Package 'resemble'

May 20, 2025

Type Package

Title Memory-Based Learning in Spectral Chemometrics

Version 2.2.4

Date 2025-05-19

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

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

Description Functions for dissimilarity analysis and memory-based learning (MBL, a.k.a local modeling) in complex spectral data sets. Most of these functions are based on the methods presented in Ramirez-Lopez et al. (2013) <doi:10.1016/j.geoderma.2012.12.014>.

License MIT + file LICENSE

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

Depends R (>= 3.5.0)

Imports foreach, iterators, Rcpp (>= 1.0.3), mathjaxr (>= 1.0), magrittr (>= 1.5.0), lifecycle (>= 0.2.0), data.table (>= 1.9.8)

Suggests prospectr, parallel, doParallel, testthat, formatR, rmarkdown, bookdown, knitr

LinkingTo Rcpp, RcppArmadillo

RdMacros mathjaxr

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

RoxygenNote 7.3.2

Encoding UTF-8

Config/VersionName olbap

Author Leonardo Ramirez-Lopez [aut, cre] (ORCID: <https://orcid.org/0000-0002-5369-5120>), Antoine Stevens [aut, ctb] (ORCID:

<https://orcid.org/0000-0002-1588-7519>),
Claudio Orellano [ctb],
Raphael Viscarra Rossel [ctb] (ORCID:
<https://orcid.org/0000-0003-1540-4748>),
Alex Wadoux [ctb] (ORCID: <https://orcid.org/0000-0001-7325-9716>)

Date/Publication 2025-05-19 22:30:02 UTC

Contents

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

Description

Maturing

Functions for memory-based learning

Details

This is the version 2.2.4 – olbap of the package. It implements a number of functions useful for modeling complex spectral spectra (e.g. NIR, IR). The package includes functions for dimensionality reduction, computing spectral dissimilarity matrices, nearest neighbor search, and modeling spectral data using memory-based learning. This package builds upon the methods presented in Ramirez-Lopez et al. (2013) doi:10.1016/j.geoderma.2012.12.014.

Development versions can be found in the github repository of the package at https://github.com/l-ramirez-lopez/resemble.

The functions available for dimensionality reduction are:

- ortho_projection
- pc_projection
- pls_projection
- predict.ortho_projection

The functions available for computing dissimilarity matrices are:

- dissimilarity
- f_diss
- cor_diss
- sid
- ortho_diss

The functions available for evaluating dissimilarity matrices are:

- sim_eval

The functions available for nearest neighbor search:

- search_neighbors

The functions available for modeling spectral data:

- mbl
- mbl_control

Other supplementary functions:

- plot.mbl
- plot.ortho_projection

Author(s)

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

Authors:

- Leonardo Ramirez-Lopez (ORCID)
- Antoine Stevens (ORCID)
- Claudio Orellano
- Raphael Viscarra Rossel (ORCID)
- Zefang Shen
- Craig Lobsey (ORCID)
- Alex Wadoux (ORCID)

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 data sets. Geoderma 195-196, 268-279.

See Also

Useful links:

- https://github.com/l-ramirez-lopez/resemble
- Report bugs at https://github.com/l-ramirez-lopez/resemble/issues

---

`cor_diss` *Correlation and moving correlation dissimilarity measurements (cor_diss)*

Description

Stable

Computes correlation and moving correlation dissimilarity matrices.

Usage

```
cor_diss(Xr, Xu = NULL, ws = NULL,
         center = TRUE, scale = FALSE)
```

Arguments

`Xr`	a matrix.
`Xu`	an optional matrix containing data of a second set of observations.
`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 `Xu` if specified) must be centered. If `Xu` is provided, the data is scaled on the basis of $Xr \cup Xu$.
`scale`	a logical indicating if `Xr` (and `Xu` if specified) must be scaled. If `Xu` is provided the data is scaled on the basis of $Xr \cup Xu$.

Details

The correlation dissimilarity d between two observations x_i and x_j is based on the Perason's correlation coefficient (ρ) and it can be computed as follows:

$$d(x_i, x_j) = \frac{1}{2}((1 - \rho(x_i, x_j)))$$

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

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

where ws represents a given window size which rolls sequentially from 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

```
library(prospectr)
data(NIRsoil)

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

cor_diss(Xr = Xr)

cor_diss(Xr = Xr, Xu = Xu)

cor_diss(Xr = Xr, ws = 41)

cor_diss(Xr = Xr, Xu = Xu, ws = 41)
```

---

dissimilarity *Dissimilarity computation between matrices*

Description

This is a wrapper to integrate the different dissimilarity functions of the offered by package.It computes the dissimilarities between observations in numerical matrices by using an specifed dissmilarity measure.

Usage

```
dissimilarity(Xr, Xu = NULL,
              diss_method = c("pca", "pca.nipals", "pls", "mpls",
                              "cor", "euclid", "cosine", "sid"),
              Yr = NULL, gh = FALSE, pc_selection = list("var", 0.01),
              return_projection = FALSE, ws = NULL,
              center = TRUE, scale = FALSE, documentation = character(),
              ...)
```

Arguments

Xr	a matrix of containing n observations/rows and p variables/columns.
Xu	an optional matrix containing data of a second set of observations with p variables/columns.
diss_method	a character string indicating the method to be used to compute the dissimilarities between observations. Options are: • "pca": Mahalanobis distance computed on the matrix of scores of a Principal Component (PC) projection of Xr (and Xu if provided). PC projection is done using the singular value decomposition (SVD) algorithm. See ortho_diss function. • "pca.nipals": Mahalanobis distance computed on the matrix of scores of a Principal Component (PC) projection of Xr (and Xu if provided). PC projection is done using the non-linear iterative partial least squares (nipals) algorithm. See ortho_diss function. • "pls": Mahalanobis distance computed on the matrix of scores of a partial least squares projection of Xr (and Xu if provided). In this case, Yr is always required. See ortho_diss function. • "mpls": Mahalanobis distance computed on the matrix of scores of a modified partial least squares projection (Shenk and Westerhaus, 1991; Westerhaus, 2014) of Xr (and Xu if provided). In this case, Yr is always required. See ortho_diss function. • "cor": based on the correlation coefficient between observations. See cor_diss function. • "euclid": Euclidean distance between observations. See f_diss function. • "cosine": Cosine distance between observations. See f_diss function. • "sid": spectral information divergence between observations. See sid function.
Yr	a numeric matrix of n observations used as side information of Xr for the ortho_diss methods (i.e. pca, pca.nipals or pls). It is required when: • diss_method = "pls" • diss_method = "pca" with "opc" used as the method in the pc_selection argument. See ortho_diss. • gh = TRUE
gh	a logical indicating if the Mahalanobis distance (in the pls score space) between each observation and the pls centre/mean must be computed.
pc_selection	a list of length 2 to be passed onto the ortho_diss methods. It is required if the method selected in diss_method is any of "pca", "pca.nipals" or "pls" or if gh = TRUE. This argument is used for optimizing the number of components (principal components or pls factors) to be retained. This list must contain two elements in the following order: method (a character indicating the method for selecting the number of components) and value (a numerical value that complements the selected method). The methods available are: • "opc": optimized principal component selection based on Ramirez-Lopez et al. (2013a, 2013b). The optimal number of components (of set of observations) is the one for which its distance matrix minimizes the differences

between the Yr value of each observation and the Yr value of its closest observation. In this case value must be a value ((larger than 0 and below the minimum dimension of Xr or Xr and Xu combined) indicating the maximum number of principal components to be tested. See the ortho_projection function for more details.

- "cumvar": selection of the principal components based on a given cumulative amount of explained variance. In this case, value must be a value (larger than 0 and below or equal to 1) indicating the minimum amount of cumulative variance that the combination of retained components should explain.
- "var": selection of the principal components based on a given amount of explained variance. In this case, value must be a value (larger than 0 and below or equal to 1) indicating the minimum amount of variance that a single component should explain in order to be retained.
- "manual": for manually specifying a fix number of principal components. In this case, value must be a value (larger than 0 and below the minimum dimension of Xr or Xr and Xu combined). indicating the minimum amount of variance that a component should explain in order to be retained.

The default is list(method = "var", value = 0.01).

