Package 'resemble'

August 29, 2016

Type Package

Title Regression and Similarity Evaluation for Memory-Based Learning in Spectral Chemometrics

Version 1.2.2

Date 2016-03-04

Author Leonardo Ramirez-Lopez [aut, cre], Antoine Stevens [aut]

Maintainer Leonardo Ramirez-Lopez <ramirez.lopez.leo@gmail.com>

BugReports https://github.com/l-ramirez-lopez/resemble/issues

Description Implementation of functions for spectral similarity/dissimilarity analysis and memory-based learning (MBL) for non-linear modeling in complex spectral datasets. In chemometrics MBL is also known as local modeling.

License GPL (>= 3)

URL http://l-ramirez-lopez.github.io/resemble/

Depends R (>= 3.2.2)

Imports foreach, iterators, Rcpp (>= 0.12.0)

Suggests prospectr, parallel, doParallel

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

Repository CRAN

RoxygenNote 5.0.1

Date/Publication 2016-03-04 00:11:00

R topics documented:

`resemble-package` *Overview of the functions in the resemble package*

Description

This is the version `1.2` (`'alma-de-coco'`) of the package. It implements a number of `R` functions useful for modeling complex visible and infrared spectra (VIS-IR). The package includes functions for projecting spectral data onto orthogonal spaces, computing spectral dissimilarity matrices, removing irrelevant spectra from a reference set, and modeling spectral data using memory-based learning.

Details

The functions available for projecting the spectra are:

- `orthoProjection`
- `pcProjection`
- `plsProjection`
- `predict.orthoProjection`

The functions available for computing similarity/dissimilarity matrices are:

- `fDiss`
- `corDiss`
- `sid`
- `orthoDiss`

The functions available for evaluating similarity/dissimilarity matrices are:

- `simEval`

The functions available for removing irrelevant spectra from a reference set are:

- `neigCleaning`

The functions available for modeling spectral data using memory-based learning are:

- mblControl
- mbl

Other supplementary functions are:

- print.localOrthoDiss
- print.mbl
- plot.mbl
- plot.orthoProjection
- print.orthoProjection

Author(s)

Leonardo Ramirez-Lopez <ramirez.lopez.leo@gmail.com> & Antoine Stevens

---

corDiss *Correlation and moving correlation dissimilarity measurements (corDiss)*

Description

Computes correlation and moving correlation dissimilarity matrices.

Usage

```
corDiss(Xr, X2 = NULL, ws = NULL,
        center = TRUE, scaled = TRUE)
```

Arguments

Xr	a matrix (or data.frame) containing the (reference) data.
X2	an optional matrix (or data.frame) containing data of a second set of observations (samples).
ws	for moving correlation dissimilarity, an odd integer value which specifies the window size. If ws = NULL, then the window size will be equal to the number of variables (columns), i.e. instead moving correlation, the normal correlation will be used. See details.
center	a logical indicating if the spectral data Xr (and X2 if specified) must be centered. If X2 is specified the data is scaled on the basis of $Xr \cup X2$.
scaled	a logical indicating if Xr (and X2 if specified) must be scaled. If X2 is specified the data is scaled on the basis of $Xr \cup X2$.

Details

The correlation dissimilarity cd between two obsvervations x_i and x_j is computed as follows:

$$cd(x_i, x_j) = \frac{1}{2}(1 - cor(x_i, x_j))$$

The avobe formlula is used when `ws = NULL`. On the other hand (when `ws != NULL`) the moving correlation dissimilarity mcd between two obsvervations x_i and x_j is computed as follows:

$$mcd(x_i, x_j) = \frac{1}{2ws} \sum_{k=1}^{p-ws} (1 - cor(x_{i,(k:k+ws)}, x_{j,(k:k+ws)}))$$

where ws represents a given window size which rolls sequantially fom 1 up to $p - ws$ and p is the number of variables of the observations. The function does not accept input data containing missing values.

Value

a `matrix` of the computed dissimilarities.

Author(s)

Antoine Stevens and Leonardo Ramirez-Lopez

Examples

```
## Not run:
require(prospectr)
data(NIRsoil)

Xu <- NIRsoil$spc[!as.logical(NIRsoil$train),]
Xr <- NIRsoil$spc[as.logical(NIRsoil$train),]

corDiss(Xr = Xr)

corDiss(Xr = Xr, X2 = Xu)

corDiss(Xr = Xr, ws = 41)

corDiss(Xr = Xr, X2 = Xu, ws = 41)

## End(Not run)
```

`fDiss` *Euclidean, Mahalanobis and cosine dissimilarity measurements*

Description

This function is used to compute the dissimilarity between observations based on Euclidean or Mahalanobis distance measures or on cosine dissimilarity measures (a.k.a spectral angle mapper).

Usage

```
fDiss(Xr, X2 = NULL, method = "euclid",
      center = TRUE, scaled = TRUE)
```

Arguments

`Xr`	a `matrix` (or `data.frame`) containing the (reference) data.
`X2`	an optional `matrix` (or `data.frame`) containing data of a second set of observations(samples).
`method`	the method for computing the dissimilarity matrix. Options are `"euclid"` (Euclidean distance), `"mahalanobis"` (Mahalanobis distance) and `"cosine"` (cosine distance, a.k.a spectral angle mapper).
`center`	a logical indicating if the spectral data `Xr` (and `X2` if specified) must be centered. If X2 is specified the data is scaled on the basis of $Xr \cup X2$.
`scaled`	a logical indicating if `Xr` (and `X2` if specified) must be scaled. If `X2` is specified the data is scaled on the basis of $Xr \cup X2$.

Details

In the case of both the Euclidean and Mahalanobis distances, the dissimilarity matrix D between between samples in a given matrix X is computed as follows:

$$D(x_i, x_j) = \sqrt{(x_i - x_j)M^{-1}(x_i - x_j)^{\mathrm{T}}}$$

where M is the identity matrix in the case of the Euclidean distance and the variance-covariance matrix of M in the case of the Mahalanobis distance. The Mahalanobis distance can also be viewed as the Euclidean distance after applying a linear transformation of the original variables. Such a linear transformation is carried by using a factorization of the inverse covariance matrix as $M^{-1} = W^{\mathrm{T}}W$, where M is merely the square root of M^{-1} which can be found by using a singular value decomposition. Note that when attempting to compute the Mahalanobis distance on a dataset with highly correlated variables (i.e. spectral variables) the variance-covariance matrix may result in a singular matrix which cannot be inverted and therefore the distance cannot be computed. This is also the case when the number of samples in the dataset is smaller than the number of variables. For the computation of the Mahalanobis distance, the mentioned method is used. On the other hand the cosine dissimilarity S between two obsvervations x_i and x_j is computed as follows:

$$S(x_i, x_j) = cos^{-1} \frac{\sum_{k=1}^{p} x_{i,k} x_{j,k}}{\sqrt{\sum_{k=1}^{p} x_{i,k}^2} \sqrt{\sum_{k=1}^{p} x_{j,k}^2}}$$

where p is the number of variables of the observations. The function does not accept input data containing missing values.

Value

a `matrix` of the computed dissimilarities.

Author(s)

Leonardo Ramirez-Lopez and Antoine Stevens

Examples

```
require(prospectr)

data(NIRsoil)

Xu <- NIRsoil$spc[!as.logical(NIRsoil$train),]
Xr <- NIRsoil$spc[as.logical(NIRsoil$train),]

# Euclidean distances between all the samples in Xr
ed <- fDiss(Xr = Xr, method = "euclid",
            center = TRUE, scaled = TRUE)

# Euclidean distances between samples in Xr and samples in Xu
ed.xr.xu <- fDiss(Xr = Xr, X2 = Xu, method = "euclid",
                  center = TRUE, scaled = TRUE)

# Mahalanobis distance computed on the first 20 spectral variables
md.xr.xu <- fDiss(Xr = Xr[,1:20], X2 = Xu[,1:20],
                 method = "mahalanobis",
                 center = TRUE, scaled = TRUE)

# Cosine dissimilarity matrix
cdiss.xr.xu <- fDiss(Xr = Xr, X2 = Xu,
                     method = "cosine",
                     center = TRUE, scaled = TRUE)
```

---

getPredictions *Extract predictions from an object of class* mbl

---

Description

Extract predictions from an object of class mbl

Usage

```
getPredictions(object)
```

Arguments

object	an object of class mbl as returned by mbl

Value

a data.frame of predicted values according to either k or k.dist

Author(s)

Leonardo Ramirez-Lopez and Antoine Stevens

See Also

mbl

Examples

```
## Not run:
require(prospectr)

data(NIRsoil)

Xu <- NIRsoil$spc[!as.logical(NIRsoil$train),]
Yu <- NIRsoil$CEC[!as.logical(NIRsoil$train)]
Yr <- NIRsoil$CEC[as.logical(NIRsoil$train)]
Xr <- NIRsoil$spc[as.logical(NIRsoil$train),]

Xu <- Xu[!is.na(Yu),]
Yu <- Yu[!is.na(Yu)]

Xr <- Xr[!is.na(Yr),]
Yr <- Yr[!is.na(Yr)]

ctrl <- mblControl(sm = "pls",
                   pcSelection = list("opc", 40),
                   valMethod = c("NNv"),
                   scaled = TRUE, center = TRUE)

ex1 <- mbl(Yr = Yr, Xr = Xr, Yu = NULL, Xu = Xu,
           mblCtrl = ctrl,
           distUsage = "predictors",
           k = seq(30, 150, 15),
           method = "wapls1",
           plsC = c(7, 20))

getPredictions(ex1)

## End(Not run)
```

---

mbl *A function for memory-based learning (mbl)*

Description

This function is implemented for memory-based learning (a.k.a. instance-based learning or local regression) which is a non-linear lazy learning approach for predicting a given response variable from a set of (spectral) predictor variables. For each sample in an prediction set a specific local

regression is carried out based on a subset of similar samples (nearest neighbours) selected from a reference set. The local model is then used to predict the response value of the target (prediction) sample. Therefore this function does not yield a global regression model.

