Package ‘prospectr’

February 16, 2024

Type Package

Title Miscellaneous Functions for Processing
and Sample Selection of Spectroscopic Data

Version 0.2.7

Date 2024-02-16

Author Antoine Stevens [aut, cre] (<https://orcid.org/0000-0002-1588-7519>),
Leonardo Ramirez-Lopez [aut, cre] (<https://orcid.org/0000-0002-5369-5120>)

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

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

Description Functions to preprocess spectroscopic data
and conduct (representative) sample selection/calibration sampling.

License MIT + file LICENSE

URL https://github.com/l-ramirez-lopez/prospectr

VignetteBuilder knitr

Suggests knitr,
rmarkdown,
formatR,
testthat,
bookdown

LinkingTo Rcpp,
RcppArmadillo

Depends R (>= 3.5.0),

Imports foreach,
iterators,
Rcpp (>= 1.0.1),
mathjaxr (>= 1.0),
lifecycle (>= 0.2.0)

RdMacros mathjaxr

RoxygenNote 7.3.1

NeedsCompilation yes

LazyData true

LazyDataCompression xz

Repository CRAN

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/VersionName cakes

R topics documented:

prospectr-package *Overview of the functions in the prospectr package*

Description

Stable

Misc functions for spectral data

This package implements a number of functions useful for pre-processing spectral data well as for selecting representative samples/spectra. The functions included here are particularly useful in Near-Infrared and Infrared Spectroscopy applications.

Details

#' This is the version 0.2.7 – cakes of the package. The main functionality is listed here.

Currently, the following preprocessing functions are available:

- `resample`
- `resample2`
- `movav`
- `standardNormalVariate`
- `msc`
- `detrend`
- `baseline`
- `blockScale`
- `blockNorm`
- `binning`
- `savitzkyGolay`
- `gapDer`
- `continuumRemoval`

For the selection of representative samples/observations for calibrating spectral models the following functions ca be used:

- `naes`
- `honigs`
- `shenkWest`
- `kenStone`
- `duplex`
- `puchwein`

Other useful functions are also available:

- `read_nircal`
- `readASD`
- `spliceCorrection`
- `cochranTest`

Author(s)

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

Authors:

- Antoine Stevens (ORCID)
- Leonardo Ramirez-Lopez (ORCID)

See Also

Useful links:

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

---

`baseline` *baseline*

Description

Maturing

Fits a baseline to each spectrum in a matrix and removes it from the corresponding input spectrum. A vector can also be passed to this function.

Usage

```
baseline(X, wav)
```

Arguments

`X`	a numeric matrix or vector to process (optionally a data frame that can be coerced to a numerical matrix).
`wav`	optional. A numeric vector of band positions.

Details

The baseline function find points lying on the convex hull of a spectrum, connects the points by linear interpolation and subtracts the interpolated line (baseline) from the corresponding spectrum.

Value

a matrix or vector with the baselined spectra. The resulting matrix is output with an attribute called `baselines` which contain the spectra of the fitted baselines.

This function is similar to `continuumRemoval` and it might replace some of its functionality in the future.

Author(s)

Leonardo Ramirez-Lopez with contributions from Mervin Manalili

See Also

`savitzkyGolay`, `movav`, `gapDer`, `binning`, `continuumRemoval`

Examples

```
data(NIRsoil)
wav <- as.numeric(colnames(NIRsoil$spc))
# plot of the 5 first absorbance spectra
matplot(wav,
  t(NIRsoil$spc[1:5, ]),
  type = "l",
  ylim = c(0, .6),
  xlab = "Wavelength /nm",
  ylab = "Absorbance"
)

bs <- baseline(NIRsoil$spc, wav)
matlines(wav, t(bs[1:5, ]))

fitted_baselines <- attr(bs, "baselines")
matlines(wav, t(fitted_baselines[1:5, ]))
title("Original spectra, baselines and baselined spectra")
```

`binning` *Signal binning*

Description

Compute average values of a signal in pre-determined bins (col-wise subsets). The bin size can be determined either directly or by specifying the number of bins. Sometimes called boxcar transformation in signal processing

Usage

```
binning(X, bins, bin.size)
```

Arguments

`X` a numeric matrix or vector to process (optionally a data frame that can be coerced to a numerical matrix).

`bins` the number of bins.

`bin.size` the desired size of the bins.

Value

a matrix or vector with average values per bin.

Author(s)

Antoine Stevens & Leonardo Ramirez-Lopez

See Also

savitzkyGolay, movav, gapDer, continuumRemoval

Examples

```
data(NIRsoil)
wav <- as.numeric(colnames(NIRsoil$spc))

# 5 first spectra
matplot(wav, t(NIRsoil$spc[1:5, ]),
  type = "l",
  xlab = "Wavelength /nm",
  ylab = "Absorbance"
)

NIRsoil$spc_binned <- binning(NIRsoil$spc, bin.size = 20)

# bin means
matpoints(as.numeric(colnames(NIRsoil$spc_binned)),
  t(NIRsoil$spc_binned[1:5, ]),
  pch = 1:5
)

NIRsoil$spc_binned <- binning(NIRsoil$spc, bins = 20)
dim(NIRsoil$spc_binned) # 20 bins

# 5 first spectra
matplot(wav,
  t(NIRsoil$spc[1:5, ]),
  type = "l",
  xlab = "Wavelength /nm",
  ylab = "Absorbance"
)

# bin means
matpoints(as.numeric(colnames(NIRsoil$spc_binned)),
  t(NIRsoil$spc_binned[1:5, ]),
  pch = 1:5
)
```

blockNorm *Sum of squares block weighting*

Description

Sum of squares block weighting: allows to scale blocks of variables, but keeping the relative weights of the variables inside a block.

Usage

```
blockNorm(X, targetnorm = 1)
```

Arguments

`X`	a numeric matrix to transform (optionally a data frame that can be coerced to a numerical matrix).
`targetnorm`	desired sum of squares for a block of variables (default = 1)

Details

The function computes a scaling factor, which, multiplied by the input matrix, produces a matrix with a pre–determined sum of squares.

Value

a list with components `Xscaled`, the scaled matrix and `f`, the scaling factor

Note

This is a R port of the '`MBnorm.m`' function of the MB matlab toolbox by Fran van den Berg.

Author(s)

Antoine Stevens

References

Eriksson, L., Johansson, E., Kettaneh, N., Trygg, J., Wikstrom, C., and Wold, S., 2006. Multi- and Megavariate Data Analysis. MKS Umetrics AB.

See Also

`blockScale`, `standardNormalVariate`, `detrend`

Examples

```
X <- matrix(rnorm(100), ncol = 10)
# Block normalize to sum of square equals to 1
res <- blockNorm(X, targetnorm = 1)
sum(res$Xscaled^2) # check
```

---

blockScale *Hard or soft block scaling*

Description

Hard or soft block scaling of a spectral matrix to constant group variance. In multivariate calibration, block scaling is used to down-weight variables, when one block of variables dominates other blocks. With hard block scaling, the variables in a block are scaled so that the sum of their variances equals 1. When soft block scaling is used, the variables are scaled such that the sum of variable variances is equal to the square root of the number of variables in a particular block.

