spectralAnalysis

October 22, 2018

R topics documented:

`baselineCorrect` *generic function to perfom baseline correction*

Description

generic function to perfom baseline correction

Usage

```
baselineCorrect(object, ...)

## S4 method for signature 'SpectraInTime'
baselineCorrect(object, method = "modpolyfit",
  degree = 4, ...)
```

Arguments

`object`	a S4 class object
`...`	other parameters passed to `baseline`
`method`	method of baseline correction, default value is to `'modpolyfit'`, see `baseline.modpolyfit`
`degree`	numeric value, degree of the polynomial used only if `method` is code'modpolyfit'

Note

baseline correction in the wavelength domain by linking to the `baseline`

Examples

```
spectralEx           <-  getSpectraInTimeExample()
plot( spectralEx )
timeRange            <-  range( getTimePoints( spectralEx ) )
timesToSelect        <-  e(  seq( timeRange[1] , timeRange[2] , length.out = 5  )   )
baselineDefault      <-  baselineCorrect( spectralEx )
baselineHighPolynomial  <-  baselineCorrect( spectralEx,
  method = 'modpolyfit', degree = 4 )

# filtering with fast fourier transform, not so good on example
baselineLowpass         <-  baselineCorrect( spectralEx , method = "lowpass" )

# visual inspection
plot( baselineDefault[ timesToSelect , ] , type = "time"  )
plot( baselineHighPolynomial[ timesToSelect , ] , type = "time"  )
plot( baselineLowpass[ timesToSelect , ] , type = "time"  )
```

`checkCompatible` *Check whether 2 objects are compatible before using them together For instance, same experiment name and matching time frames*

Description

Check whether 2 objects are compatible before using them together For instance, same experiment name and matching time frames

Usage

```
checkCompatible(x, y, ...)

## S4 method for signature 'SpectraInTime,ProcessTimes'
checkCompatible(x, y)

## S4 method for signature 'ProcessTimes,SpectraInTime'
checkCompatible(x, y)
```

Arguments

`x`	first object
`y`	second object
`...`	additional parameters

checkForRedundantSources *Check if any of the source vectors in the initialized NMF model are redundant, and should be omitted from the actual NMF analysis*

Description

Check if any of the source vectors in the initialized NMF model are redundant, and should be omitted from the actual NMF analysis

Usage

```
checkForRedundantSources(seed)
```

Arguments

seed	nmfModel object containing initialization of the factor matrices

Value

boolean vector, indicating which source vector(s) are redundant

Author(s)

Nicolas Sauwen

checkIdenticalClass *check wether all elements of of the same class*

Description

check wether all elements of of the same class

Usage

```
checkIdenticalClass(listOfObjects, class)
```

Arguments

listOfObjects	a list of S4 objects to check
class	a class to compare with

Value

logical value TRUE if all objects are of the correct class

Author(s)

Adriaan Blommaert

computeNMFResidu — *Compute relative residual per observation of an NMF fit to a spectral data set*

Description

Compute relative residual per observation of an NMF fit to a spectral data set

Usage

```
computeNMFResidu(object, NMFResult)
```

Arguments

object	SpectraInTime-class
NMFResult	Fitted NMF model

Value

Dataframe, containing time (observation) vector and residual vector

Author(s)

nsauwen

e — *Create an ElementsToSelect-class from a numeric vector or multiple numeric values or vectors*

Description

Create an ElementsToSelect-class from a numeric vector or multiple numeric values or vectors

Usage

```
e(x, ...)
```

Arguments

x	numeric vector
...	additional numeric vectors

Value

ElementsToSelect-class with unique elements

Examples

```
e( 1 , 5, 4.5  )
e( 1:10 , c(4 , 5 , 6 ) , 7 )
```

ElementsToSelect-class *Elements S4 class useful for closest elements subsetting*

Description

Elements S4 class useful for closest elements subsetting

Slots

`elements` numeric vector of elements

Author(s)