Usage

```
mbl(Yr, Xr, Yu = NULL, Xu,
    mblCtrl = mblControl(),
    dissimilarityM,
    group = NULL,
    dissUsage = "predictors",
    k, k.diss, k.range,
    method,
    pls.c, pls.max.iter = 1, pls.tol = 1e-6,
    noise.v = 0.001,
    ...)
```

Arguments

Yr	a numeric vector containing the values of the response variable corresponding to the reference data
Xr	input matrix (or data.frame) of predictor variables of the reference data (observations in rows and variables in columns).
Yu	an optional numeric vector containing the values of the response variable corresponding to the data to be predicted
Xu	input matrix (or data.frame) of predictor variables of the data to be predicted (observations in rows and variables in columns).
mblCtrl	a list (created with the mblControl function) which contains some parameters that control the some aspects of the mbl function. See the mblControl function for more details.
dissimilarityM	(optional) a dissimilarity matrix. This argument can be used in case a user-defined dissimilarity matrix is preferred over the automatic dissimilarity matrix computation specified in the sm argument of the mblControl function. When dissUsage = "predictors", dissimilarityM must be a square symmetric dissimilarity matrix (derived from a matrix of the form rbind(Xr, Xu)) for which the diagonal values are zeros (since the dissimilarity between an object and itself must be 0). On the other hand if dissUsage is set to either "weights" or "none", dissimilarityM must be a matrix representing the dissimilarity of each element in Xu to each element in Xr. The number of columns of the object correspondent to dissimilarityM must be equal to the number of rows in Xu and the number of rows equal to the number of rows in Xr. If both dissimilarityM and sm are specified, only the dissimilarityM argument will be taken into account.
group	an optional factor (or vector that can be coerced to a factor by as.factor) to be taken into account for internal validations. The length of the vector must be equal to nrow(Xr), giving the identifier of related observations (e.g. spectra collected from the same batch of measurements, from the same sample, from

samples with very similar origin, etc). When one observation is selected for cross validation, all observations of the same group are removed together and assigned to validation. See details.

`dissUsage` specifies how the dissimilarity information shall be used. The possible options are: `"predictors"`, `"weights"` and `"none"` (see details below). Default is "predictors".

`k` a numeric (integer) `vector` containing the sequence of k nearest neighbours to be tested. Either `k` or `k.diss` must be specified. Numbers with decimal values will be coerced to their next higher integer values. This `vector` will be automatically sorted into ascending order.

`k.diss` a `vector` containing the sequence of dissimilarity thresholds to be tested. When the dissimilarity between a sample in `Xr` and a sample `Xu` is below the given threshold, the sample in sample in `Xr` is treated as a neighbour of the sample in `Xu`, otherwise it is ignored. These thresholds depend on the corresponding dissimilarity measure specified in `sm`. Either `k` or `k.diss` must be specified.

`k.range` a `vector` of length 2 which specifies the minimum (first value) and the maximum (second value) number of neighbours allowed when the `k.diss` argument is used.

`method` a character string indicating the method to be used at each local multivariate regression. Options are: `"pls"`, `"wapls1"` and `"gpr"` (see details below). Note: `"wapls2"` from the previos version of the package is no longer available/supported.

`pls.c` the number of pls components to be used in the regressions if either `"pls"` or `"wapls1"` is used. When `"pls"` is used, this argument must be a single numerical value. When `"wapls1"` is used, this argument must be a `vector` of length 2 indicating the minimum (first value) and the maximum (second value) number of pls components used for the regressions (see details below).

`pls.max.iter` maximum number of iterations for the partial least squares methods.

`pls.tol` limit for convergence in the partial orthogonal scores partial least squares regressions using the nipals algorithm. Default is 1e-6.

`noise.v` a value indicating the variance of the noise for Gaussian process regression. Default is 0.001.

`...` additional arguments to be passed to other functions.

Details

By using the `group` argument one can specify observations (spectra) groups of samples that have something in common e.g. spectra collected from the same batch of measurements, from the same sample, from samples with very similar origin, etc) which could produce biased cross-validation results due to pseudo-replication. This argument allows to select calibration points that are independent from the validation ones in cross-validation. In this regard, when `valMethod = "loc_crossval"` (used in `mblControl` function), then the `p` argument refer to the percentage of groups of samples (rather than single samples) to be retained in each resampling iteration at each local segment. The `dissUsage` argument is used to specifiy whether the dissimilarity information must be used within the local regressions and (if so), how. When `dissUsage = "predictors"` the local (square symmetric) dissimilarity matrix corresponding the selected neighbourhood is used as source of

additional predictors (i.e the columns of this local matrix are treated as predictor variables). In some cases this may result in an improvement of the prediction performance (Ramirez-Lopez et al., 2013a). If `dissUsage = "weights"`, the neighbours of the query point (xu_j) are weighted according to their dissimilarity (e.g. distance) to xu_j prior carrying out each local regression. The following tricubic function (Cleveland and Delvin, 1988; Naes et al., 1990) is used for computing the final weights based on the measured dissimilarities:

$$W_j = (1 - v^3)^3$$

where if $xr_i \in$ neighbours of xu_j:

$$v_j(xu_j) = d(xr_i, xu_j)$$

otherwise:

$$v_j(xu_j) = 0$$

In the above formulas $d(xr_i, xu_j)$ represents the dissimilarity between the query point and each object in Xr. When `dissUsage = "none"` is chosen the dissimilarity information is not used. The possible options for performing regressions at each local segment implemented in the `mbl` function are described as follows:

- Partial least squares (`"pls"`): It uses the orthogonal scores (non-linear iterative partial least squares, nipals) algorithm. The only parameter which needs to be optimized is the number of pls components. This can be done by cross-validation at each local segment.
- Weighted average pls (`"wapls1"`): It uses multiple models generated by multiple pls components (i.e. between a minimum and a maximum number of pls components). At each local partition the final predicted value is a weighted average of all the predicted values generated by the multiple pls models. The weight for each component is calculated as follows:

  $$w_j = \frac{1}{s_{1:j} \times g_j}$$

  where $s_{1:j}$ is the root mean square of the spectral residuals of the unknown (or target) sample when a total of j pls components are used and g_j is the root mean square of the regression coefficients corresponding to the jth pls component (see Shenk et al., 1997 for more details). `"wapls1"` is not compatible with `valMethod = "loc_crossval"` since the weights are computed based on the sample to be predicted at each local iteration.
- Gaussian process with dot product covariance (`"gpr"`): Gaussian process regression is a probabilistic and non-parametric Bayesian approach. It is commonly described as a collection of random variables which have a joint Gaussian distribution and it is characterized by both a mean and a covariance function (Williams and Rasmussen, 1996). The covariance function used in the implemented method is the dot product, which inplies that there are no parameters to be optimized for the computation of the covariance. Here, the process for predicting the response variable of a new sample (y_{new}) from its predictor variables (x_{new}) is carried out first by computing a prediction vector (A). It is derived from a set of reference spectra (X) and their respective response vector (Y) as follows:

  $$A = (XX^{\mathrm{T}} + \sigma^2 I)^{-1} Y$$

  where σ^2 denotes the variance of the noise and I the identity matrix (with dimensions equal to the number of observations in X). The prediction of y_{new} is then carried out by:

  $$y_{new} = (x_{new} x_{new}^{\mathrm{T}}) A$$

The loop used to iterate over the Xu samples in mbl uses the %dopar% operator of the foreach package, which can be used to parallelize this internal loop. The last example given in the mbl function ilustrates how to parallelize the mbl function. Note that the computational cost depends largely on the way on which the arguments of the function are set. For big datasets, it is recommended to carefully select the values of the parameters to test (e.g. validation method, regression method, dissimilarity information usage, dissimilarity method).

Value

a list of class mbl with the following components (sorted by either k or k.diss according to the case):

- call: the call used.
- cntrlParam: the list with the control parameters used. If one or more control parameters were reset automatically, then a list containing a list with the initial control parameters specified and a list with the parameters which were finally used.
- dissimilarities: a list with the method used to obtain the dissimilarity matrices and the dissimilarity matrices corresponding to $D(Xr, Xu)$ and $D(Xr, Xr)$ if dissUsage = "predictors". This object is returned only if the returnDiss argument in the mblCtrl list was set to TRUE in the the call used.
- totalSamplesPredicted the total number of samples predicted.
- pcAnalysis: a list containing the results of the principal component analysis. The first two objects (scores_Xr and scores_Xu) are the scores of the Xr and Xu matrices. It also contains the number of principal components used (n.componentsUsed) and another object which is a vector containing the standardized Mahalanobis dissimilarities (also called GH, Global H distance) between each sample in Xu and the centre of Xr.
- components: a list containing either the number of principal components or partial least squares components used for the computation of the orthogonal dissimilarities. This object is only returned if the dissimilarity meausre specified in mblCtrl is any of the following options: 'pc', 'loc.pc', "pls", 'loc.pls'. If any of the local orthogonal dissimilarities was used ('loc.pc' or "pls") a data.frame is also returned in his list. This object is equivalent to the loc.n.components object returned by the orthoDiss function. It specifies the number of local components (either principal components or partial least squares components) used for computing the dissimilarity between each query sample and its neighbour samples, as returned by the orthoDiss function.
- nnValStats: a data frame containing the statistics of the nearest neighbour cross-validation for each either k or k.diss depending on the arguments specified in the call. It is returned only if 'NNv' or 'both' were selected as validation method
- localCrossValStats: a data frame containing the statistics of the local leave-group-out cross validation for each either k or k.diss depending on the arguments specified in the call. It is returned only if 'local_crossval' or 'both' were selected as validation method
- YuPredictionStats: a data frame containing the statistics of the cross-validation of the prediction of Yu for each either k or k.diss depending on the arguments specified in the call. It is returned only if Yu was provided.
- results: a list of data frames which contains the results of the predictions for each either k or k.diss. Each data.frame contains the following columns:

- o.index: The index of the sample predicted in the input matrix
- k.diss: This column is only ouput if the k.diss argument is used. It indicates the corresponding dissimilarity threshold for selecting the neighbors used to predict a given sample.
- distance: This column is only ouput if the k.diss argument is used. It is a logical that indicates whether the neighbors selected by the given dissimilarity threshold were outside the boundaries specified in the k.range argument. In that case the number of neighbors used is coerced to on of the boundaries.
- k.org: This column is only ouput if the k.diss argument is used. It indicates the number of neighbors that are retained when the given dissimilarity threshold is used.
- pls.comp: This column is only ouput if pls regression was used. It indicates the final number of pls components used. If no optimization was set, it retrieves the original pls components specified in the pls.c argument.
- min.pls: This column is only ouput if wapls1 regression was used. It indicates the final number of minimum pls components used. If no optimization was set, it retrieves the original minimum pls components specified in the pls.c argument.
- max.pls: This column is only ouput if wapls1 regression was used. It indicates the final number of maximum pls components used. If no optimization was set, it retrieves the original maximum pls components specified in the pls.c argument.
- yu.obs: This column is only ouput if the Yu argument is used. It indicates the input values given in Yu (the response variable corresponding to the data to be predicted).
- pred: The predicted values
- yr.min.obs: The minimum reference value (of the response variable) in the neighborhood.
- yr.max.obs: The maximum reference value (of the response variable) in the neighborhood.
- index.nearest.in.ref The index in Xr of the nearest neighbor.
- y.nearest: The reference value (of the response variable) of the nearest neighbor in Xr.
- y.nearest.pred: This column is only ouput if the validation method (selected with the mblControl function) is equal to 'NNv'. It represents the predicted value of the nearest neighbor sample in Xr using the neighborhood of the predicted sample in Xu.
- loc.rmse.cv: This column is only ouput if the validation method (selected with the mblControl function) is equal to 'loc_crossval'. It represents the cross validation RMSE value computed in for the neighborhood of the sample of the predicted sample in Xu.
- loc.st.rmse.cv: This column is only ouput if the validation method (selected with the mblControl function) is equal to 'loc_crossval'. It represents the cross validation standardized RMSE value computed in for the neighborhood of the sample of the predicted sample in Xu.
- dist.nearest: The distance to the nearest neighbor.
- dist.k.farthest: The distance to the farthest neighbor selected.