Usage

```
blockScale(X, type = 'hard', sigma2 = 1)
```

Arguments

`X`	a numeric matrix or vector to process (optionally a data frame that can be coerced to a numerical matrix).
`type`	the type of block scaling: 'hard' or 'soft'.
`sigma2`	the desired total variance of a block (ie sum of the variances of all variables, default = 1), applicable when `type = 'hard'`.

Value

a `list` with `Xscaled`, the scaled matrix and `f`, the scaling factor.

Author(s)

Antoine Stevens

References

Eriksson, L., Johansson, E., Kettaneh, N., Trygg, J., Wikstrom, C., and Wold, S., 2006. Multi- and Megavariate Data Analysis. MKS Umetrics AB.

See Also

`blockNorm`, `standardNormalVariate`, `detrend`

Examples

```
X <- matrix(rnorm(100), ncol = 10)
# Hard block scaling
res <- blockScale(X)
# sum of column variances == 1
apply(res$Xscaled, 2, var)
```

---

cochranTest *Cochran* C *Test*

Description

Detects and removes replicate outliers in data series based on the Cochran *C* test for homogeneity in variance.

Usage

```
cochranTest(X, id, fun = 'sum', alpha = 0.05)
```

Arguments

`X`	a a numeric matrix (optionally a data frame that can be coerced to a numerical matrix).
`id`	factor of the replicate identifiers.
`fun`	function to aggregate data: 'sum' (default), 'mean', 'PC1' or 'PC2'.
`alpha`	*p*-value of the Cochran *C* test.

Details

The Cochran *C* test is test whether a single estimate of variance is significantly larger than a a group of variances. It can be computed as:

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

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

For multivariate data, the variance S_i^2 can be computed on aggregated data, using a summary function (`fun` argument) such as `sum`, `mean`, or first principal components ('PC1' and 'PC2').

An observation is considered to have an outlying variance if the Cochran *C* statistic is higher than an upper limit critical value C_{UL} which can be evaluated with ('t Lam, 2010):

$$C_{UL}(\alpha, n, N) = 1 + [\frac{N-1}{F_c(\alpha/N, (n-1), (N-1)(n-1))}]^{-1}$$

where α is the p-value of the test, n is the (average) number of replicates and F_c is the critical value of the Fisher's F ratio.

The replicates with outlying variance are removed and the test can be applied iteratively until no outlying variance is detected under the given p-value. Such iterative procedure is implemented in `cochranTest`, allowing the user to specify whether a set of replicates must be removed or not from the dataset by graphical inspection of the outlying replicates. The user has then the possibility to (i) remove all replicates at once, (ii) remove one or more replicates by giving their indices or (iii) remove nothing.

Value

a list with components:

- '`X`': input matrix from which outlying observations (rows) have been removed
- '`outliers`': numeric vector giving the row indices of the input data that have been flagged as outliers

Note

The test assumes a balanced design (i.e. data series have the same number of replicates).

Author(s)

Antoine Stevens

References

Centner, V., Massart, D.L., and De Noord, O.E., 1996. Detection of inhomogeneities in sets of NIR spectra. Analytica Chimica Acta 330, 1-17.

R.U.E. 't Lam (2010). Scrutiny of variance results for outliers: Cochran's test optimized. Analytica Chimica Acta 659, 68-84.

https://en.wikipedia.org/wiki/Cochran's_C_test

`continuumRemoval` *Continuum Removal*

Description

Maturing

Compute the continuum removed values of a data matrix or vector

Usage

```
continuumRemoval(X, wav, type = c("R", "A"),
                 interpol = c("linear", "spline"),
                 method = c("division", "substraction"))
```

Arguments

`X` a numeric matrix or vector to process (optionally a data frame that can be coerced to a numerical matrix).

`wav` optional. A numeric vector of band positions.

`type` the type of data: 'R' for reflectance (default), 'A' for absorbance.

`interpol` the interpolation method between points on the convex hull: 'linear' (default) or 'spline'.

`method` normalization method: 'division' (default) or 'subtraction' (see details section).

Details

The continuum removal technique was introduced by Clark and Roush (1984) as a method to highlight energy absorption features of minerals. It can be viewed as a way to perform albedo normalization. The algorithm find points lying on the convex hull (local maxima or envelope) of a spectrum, connects the points by linear or spline interpolation and normalizes the spectrum by dividing (or subtracting) the input data by the interpolated line.

Value

a matrix or vector with the filtered spectra.

Author(s)

Antoine Stevens & Leonardo Ramirez-Lopez

References

Clark, R.N., and Roush, T.L., 1984. Reflectance Spectroscopy: Quantitative Analysis Techniques for Remote Sensing Applications. J. Geophys. Res. 89, 6329-6340.

See Also

`savitzkyGolay`, `movav`, `gapDer`, `binning`

Examples

```
data(NIRsoil)
wav <- as.numeric(colnames(NIRsoil$spc))
# plot of the 10 first abs spectra
matplot(wav,
  t(NIRsoil$spc[1:10, ]),
  type = "l",
  ylim = c(0, .6),
  xlab = "Wavelength /nm",
  ylab = "Abs"
)
#  # type = 'A' is used for absorbance spectra
cr <- continuumRemoval(NIRsoil$spc, wav, type = "A")
matlines(wav, t(cr[1:10, ]))
```

`detrend` *Detrending spectral data*

Description

Normalizes each row of an input matrix by applying a SNV transformation followed by fitting a second order linear model and returning the fitted residuals.

Usage

```
detrend(X, wav, p = 2)
```

Arguments

`X`	a numeric matrix or vector to process (optionally a data frame that can be coerced to a numerical matrix)
`wav`	the wavelengths/ band centers.
`p`	an integer larger than 1 indicating the polynomial order (default is 2, as in the original paper of Barnes et al., 1989).

Details

The detrend is a row-wise transformation that allows to correct for wavelength-dependent scattering effects (variations in curvilinearity). A p order polynomial is fit for each spectrum (x_i) using the vector of bands (λ, e.g. wavelengths) as explanatory variable as follows:

$$x_i = a\lambda^p + ... + b\lambda + c + e_i$$

were a, b, c are estimated by least squares, and e_i are the spectral residuals of the least square fit. The residuals of the ith correspond to the ith detrended spectrum.

Value

a matrix or vector with the detrended data.

Author(s)

Antoine Stevens and Leonardo Ramirez-Lopez

References

Barnes RJ, Dhanoa MS, Lister SJ. 1989. Standard normal variate transformation and de-trending of near-infrared diffuse reflectance spectra. Applied spectroscopy, 43(5): 772-777.

