

# Package ‘spectralAnalysis’

January 11, 2023

**Title** Pre-Process, Visualize and Analyse Spectral Data

**Version** 4.3.1

**Maintainer** Adriaan Blommaert <adriaan.blommaert@openanalytics.eu>

**URL** <https://openanalytics.eu>

**Description** Infrared, near-infrared and Raman spectroscopic data measured during chemical reactions, provide structural fingerprints by which molecules can be identified and quantified. The application of these spectroscopic techniques as inline process analytical tools (PAT), provides the pharmaceutical and chemical industry with novel tools, allowing to monitor their chemical processes, resulting in a better process understanding through insight in reaction rates, mechanistics, stability, etc.  
Data can be read into R via the generic spc-format, which is generally supported by spectrometer vendor software. Versatile pre-processing functions are available to perform baseline correction by linking to the 'baseline' package; noise reduction via the 'signal' package; as well as time alignment, normalization, differentiation, integration and interpolation. Implementation based on the S4 object system allows storing a pre-processing pipeline as part of a spectral data object, and easily transferring it to other datasets. Interactive plotting tools are provided based on the 'plotly' package. Non-negative matrix factorization (NMF) has been implemented to perform multivariate analyses on individual spectral datasets or on multiple datasets at once. NMF provides a parts-based representation of the spectral data in terms of spectral signatures of the chemical compounds and their relative proportions.  
See 'hNMF'-package for references on available methods. The functionality to read in spc-files was adapted from the 'hyperSpec' package.

**License** GPL-3

**biocViews**

**Imports** baseline, BiocGenerics, data.table, ggplot2, graphics, jsonlite, magrittr, methods, nnls, NMF, plotly, plyr, dplyr, RColorBrewer, signal, stats, viridis, hNMF, zoo, pls

**RoxygenNote** 7.1.2

**Suggests** testthat, knitr, rmarkdown, webshot, bookdown

**Collate** 'internalHelpers.R' 'allGenericFunctions.R' 'objectSpectraInTime.R' 'objectProcessTimes.R'

'objectLinking.R' 'alignmentFunctions.R'  
 'combineSpectralObjects.R' 'dataManagementTools.R' 'defaults.R'  
 'objectSpectraInTimeComp.R' 'readSPC.R' 'saveSpectraInTime.R'  
 'spectralIntegration.R' 'spectralNMF.R' 'spectralPLS.R'  
 'spectralPreprocessing.R' 'spectralVisualization.R'  
 'subsetting.R'

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Robin Van Oirbeek [aut],  
 Adriaan Blommaert [aut, cre],  
 Nicolas Sauwen [aut],  
 Tor Maes [ctb],  
 Jan Dijkmans [ctb],  
 Jef Cuypers [ctb],  
 Tatsiana Khamiakova [ctb],  
 Michel Thiel [ctb],  
 Claudia Beleites [ctb]

**Repository** CRAN

**Date/Publication** 2023-01-11 10:00:02 UTC

## R topics documented:

baselineCorrect . . . . .	3
checkCompatible . . . . .	5
checkForRedundantSources . . . . .	5
checkIdenticalClass . . . . .	6
combineSpectralObjects . . . . .	6
computeNMFResidu . . . . .	7
e . . . . .	8
ElementsToSelect-class . . . . .	8
firstSpectrum . . . . .	9
getDefaultSumFunc . . . . .	9
getDefaultTimeFormat . . . . .	10
getDimensionReduction . . . . .	10
getElements . . . . .	11
getExperimentName . . . . .	11
getExtraInfo . . . . .	12
getListOfSpectraExample . . . . .	12
getNMFInputMatrix . . . . .	13
getPathProcessTimesExample . . . . .	13
getPreprocessing . . . . .	14
getProcessTimesExample . . . . .	14
getProcessTimesFrameExample . . . . .	15
getRange . . . . .	15
getSpectra . . . . .	16
getSpectraInTimeCompExample . . . . .	16