Adriaan Blommaert

firstSpectrum *Get the first spectrum*

Description

Get the first spectrum

Usage

```
firstSpectrum(object, ...)

## S4 method for signature 'SpectraInTime'
firstSpectrum(object)

## S4 method for signature 'numeric'
firstSpectrum(object)
```

Arguments

`object`	S4 object
`...`	additional parameters

getDefaultSumFunc *function to get default summary functions*

Description

function to get default summary functions

Usage

```
getDefaultSumFunc()
```

Value

character vector of functions

getDefaultTimeFormat *function to get default time format in the package*

Description

function to get default time format in the package

Usage

```
getDefaultTimeFormat()
```

Value

character vector specifiying a time format

getElements *generic function to extract* elements*-slot*

Description

generic function to extract elements-slot

Usage

```
getElements(object, ...)

## S4 method for signature 'ElementsToSelect'
getElements(object)
```

Arguments

object	a S4 class object
...	additional parameters

getExperimentName *generic function to extract* experimentName-*slot*

Description

generic function to extract experimentName-slot

Usage

```
getExperimentName(object, ...)

## S4 method for signature 'SpectraInTime'
getExperimentName(object)
```

Arguments

object	a S4 class object
...	additional parameters

getExtraInfo *generic function to extract* extraInfo-*slot*

Description

generic function to extract extraInfo-slot

Usage

```
getExtraInfo(object, ...)

## S4 method for signature 'SpectraInTime'
getExtraInfo(object)
```

Arguments

object	a S4 class object
...	additional parameters

getListOfSpectraExample *get example list of spectra*

Description

get example list of spectra

Usage

```
getListOfSpectraExample()
```

`getNMFInputMatrix` *Extract spectral input matrix from SPC file and condition properly for NMF*

Description

Extract spectral input matrix from SPC file and condition properly for NMF

Usage

```
getNMFInputMatrix(object, method = "")
```

Arguments

`object`	object of the 'spectralData' class, such as a raw SPC file
`method`	name of the NMF method to be used.

Value

spectral matrix, with wavelengths as its rows and time points as its columns

Author(s)

Nicolas Sauwen

`getPathProcessTimesExample` *example path process times ecport*

Description

example path process times ecport

Usage

```
getPathProcessTimesExample()
```

getPreprocessing *generic function to extract* preprocessing*-slot*

Description

generic function to extract preprocessing-slot

Usage

```
getPreprocessing(object, ...)

## S4 method for signature 'SpectraInTime'
getPreprocessing(object)
```

Arguments

object	a S4 class object
...	additional parameters

getProcessTimesExample *get a minimal* ProcessTimes-class *example based on* getSpectraInTimeExample

Description

get a minimal ProcessTimes-class example based on getSpectraInTimeExample

Usage

```
getProcessTimesExample()
```

Author(s)

Adriaan Blommaert

Examples

```
getProcessTimesExample()
```

getProcessTimesFrameExample *get mimimal example* ProcessTimesFrame-class

Description

get mimimal example ProcessTimesFrame-class

Usage

```
getProcessTimesFrameExample()
```

Author(s)

Adriaan Blommaert

getRange *generic function to extract* range-*slot*

Description

generic function to extract range-slot

Usage

```
getRange(object, ...)

## S4 method for signature 'RangeToSubset'
getRange(object)
```

Arguments

object	a S4 class object
...	additional parameters

getSpectra *generic function to extract* spectra*-slot*

Description

generic function to extract spectra-slot

Usage

```
getSpectra(object, ...)

## S4 method for signature 'SpectraInTime'
getSpectra(object)

## S4 method for signature 'SpectraInTime'
getSpectra(object)
```

Arguments

object	a S4 class object
...	additional parameters

getSpectraInTimeExample *Artificial example* SpectraInTime-class

Description

exponential conversion from 2 concentrations with gaussion curves for spectra at different wavelength per compounds

Usage

```
getSpectraInTimeExample(showPlots = FALSE)
```

Arguments

showPlots	logical indicator to show plots

Author(s)