Optionally, the pc_selection argument admits "opc" or "cumvar" or "var" or "manual" as a single character string. In such a case the default "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.

return_projection
: a logical indicating if the projection(s) must be returned. Projections are used if the ortho_diss methods are called (i.e. diss_method = "pca", diss_method = "pca.nipals" or diss_method = "pls") or when gh = TRUE. In case gh = TRUE and a ortho_diss method is used (in the diss_method argument), both projections are returned.

ws
: an odd integer value which specifies the window size, when diss_method = "cor" (cor_diss method) for moving correlation dissimilarity. If ws = NULL (default), 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 cor_diss function.

center
: a logical indicating if Xr (and Xu if provided) must be centered. If Xu is provided the data is centered around the mean of the pooled Xr and Xu matrices ($Xr \cup Xu$). For dissimilarity computations based on diss_method = pls, the data is always centered.

scale
: a logical indicating if Xr (and Xu if provided) must be scaled. If Xu is provided the data is scaled based on the standard deviation of the the pooled Xr and Xu matrices ($Xr \cup Xu$). If center = TRUE, scaling is applied after centering.

documentation
: an optional character string that can be used to describe anything related to the mbl call (e.g. description of the input data). Default: character(). NOTE: his is an experimental argument.

...
: other arguments passed to the dissimilarity functions (ortho_diss, cor_diss, f_diss or sid).

Details

This function is a wrapper for ortho_diss, cor_diss, f_diss, sid. Check the documentation of these functions for further details.

Value

A list with the following components:

- dissimilarity: the resulting dissimilarity matrix.
- projection: an ortho_projection object. Only output if return_projection = TRUE and if diss_method = "pca", diss_method = "pca.nipals", diss_method = "pls" or diss_method = "mpls". This object contains the projection used to compute the dissimilarity matrix. In case of local dissimilarity matrices, the projection corresponds to the global projection used to select the neighborhoods (see ortho_diss function for further details).
- gh: a list containing the GH distances as well as the pls projection used to compute the GH.

Author(s)

Leonardo Ramirez-Lopez

References

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.

Westerhaus, M. 2014. Eastern Analytical Symposium Award for outstanding Wachievements in near infrared spectroscopy: my contributions to Wnear infrared spectroscopy. NIR news, 25(8), 16-20.

See Also

ortho_diss cor_diss f_diss sid.

Examples

```
library(prospectr)
data(NIRsoil)

# Filter the data using the first derivative with Savitzky and Golay
# smoothing filter and a window size of 11 spectral variables and a
# polynomial order of 4
sg <- savitzkyGolay(NIRsoil$spc, m = 1, p = 4, w = 15)

# 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)]

dsm_pca <- dissimilarity(
  Xr = Xr, Xu = Xu,
  diss_method = c("pca"),
  Yr = Yr, gh = TRUE,
  pc_selection = list("opc", 30),
  return_projection = TRUE
)
```

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

Description

Stable

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

```
f_diss(Xr, Xu = NULL, diss_method = "euclid",
       center = TRUE, scale = FALSE)
```

Arguments

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

Details

The results obtained for Euclidean dissimilarity are equivalent to those returned by the `stats::dist()` function, but are scaled differently. However, `f_diss` is considerably faster (which can be advantageous when computing dissimilarities for very large matrices). The final scaling of the dissimilarity

scores in `f_diss` where the number of variables is used to scale the squared dissimilarity scores. See the examples section for a comparison between `stats::dist()` and `f_diss`.

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

$$d(x_i, x_j)^2 = \sum (x_i - x_j) M^{-1} (x_i - x_j)^{\mathrm{T}}$$

$$d_{scaled}(x_i, x_j) = \sqrt{\frac{1}{p} d(x_i, x_j)^2}$$

where p is the number of variables in X, M is the identity matrix in the case of the Euclidean distance and the variance-covariance matrix of X 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 done by using a factorization of the inverse covariance matrix as $M^{-1} = W^T W$, 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 observations in the dataset is smaller than the number of variables.

For the computation of the Mahalanobis distance, the mentioned method is used.

The cosine dissimilarity c between two observations x_i and x_j is computed as follows:

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

where p is the number of variables of the observations. The function does not accept input data containing missing values. NOTE: The computed distances are divided by the number of variables/columns in `Xr`.

Value

a matrix of the computed dissimilarities.

Author(s)

Leonardo Ramirez-Lopez and Antoine Stevens

Examples

```
library(prospectr)
data(NIRsoil)

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

# Euclidean distances between all the observations in Xr
```

```
ed <- f_diss(Xr = Xr, diss_method = "euclid")

# Equivalence with the dist() fucntion of R base
ed_dist <- (as.matrix(dist(Xr))^2 / ncol(Xr))^0.5
round(ed_dist - ed, 5)

# Comparing the computational time
iter <- 20
tm <- proc.time()
for (i in 1:iter) {
  f_diss(Xr)
}
f_diss_time <- proc.time() - tm

tm_2 <- proc.time()
for (i in 1:iter) {
  dist(Xr)
}
dist_time <- proc.time() - tm_2

f_diss_time
dist_time

# Euclidean distances between observations in Xr and observations in Xu
ed_xr_xu <- f_diss(Xr, Xu)

# Mahalanobis distance computed on the first 20 spectral variables
md_xr_xu <- f_diss(Xr[, 1:20], Xu[, 1:20], "mahalanobis")

# Cosine dissimilarity matrix
cdiss_xr_xu <- f_diss(Xr, Xu, "cosine")
```

`get_predictions` *Extract predictions from an object of class* `mbl`

Description

Stable

Extract predictions from an object of class `mbl`

Usage

```
get_predictions(object)
```

Arguments

`object`	an object of class `mbl` as returned by `mbl`

Value

a data.table of predicted values according to either k or k_dist

Author(s)

Leonardo Ramirez-Lopez and Antoine Stevens

See Also

mbl

---

local_fit *Local fit functions*

Description

These functions define the way in which each local fit/prediction is done within each iteration in the mbl function.

Usage

```
local_fit_pls(pls_c, modified = FALSE, max_iter = 100, tol = 1e-6)

local_fit_wapls(min_pls_c, max_pls_c, modified = FALSE,
                max_iter = 100, tol = 1e-6)

local_fit_gpr(noise_variance = 0.001)
```

Arguments

pls_c	an integer indicating the number of pls components to be used in the local regressions when the partial least squares (local_fit_pls) method is used.
modified	a logical indicating whether the modified version of the pls algorithm (Shenk and Westerhaus, 1991 and Westerhaus, 2014). Default is FALSE.
max_iter	an integer indicating the maximum number of iterations in case tol is not reached. Defaul is 100.
tol	a numeric value indicating the convergence for calculating the scores. Default is 1-e6.
min_pls_c	an integer indicating the minimum number of pls components to be used in the local regressions when the weighted average partial least squares (local_fit_wapls) method is used. See details.
max_pls_c	integer indicating the maximum number of pls components to be used in the local regressions when the weighted average partial least squares (local_fit_wapls) method is used. See details.
noise_variance	a numeric value indicating the variance of the noise for Gaussian process local regressions (local_fit_gpr). Default is 0.001.

Details

These functions are used to indicate how to fit the regression models within the `mbl` function.

There are three possible options for performing these regressions:

- Partial least squares (pls, `local_fit_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.
- Weighted average pls (`local_fit_wapls`): This method was developed by Shenk et al. (1997) and it used as the regression method in the widely known LOCAL algorithm. 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 ensemble (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 reconstruction error of the unknown (or target) observation(s) when a total of j pls components are used and g_j is the root mean square of the squared regression coefficients corresponding to the jth pls component (see Shenk et al., 1997 for more details).
- Gaussian process with dot product covariance (`local_fit_gpr`): Gaussian process regression is a probabilistic and non-parametric Bayesian method. 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 (Rasmussen and Williams, 2006). The covariance function used in the implemented method is the dot product. The only parameter to be taken into account in this method is the noise. In this method, the process for predicting the response variable of a new sample (y_u) from its predictor variables (x_u) is carried out first by computing a prediction vector (A). It is derived from a reference/training observations congaing both a response vector (Y) and predictors (X) as follows:

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

  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_u is then done as follows:

$$\hat{y}_u = (x_u x_u^T) A$$

The `modified` argument in the pls methods (`local_fit_pls()` and `local_fit_wapls()`) is used to indicate if a modified version of the pls algorithm (modified pls or mpls) is to be used. The modified pls was proposed Shenk and Westerhaus (1991, see also Westerhaus, 2014) and it differs from the standard pls method in the way the weights of the predictors (used to compute the matrix of scores) are obtained. While pls uses the covariance between response(s) and predictors (and later their deflated versions corresponding at each pls component iteration) to obtain these weights, the modified pls uses the correlation as weights. The authors indicate that by using correlation, a larger potion of the response variable(s) can be explained.

Value

An object of class `local_fit` mirroring the input arguments.

Author(s)

Leonardo Ramirez-Lopez

References

Shenk, J. S., & Westerhaus, M. O. 1991. Populations structuring of near infrared spectra and modified partial least squares regression. Crop Science, 31(6), 1548-1555.