getSpectraInTimeExample . . . . .	17
getSpectralAxis . . . . .	17
getStartTime . . . . .	18
getTimePoints . . . . .	18
getUnits . . . . .	19
includeRedundantSources . . . . .	20
initializeNMFModel . . . . .	20
lastSpectrum . . . . .	21
loadAllSPCFiles . . . . .	21
localBaselineCorrect . . . . .	22
nonNegativePreprocessing . . . . .	23
normalize . . . . .	23
predictNNLS . . . . .	25
preprocess . . . . .	25
ProcessTimes-class . . . . .	26
ProcessTimesFrame-class . . . . .	27
r . . . . .	27
RangeToSubset-class . . . . .	28
readProcessTimes . . . . .	28
readSPC . . . . .	29
removeRedundantSources . . . . .	29
runNMF . . . . .	30
saveSpectra . . . . .	31
scaleNMFResult . . . . .	32
setExperimentName<- . . . . .	32
setTimePointsAlt<- . . . . .	33
smooth . . . . .	33
SpectraInTimeComp-class . . . . .	35
spectralIntegration . . . . .	36
spectralNMF . . . . .	37
spectralNMFList . . . . .	38
spectralPLSCalibration . . . . .	39
spectralPlsPrediction . . . . .	40
subset-methods . . . . .	40
timeAlign . . . . .	41
upsampleNMFResult . . . . .	43
wavelengthAlign . . . . .	43
<b>Index</b>	<b>45</b>

---

baselineCorrect	<i>generic function to perform baseline correction</i>
-----------------	--

---

## Description

generic function to perform baseline correction

**Usage**

```
baselineCorrect(object, ...)

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

## S4 method for signature 'SpectraInTimeComp'
baselineCorrect(object, ...)
```

**Arguments**

object	a S4 class object
...	other parameters passed to <a href="#">baseline</a>
method	method of baseline correction, default value is to 'modpolyfit', see <a href="#">baseline.modpolyfit</a>
degree	numeric value, degree of the polynomial used only if method is code 'modpolyfit'

**Value**

[SpectraInTime-class](#)

**Note**

baseline correction in the wavelength domain by linking to the [baseline](#)

**Examples**

```
spectralEx      <- getSpectraInTimeExample()
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( spectralEx )
plot( baselineDefault[ timesToSelect , ] , type = "time" )
plot( baselineHighPolynomial[ timesToSelect , ] , type = "time" )
plot( baselineLowpass[ timesToSelect , ] , type = "time" )
```

---

checkCompatible	<i>Check object compatibility</i>
-----------------	-----------------------------------

---

**Description**

Check wheter 2 objects are compatible before using them together e.g. in time alignment using a time file with matching experiment name.

**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

**Value**

no output, produces an error when object are not compatible with each other  
TRUE when the aer compatible, otherwise it stops and prints a list of error messages

---

checkForRedundantSources	<i>Check for redunt NMF source vectors</i>
--------------------------	--

---

**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

---

combineSpectralObjects  
*Function to combine SpectraInTime-class objects containing 1 spectrum each*

---

**Description**

Function to combine SpectraInTime-class objects containing 1 spectrum each

**Usage**

```
combineSpectralObjects(objectList, timeRange, checkNames = TRUE)
```

**Arguments**

- objectList      List of [SpectraInTime-class](#) objects to be combined
- timeRange      Numeric value, equal to the maximum time of the measured spectra.
- checkNames     Boolean - if TRUE, the experiment name of the spectral objects will be compared to see if these spectral objects belong to the same experiment

**Value**

[SpectraInTime-class](#)

**Author(s)**

Nicolas Sauwen

---

computeNMFResidu      *NMF relative residual per observation*

---

**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	<i>Get the first spectrum</i>
---------------	-------------------------------

---

**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

**Value**

numeric vector containing observations first spectrum

---

getDefaultSumFunc	<i>function to get default summary functions</i>
-------------------	--

---

**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 specifying a time format

character string iwth default time format

---

getDimensionReduction *generic function to extract dimensionReduction-slot*

---

**Description**

generic function to extract dimensionReduction-slot

**Usage**

```
getDimensionReduction(object, ...)
```

**Arguments**

object            a S4 class object

...               additional parameters

**Value**

dimension reduction slot of an object

---

getElements	<i>generic function to extract elements-slot</i>
-------------	--

---

**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	<i>generic function to extract experimentName-slot</i>
-------------------	--

