Package ‘prospectr’

February 20, 2015

Type Package

Title Miscellaneous functions for processing and sample selection of vis-NIR diffuse reflectance data

Version 0.1.3

Date 2013-12-11

Author Antoine Stevens and Leonardo Ramirez-Lopez

Maintainer Antoine Stevens <antoine.stevens@uclouvain.be>

Description The package provides functions for pretreatment and sample selection of visible and near infrared diffuse reflectance spectra

License GPL (>= 3)

URL https://github.com/antoinestevens/prospectr

VignetteBuilder knitr

Suggests hyperSpec, knitr

LinkingTo Rcpp, RcppArmadillo

Depends R (>= 3.0.0), RcppArmadillo (>= 0.4.000)

Imports foreach, iterators, Rcpp (>= 0.11.0)

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-02-14 14:00:28

R topics documented:

`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`	numeric `data.frame`, `matrix` or `vector` to process
`bins`	number of bins
`bin.size`	desired size of the bins

Value

a `matrix` or `vector` with average values per bin

Author(s)

Antoine Stevens & Leonardo Ramirez-Lopez

See Also

`sgolayfilt`, `savitzkyGolay`, `movav`, `gapDer`, `continuumRemoval`

Examples

```
data(NIRsoil)
spc <- 1/10^NIRsoil$spc # conversion to reflectance
wav <- as.numeric(colnames(spc))
matplot(wav,t(spc[1:5,]),type='l',xlab='Wavelength /nm',ylab='Reflectance') # 5 first spectra
binned <- binning(spc,bin.size=20)
matpoints(as.numeric(colnames(binned)),t(binned[1:5,]),pch=1:5) # bin means
binned <- binning(spc,bins=20)
dim(binned) # 20 bins
matplot(wav,t(spc[1:5,]),type='l',xlab='Wavelength /nm',ylab='Reflectance') # 5 first spectra
matpoints(as.numeric(colnames(binned)),t(binned[1:5,]),pch=1:5) # bin means
```

---

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`	`data.frame` or `matrix` to transform
`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 (`http://www.models.life.ku.dk/~courses/MBtoolbox/mbtmain.htm`)

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 = 1
res <- blockNorm(X,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. Wen 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`	`data.frame` or `matrix` to transform
`type`	type of block scaling: 'hard' or 'soft'
`sigma2`	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)
apply(res$Xscaled,2,var) # sum of column variances == 1
```

---

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`	input `data.frame` or `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):

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

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 should 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.

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

`continuumRemoval` *Continuum Removal*

Description

Compute the continuum removed values of a data `matrix`, `data.frame`, or `vector` as implemented in ENVI

Usage

```
continuumRemoval(X,wav,type,interpol,method)
```

Arguments

`X`	numeric `data.frame`, `matrix` or `vector` to process
`wav`	optional. numeric `vector` of band positions
`type`	type of data: 'R' for reflectance (default), 'A' for absorbance
`interpol`	interpolation method between points on the convex hull: 'linear' (default) or 'spline'
`method`	normalization method: 'division' (default) or 'substraction' (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 substracting) the input data by the interpolated line.

Value

a `matrix` or `vector` with the filtered signal(s)

Author(s)

Antoine Stevens

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

`sgolayfilt`, `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` *Detrend transformation*

Description

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

Usage

```
detrend(X,wav)
```

Arguments

`X`	numeric `data.frame`, `matrix` or `vector` to process
`wav`	wavelengths/ band centers

Details

The detrend is a row-wise transformation that allows to correct for wavelength-dependent scattering effects (variations in curvilinearity). A second-degree polynomial is fit through each spectrum:

$$x_i = a\lambda^2 + b\lambda + c + e_i$$

were x_i is the spectrum, λ is the wavelength vector, a, b, c are estimated by least square, and e_i are the residuals of the least square fit. Then, a detrend spectrum corresponds to:

$$x*_i = x_i - (a\lambda^2 + b\lambda + c = e_i)$$

Value

a `matrix` or `vector` with detrend values

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

standardNormalVariate, blockScale, blockNorm

Examples

