Package 'baseline'

January 27, 2015

Encoding latin1

Type Package

Title Baseline Correction of Spectra

Version 1.1-4

Date 2015-01-20

Maintainer Kristian Hovde Liland <kristian.liland@nmbu.no>

Description Collection of baseline correction algorithms, along with a framework and a GUI for optimising baseline algorithm parameters.

License GPL-2

Depends R (>= 2.15), methods

Imports graphics, SparseM

Suggests gWidgets, IDPmisc, lattice, pls, MASS

LazyLoad true

LazyData true

ZipData true

NeedsCompilation no

Author Kristian Hovde Liland [aut, cre],
BjÃ¸rn-Helge Mevik [aut]

Repository CRAN

Date/Publication 2015-01-21 00:33:30

R topics documented:

`baseline-package` *Baseline correction*

Description

A common framework with implementations of several baseline correction methods

Details

Package:	baseline
Type:	Package
Version:	1.1-4
Date:	2015-01-19
License:	GPL-2

Use function baseline for baseline correction. This function takes matrices of spectra, a method name and parameters needed for the specific method. See helpfiles for details.

Author(s)

Kristian Hovde Liland and Bjørn-Helge Mevik

Maintainer: Kristian Hovde Liland <kristian.liland@umb.no>

References

Andreas F. Ruckstuhl, Matthew P. Jacobson, Robert W. Field, James A. Dodd: Baseline subtraction using robust local regression estimation; CHAD A. LIEBER and ANITA MAHADEVAN-JANSEN: Automated Method for Subtraction of Fluorescence from Biological Raman Spectra; Mark S. Friedrichs: A model-free algorithm for the removal of baseline artifacts; AHMET K. ATAKAN, W. E. BLASS, and D. E. JENNINGS: Elimination of Baseline Variations from a Recorded Spectrum by Ultra-low Frequency Filtering; M.A. Kneen, H.J. Annegarn: Algorithm for fitting XRF, SEM and PIXE X-ray spectra backgrounds; K.H. Liland, B.-H. Mevik, E.-O. Rukke, T. Almøy, M. Skaugen and T. Isaksson (2009) Quantitative whole spectrum analysis with MALDI-TOF MS, Part I: Measurement optimisation. *Chemometrics and Intelligent Laboratory Systems*, **96**(2), 210–218.

Examples

```
data(milk)
bc.irls <- baseline(milk$spectra[1,, drop=FALSE])
## Not run:
plot(bc.irls)

## End(Not run)
```

`algorithm` *Extraction methods for "baselineAlgTest" objects*

Description

Extraction methods specifically for objects of class `baselineAlgTest`

Usage

```
algorithm(object)
extraArgs(object)
```

Arguments

object	Object of class baselineAlgTest

Value

The corresponding slot

Author(s)

Bjørn-Helge Mevik and Kristian Hovde Liland

See Also

baselineAlgTest

---

baseline *Baseline correction*

Description

Common framework for baseline correction

Usage

```
baseline(spectra, method = "irls", ...)
```

Arguments

spectra	Matrix with spectra in rows
method	Baseline correction method
...	Additional parameters, sent to the method

Details

Estimates baselines for the spectra, using the algorithm named in method.

Value

An object of class baseline.

Author(s)

Kristian Hovde Liland and Bjørn-Helge Mevik

See Also

The functions implementing the baseline algorithms: baseline.als, baseline.fillPeaks, baseline.irls, baseline.lowpass, baseline.medianWindow, baseline.modpolyfit, baseline.peakDetection, baseline.rfbaseline, baseline.rollingBall

Examples

```
data(milk)
bc.irls <- baseline(milk$spectra[1,, drop=FALSE])
## Not run:
plot(bc.irls)

## End(Not run)
```

---

baseline-class *Class "baseline"*

Description

Stores the result of estimating baselines for one or more spectra.

Objects from the Class

The normal way to create objects is with the function baseline. Several baseline algorithms are available. See baseline for details. There is a plot method for the class; see plot,baseline-method.

Slots

baseline: A matrix with the estimated baselines

corrected: A matrix with the corrected spectra

spectra: A matrix with the original spectra

call: The call to baseline

Methods

getBaseline signature(object = "baseline"): Extract the estimated baselines

getCall signature(object = "baseline"): Extract the call to baseline used to create the object

getCorrected signature(object = "baseline"): Extract the corrected spectra

getSpectra signature(object = "baseline"): Extract the original spectra

Warning

In a future versoion, one of the slots might be removed from the class definition and calculated on the fly instead, in order to save space. Therefore, *do* use the extractor functions (getSpectra, getBaseline and getCorrected) instead of accessing the slots directly.

Author(s)

Bjørn-Helge Mevik and Kristian Hovde Liland

See Also

baseline, getBaseline, getSpectra, getCorrected, getCall

Examples

```
showClass("baseline")
```

---

baseline.als *Asymmetric Least Squares*

Description

Baseline correction by 2nd derivative constrained weighted regression. Original algorithm proposed by Paul H. C. Eilers and Hans F.M. Boelens

Usage

```
baseline.als(spectra, lambda = 6, p = 0.05, maxit = 20)
```

Arguments

spectra	Matrix with spectra in rows
lambda	2nd derivative constraint
p	Weighting of positive residuals
maxit	Maximum number of iterations

Details

Iterative algorithm applying 2nd derivative constraints. Weights from previous iteration is p for positive residuals and 1-p for negative residuals.