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.

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

Westerhaus, M. 2014. Eastern Analytical Symposium Award for outstanding Wachievements in near infrared spectroscopy: my contributions to Wnear infrared spectroscopy. NIR news, 25(8), 16-20.

See Also

mbl

Examples

```
local_fit_wapls(min_pls_c = 3, max_pls_c = 12)
```

---

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 predictor variables. For each observation in a prediction set, a specific local regression is carried out based on a subset of similar observations (nearest neighbors) selected from a reference set. The local model is then used to predict the response value of the target (prediction) observation. Therefore this function does not yield a global regression model.

Usage

```
mbl(Xr, Yr, Xu, Yu = NULL, k, k_diss, k_range, spike = NULL,
    method = local_fit_wapls(min_pls_c = 3, max_pls_c = min(dim(Xr), 15)),
    diss_method = "pca", diss_usage = "predictors", gh = TRUE,
    pc_selection = list(method = "opc", value = min(dim(Xr), 40)),
    control = mbl_control(), group = NULL, center = TRUE, scale = FALSE,
    verbose = TRUE, documentation = character(), seed = NULL, ...)
```

Arguments

Xr	a matrix of predictor variables of the reference data (observations in rows and variables in columns).
Yr	a numeric matrix of one column containing the values of the response variable corresponding to the reference data.
Xu	a matrix of predictor variables of the data to be predicted (observations in rows and variables in columns).
Yu	an optional matrix of one column containing the values of the response variable corresponding to the data to be predicted. Default is NULL.
k	a vector of integers specifying the sequence of k-nearest neighbors to be tested. Either k or k_diss must be specified. This vector will be automatically sorted into ascending order. If non-integer numbers are passed, they will be coerced to the next upper integers.
k_diss	a numeric vector specifying the sequence of dissimilarity thresholds to be tested for the selection of the nearest neighbors found in Xr around each observation in Xu. These thresholds depend on the corresponding dissimilarity measure specified in the object passed to control. Either k or k_diss must be specified.
k_range	an integer vector of length 2 which specifies the minimum (first value) and the maximum (second value) number of neighbors to be retained when the k_diss is given.
spike	an integer vector (with positive and/or negative values) indicating the indices of observations in Xr that must be either be forced into or avoided in the neighborhoods of every Xu observation. Default is NULL (i.e. no observations are forced or avoided). Note that this argument is not intended for increasing or reducing the neighborhood size which is only controlled by k or k_diss and k_range. By forcing observations into the neighborhood, some of the farthest observations may be forced out of the neighborhood. In contrast, by avoiding observations in the neighborhood, some of farthest observations may be included into the neighborhood. See details.
method	an object of class local_fit which indicates the type of regression to conduct at each local segment as well as additional parameters affecting this regression. See local_fit function.
diss_method	a character string indicating the spectral dissimilarity metric to be used in the selection of the nearest neighbors of each observation. Options are: • "pca" (Default): Mahalanobis distance computed on the matrix of scores of a Principal Component (PC) projection of Xr and Xu. PC projection is done using the singular value decomposition (SVD) algorithm. See ortho_diss function. • "pca.nipals": Mahalanobis distance computed on the matrix of scores of a Principal Component (PC) projection of Xr and Xu. PC projection is done using the non-linear iterative partial least squares (nipals) algorithm. See ortho_diss function. • "pls": Mahalanobis distance computed on the matrix of scores of a partial least squares projection of Xr and Xu. In this case, Yr is always required. See ortho_diss function.

- "cor": correlation coefficient between observations. See cor_diss function.
- "euclid": Euclidean distance between observations. See f_diss function.
- "cosine": Cosine distance between observations. See f_diss function.
- "sid": spectral information divergence between observations. See sid function.

Alternatively, a matrix of dissimilarities can also be passed to this argument. This matrix is supposed to be a user-defined matrix representing the dissimilarities between observations in Xr and Xu. When diss_usage = "predictors", this matrix must be squared (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 diss_usage is set to either "weights" or "none", it must be a matrix representing the dissimilarity of each observation in Xu to each observation in Xr. The number of columns of the input matrix must be equal to the number of rows in Xu and the number of rows equal to the number of rows in Xr.

diss_usage — a character string specifying how the dissimilarity information shall be used. The possible options are: "predictors", "weights" and "none" (see details below). Default is "predictors".

gh — a logical indicating if the global Mahalanobis distance (in the pls score space) between each observation and the pls mean (centre) must be computed. This metric is known as the GH distance in the literature. Note that this computation is based on the number of pls components determined by using the pc_selection argument. See details.

pc_selection — a list of length 2 used for the computation of GH (if gh = TRUE) as well as in the computation of the dissimilarity methods based on ortho_diss (i.e. when diss_method is one of: "pca", "pca.nipals" or "pls") or when gh = TRUE. This argument is used for optimizing the number of components (principal components or pls factors) to be retained for dissimilarity/distance computation purposes only (i.e not for regression). This list must contain two elements in the following order: method (a character indicating the method for selecting the number of components) and value (a numerical value that complements the selected method). The methods available are:

- "opc": optimized principal component selection based on Ramirez-Lopez et al. (2013a, 2013b). The optimal number of components (of set of observations) is the one for which its distance matrix minimizes the differences between the Yr value of each observation and the Yr value of its closest observation. In this case value must be a value (larger than 0 and below the minimum dimension of Xr or Xr and Xu combined) indicating the maximum number of principal components to be tested. See the ortho_projection function for more details.
- "cumvar": selection of the principal components based on a given cumulative amount of explained variance. In this case, value must be a value (larger than 0 and below or equal to 1) indicating the minimum amount of cumulative variance that the combination of retained components should explain.

- "var": selection of the principal components based on a given amount of explained variance. In this case, value must be a value (larger than 0 and below or equal to 1) indicating the minimum amount of variance that a single component should explain in order to be retained.
- "manual": for manually specifying a fix number of principal components. In this case, value must be a value (larger than 0 and below the minimum dimension of Xr or Xr and Xu combined). indicating the minimum amount of variance that a component should explain in order to be retained.

The list list(method = "opc", value = min(dim(Xr), 40)) is the default. Optionally, the pc_selection argument admits "opc" or "cumvar" or "var" or "manual" as a single character string. In such a case the default "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.

control — a list created with the mbl_control function which contains additional parameters that control some few aspects of the mbl function (cross-validation, parameter tuning, etc). The default list is as returned by mbl_control(). See the mbl_control function for more details.

group — an optional factor (or character vector vector that can be coerced to factor by as.factor) that assigns a group/class label to each observation in Xr (e.g. groups can be given by spectra collected from the same batch of measurements, from the same observation, from observations with very similar origin, etc). This is taken into account for internal leave-group-out cross validation for pls tuning (factor optimization) to avoid pseudo-replication. When one observation is selected for cross-validation, all observations of the same group are removed together and assigned to validation. The length of the vector must be equal to the number of observations in the reference/training set (i.e. nrow(Xr)). See details.

center — a logical if the predictor variables must be centred at each local segment (before regression). In addition, if TRUE, Xr and Xu will be centred for dissimilarity computations.

scale — a logical indicating if the predictor variables must be scaled to unit variance at each local segment (before regression). In addition, if TRUE, Xr and Xu will be scaled for dissimilarity computations.

verbose — a logical indicating whether or not to print a progress bar for each observation to be predicted. Default is TRUE. Note: In case parallel processing is used, these progress bars will not be printed.

documentation — an optional character string that can be used to describe anything related to the mbl call (e.g. description of the input data). Default: character(). NOTE: his is an experimental argument.

seed — an integer value containing the random number generator (RNG) state for random number generation. This argument can be used for reproducibility purposes (for random sampling) in the cross-validation results. Default is NULL, i.e. no RNG is applied.

... — further arguments to be passed to the dissimilarity function. See details.

Details

The argument spike can be used to indicate what reference observations in Xr must be kept in the neighborhood of every single Xu observation. If a vector of length m is passed to this argument, this means that the m original neighbors with the largest dissimilarities to the target observations will be forced out of the neighborhood. Spiking might be useful in cases where some reference observations are known to be somehow related to the ones in Xu and therefore might be relevant for fitting the local models. See Guerrero et al. (2010) for an example on the benefits of spiking.

The mbl function uses the dissimilarity function to compute the dissimilarities between Xr and Xu. The dissimilarity method to be used is specified in the diss_method argument. Arguments to dissimilarity as well as further arguments to the functions used inside dissimilarity (i.e. ortho_diss cor_diss f_diss sid) can be passed to those functions by using ....