---

**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

**Value**

string containing experiment name

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

**Value**

list of extraInfo

---

getListOfSpectraExample  
*get example list of spectra*

---

**Description**

get example list of spectra

**Usage**

```
getListOfSpectraExample()
```

**Value**

list of [SpectraInTime-class](#)

---

getNMFInputMatrix      *Get spectralData as input NMF model*

---

**Description**

Extract spectral input matrix from [SpectraInTime-class](#) and condition properly for NMF modeling

**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*

---

**Description**

example path process times

**Usage**

```
getPathProcessTimesExample()
```

**Value**

[ProcessTimes-class](#)

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

**Value**

list with preprocessing steps

---

getProcessTimesExample  
*get a minimal [ProcessTimes-class](#) example based on [getSpectraInTimeExample](#)*

---

**Description**

get a minimal [ProcessTimes-class](#) example based on [getSpectraInTimeExample](#)

**Usage**

```
getProcessTimesExample()
```

**Value**

[ProcessTimes-class](#)

**Author(s)**

Adriaan Blommaert

**Examples**

```
getProcessTimesExample()
```

---

getProcessTimesFrameExample  
*get minimal example* [ProcessTimesFrame-class](#)

---

**Description**

get minimal example [ProcessTimesFrame-class](#)

**Usage**

getProcessTimesFrameExample()

**Value**

[ProcessTimes-class](#)

**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

**Value**

matrix of spectra

---

getSpectraInTimeCompExample  
*Artificial example of [SpectraInTimeComp-class](#)*

---

**Description**

Example [SpectraInTime-class](#) with nmf result using random initialization with rank 2

**Usage**

```
getSpectraInTimeCompExample()
```

**Value**

[SpectraInTimeComp-class](#)

**Author(s)**

Adriaan Blommaert



## Examples

```
test <- getSpectraInTimeCompExample()
```

---

```
getSpectraInTimeExample
```

*Artificial example* [SpectraInTime-class](#)

---

## Description

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

## Usage

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

## Arguments

showPlots      logical indicator to show plots

## Value

[SpectraInTime-class](#)

## Author(s)

Adriaan Blommaert

## Examples

```
ex1 <- getSpectraInTimeExample()
```

---

```
getSpectralAxis
```

*generic function to extract spectralAxis-slot*

---

## Description

generic function to extract spectralAxis-slot

## Usage

```
getSpectralAxis(object, ...)
```

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

**Arguments**

object            a S4 class object  
...               additional parameters

**Value**

numeric vector containing wavelengths

---

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

**Value**

character vector with start time

---

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

**Value**

numeric vector containing timepoints

**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

**Value**

list of units

---

```
includeRedundantSources
```

*Re-introduce redundant sources in NMF-model*

---

### 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 <a href="#">checkForRedundantSources</a>

### 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, spectralAxis = 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
spectralAxis	vector of wavelength/spectralAxis values

**Value**

an object that inherits from the class [NMF](#)

---

lastSpectrum	<i>Get the last spectrum</i>
--------------	------------------------------

---

**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

**Value**

numeric vector containing values last spectrum

---

loadAllSPCFiles	<i>Load all or a selection of SPC files from a given directory.</i>
-----------------	---

---

**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 correction*

---

**Description**

Substract a baseline either through 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 ) )

plot( spectra )
plot( spectraConstCorrect )
plot( spectraLinCorrect )
```

---

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, and 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",
  spectralRange = r(-Inf, Inf),
  spectralAxisVal = NULL,
  scaleFunction = "sd",
  meanFunction = NULL
)

## S4 method for signature 'SpectraInTimeComp'
normalize(object, ...)
```

**Arguments**

object	a S4 class object
...	additional parameters
method	a method for normalization or peak correction , choose from: * normalize subtract mean and divide by scale * peak scale by reference spectralAxisVal * integration scale by integrating over spectralAxisRange
spectralRange	range for integration if method = integration , defaults to complete range
spectralAxisVal	reference spectral axis value (wavelength or other) for peak regression
scaleFunction	scale function used when method = normalize defaults to <a href="#">sd</a>
meanFunction	mean function used when method = normalize defaults to <a href="#">mean</a>

**Value**

[SpectraInTime-class](#)