```
data(NIRsoil)
wav <- as.numeric(colnames(NIRsoil$spc))
spc <- 1/10^NIRsoil$spc # conversion to reflectance
opar <- par(no.readonly = TRUE)
par(mfrow=c(2,1),mar=c(4,4,2,2))
matplot(wav,t(spc[1:10,]),type='l',xlab='',ylab='Reflectance') # plot of the 10 first spectra
mtext('Raw spectra')
det <- detrend(spc,wav)
matplot(wav,t(det[1:10,]),type='l',xlab='Wavelength /nm',ylab='Reflectance')
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,pc,group,.center = TRUE,.scale = FALSE)
```

Arguments

X	a matrix
k	number of calibration/validation samples
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, , 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 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.

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`, `data.frame` or `vector`

Usage

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

Arguments

`X`	numeric `matrix` , `data.frame` or `vector` to transform
`m`	order of the derivative, between 1 and 4 (default = 1)
`w`	filter length (should be odd and >=1), i.e. the spacing between points over which the derivative is computed
`s`	segment size, i.e. the range over which the points are averaged (default = 1, i.e. no smoothing corresponding to 'Norris' Gap Derivative)
`delta.wav`	sampling interval (or band spacing)

Details

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

References

Hopkins (2002). NIR News 14(5), 10.

See Also

sgolayfilt, savitzkyGolay, movav, binning, continuumRemoval

Examples

```
data(NIRsoil)
spc <- 1/10^NIRsoil$spc # conversion to reflectance
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(spc)),t(spc[1:10,]),
        type='l',xlab='',ylab='Reflectance')
mtext('Raw spectra')
der <- gapDer(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(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(spc,m=1,w=11,s = 10,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, smoothing of 10 nm')
par(opar)
```

---

honigs *Honigs algorithm for calibration sampling*

Description

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

Usage

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

Arguments

X	numeric `data.frame` or `matrix` with absorbance or continuum-removed reflectance values
k	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 substracted 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 substraction. Samples with low concentration related to this absorption will then have large negative absorption after the substraction 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,pc,group,.center = TRUE,.scale = FALSE)
```

Arguments

`X`	a numeric `matrix`
`k`	number of desired calibration samples
`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, , 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 set. This allows to select calibration points that are independent from the remaining points.
`.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.

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 i_0 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 analysis 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.

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

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.

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(sel$pc[sel$model,1:2],pch=19,col=2)  # points selected for calibration
# 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)
```

---

movav *Moving average*

Description

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

Usage

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

Arguments

X	numeric `data.frame`, `matrix` or `vector` to process
w	filter length

Value

a `matrix` or `vector` with the filtered signal(s)

Author(s)

Antoine Stevens

See Also

`sgolayfilt`, `savitzkyGolay`, `gapDer`, `binning`, `continuumRemoval`

Examples

```
data(NIRsoil)
wav <- as.numeric(colnames(NIRsoil$spc))
spc <- 1/10^NIRsoil$spc # conversion to reflectance
spc <- spc + rnorm(length(spc),0,0.001) # adding some noise
matplot(wav,t(spc[1:10,]),type='l',xlab='Wavelength /nm',ylab='Reflectance')
mov <- movav(spc,w=11) # window size of 11 bands
matlines(as.numeric(colnames(mov)),t(mov[1:10,])) # smoothed data
```

naes *k-means sampling*

Description

Perform a k-means sampling on a `matrix` or `data.frame` for multivariate calibration

Usage

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

Arguments

`X`	numeric `matrix` or `data.frame`
`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 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.

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

Details

The dataset includes 618 training and 207 test samples with 5 variables: Nt (Total Nitrogen), Ciso (Carbon), CEC (Cation Exchange Capacity), train (`vector` of 0,1 indicating training (1) and validation (0) samples) and spc (a `matrix` with absorbance NIR data and band positions as `colnames`). Nt, Ciso and CEC have respectively 22 %, 11 % and 46 % of the observations 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.

Minasny, B., and McBratney, A.B., 2008. Regression rules as a tool for predicting soil properties from infrared reflectance spectroscopy. Chemometrics and Intelligent Laboratory Systems 94, 72-79.

`prospectr` *prospectr package*

Description

This package implements a number of R functions for preprocessing, and sample selection calibration sampling of visible and near infrared diffuse (VIS-NIR) reflectance data

Currently, the following preprocessing functions are available:

- `continuumRemoval`
- `savitzkyGolay`
- `detrend`
- `gapDer`
- `movav`
- `standardNormalVariate`
- `binning`
- `resample`

- resample2
- blockScale
- blockNorm

The selection of samples/observations for calibration of VIS-NIR data can be achieved with one of the following functions:

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

Other useful functions are also available:

- readASD
- spliceCorrection
- cochranTest

Author(s)

Antoine Stevens & Leonardo Ramirez-Lopez

---

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	input `data.frame` or `matrix` from which to select calibration samples
pc	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`	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 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.

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 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

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	character vector of the name(s) (with absolute path) of the file(s) to read
in_format	format of the input file: 'binary' or 'txt'
out_format	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

The 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) and Iain Robinson (matlab function)

References

http://fsf.nerc.ac.uk/user_group/user_group.shtml

http://www.mathworks.com/matlabcentral/fileexchange/31547

Indico Version 8 file format (http://support.asdi.com/Document/Documents.aspx)

resample *Resample spectral data*

Description

Resample a data `matrix`, `data.frame` 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)
```