The diss_usage argument is used to specify whether the dissimilarity information must be used within the local regressions and, if so, how. When diss_usage = "predictors" the local (square symmetric) dissimilarity matrix corresponding the selected neighborhood is used as source of additional predictors (i.e the columns of this local matrix are treated as predictor variables). In some cases this results in an improvement of the prediction performance (Ramirez-Lopez et al., 2013a). If diss_usage = "weights", the neighbors of the query point (xu_j) are weighted according to their dissimilarity to xu_j before 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 $xr_i \in$ neighbors 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 diss_usage = "none" is chosen the dissimilarity information is not used.

The global Mahalanobis distance (a.k.a GH) is computed based on the scores of a pls projection. A pls projection model is built with for {Yr}, {Xr} and this model is used to obtain the pls scores of the Xu observations. The Mahalanobis distance between each Xu observation in (the pls space) and the centre of Xr is then computed. The number of pls components is optimized based on the parameters passed to the pc_selection argument. In addition, the mbl function also reports the GH distance for the observations in Xr.

Some aspects of the mbl process, such as the type of internal validation, parameter tuning, what extra objects to return, permission for parallel execution, prediction limits, etc, can be specified by using the mbl_control function.

By using the group argument one can specify groups of observations that have something in common (e.g. observations with very similar origin). The purpose of group is to avoid biased cross-validation results due to pseudo-replication. This argument allows to select calibration points that are independent from the validation ones. In this regard, when validation_type = "local_cv"

(used in mbl_control function), then the p argument refers to the percentage of groups of observations (rather than single observations) to be retained in each sampling iteration at each local segment.

Value

a list of class mbl with the following components (sorted either by k or k_diss):

- call: the call to mbl.
- cntrl_param: the list with the control parameters passed to control.
- Xu_neighbors: a list containing two elements: a matrix of Xr indices corresponding to the neighbors of Xu and a matrix of dissimilarities between each Xu observation and its corresponding neighbor in Xr.
- dissimilarities: a list with the method used to obtain the dissimilarity matrices and the dissimilarity matrix corresponding to $D(Xr, Xu)$. This object is returned only if the return_dissimilarity argument in the control list was set to TRUE.
- n_predictions: the total number of observations predicted.
- gh: if gh = TRUE, a list containing the global Mahalanobis distance values for the observations in Xr and Xu as well as the results of the global pls projection object used to obtain the GH values.
- validation_results: a list of validation results for "local cross validation" (returned if the validation_type in control list was set to "local_cv"), "nearest neighbor validation" (returned if the validation_type in control list was set to "NNv") and "Yu prediction statistics" (returned if Yu was supplied).“
- results: a list of data tables containing the results of the predictions for each either k or k_diss. Each data table contains the following columns:
  - o_index: The index of the predicted observation.
  - k_diss: This column is only output if the k_diss argument is used. It indicates the corresponding dissimilarity threshold for selecting the neighbors.
  - k_original: This column is only output if the k_diss argument is used. It indicates the number of neighbors that were originally found when the given dissimilarity threshold is used.
  - k: This column indicates the final number of neighbors used.
  - npls: This column is only output if the pls regression method was used. It indicates the final number of pls components used.
  - min_pls: This column is only output if wapls regression method was used. It indicates the final number of minimum pls components used. If no optimization was set, it retrieves the original minimum pls components passed to the method argument.
  - max_pls: This column is only output if the wapls regression method was used. It indicates the final number of maximum pls components used. If no optimization was set, it retrieves the original maximum pls components passed to the method argument.
  - yu_obs: The input values given in Yu (the response variable corresponding to the data to be predicted). If Yu = NULL, then NAs are retrieved.
  - pred: The predicted Yu 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_Xr: The index of the nearest neighbor found in Xr.
  - index_farthest_in_Xr: The index of the farthest neighbor found in Xr.
  - y_nearest: The reference value (Yr) corresponding to the nearest neighbor found in Xr.
  - y_nearest_pred: This column is only output if the validation method in the object passed to control was set to "NNv". It represents the predicted value of the nearest neighbor observation found in Xr. This prediction come from model fitted with the remaining observations in the neighborhood of the target observation in Xu.
  - loc_rmse_cv: This column is only output if the validation method in the object passed to control was set to 'local_cv'. It represents the RMSE of the cross-validation computed for the neighborhood of the target observation in Xu.
  - loc_st_rmse_cv: This column is only output if the validation method in the object passed to control was set to 'local_cv'. It represents the standardized RMSE of the cross-validation computed for the neighborhood of the target observation in Xu.
  - dist_nearest: The distance to the nearest neighbor.
  - dist_farthest: The distance to the farthest neighbor.
  - loc_n_components: This column is only output if the dissimilarity method used is one of "pca", "pca.nipals" or "pls" and in addition the dissimilarities are requested to be computed locally by passing .local = TRUE to the mbl function. See .local argument in the ortho_diss function.

- seed: a value mirroring the one passed to seed.
- documentation: a character string mirroring the one provided in the documentation argument.

When the k_diss argument is used, the printed results show a table with a column named 'p_bounded. It represents the percentage of observations 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.

Guerrero, C., Zornoza, R., Gómez, I., Mataix-Beneyto, J. 2010. Spiking of NIR regional models using observations from target sites: Effect of model size on prediction accuracy. Geoderma, 158(1-2), 66-77.

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 data sets. 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

mbl_control, f_diss, cor_diss, sid, ortho_diss, search_neighbors, local_fit

Examples

```
library(prospectr)
data(NIRsoil)

# Proprocess the data using detrend plus first derivative with Savitzky and
# Golay smoothing filter
sg_det <- savitzkyGolay(
  detrend(NIRsoil$spc,
    wav = as.numeric(colnames(NIRsoil$spc))
  ),
  m = 1,
  p = 1,
  w = 7
)

NIRsoil$spc_pr <- sg_det

# split into training and testing sets
test_x <- NIRsoil$spc_pr[NIRsoil$train == 0 & !is.na(NIRsoil$CEC), ]
test_y <- NIRsoil$CEC[NIRsoil$train == 0 & !is.na(NIRsoil$CEC)]

train_y <- NIRsoil$CEC[NIRsoil$train == 1 & !is.na(NIRsoil$CEC)]
train_x <- NIRsoil$spc_pr[NIRsoil$train == 1 & !is.na(NIRsoil$CEC), ]

# 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
my_control <- mbl_control(validation_type = "NNv")

## The neighborhood sizes to test
ks <- seq(40, 140, by = 20)

sbl <- mbl(
  Xr = train_x,
  Yr = train_y,
```

```
  Xu = test_x,
  k = ks,
  method = local_fit_gpr(),
  control = my_control,
  scale = TRUE
)
sbl
plot(sbl)
get_predictions(sbl)

# Example 1.2
# If Yu is actually known...
sbl_2 <- mbl(
  Xr = train_x,
  Yr = train_y,
  Xu = test_x,
  Yu = test_y,
  k = ks,
  method = local_fit_gpr(),
  control = my_control
)
sbl_2
plot(sbl_2)

# Example 2
# the LOCAL algorithm (Shenk et al., 1997)
local_algorithm <- mbl(
  Xr = train_x,
  Yr = train_y,
  Xu = test_x,
  Yu = test_y,
  k = ks,
  method = local_fit_wapls(min_pls_c = 3, max_pls_c = 15),
  diss_method = "cor",
  diss_usage = "none",
  control = my_control
)
local_algorithm
plot(local_algorithm)

# Example 3
# A variation of the LOCAL algorithm (using the optimized pc
# dissmilarity matrix) and dissimilarity matrix as source of
# additional preditors
local_algorithm_2 <- mbl(
  Xr = train_x,
  Yr = train_y,
  Xu = test_x,
  Yu = test_y,
  k = ks,
  method = local_fit_wapls(min_pls_c = 3, max_pls_c = 15),
  diss_method = "pca",
  diss_usage = "predictors",
```

```
  control = my_control
)
local_algorithm_2
plot(local_algorithm_2)

# Example 4
# Running the mbl function in parallel with example 2

n_cores <- 2

if (parallel::detectCores() < 2) {
  n_cores <- 1
}

# Alternatively:
# n_cores <- parallel::detectCores() - 1
# if (n_cores == 0) {
#  n_cores <- 1
# }

library(doParallel)
clust <- makeCluster(n_cores)
registerDoParallel(clust)

# Alernatively:
# library(doSNOW)
# clust <- makeCluster(n_cores, type = "SOCK")
# registerDoSNOW(clust)
# getDoParWorkers()

local_algorithm_par <- mbl(
  Xr = train_x,
  Yr = train_y,
  Xu = test_x,
  Yu = test_y,
  k = ks,
  method = local_fit_wapls(min_pls_c = 3, max_pls_c = 15),
  diss_method = "cor",
  diss_usage = "none",
  control = my_control
)
local_algorithm_par

registerDoSEQ()
try(stopCluster(clust))

# Example 5
# Using local pls distances
with_local_diss <- mbl(
  Xr = train_x,
  Yr = train_y,
  Xu = test_x,
  Yu = test_y,
```

```
  k = ks,
  method = local_fit_wapls(min_pls_c = 3, max_pls_c = 15),
  diss_method = "pls",
  diss_usage = "predictors",
  control = my_control,
  .local = TRUE,
  pre_k = 150,
)
with_local_diss
plot(with_local_diss)
```

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

Description

Maturing

This function is used to further control some aspects of the memory-based learning process in the `mbl` function.