See Also

`standardNormalVariate`, `blockScale`, `blockNorm`

Examples

```
data(NIRsoil)
wav <- as.numeric(colnames(NIRsoil$spc))
# conversion to reflectance
opar <- par(no.readonly = TRUE)
par(mfrow = c(2, 1), mar = c(4, 4, 2, 2))
# plot of the 10 first spectra
matplot(wav, t(NIRsoil$spc[1:10, ]),
  type = "l",
  xlab = "",
  ylab = "Absorbance"
)
mtext("Raw spectra")
det <- detrend(NIRsoil$spc, wav)
matplot(wav, t(det[1:10, ]),
  type = "l",
  xlab = "Wavelength /nm",
  ylab = "Absorbance"
)
mtext("Detrend spectra")
par(opar)
```

duplex *DUPLEX algorithm for calibration sampling*

Description

Select calibration samples from a large multivariate data using the DUPLEX algorithm

Usage

```
duplex(X,
       k,
       metric = c("mahal", "euclid"),
       pc,
       group,
       .center = TRUE,
       .scale = FALSE)
```

Arguments

X	a numeric matrix.
k	the number of calibration/validation samples.
metric	the distance metric to be used: 'euclid' (Euclidean distance) or 'mahal' (Mahalanobis distance, default).

`pc` optional. The number of Principal Components to be used to select the samples. If not specified, distance are computed in the Euclidean space. Alternatively, distances are computed in the principal component space and `pc` is the number of principal components retained. If `pc < 1`, the number of principal components kept corresponds to the number of components explaining at least (`pc * 100`) percent of the total variance.

`group` An optional `factor` (or vector that can be coerced to a factor by `as.factor`) of length equal to nrow(X), giving the identifier of related observations (e.g. samples of the same batch of measurements, samples of the same origin, or of the same soil profile). When one observation is selected by the procedure all observations of the same group are removed together and assigned to the calibration/validation sets. This allows to select calibration and validation samples that are independent from each other.

`.center` logical value indicating whether the input matrix must be centered before projecting `X` onto the Principal Component space. Analysis. Default set to `TRUE`.

`.scale` logical value indicating whether the input matrix must be scaled before `X` onto the Principal Component space. Analysis. Default set to `FALSE`.

Details

The DUPLEX algorithm is similar to the Kennard-Stone algorithm (see `kenStone`) but allows to select both calibration and validation points that are independent. Similarly to the Kennard-Stone algorithm, it starts by selecting the pair of points that are the farthest apart. They are assigned to the calibration sets and removed from the list of points. Then, the next pair of points which are farthest apart are assigned to the validation sets and removed from the list. In a third step, the procedure assigns each remaining point alternatively to the calibration and validation sets based on the distance to the points already selected. Similarly to the Kennard-Stone algorithm, the default distance metric used by the procedure is the Euclidean distance, but the Mahalanobis distance can be used as well using the `pc` argument (see `kenStone`).

Value

a `list` with components:

- '`model`': numeric vector giving the row indices of the input data selected for calibration
- '`test`': numeric vector giving the row indices of the input data selected for validation
- '`pc`': if the `pc` argument is specified, a numeric matrix of the scaled pc scores

Author(s)

Antoine Stevens & Leonardo Ramirez-Lopez

References

Kennard, R.W., and Stone, L.A., 1969. Computer aided design of experiments. Technometrics 11, 137-148.

Snee, R.D., 1977. Validation of regression models: methods and examples. Technometrics 19, 415-428.

See Also

kenStone, honigs, shenkWest, naes

Examples

```
data(NIRsoil)
sel <- duplex(NIRsoil$spc, k = 30, metric = "mahal", pc = .99)
plot(sel$pc[, 1:2], xlab = "PC1", ylab = "PC2")
points(sel$pc[sel$model, 1:2], pch = 19, col = 2) # points selected for calibration
points(sel$pc[sel$test, 1:2], pch = 18, col = 3) # points selected for validation
# Test on artificial data
X <- expand.grid(1:20, 1:20) + rnorm(1e5, 0, .1)
plot(X[, 1], X[, 2], xlab = "VAR1", ylab = "VAR2")
sel <- duplex(X, k = 25, metric = "mahal")
points(X[sel$model, ], pch = 19, col = 2) # points selected for calibration
points(X[sel$test, ], pch = 15, col = 3) # points selected for validation
```

gapDer *Gap-Segment derivative*

Description

Gap-Segment derivatives of a data matrix or vector

Usage

```
gapDer(X, m = 1, w = 1, s = 1, delta.wav)
```

Arguments

`X`	a numeric matrix or vector' to transform (optionally a data frame that can be coerced to a numerical matrix).
`m`	an integer indicating the order of the derivative, larger than 1 (default is 1). Note that this function allows for high order derivatives (e.g. m = 6).
`w`	an integer indicating the gap size (must be odd and >=1), i.e. the spacing between points over which the derivative is computed.
`s`	an integer indicating the segment size (must be odd and >=1), i.e. the range over which the points are averaged (default = 1, i.e. no smoothing corresponding to Norris-Gap Derivative).
`delta.wav`	the sampling interval (or band spacing).

Details

In this type of derivatives, the gap size denotes the length of the x interval that separates the two segments that are averaged. A detailed explanation of gap segment derivatives can be found in Hopkins (2001).

The sampling interval specified with the `delta.wav` argument is used for scaling and get numerically correct derivatives.

The convolution function is written in C++/Rcpp for faster computations.

Value

a matrix or vector with the filtered signal(s)

Author(s)

Antoine Stevens and Leonardo Ramirez-Lopez

References

Hopkins, D. W. (2001). What is a Norris derivative?. NIR news, 12(3), 3-5.

See Also

`savitzkyGolay`, `movav`, `binning`, `continuumRemoval`

Examples

```
data(NIRsoil)
opar <- par(no.readonly = TRUE)
par(mfrow = c(2, 2), mar = c(4, 4, 2, 2))
# plot of the 10 first spectra
matplot(as.numeric(colnames(NIRsoil$spc)),
  t(NIRsoil$spc[1:10, ]),
  type = "l",
  xlab = "",
  ylab = "Absorbance"
)
mtext("Raw spectra")

der <- gapDer(NIRsoil$spc, m = 1, w = 1, s = 1, delta.wav = 2)
matplot(as.numeric(colnames(der)),
  t(der[1:10, ]),
  type = "l",
  xlab = "Wavelength /nm",
  ylab = "gap derivative"
)

mtext("1st derivative spectra")
der <- gapDer(NIRsoil$spc, m = 1, w = 11, s = 1, delta.wav = 2)
matplot(as.numeric(colnames(der)), t(der[1:10, ]),
  type = "l",
  xlab = "Wavelength /nm",
  ylab = "gap derivative"
)

mtext("1st derivative spectra with a window size = 11 nm")
der <- gapDer(NIRsoil$spc, m = 1, w = 11, s = 5, delta.wav = 2)
matplot(as.numeric(colnames(der)), t(der[1:10, ]),
  type = "l",
  xlab = "Wavelength /nm",
  ylab = "gap derivative"
)
mtext("1st derivative spectra with: window size: 11 nm, smoothing: 5 nm")
par(opar)
```

honigs *Honigs algorithm for calibration sampling*

Description

Select calibration samples from a data matrix using the Honings et al. (1985) method

Usage

```
honigs(X, k, type)
```

Arguments

`X` a numeric matrix with absorbance or continuum-removed reflectance values (optionally a data frame that can be coerced to a numerical matrix).

