for the multi objective optimization? Either 'additive' (default) or 'multiplicative'

Value

A list with the components

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

smdda with free parameters tau and epsilon.

### Usage

```r
stop_smddae(
  dis,
  theta = c(100, 100),
  type = "ratio",
  weightmat = 1 - diag(nrow(dis)),
  init = NULL,
  ndim = 2,
  itmaxi = 10000,
  ...
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
                  "cassociation", "cnonmonotonicity", "cfunctionality", "ccomplexity", "cfaithfulness",
                  "cregularity", "chierarchy", "cconvexity", "cstriatedness", "coutlying",
                  "cskininess", "csparsity", "cstringiness", "cclumpiness", "cinequality"),
  strucweight = rep(1/length(structures), length(structures)),
  strucpars,
  verbose = 0,
  stoptype = c("additive", "multiplicative")
)
```

### Arguments

- **dis**: numeric matrix or dist object of a matrix of proximities
- **theta**: the theta vector of explicit parameters; first is tau for the neighbourhood, second is epsilon for isomapdist. Defaults to 100, 100.
- **type**: MDS type.
- **weightmat**: (optional) a matrix of nonnegative weights
- **init**: (optional) initial configuration
- **ndim**: number of dimensions of the target space
- **itmaxi**: number of iterations
- **...**: additional arguments to be passed to the fitting procedure
- **stressweight**: weight to be used for the fit measure; defaults to 1
- **structures**: a character vector listing the structure indices to use. They always are called "cfoo" with foo being the structure.
strucweight  weight to be used for the structures; defaults to 1/number of structures
strucpars  a list of parameters for the structuredness indices; each list element corresponds to one index in the order of the appearance in structures
verbose  numeric value hat prints information on the fitting process; >2 is extremely verbose
stoptype  which weighting to be used in the multi-objective optimization? Either 'additive' (default) or 'multiplicative'.

Value

A list with the components

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

---

stop_smddak  

STOPS version of sparsified multidimensional distance analysis for fixed k and tau

Description

smdda with free parameters tau and k.

Usage

stop_smddak(
  dis,
  theta = c(100, 10),
  type = "ratio",
  weightmat = 1 - diag(nrow(dis)),
  init = NULL,
  ndim = 2,
  itmaxi = 10000,
  ...
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
  "cassociation", "cnonmonotonicity", "cfunctionality", "ccomplexity", "cfaithfulness",
  "cregularity", "chierarchy", "cconvexity", "cstriatedness", "coutlying",
  "cskinniness", "csparsity", "cstringiness", "cclumpiness", "cinequality"),
  strucweight = rep(1/length(structures), length(structures)),
)
strucpars,
verbose = 0,
stoptype = c("additive", "multiplicative")
)

Arguments

dis numeric matrix or dist object of a matrix of proximities
theta the theta vector of explicit parameters; first is tau for the neighbourhood, second is k. Defaults to 100, 10.
type MDS type.
weightmat (optional) a matrix of nonnegative weights
init (optional) initial configuration
ndim number of dimensions of the target space
itmaxi number of iterations
... additional arguments to be passed to the fitting procedure
stressweight weight to be used for the fit measure; defaults to 1
structures a character vector listing the structure indices to use. They always are called "cfoo" with foo being the structure.
strucweight weight to be used for the structures; defaults to 1/number of structures
strucpars a list of parameters for the structuredness indices; each list element corresponds to one index in the order of the appearance in structures
verbose numeric value hat prints information on the fitting process; >2 is extremely verbose
stoptype which weighting to be used in the multi-objective optimization? Either 'additive' (default) or 'multiplicative'.

Value

A list with the components

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

STOPS version of sparsified MDS.

Description

smds with free tau.