Usage

```
mbl_control(
  return_dissimilarity = FALSE,
  validation_type = c("NNv", "local_cv"),
  tune_locally = TRUE,
  number = 10,
  p = 0.75,
  range_prediction_limits = TRUE,
  progress = TRUE,
  allow_parallel = TRUE
)
```

Arguments

`return_dissimilarity`
: a logical indicating if the dissimilarity matrix between `Xr` and `Xu` must be returned.

`validation_type`
: 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 `"local_cv"`. Alternatively `"none"` can be used when cross-validation is not required (see details below).

`tune_locally`
: a logical. It only applies when `validation_type = "local_cv"` and "pls" or "wapls" fitting algorithms are used. If `TRUE`, the parameters of the local pls-based models (i.e. pls factors for the "pls" method and minimum and maximum pls factors for the "wapls" method). Default is `TRUE`.

`number` an integer indicating the number of sampling iterations at each local segment when `"local_cv"` is selected in the `validation_type` argument. Default is 10.

`p` a numeric value indicating the percentage of observations to be retained at each sampling iteration at each local segment when `"local_cv"` is selected in the `validation_type` argument. Default is 0.75 (75 %).

`range_prediction_limits`
a logical. It indicates whether the prediction limits at each local regression are determined by the range of the response variable within each neighborhood. When the predicted value is outside this range, it will be automatically replaced with the value of the nearest range value. If `FALSE`, no prediction limits are imposed. Default is `TRUE`.

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

`allow_parallel` a logical indicating if parallel execution is allowed. If `TRUE`, this parallelism is applied to the loop in `mbl` in which each iteration takes care of a single observation in `Xu`. The parallelization of this for loop is implemented using the foreach function of the **foreach** package. Default is `TRUE`.

Details

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

- Leave-nearest-neighbor-out cross-validation (`"NNv"`): From the group of neighbors of each observation to be predicted, the nearest observation (i.e. the most similar observation) is excluded and then a local model is fitted using the remaining neighbors. This model is then used to predict the value of the target response variable of the nearest observation. 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 `"local_cv"`.
- Local leave-group-out cross-validation (`"local_cv"`): The group of neighbors of each observation 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 neighbors. The selected local subset is used as local validation subset and the remaining observations 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 `number` argument and the size of the subsets is controlled by the `p` argument which indicates the percentage of observations 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 selected along with `"NNv"` and/or `"local_cv"`, then it will be ignored and the respective validation(s) will be carried out.

Value

a `list` mirroring 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 data sets. 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

f_diss, cor_diss, sid, ortho_diss, mbl

Examples

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

---

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

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

```
ortho_diss(Xr, Xu = NULL,
           Yr = NULL,
           pc_selection = list(method = "var", value = 0.01),
           diss_method = "pca",
           .local = FALSE,
           pre_k,
           center = TRUE,
           scale = FALSE,
           compute_all = FALSE,
           return_projection = FALSE,
           allow_parallel = TRUE, ...)
```

Arguments

Xr	a matrix containing n reference observations rows and p variablescolumns.
Xu	an optional matrix containing data of a second set of observations with p variables/columns.
Yr	a matrix of n rows and one or more columns (variables) with side information corresponding to the observations in Xr (e.g. response variables). It can be numeric with multiple variables/columns, or character with one single column. This argument is required if: • diss_method == 'pls': Yr is required to project the variables to orthogonal directions such that the covariance between the extracted pls components and Yr is maximized. • pc_selection$method == 'opc': Yr is required to optimize the number of components. The optimal number of projected components is the one for which its distance matrix minimizes the differences between the Yr value of each observation and the Yr value of its closest observation. See sim_eval.
pc_selection	a list of length 2 which specifies the method to be used for optimizing the number of components (principal components or pls factors) to be retained. This list must contain two elements (in the following order): method (a character indicating the method for selecting the number of components) and value (a numerical value that complements the selected method). The methods available are: • "opc": optimized principal component selection based on Ramirez-Lopez et al. (2013a, 2013b). The optimal number of components (of a given set of observations) is the one for which its distance matrix minimizes the differences between the Yr value of each observation and the Yr value of its closest observation. In this case, value must be a value (larger than 0 and below min(nrow(Xr) + nrow(Xu), ncol(Xr)) indicating the maximum number of principal components to be tested. See the ortho_projection function for more details. • "cumvar": selection of the principal components based on a given cumulative amount of explained variance. In this case, value must be a value (larger than 0 and below or equal to 1) indicating the minimum amount of cumulative variance that the combination of retained components should explain. • "var": selection of the principal components based on a given amount of explained variance. In this case, value must be a value (larger than 0 and below or equal to 1) indicating the minimum amount of variance that a single component should explain in order to be retained. • "manual": for manually specifying a fix number of principal components. In this case, value must be a value (larger than 0 and below the minimum dimension of Xr or Xr and Xu combined). indicating the minimum amount of variance that a component should explain in order to be retained. Default is list(method = "var", value = 0.01). Optionally, the pc_selection argument admits "opc" or "cumvar" or "var" or "manual" as a single character string. In such case, the default "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.

diss_method — a character value indicating the type of projection on which the dissimilarities must be computed. This argument is equivalent to method argument in the ortho_projection function. 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.
- "pls": partial least squares.
- "mpls": modified partial least squares (Shenk and Westerhaus, 1991 and Westerhaus, 2014).

See the ortho_projection function for further details on the projection methods.

.local — a logical indicating whether or not to compute the dissimilarities locally (i.e. projecting locally the data) by using the pre_k nearest neighbor observations of each target observation. Default is FALSE. See details.

pre_k — if .local = TRUE a numeric integer value which indicates the number of nearest neighbors to (pre-)retain for each observation to compute the (local) orthogonal dissimilarities to each observation in its neighborhhod.

center — a logical indicating if the Xr and Xu must be centered. If Xu is provided the data is centered around the mean of the pooled Xr and Xu matrices ($Xr \cup Xu$). For dissimilarity computations based on pls, the data is always centered for the projections.

scale — a logical indicating if the Xr and Xu must be scaled. If Xu is provided the data is scaled based on the standard deviation of the the pooled Xr and Xu matrices ($Xr \cup Xu$). if center = TRUE, scaling is applied after centering.

compute_all — a logical. In case Xu is specified it indicates whether or not the distances between all the elements resulting from the pooled Xr and Xu matrices ($Xr \cup Xu$ must be computed).

return_projection — a logical. If TRUE the ortho_projection object on which the dissimilarities are computed will be returned. Default is FALSE. Note that for .local = TRUE only the initial projection is returned (i.e. local projections are not).

allow_parallel — a logical (default TRUE). It allows parallel computing of the local distance matrices (i.e. when .local = TRUE). This is done via foreach function of the 'foreach' package.

... — additional arguments to be passed to the ortho_projection function.

Details

When .local = TRUE, first a global dissimilarity matrix is computed based on the parameters specified. Then, by using this matrix for each target observation, a given set of nearest neighbors (pre_k) are identified. These neighbors (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 observation and its neighbors is recomputed. A missing value is assigned to the observations that do not belong to this set of neighbors (non-neighbor observations). 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 observations 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.

In the cases where `"Yr"` is required to compute the dissimilarities and if `.local = TRUE`, care must be taken as some neighborhoods might not have enough observations with non-missing `"Yr"` values, which might retrieve unreliable dissimilarity computations.

If `"opc"` or `"manual"` are used in `pc_selection$method` and `.local = TRUE`, the minimum number of observations with non-missing `"Yr"` values at each neighborhood is determined by `pc_selection$value` (i.e. the maximum number of components to compute).