`k` the number of samples to select for calibration.

`type` type of data: 'A' for absorbance (default), 'R' for reflectance, 'CR' for continuum-removed reflectance

Details

The Honigs algorithm is a simple method to select calibration samples based on their absorption features. Absorbance, reflectance and continuum-removed reflectance values (see `continuumRemoval`) can be used (`type` argument). The algorithm can be described as follows: let A be a matrix of $(i \times j)$ absorbance values:

1. the observation (row) with the maximum absolute absorbance ($max(|A|)$) is selected and assigned to the calibration set.
2. a vector of weights W is computed as A_j/max_A where A_j is the column of A having the maximum absolute absorbance and max_A is the absorbance value corresponding to the maximum absolute absorbance of A
3. each row A_i is multiplied by the corresponding weight W_i and the resulting vector is subtracted from the original row A_i.

4. the row of the selected observation and the column with the maximum absolute absorbance is removed from the matrix
5. go back to step 1 and repeat the procedure until the desired number of selected samples is reached

The observation with the maximum absorbance is considered to have an unusual composition. The algorithm selects therefore this observation and remove from other samples the selected absorption feature by subtraction. Samples with low concentration related to this absorption will then have large negative absorption after the subtraction step and hence will be likely to be selected rapidly by the selection procedure as well.

Value

a `list` with components:

- '`model`': numeric vector giving the row indices of the input data selected for calibration
- '`test`': numeric vector giving the row indices of the remaining observations
- '`bands`': indices of the columns used during the selection procedure

Note

The selection procedure is sensitive to noisy features in the signal. The number of samples selected `k` selected by the algorithm cannot be greater than the number of wavelengths.

Author(s)

Antoine Stevens

References

Honigs D.E., Hieftje, G.M., Mark, H.L. and Hirschfeld, T.B. 1985. Unique-sample selection via Near-Infrared spectral substraction. Analytical Chemistry, 57, 2299-2303

See Also

`kenStone`, `naes`, `duplex`, `shenkWest`

Examples

```
data(NIRsoil)
sel <- honigs(NIRsoil$spc, k = 10, type = "A")
wav <- as.numeric(colnames(NIRsoil$spc))
# spectral library
matplot(wav,
  t(NIRsoil$spc),
  type = "l",
  xlab = "wavelength /nm",
  ylab = "Abs",
  col = "grey50"
)
```

```
# plot calibration spectra
matlines(wav,
  t(NIRsoil$spc[sel$model, ]),
  type = "l",
  xlab = "wavelength /nm",
  ylab = "Abs",
  lwd = 2,
  lty = 1
)
# add bands used during the selection process
abline(v = wav[sel$bands])
```

kenStone *Kennard-Stone algorithm for calibration sampling*

Description

Select calibration samples from a large multivariate data using the Kennard-Stone algorithm

Usage

```
kenStone(X, k, metric = "mahal", pc, group,
         .center = TRUE, .scale = FALSE, init = NULL)
```

Arguments

`X`	a numeric matrix.
`k`	number of calibration samples to be selected.
`metric`	distance metric to be used: 'euclid' (Euclidean distance) or 'mahal' (Mahalanobis distance, default).
`pc`	optional. If not specified, distance are computed in the Euclidean space. Alternatively, distance are computed in the principal component score space and `pc` is the number of principal components retained. If `pc < 1`, the number of principal components kept corresponds to the number of components explaining at least (`pc * 100`) percent of the total variance.
`group`	An optional `factor` (or vector that can be coerced to a factor by `as.factor`) of length equal to `nrow(X)`, giving the identifier of related observations (e.g. samples of the same batch of measurements, samples of the same origin, or of the same soil profile). Note that by using this option in some cases, the number of samples retrieved is not exactly the one specified in `k` as it will depend on the groups. See details.
`.center`	logical value indicating whether the input matrix should be centered before Principal Component Analysis. Default set to `TRUE`.
`.scale`	logical value indicating whether the input matrix should be scaled before Principal Component Analysis. Default set to `FALSE`.

`init` (optional) a vector of integers indicating the indices of the observations/rows that are to be used as observations that must be included at the first iteration of the search process. Default is `NULL`, i.e. no fixed initialization. The function will take by default the two most distant observations. If the `group` argument is used, then all the observations in the groups covered by the `init` observations will be also included in the `init` subset.

Details

The Kennard–Stone algorithm allows to select samples with a uniform distribution over the predictor space (Kennard and Stone, 1969). It starts by selecting the pair of points that are the farthest apart. They are assigned to the calibration set and removed from the list of points. Then, the procedure assigns remaining points to the calibration set by computing the distance between each unassigned points i_0 and selected points i and finding the point for which:

$$d_{selected} = \max_{i_0}(\min_{i}(d_{i,i_0}))$$

This essentially selects point i_0 which is the farthest apart from its closest neighbors i in the calibration set. The algorithm uses the Euclidean distance to select the points. However, the Mahalanobis distance can also be used. This can be achieved by performing a PCA on the input data and computing the Euclidean distance on the truncated score matrix according to the following definition of the Mahalanobis H distance:

$$H_{ij}^2 = \sum_{a=1}^{A}(\hat{t}_{ia} - \hat{t}_{ja})^2/\hat{\lambda}_a$$

where $\hat{t}_{ia}$ is the a^{th} principal component score of point i, $\hat{t}_{ja}$ is the corresponding value for point j, $\hat{\lambda}_a$ is the eigenvalue of principal component a and A is the number of principal components included in the computation.

When the `group` argument is used, the sampling is conducted in such a way that at each iteration, when a single sample is selected, this sample along with all the samples that belong to its group, are assigned to the final calibration set. In this respect, at each iteration, the algorithm will select one sample (in case that sample is the only one in that group) or more to the calibration set. This also implies that the argument `k` passed to the function will not necessary reflect the exact number of samples selected. For example, if `k = 2` and if the first sample identified belongs to with group of 5 samples and the second one belongs to a group with 10 samples, then, the total amount of samples retrieved by the function will be 15.

Value

a list with the following components:

- `model`: numeric vector giving the row indices of the input data selected for calibration
- `test`: numeric vector giving the row indices of the remaining observations
- `pc`: if the `pc` argument is specified, a numeric matrix of the scaled pc scores

Author(s)

Antoine Stevens & Leonardo Ramirez-Lopez with contributions from Thorsten Behrens and Philipp Baumann

References

Kennard, R.W., and Stone, L.A., 1969. Computer aided design of experiments. Technometrics 11, 137-148.