When the k.diss argument is used, the printed results show a table with a column named 'p.bounded'. It represents the percentage of samples for which the neighbors selected by the given dissimilarity threshold were outside the boundaries specified in the k.range argument.

Author(s)

Leonardo Ramirez-Lopez and Antoine Stevens

References

Cleveland, W. S., and Devlin, S. J. 1988. Locally weighted regression: an approach to regression analysis by local fitting. Journal of the American Statistical Association, 83, 596-610.

Fernandez Pierna, J.A., Dardenne, P. 2008. Soil parameter quantification by NIRS as a Chemometric challenge at "Chimiomitrie 2006". Chemometrics and Intelligent Laboratory Systems 91, 94-98

Naes, T., Isaksson, T., Kowalski, B. 1990. Locally weighted regression and scatter correction for near-infrared reflectance data. Analytical Chemistry 62, 664-673.

Ramirez-Lopez, L., Behrens, T., Schmidt, K., Stevens, A., Dematte, J.A.M., Scholten, T. 2013a. The spectrum-based learner: A new local approach for modeling soil vis-NIR spectra of complex datasets. Geoderma 195-196, 268-279.

Ramirez-Lopez, L., Behrens, T., Schmidt, K., Viscarra Rossel, R., Dematte, J. A. M., Scholten, T. 2013b. Distance and similarity-search metrics for use with soil vis-NIR spectra. Geoderma 199, 43-53.

Rasmussen, C.E., Williams, C.K. Gaussian Processes for Machine Learning. Massachusetts Institute of Technology: MIT-Press, 2006.

Shenk, J., Westerhaus, M., and Berzaghi, P. 1997. Investigation of a LOCAL calibration procedure for near infrared instruments. Journal of Near Infrared Spectroscopy, 5, 223-232.

See Also

mblControl, fDiss, corDiss, sid, orthoDiss, neigCleaning

Examples

```
## Not run:
require(prospectr)

data(NIRsoil)

# Filter the data using the Savitzky and Golay smoothing filter with
# a window size of 11 spectral variables and a polynomial order of 3
# (no differentiation).
sg <- savitzkyGolay(NIRsoil$spc, p = 3, w = 11, m = 0)

# Replace the original spectra with the filtered ones
NIRsoil$spc <- sg

Xu <- NIRsoil$spc[!as.logical(NIRsoil$train),]
Yu <- NIRsoil$CEC[!as.logical(NIRsoil$train)]

Yr <- NIRsoil$CEC[as.logical(NIRsoil$train)]
Xr <- NIRsoil$spc[as.logical(NIRsoil$train),]
```

```
Xu <- Xu[!is.na(Yu),]
Xr <- Xr[!is.na(Yr),]

Yu <- Yu[!is.na(Yu)]
Yr <- Yr[!is.na(Yr)]

# Example 1
# A mbl implemented in Ramirez-Lopez et al. (2013,
# the spectrum-based learner)
# Example 1.1
# An exmaple where Yu is supposed to be unknown, but the Xu
# (spectral variables) are known
ctrl1 <- mblControl(sm = "pc", pcSelection = list("opc", 40),
                    valMethod = "NNv",
                    scaled = FALSE, center = TRUE)

sbl.u <- mbl(Yr = Yr, Xr = Xr, Yu = NULL, Xu = Xu,
             mblCtrl = ctrl1,
             dissUsage = "predictors",
             k = seq(40, 150, by = 10),
             method = "gpr")
sbl.u
plot(sbl.u)



# Example 1.2
# If Yu is actually known...
sbl.u2 <- mbl(Yr = Yr, Xr = Xr, Yu = Yu, Xu = Xu,
              mblCtrl = ctrl1,
              dissUsage = "predictors",
              k = seq(40, 150, by = 10),
              method = "gpr")
sbl.u2

# Example 1.3
# A variation of the spectrum-based learner implemented in
# Ramirez-Lopez et al. (2013) where the dissimilarity matrices are
# recomputed based on partial least squares scores
ctrl_1.3 <- mblControl(sm = "pls", pcSelection = list("opc", 40),
                       valMethod = "NNv",
                       scaled = FALSE, center = TRUE)

sbl_1.3 <- mbl(Yr = Yr, Xr = Xr, Yu = Yu, Xu = Xu,
               mblCtrl = ctrl_1.3,
               dissUsage = "predictors",
               k = seq(40, 150, by = 10),
               method = "gpr")
sbl_1.3

# Example 2
# A mbl similar to the ones implemented in
# Ramirez-Lopez et al. (2013)
```

```
# and Fernandez Pierna and Dardenne (2008)
ctrl.mbl <- mblControl(sm = "cor",
                       pcSelection = list("cumvar", 0.999),
                       valMethod = "NNv",
                       scaled = FALSE, center = TRUE)

local.mbl <- mbl(Yr = Yr, Xr = Xr, Yu = Yu, Xu = Xu,
                 mblCtrl = ctrl.mbl,
                 dissUsage = "none",
                 k = seq(40, 150, by = 10),
                 pls.c = c(5, 15),
                 method = "wapls1")
local.mbl

# Example 3
# A variation of the previous example (using the optimized pc
# dissmilarity matrix) using the control list of the example 1

local.mbl2 <- mbl(Yr = Yr, Xr = Xr, Yu = Yu, Xu = Xu,
                  mblCtrl = ctrl1,
                  dissUsage = "none",
                  k = seq(40, 150, by = 10),
                  pls.c = c(5, 15),
                  method = "wapls1")
local.mbl2

# Example 4
# Using the function with user-defined dissimilarities:
# Examples 4.1 - 4.2: Compute a square symetric matrix of
# dissimilarities between
# all the elements in Xr and Xu (dissimilarities will be used as
# additional predictor variables later in the mbl function)

# Examples 4.3 - 4.4: Derive a dissimilarity value of each element
# in Xu to each element in Xr (in this case dissimilarities will
# not be used as additional predictor variables later in the
# mbl function)

# Example 4.1
# the manhattan distance
manhattanD <- dist(rbind(Xr, Xu), method = "manhattan")
manhattanD <- as.matrix(manhattanD)

ctrl.udd <- mblControl(sm = "none",
                       pcSelection = list("cumvar", 0.999),
                       valMethod = c("NNv", "loc_crossval"),
                       resampling = 10, p = 0.75,
                       scaled = FALSE, center = TRUE)

mbl.udd1 <- mbl(Yr = Yr, Xr = Xr, Yu = Yu, Xu = Xu,
                mblCtrl = ctrl.udd,
                dissimilarityM = manhattanD,
                dissUsage = "predictors",
```

```
                k = seq(40, 150, by = 10),
                method = "gpr")
mbl.udd1

#Example 4.2
# first derivative spectra
Xr.der.sp <- t(diff(t(rbind(Xr, Xu)), lag = 7, differences = 1))
Xu.der.sp <- t(diff(t(Xu), lag = 7, differences = 1))

# The principal components dissimilarity on the derivative spectra
der.ortho <- orthoDiss(Xr = Xr.der.sp, X2 = Xu.der.sp,
                       Yr = Yr,
                       pcSelection = list("opc", 40),
                       method = "pls",
                       center = FALSE, scale = FALSE)

der.ortho.diss <- der.ortho$dissimilarity

# mbl applied to the absorbance spectra
mbl.udd2 <- mbl(Yr = Yr, Xr = Xr, Yu = Yu, Xu = Xu,
                mblCtrl = ctrl.udd,
                dissimilarityM = der.ortho.diss,
                dissUsage = "none",
                k = seq(40, 150, by = 10),
                method = "gpr")

#Example 4.3
# first derivative spectra
der.Xr <- t(diff(t(Xr), lag = 1, differences = 1))
der.Xu <- t(diff(t(Xu), lag = 1, differences = 1))
# the sid on the derivative spectra
der.sid <- sid(Xr = der.Xr, X2 = der.Xu, mode = "density",
               center = TRUE, scaled = FALSE)
der.sid <- der.sid$sid

mbl.udd3 <- mbl(Yr = Yr, Xr = Xr, Yu = Yu, Xu = Xu,
                mblCtrl = ctrl.udd,
                dissimilarityM = der.sid,
                dissUsage = "none",
                k = seq(40, 150, by = 10),
                method = "gpr")
mbl.udd3

# Example 5
# For running the mbl function in parallel
n.cores <- detectCores() - 1
if(n.cores == 0) n.cores <- 1

# Set the number of cores according to the OS
if (.Platform$OS.type == "windows") {
  require(doParallel)
  clust <- makeCluster(n.cores)
  registerDoParallel(clust)
```

```
}else{
  require(doSNOW)
  clust <- makeCluster(n.cores, type = "SOCK")
  registerDoSNOW(clust)
  ncores <- getDoParWorkers()
}

ctrl <- mblControl(sm = "pc", pcSelection = list("opc", 40),
                   valMethod = "NNv",
                   scaled = FALSE, center = TRUE)

mbl.p <- mbl(Yr = Yr, Xr = Xr, Yu = Yu, Xu = Xu,
             mblCtrl = ctrl,
             dissUsage = "none",
             k = seq(40, 150, by = 10),
             method = "gpr")
registerDoSEQ()
try(stopCluster(clust))
mbl.p

## End(Not run)
```

`mblControl` *A function that controls some aspects of the memory-based learning process in the* `mbl` *function*

Description

This function is used to specify various aspects in the memory-based learning process in the `mbl` function

Usage

```
mblControl(sm = "pc",
           pcSelection = list("opc", 40),
           pcMethod = "svd",
           ws = if(sm == "movcor") 41,
           k0,
           returnDiss = FALSE,
           center = TRUE,
           scaled = TRUE,
           valMethod = c("NNv", "loc_crossval"),
           localOptimization = TRUE,
           resampling = 10,
           p = 0.75,
           range.pred.lim = TRUE,
           progress = TRUE,
           cores = 1,
           allowParallel = TRUE)
```

Arguments

`sm` a character string indicating the spectral dissimilarity metric to be used in the selection of the nearest neighbours of each observation for which a prediction is required (see `mbl`). Options are:

- `"euclid"`: Euclidean dissimilarity.
- `"cosine"`: Cosine dissimilarity.
- `"sidF"`: Spectral information divergence computed on the spectral variables.
- `"sidD"`: Spectral information divergence computed on the density distributions of the spectra.
- `"cor"`: Correlation dissimilarity.
- `"movcor"`: Moving window correlation dissimilarity.
- `"pc"`: Principal components dissimilarity: Mahalanobis dissimilarity computed on the principal components space.
- `"loc.pc"`: Dissimilarity estimation based on local principal components.
- `"pls"`: Partial least squares dissimilarity: Mahalanobis dissimilarity computed on the partial least squares space.
- `"loc.pls"` Dissimilarity estimation based on local partial least squares.