Value

baseline	Matrix of baselines corresponding to spectra spectra
corrected	Matrix of baseline corrected spectra
wgts	Matrix of final regression weights

Author(s)

Kristian Hovde Liland and Bjørn-Helge Mevik

References

Paul H. C. Eilers and Hans F.M. Boelens: Baseline Correction with Asymmetric Least Squares Smoothing

Examples

```
data(milk)
bc.als <- baseline(milk$spectra[1,, drop=FALSE], lambda=10, method='als')
## Not run:
plot(bc.als)

## End(Not run)
```

`baseline.fillPeaks` *Fill peaks*

Description

An iterative algorithm using suppression of baseline by means in local windows

Usage

```
baseline.fillPeaks(spectra, lambda, hwi, it, int)
```

Arguments

`spectra`	Matrix with spectra in rows
`lambda`	2nd derivative penalty for primary smoothing
`hwi`	Half width of local windows
`it`	Number of iterations in suppression loop
`int`	Number of buckets to divide spectra into

Details

In local windows of buckets the minimum of the mean and the previous iteration is chosen as the new baseline

Value

`baseline`	Matrix of baselines corresponding to spectra `spectra`
`corrected`	Matrix of baseline corrected spectra

Author(s)

Kristian Hovde Liland and Bjørn-Helge Mevik

Examples

```
data(milk)
bc.fillPeaks <- baseline(milk$spectra[1,, drop=FALSE], lambda=6,
hwi=50, it=10, int=2000, method='fillPeaks')
## Not run:
plot(bc.fillPeaks)

## End(Not run)
```

`baseline.irls` *Iterative Restricted Least Squares*

Description

An algorithm with primary smoothing and repeated baseline suppressions and regressions with 2nd derivative constraint

Usage

```
baseline.irls(spectra, lambda1 = 5, lambda2 = 9, maxit = 200, wi = 0.05)
```

Arguments

`spectra`	Matrix with spectra in rows
`lambda1`	2nd derivative constraint for primary smoothing
`lambda2`	2nd derivative constraint for secondary smoothing
`maxit`	Maximum number of iterations
`wi`	Weighting of positive residuals

Value

`baseline`	Matrix of baselines corresponding to spectra `spectra`
`corrected`	Matrix of baseline corrected spectra
`smoothed`	Matrix of primary smoothed spectra

Author(s)

Kristian Hovde Liland and Bjørn-Helge Mevik

Examples

```
data(milk)
bc.irls <- baseline(milk$spectra[1,, drop=FALSE], method='irls')
## Not run:
plot(bc.irls)

## End(Not run)
```

`baseline.lowpass` *Low-pass FFT filter*

Description

An algorithm for removing baselines based on Fast Fourier Transform filtering

Usage

```
baseline.lowpass(spectra, steep = 2, half = 5)
```

Arguments

`spectra`	Matrix with spectra in rows
`steep`	Steepness of filter curve
`half`	Half-way point of filter curve

Details

Since the scale of the spectra will be different after filtering, baselines will not be returned by the algorithm

Value

`baseline`	Matrix of baselines corresponding to spectra `spectra`
`corrected`	Matrix of baseline corrected spectra

Author(s)

Kristian Hovde Liland and Bjørn-Helge Mevik

References

AHMET K. ATAKAN, W. E. BLASS, and D. E. JENNINGS: Elimination of Baseline Variations from a Recorded Spectrum by Ultra-low Frequency Filtering

Examples

```
data(milk)
bc.lowpass <- baseline(milk$spectra[1,, drop=FALSE], method='lowpass')
## Not run:
plot(bc.lowpass)

## End(Not run)
```

baseline.medianWindow *Median window*

Description

An implementation and extention of Mark S. Friedrichs' model-free algorithm

Usage

```
baseline.medianWindow(spectra, hwm, hws, end)
```

Arguments

spectra	Matrix with spectra in rows
hwm	Window half width for local medians
hws	Window half width for local smoothing (optional)
end	Original endpoint handling (optional boolean)

Details

An algorithm finding medians in local windows and smoothing with gaussian weighting

Value

baseline	Matrix of baselines corresponding to spectra spectra
corrected	Matrix of baseline corrected spectra

Author(s)

Kristian Hovde Liland and Bjørn-Helge Mevik

References

Mark S. Friedrichs: A model-free algorithm for the removal of baseline artifacts

Examples

```
data(milk)
bc.medianWindow <- baseline(milk$spectra[1,, drop=FALSE], hwm=300,
method='medianWindow')
## Not run:
plot(bc.medianWindow)

## End(Not run)
```

`baseline.modpolyfit` *Modified polynomial fitting*

Description

An implementation of CHAD A. LIEBER and ANITA MAHADEVAN-JANSENs algorithm for polynomial fiting

Usage

```
baseline.modpolyfit(spectra, t, degree = 4, tol = 0.001, rep = 100)
```

Arguments

`spectra`	Matrix with spectra in rows
`t`	Optional vector of spectrum abcissa
`degree`	Degree of polynomial
`tol`	Tolerance of difference between iterations
`rep`	Maximum number of iterations

Details

Polynomial fitting with baseline suppression relative to original spectrum

Value

`baseline`	Matrix of baselines corresponding to spectra `spectra`
`corrected`	Matrix of baseline corrected spectra

Author(s)

Kristian Hovde Liland and Bjørn-Helge Mevik

References

CHAD A. LIEBER and ANITA MAHADEVAN-JANSEN: Automated Method for Subtraction of Fluorescence from Biological Raman Spectra