See Also

duplex, shenkWest, naes, honigs

Examples

```
data(NIRsoil)
sel <- kenStone(NIRsoil$spc, k = 30, pc = .99)
plot(sel$pc[, 1:2], xlab = "PC1", ylab = "PC2")
# points selected for calibration
points(sel$pc[sel$model, 1:2], pch = 19, col = 2)
# Test on artificial data
X <- expand.grid(1:20, 1:20) + rnorm(1e5, 0, .1)
plot(X, xlab = "VAR1", ylab = "VAR2")
sel <- kenStone(X, k = 25, metric = "euclid")
points(X[sel$model, ], pch = 19, col = 2)

# Using the group argument
library(prospectr)

# create groups
set.seed(1)
my_groups <- sample(1:275, nrow(NIRsoil$spc), replace = TRUE) |> as.factor()

# check the group size
table(my_groups)

results_group <- kenStone(X = NIRsoil$spc, k = 2, pc = 3, group = my_groups)

# as the first two samples selected belong to groups
# which have in total more than 2 samples (k).
my_groups[results_group$model] |>  factor() |> table()
```

movav *Moving average*

Description

A simple moving average of a matrix or vector using a convolution function written in C++/Rcpp for fast computing

Usage

```
movav(X, w)
```

Arguments

X	a numeric matrix or vector to process (optionally a data frame that can be coerced to a numerical matrix).
w	filter length.

Value

a matrix or vector with the filtered signal(s)

Author(s)

Antoine Stevens

See Also

savitzkyGolay, gapDer, binning, continuumRemoval

Examples

```
data(NIRsoil)
wav <- as.numeric(colnames(NIRsoil$spc))
# adding some noise
NIRsoil$spc_noise <- NIRsoil$spc + rnorm(length(NIRsoil$spc), 0, 0.001)
matplot(wav,
  t(NIRsoil$spc_noise[1:10, ]),
  type = "l",
  lty = 1,
  xlab = "Wavelength /nm",
  ylab = "Absorbance",
  col = "grey"
)

# window size of 11 bands
NIRsoil$spc_mov <- movav(NIRsoil$spc_noise, w = 15)
# smoothed data
matlines(as.numeric(colnames(NIRsoil$spc_mov)),
  t(NIRsoil$spc_mov[1:10, ]),
  type = "l",
  lty = 1
)
```

msc *Multiplicative Scatter Correction (msc)*

Description

Maturing

This function implements the multiplicative scatter correction method which attempts to remove physical light scatter by accounting for additive and multiplicative effects (Geladi et al., 1985).

Usage

```
msc(X, ref_spectrum = colMeans(X))
```

Arguments

`X`	a numeric matrix of spectral data.
`ref_spectrum`	a numeric vector corresponding to an "ideal" reference spectrum (e.g. free of scattering effects). By default the function uses the mean spectrum of the input `X`. See details. Note that this argument was previously named as `reference_spc`, however, it has been renamed to `ref_spectrum` to emphasize that this argument is a vector and not a matrix of spectra.

Details

The Multiplicative Scatter Correction (MSC) is a normalization method that attempts to account for additive and multiplicative effects by aligning each spectrum (x_i) to an ideal reference one (x_r) as follows:

$$x_i = m_i x_r + a_i$$

$$MSC(x_i) = \frac{a_i - x_i}{m_i}$$

where a_i and m_i are the additive and multiplicative terms respectively.

Value

a matrix of normalized spectral data with an attribute which indicates the reference spectrum used.

Author(s)

Leonardo Ramirez-Lopez and Guillaume Hans

References

Geladi, P., MacDougall, D., and Martens, H. 1985. Linearization and Scatter-Correction for Near-Infrared Reflectance Spectra of Meat. Applied Spectroscopy, 39(3):491-500.

See Also

standardNormalVariate, detrend, blockScale, blockNorm

Examples

```
data(NIRsoil)
NIRsoil$msc_spc <- msc(X = NIRsoil$spc)

# 10 first msc spectra
matplot(
  x = as.numeric(colnames(NIRsoil$msc_spc)),
  y = t(NIRsoil$msc_spc[1:10, ]),
  type = "l",
  xlab = "wavelength, nm",
  ylab = "msc"
)

# another example
spectra_a <- NIRsoil$spc[1:40, ]
spectra_b <- NIRsoil$spc[-(1:40), ]

spectra_a_msc <- msc(spectra_a, colMeans(spectra_a))

# correct spectra_a based on the reference spectrum used to correct
# spectra_a

spectra_b_msc <- msc(
  spectra_b,
  ref_spectrum = attr(spectra_a_msc, "Reference spectrum")
)
```

naes *k-means sampling*

Description

Perform a k-means sampling on a matrix for multivariate calibration

Usage

```
naes(X, k, pc, iter.max = 10, method = 0, .center = TRUE, .scale = FALSE)
```

Arguments

`X` a numeric matrix (optionally a data frame that can be coerced to a numerical matrix).

`k` either the number of calibration samples to select or a set of cluster centres to initiate the k-means clustering.

`pc` optional. If not specified, k-means is run directly on the variable (Euclidean) space. Alternatively, a PCA is performed before k-means and `pc` is the number of principal components kept. If `pc < 1`,the number of principal components kept corresponds to the number of components explaining at least (`pc * 100`) percent of the total variance.

`iter.max` maximum number of iterations allowed for the k-means clustering. Default is `iter.max = 10` (see `?kmeans`).

`method` the method used for selecting calibration samples within each cluster: either samples closest to the cluster. centers (`method = 0`, default), samples farthest away from the centre of the data (`method = 1`) or random selection (`method = 2`).

`.center` logical value indicating whether the input matrix must be centered before Principal Component Analysis. Default set to `TRUE`.

`.scale` logical value indicating whether the input matrix must be scaled before Principal Component Analysis. Default set to `FALSE`.

Details

K-means sampling is a simple procedure based on cluster analysis to select calibration samples from large multivariate datasets. The method can be described in three points (Naes et al.,2001):

1. Perform a PCA and decide how many principal component to keep,
2. Carry out a k-means clustering on the principal component scores and choose the number of resulting clusters to be equal to the number of desired calibration samples,
3. Select one sample from each cluster.

Value

a list with components:

- '`model`': numeric vector giving the row indices of the input data selected for calibration
- '`test`': numeric vector giving the row indices of the remaining observations
- '`pc`': if the `pc` argument is specified, a numeric matrix of the scaled pc scores
- '`cluster`': integer vector indicating the cluster to which each point was assigned
- '`centers`': a matrix of cluster centres

Author(s)

Antoine Stevens & Leonardo Ramirez-Lopez

References

Naes, T., 1987. The design of calibration in near infra-red reflectance analysis by clustering. Journal of Chemometrics 1, 121-134.