The `"pc"` spectral dissimilarity metric is the default. If the `"sidD"` is chosen, the default parameters of the `sid` function are used however they cab be modified by specifying them as additional arguments in the `mbl` function.

This argument can also be set to `"none"`, in such a case, a dissimilarity matrix must be specified in the `dissimilarityM` argument of the `mbl` function.

`pcSelection` a list which specifies the method to be used for identifying the number of principal components to be retained for computing the Mahalanobis dissimilarity of each sample in `sm = "Xu"` to the centre of `sm = "Xr"`. It also specifies the number of components in any of the following cases: `sm = "pc"`, `sm = "loc.pc"`, `sm = "pls"` and `sm = "loc.pls"`. This list must contain two objects in the following order:

- `method`:the method for selecting the number of components. Possible options are: `"opc"` (optimized pc selection based on Ramirez-Lopez et al. (2013a, 2013b). See the `orthoProjection` function for more details; `"cumvar"` (for selecting the number of principal components based on a given cumulative amount of explained variance); `"var"` (for selecting the number of principal components based on a given amount of explained variance); and `"manual"` (for specifying manually the desired number of principal components)
- `value`:a numerical value that complements the selected method. If `"opc"` is chosen, it must be a value indicating the maximal number of principal components to be tested (see Ramirez-Lopez et al., 2013a, 2013b). If `"cumvar"` is chosen, it must be a value (higher than 0 and lower than 1) indicating the maximum amount of cumulative variance that the retained components should explain. If `"var"` is chosen, it must be a value (higher than 0 and lower than 1) indicating that components that explain (individually) a variance lower than this threshold must be excluded. If `"manual"` is chosen, it

must be a value specifying the desired number of principal components to retain.

The default method for the `pcSelection` argument is `"opc"` and the maximal number of principal components to be tested is set to 40. Optionally, the `pcSelection` argument admits `"opc"` or `"cumvar"` or `"var"` or `"manual"` as a single character string. In such a case the default for `"value"` when either `"opc"` or `"manual"` are used is 40. When `"cumvar"` is used the default `"value"` is set to 0.99 and when `"var"` is used the default `"value"` is set to 0.01.

`pcMethod` a character string indicating the principal component analysis algorithm to be used. Options are: `"svd"` (default) and `"nipals"`. See `orthoDiss`.

`ws` an odd integer value which specifies the window size when the moving window correlation dissimilarity is used (i.e `sm = "movcor"`). The default is 41.

`k0` if any of the local dissimilarity methods is used (i.e. either `sm = "loc.pc"` or `sm = "loc.pls"`) a numeric integer value. This argument controls the number of initial neighbours($k0$) to retain in order to compute the local principal components (at each neighbourhood).

`returnDiss` a logical indicating if the dissimilarity matrices must be returned.

`center` a logical indicating whether or not the predictor variables must be centered at each local segment (before regression).

`scaled` a logical indicating whether or not the predictor variables must be scaled at each local segment (before regression).

`valMethod` a character vector which indicates the (internal) validation method(s) to be used for assessing the global performance of the local models. Possible options are: `"NNv"` and `"loc_crossval"`. Alternatively `"none"` can be used when corss-validation is not required (see details below).

`localOptimization`
a logical. If `valMethod = "loc_crossval"`, it optmizes the parameters of the local pls models (i.e. pls factors for `pls` and minimum and maximum pls factors for `wapls1`).

`resampling` a value indicating the number of resampling iterations at each local segment when `"loc_crossval"` is selected in the `valMethod` argument. Default is 10.

`p` a value indicating the percentage of samples to be retained in each resampling iteration at each local segment when `"loc_crossval"` is selected in the `valMethod` argument. Default is 0.75 (i.e. 75 "%")

`range.pred.lim` a logical value. It indicates whether the prediction limits at each local regression are determined by the range of the response variable values employed at each local regression. If `FALSE`, no prediction limits are imposed. Default is `TRUE`.

`progress` a logical indicating whether or not to print a progress bar for each sample to be predicted. Default is `TRUE`. Note: In case multicore processing is used, this progress bar will not be printed.

`cores` number of cores used for the computation of dissimilarities when `method` in `pcSelection` is `"opc"` (which can be computationally intensive) (default = 1). See details.

`allowParallel` To allow parallel execution of the sample loop (default is `TRUE`)

Details

The validation methods avaliable for assessing the predictive performance of the memory-based learning method used are described as follows:

- Leave-nearest-neighbour-out cross validation (`"NNv"`): From the group of neighbours of each sample to be predicted, the nearest sample (i.e. the most similar sample) is excluded and then a local model is fitted using the remaining neighbours. This model is then used to predict the value of the target response variable of the nearest sample. These predicted values are finally cross validated with the actual values (See Ramirez-Lopez et al. (2013a) for additional details). This method is faster than `"loc_crossval"`
- Local leave-group-out cross validation (`"loc_crossval"`): The group of neighbours of each sample to be predicted is partitioned into different equal size subsets. Each partition is selected based on a stratified random sampling which takes into account the values of the response variable of the corresponding set of neighbours. The selected local subset is used as local validation subset and the remaining samples are used for fitting a model. This model is used to predict the target response variable values of the local validation subset and the local root mean square error is computed. This process is repeated m times and the final local error is computed as the average of the local root mean square error of all the m iterations. In the `mbl` function m is controlled by the `resampling` argument and the size of the subsets is controlled by the `p` argument which indicates the percentage of samples to be selected from the subset of nearest neighbours. The global error of the predictions is computed as the average of the local root mean square errors.
- No validation (`"none"`): No validation is carried out. If `"none"` is seleceted along with `"NNv"` and/or `"loc_crossval"`, then it will be ignored and the respective validation(s) will be carried out.

Multi-threading for the computation of dissimilarities is based on OpenMP and hence works only on windows and linux. However, the loop used to iterate over the `Xu` samples in `mbl` uses the `%dopar%` operator of the `foreach` package, which can be used to parallelize this internal loop. The last example given in the `mbl` function ilustrates how to parallelize the `mbl` function.

Value

`mblControl` returns a `list` of class `mbl` with the specified parameters

Author(s)

Leonardo Ramirez-Lopez and Antoine Stevens

References

Ramirez-Lopez, L., Behrens, T., Schmidt, K., Stevens, A., Dematte, J.A.M., Scholten, T. 2013a. The spectrum-based learner: A new local approach for modeling soil vis-NIR spectra of complex datasets. Geoderma 195-196, 268-279.

Ramirez-Lopez, L., Behrens, T., Schmidt, K., Viscarra Rossel, R., Dematte, J. A. M., Scholten, T. 2013b. Distance and similarity-search metrics for use with soil vis-NIR spectra. Geoderma 199, 43-53.

See Also

fDiss, corDiss, sid, orthoDiss, mbl

Examples

```
#A control list with the default parameters
mblControl()

#A control list which specifies the moving correlation
#dissimilarity metric with a moving window of 30
mblControl(sm = "movcor", ws = 31)
```

---

neigCleaning *A function for identifying samples that do not belong to any of the neighbourhoods of a given set of samples (neigCleaning)*

Description

This function can be used to identify the samples in a spectral dataset Xr that do not belong to the neighbourhood of any sample in another spectral dataset Xu.

Usage

```
neigCleaning(Xr, Xu,
             sm = "pc",
             pcSelection = list("cumvar", 0.99),
             pcMethod = "svd",
             Yr = NULL,
             ws,
             k0,
             center = TRUE,
             scaled = TRUE,
             k.thr,
             k.dist.thr,
             k.range,
             returnDiss = FALSE,
             cores = 1)
```

Arguments

Xr	input (spectral) matrix (or data.frame) in which the neighbours of the samples in Xu shall be searched.
Xu	input (spectral) matrix (or data.frame) containing the samples for which their neighbours will be searched in Xr.
sm	a character string indicating the spectral dissimilarity metric to be used in the selection of the nearest neighbours of each observation for which a prediction is required (see mbl). Options are:

- "euclid": Euclidean dissimilarity.
- "cosine": Cosine dissimilarity.
- "sidF": Spectral information divergence computed on the spectral variables.
- "sidD": Spectral information divergence computed on the density distributions of the spectra.
- "cor": Correlation dissimilarity.
- "movcor": Moving window correlation dissimilarity.
- "pc": Principal components dissimilarity: Mahalanobis dissimilarity computed on the principal components space.
- "loc.pc": Dissimilarity estimation based on local principal components.
- "pls": Partial least squares dissimilarity: Mahalanobis dissimilarity computed on the partial least squares space.
- "loc.pls" Dissimilarity estimation based on local partial least squares.

The "pc" spectral dissimilarity metric is the default. If the "sidD" is chosen, the default parameters of the sid function are used however they cab be modified by specifying them as additional arguments in the mbl function. This argument can also be set to NULL, in such a case, a dissimilarity matrix must be specified in the dissimilarityM argument of the mbl function.

pcSelection if sm = "pc", sm = "loc.pc", sm = "pls" or sm = "loc.pls" a list which specifies the method to be used for identifying the number of principal components to be retained for computing the Mahalanobis distance of each sample in sm = "Xu" to the centre of sm = "Xr". It also specifies the number of components in any of the following cases: sm = "pc", sm = "loc.pc", sm = "pls" and sm = "loc.pls". This list must contain two objects in the following order:

- method:the method for selecting the number of components. Possible options are: "opc" (optimized pc selection based on Ramirez-Lopez et al. (2013a, 2013b). See the orthoProjection function for more details; "cumvar" (for selecting the number of principal components based on a given cumulative amount of explained variance); "var" (for selecting the number of principal components based on a given amount of explained variance); and "manual" (for specifying manually the desired number of principal components)
- value:a numerical value that complements the selected method. If "opc" is chosen, it must be a value indicating the maximal number of principal components to be tested (see Ramirez-Lopez et al., 2013a, 2013b). If "cumvar" is chosen, it must be a value (higher than 0 and lower than 1) indicating the maximum amount of cumulative variance that the retained components should explain. If "var" is chosen, it must be a value (higher than 0 and lower than 1) indicating that components that explain (individually) a variance lower than this treshold must be excluded. If "manual" is chosen, it must be a value specifying the desired number of principal components to retain.