Adriaan Blommaert

Examples

```
ex1  <-   getSpectraInTimeExample()
ex2  <-  getSpectraInTimeExample( showPlots = TRUE )
```

getStartTime *generic function to extract* startTime-*slot*

Description

generic function to extract startTime-slot

Usage

```
getStartTime(object, ...)

## S4 method for signature 'SpectraInTime'
getStartTime(object)
```

Arguments

object	a S4 class object
...	additional parameters

getTimePoints *generic function to extract* timePoints-*slot*

Description

generic function to extract timePoints-slot

Usage

```
getTimePoints(object, ...)

## S4 method for signature 'SpectraInTime'
getTimePoints(object, timePointsAlt = FALSE,
  timeUnit = "seconds")
```

Arguments

object	a S4 class object
...	additional parameters
timePointsAlt	logical indicator to get alternative (shifted) instead of recorded time points, defaults to FALSE
timeUnit	unit to use , choose between: seconds , minutes or hours, defaults equal to seconds

Examples

```
spectra    <-  getSpectraInTimeExample()
getTimePoints( spectra )
getTimePoints( spectra , timePointsAlt = TRUE  )
getTimePoints( spectra , timeUnit = "hours"  )
```

getUnits *generic function to extract* units-*slot*

Description

generic function to extract units-slot

Usage

```
getUnits(object, ...)

## S4 method for signature 'SpectraInTime'
getUnits(object)
```

Arguments

object	a S4 class object
...	additional parameters

getWavelengths *generic function to extract* wavelengths-*slot*

Description

generic function to extract wavelengths-slot

Usage

```
getWavelengths(object, ...)

## S4 method for signature 'SpectraInTime'
getWavelengths(object)
```

Arguments

object	a S4 class object
...	additional parameters

includeRedundantSources *Re-introduce redundant source vectors and corresponding zero abundances into final NMF result*

Description

Re-introduce redundant source vectors and corresponding zero abundances into final NMF result

Usage

```
includeRedundantSources(NMFResult, seed_orig, redundantSources)
```

Arguments

NMFResult	Fitted NMF model
seed_orig	Initial NMF model
redundantSources	boolean vector, obtained from checkForRedundantSources

Value

Final NMF model with redundant sources re-introduced

Author(s)

Nicolas Sauwen

initializeNMFModel *Initialize NMF model with initial spectral data*

Description

Initialize NMF model with initial spectral data

Usage

```
initializeNMFModel(initSpectralData, spectra, wavelengths = NULL)
```

Arguments

initSpectralData	this can be a list of spectralData objects, containing the pure component spectra. It can also be either of the NMF factor matrices with initial values
spectra	spectral matrix, with wavelengths as its rows and time points as its columns
wavelengths	vector of wavelength values

lastSpectrum *Get the last spectrum*

Description

Get the last spectrum

Usage

```
lastSpectrum(object, ...)

## S4 method for signature 'numeric'
lastSpectrum(object)

## S4 method for signature 'SpectraInTime'
lastSpectrum(object)
```

Arguments

object	S4 object
...	additional parameters

loadAllSPCFiles *Load all or a selection of SPC files from a given directory.*

Description

This function automatically recognizes all the files bearing an '.spc' extension and returns a list in which each element corresponds to a different xml file.

Usage

```
loadAllSPCFiles(directoryFiles, selectedFiles = NULL)
```

Arguments

directoryFiles	Character vector indicating the directory from which the files needs to be downloaded. Note that files with an other extension than '.spc' can be stored in this directory.
selectedFiles	Character vector listing which files of the chosen directory (as expressed by the 'directoryFiles' argument) should be processed. This argument is used when one wants to process a subset of the spc files of the selected directory only. Note that one should add the complete file name to this list, including the file extension! This is an optional argument with as default value NULL, meaning that by default all files of the selected directory are considered.