Naes, T., Isaksson, T., Fearn, T., and Davies, T., 2002. A user friendly guide to multivariate calibration and classification. NIR Publications, Chichester, United Kingdom.

See Also

kenStone, honigs, duplex, shenkWest

Examples

```
data(NIRsoil)
sel <- naes(NIRsoil$spc, k = 5, p = .99, method = 0)
# clusters
plot(sel$pc[, 1:2], col = sel$cluster + 2)
# points selected for calibration with method = 0
points(sel$pc[sel$model, 1:2],
  col = 2,
  pch = 19,
  cex = 1
)
# pre-defined centers can also be provided
sel2 <- naes(NIRsoil$spc,
  k = sel$centers,
  p = .99, method = 1
)
# points selected for calibration with method = 1
points(sel$pc[sel2$model, 1:2],
  col = 1,
  pch = 15,
  cex = 1
)
```

NIRsoil *NIRSoil*

Description

Soil spectral library of the ‘Chimiometrie 2006’ challenge. The database contains absorbance spectra of dried and sieved soil samples measured between 1100 nm and 2498 nm at 2 nm interval. The soil samples come from agricultural fields collected from all over the Walloon region in Belgium. Three parameters are associated with the spectral library: Nt (Total Nitrogen in g/Kg of dry soil), CEC (Cation Exchange Capacity in meq/100 g of dry soil) and Ciso (Carbon in g/100 g of dry soil). Carbon content has been measured following the ISO14235 method.

Usage

```
data(NIRsoil)
```

Format

A `data.frame` of 825 observations and 5 variables (where the spectral data is embedded in one variable `NIRSoil$spc`).

Details

The dataset includes 618 training and 207 test samples with 5 variables:

- Nt (Total Nitrogen).
- Ciso (Carbon).
- CEC (Cation Exchange Capacity).
- train (binary vector indicating training (1) and validation (0) samples).
- and spc (a matrix of spectral NIR absorbance values, where the band/wavelength positions are stored as `colnames`).

Nt, Ciso and CEC have respectively 22 \ with missing values.

Source

Pierre Dardenne from Walloon Agricultural Research Centre, Belgium.

References

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

`puchwein` *Puchwein algorithm for calibration sampling*

Description

Select calibration samples from multivariate data using the Puchwein algorithm

Usage

```
puchwein(X,
         pc = 0.95,
         k,
         min.sel,
         details = FALSE,
         .center = TRUE,
         .scale = FALSE)
```

Arguments

`X`	a matrix from which the calibration samples are to be selected (optionally a data frame that can be coerced to a numerical matrix).

`pc` the number of principal components retained in the computation of the distance in the standardized Principal Component space (Mahalanobis distance). If `pc < 1`, the number of principal components kept corresponds to the number of components explaining at least (`pc * 100`) percent of the total variance (default = 0.95 as in the Puchwein paper).

`k` the initial limiting distance parameter, if not specified (default), set to 0.2. According to Puchwein, a good starting value for the limiting distance is $d_{ini} = k(p-2)$ where p is the number of principal components

`min.sel` minimum number of samples to select for calibration (default = 5).

`details` logical value, if `TRUE`, adds a component in the output list with the indices of the objects kept in each loop (default to `FALSE`).

`.center` logical value indicating whether the input matrix must be centered before Principal Component. Analysis. Default set to TRUE.

`.scale` logical value indicating whether the input matrix must be scaled before Principal Component Analysis. Default set to FALSE.

Details

The Puchwein algorithm select samples from a data matrix by iteratively eliminating similar samples using the Mahalanobis distance. It starts by performing a PCA on the input matrix and extracts the score matrix truncated to A, the number of principal components. The score matrix is then normalized to unit variance and the Euclidean distance of each sample to the centre of the data is computed, which is identical to the Mahalanobis distance H. Additionally, the Mahalanobis distances between samples are comptuted. The algorithm then proceeds as follows:

1. Choose a initial limiting distance d_{ini}
2. Select the sample with the highest H distance to the centre
3. Remove all samples within the minimum distance d_{ini} from the sample selected in step 2
4. Go back to step 2 and proceed until there are no samples/observations left in the dataset
5. Go back to step 1 and increase the minimum distance by multiplying the limiting distance by the loop number

It is not possible to obtain a pre-defined number of samples selected by the method. To choose the adequate number of samples, a data frame is returned by `puchwein` function (`leverage`) giving the observed and theoretical cumulative sum of leverages of the points selected in each iteration. The theoretical cumulative sum of leverage is computed such as each point has the same leverage (the sum of leverages divided by the number of observations). The loop having the largest difference between the observed and theoretical sums is considered as producing the optimal selection of points (the subset that best reproduces the variability of the predictor space).

Value

a `list` with components:

- '`model`': indices of the observations (row indices of the input data) selected for calibration
- '`test`': indices of the remaining observations (row indices of the input data)

- ’`pc`’: a numeric matrix of the scaled pc scores
- ’`loop.optimal`’: index of the loop producing the maximum difference between the observed and theoretical sum of leverages of the selected samples
- ’`leverage`’: data frame giving the observed and theoretical cumulative sums of leverage of the points selected in each loop
- ’`details`’: list with the indices of the observations kept in each loop

Note

The Puchwein algorithm is an iterative method and can be slow for large data matrices.

Author(s)

Antoine Stevens

References

Puchwein, G., 1988. Selection of calibration samples for near-infrared spectrometry by factor analysis of spectra. Analytical Chemystry 60, 569-573.

Shetty, N., Rinnan, A., and Gislum, R., 2012. Selection of representative calibration sample sets for near-infrared reflectance spectroscopy to predict nitrogen concentration in grasses. Chemometrics and Intelligent Laboratory Systems 111, 59-65.

See Also

`kenStone`, `duplex`, `shenkWest`, `honigs`, `naes`

Examples

```
data(NIRsoil)
sel <- puchwein(NIRsoil$spc, k = 0.2, pc = .99)
plot(sel$pc[, 1:2])
# points selected for calibration
points(NIRsoil$spc[sel$model, 1:2], col = 2, pch = 2)
# Leverage plot
opar <- par(no.readonly = TRUE)
par(mar = c(4, 5, 2, 2))
plot(sel$leverage$loop, sel$leverage$diff,
  type = "l",
  xlab = "# loops",
  ylab = "Difference between theoretical and \n observed sum of leverages"
)
par(opar)
```

readASD *Read ASD FieldSpec Pro binary and ASCII files*

Description

Maturing

Read single or multiple binary and ASCII files acquired with an ASD FieldSpec Pro (ASDi, Boulder, CO) spectroradiometer

Usage

```
readASD(fnames, in_format, out_format)
```

Arguments

`fnames`	a character vector of the name(s) (with absolute path) of the file(s) to read.
`in_format`	the format of the input file: `'binary'` or `'txt'`.
`out_format`	the format of the output: 'matrix' (default) or 'list' (see below).