The default method for the pcSelection argument is "opc" and the maximal number of principal components to be tested is set to 40. Optionally, the pcSelection argument admits "opc" or "cumvar" or "var" or "manual" as a

single character string. In such a case the default for "value" when either "opc" or "manual" are used is 40. When "cumvar" is used the default "value" is set to 0.99 and when "var" is used the default "value" is set to 0.01.

pcMethod — a character string indicating the principal component analysis algorithm to be used. Options are: "svd" (default) and "nipals". See orthoDiss.

Yr — either if the method used in the pcSelection argument is "opc" or if the sm argument is either "pls" or "loc.pls", then it must be a vector containing the side information corresponding to the spectra in Xr. It is equivalent to the sideInf parameter of the simEval function. It can be a numeric vector or matrix (regarding one or more continuous variables). The root mean square of differences (rmsd) is used for assessing the similarity between the samples and their corresponding most similar samples in terms of the side information provided. When sm = "pc", this parameter can also be a single discrete variable of class factor. In such a case the kappa index is used. See simEval function for more details.

ws — an odd integer value which specifies the window size when the moving window correlation similarity/dissimilarity is used (i.e sm = "movcor"). The default value is 41.

k0 — if any of the local similarity/dissimilarity methods is used (i.e. either sm = "loc.pc" or sm = "loc.pls") a numeric integer value. This argument controls the number of initial neighbours($k0$) to retain in order to compute the local principal components (at each neighbourhood).

center — a logical indicating if Xr and Xu must be centered (on the basis of $Xr \cup Xu$).

scaled — a logical indicating if Xr and Xu must be scaled (on the basis of $Xr \cup Xu$).

k.thr — an integer value indicating the k-nearest neighbours of each sample in Xu that must be selected from Xr.

k.dist.thr — an integer value indicating a distance treshold. When the distance between a sample in Xr and a sample in Xu is below the given treshold, the sample in sample in Xr is retained, otherwise it is ignored. The treshold depends on the corresponding similarity/dissimilarity metric specified in sm. Either k.thr or k.dist.thr must be specified.

k.range — a vector of length 2 which specifies the minimum (first value) and the maximum (second value) number of neighbours allowed when the k.dist.thr argument is used.

returnDiss — a logical indicating if the similarity/dissimilarity matrix must be returned. Default is FALSE.

cores — number of cores used when method in pcSelection is "opc" (which can be computationally intensive) (default = 1).

Details

This function may be specially useful when the reference set (Xr) is very large. In some cases the number of observations in the reference set can be reduced by removing irrelevant samples (i.e. samples that are not neighbours of a particular target set). If Xr is very large, it is recommended to consider the use this function prior using the mbl function.

Value

`neigCleaning` returns a `list` containing the following objects:

- `select` the indices of the observations in `Xr` that belong to the negihborhood of the samples in `Xu`.
- `reject` the indices of the observations in `Xr` that do not belong to the negihborhood of the samples in `Xu`.
- `rn.lower.k.dist` a `data.frame` that is returned only if the `k.dist.thr` argument was used. It comprises three columns, the first one (`sampleIndex`) indicates the index of the samples in `Xu`, the second column (`nk`) indicates the number of neighbours found in `Xr` for each sample in `Xr` and the third column (`neighbours.used`) indicates whether the original number of neighbours (below the distance treshold) was used or if the number of neighbours was reset to one of the range values specified in the `k.range` argument.
- `dissimilarity` the distance matrix used.

Author(s)

Leonardo Ramirez-Lopez

References

Ramirez-Lopez, L., Behrens, T., Schmidt, K., Stevens, A., Dematte, J.A.M., Scholten, T. 2013a. The spectrum-based learner: A new local approach for modeling soil vis-NIR spectra of complex datasets. Geoderma 195-196, 268-279.

Ramirez-Lopez, L., Behrens, T., Schmidt, K., Viscarra Rossel, R., Dematte, J. A. M., Scholten, T. 2013b. Distance and similarity-search metrics for use with soil vis-NIR spectra. Geoderma 199, 43-53.

See Also

`fDiss`, `corDiss`, `sid`, `orthoDiss`, `mbl`

Examples

```
## Not run:
require(prospectr)

data(NIRsoil)

Xu <- NIRsoil$spc[!as.logical(NIRsoil$train),]
Yu <- NIRsoil$CEC[!as.logical(NIRsoil$train)]
Yr <- NIRsoil$CEC[as.logical(NIRsoil$train)]
Xr <- NIRsoil$spc[as.logical(NIRsoil$train),]

Xu <- Xu[!is.na(Yu),]
Yu <- Yu[!is.na(Yu)]

Xr <- Xr[!is.na(Yr),]
Yr <- Yr[!is.na(Yr)]
```

```
# Identify the non-neighbour samples using the default parameters
# (In this example all the samples in Xr belong at least to the
# first 100 neighbours of one sample in Xu)
ex1 <- neigCleaning(Xr = Xr, Xu = Xu,
                            k.thr = 100)

# Identify the non-neighbour samples using principal component(PC)
# and partial least squares (PLS) distances, and using the "opc"
# approach for selecting the number of components
ex2 <- neigCleaning(Xr = Xr, Xu = Xu,
                            Yr = Yr,
                            sm = "pc",
                            pcSelection = list("opc", 40),
                            k.thr = 150)

ex3 <- neigCleaning(Xr = Xr, Xu = Xu,
                            Yr = Yr,
                            sm = "pls",
                            pcSelection = list("opc", 40),
                            k.thr = 150)

# Identify the non-neighbour samples using distances computed
# based on local PC analysis and using the "cumvar" and "var"
# approaches for selecting the number of PCs
ex4 <- neigCleaning(Xr = Xr, Xu = Xu,
                            sm = "loc.pc",
                            pcSelection = list("cumvar", 0.999),
                            k0 = 200,
                            k.thr = 150)

ex5 <- neigCleaning(Xr = Xr, Xu = Xu,
                            sm = "loc.pc",
                            pcSelection = list("var", 0.001),
                            k0 = 200,
                            k.thr = 150)

## End(Not run)
```

`orthoDiss` *A function for computing dissimilarity matrices from orthogonal projections (orthoDiss)*

Description

This function computes dissimilarities (in an orthogonal space) between either observations in a given set or between observations in two different sets. The dissimilarities are computed based on either principal component projection or partial least squares projection of the data. After projecting the data, the Mahalanobis distance is applied.

Usage

```
orthoDiss(Xr, X2 = NULL,
          Yr = NULL,
          pcSelection = list("cumvar", 0.99),
          method = "pca",
          local = FALSE,
          k0,
          center = TRUE, scaled = FALSE,
          return.all = FALSE, cores = 1, ...)
```

Arguments

Xr	a matrix (or data.frame) containing the (reference) data.
X2	an optional matrix (or data.frame) containing data of a second set of observations(samples).
Yr	either if the method used in the pcSelection argument is "opc" or if the sm argument is either "pls" or "loc.pls", then it must be a vector containing the side information corresponding to the spectra in Xr. It is equivalent to the sideInf parameter of the simEval function. It can be a numeric vector or matrix (regarding one or more continuous variables). The root mean square of differences (rmsd) is used for assessing the similarity between the samples and their corresponding most similar samples in terms of the side information provided. When sm = "pc", this parameter can also be a single discrete variable of class factor. In such a case the kappa index is used. See simEval function for more details.
pcSelection	a list which specifies the method to be used for identifying the number of principal components to be retained for computing the Mahalanobis distance of each sample in sm = "Xu" to the centre of sm = "Xr". It also specifies the number of components in any of the following cases: sm = "pc", sm = "loc.pc", sm = "pls" and sm = "loc.pls". This list must contain two objects in the following order: • method:the method for selecting the number of components. Possible options are: "opc" (optimized pc selection based on Ramirez-Lopez et al. (2013a, 2013b). See the orthoProjection function for more details; "cumvar" (for selecting the number of principal components based on a given cumulative amount of explained variance); "var" (for selecting the number of principal components based on a given amount of explained variance); and "manual" (for specifying manually the desired number of principal components) • value:a numerical value that complements the selected method. If "opc" is chosen, it must be a value indicating the maximal number of principal components to be tested (see Ramirez-Lopez et al., 2013a, 2013b). If "cumvar" is chosen, it must be a value (higher than 0 and lower than 1) indicating the maximum amount of cumulative variance that the retained components should explain. If "var" is chosen, it must be a value (higher than 0 and lower than 1) indicating that components that explain (individually) a variance lower than this threshold must be excluded. If "manual" is chosen, it

must be a value specifying the desired number of principal components to retain.

The default method for the `pcSelection` argument is `"opc"` and the maximal number of principal components to be tested is set to 40. Optionally, the `pcSelection` argument admits `"opc"` or `"cumvar"` or `"var"` or `"manual"` as a single character string. In such a case the default for `"value"` when either `"opc"` or `"manual"` are used is 40. When `"cumvar"` is used the default `"value"` is set to 0.99 and when `"var"` is used the default `"value"` is set to 0.01.

`method` the method for projecting the data. Options are: "pca" (principal component analysis using the singular value decomposition algorithm), "pca.nipals" (principal component analysis using the non-linear iterative partial least squares algorithm) and "pls" (partial least squares). See the `orthoProjection` function for further details on the projection methods.

`local` a logical indicating whether or not to compute the distances locally (i.e. projecting locally the data) by using the $k0$ nearest neighbour samples of each sample. Default is `FALSE`. See details.

`k0` if `local = TRUE` a numeric integer value which indicates the number of nearest neighbours($k0$) to retain in order to recompute the local orthogonal distances.

`center` a logical indicating if the spectral data `Xr` (and `X2` if specified) must be centered. If `X2` is specified the data is centered on the basis of $Xr \cup Xu$. For dissimilarity computations based on pls, the data is always centered for the projections.

`scaled` a logical indicating if `Xr` (and `X2` if specified) must be scaled. If `X2` is specified the data is scaled on the basis of $Xr \cup Xu$.