Examples

```
data(milk)
bc.modpolyfit <- baseline(milk$spectra[1,, drop=FALSE], method='modpolyfit', deg=6)
## Not run:
plot(bc.modpolyfit)

## End(Not run)
```

`baseline.peakDetection` *Simultaneous Peak Detection and Baseline Correction*

Description

A translation from Kevin R. Coombes et al.'s MATLAB code for detecting peaks and removing baselines

Usage

```
baseline.peakDetection(spectra, left, right, lwin, rwin, snminimum,
mono=0, multiplier=5, left.right, lwin.rwin)
```

Arguments

`spectra`	Matrix with spectra in rows
`left`	Smallest window size for peak widths
`right`	Largest window size for peak widths
`lwin`	Smallest window size for minimums and medians in peak removed spectra
`rwin`	Largest window size for minimums and medians in peak removed spectra
`snminimum`	Minimum signal to noise ratio for accepting peaks
`mono`	Monotonically decreasing baseline if `mono`>0
`multiplier`	Internal window size multiplier
`left.right`	Sets eflt and right to value of `left.right`
`lwin.rwin`	Sets lwin and rwin to value of `lwin.rwin`

Details

Peak detection is done in several steps sorting out real peaks through different criteria. Peaks are removed from spectra and minimums and medians are used to smooth the remaining parts of the spectra. If `snminimum` is omitted, y3, midspec, y and y2 are not returned (faster)

Value

`baseline`	Matrix of baselines corresponding to spectra `spectra`
`corrected`	Matrix of baseline corrected spectra
`peaks`	Final list of selected peaks
`sn`	List signal to noise ratios for peaks
`y3`	List of peaks prior to singal to noise selection
`midspec`	Mid-way baseline estimation
`y`	First estimate of peaks
`y2`	Second estimate of peaks

Author(s)

Kristian Hovde Liland and Bjørn-Helge Mevik

References

KEVIN R. COOMBES et al.: Quality control and peak finding for proteomics data collected from nipple aspirate fluid by surface-enhanced laser desorption and ionization.

Examples

```
data(milk)
bc.peakDetection <- baseline(milk$spectra[1,, drop=FALSE], method='peakDetection',
left=300, right=300, lwin=50, rwin=50)
## Not run:
plot(bc.peakDetection)

## End(Not run)
```

---

baseline.rfbaseline *Robust Baseline Estimation*

Description

Wrapper for Andreas F. Ruckstuhl, Matthew P. Jacobson, Robert W. Field, James A. Dodd's algorithm based on LOWESS and weighted regression

Usage

```
baseline.rfbaseline(spectra, span = 2/3, NoXP = NULL, maxit = c(2, 2),
  b = 3.5, weight = NULL, Scale = function(r) median(abs(r))/0.6745,
  delta = NULL, SORT = FALSE, DOT = FALSE, init = NULL)
```

Arguments

spectra	Matrix with spectra in rows
span	Amount of smoothing (by fraction of points)
NoXP	Amount of smoothing (by number of points)
maxit	Maximum number of iterations in robust fit
b	Tuning constant in the biweight function
weight	Optional weights to be given to individual observations
Scale	S function specifying how to calculate the scale of the residuals
delta	Nonnegative parameter which may be used to save computation. (See rfbaseline
SORT	Boolean variable indicating whether x data must be sorted.
DOT	Disregard outliers totally (boolean)
init	Values of initial fit

Details

Most of the code is the original code as given by the authors. The ability to sort by X-values has been removed and ability to handle multiple spectra has been added

Value

`baseline`	Matrix of baselines corresponding to spectra `spectra`
`corrected`	Matrix of baseline corrected spectra

Author(s)

Kristian Hovde Liland and Bjørn-Helge Mevik

References

Andreas F. Ruckstuhl, Matthew P. Jacobson, Robert W. Field, James A. Dodd: Baseline subtraction using robust local regression estimation

Examples

```
data(milk)
bc.rbe <- baseline(milk$spectra[1,, drop=FALSE], method='rfbaseline',
  span=NULL, NoXP=1000)
## Not run:
plot(bc.rbe)

## End(Not run)
```

`baseline.rollingBall` *Rolling ball*

Description

Ideas from Rolling Ball algorithm for X-ray spectra by M.A.Kneen and H.J. Annegarn. Variable window width has been left out

Usage

```
baseline.rollingBall(spectra, wm, ws)
```

Arguments

`spectra`	Matrix with spectra in rows
`wm`	Width of local window for minimization/maximization
`ws`	Width of local window for smoothing

Value

baseline	Matrix of baselines corresponding to spectra spectra
corrected	Matrix of baseline corrected spectra

Author(s)

Kristian Hovde Liland and Bjørn-Helge Mevik

References

M.A. Kneen, H.J. Annegarn: Algorithm for fitting XRF, SEM and PIXE X-ray spectra backgrounds

Examples

```
data(milk)
bc.rollingBall <- baseline(milk$spectra[1,, drop=FALSE], wm=200, ws=200,
method='rollingBall')
## Not run:
plot(bc.rollingBall)

## End(Not run)
```

---

baselineAlg-class *Class "baselineAlg"*

Description

A class that describes a baseline correction algorithm. The idea is that it contains all information needed to use an algorithm with the optimisation framework and the graphical user interface (but see Notes below).

Objects from the Class

Objects can be created by calls of the form new("baselineAlg", ...).