Value

if `out_format` = `'matrix'`, reflectance values of the input file(s) in a single matrix.

if `out_format` = `'list'`, a `list` of the input file(s) data consisting of a list with components:

- `Name`: name of the file imported
- `datetime`: date and time of acquisition in `POSIXct` format
- `header`: list with information from the header file
- `radiance`: if applicable, a numeric vector of radiance values
- `reference`: if applicable, a numeric vector of radiance values of the white reference
- `reflectance`: numeric vector of reflectance values
- `wavelength`: numeric vector of the band positions

Note

There is a R port of the '`importasd.m`' function from the 'FSFPostProcessing' Matlab toolbox by Iain Robinson (University of Edinburgh), which is based on some Java code provided by Andreas Hunei (University of Zurich).

It seems that ASD file format has changed quite a lot with file versions. The function will possibly not work as expected for all versions. Please report any bugs to the package maintainer.

Author(s)

Antoine Stevens (R port), Iain Robinson (matlab function) & Leonardo Ramirez-Lopez (R port)

References

Robinson, I., and A. MacArthur. 2011. The Field Spectroscopy Facility Post Processing Toolbox User Guide. Post processing spectral data in MATLAB, University of Edinburgh, Edinburgh, UK.

`read_nircal` *Import BUCHI NIRCal files*

Description

Maturing

This function imports .nir files generated by BUCHI NIRCal software.

Usage

```
read_nircal(file, response = TRUE, spectra = TRUE,
            metadata = TRUE, progress = TRUE, verbose = TRUE)
```

Arguments

`file`	the name of the NIRCal (.nir) file which the data are to be read from. For URLs a temporary file is first downloaded and is then read.
`response`	a logical indicating if the data of the response variables must be returned (default is `TRUE`).
`spectra`	a logical indicating if the spectral data must be returned (default is `TRUE`).
`metadata`	a logical indicating if the metadada must be returned (default is `TRUE`).
`progress`	a logical indicating if a progress bar must be printed (default is `TRUE`).
`verbose`	a logical indicating if the number of spectra and response variables (an also the ID's of the spectra without gain and/or temperature information) must be printed (default is `TRUE`).

Details

The extension of the BUCHI NIRCal files is .nir. These files are used to store spectra generated by BUCHI N-500 and BUCHI NIRMaster FT-NIR sensors. See NIRCal technical data sheet.

Value

a data.frame containing the metadata, response variables (if `response = TRUE`) and spectra (if `spectra = TRUE`, embedded in the `data.frame` as a matrix named `...$spc`).

Author(s)

Leonardo Ramirez-Lopez

`resample` *Resample spectral data*

Description

Resample a data matrix or vector to new coordinates (e.g. band positions) using spline or linear interpolation. This function is a simple wrapper around `approx` and `splinefun` in **base**.

Usage

```
resample(X, wav, new.wav, interpol = ”spline”, ...)
```

Arguments

`X` numeric matrix or vector to resample (optionally a data frame that can be coerced to a numerical matrix).

`wav` a numeric vector giving the original band positions.

`new.wav` a numeric vector giving the new band positions.

`interpol` the interpolation method: ’linear’ or ’spline’ (default).

`...` additional arguments to be passed to the `splinefun` function when `interpol = 'spline'`.

Value

a matrix or vector with resampled values.

Author(s)

Antoine Stevens and Leonardo Ramirez-Lopez

See Also

`resample2`

Examples

```
data(NIRsoil)
wav <- as.numeric(colnames(NIRsoil$spc))
# increase spectral resolution by 2
NIRsoil$spc_resampled <- resample(NIRsoil$spc, wav, seq(1100, 2498, 2))
dim(NIRsoil$spc)
dim(NIRsoil$spc_resampled)
```

`resample2` *Resample a high resolution signal to a low resolution signal using full width half maximum (FWHM) values*

Description

Resample a data matrix or vector to match the response of another instrument using full width half maximum (FWHM) values

Usage

```
resample2(X, wav, new.wav, fwhm)
```

Arguments

`X`	a numeric matrix or vector to resample (optionally a data frame that can be coerced to a numerical matrix).
`wav`	a numeric vector giving the original band positions.
`new.wav`	a numeric vector giving the new band positions.
`fwhm`	a numeric vector giving the full width half maximums of the new band positions. If no value is specified, it is assumed that the fwhm is equal to the sampling interval (i.e. band spacing). If only one value is specified, the fwhm is assumed to be constant over the spectral range.

Details

The function uses gaussian models defined by fwhm values to resample the high resolution data to new band positions and resolution. It assumes that band spacing and fwhm of the input data is constant over the spectral range. The interpolated values are set to 0 if input data fall outside by 3 standard deviations of the gaussian densities defined by fwhm.

Value

a matrix or vector with resampled values

Author(s)

Antoine Stevens

See Also

`resample`

Examples

```
data(NIRsoil)
wav <- as.numeric(colnames(NIRsoil$spc))
# Plot 10 first spectra
matplot(wav, t(NIRsoil$spc[1:10, ]),
  type = "l", xlab = "Wavelength /nm",
  ylab = "Absorbance"
)
# ASTER SWIR bands (nm)
new_wav <- c(1650, 2165, 2205, 2260, 2330, 2395) # positions
fwhm <- c(100, 40, 40, 50, 70, 70) #  fwhm's
# Resample NIRsoil to ASTER band positions
aster <- resample2(NIRsoil$spc, wav, new_wav, fwhm)
matpoints(as.numeric(colnames(aster)), t(aster[1:10, ]), pch = 1:5)
```

---

savitzkyGolay *Savitzky-Golay smoothing and differentiation*

Description

Savitzky-Golay smoothing and derivative of a data matrix or vector.

Usage

```
savitzkyGolay(X, m, p, w, delta.wav)
```

Arguments

X	a numeric matrix or vector to process (optionally a data frame that can be coerced to a numerical matrix).
m	an integer indcating the differentiation order.
p	an integer indicating the polynomial order.
w	an integer indicating the window size (must be odd).
delta.wav	(optional) sampling interval.

Details

The Savitzky-Golay algorithm fits a local polynomial regression on the signal. It requires evenly spaced data points. Mathematically, it operates simply as a weighted sum over a given window:

$$x_j* = \frac{1}{N}\sum_{h=-k}^{k} c_h x_{j+h}$$

where x_j* is the new value, N is a normalizing coefficient, k is the gap size on each side of j and c_h are pre-computed coefficients, that depends on the chosen polynomial order and degree.

The sampling interval specified with the `delta.wav` argument is used for scaling and get numerically correct derivatives.

The convolution function is written in C++/Rcpp for faster computations.

Author(s)

Antoine Stevens and Leonardo Ramirez-Lopez

References

Luo, J., Ying, K., He, P., & Bai, J. (2005). Properties of Savitzky–Golay digital differentiators. Digital Signal Processing, 15(2), 122-136.