Value

a `list` of class `ortho_diss` with the following elements:

- `n_components`: the number of components (either principal components or partial least squares components) used for computing the global dissimilarities.
- `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.
- `local_n_components`: if `.local = TRUE`, a data.table which specifies the number of local components (either principal components or partial least squares components) used for computing the dissimilarity between each target observation and its neighbor observations.
- `dissimilarity`: the computed dissimilarity matrix. If `.local = FALSE` a distance matrix. If `.local = TRUE` a matrix of class `local_ortho_diss`. In this case, each column represent the dissimilarity between a target observation and its neighbor observations.
- `projection`: if `return_projection = TRUE`, an `ortho_projection` object.

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 data sets. 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

`ortho_projection`, `sim_eval`

Examples

```
library(prospectr)
data(NIRsoil)

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

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

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

# Computation of the orthogonal dissimilarity matrix using the
# default parameters
pca_diss <- ortho_diss(Xr, Xu)

# Computation of a principal component dissimilarity matrix using
# the "opc" method for the selection of the principal components
pca_diss_optim <- ortho_diss(
  Xr, Xu, Yr,
  pc_selection = list("opc", 40),
  compute_all = TRUE
)

# Computation of a partial least squares (PLS) dissimilarity
# matrix using the "opc" method for the selection of the PLS
# components
pls_diss_optim <- ortho_diss(
  Xr = Xr, Xu = Xu,
  Yr = Yr,
  pc_selection = list("opc", 40),
  diss_method = "pls"
)
```

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

Description

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

Usage

```
ortho_projection(Xr, Xu = NULL,
```

```
                  Yr = NULL,
                  method = "pca",
                  pc_selection = list(method = "var", value = 0.01),
                  center = TRUE, scale = FALSE, ...)

pc_projection(Xr, Xu = NULL, Yr = NULL,
              pc_selection = list(method = "var", value = 0.01),
              center = TRUE, scale = FALSE,
              method = "pca",
              tol = 1e-6, max_iter = 1000, ...)

pls_projection(Xr, Xu = NULL, Yr,
               pc_selection = list(method = "opc", value = min(dim(Xr), 40)),
               scale = FALSE, method = "pls",
               tol = 1e-6, max_iter = 1000, ...)

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

Arguments

Xr	a matrix of observations.
Xu	an optional matrix containing data of a second set of observations.
Yr	if the method used in the pc_selection argument is "opc" or if method = "pls", then it must be a matrix containing the side information corresponding to the spectra in Xr. It is equivalent to the side_info parameter of the sim_eval function. In case method = "pca", a matrix (with one or more continuous variables) can also be used as input. The root mean square of differences (rmsd) is used for assessing the similarity between the observations and their corresponding most similar observations in terms of the side information provided. A single discrete variable of class factor can also be passed. In that case, the kappa index is used. See sim_eval 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. • "pls": partial least squares. • "mpls": modified partial least squares. See details.
pc_selection	a list of length 2 which specifies the method to be used for optimizing the number of components (principal components or pls factors) to be retained. This list must contain two elements (in the following order): method (a character indicating the method for selecting the number of components) and value (a numerical value that complements the selected method). The methods available are: • "opc": optimized principal component selection based on Ramirez-Lopez et al. (2013a, 2013b). The optimal number of components of a given set of

observations is the one for which its distance matrix minimizes the differences between the `Yr` value of each observation and the `Yr` value of its closest observation. In this case `value` must be a value (larger than 0 and below `min(nrow(Xr) + nrow(Xu), ncol(Xr))` indicating the maximum number of principal components to be tested. See details.

- `"cumvar"`: selection of the principal components based on a given cumulative amount of explained variance. In this case, `value` must be a value (larger than 0 and below or equal to 1) indicating the minimum amount of cumulative variance that the combination of retained components should explain.
- `"var"`: selection of the principal components based on a given amount of explained variance. In this case, `value` must be a value (larger than 0 and below or equal to 1) indicating the minimum amount of variance that a single component should explain in order to be retained.
- `"manual"`: for manually specifying a fix number of principal components. In this case, `value` must be a value (larger than 0 and below the minimum dimension of `Xr` or `Xr` and `Xu` combined). indicating the minimum amount of variance that a component should explain in order to be retained.

The list `list(method = "var", value = 0.01)` is the default. Optionally, the `pc_selection` argument admits `"opc"` or `"cumvar"` or `"var"` or `"manual"` as a single character string. In such a case the default `"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 `Xu` if specified) must be centered. If `Xu` is specified the data is centered on the basis of $Xr \cup Xu$. NOTE: This argument only applies to the principal components projection. For pls projections the data is always centered.

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

`...` additional arguments to be passed to `pc_projection` or `pls_projection`.

`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 one variable.

`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 `"ortho_projection"`.

`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 algorithm used is the singular value decomposition in which a given data matrix (X) is factorized as follows:

$$X = UDV^T$$

where U and V are orthogonal matrices, being the left and right singular vectors of X respectively, 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 nonlinear 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 is used by default. Details on the "nipals" algorithm are presented in Martens (1991). Another method called modified pls (`'mpls'`) can also be used. The modified pls was proposed Shenk and Westerhaus (1991, see also Westerhaus, 2014) and it differs from the standard pls method in the way the weights of the `Xr` (used to compute the matrix of scores) are obtained. While pls uses the covariance between `Yr` and `Xr` (and later their deflated versions corresponding at each pls component iteration) to obtain these weights, the modified pls uses the correlation as weights. The authors indicate that by using correlation, a larger potion of the response variable(s) can be explained.

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 in this side information variable are compared against the side information values of their most spectrally similar observations (closest `Xr` observation). 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 `sim_eval` function is used. Note that for the `"opc"` method `Yr` is required (i.e. the side information of the observations).

Value

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

- `scores`: a matrix of scores corresponding to the observations in `Xr` (and `Xu` if it was provided). The components retrieved correspond to the ones optimized or specified.
- `X_loadings`: a matrix of loadings corresponding to the explanatory variables. The components retrieved correspond to the ones optimized or specified.
- `Y_loadings`: a matrix of partial least squares loadings corresponding to `Yr`. The components retrieved correspond to the ones optimized or specified. 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.
- `projection_mat`: 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 list with information on the original variance and the explained variances. This list contains a matrix indicating the amount of variance explained by each component (var), the ratio between explained variance by each single component and the original variance (explained_var) and the cumulative ratio of explained variance (cumulative_explained_var). The amount of variance explained by each component is computed by multiplying its score vector by its corresponding loading vector and calculating the variance of the result. These values are computed based on the observations 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 Xu (if it applies) since both matrices are employed in the computation of the projection matrix (loadings in this case).

- sdv: the standard deviation of the retrieved scores. This vector can be different from the "sd" in variance.
- n_components: the number of components (either principal components or partial least squares components) used for computing the global dissimilarity scores.
- opc_evaluation: a matrix 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 pc_selection argument. See the sim_eval function for more details.
- method: the ortho_projection method used.

predict.ortho_projection, 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 data sets. 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.

Shenk, J. S., & Westerhaus, M. O. 1991. Populations structuring of near infrared spectra and modified partial least squares regression. Crop Science, 31(6), 1548-1555.

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.

Westerhaus, M. 2014. Eastern Analytical Symposium Award for outstanding Wachievements in near infrared spectroscopy: my contributions to Wnear infrared spectroscopy. NIR news, 25(8), 16-20.

See Also

ortho_diss, sim_eval, mbl

Examples

```
library(prospectr)
data(NIRsoil)

# Proprocess the data using detrend plus first derivative with Savitzky and
# Golay smoothing filter
sg_det <- savitzkyGolay(
  detrend(NIRsoil$spc,
    wav = as.numeric(colnames(NIRsoil$spc))
  ),
  m = 1,
  p = 1,
  w = 7
)
NIRsoil$spc_pr <- sg_det

# split into training and testing sets
test_x <- NIRsoil$spc_pr[NIRsoil$train == 0 & !is.na(NIRsoil$CEC), ]
test_y <- NIRsoil$CEC[NIRsoil$train == 0 & !is.na(NIRsoil$CEC)]

train_y <- NIRsoil$CEC[NIRsoil$train == 1 & !is.na(NIRsoil$CEC)]
train_x <- NIRsoil$spc_pr[NIRsoil$train == 1 & !is.na(NIRsoil$CEC), ]

# A principal component analysis using 5 components
pca_projected <- ortho_projection(train_x, pc_selection = list("manual", 5))
pca_projected

# A principal components projection using the "opc" method
# for the selection of the optimal number of components
pca_projected_2 <- ortho_projection(
  Xr = train_x, Xu = test_x, Yr = train_y,
  method = "pca",
  pc_selection = list("opc", 40)
)
pca_projected_2
plot(pca_projected_2)

# A partial least squares projection using the "opc" method
# for the selection of the optimal number of components
pls_projected <- ortho_projection(
  Xr = train_x, Xu = test_x, Yr = train_y,
  method = "pls",
  pc_selection = list("opc", 40)
)
pls_projected
plot(pls_projected)