**Examples**

```
spectralEx      <- getSpectraInTimeExample()
timeRange      <- range( getTimePoints( spectralEx ))
timesToSelect  <- e( seq( timeRange[1] , timeRange[2] , length.out = 5 ) )

plot( spectralEx )
plot( spectralEx[ timesToSelect , ] , type = "time" )

normalizePeak   <- normalize( spectralEx , method = "peak" , spectralAxisVal = 400 )
getPreprocessing( normalizePeak )

plot( normalizePeak[ timesToSelect , ] , type = "time" )
plot( normalizePeak )
```



```

normalizeIntegration <- normalize( spectralEx , method = "integration" )
plot( normalizeIntegration[ timesToSelect , ] , type = "time" )

normalizedUser <- normalize( spectralEx , method = "normalize" , mean = "median" , scale = "sd" )
plot( normalizedUser[ timesToSelect , ] , type = "time" )

```

---

predictNNLS	<i>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.</i>
-------------	---

---

### 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	<a href="#">SpectraInTime-class</a>
NMFResult	Fitted NMF model

### Value

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

### Author(s)

nsauwen

---

preprocess	<i>generic function to preprocess an S4 object</i>
------------	--

---

### 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)

## S4 method for signature 'SpectraInTimeComp,ANY'
preprocess(object, with)

```

**Arguments**

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

**Value**

[SpectraInTime-class](#)

---

ProcessTimes-class	<i>S4 Class key process times</i>
--------------------	-----------------------------------

---

**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  
*ProcessTimes-class*

---

**Description**

S4 Class key process times in a data frame, every line is convertible to a [ProcessTimes-class](#)

**Value**

[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

**Value**

[RangeToSubset-class](#)

---

RangeToSubset-class     *RangeToSubset-class*

---

**Description**

Allows for subsetting a range of actual values instead of a range 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](#)  
[ProcessTimes-class](#)

**Examples**

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

---

readSPC	<i>Read-in of a SPC file.</i>
---------	-------------------------------

---

### Description

This function is an adaptation of the 'read.spc' function of the 'hyperSpec' package : Claudia Beleites and Valter Sergio: '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	<i>Remove redundant sources from the initial NMF model</i>
------------------------	--

---

### 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 <a href="#">checkForRedundantSources</a>

**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,
  maxIter = 1000
)
```

**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
maxIter	maximum number of iterations per NMF run

**Value**

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

**Author(s)**

Nicolas Sauwen

---

saveSpectra                      *read or save a [SpectraInTime-class](#) from or to a .txt file*

---

**Description**

read or save a [SpectraInTime-class](#) from or to 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 precision
file	to be read

**Value**

the path to which the file is saved

[SpectraInTime-class](#)

**Note**

experiment name is used to save the experiment

default time formats are assumed to convert to [SpectraInTime-class](#)

some data precision 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*

---

**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

**Value**

[SpectraInTime-class](#) with modified experiment name

---

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

**Value**

[SpectraInTime-class](#) with modified timePointsAlt axis

---

smooth                *generic smoothing function*

---

**Description**

smoothing is applied along the spectral axis, not the time axis

**Usage**

```
smooth(object, ...)

## S4 method for signature 'SpectraInTime'
smooth(
  object,
  method = "sg",
  order = 3,
  window = order + 7 - order%%2,
  derivative = 0,
  dim = "spectralAxis"
)

## S4 method for signature 'SpectraInTimeComp'
smooth(object, ...)
```

**Arguments**

object	a S4 class object
...	additional parameters
method	character vector smoothing method, options are 'sg' (= default, Savitsky-Golay filter) or 'mean'.
order	numeric value, order of the polynomial used to interpolate (only used when method = 'sg'), should be larger than derivative order, defaults to 3 + derivative
window	width of the smoothing default value slightly higher than in the signal package, the user might consider a large value, otherwise smoothing has little effect
derivative	derivative to be taken (only used when method = 'sg'), defaults to 0
dim	character string, specifying along which dimension smoothing should be applied. Options are "spectralAxis" (= default) or "time"

**Value**

[SpectraInTime-class](#)