Slots

name: Short-name of the algorithm. This must match the name of the object in the baselineAlgorithms list of algorithms, and is used throughout the code to identify the algorithm. It should thus start with a letter and contain only letters, digits, underscores ("_") or dots (".").

description: Description of the algorithm, typically the full name. This will be used in the code to describe the algorith, so it should not be too long, and not contain newline characters.

funcName: The name of the function used to estimate the baseline. The function must take an argument spectra, and return a list with the estimated baselines (baseline) original spectra (spectra) and the corrected spectra (corrected). It can also take other arguments (typically parameters) and return additional components in the list.

`param`: A data frame with information about the parameters of the algorithm. It should contain the following coloumns: `name` - the name of the parameter; `integer` - `TRUE` if the parameter only takes integer values, otherwise `FALSE`; `min` - the lower limit of allowed values; `incl.min` - `TRUE` if the lower limit is an allowed value, otherwise `FALSE`; `default` - the default value; `max` - the upper limit of allowed values; `incl.max` - `TRUE` if the upper limit is an allowed value, otherwise `FALSE`

Methods

description `signature(object = "baselineAlg")`: Extract the `description` slot

funcName `signature(object = "baselineAlg")`: Extract the `funcName` slot

name `signature(object = "baselineAlg")`: Extract the `name` slot

param `signature(object = "baselineAlg")`: Extract the `param` slot

Note

The goal is that the optimisation framework and the GUI code should get all information about available baseline algorithms through a list of `baselineAlg` objects. This will make it relatively simple to add new baseline algorithms.

Currenly, there is information about the algorithms spread around in the code. We plan to move that information into the `baselineAlg` objects, and expand the class accordingly.

Author(s)

Bjørn-Helge Mevik and Kristian Hovde Liland

Examples

```
showClass("baselineAlg")
```

`baselineAlgorithms` *List of available baseline algorithms*

Description

A list with descriptions of all baseline algorithms available through the optimisaiont framework and graphical user interface. The elements of the list are `baselineAlg` objects. The list is used by the code to extract names and information about the baseline algorithms.

Details

The list is not meant for usage by end-users, but is extendable and customizable, allowing for extra algorithms or removal of algoritms.

The names of the list must match the `name` slot of the elements.

Examples

```
## Get a list of all algorithms:
names(baselineAlgorithms)
## Show the descriptions
sapply(baselineAlgorithms, description)
## Add new algorithm
baseline.my.alg <- function(spectra, kappa=1, gamma=1){
   baseline  <- spectra-kappa+gamma
   corrected <- spectra-baseline
   list(baseline=baseline,corrected=corrected)
}

baselineAlgorithms$my.alg = new("baselineAlg",
     name = "my.alg",
     description = "A new baseline correction algorithm",
     funcName = "baseline.my.alg",
     param = data.frame(
        name = c("kappa","gamma"), # maxit
        integer = c(FALSE, FALSE),
        min = c(0, 0),
        incl.min = c(TRUE, TRUE),
        default = c(1, 1),
        max = c(Inf, 1),
        incl.max = c(FALSE, TRUE)
    ))
```

---

baselineAlgorithmsGUI *List of available baseline algorithms for GUI function*

Description

A list with data.frames containing parameters, minimum and maximum values for GUIs, step lengths for sliders, default values and currently selected values, plus a short description of each parameter. The list is used by the GUIs, and is user customizable.

Details

The list is not meant for usage by end-users, but is extendable and customizable, allowing for extra algorithms, removal of algoritms or changing of parameter sets.

Examples

```
## Get a list of all algorithms:
names(baselineAlgorithmsGUI)
## Add new algorithm:
baselineAlgorithmsGUI$my.alg <- as.data.frame(matrix(c(0,20,1,1, 0,20,1,1), 2,4, byrow=TRUE))
dimnames(baselineAlgorithmsGUI$my.alg) <- list(par=c("kappa", "gamma"),
val=c("min","max","step","default"))
baselineAlgorithmsGUI$my.alg$current <- c(1,1)
baselineAlgorithmsGUI$my.alg$name <- c("Subtractive constand", "Additive constant")
```

baselineAlgResult-class
Class "baselineAlgResult"

Description

A class describing the result of a baseline algorithm test

Objects from the Class

Objects are typically created by running `runTest` on a `baselineAlgTest` object.

Slots

`param`: A named list with the parameter values that were tested. This includes both the predictor parameters and the baseline algorithm parameters. All combinations of values are tested.

`qualMeas`: A matrix of quality measure values for the different combinations of parameter values. Each row corresponds to one prediction parameter value, and each coloumn to one combination of baseline parameters.

`qualMeas.ind.min`: The index in `qualMeas` of the minimum quality measure value

`minQualMeas`: The minimum quality measure value

`param.ind.min`: A vector of indices into the elemets of `param` of the parameter values corresponding to the minimum quality measure value

`param.min`: A list of the parameter values corresponding to the minimum quality measure value

`qualMeasName`: The name of the quality measure

Methods

minQualMeas `signature(object = "baselineAlgResult")`: Extract the `minQualMeas` slot

param `signature(object = "baselineAlgResult")`: Extract the `param` slot

param.ind.min `signature(object = "baselineAlgResult")`: Extract the `param.ind.min` slot

param.min `signature(object = "baselineAlgResult")`: Extract the `param.min` slot

qualMeas `signature(object = "baselineAlgResult")`: Extract the `qualMeas` slot

qualMeas.ind.min `signature(object = "baselineAlgResult")`: Extract the `qualMeas.ind.min` slot

qualMeasName `signature(object = "baselineAlgResult")`: Extract the `qualMeasName` slot

Author(s)

Bjørn-Helge Mevik and Kristian Hovde Liland

See Also

Class `baselineAlgTest`, function `runTest`.