Value

A list is returned of which each element contains a processed SPC file

`localBaselineCorrect` *local baseline correct, substract a baseline either trough 1 or 2 points*

Description

local baseline correct, substract a baseline either trough 1 or 2 points

Usage

```
localBaselineCorrect(object, baseWavelengths = NULL)
```

Arguments

`object`	`SpectraInTime-class`
`baseWavelengths`	numeric vector of 1 or 2 wavelength use to draw a baseline trough, defaults to `NULL` when no baseline correction is performed

Value

`SpectraInTime-class` with baseline subset

Author(s)

Adriaan Blommaert

Examples

```
spectra             <-  getSpectraInTimeExample()
spectraConstCorrect  <-  localBaselineCorrect( spectra , baseWavelengths = 240  )
spectraLinCorrect    <-  localBaselineCorrect( spectra , c( 250 , 330 )  )
## Not run:
plot( spectra )
plot( spectraConstCorrect )
plot( spectraLinCorrect )

## End(Not run)
```

`nonNegativePreprocessing` *condition datamatrix to input in and condition properly for NMF*

Description

condition datamatrix to input in and condition properly for NMF

Usage

```
nonNegativePreprocessing(spectra, method = "")
```

Arguments

spectra	matrix of spectra
method	name of the NMF method to be used.

Details

put negative values to zero, transpose, an add small value zero row (wavelength with only zeros)

Value

matrix, with wavelengths as its rows and time points as its columns

---

normalize *generic normalization function*

Description

generic normalization function

Usage

```
normalize(object, ...)

## S4 method for signature 'SpectraInTime'
normalize(object, method = "normalize",
  wavelengthRange = r(-Inf, Inf), wavelength = NULL, scaleFunction = "sd",
  meanFunction = NULL)
```

Arguments

object	a S4 class object
...	additional parameters
method	a method for normalization or peak correction , choose from: * normalize substract mean and divide by scale * peak scale by reference wavelength * integrate scale by integrating over wavelengthRange
wavelengthRange	range for integration if method = integration , defaults to complete range
wavelength	reference wavelength for peak regresssion
scaleFunction	scale function used when method = normalize defaults to sd
meanFunction	mean function used when method = normalize defaults to mean

Examples

```
spectralEx            <-  getSpectraInTimeExample()
timeRange                       <-  range( getTimePoints( spectralEx ))
timesToSelect              <- e( seq( timeRange[1] , timeRange[2] , length.out = 5 )  )
## Not run:
plot( spectralEx )
plot( spectralEx[ timesToSelect ,  ] , type = "time" )
```

```
## End(Not run)
 normalizePeak         <-  normalize( spectralEx , method = "peak" , wavelength = 400 )
 getPreprocessing( normalizePeak )
## Not run:
 plot( normalizePeak[ timesToSelect ,  ] , type = "time" )
 plot( normalizePeak )

## End(Not run)
 normalizeIntegration  <-  normalize( spectralEx , method = "integration" )
## Not run:
 plot( normalizeIntegration[ timesToSelect ,  ] , type = "time" )

## End(Not run)
 normalizedUser <- normalize( spectralEx , method = "normalize" , mean = "median" , scale = "sd" )
## Not run:
 plot( normalizedUser[ timesToSelect ,  ] , type = "time" )

## End(Not run)
```

`predictNNLS` *Based on previously obtained NMF result* `NMFResult`*, estimate coefficients for a new spectralData object* `object` *using non-negative least squares fitting. The result is returned as as an NMF model.*

Description

Based on previously obtained NMF result `NMFResult`, estimate coefficients for a new spectralData object `object` using non-negative least squares fitting. The result is returned as as an NMF model.