Arguments

`X`	numeric `data.frame`, `matrix` or `vector` to resample
`wav`	a numeric vector giving the original band positions
`new.wav`	a numeric vector giving the new band positions
`interpol`	interpolation method: 'linear' or 'spline'

Value

a `matrix` or `vector` with resampled values

Author(s)

Antoine Stevens

See Also

`resample2`

Examples

```
data(NIRsoil)
wav <- as.numeric(colnames(NIRsoil$spc))
spc <- 1/10^NIRsoil$spc # conversion to reflectance
resampled <- resample(spc,wav,1100:2498) # increase spectral resolution by 2
dim(spc);dim(resampled)
```

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

Description

Resample a data `matrix`, `data.frame` 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`	numeric `data.frame`, `matrix` or `vector` to resample
`wav`	a numeric `vector` giving the original band positions
`new.wav`	a numeric `vector` giving the new band positions
`fwhm`	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$spc))
spc <- 1/10^NIRsoil$spc # conversion to reflectance
# Plot 10 first spectra
matplot(wav,t(spc[1:10,]),type='l',xlab='Wavelength /nm',ylab='Reflectance')
# 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(spc,wav,new.wav,fwhm)
matpoints(as.numeric(colnames(aster)),t(aster[1:10,]),pch=1:5)
```

`savitzkyGolay` *Savitzky-Golay transformation*

Description

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

Usage

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

Arguments

`X`	a numeric `data.frame`, `matrix` or `vector` to transform
`m`	differentiation order
`p`	polynomial order
`w`	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

References

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

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

See Also

sgolayfilt

Examples

```
data(NIRsoil)
spc <- 1/10^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(as.numeric(colnames(spc)),t(spc[1:10,]),type='l',xlab='',ylab='Reflectance')
mtext('Raw spectra')
sg <- savitzkyGolay(X = spc,1,3,11,delta.wav=2)
matplot(as.numeric(colnames(sg)),t(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	numeric `data.frame` or `matrix`
d.min	minimum distance (default = 0.6)

pc — 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 value. if `TRUE`, remove observations with a standardized mahalanobis distance to the center of the data greater than 3 (default = `FALSE`)

.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.

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

`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`	numeric `data.frame`, `matrix` or `vector` to transform
`wav`	numeric `vector` with band positions
`splice`	numeric `vector` of the two positions of the splices, default = c(1000,1830) corresponding to the splices of the ASD FieldSpec Pro spectrometer.
`interpol.bands`	number of interpolation bands

Details

Spectra acquired with an ASD FieldSpec Pro spectroradiometer usually exhibit steps at the splice of the three built-in sensors, positioned at 1000 nm (end of VNIR detector) and 1830 nm (end of SWIR1 detector).

Value

a `matrix` with the splice corrected data

Author(s)

Antoine Stevens

`standardNormalVariate` *Standard normal variate transformation*

Description

`standardNormalVariate` normalizes each row of an input `data.frame` or `matrix` by substracting each row by its mean and dividing by its standard deviation

Usage

```
standardNormalVariate(X)
```

Arguments

`X`	numeric `data.frame` or `matrix` to transform

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 a sample i, $\bar{x_i}$ is its mean and s_i its standard deviation

Value

a `matrix` of the transformed 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

`detrend`, `blockScale`, `blockNorm`, `msc`

Examples

```
data(NIRsoil)
spc <- 1/10^NIRsoil$spc # conversion to reflectance
snv <- standardNormalVariate(X = spc)
# 10 first snv spectra
matplot(as.numeric(colnames(snv)),t(snv[1:10,]),type='l',xlab='wavelength /nm',ylab='snv')
## Not run:
apply(snv,1,sd) # check

## End(Not run)
```

Index