Usage

stop_smds(
  dis,
  theta = c(100),
  type = "ratio",
  weightmat = 1 - diag(nrow(dis)),
  init = NULL,
  ndim = 2,
  itmax = 10000,
  ...
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
  "cassociation", "cmonotonicity", "cfactoriality", "ccomplexity", "cfactoriality",
  "cregularity", "chierarchy", "ccomplexity", "cstriatedness", "coutlying",
  "cskinniness", "csparsity", "cstringiness", "cclumpiness", "cinequality"),
  strucweight = rep(1/length(structures), length(structures)),
  strucpars,
  verbose = 0,
  stoptype = c("additive", "multiplicative")
)

Arguments

dis  numerical matrix or dist object of a matrix of proximities
theta the theta vector of explicit parameters; tau for the neighbourhood. Defaults to 100.
type MDS type.
weightmat (optional) a matrix of nonnegative weights
init (optional) initial configuration
ndim number of dimensions of the target space
itmax number of iterations
... additional arguments to be passed to the fitting procedure
stressweight weight to be used for the fit measure; defaults to 1
structures a character vector listing the structure indices to use. They always are called "cfoo" with foo being the structure.
strucweight weight to be used for the structures; defaults to 1/number of structures
stop_spmdade

strucpars a list of parameters for the structuredness indices; each list element corresponds to one index in the order of the appearance in structures
verbose numeric value that prints information on the fitting process; >2 is extremely verbose
stoptype which weighting to be used in the multi-objective optimization? Either 'additive' (default) or 'multiplicative'.

Value

A list with the components

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

Description

Sparsified POST MDDA with free kappa, lambda, rho, tau and epsilon. Phew.

Usage

stop_spmddae(
  dis,
  theta = c(1, 1, 1, 100, 100),
  type = "ratio",
  weightmat = 1 - diag(nrow(dis)),
  init = NULL,
  ndim = 2,
  itmaxi = 10000,
  ...,
  stressweight = 1,
  strucweight = rep(1/length(structures), length(structures)),
  strucpars,
  verbose = 0,
  stoptype = c("additive", "multiplicative")
)
Arguments

- **dis**: numeric matrix or dist object of a matrix of proximities
- **theta**: the theta vector of explicit parameters; the first is kappa (for the fitted distances), the second lambda (for the observed proximities), the third nu (for the weights), the fourth tau (for the neighbourhood), the fifth the epsilon for the geodesic distances. If a scalar or vector shorter than 5 is given it is recycled. Defaults to 1 1 1 100 10.
- **type**: MDS type.
- **weightmat**: (optional) a matrix of nonnegative weights
- **init**: (optional) initial configuration
- **ndim**: number of dimensions of the target space
- **itmaxi**: number of iterations
- **...**: additional arguments to be passed to the fitting procedure
- **stressweight**: weight to be used for the fit measure; defaults to 1
- **structures**: a character vector listing the structure indices to use. They always are called "cfoo" with foo being the structure.
- **strucweight**: weight to be used for the structures; defaults to 1/number of structures
- **strucpars**: a list of parameters for the structuredness indices; each list element corresponds to one index in the order of the appearance in structures
- **verbose**: numeric value that prints information on the fitting process; >2 is extremely verbose
- **stoptype**: which weighting to be used in the multi-objective optimization? Either 'additive' (default) or 'multiplicative'.

Value

A list with the components

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

Sparsified Post MDDA with free kappa, lambda, rho, tau and k. Phew.

**Usage**

```r
stop_spmddak(
  dis,
  theta = c(1, 1, 1, 100, 10),
  type = "ratio",
  weightmat = 1 - diag(nrow(dis)),
  init = NULL,
  ndim = 2,
  itmaxi = 10000,
  ...,
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
  "cassociation", "ccomonotonicity", "cfonctionality", "ccomplexity", "cfaithfulness",
  "cregularity", "cchierarchy", "ccconvexity", "cstriatedness", "coutlying",
  "cskinniness", "csparsity", "cstringiness", "cclumpiness", "cinequality"),
  strucweight = rep(1/length(structures), length(structures)),
  strucpars,
  verbose = 0,
  stoptype = c("additive", "multiplicative")
)
```