`return.all` a logical. In case `X2` is specified it indicates whether or not the distances between all the elements resulting from $Xr \cup Xu$ must be computed.

`cores` number of cores used when `method` in `pcSelection` is `"opc"` (which can be computationally intensive) and `local = FALSE` (default = 1). Dee details.

`...` additional arguments to be passed to the `orthoProjection` function.

Details

When `local = TRUE`, first a global distance matrix is computed based on the parameters specified. Then, by using this matrix for each target observation, a given set of nearest neighbours ($k0$) are identified. These neighbours (together with the target observation) are projected (from the original data space) onto a (local) orthogonal space (using the same parameters specified in the function). In this projected space the Mahalanobis distance between the target sample and the neighbours is recomputed. A missing value is assigned to the samples that do not belong to this set of neighbours (non-neighbour samples). In this case the dissimilarity matrix cannot be considered as a distance metric since it does not necessarily satisfies the symmetry condition for distance matrices (i.e. given two samples x_i and x_j, the local dissimilarity (d) between them is relative since generally $d(x_i, x_j) \neq d(x_j, x_i)$). On the other hand, when `local = FALSE`, the dissimilarity matrix obtained can be considered as a distance matrix.

Value

a `list` of class `orthoDiss` with the following components:

- `n.components` the number of components (either principal components or partial least squares components) used for computing the global distances.
- `global.variance.info` the information about the expalined variance(s) of the projection. When `local = TRUE`, the information corresponds to the global projection done prior computing the local projections.
- `loc.n.components` if `local = TRUE`, a `data.frame` which specifies the number of local components (either principal components or partial least squares components) used for computing the dissimilarity between each target sample and its neighbour samples.
- `dissimilarity` the computed dissimilarity matrix. If `local = FALSE` a distance `matrix`. If `local = TRUE` a `matrix` of class `orthoDiss`. In this case each column represent the dissimilarity between a target sample and its neighbourhood.

Multi-threading for the computation of dissimilarities (see `cores` parameter) is based on OpenMP and hence works only on windows and linux.

Author(s)

Leonardo Ramirez-Lopez

References

Ramirez-Lopez, L., Behrens, T., Schmidt, K., Stevens, A., Dematte, J.A.M., Scholten, T. 2013a. The spectrum-based learner: A new local approach for modeling soil vis-NIR spectra of complex datasets. Geoderma 195-196, 268-279.

Ramirez-Lopez, L., Behrens, T., Schmidt, K., Viscarra Rossel, R., Dematte, J. A. M., Scholten, T. 2013b. Distance and similarity-search metrics for use with soil vis-NIR spectra. Geoderma 199, 43-53.

See Also

`orthoProjection`, `simEval`

Examples

```
## Not run:
require(prospectr)

data(NIRsoil)

Xu <- NIRsoil$spc[!as.logical(NIRsoil$train),]
Yu <- NIRsoil$CEC[!as.logical(NIRsoil$train)]
Yr <- NIRsoil$CEC[as.logical(NIRsoil$train)]
Xr <- NIRsoil$spc[as.logical(NIRsoil$train),]

Xu <- Xu[!is.na(Yu),]
Yu <- Yu[!is.na(Yu)]

Xr <- Xr[!is.na(Yr),]
Yr <- Yr[!is.na(Yr)]
```

```
# Computation of the orthogonal dissimilarity matrix using the
# default parameters
ex1 <- orthoDiss(Xr = Xr, X2 = Xu)

# Computation of a principal component dissimilarity matrix using
# the "opc" method for the selection of the principal components
ex2 <- orthoDiss(Xr = Xr, X2 = Xu,
                 Yr = Yr,
                 pcSelection = list("opc", 40),
                 method = "pca",
                 return.all = TRUE)

# Computation of a partial least squares (PLS) dissimilarity
# matrix using the "opc" method for the selection of the PLS
# components
ex3 <- orthoDiss(Xr = Xr, X2 = Xu,
                 Yr = Yr,
                 pcSelection = list("opc", 40),
                 method = "pls")

# Computation of a partial least squares (PLS) local dissimilarity
# matrix using the "opc" method for the selection of the PLS
# components
ex4 <- orthoDiss(Xr = Xr, X2 = Xu,
                 Yr = Yr,
                 pcSelection = list("opc", 40),
                 method = "pls",
                 local = TRUE,
                 k0 = 200)

## End(Not run)
```

`orthoProjection` *Orthogonal projections using partial least squares and principal component analysis*

Description

Functions to perform orthogonal projections of high dimensional data matrices using partial least squares (pls) and principal component analysis (pca)

Usage

```
orthoProjection(Xr, X2 = NULL,
                Yr = NULL,
                method = "pca", pcSelection = list("cumvar", 0.99),
                center = TRUE, scaled = FALSE, cores = 1, ...)

pcProjection(Xr, X2 = NULL, Yr = NULL,
```

```
              pcSelection = list("cumvar", 0.99),
              center = TRUE, scaled = FALSE,
              method = "pca",
              tol = 1e-6, max.iter = 1000,
              cores = 1, ...)

plsProjection(Xr, X2 = NULL, Yr,
              pcSelection = list("opc", 40),
              scaled = FALSE,
              tol = 1e-6, max.iter = 1000,
              cores = 1, ...)

## S3 method for class 'orthoProjection'
predict(object, newdata, ...)

pcProjection(Xr, X2 = NULL, Yr = NULL, pcSelection = list("cumvar", 0.99),
  center = TRUE, scaled = FALSE, method = "pca", tol = 1e-06,
  max.iter = 1000, cores = 1, ...)

plsProjection(Xr, X2 = NULL, Yr, pcSelection = list("opc", 40),
  scaled = FALSE, tol = 1e-06, max.iter = 1000, cores = 1, ...)

## S3 method for class 'orthoProjection'
predict(object, newdata, ...)
```

Arguments

Xr	a matrix (or data.frame) containing the (reference) data.
X2	an optional matrix (or data.frame) containing data of a second set of observations(samples).
Yr	if the method used in the pcSelection argument is "opc" or if the sm argument is either "pls" or "loc.pls", then it must be a vector containing the side information corresponding to the spectra in Xr. It is equivalent to the sideInf parameter of the simEval function. It can be a numeric vector or matrix (regarding one or more continuous variables). The root mean square of differences (rmsd) is used for assessing the similarity between the samples and their corresponding most similar samples in terms of the side information provided. When sm = "pc", this parameter can also be a single discrete variable of class factor. In such a case the kappa index is used. See simEval function for more details.
method	the method for projecting the data. Options are: "pca" (principal component analysis using the singular value decomposition algorithm), "pca.nipals" (principal component analysis using the non-linear iterative partial least squares algorithm) and "pls" (partial least squares).
pcSelection	a list which specifies the method to be used for identifying the number of principal components to be retained for computing the Mahalanobis distance of each sample in sm = "Xu" to the centre of sm = "Xr". It also specifies the number of components in any of the following cases: sm = "pc", sm = "loc.pc",

sm = "pls" and sm = "loc.pls". This list must contain two objects in the following order:

- method:the method for selecting the number of components. Possible options are: "opc" (optimized pc selection based on Ramirez-Lopez et al. (2013a, 2013b) in which the side information concept is used, see details), "cumvar" (for selecting the number of principal components based on a given cumulative amount of explained variance); "var" (for selecting the number of principal components based on a given amount of explained variance); and "manual" (for specifying manually the desired number of principal components)
- value:a numerical value that complements the selected method. If "opc" is chosen, it must be a value indicating the maximal number of principal components to be tested (see Ramirez-Lopez et al., 2013a, 2013b). If "cumvar" is chosen, it must be a value (higher than 0 and lower than 1) indicating the maximum amount of cumulative variance that the retained components should explain. If "var" is chosen, it must be a value (higher than 0 and lower than 1) indicating that components that explain (individually) a variance lower than this threshold must be excluded. If "manual" is chosen, it must be a value specifying the desired number of principal components to retain.

The default method for the pcSelection argument is "opc" and the maximal number of principal components to be tested is set to 40. Optionally, the pcSelection argument admits "opc" or "cumvar" or "var" or "manual" as a single character string. In such a case the default for "value" when either "opc" or "manual" are used is 40. When "cumvar" is used the default "value" is set to 0.99 and when "var" is used the default "value" is set to 0.01.

center — a logical indicating if the data Xr (and X2 if specified) must be centered. If X2 is specified the data is centered on the basis of $Xr \cup Xu$. This argument only applies to the principal components projection. For pls projections the data is always centered.

scaled — a logical indicating if Xr (and X2 if specified) must be scaled. If X2 is specified the data is scaled on the basis of $Xr \cup Xu$.

cores — number of cores used when method in pcSelection is "opc" (which can be computationally intensive) (default = 1). Dee details.

... — additional arguments to be passed to pcProjection or plsProjection.

tol — tolerance limit for convergence of the algorithm in the nipals algorithm (default is 1e-06). In the case of PLS this applies only to Yr with more than two variables.

max.iter — maximum number of iterations (default is 1000). In the case of method = "pls" this applies only to Yr matrices with more than one variable.

object — object of class "orthoProjection" (as returned by orthoProjection, pcProjection or plsProjection).

newdata — an optional data frame or matrix in which to look for variables with which to predict. If omitted, the scores are used. It must contain the same number of columns, to be used in the same order.

Details

In the case of `method = "pca"`, the algrithm used is the singular value decomposition in which given a data matrix X, is factorized as follows:

$$X = UDV^{\mathrm{T}}$$

where U and V are othogonal matrices, and where U is a matrix of the left singular vectors of X, D is a diagonal matrix containing the singular values of X and V is the is a matrix of the right singular vectors of X. The matrix of principal component scores is obtained by a matrix multiplication of U and D, and the matrix of principal component loadings is equivalent to the matrix V. When `method = "pca.nipals"`, the algorithm used for principal component analysis is the non-linear iterative partial least squares (nipals). In the case of the of the partial least squares projection (a.k.a projection to latent structures) the nipals regression algorithm. Details on the "nipals" algorithm are presented in Martens (1991). When `method = "opc"`, the selection of the components is carried out by using an iterative method based on the side information concept (Ramirez-Lopez et al. 2013a, 2013b). First let be P a sequence of retained components (so that $P = 1, 2, ..., k$. At each iteration, the function computes a dissimilarity matrix retaining p_i components. The values of the side information of the samples are compared against the side information values of their most spectrally similar samples. The optimal number of components retrieved by the function is the one that minimizes the root mean squared differences (RMSD) in the case of continuous variables, or maximizes the kappa index in the case of categorical variables. In this process the `simEval` function is used. Note that for the `"opc"` method is necessary to specify `Yr` (the side information of the samples). Multi-threading for the computation of dissimilarities (see `cores` parameter) is based on OpenMP and hence works only on windows and linux.