Usage

```
predictNNLS(object, NMFResult)
```

Arguments

`object`	`SpectraInTime-class`
`NMFResult`	Fitted NMF model

Value

Fitted non-negative least squares result in the form of an NMF model

Author(s)

nsauwen

preprocess *generic function to preprocess an S4 object*

Description

generic function to preprocess an S4 object

Usage

```
preprocess(object, with)

## S4 method for signature 'SpectraInTime,list'
preprocess(object, with)

## S4 method for signature 'SpectraInTime,SpectraInTime'
preprocess(object, with)
```

Arguments

object	a S4 class object
with	an other object containing preprocessing information: other S4 object, list or expression

Examples

```
object1  <-  getSpectraInTimeExample()
object2  <-  getSpectraInTimeExample
```

ProcessTimes-class *S4 Class key process times*

Description

S4 Class key process times

Slots

`experimentName` character vector with name of the experiment

`timeHeatingAboveMin` time when experiment above minimum temperature

`timeStartReaction` time start reaction (end of heating ramp)

`timeEndProcess` time timeEndProcess time end of the process, when cooling down starts

`Tset` the maximum temperature to indicate `timeStartReaction`

`comments` character vector of comments when `NA` values are produced

Author(s)

Adriaan Blommaert

`ProcessTimesFrame-class` *S4 Class key process times in a data frame, every line is convertable to a* `ProcessTimes-class`

Description

S4 Class key process times in a data frame, every line is convertable to a `ProcessTimes-class`

Slots

`processTimes` data.frame with every line process times of an experiment

Author(s)

Adriaan Blommaert

`r` *create a* `RangeToSubset-class` *object from 2 elements or from a vector*

Description

create a `RangeToSubset-class` object from 2 elements or from a vector

Usage

```
r(x, y)

## S4 method for signature 'numeric,numeric'
r(x, y)

## S4 method for signature 'RangeToSubset,missing'
r(x, y)
```

Arguments

x	numeric value or vector of numeric values
y	numeric value missing when x is a vector of values

RangeToSubset-class *Range S4 class (range) useful for subsetting with actual values instead of indicators*

Description

Range S4 class (range) useful for subsetting with actual values instead of indicators

Slots

`range` numeric vector with min and max value

Author(s)

Adriaan Blommaert

readProcessTimes *read .csv file as process times*

Description

read .csv file as process times

Usage

```
readProcessTimes(path, timeFormat = "%Y-%m-%d %H:%M:%OS")
```

Arguments

`path`	to the file containing process times information
`timeFormat`	character specifying time format `as.POSIXct`

Value

`ProcessTimesFrame-class`

Examples

```
readProcessTimes( getPathProcessTimesExample() , timeFormat =  "%Y-%m-%d %H:%M:%S" )
```

readSPC *Read-in of a SPC file.*

Description

This function is an adaptation of of the 'read.spc' function of the 'hyperSpec' package : Claudia Beleites and Valter Sergo: 'hyperSpec: a package to handle hyperspectral data sets in R, R package version 0.98-20161118. http://hyperspec.r-forge.r-project.org.

Usage

```
readSPC(filename, keys.log2data = TRUE, keys.hdr2data = FALSE)
```

Arguments

filename	Character vector expressing the name of the SPC file (just the name, not the directory).
keys.log2data	Logical vector indicating whether the full information (consisting of additional information on the experimental conditions) needs to be parsed from the SPC file or not (TRUE indicates that the full information should be parsed from the SPC file). The default value is FALSE.
keys.hdr2data	a character vector of header object to add to backgroundInformation

Value

SpectraInTime-class

removeRedundantSources *Remove redundant sources from the initial NMF model*

Description

Remove redundant sources from the initial NMF model

Usage

```
removeRedundantSources(seed, redundantSources)
```

Arguments

seed	nmfModel object containing initialization of the factor matrices
redundantSources	boolean vector, obtained from checkForRedundantSources

Value

nmfModel object with redundant sources removed from initial factor matrices

Author(s)

Nicolas Sauwen

runNMF *Actual NMF analysis*

Description

Actual NMF analysis

Usage

```
runNMF(spectra, rank, method = "PGNMF", seed = NULL, nruns = 10,
  checkDivergence = TRUE, timePointsList = NULL, subsamplingFactor = 3)
```

Arguments

spectra	spectral input matrix, with wavelengths as its rows and time points as its columns
rank	number of NMF components to be found
method	name of the NMF method to be used, consult the help of the 'nmf' function from the NMF package for the methods available by default
seed	nmfModel object containing initialization of the factor matrices
nruns	number of NMF runs. It is recommended to run the NMF analyses multiple times when random seeding is used, to avoid a suboptimal solution
checkDivergence	Boolean indicating whether divergence checking should be performed, defaults to TRUE
timePointsList	list of time point vectors of the individual experiments
subsamplingFactor	subsampling factor used during NMF analysis