**Arguments**

- `dis` numeric matrix or dist object of a matrix of proximities
- `theta` the theta vector of explicit parameters; the first is kappa (for the fitted distances), the second lambda (for the observed proximities), the third nu (for the weights), the fourth tau (for the neighbourhood), the fifth the k for the geodesic distances. If a scalar or vector shorter than 5 is given it is recycled. Defaults to 1 1 1 100 10.
- `type` MDS type.
- `weightmat` (optional) a matrix of nonnegative weights
- `init` (optional) initial configuration
- `ndim` number of dimensions of the target space
- `itmaxi` number of iterations
- `...` additional arguments to be passed to the fitting procedure
- `stressweight` weight to be used for the fit measure; defaults to 1
structures a character vector listing the structure indices to use. They always are called "cfoo" with foo being the structure.

strucweight weight to be used for the structures; defaults to 1/number of structures

strucpars a list of parameters for the structuredness indices; each list element corresponds to one index in the order of the appearance in structures

verbose numeric value that prints information on the fitting process; >2 is extremely verbose

stoptype which weighting to be used in the multi-objective optimization? Either 'additive' (default) or 'multiplicative'.

Value

A list with the components

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

Description

Sparsified power stress with free kappa, lambda, rho and tau.

Usage

stop_spmds(
  dis,
  theta = c(1, 1, 1, 100),
  type = "ratio",
  weightmat = 1 - diag(nrow(dis)),
  init = NULL,
  ndim = 2,
  itmaxi = 10000,
  ...
  stressweight = 1,
  structures = c("cclusteredness", "clinearity", "cdependence", "cmanifoldness",
                   "cassociation", "ccomonotonicity", "cfuctionality", "ccomplexity", "cfaithfulness",
                   "cregularity", "chierarchy", "cconvexity", "cstriatedness", "coutlying",
                   ...)
"cskinniness", "csparsity", "cstringiness", "cclumpiness", "cinequality"),
strucweight = rep(1/length(structures), length(structures)),
strucpars,
verbose = 0,
stoprtype = c("additive", "multiplicative")
)

Arguments

dis numeric matrix or dist object of a matrix of proximities
theta the theta vector of explicit parameters; the first is kappa (for the fitted
distances), the second lambda (for the observed proximities), the third nu (for the weights),
the fourth tau (for the neighbourhood). If a scalar or vector shorter than 4 is
given it is recycled. Defaults to 1 1 1 100.
type MDS type.
weightmat (optional) a matrix of nonnegative weights
init (optional) initial configuration
ndim number of dimensions of the target space
itmaxi number of iterations
... additional arguments to be passed to the fitting procedure
stressweight weight to be used for the fit measure; defaults to 1
structures a character vector listing the structure indices to use. They always are called
"cfoo" with foo being the structure.
strucweight weight to be used for the structures; defaults to 1/number of structures
strucpars a list of parameters for the structuredness indices; each list element corresponds
to one index in the order of the appearance in structures
verbose numeric value hat prints information on the fitting process; >2 is extremely ver-
bose
stoprtype which weighting to be used in the multi-objective optimization? Either 'addi-
tive' (default) or 'multiplicative'.

Value

A list with the components

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

**STOPS version of sstress**

**Description**

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

**Usage**

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

**Arguments**

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

Value
A list with the components

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

summary.stops  S3 summary method for stops

Description
S3 summary method for stops

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

Arguments

- object object of class stops
- ... additional arguments

Value
object of class 'summary.stops'
Swissroll  Swiss roll

Description

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

Usage

data(Swissroll)

Format

A data frame with 150 rows and 4 columns

Details

A data frame with the variables (columns)

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

tgpoptim

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

Description

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

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

Arguments

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

Value

A list with the components (for compatibility with optim)