Value

`orthoProjection`, `pcProjection`, `plsProjection`, return a `list` of class `orthoProjection` with the following components:

- `scores` a `matrix` of scores corresponding to the samples in `Xr` and `X2` (if it applies). The number of components that the scores represent is given by the number of components chosen in the function.
- `X.loadings` a `matrix` of loadings corresponding to the explanatory variables. The number of components that these loadings represent is given by the number of components chosen in the function.
- `Y.loadings` a `matrix` of partial least squares loadings corresponding to `Yr`. The number of components that these loadings represent is given by the number of components chosen in the function. This object is only returned if the partial least squares algorithm was used.
- `weigths` a `matrix` of partial least squares ("pls") weights. This object is only returned if the "pls" algorithm was used.
- `projectionM` a `matrix` that can be used to project new data onto a "pls" space. This object is only returned if the "pls" algorithm was used.
- `variance` a `matrix` indicating the standard deviation of each component (sdv), the cumulative explained variance (cumExplVar) and the variance explained by each single component (explVar). These values are computed based on the data used to create the projection matrices. For example if the "pls" method was used, then these values are computed based only on the

data that contains information on Yr (i.e. the Xr data) If the principal component method is used, the this data is computed on the basis of Xr and X2 (if it applies) since both matrices are employed in the computation of the projection matrix (loadings in this case).

- svd the standard deviation of the retrieved scores.
- n.components the number of components (either principal components or partial least squares components) used for computing the global distances.
- opcEval a data.frame containing the statistics computed for optimizing the number of principal components based on the variable(s) specified in the Yr argument. If Yr was a continuous was a continuous vector or matrix then this object indicates the root mean square of differences (rmse) for each number of components. If Yr was a categorical variable this object indicates the kappa values for each number of components. This object is returned only if "opc" was used within the pcSelection argument. See the simEval function for more details.
- method the orthoProjection method used.

predict.orthoProjection, returns a matrix of scores proprojected for newdtata.

Author(s)

Leonardo Ramirez-Lopez

References

Martens, H. (1991). Multivariate calibration. John Wiley & Sons.

Ramirez-Lopez, L., Behrens, T., Schmidt, K., Stevens, A., Dematte, J.A.M., Scholten, T. 2013a. The spectrum-based learner: A new local approach for modeling soil vis-NIR spectra of complex datasets. Geoderma 195-196, 268-279.

Ramirez-Lopez, L., Behrens, T., Schmidt, K., Viscarra Rossel, R., Dematte, J. A. M., Scholten, T. 2013b. Distance and similarity-search metrics for use with soil vis-NIR spectra. Geoderma 199, 43-53.

See Also

orthoDiss, simEval, mbl

Examples

```
## Not run:
require(prospectr)

data(NIRsoil)

Xu <- NIRsoil$spc[!as.logical(NIRsoil$train),]
Yu <- NIRsoil$CEC[!as.logical(NIRsoil$train)]
Yr <- NIRsoil$CEC[as.logical(NIRsoil$train)]
Xr <- NIRsoil$spc[as.logical(NIRsoil$train),]

Xu <- Xu[!is.na(Yu),]
Yu <- Yu[!is.na(Yu)]
```

```
Xr <- Xr[!is.na(Yr),]
Yr <- Yr[!is.na(Yr)]

# A partial least squares projection using the "opc" method
# for the selection of the optimal number of components
plsProj <- orthoProjection(Xr = Xr, Yr = Yr, X2 = Xu,
                           method = "pls",
                           pcSelection = list("opc", 40))

# A principal components projection using the "opc" method
# for the selection of the optimal number of components
pcProj <- orthoProjection(Xr = Xr, Yr = Yr, X2 = Xu,
                          method = "pca",
                          pcSelection = list("opc", 40))

# A partial least squares projection using the "cumvar" method
# for the selection of the optimal number of components
plsProj2 <- orthoProjection(Xr = Xr, Yr = Yr, X2 = Xu,
                            method = "pls",
                            pcSelection = list("cumvar", 0.99))

## End(Not run)
```

`plot.mbl` *Plot method for an object of class* `mbl`

Description

Plots the content of an object of class `mbl`

Usage

```
## S3 method for class 'mbl'
plot(x, g = c("validation", "pca"), param = "rmse", pcs = c(1,2), ...)
```

Arguments

`x`	an object of class `mbl` (as returned by `mbl`).
`g`	a character `vector` indicating what results shall be plotted. Options are: "validation" (for plotting the validation results) and/or "pca" (for plotting the principal components).
`param`	one of the following options "rmse", "st.rmse" or "r2". The respective validation statistic is then plotted. It is only available if `"validation"` is specified in the `g` argument.
`pcs`	a vector of length one or two indicating the principal components to be plotted. Default is `c(1, 2)`. It is only available if `"pca"` is specified in the `g` argument.
`...`	some arguments to be passed to methods as graphical parameters.

Author(s)

Leonardo Ramirez-Lopez and Antoine Stevens

See Also

mbl

Examples

```
## Not run:
require(prospectr)

data(NIRsoil)

Xu <- NIRsoil$spc[!as.logical(NIRsoil$train),]
Yu <- NIRsoil$CEC[!as.logical(NIRsoil$train)]
Yr <- NIRsoil$CEC[as.logical(NIRsoil$train)]
Xr <- NIRsoil$spc[as.logical(NIRsoil$train),]

Xu <- Xu[!is.na(Yu),]
Yu <- Yu[!is.na(Yu)]

Xr <- Xr[!is.na(Yr),]
Yr <- Yr[!is.na(Yr)]

ctrl <- mblControl(sm = "cor", ws = 51,
                   pcSelection = list("cumvar", 0.999),
                   valMethod = c("NNv"),
                   scaled = TRUE, center = TRUE)

ex1 <- mbl(Yr = Yr, Xr = Xr, Yu = NULL, Xu = Xu,
           mblCtrl = ctrl,
           dissUsage = "none",
           k = seq(30, 250, 30),
           method = "wapls1",
           plsC = c(7, 20))

plot(ex1)

## End(Not run)
```

---

plot.orthoProjection *Plot method for an object of class* orthoProjection

---

Description

Plots the content pf an object of class orthoProjection

Usage

```
## S3 method for class 'orthoProjection'
plot(x, ...)
```

Arguments

x	an object of class orthoProjection (as returned by orthoProjection).
...	arguments to be passed to methods.

Author(s)

Leonardo Ramirez-Lopez and Antoine Stevens

See Also

orthoProjection

Examples

```
## Not run:
require(prospectr)

data(NIRsoil)

Xu <- NIRsoil$spc[!as.logical(NIRsoil$train),]
Yu <- NIRsoil$CEC[!as.logical(NIRsoil$train)]
Yr <- NIRsoil$CEC[as.logical(NIRsoil$train)]
Xr <- NIRsoil$spc[as.logical(NIRsoil$train),]

Xu <- Xu[!is.na(Yu),]
Yu <- Yu[!is.na(Yu)]

Xr <- Xr[!is.na(Yr),]
Yr <- Yr[!is.na(Yr)]

# A partial least squares projection using the "opc" method
# for the selection of the optimal number of components
plsProj <- orthoProjection(Xr = Xr, Yr = Yr, X2 = Xu,
                           method = "pls",
                           pcSelection = list("opc", 40))

plot(plsProj)

## End(Not run)
```

print.localOrthoDiss *Print method for an object of class* orthoDiss

Description

Prints the content of an object of class orthoDiss

Usage

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

Arguments

x	an object of class localOrthoDiss (returned by orthoDiss when it uses local = TRUE).
...	arguments to be passed to methods (not yet functional).

Author(s)

Leonardo Ramirez-Lopez and Antoine Stevens

print.mbl *Print method for an object of class* mbl

Description

Prints the content of an object of class mbl

Usage

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

Arguments

x	an object of class mbl (as returned by the mbl function).
...	arguments to be passed to methods (not yet functional).

Author(s)

Leonardo Ramirez-Lopez and Antoine Stevens

print.orthoProjection *Print method for an object of class* orthoProjection

Description

Prints the contents of an object of class orthoProjection

Usage

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

Arguments

x	an object of class orthoProjection (as returned by the orthoProjection function).
...	arguments to be passed to methods (not yet functional).

Author(s)

Leonardo Ramirez-Lopez

sid *A function for computing the spectral information divergence between spectra (sid)*

Description

This function computes the spectral information divergence (distance) between spectra based on the kullback-leibler divergence algorithm (see details).

Usage

```
sid(Xr, X2 = NULL,
    mode = "density",
    center = FALSE, scaled = TRUE,
    kernel = "gaussian",
    n = if(mode == "density") round(0.5 * ncol(Xr)),
    bw = "nrd0",
    reg = 1e-04,
    ...)
```

Arguments

Xr	a matrix (or data.frame) containing the spectral (reference) data.
X2	an optional matrix (or data.frame) containing the spectral data of a second set of samples.
mode	the method to be used for computing the spectral information divergence. Options are "density" (default) for computing the divergence values on the density distributions of the spectral observations, and "feature" for computing the divergence vales on the spectral features. See details.
center	a logical indicating if the computations must be carried out on the centered X and X2 (if specified) matrices. If mode = "feature" centering is not carried out since this option does not accept negative values which are generated after centering the matrices. Default is FALSE. See details.
scaled	a logical indicating if the computations must be carried out on the variance scaled X and X2 (if specified) matrices. Default is TRUE.
kernel	if mode = "density" a character string indicating the smoothing kernel to be used. It must be one of "gaussian" (default), "rectangular", "triangular", "epanechnikov", "biweight", "cosine" or "optcosine". See the density function of the stats package.
n	if mode = "density" a numerical value indicating the number of equally spaced points at which the density is to be estimated. See the density function of the stats package for further details. Default is round(0.5 * ncol(X)).
bw	if mode = "density" a numerical value indicating the smoothing kernel bandwidth to be used. Optionally the character string "nrd0" can be used, it computes the bandwidth using the bw.nrd0 function of the stats package. See the density and the bw.nrd0 functions for more details. By default "nrd0" is used, in this case the bandwidth is computed as bw.nrd0(as.vector(X)), if X2 is specified the bandwidth is computed as bw.nrd0(as.vector(rbind(X, X2))).
reg	a numerical value higher than 0 which indicates a regularization parameter. Values (probabilities) below this threshold are replaced by this value for numerical stability. Default is 1e-4.
...	additional arguments to be passed to the density function of the base package.