Examples

```
showClass("baselineAlgResult")
```

`baselineAlgTest-class` *Class "baselineAlgTest"*

Description

A class that describes a baseline algorithm test. The test is performed with the function `runTest`.

Objects from the Class

Objects can be created by calls of the form `new("baselineAlgTest", ...)`.

Slots

`algorithm`**:** A `"baselineAlg"` object. The baseline algorithm to test.

`param`**:** A named list with parameter values to test. All combinations of parameters are tested.

`extraArgs`**:** A named list of extra parameters to the baseline algorithm. These will be held fixed during the testing.

Methods

algorithm `signature(object = "baselineAlgTest")`: Extract the `algorithm` slot

extraArgs `signature(object = "baselineAlgTest")`: Extract the `extraArgs` slot ...

funcName `signature(object = "baselineAlgTest")`: Extract the `funcName` slot ...

param `signature(object = "baselineAlgTest")`: Extract the `param` slot

runTest `signature(object = "baselineAlgTest")`: Run the test.

Author(s)

Bjørn-Helge Mevik and Kristian Hovde Liland

See Also

Classes `baselineAlg`, `baselineAlgResult`. Function `runTest`.

Examples

```
showClass("baselineAlgTest")
```

baselineEnv *Baseline environment*

Description

Methods to access the baseline environment.

Usage

```
baselineEnv()
getBaselineEnv(x, mode="any")
putBaselineEnv(x, value)
```

Arguments

x	Name of object to put/get.
mode	Mode of object to get.
value	Object to put.

Value

getBaseline retrieves an object.

Author(s)

Kristian Hovde Liland and Bjørn-Helge Mevik

See Also

The functions implementing the baseline algorithms: baseline.als, baseline.fillPeaks, baseline.irls, baseline.lowpass, baseline.medianWindow, baseline.modpolyfit, baseline.peakDetection, baseline.rfbaseline, baseline.rollingBall

Examples

```
putBaselineEnv('fish', '<==x-<')
getBaselineEnv('fish')
```

baselineGUI *Interactive plotting tool*

Description

An interactive plotting tool for dynamic visualization of baselines and their effect using the gWidgets package with GTK+ or Tcl/Tk.

Usage

```
baselineGUI(spectra, method='irls', labels, rev.x = FALSE)
```

Arguments

spectra	Matrix with spectra in rows
method	Baseline correction method (optional)
labels	Labels for X-axis (optional)
rev.x	Reverse X-axis (optional, default=FALSE)

Details

Creates and updates a list containing current baseline and spectrum (baseline.result). Make sure a gWidget implementation is available, e.g gWidgetsRGtk2 or gWidgetstcltk and a corresponding backend like GTK+ or Tcl/Tk. The GUI was developed using GTK which is an external dependency in Windows ans OS X.

Author(s)

Kristian Hovde Liland and Bjørn-Helge Mevik

Examples

```
data(milk)
## Not run: baselineGUI(milk$spectra)
```

custom.baseline *Customized baseline correction*

Description

This function rescales spectrum abscissa by use of breaks and gaps before baseline correction. The effect is that the chosen baseline correction algorithm and paramters will have varying effects along the spectra, effectively giving local control of the amount of rigidity/flexibility of the estimated baseline.

Usage

```
custom.baseline(spectra, breaks, gaps, trans.win = NULL, just.plot = FALSE, method, ...)
```

Arguments

spectra	Matrix with spectra in rows.
breaks	Vector of locations of break points between sections of varying baseline flexibility (given as abscissa numbers).
gaps	Vector giving the abscissa spacing between each instance of breaks (and endpoints if not specified in breaks).
trans.win	Optional width of transition window around break points used for smoothing rough breaks by LOWESS (default = NULL).
just.plot	Plot the rescaled spectra instead of applying the customized baseline correction if just.plot=TRUE (default = FALSE).
method	Baseline correction method to use (class character).
...	Additional named arguments to be passed to the baseline correction method.

Details

This is an implementation of the customized baseline correction suggested by Liland et al. 2011 for local changes in baseline flexibility.

Value

baseline	Estimated custom baselines.
corrected	Spectra corrected by custom baselines.
spectra.scaled	Re-scaled spectra.
baseline.scaled	Estimated baselines of re-scaled spectra.

Author(s)

Kristian Hovde Liland and Bjørn-Helge Mevik

References

Kristian Hovde Liland et al.: Customized baseline correction

Examples

```
data(milk)
spectrum1  <- milk$spectra[1,1:10000,drop=FALSE]
ordinary   <- baseline(spectrum1, method="als", lambda=6, p=0.01)
customized <- custom.baseline(spectrum1, 2900, c(1,20), trans.win=100,
just.plot=FALSE, method="als", lambda=6, p=0.01)
## Not run:
```

```
plot(1:10000,spectrum1, type='l')
lines(1:10000,getBaseline(ordinary), lty=2, col=2, lwd=2)
lines(1:10000,customized$baseline, lty=3, col=3, lwd=2)

## End(Not run)
```

`doOptim` *Optimise several baseline algorithms on a data set*

Description

Tests several baseline algorithms with one predictor for a given data set. The baseline algorithms are represented as a list of `baselineAlgTest` objects, and the predictor as a `predictionTest` object.