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

\[
\text{fbana} <- \text{function}(x) \{ \\
x1 <- x[1] \\
x2 <- x[2] \\
100 * (x2 - x1 * x1)^2 + (1 - x1)^2 \\
\}
\]

res1 <- \text{tgpoptim}(\text{c(-1.2,1)}, \text{fbana}, \text{lower}=\text{c(-5,-5)}, \text{upper}=\text{c(5,5)}, \text{acc}=1e-16, \text{itmax}=20)

res1

\[
\text{fwild} <- \text{function}(x) 10*\sin(0.3*x)*\sin(1.3*x^2) + 0.00001*x^4 + 0.2*x+80
\]

\text{plot(fwild, -50, 50, n = 1000, main = "Bayesian GP Optimization minimizing 'wild function'")}

\text{set.seed(210485)}

res2 <- \text{tgpoptim}(50, \text{fwild}, \text{lower}=-50, \text{upper}=50, \text{acc}=1e-16, \text{itmax}=20, \text{model}="btgpllm")

\text{points(res2$par, res2$value, col="red", pch=19)}

res2
Index

* clustering
  stops. 29
* multivariate
  stop_apstress, 33
  stop_bccmds, 34
  stop_clca, 36
  stop_cldae, 37
  stop_cldak, 39
  stop_cmds, 40
  stop_elastic, 42
  stop_isomap1, 43
  stop_isomap2, 45
  stop_lmds, 46
  stop_powerelastic, 48
  stop_powermds, 49
  stop_powersammon, 51
  stop_powerstress, 52
  stop_rpowerstress, 54
  stop_rstress, 55
  stop_sammon, 57
  stop_sammon2, 58
  stop_smacofSphere, 60
  stop_smacofSym, 61
  stop_smddae, 63
  stop_smddak, 64
  stop_smds, 66
  stop_spmddae, 67
  stop_spmddak, 69
  stop_spmds, 70
  stop_sstress, 72
  stops, 29
ace, 15
BankingCrisesDistances, 3
bcp, 31
bootmds, 4, 5
bootmds.stops, 4
c_association, 6
c_clumpiness, 7
c_clusteredness, 7
c_complexity, 9
c_convexity, 10
c_dependence, 10
c_faithfulness, 11
c_functionality, 12
c_hierarchy, 13
c_inequality, 13
c_linearity, 14
c_manifoldness, 15
c_mine, 16
c_nonmonotonicity, 16
c_outlying, 17
c_regularity, 18
c_shepardness, 19
c_skinniness, 20
c_sparcity, 20
c_striatedness, 21
c_stringiness, 22
c_validity, 13
cma_es, 31
cobyla, 31
coeef.stops, 5
cordillera, 8, 19
crs2lm, 31
direct, 31
dopt.gp, 31
EGO.nsteps, 31
hclust, 13
hjk, 31
isres, 31
jackmids, 23
jackmids.stops, 22
knn_dist, 23
ljoptim, 24
mine, 6, 9, 12, 17
mlsl, 31
neldermead, 31
optics, 8
optim, 25, 31, 75
Pendigits500, 25
plot.stops, 26
print.stops, 27
print.summary.stops, 27
psoptim, 31
residuals.stops, 28
sbplx, 31
scagnostics, 7, 10, 18, 20–22
sim2diss, 4
stogo, 31
stop_apstress, 33
stop_bcmds, 34
stop_clda, 36
stop_cldae, 37
stop_cldak, 39
stop_cmdscale, 40
stop_elastic, 42
stop_isomap1, 43
stop_isomap2, 45
stop_lmds, 46
stop_powerelastic, 48
stop_powermds, 49
stop_powersammon, 51
stop_powerstress, 52
stop_rpowerstress, 54
stop_sstress, 55
stop_sammon, 57
stop_sammon2, 58
stop_smacofSphere, 60
stop_smacofSym, 61
stop_smddae, 63
stop_smddak, 64
stop_smds, 66
stop_spmmdae, 67
stop_spmmdak, 69
stop_spmmds, 70
stop_sssstress, 72
stoploss, 28
summary.stops, 73
Swissroll, 74
tgoptim, 74