**Note**

equal distances between wavelength intervals are assumed

**Examples**

```
spectralEx <- getSpectraInTimeExample()
smoothDefault <- smooth( spectralEx )
timeRange <- range( getTimePoints( spectralEx ))
timesToSelect <- e( seq( timeRange[1] , timeRange[2] , length.out = 5 ) )
smoothALot <- smooth( spectralEx , order = 2 , window = 301 )
derivative1 <- smooth( spectralEx , derivative = 1 )
derivative2 <- smooth( spectralEx , derivative = 2 )
```

---

 SpectraInTimeComp-class

*SpectraInTimeComp-class (time resolved spectra)*


---

### Description

Spectral-time data for 1 experiment with dimension reduction technique NMF and/or PCA decomposition included

### Usage

```
## S4 method for signature 'SpectraInTimeComp'
getDimensionReduction(object, type = NULL)
```

### Arguments

object            of class SpectraInTimeComp-class  
 type             type of regression method specified, if NULL the entire slot is returned as a list

### Slots

dimensionReduction list containing dimension reduction technique, either PCA or NMF, but only one per kind.

### Author(s)

Adriaan Blommaert

### Examples

```
# generate example
exampleSpectra <- getSpectraInTimeCompExample()

# methods
PCAResult <- getDimensionReduction( exampleSpectra, type = "PCA" )
NMFResult <- getDimensionReduction( exampleSpectra, type = "NMF" )

dimensionReductions <- getDimensionReduction( exampleSpectra )
str(dimensionReductions )

# subsetting works by reducing to \link{SpectraInTime-class}
subsetting <- exampleSpectra[1:3 , r(400, 450)]
# preprocessing methods also reduce the object to \link{SpectraInTime-class}
```

---

spectralIntegration    *Integrate spectralInTime object*

---

### Description

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

### Usage

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

### Arguments

object	<a href="#">SpectraInTime-class</a>
spectralRange	numeric vector of 2 elements i.e. integration limits
smoothingValue	numeric value between 0 and 1, amount of <a href="#">codelowess</a> -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 = 1,
  checkDivergence = TRUE,
  maxIter = 1000,
  includeRefs = FALSE
)
```

### Arguments

object	<a href="#">SpectraInTime-class</a>
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
maxIter	maximum number of iterations per NMF run
includeRefs	boolean, indicating whether references should be included in the input matrix for the NMF analysis

### Value

[SpectraInTimeComp-class](#) which includes a scaled NMF model (in accordance with the NMF package definition)

[SpectraInTimeComp-class](#)

**Author(s)**

Nicolas Sauwen

**Examples**

```

spectralExample <- getSpectraInTimeExample()
nmfResult      <- spectralNMF( spectralExample , rank = 2 , subsamplingFactor = 5 )
nmfObject      <- getDimensionReduction( nmfResult , type = "NMF" )$NMF
nmfTrends      <- t( NMF::coef( nmfObject ) )
matplot( nmfTrends , type = "l" , x = getTimePoints( spectralExample , timeUnit = "hours" ) ,
xlab = "time in hours" )

```

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,
  maxIter = 1000
)

```

**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
maxIter	maximum number of iterations per NMF run

**Value**

list of [SpectraInTimeComp-class](#)

**Author(s)**

Nicolas Sauwen

**Examples**

```
spectralData <- getListOfSpectraExample()  
spectraWithNmf <- spectralNMFList( spectralData , rank = 2 )
```

---

spectralPLSCalibration

*Compute PLS model*

---

**Description**

Compute PLS model

**Usage**

```
spectralPLSCalibration(objectList, UPLC_DF, ncomp = 10)
```

**Arguments**

objectList	list of SPC files
UPLC_DF	dataframe with UPLC data, which should contain the following columns: experiment, time, and 1 column per compound
ncomp	number of PLS components, defaults to 10

**Value**

PLS model, as obtained from [plsr](#)

**Author(s)**

Nicolas Sauwen

---

spectralPlsPrediction *Perform PLS prediction*

---

**Description**

Perform PLS prediction

**Usage**

```
spectralPlsPrediction(spectralObject, plsModel, nComp)
```

**Arguments**

spectralObject [SpectraInTime-class](#)  
plsModel PLS model as obtained from [spectralPLSCalibration](#)  
nComp Number of components

**Value**

[SpectraInTimeComp-class](#) which includes PLS model + prediction

**Author(s)**

Nicolas Sauwen

---

subset-methods *Subsetting [SpectraInTime-class](#)*

---

**Description**

Subsetting [SpectraInTime-class](#)