Usage

```
doOptim(baselineTests, X, y, predictionTest, postproc = NULL,
        tmpfile = "tmp.baseline", verbose = FALSE, cleanTmp = FALSE)
```

Arguments

`baselineTests`	a list of `baselineAlgTest` objects. The baseline algorithms and parameter values to test
`X`	A matrix. The spectra to use in the test
`y`	A vector or matrix. The response(s) to use in the test
`predictionTest`	A `predictionTest` object. The predictor and parameter values to use in the test
`postproc`	A function, used to postprocess the baseline corrected spectra prior to prediction testing. The function should take a matrix of spectra as its only argument, and return a matrix of postprocessed spectra
`tmpfile`	The basename of the files used to store intermediate calculations for checkpointing. Defaults to `"tmp.baseline"`
`verbose`	Logical, specifying whether the test should print out progress information. Default is `FALSE`
`cleanTmp`	Logical, specifying whether the intermediate files should be deleted when the optimisation has finished. Default is `FALSE`

Details

The function loops through the baseline algorithm tests in `baselineTests`, testing each of them with the given data and prediction test, and collects the results. The results of each baseline algorithm test is saved in a temporary file so that if the optimisation is interrupted, it can be re-run and will use the pre-calculated results. If `cleanTmp` is `TRUE`, the temporary files are deleted when the whole optimisation has finished.

Value

A list with components

baselineTests	The baselineTests argument
results	A list with the baselineAlgResult objects for each baseline test
minQualMeas	The minimum quality measure value
baselineAlg.min	The name of the baseline algorithm giving the minimum quality measure value
param.min	A list with the parameter values corresponding to the minimum quality measure value

Author(s)

BjÃ¸rn-Helge Mevik and Kristian Hovde Liland

See Also

baselineAlgTest,predictionTest

---

funcName *Extract the "funcName" slot.*

Description

Extract the funcName slot from an object of class baselineAlg or baselineAlgTest

Usage

```
funcName(object)
```

Arguments

object	An object of class baselineAlg or baselineAlgTest

Value

The funcName slot of the object.

Author(s)

Bjørn-Helge Mevik and Kristian Hovde Liland

See Also

baselineAlg, baselineAlgTest

getBaseline *Functions to extract the components of a "baseline" object*

Description

The functions extract the baseline, spectra, corrected or call slot of a baseline object; usually the result of a call to baseline.

Usage

```
getBaseline(object)
getSpectra(object)
getCorrected(object)
getCall(object)
```

Arguments

object	A baseline object

Value

getCall returns the baseline call used to create the object. The other functions return a matrix with the original spectra, estimated baselines or corrected spectra.

Warning

In a future versoion, one of the slots might be removed from the class definition and calculated on the fly instead, in order to save space. Therefore, *do* use the extractor functions (getSpectra, getBaseline and getCorrected) instead of accessing the slots directly.

Author(s)

Bjørn-Helge Mevik and Kristian Hovde Liland

See Also

The function baseline, the class baseline

Examples

```
data(milk)
bl <- baseline(milk$spectra[1:2,])
getBaseline(bl)
getSpectra(bl)
getCorrected(bl)
getCall(bl)
```

ind.min *Extraction methods specific for "predictionResult" objects*

Description

Extract information from objects of class predictionResult.

Usage

```
ind.min(object)
paramName(object)
```

Arguments

object	Object of class predictionResult

Value

The corresponding slot of the object.

Author(s)

Bjørn-Helge Mevik and Kristian Hovde Liland

See Also

predictionResult

milk *MALDI-TOF mass spectra*

Description

Matrix of 45 spectra of 21451 m/z values from MALDI-TOF on mixed milk samples.

Usage

```
data(milk)
```

Format

A data frame with 45 observations on the following 2 variables.

cow a numeric vector

spectra a matrix with 21451 columns

Details

cow is the concentration of cow milk in mixed samples of cow, goat, and ewe milk.

References

Kristian Hovde Liland, Bjørn-Helge Mevik, Elling-Olav Rukke, Trygve Almøy, Morten Skaugen and Tomas Isaksson (2009) Quantitative whole spectrum analysis with MALDI-TOF MS, Part I: Measurement optimisation. *Chemometrics and Intelligent Laboratory Systems*, **96**(2), 210–218.

Examples

```
data(milk)
## Not run:
plot(milk$spectra[1,], type = "l")

## End(Not run)
```

---

name *Extraction methods for "baselineAlg" objects*

Description

Extraction methods specifically for objects of class baselineAlg

Usage

```
name(object)
description(object)
```

Arguments

object	Object of class baselineAlg

Value

The methods return the corresponding slot of the object.

Author(s)

Bjørn-Helge Mevik and Kristian Hovde Liland

See Also

baselineAlg, funcName.

`optimWizard` *Visual tool for setting up optimization*

Description

Set up optimization through a graphical user interface. Optionally collecting values directly from 'baselineGUI'. Retrieve optimisation parameters and results with `getOptim` and `getOptimRes`, respectively.