# A partial least squares projection using the "cumvar" method
# for the selection of the optimal number of components
pls_projected_2 <- ortho_projection(
  Xr = train_x, Xu = test_x, Yr = train_y,
  method = "pls",
```

```
  pc_selection = list("cumvar", 0.99)
)
```

`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", "gh"), param = "rmse", pls_c = 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 `"gh"` (for plotting the pls scores used to compute the GH distance. See details).
`param`	a character string indicating what validation statistics shall be plotted. The following options are available: `"rmse"`, `"st_rmse"` or `"r2"`. These options only available if the `mbl` object contains validation results.
`pls_c`	a numeric vector of length one or two indicating the pls factors to be plotted. Default is `c(1, 2)`. It is only available if `"gh"` is specified in the `g` argument.
`...`	some arguments to be passed to the plot methods.

Details

For plotting the pls scores from the pls score matrix (of more than one column), this matrix is first transformed from the Euclidean space to the Mahalanobis space. This is done by multiplying the score matrix by the root square of its covariance matrix. The root square of this matrix is estimated using a singular value decomposition.

Author(s)

Leonardo Ramirez-Lopez and Antoine Stevens

See Also

`mbl`

Examples

```
library(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 <- mbl_control(validation_type = "NNv")

ex_1 <- mbl(
  Yr = Yr, Xr = Xr, Xu = Xu,
  diss_method = "cor",
  diss_usage = "none",
  gh = TRUE,
  mblCtrl = ctrl,
  k = seq(50, 250, 30)
)

plot(ex_1)
plot(ex_1, g = "gh", pls_c = c(2, 3))
```

---

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

---

Description

Plots objects of class ortho_projection

Usage

```
## S3 method for class 'ortho_projection'
plot(x, col = "dodgerblue", ...)
```

Arguments

x	an object of class ortho_projection (as returned by ortho_projection).
col	the color of the plots (default is "dodgerblue")
...	arguments to be passed to methods.

Author(s)

Leonardo Ramirez-Lopez and Antoine Stevens

See Also

ortho_projection

---

search_neighbors *A function for searching in a given reference set the neighbors of another given set of observations (search_neighbors)*

Description

This function searches in a reference set the neighbors of the observations provided in another set.

Usage

```
search_neighbors(Xr, Xu, diss_method = c("pca", "pca.nipals", "pls", "mpls",
                                         "cor", "euclid", "cosine", "sid"),
                 Yr = NULL, k, k_diss, k_range, spike = NULL,
                 pc_selection = list("var", 0.01),
                 return_projection = FALSE, return_dissimilarity = FALSE,
                 ws = NULL,
                 center = TRUE, scale = FALSE,
                 documentation = character(), ...)
```

Arguments

Xr	a matrix of reference (spectral) observations where the neighbor search is to be conducted. See details.
Xu	an optional matrix of (spectral) observations for which its neighbors are to be searched in Xr. Default is NULL. See details.
diss_method	a character string indicating the spectral dissimilarity metric to be used in the selection of the nearest neighbors of each observation.

- "pca": Mahalanobis distance computed on the matrix of scores of a Principal Component (PC) projection of Xr (and Xu if supplied). PC projection is done using the singular value decomposition (SVD) algorithm. See ortho_diss function.
- "pca.nipals": Mahalanobis distance computed on the matrix of scores of a Principal Component (PC) projection of Xr (and Xu if supplied). PC projection is done using the non-linear iterative partial least squares (niapls) algorithm. See ortho_diss function.
- "pls": Mahalanobis distance computed on the matrix of scores of a partial least squares projection of Xr (and Xu if supplied). In this case, Yr is always required. See ortho_diss function.

- "mpls": Mahalanobis distance computed on the matrix of scores of a modified partial least squares projection (Shenk and Westerhaus, 1991; Westerhaus, 2014) of Xr (and Xu if provided). In this case, Yr is always required. See ortho_diss function.
- "cor": correlation coefficient between observations. See cor_diss function.
- "euclid": Euclidean distance between observations. See f_diss function.
- "cosine": Cosine distance between observations. See f_diss function.
- "sid": spectral information divergence between observations. See sid function.

Yr — a numeric matrix of n observations used as side information of Xr for the ortho_diss methods (i.e. pca, pca.nipals or pls). It is required when:

- diss_method = "pls"
- diss_method = "pca" with "opc" used as the method in the pc_selection argument. See ortho_diss().

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

k_diss — an integer value indicating a dissimilarity treshold. For each observation in Xu, its nearest neighbors in Xr are selected as those for which their dissimilarity to Xu is below this k_diss threshold. This treshold depends on the corresponding dissimilarity metric specified in diss_method. Either k or k_diss must be specified.

k_range — an integer vector of length 2 which specifies the minimum (first value) and the maximum (second value) number of neighbors to be retained when the k_diss is given.

spike — a vector of integers (with positive and/or negative values) indicating what observations in Xr (and Yr) must be forced into or avoided in the neighborhoods.

pc_selection — a list of length 2 to be passed onto the ortho_diss methods. It is required if the method selected in diss_method is any of "pca", "pca.nipals" or "pls". This argument is used for optimizing the number of components (principal components or pls factors) to be retained. This list must contain two elements in the following order: method (a character indicating the method for selecting the number of components) and value (a numerical value that complements the selected method). The methods available are:

- "opc": optimized principal component selection based on Ramirez-Lopez et al. (2013a, 2013b). The optimal number of components (of set of observations) is the one for which its distance matrix minimizes the differences between the Yr value of each observation and the Yr value of its closest observation. In this case value must be a value (larger than 0 and below the minimum dimension of Xr or Xr and Xu combined) indicating the maximum number of principal components to be tested. See the ortho_projection function for more details.
- "cumvar": selection of the principal components based on a given cumulative amount of explained variance. In this case, value must be a value (larger than 0 and below or equal to 1) indicating the minimum amount of

cumulative variance that the combination of retained components should explain.

- `"var"`: selection of the principal components based on a given amount of explained variance. In this case, `value` must be a value (larger than 0 and below or equal to 1) indicating the minimum amount of variance that a single component should explain in order to be retained.
- `"manual"`: for manually specifying a fix number of principal components. In this case, `value` must be a value (larger than 0 and below the minimum dimension of `Xr` or `Xr` and `Xu` combined) indicating the minimum amount of variance that a component should explain in order to be retained.

The default is `list(method = "var", value = 0.01)`.

Optionally, the `pc_selection` argument admits `"opc"` or `"cumvar"` or `"var"` or `"manual"` as a single character string. In such a case the default `"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.

`return_projection`
: a logical indicating if the projection(s) must be returned. Projections are used if the `ortho_diss` methods are called (i.e. `method = "pca"`, `method = "pca.nipals"` or `method = "pls"`).

`return_dissimilarity`
: a logical indicating if the dissimilarity matrix used for neighbor search must be returned.

`ws`
: an odd integer value which specifies the window size, when `diss_method = cor` (`cor_diss` method) for moving correlation dissimilarity. If `ws = NULL` (default), 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 `cor_diss` function.

`center`
: a logical indicating if the `Xr` and `Xu` matrices must be centered. If `Xu` is provided the data is centered around the mean of the pooled `Xr` and `Xu` matrices ($Xr \cup Xu$). For dissimilarity computations based on `diss_method = pls`, the data is always centered.

`scale`
: a logical indicating if the `Xr` and `Xu` matrices must be scaled. If `Xu` is provided the data is scaled based on the standard deviation of the the pooled `Xr` and `Xu` matrices ($Xr \cup Xu$). If `center = TRUE`, scaling is applied after centering.

`documentation`
: an optional character string that can be used to describe anything related to the `mbl` call (e.g. description of the input data). Default: `character()`. NOTE: his is an experimental argument.

`...`
: further arguments to be passed to the `dissimilarity` function. See details.

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 observations (i.e. observations that are not neighbors of a particular target set). For example, this fucntion can be used to reduce the size of the reference set before before running the `mbl` function.

This function uses the dissimilarity fucntion to compute the dissimilarities between Xr and Xu. Arguments to dissimilarity as well as further arguments to the functions used inside dissimilarity (i.e. ortho_diss cor_diss f_diss sid) can be passed to those functions as additional arguments (i.e. ...).

If no matrix is passed to Xu, the neighbor search is conducted for the observations in Xr that are found whiting that matrix. If a matrix is passed to Xu, the neighbors of Xu are searched in the Xr matrix.

Value

a list containing the following elements:

- neighbors_diss: a matrix of the Xr dissimilarity scores corresponding to the neighbors of each Xr observation (or Xu observation, in case Xu was supplied). The neighbor dissimilarity scores are organized by columns and are sorted in ascending order.
- neighbors: a matrix of the Xr indices corresponding to the neighbors of each observation in Xu. The neighbor indices are organized by columns and are sorted in ascending order by their dissimilarity score.
- unique_neighbors: a vector of the indices in Xr identified as neighbors of any observation in Xr (or in Xu, in case it was supplied). This is obtained by converting the neighbors matrix into a vector and applying the unique function.
- k_diss_info: a data.table that is returned only if the k_diss argument was used. It comprises three columns, the first one (Xr_index or Xu_index) indicates the index of the observations in Xr (or in Xu, in case it was suppplied), the second column (n_k) indicates the number of neighbors found in Xr and the third column (final_n_k) indicates the final number of neighbors selected bounded by k_range. argument.
- dissimilarity: If return_dissimilarity = TRUE the dissimilarity object used (as computed by the dissimilarity function.
- projection: an ortho_projection object. Only output if return_projection = TRUE and if diss_method = "pca", diss_method = "pca.nipals" or diss_method = "pls".
  This object contains the projection used to compute the dissimilarity matrix. In case of local dissimilarity matrices, the projection corresponds to the global projection used to select the neighborhoods. (see ortho_diss function for further details).

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 data sets. 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

dissimilarity ortho_diss cor_diss f_diss sid mbl

Examples