**Usage**

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



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

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

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

### Arguments

x	object to subset
i	subsetting rows ( timePoints )
j	subsetting columns ( spectral axis )
...	additional parameters <ul style="list-style-type: none"> <li>timeUnit unit at which subsetting should be done choose between seconds , minutes or hours defaults to seconds</li> <li>timePointsAlt logical indicators whater alternative timePoints should be used</li> </ul>
drop	for consistancy, not used

### Value

[SpectraInTime-class](#)

### Examples

```
### subsetting [ time , spectral axis, options ]

spectralEx          <- getSpectraInTimeExample()
spectraSubset       <- spectralEx[ r( 1000 , 30000 ) , r(130 , 135 ) ]
spectraSubsetTime   <- spectralEx[ r( 1000 , 30000 ) , ]
spectraSubsetSpectralVals <- spectralEx[ , r(130 , 135 ) ]
spectraSubsetHours   <- spectralEx[ r( 1 , 3 ) , r(130 , 135 ) , timeUnit = "hours" ]
closestSpectralVals  <- spectralEx[ , e( 150, 4, 300, 500 ) ] # remark only unique values
spectraSubsetLogical <- spectralEx[ getTimePoints( spectralEx ) > 300 ,
getSpectralAxis( 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 <a href="#">as.POSIXct</a>

**Value**

[SpectraInTime-class](#) or list of spectra depending on input

**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      *Wavelength align spectral data*

---

**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.  
one or a list of [SpectraInTime-class](#)

**Examples**

```
spectra          <- getSpectraInTimeExample()
listOfSpectra    <- getListOfSpectraExample()