Usage

```
optimWizard(X, y, postproc, predictionTest, cvsegments)
getOptim()
getOptimRes()
```

Arguments

`X`	Matrix with spectra in rows
`y`	Response vector or matrix in analysis
`postproc`	Custum function for post processing of spectra (optional)
`predictionTest`	Custom prediction object (optional)
`cvsegments`	Cross-validation segments (optional)

Author(s)

Kristian Hovde Liland and Bjørn-Helge Mevik

Examples

```
## Not run:
data(milk)
X <- milk$spectra[,-1]
y <- milk$spectra[,1]
optimWizard(X,y)

# After optimisation is complete
plotOptim(myResults)

## End(Not run)
```

overall.min *Extract the minimum from a baseline optimisation*

Description

Takes the result of an optimisation (a call to doOptim) and extracts the minimum quality measure value along with the parameters giving rise to the value.

Usage

```
overall.min(results)
```

Arguments

results	Result of call to doOptim

Value

A list with components

qualMeas	The minimum quality measure value
algorithm	The name of the baseline algorithm corresponding to the minimum
param	A list with the parameter values corresponding to the minimum quality measure value

Author(s)

Bjørn-Helge Mevik and Kristian Hovde Liland

See Also

doOptim

param *Extract the "param" slot*

Description

Extracts the param slot of the object.

Usage

```
param(object)
```

Arguments

object	An object of class baselineAlg, baselineAlgTest, baselineAlgResult or predictionResult.

Value

The param slot of the object. Usually a data frame, list or numeric.

Author(s)

Bjørn-Helge Mevik and Kristian Hovde Liland

See Also

Classes baselineAlg, baselineAlgTest, baselineAlgResult, predictionResult

---

param.ind.min *Extraction methods for "baselineAlgResult" objects*

Description

Extraction methods that are specific for objects of class baselineAlgResult

Usage

```
param.ind.min(object)
qualMeas.ind.min(object)
```

Arguments

object	Object of class baselineAlgResult

Value

The corresponding slot

Author(s)

Bjørn-Helge Mevik and Kristian Hovde Liland

See Also

Class baselineAlgResult

plotBaseline *Plot method for "baseline" objects*

Description

Plot the original spectrum, the estimated baseline, and the corrected spectrum. Optionally zoom and pan plot, either with arguments or interactively.

Usage

```
## S4 method for signature 'baseline'
plot(x, y, specNo = 1, grid = FALSE, labels = 1:n, rev.x = FALSE,
     zoom = NULL, ...)
plotBaseline(x, y, specNo = 1, grid = FALSE, labels = 1:n, rev.x = FALSE,
             zoom = list(xz = 1, yz = 1, xc = 0, yc = 0), ...)
```

Arguments

x	The baseline object to be plotted
y	Unused. Ignored with a warning
specNo	The row number of the spectrum and baseline to plot. Defaults to 1
grid	Logical. Whether to show a grid or not. Defaults to FALSE
labels	Vector. Labels for the x tick marks. Defaults to 1:n
rev.x	Logical. Whether the spectrum should be reversed. Defaults to FALSE
zoom	Either TRUE (only for the plot method), which turns on the interactive zoom controls, or a list with components xz, xc, yz and yc, which specifies the desired zoom and pan. Defaults to no zoom or pan
...	Other arguments. Currently ignored

Details

The normal way to plot baseline objects is to use the plot method. The plotBaseline function is the underlying work-horse function, and is not meant for interactive use.

Note

Because the argument list of any plot method must start with x, y, and the plot method for the baseline class does not use the y argument, all arguments except x must be named explicitly. Positional matching will not work.

Author(s)

Kristian Hovde Liland and Bjørn-Helge Mevik

See Also

baseline, baseline, baselineGUI

Examples

```
data(milk)
bl <- baseline(milk$spectra[1,, drop=FALSE])
## Not run:
plot(bl)
plot(bl, zoom = TRUE)
## End(Not run)
```

`plotOptim` *Plotting tool for result objects from optimization*

Description

A graphical user interface for plotting optimisation results, either one algorithm at the time or comparing algorithms.

Usage

```
plotOptim(results)
```

Arguments

`results`	Result list from optimization

Details

`plotOptim` creates a user interface based on the supplied results. Curve and level plots from single algorithms or comparison of algorithms is avilable.

For single algorithms subsets, levels corresponding to local or global minima, and averages can be extracted for plotting. For comparison of algorithms levels corresponding to local or global minima can be used, or levels corresponding to the minimum when averaging over selected values of the regression parameter, e.g. selected components in PLSR.

Author(s)

Kristian Hovde Liland and Bjørn-Helge Mevik

PLSRTest-class *Class "PLSRTest"*

Description

A class describing a PLSR prediction test. To run the test, the "pls" package must be installed.

Objects from the Class

Objects can be created by calls of the form `new("PLSRTest", ...)`.

Slots

`ncomp`**:** Integer vector. The number of PLSR components to test

`cvsegments`**:** A list of the segments to use in the cross-validation

Extends

Class `predictionTest`, directly.

Methods

runTest `signature(object = "PLSRTest")`: Run the test

Author(s)

Bjørn-Helge Mevik and Krisitan Hovde Liland

See Also

The base class `predictionTest`. The `runTest` function. The `plsr` function from the "pls" package.

Examples

```
showClass("PLSRTest")
```

predictionResult-class *Class "predictionResult"*

Description

A class containing the result of running a predictionTest.

Objects from the Class

The normal way to create objects is by calling the method runTest for any object of subclass of predictionTest.