Value

Resulting NMF model (in accordance with the NMF package definition)

Author(s)

Nicolas Sauwen

saveSpectra *save a* SpectraInTime-class *as a* .txt *file*

Description

save a SpectraInTime-class as a .txt file

Usage

```
saveSpectra(object, directory, precision = 32)

readSpectra(file)
```

Arguments

object	object to save
directory	directory to save object
precision	number of significant digits controlling precission
file	to be read

Value

the path to which the file is saved

Note

experiment name is used to save the experiment

default time formats are assumed to convert to SpectraInTime-class

some data precession is lost because of internal conversion to JSON format

Author(s)

Adriaan Blommaert

Examples

```
 spectra         <-  getSpectraInTimeExample()
saveSpectra(  spectra , directory )
experimentName  <-  getExperimentName( spectra )
file            <-  file.path( directory , paste0( experimentName , ".txt")  )
spectraRead     <-  readSpectra( file )
```

---

scaleNMFResult — *Apply fixed scaling to NMF model matrices by normalizing the basis vectors*

Description

Apply fixed scaling to NMF model matrices by normalizing the basis vectors

Usage

```
scaleNMFResult(NMFResult)
```

Arguments

NMFResult	Fitted NMF model

Value

NMFResult Rescaled NMF model

Author(s)

Nicolas Sauwen

setExperimentName<- *set the experiment name*

Description

set the experiment name

Usage

```
setExperimentName(object) <- value

## S4 replacement method for signature 'SpectraInTime'
setExperimentName(object) <- value

## S4 replacement method for signature 'SpectraInTime'
setTimePointsAlt(object) <- value
```

Arguments

object	a S4 class object
value	a vector of time points

setTimePointsAlt<- *set time alternative time axis*

Description

set time alternative time axis

Usage

```
setTimePointsAlt(object) <- value
```

Arguments

object	a S4 class object
value	a vector of time points

smooth *generic smoothing function*

Description

generic smoothing function

smoothing is applied in the wavelength domain, not in the time domain

Usage

```
smooth(object, ...)

## S4 method for signature 'SpectraInTime'
smooth(object, method = "sg", order = 3,
  window = order + 3 - order%%2, derivative = 0)
```

Arguments

object	a S4 class object
...	additional parameters
method	character vector smoothing method, default = 'sg', i.e Savitsky-Golay filter. currently only implemented smoothing method
order	numeric value, order of the polynomial used to interpolate, should be larger than derivative order, defaults to 3 + derivative
window	width of the smoothing
derivative	derivative to be taken, defaults to 0

Note

equal distances between wavelenght intervals are assumed

Examples

```
 spectralEx     <-  getSpectraInTimeExample()
smoothDefault   <-  smooth( spectralEx )
timeRange       <-  range( getTimePoints( spectralEx ))
timesToSelect   <-  e( seq( timeRange[1] , timeRange[2] , length.out = 5  )  )
# plot( smoothDefault  )
# plot( smoothDefault[ timesToSelect , ] , type = "time")
smoothALot      <-  smooth( spectralEx ,  order = 2 , window = 301  )
# plot( smoothALot )
# plot( smoothALot[  timesToSelect , ] , type = "time" )
derivative1     <-  smooth( spectralEx , derivative = 1 )
# plot( derivative1 )
# plot( derivative1[  timesToSelect ,] , type = "time" )

derivative2     <-  smooth( spectralEx , derivative = 2 )
# plot(  derivative2 )
# plot( derivative2[  timesToSelect , ] , type = "time" )
```

spectralAnalysis *spectralAnalysis: a package to read-in, pre-process, visualise and analyse spectral data*