Savitzky, A., and Golay, M.J.E., 1964. Smoothing and differentiation of data by simplified least squares procedures. Anal. Chem. 36, 1627-1639.

Schafer, R. W. (2011). What is a Savitzky-Golay filter? (lecture notes). IEEE Signal processing magazine, 28(4), 111-117.

Wentzell, P.D., and Brown, C.D., 2000. Signal processing in analytical chemistry. Encyclopedia of Analytical Chemistry, 9764-9800.

Examples

```
data(NIRsoil)
opar <- par(no.readonly = TRUE)
par(mfrow = c(2, 1), mar = c(4, 4, 2, 2))

# plot of the 10 first spectra
matplot(as.numeric(colnames(NIRsoil$spc)),
  t(NIRsoil$spc[1:10, ]),
  type = "l",
  xlab = "",
  ylab = "Absorbance"
)

mtext("Raw spectra")
NIRsoil$spc_sg <- savitzkyGolay(
  X = NIRsoil$spc,
  m = 1,
  p = 3,
  w = 11,
  delta.wav = 2
)

matplot(as.numeric(colnames(NIRsoil$spc_sg)),
  t(NIRsoil$spc_sg[1:10, ]),
  type = "l",
  xlab = "Wavelength /nm",
  ylab = "1st derivative"
)

mtext("1st derivative spectra")
par(opar)
```

shenkWest *SELECT algorithm for calibration sampling*

Description

Select calibration samples from a large multivariate data using the SELECT algorithm as described in Shenk and Westerhaus (1991).

Usage

```
shenkWest(X,
          d.min = 0.6,
          pc = 0.95,
          rm.outlier = FALSE,
          .center = TRUE,
          .scale = FALSE)
```

Arguments

X	a numeric matrix (optionally a data frame that can be coerced to a numerical matrix).
d.min	a minimum distance (default = 0.6).
pc	the number of principal components retained in the computation distance in the standardized Principal Component space (Mahalanobis distance). If pc < 1, the number of principal components kept corresponds to the number of components explaining at least (pc * 100) percent of the total variance (default = 0.95).
rm.outlier	logical. If TRUE, remove observations with a standardized mahalanobis distance to the center of the data greater than 3 (default = FALSE).
.center	logical. Indicates whether the input matrix should be centered before Principal Component Analysis. Default set to TRUE.
.scale	logical. Indicates whether the input matrix should be scaled before Principal Component Analysis. Default set to FALSE.

Details

The SELECT algorithm is an iterative procedure based on the standardized Mahalanobis distance between observations. First, the observation having the highest number of neighbours within a given minimum distance is selected and its neighbours are discarded. The procedure is repeated until there is no observation left.

If the rm.outlier argument is set to TRUE, outliers will be removed before running the SELECT algorithm, using the CENTER algorithm of Shenk and Westerhaus (1991), i.e. samples with a standardized Mahalanobis distance >3 are removed.

Value

a `list` with components:

- '`model`': numeric vector giving the row indices of the input data selected for calibration
- '`test`': numeric vector giving the row indices of the remaining observations
- '`pc`': a numeric matrix of the scaled pc scores

Author(s)

Antoine Stevens

References

Shenk, J.S., and Westerhaus, M.O., 1991. Population Definition, Sample Selection, and Calibration Procedures for Near Infrared Reflectance Spectroscopy. Crop Science 31, 469-474.

See Also

`kenStone`, `duplex`, `puchwein`

Examples

```
data(NIRsoil)
# reduce data size
NIRsoil$spc <- binning(X = NIRsoil$spc, bin.size = 5)
sel <- shenkWest(NIRsoil$spc, pc = .99, d.min = .3, rm.outlier = FALSE)
plot(sel$pc[, 1:2], xlab = "PC1", ylab = "PC2")
# points selected for calibration
points(sel$pc[sel$model, 1:2], pch = 19, col = 2)
# without outliers
sel <- shenkWest(NIRsoil$spc, pc = .99, d.min = .3, rm.outlier = TRUE)
plot(sel$pc[, 1:2], xlab = "PC1", ylab = "PC2")
# points selected for calibration
points(sel$pc[sel$model, 1:2], pch = 15, col = 3)
```

`spliceCorrection` *Splice correction of a spectral matrix acquired with an ASD spectrometer*

Description

Corrects steps in an input spectral matrix by linear interpolation of the values of the edges of the middle sensor

Usage

```
spliceCorrection(X, wav, splice = c(1000, 1830), interpol.bands = 10)
```

Arguments

`X` a numeric matrix or vector to transform (optionally a data frame that can be coerced to a numerical matrix).

`wav` a numeric vector with band positions.

`splice` a numeric vector of length 1 or 2 with the positions of the splice(s). Default: `c(1000,1830)` (as for the ASD FieldSpec Pro spectrometer of Malvern Panalytical). See details.

`interpol.bands` the number of interpolation bands.

Details

This function uses by default the positions for the ASD FieldSpec Pro spectroradiometer (Malvern Panalytical) which usually exhibit steps at the splice of the three built-in detectors, positioned at 1000 nm (end of VNIR detector) and 1830 nm (end of SWIR1 detector). The data corresponding to the spectral region after the first step is used as reference for correcting the first region and the laste region (if 2 steps are supplied). Other typical examples of splice artifacts caused by concatenating data captured by different detectors inside the spectrometer:

- XDS (FOSS): 1100 nm
- ProxiMate (BUCHI Labortechnik): 900 nm

Value

a matrix with the splice corrected data.

Author(s)

Antoine Stevens and Leonardo Ramirez-Lopez

`standardNormalVariate` *Standard normal variate transformation*

Description

This function normalizes each row of an input matrix by subtracting each row by its mean and dividing it by its standard deviation

Usage

```
standardNormalVariate(X)
```

Arguments

`X` a numeric matrix of spectral data (optionally a data frame that can be coerced to a numerical matrix).

Details

SNV is simple way for normalizing spectral data that intends to correct for light scatter. It operates row-wise:

$$SNV_i = \frac{x_i - \bar{x}_i}{s_i}$$

where x_i is the signal of the ith observation, $\bar{x}_i$ is its mean and s_i its standard deviation.

Value

a matrix of normalized spectral data.

Author(s)

Antoine Stevens

References

Barnes RJ, Dhanoa MS, Lister SJ. 1989. Standard normal variate transformation and de-trending of near-infrared diffuse reflectance spectra. Applied spectroscopy, 43(5): 772-777.

See Also

msc, detrend, blockScale, blockNorm

Examples

```
data(NIRsoil)
NIRsoil$spc_snv <- standardNormalVariate(X = NIRsoil$spc)
# 10 first snv spectra
matplot(
  x = as.numeric(colnames(NIRsoil$spc_snv)),
  y = t(NIRsoil$spc_snv[1:10, ]),
  type = "l",
  xlab = "wavelength, nm",
  ylab = "snv"
)
## Not run:
apply(NIRsoil$spc_snv, 1, sd) # check

## End(Not run)
```

Index