Slots

param: Numeric vector. The regression parameter values tested.

qualMeas: Numeric vector. The quality measure values for each of the values of the param slot

ind.min: The index (into qualMeas) of the minimum quality measure value

minQualMeas: The minimum quality measure value

param.min: The value of the parameter value corresponding to the minimum quality measure value

qualMeasName: The name of the quality measure

paramName: The name of the regression parameter

Methods

ind.min signature(object = "predictionResult"): Extract the ind.min slot

minQualMeas signature(object = "predictionResult"): Extract the minQualMeas slot

param signature(object = "predictionResult"): Extract the param slot

param.min signature(object = "predictionResult"): Extract the param.min slot

paramName signature(object = "predictionResult"): Extract the paramName slot

qualMeas signature(object = "predictionResult"): Extract the qualMeas slot

qualMeasName signature(object = "predictionResult"): Extract the qualMeasName slot

Author(s)

Bjørn-Helge Mevik and Kristian Hovde Liland

See Also

Function runTest, class predictionTest, subclasses PLSRTest and ridgeRegressionTest

Examples

```
showClass("predictionResult")
```

predictionTest-class *Class "predictionTest"*

Description

A virtual class for all predictor test subclasses. Currently subclasses PLSRTest and ridgeRegressionTest are defined.

Objects from the Class

A virtual Class: No objects may be created from it.

Methods

No methods defined with class "predictionTest" in the signature.

Author(s)

Bjørn-Helge Mevik and Kristian Hovde Liland

See Also

Subclasses PLSRTest and ridgeRegressionTest.

qualMeas *Extraction functions for "predictionResult" or "baselineAlgResult" objects*

Description

Extract slots from objects of class predictionResult or baselineAlgResult.

Usage

```
qualMeas(object, ...)
## S4 method for signature 'predictionResult'
qualMeas(object, ...)
## S4 method for signature 'baselineAlgResult'
qualMeas(object, ..., MIN, AVG,
  DEFAULT = c("all", "cond.min", "overall.min", "avg"))
minQualMeas(object)
param.min(object)
qualMeasName(object)
```

Arguments

object	An object of class predictionResult or baselineAlgResult
MIN	List or vector of parameter names to take the minimum over. Not used if DEFAULT is "cond.min". See Details
AVG	List or vector of parameter names to take the average over. Not used if DEFAULT is "avg". See Details
DEFAULT	Character string. The default way to calculate the minimum (or average) for all parameters. See Details
...	Other arguments. Selection of subsets of parameter levels. See Details

Details

The arguments to the baselineAlgResult method are interpreted in the following way:

Subsets of parameters levels can be selected by supplying their names and specifying the level indices as vectors. Substituting a vector with "all" will return all levels of the corresponding parameter, and substituting it with "overall" will return the level corresponding to the overall minimum. Minimum and average values for selected parameters can be chosen using MIN and AVG, respectively, together with a vector of parameter names.

DEFAULT specifies the action for each remaining parameters: If "all" (default): returns all levels. If "cond.min": take minimum for each remaining parameter (MIN is not used). If "overall.min": set any remaining parameters to their value corresponding to the overall min. If "avg": take average for each remaining parameter (AVG is not used).

Value

The qualMeas method for baselineAlgResult objects returns the subsets or minimum values of the qualMeas slot of the object as specified above. All other methods simply return the corresponding slot.

Author(s)

Bjørn-Helge Mevik and Kristian Hovde Liland

See Also

Function runTest, classes baselineAlgResult and predictionResult

---

ridgeRegressionTest-class

Class "ridgeRegressionTest"

Description

A class describing a ridge regression test.

Objects from the Class

Objects can be created by calls of the form `new("ridgeRegressionTest", ...)`.

Slots

`lambda`**:** Numeric vector. The smoothing parameter values to test

Extends

Class `predictionTest`, directly.

Methods

runTest `signature(object = "ridgeRegressionTest")`: Run the test

Author(s)

Bjørn-Helge Mevik and Kristian Hovde Liland

See Also

The base class `predictionTest`. The `runTest` function.

Examples

```
showClass("ridgeRegressionTest")
```

---

runTest *Run a predictionTest or baselineAlgTest*

Description

Runs the test defined in a `predictionTest` or `baselineAlgTest` object

Usage

```
runTest(object, X, y, ...)
## S4 method for signature 'PLSRTest'
runTest(object, X, y)
## S4 method for signature 'ridgeRegressionTest'
runTest(object, X, y)
## S4 method for signature 'baselineAlgTest'
runTest(object, X, y, predictionTest, postproc, verbose = FALSE)
```

Arguments

object	An object of class baselineAlgTest or subclass of predictionTest (currently PLSRTest or ridgeRegressionTest). The object specify the test to be run
X	A matrix. The spectra to use in the test
y	A vector or matrix. The response(s) to use in the test
predictionTest	A predictionTest object, describing the prediction test to use for this baseline algorithm test
postproc	A function, used to postprocess the baseline corrected spectra prior to prediction testing. The function should take a matrix of spectra as its only argument, and return a matrix of postprocessed spectra
verbose	Logical, specifying whether the test should print out progress information. Default is FALSE
...	Other arguments. Currently only used by the baselineAlgTest method.

Value

runTest returns an object of class predictionResult or baselineAlgResult.

Methods

signature(object = "baselineAlgTest") Baseline corrects the spectra, optionally postprocesses them, and runs a prediction test on the corrected spectra.

signature(object = "PLSRTest") Runs PLSR on the data and calculates the cross-validated RMSEP

signature(object = "ridgeRegressionTest") Runs ridge regression on the data and calculates the GCV

Author(s)

Bjørn-Helge Mevik and Kristian Hovde Liland

See Also

baselineAlgTest, predictionTest, PLSRTest, ridgeRegressionTest

Index