Description

spectralAnalysis: a package to read-in, pre-process, visualise and analyse spectral data

spectralIntegration *Integrate spectraInTime object*

Description

The integrated value over a user-specified wavelength range is calculated (trapezium rule) per time point, afterwards smoothing over time can be applied

Usage

```
spectralIntegration(object, wavelenghtRange, smoothingValue = 0,
  timeUnit = "seconds")
```

Arguments

object	SpectraInTime-class
wavelenghtRange	numeric vector of 2 elements i.e. integration limits
smoothingValue	numeric value between 0 and 1, amount of codelowess-smoothing, default to 0 i.e no smoothing. Note that smoothing is applied after integration
timeUnit	character value, choose between: second , minutes and hours, defaults to seconds

Value

data.frame with variables time and integratedValue

Examples

```
spectra                 <-  getSpectraInTimeExample()
defaults             <-  spectralIntegration( spectra , c(200 , 300) , timeUnit = "hours" )
unsmoothedTrend       <-  spectralIntegration( spectra , c(200 , 300) , timeUnit = "hours" )
smoothedTrend            <-  spectralIntegration( spectra , c(200 , 300) ,
  smoothingValue = 0.5 , timeUnit = "hours" )
```

spectralNMF *Perform Non-Negative Matrix factorization on spectral data*

Description

Perform Non-Negative Matrix factorization on spectral data

Usage

```
spectralNMF(object, rank, method = "PGNMF", initSpectralData = NULL,
  nruns = 10, subsamplingFactor = 3, checkDivergence = TRUE)
```

Arguments

object	SpectraInTime-class
rank	number of NMF components to be found
method	name of the NMF method to be used. "PGNMF" (default), "HALSacc" and "semiNMF" are methods derived from the hNMF package. All methods from the NMF package are also available.
initSpectralData	this can be a list of spectralData objects, containing the pure component spectra. It can also be either of the NMF factor matrices with initial values
nruns	number of NMF runs. It is recommended to run the NMF analyses multiple times when random seeding is used, to avoid a suboptimal solution
subsamplingFactor	subsampling factor used during NMF analysis
checkDivergence	Boolean indicating whether divergence checking should be performed

Value

Scaled NMF model (in accordance with the NMF package definition)

Author(s)

Nicolas Sauwen

spectralNMFList *Perform Non-Negative Matrix factorization on list of SPC files*

Description

Perform Non-Negative Matrix factorization on list of SPC files

Usage

```
spectralNMFList(objectList, rank, method = "PGNMF", initSpectralData = NULL,
  nruns = 10, subsamplingFactor = 3, checkDivergence = TRUE)
```

Arguments

objectList	list of SPC files
rank	number of NMF components to be found
method	name of the NMF method to be used, consult the help of the 'nmf' function from the NMF package for the methods available by default
initSpectralData	list of SPC files containing pure component spectra
nruns	number of NMF runs.
subsamplingFactor	subsampling factor used during NMF analysis
checkDivergence	Boolean indicating whether divergence checking should be performed

Value

list of NMF models

Author(s)

Nicolas Sauwen

---

subset-methods *Subsetting* SpectraInTime-class

Description

Subsetting SpectraInTime-class

Usage

```
## S4 method for signature 'SpectraInTime,ANY,ANY,ANY'
x[i, j, ..., drop = ""]

## S4 method for signature 'SpectraInTime,missing,ANY,ANY'
x[i, j, ..., drop = ""]

## S4 method for signature 'SpectraInTime,ANY,missing,ANY'
x[i, j, ..., drop = ""]

## S4 method for signature 'SpectraInTime,missing,missing,ANY'
x[i, j, ..., drop = ""]
```

Arguments

x	object to subset
i	subsetting rows (timePoints)
j	subsetting columns (wavelengths)
...	additional parameters