Details

This function computes the spectral information divergence (distance) between spectra. When mode = "density", the function first computes the probability distribution of each spectrum which result in a matrix of density distribution estimates. The density distributions of all the samples in the datasets are compared based on the kullback-leibler divergence algorithm. When mode = "feature", the kullback-leibler divergence between all the samples is computed directly on the spectral variables. The spectral information divergence (SID) algorithm (Chang, 2000) uses the Kullback-Leibler divergence (KL) or relative entropy (Kullback and Leibler, 1951) to account for the vis-NIR information provided by each spectrum. The SID between two spectra (x_i and x_j) is computed as follows:

$$SID(x_i, x_j) = KL(x_i \,||\, x_j) + KL(x_j \,||\, x_i)$$

$$SID(x_i, x_j) = \sum_{l=1}^{k} p_l \, log(\frac{p_l}{q_l}) + \sum_{l=1}^{k} q_l \, log(\frac{q_l}{p_l})$$

where k represents the number of variables or spectral features, p and q are the probability vectors of x_i and x_j respectively which are calculated as:

$$p = \frac{x_i}{\sum_{l=1}^{k} x_{i,l}}$$

$$q = \frac{x_j}{\sum_{l=1}^{k} x_{j,l}}$$

From the above equations it can be seen that the original SID algorithm assumes that all the components in the data matrices are nonnegative. Therefore centering cannot be applied when `mode = "feature"`. If a data matrix with negative values is provided and `mode = "feature"`, the `sid` function automatically scales the matrix as follows:

$$X_s = \frac{X - min(X)}{max(X) - min(X)}$$

or

$$X_s = \frac{X - min(X, X2)}{max(X, X2) - min(X, X2)}$$

$$X2_s = \frac{X2 - min(X, X2)}{max(X, X2) - min(X, X2)}$$

if `X2` is specified. The 0 values are replaced by a regularization parameter (`reg` argument) for numerical stability. The default of the `sid` function is to compute the SID based on the density distributions of the spectra (`mode = "density"`). For each spectrum in `X` the density distribution is computed using the `density` function of the `stats` package. The 0 values of the estimated density distributions of the spectra are replaced by a regularization parameter (`"reg"` argument) for numerical stability. Finally the divergence between the computed spectral histogramas is computed using the SID algorithm. Note that if `mode = "density"`, the `sid` function will accept negative values and matrix centering will be possible.

Value

a `list` with the following components:

- `sid` if only `"X"` is specified (i.e. `X2 = NULL`), a square symmetric matrix of SID distances between all the components in `"X"`. If both `"X"` and `"X2"` are specified, a matrix of SID distances between the components in `"X"` and the components in `"X2"`) where the rows represent the objects in `"X"` and the columns represent the objects in `"X2"`
- `Xr` the (centered and/or scaled if specified) spectral `X` matrix
- `X2` the (centered and/or scaled if specified) spectral `X2` matrix
- `densityDisXr` if `mode = "density"`, the computed density distributions of `Xr`
- `densityDisX2` if `mode = "density"`, the computed density distributions of `X2`

Author(s)

Leonardo Ramirez-Lopez

References

Chang, C.I. 2000. An information theoretic-based approach to spectral variability, similarity and discriminability for hyperspectral image analysis. IEEE Transactions on Information Theory 46, 1927-1932.

See Also

density

Examples

```
## Not run:
require(prospectr)

data(NIRsoil)

Xu <- NIRsoil$spc[!as.logical(NIRsoil$train),]
Yu <- NIRsoil$CEC[!as.logical(NIRsoil$train)]
Yr <- NIRsoil$CEC[as.logical(NIRsoil$train)]
Xr <- NIRsoil$spc[as.logical(NIRsoil$train),]

Xu <- Xu[!is.na(Yu),]
Xr <- Xr[!is.na(Yr),]

# Example 1
# Compute the SID distance between all the samples in Xr
xr.sid <- sid(Xr = Xr)
xr.sid

# Example 2
# Compute the SID distance between the samples in Xr and the samples
# in Xu
xru.sid <- sid(Xr = Xr, X2 = Xu)
xru.sid

# Example 3
# Compute the SID distance between the samples in Xr and the samples
# in Xu using the histograms
xru.sid.hist <- sid(Xr = Xr, X2 = Xu, mode = "feature")
xru.sid.hist

## End(Not run)
```

---

simEval *A function for evaluating similarity/dissimilarity matrices (simEval)*

Description

This function searches for the most similar sample of each sample in a given data set based on a similarity/dissimilarity (e.g. distance matrix). The samples are compared against their corresponding most similar samples in terms of the side information provided. The root mean square of differences and the correlation coefficient are computed for continuous variables and for discrete variables the kappa index is calculated.

Usage

```
simEval(d, sideInf, lower.tri = FALSE, cores = 1, ...)
```

Arguments

d	a vector or a square symmetric matrix (or data.frame) of similarity/dissimilarity scores between samples of a given dataset (see lower.tri).
sideInf	a vector containing the side information corresponding to the samples in the dataset from which the similarity/dissimilarity matrix was computed. It can be either a numeric vector (continuous variable) or a factor (discrete variable). If it is a numeric vector, the root mean square of differences is used for assessing the similarity between the samples and their corresponding most similar samples in terms of the side information provided. If it is a factor, then the kappa index is used. See details.
lower.tri	a logical indicating whether the input similarities/dissimilarities are given as a vector of the lower triangle of the distance matrix (as returned e.g. by base::dist) or as a square symmetric matrix (or data.frame) (default = FALSE)
cores	number of cores used to find the neareast neighbours of similarity/dissimilarity scores (default = 1). See details.
...	additional parameters (for internal use only).

Details

For the evaluation of similarity/dissimilarity matrices this function uses side information (information about one variable which is available for a group of samples, Ramirez-Lopez et al., 2013). It is assumed that there is a correlation (or at least an indirect or secondary correlation) between this side informative variable and the spectra. In other words, this approach is based on the assumption that the similarity measures between the spectra of a given group of samples should be able to reflect their similarity also in terms of the side informative variable (e.g. compositional similarity). If sideInf is a numeric vector the root mean square of differences (RMSD) is used for assessing the similarity between the samples and their corresponding most similar samples in terms of the side information provided. It is computed as follows: It can be computed as:

$$RMSD = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \ddot{y}_i)^2}$$

where y_i is the value of the side variable of the ith sample, $\ddot{y}_i$ is the value of the side variable of the nearest neighbour of the ith sample and n is the total number of observations. If sideInf is a factor

the kappa index (κ) is used instead the RMSD. It is computed as follows:

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$

where both p_o and p_e are two different agreement indexes between the the side information of the samples and the side information of their corrresponding nearest samples (i.e. most similar samples). While p_o is the relative agreement p_e is the the agreement expected by chance. Multi-threading for the computation of dissimilarities (see `cores` parameter) is based on OpenMP and hence works only on windows and linux.

Value

`simEval` returns a `list` with the following components:

- "`eval` either the RMSD (and the correlation coefficient) or the kappa index
- `firstNN` a `data.frame` containing the original side informative variable in the first column and the side informative values of the corresponding nearest neighbours in the second column

Author(s)

Leonardo Ramirez-Lopez

References

Ramirez-Lopez, L., Behrens, T., Schmidt, K., Stevens, A., Dematte, J.A.M., Scholten, T. 2013a. The spectrum-based learner: A new local approach for modeling soil vis-NIR spectra of complex datasets. Geoderma 195-196, 268-279.

Ramirez-Lopez, L., Behrens, T., Schmidt, K., Viscarra Rossel, R., Dematte, J. A. M., Scholten, T. 2013b. Distance and similarity-search metrics for use with soil vis-NIR spectra. Geoderma 199, 43-53.

Examples

```
## Not run:
require(prospectr)

data(NIRsoil)

Yr <- NIRsoil$Nt[as.logical(NIRsoil$train)]
Xr <- NIRsoil$spc[as.logical(NIRsoil$train),]

# Example 1
# Compute a principal components distance
pca.d <- orthoDiss(Xr = Xr, pcSelection = list("cumvar", 0.999),
                   method = "pca",
                   local = FALSE,
                   center = TRUE, scaled = TRUE)

# The final number of pcs used for computing the distance
# matrix of objects in Xr
```

```
pca.d$n.components

# The final distance matrix
ds <- pca.d$dissimilarity

# Example 1.1
# Evaluate the distance matrix on the baisis of the
# side information (Yr) associated with Xr
se <- simEval(d = ds, sideInf = Yr)

# The final evaluation results
se$eval

# The final values of the side information (Yr) and the values of
# the side information corresponding to the first nearest neighbours
# found by using the distance matrix
se$firstNN

# Example 1.2
# Evaluate the distance matrix on the baisis of two side
# information (Yr and Yr2)
# variables associated with Xr
Yr2 <- NIRsoil$CEC[as.logical(NIRsoil$train)]
se2 <- simEval(d = ds, sideInf = cbind(Yr, Yr2))

# The final evaluation results
se2$eval

# The final values of the side information variables and the values
# of the side information variables corresponding to the first
# nearest neighbours found by using the distance matrix
se2$firstNN

###
# Example 2
# Evaluate the distances produced by retaining different number of
# principal components (this is the same principle used in the
# optimized principal components approach ("opc"))

# first project the data
pca <- orthoProjection(Xr = Xr, method = "pca",
                       pcSelection = list("manual", 30),
                       center = TRUE, scaled = TRUE)

# standardize the scores
scores.s <- sweep(pca$scores, MARGIN = 2,
                  STATS = pca$sc.sdv, FUN = "/")
rslt <-  matrix(NA, ncol(scores.s), 3)
colnames(rslt) <- c("pcs", "rmsd", "r")
rslt[,1] <- 1:ncol(scores.s)
for(i in 1:ncol(scores.s))
{
  sc.ipcs <- scores.s[ ,1:i, drop = FALSE]
```

```
  di <- fDiss(Xr = sc.ipcs, method = "euclid",
              center = FALSE, scaled = FALSE)
  se <- simEval(d = di, sideInf = Yr)
  rslt[i,2:3] <- unlist(se$eval)
}
plot(rslt)

###
# Example 3
# Example 3.1
# Evaluate a dissimilarity matrix computed using a moving window
# correlation method
mwcd <- mcorDiss(Xr = Xr, ws = 35, center = FALSE, scaled = FALSE)
se.mw <- simEval(d = mwcd, sideInf = Yr)
se.mw$eval

# Example 3.2
# Evaluate a dissimilarity matrix computed using the correlation
# method
cd <- corDiss(Xr = Xr, center = FALSE, scaled = FALSE)
se.nc <- simEval(d = cd, sideInf = Yr)
se.nc$eval

## End(Not run)
```

Index