```
library(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 neighbor observations using the correlation dissimilarity and
# default parameters
# (In this example all the observations in Xr belong at least to the
# first 100 neighbors of one observation in Xu)
ex1 <- search_neighbors(
  Xr = Xr, Xu = Xu,
  diss_method = "cor",
  k = 40
)

# Identify the neighbor observations using principal component (PC)
# and partial least squares (PLS) dissimilarities, and using the "opc"
# approach for selecting the number of components
ex2 <- search_neighbors(
  Xr = Xr, Xu = Xu,
  diss_method = "pca",
  Yr = Yr, k = 50,
  pc_selection = list("opc", 40),
  scale = TRUE
)

# Observations that do not belong to any neighborhood
seq(1, nrow(Xr))[!seq(1, nrow(Xr)) %in% ex2$unique_neighbors]

ex3 <- search_neighbors(
  Xr = Xr, Xu = Xu,
  diss_method = "pls",
  Yr = Yr, k = 50,
  pc_selection = list("opc", 40),
  scale = TRUE
)
# Observations that do not belong to any neighborhood
```

```
seq(1, nrow(Xr))[!seq(1, nrow(Xr)) %in% ex3$unique_neighbors]

# Identify the neighbor observations using local PC dissimialrities
# Here, 150 neighbors are used to compute a local dissimilarity matrix
# and then this matrix is used to select 50 neighbors
ex4 <- search_neighbors(
  Xr = Xr, Xu = Xu,
  diss_method = "pls",
  Yr = Yr, k = 50,
  pc_selection = list("opc", 40),
  scale = TRUE,
  .local = TRUE,
  pre_k = 150
)
```

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

Description

Experimental

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

Usage

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

Arguments

`Xr`	a matrix containing the spectral (reference) data.
`Xu`	an optional matrix containing the spectral data of a second set of observations.
`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 variables. See details.

center	a logical indicating if the computations must be carried out on the centred X and Xu (if specified) matrices. If mode = "feature" centring is not carried out since this option does not accept negative values which are generated after centring the matrices. Default is FALSE. See details.
scale	a logical indicating if the computations must be carried out on the variance scaled X and Xu (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 bw.nrd0). 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 Xu is specified the bandwidth is computed as bw.nrd0(as.vector(rbind(X, Xu))).
reg	a numerical value larger 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 observations in the datasets are compared based on the kullback-leibler divergence algorithm. When mode = "feature", the kullback-leibler divergence between all the observations 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_i 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

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

$$Xu_s = \frac{Xu - min(X, Xu)}{max(X, Xu) - min(X, Xu)}$$

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

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

```
library(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 observations in Xr
xr_sid <- sid(Xr)
xr_sid

# Example 2
# Compute the SID distance between the observations in Xr and the observations
# in Xu
xr_xu_sid <- sid(Xr, Xu)
xr_xu_sid
```

---

sim_eval *A function for evaluating dissimilarity matrices (sim_eval)*

Description

Stable

This function searches for the most similar observation (closest neighbor) of each observation in a given dataset based on a dissimilarity (e.g. distance matrix). The observations are compared against their corresponding closest observations in terms of their side information provided. The root mean square of differences and the correlation coefficient are used for continuous variables and for discrete variables the kappa index is used.

Usage

```
sim_eval(d, side_info)
```

Arguments

d	a symmetric matrix of dissimilarity scores between observations of a given dataset. Alternatively, a vector of with the dissimilarity scores of the lower triangle (without the diagonal values) can be used (see details).
side_info	a matrix containing the side information corresponding to the observations in the dataset from which the dissimilarity matrix was computed. It can be either a numeric matrix with one or multiple columns/variables or a matrix with one character variable (discrete variable). If it is numeric, the root mean square of differences is used for assessing the similarity between the observations and their corresponding most similar observations in terms of the side information provided. If it is a character variable, then the kappa index is used. See details.

Details

For the evaluation of dissimilarity matrices this function uses side information (information about one variable which is available for a group of observations, Ramirez-Lopez et al., 2013). It is assumed that there is a (direct or indirect) correlation between this side informative variable and the variables from which the dissimilarity was computed. If `side_info` is numeric, the root mean square of differences (RMSD) is used for assessing the similarity between the observations and their corresponding most similar observations in terms of the side information provided. It is computed as follows:

$$j(i) = NN(xr_i, Xr^{-i})$$

$$RMSD = \sqrt{\frac{1}{m}\sum_{i=1}^{n}(y_i - y_{j(i)})^2}$$

where $NN(xr_i, Xr^{-i})$ represents a function to obtain the index of the nearest neighbor observation found in Xr (excluding the ith observation) for xr_i, y_i is the value of the side variable of the ith observation, $y_{j(i)}$ is the value of the side variable of the nearest neighbor of the ith observation and m is the total number of observations.

If `side_info` 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 indices between the the side information of the observations and the side information of their corresponding nearest observations (i.e. most similar observations). While p_o is the relative agreement p_e is the the agreement expected by chance.

This functions accepts vectors to be passed to argument `d`, in this case, the vector must represent the lower triangle of a dissimilarity matrix (e.g. as returned by the `stats::dist()` function of `stats`).

Value

`sim_eval` returns a list with the following components:

- "`eval`: either the RMSD (and the correlation coefficient) or the kappa index
- `first_nn`: a matrix containing the original side informative variable in the first half of the columns, and the side informative values of the corresponding nearest neighbors in the second half of the columns.

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

```
library(prospectr)
data(NIRsoil)

sg <- savitzkyGolay(NIRsoil$spc, p = 3, w = 11, m = 0)

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

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

# Example 1
# Compute a principal components distance
pca_d <- ortho_diss(Xr, pc_selection = list("manual", 8))$dissimilarity

# Example 1.1
# Evaluate the distance matrix on the baisis of the
# side information (Yr) associated with Xr
se <- sim_eval(pca_d, side_info = as.matrix(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 neighbors
# found by using the distance matrix
se$first_nn
```

```
# Example 1.2
# Evaluate the distance matrix on the basis of two side
# information (Yr and Yr2)
# variables associated with Xr
Yr_2 <- NIRsoil$CEC[as.logical(NIRsoil$train)]
se_2 <- sim_eval(d = pca_d, side_info = cbind(Yr, Yr_2))

# The final evaluation results
se_2$eval

# The final values of the side information variables and the values
# of the side information variables corresponding to the first
# nearest neighbors found by using the distance matrix
se_2$first_nn

# 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_2 <- ortho_projection(Xr, pc_selection = list("manual", 30))

results <- matrix(NA, pca_2$n_components, 3)
colnames(results) <- c("pcs", "rmsd", "r")
results[, 1] <- 1:pca_2$n_components
for (i in 1:pca_2$n_components) {
  ith_d <- f_diss(pca_2$scores[, 1:i, drop = FALSE], scale = TRUE)
  ith_eval <- sim_eval(ith_d, side_info = as.matrix(Yr))
  results[i, 2:3] <- as.vector(ith_eval$eval)
}
plot(results)

# Example 3
# Example 3.1
# Evaluate a dissimilarity matrix computed using the correlation
# method
cd <- cor_diss(Xr)
eval_corr_diss <- sim_eval(cd, side_info = as.matrix(Yr))
eval_corr_diss$eval
```

Index