- timeUnit unit at which subsetting should be done choose between seconds , minutes or hours defaults to seconds
- timePointsAlt logical indicators whater alternative timePoints should be used

drop for consistancy, not used

Examples

```
### subsetting [ time , wavelength, options ]

spectralEx               <-  getSpectraInTimeExample()
spectraSubset            <-  spectralEx[ r( 1000 , 30000 ) , r(130 , 135 ) ]
spectraSubsetTime        <-  spectralEx[ r( 1000 , 30000 ) ,  ]
spectraSubsetWavelengths  <-  spectralEx[  ,  r(130 , 135 ) ]
spectraSubsetHours       <-  spectralEx[ r( 1 , 3 ) , r(130 , 135 ) , timeUnit = "hours" ]
closestWavelengths     <- spectralEx[ , e( 150, 4, 300, 500 ) ] # remark only unique values
spectraSubsetLogical      <-  spectralEx[ getTimePoints( spectralEx ) > 300   ,
   getWavelengths( spectralEx ) <= 500 ]
```

---

timeAlign *Time align first object, using info in the second object*

Description

Time align first object, using info in the second object

Usage

```
timeAlign(x, y, ...)

## S4 method for signature 'SpectraInTime,ProcessTimes'
timeAlign(x, y, cutCooling = FALSE,
  cutBeforeMinTemp = FALSE)

## S4 method for signature 'list,ProcessTimesFrame'
timeAlign(x, y, cutCooling = FALSE,
  cutBeforeMinTemp = FALSE)

## S4 method for signature 'list,character'
timeAlign(x, y, cutCooling = FALSE,
  cutBeforeMinTemp = FALSE, timeFormat = "%Y-%m-%d %H:%M:%S")
```

Arguments

x	and S4 object to be aligned
y	object to use time information from
...	additional arguments
cutCooling	logical indicator if TRUE observation after cooling starts are cut off, defaults to FALSE
cutBeforeMinTemp	logical indicator if TRUE observation before minimum temperature are cut off, defaults to FALSE
timeFormat	character vector specifying time format as.POSIXct

Examples

```
 spectra            <-  getSpectraInTimeExample()
 listOfSpectra      <-  getListOfSpectraExample()
 processTimes       <-  getProcessTimesExample()
 processTimesFrame  <-  getProcessTimesFrameExample()
 pathProcessTimes   <-  getPathProcessTimesExample()

ex1  <-  timeAlign( x = spectra , y = processTimes ,
 cutCooling = TRUE , cutBeforeMinTemp = TRUE )
ex2  <-  timeAlign( x = listOfSpectra , y = processTimesFrame ,
 cutCooling = TRUE , cutBeforeMinTemp = TRUE )
ex3  <-  timeAlign( x = listOfSpectra , y = pathProcessTimes,
 cutCooling = TRUE , cutBeforeMinTemp = TRUE  , timeFormat =  "%Y-%m-%d %H:%M:%OS" )
```

`upsampleNMFResult` *Upsample NMF result to original temporal resolution*

Description

Upsample NMF result to original temporal resolution

Usage

```
upsampleNMFResult(NMFResult, timePoints, subsamplingFactor, shift = 0)
```

Arguments

`NMFResult`	Fitted NMF model
`timePoints`	Original time points
`subsamplingFactor`	Subsampling factor
`shift`	Integer that correctly shifts subsampling index when applying NMF to multiple experiments

Value

Upsampled NMF model

Author(s)

Nicolas Sauwen

wavelengthAlign *Align SpectraInTime objects with differing wavelength axes to a common wavelength axis using cubic spline interpolation.*

Description

Align SpectraInTime objects with differing wavelength axes to a common wavelength axis using cubic spline interpolation.

Usage

```
wavelengthAlign(ref, toAlign)

## S4 method for signature 'SpectraInTime,SpectraInTime'
wavelengthAlign(ref, toAlign)

## S4 method for signature 'SpectraInTime,list'
wavelengthAlign(ref, toAlign)
```

Arguments

ref	SpectraInTime-class object with the reference wavelength vector
toAlign	SpectraInTime-class object(s) to be aligned. This can either be a single SpectraInTime object or a list of SpectraInTime objects. In case of a list, all objects in the list should have the same wavelength axis.

Value

List of aligned SpectraInTime objects, including the reference object.

Index