# Dummy alignment of spectrum with itself:
ex1              <- wavelengthAlign( ref = spectra , toAlign = spectra )
# Alignment of list of spectra with a reference spectrum:
ex2              <- wavelengthAlign( ref = spectra , toAlign = listOfSpectra )
```

# Index

- [, SpectraInTime, ANY, ANY, ANY-method (subset-methods), 40
- [, SpectraInTime, ANY, ANY-method (subset-methods), 40
- [, SpectraInTime, ANY, missing-method (subset-methods), 40
- [, SpectraInTime, missing, ANY-method (subset-methods), 40
- [, SpectraInTime, missing, missing-method (subset-methods), 40
- [, SpectraInTime-method (subset-methods), 40
- [, SpectraInTimeComp, ANY, ANY-method (subset-methods), 40
- [, SpectraInTimeComp, ANY, missing-method (subset-methods), 40
- [, SpectraInTimeComp, missing, ANY-method (subset-methods), 40
- [ProcessTimes, SpectraInTime-method (checkCompatible), 5
- [SpectraInTime, ProcessTimes-method (checkCompatible), 5
  
- as.POSIXct, 28, 42
  
- baseline, 4
- baseline.modpolyfit, 4
- baselineCorrect, 3
- baselineCorrect, SpectraInTime-method (baselineCorrect), 3
- baselineCorrect, SpectraInTimeComp-method (baselineCorrect), 3
  
- checkCompatible, 5
- checkCompatible, ProcessTimes, SpectraInTime-method (checkCompatible), 5
- checkCompatible, SpectraInTime, ProcessTimes-method (checkCompatible), 5
- checkForRedundantSources, 5, 20, 29
- checkIdenticalClass, 6
  
- combineSpectralObjects, 6
- computeNMFResidu, 7
  
- e, 8
- ElementsToSelect-class, 8, 8
  
- firstSpectrum, 9
- firstSpectrum, numeric-method (firstSpectrum), 9
- firstSpectrum, SpectraInTime-method (firstSpectrum), 9
  
- getDefaultSumFunc, 9
- getDefaultTimeFormat, 10
- getDimensionReduction, 10
- getDimensionReduction, SpectraInTimeComp-method (SpectraInTimeComp-class), 35
- getElements, 11
- getElements, ElementsToSelect-method (getElements), 11
- getExperimentName, 11
- getExperimentName, SpectraInTime-method (getExperimentName), 11
- getExtraInfo, 12
- getExtraInfo, SpectraInTime-method (getExtraInfo), 12
- getListOfSpectraExample, 12
- getNMFInputMatrix, 13
- getPathProcessTimesExample, 13
- getPreprocessing, 14
- getPreprocessing, SpectraInTime-method (getPreprocessing), 14
- getProcessTimesExample, 14
- getProcessTimesFrameExample, 15
- getRange, 15
- getRange, RangeToSubset-method (getRange), 15
- getSpectra, 16
- getSpectra, SpectraInTime-method (getSpectra), 16

- getSpectraInTimeCompExample, 16
- getSpectraInTimeExample, 14, 17
- getSpectralAxis, 17
- getSpectralAxis, SpectraInTime-method  
(getSpectralAxis), 17
- getStartTime, 18
- getStartTime, SpectraInTime-method  
(getStartTime), 18
- getTimePoints, 18
- getTimePoints, SpectraInTime-method  
(getTimePoints), 18
- getUnits, 19
- getUnits, SpectraInTime-method  
(getUnits), 19
  
- includeRedundantSources, 20
- initializeNMFModel, 20
  
- lastSpectrum, 21
- lastSpectrum, numeric-method  
(lastSpectrum), 21
- lastSpectrum, SpectraInTime-method  
(lastSpectrum), 21
- loadAllSPCFiles, 21
- localBaselineCorrect, 22
- lowess, 36
  
- mean, 24
  
- NMF, 21
- nonNegativePreprocessing, 23
- normalize, 23
- normalize, SpectraInTime-method  
(normalize), 23
- normalize, SpectraInTimeComp-method  
(normalize), 23
  
- plsr, 39
- predictNNLS, 25
- preprocess, 25
- preprocess, SpectraInTime, list-method  
(preprocess), 25
- preprocess, SpectraInTime, SpectraInTime-method  
(preprocess), 25
- preprocess, SpectraInTimeComp, ANY-method  
(preprocess), 25
- ProcessTimes (ProcessTimes-class), 26
- ProcessTimes-class, 14, 26
- ProcessTimesFrame-class, 15, 27
  
- r, 27
- r, numeric, numeric-method (r), 27
- r, RangeToSubset, missing-method (r), 27
- RangeToSubset (RangeToSubset-class), 28
- Rangetosubset (RangeToSubset-class), 28
- rangetosubset (RangeToSubset-class), 28
- RangeToSubset-class, 27, 28
- read (saveSpectra), 31
- readProcessTimes, 28
- readSPC, 29
- readSpectra (saveSpectra), 31
- removeRedundantSources, 29
- runNMF, 30
  
- save (saveSpectra), 31
- saveSpectra, 31
- scaleNMFResult, 32
- sd, 24
- setExperimentName<-, 32
- setExperimentName<-, SpectraInTime-method  
(setExperimentName<-), 32
- setTimePointsAlt<-, 33
- setTimePointsAlt<-, SpectraInTime-method  
(setExperimentName<-), 32
- smooth, 33
- smooth, SpectraInTime-method (smooth), 33
- smooth, SpectraInTimeComp-method  
(smooth), 33
- SpectraInTime-class, 6, 17, 31, 40
- SpectraInTimeComp  
(SpectraInTimeComp-class), 35
- spectraInTimeComp  
(SpectraInTimeComp-class), 35
- spectraintimecomp  
(SpectraInTimeComp-class), 35
- SpectraInTimeComp-class, 16, 35
- spectralIntegration, 36
- spectralNMF, 37
- spectralNMFList, 38
- spectralPLSCalibration, 39, 40
- spectralPlsPrediction, 40
- subset-methods, 40
- TemperatureInfo (ProcessTimes-class), 26
- temperatureInfo (ProcessTimes-class), 26
- temperatureinfo (ProcessTimes-class), 26
- timeAlign, 41
- timeAlign, list, character-method  
(timeAlign), 41

timeAlign,list,ProcessTimesFrame-method  
    (timeAlign), [41](#)

timeAlign,SpectraInTime,ProcessTimes-method  
    (timeAlign), [41](#)

upsampleNMFRresult, [43](#)

wavelengthAlign, [43](#)

wavelengthAlign,SpectraInTime,list-method  
    (wavelengthAlign), [43](#)

wavelengthAlign,SpectraInTime,SpectraInTime-method  
    (wavelengthAlign), [43](#)