Package 'spectacles'

May 9, 2026

Type Package

Title Storing, Manipulating and Analysis Spectroscopy and Associated Data

Version 0.5-5

Date 2025-10-20

Description Stores and eases the manipulation of spectra and associated data, with dedicated classes for spatial and soil-related data.

URL https://github.com/pierreroudier/spectacles/

BugReports https://github.com/pierreroudier/spectacles/issues

Depends R (>= 2.14)

Imports ggplot2, methods, plyr, reshape2, stringr, baseline, signal, epiR

Suggests caret, pls, RColorBrewer, knitr, rmarkdown, dplyr

License GPL-3

LazyData true

Collate 'spectacles-package.R' 'AAA-Classes.R' 'Spectra-methods.R' 'SpectraDataFrame-methods.R' 'ZZZ.R' 'aggregate_spectra.R' 'australia.R' 'big_head.R' 'kenstone.R' 'performance.R' 'plot-methods.R' 'preprocessing.R' 'setters.R' 'splice.R'

RoxygenNote 7.3.3

VignetteBuilder knitr

NeedsCompilation no

Author Pierre Roudier [aut, cre],
Max Kuhn [ctb],
Kristian Hovde Liland [ctb],
Bjorn-Helge Mevik [ctb],
Hadley Wickham [ctb],
Raphael Viscarra Rossel [dtc]

Maintainer Pierre Roudier <roudierp@landcareresearch.co.nz>

Repository CRAN

Date/Publication 2025-10-20 07:40:02 UTC

Contents

spectacles-package *Storing, Manipulating and Analysis Spectroscopy and Associated Data in R*

Description

Stores and eases the manipulation of spectra and associated data.

Author(s)

Pierre Roudier

aggregate_spectra *Aggregates spectral and data information*

Description

Aggregates spectral and data information of a `Spectra` object using a user-defined function

Usage

```
## S4 method for signature 'Spectra'
aggregate_spectra(obj, fun = mean, ...)

## S4 method for signature 'SpectraDataFrame'
aggregate_spectra(obj, fun = mean, id = NULL, ...)
```

Arguments

`obj`	see below
`fun`	see below
`...`	see below
`id`	see below

Details

For `SpectraDataFrame` objects, associated data is also aggregated using the function provided by the `fun` option. Additionally, the method for `SpectraDataFrame` has an `id` option that allows to specify an attribute which will be used to split the object, apply sequentially the `fun` function, and recombine the results in an unique object.

Value

An object of the same class as `obj`

Methods

x=Spectra

`aggregate_spectra(obj, fun=mean, ...)`

`obj`	A `Spectra` object
`fun`	An aggregation function
`...`	Expressions evaluated in the context of `fun`

x=SpectraDataFrame

`aggregate_spectra(obj, fun=mean, id=NULL, ...)`

`obj`	A `SpectraDataFrame` object
`fun`	An aggregation function
`id`	Attribute(s) to split the object (character vector)
`...`	Expressions evaluated in the context of `fun`

Author(s)

Pierre Roudier <pierre.roudier@gmail.com>

See Also

apply_spectra

Examples

```
# Loading example data
data(australia)
spectra(australia) <- sr_no ~ ... ~ 350:2500

# Aggregation on the whole collection
m <- aggregate_spectra(australia, fun = mean)
plot(m)

# Aggregation factor-wise

# Generate some kind of factor
australia$fact <- sample(
  LETTERS[1:3],
  size = nrow(australia),
  replace = TRUE
)
m <- aggregate_spectra(australia, fun = mean, id = 'fact')
plot(m)
```

apply_spectra *Apply a function on the spectra of a Spectra* object*

Description

Aggregates spectral and data information of a Spectra object using a user-defined function.

Apply a function and update the spectra of a Spectra object. This function is particularly interesting for pre-processing, e.g to derive the spectra, or apply pre-processing functions such as snv.

Usage

```
apply_spectra(obj, fun, ...)
```

Arguments

obj	an object inheriting from class Spectra
fun	an aggregation function
...	expressions evaluated in the context of fun

Details

The philosophy of this function is to let the user free to use any function to pre-process a spectra collection, using either functions from the stats package, functions from other packages such as signal, or personal functions.

Value

An object of the same class as obj

Author(s)

Pierre Roudier <pierre.roudier@gmail.com>

See Also

aggregate_spectra, snv, rnv

Examples

```
# Loading example data
data(australia)
spectra(australia) <- sr_no ~ ... ~ 350:2500

# Second derivative
r <- apply_spectra(australia, diff, 2)
plot(r)

# Smoothing kernel
```

```
k <- kernel("daniell", 20) # define a kernel
r <- apply_spectra(australia, kernapply, k)
plot(r)

## Not run:
# Savitzky-Golay filter (from the signal package)
library(signal)
r <- apply_spectra(australia, sgolayfilt, n = 33, p = 4)
plot(r)

## End(Not run)
```

---

australia *Australia spectra library data set*

Description

The australia data set gathers 100 soil spectra, along side with their organic carbon, pH and clay content values. This data set has been collected by CSIRO.The oz dataset is the first 5 samples of australia, and is here to keep examples fast to run.

The data.frame contains the following columns:

- sr_no: a unique identifier for each spectrum
- carbon: soil organic carbon values
- ph: soil pH values
- clay: soil clay values
- X350, X351, . . . ,X2499, X2500: reflectance in wavelengths 350 to 2500nm

Author(s)

Data kindly contributed by Raphael Viscarra Rossel. To reduce the size of the package, the original dataset has been reduced to 100 spectra using the Kennard-Stone algorithm.

References

Viscarra Rossel, R. and Behrens, T. 2010.

Examples

```
data(australia)
big.head(australia)
spectra(australia) <- sr_no ~ ... ~ 350:2500

data(oz)
big.head(oz)
spectra(oz) <- sr_no ~ ... ~ 350:2500
```

base_line *Baseline correction using the baseline package*

Description

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

Usage

```
## S4 method for signature 'Spectra'
base_line(object, ...)
```

Arguments

object	an object inheriting from class Spectra
...	additional arguments to be passed to the baseline function in the baseline package. The main option would be 'method', to switch between the several baseline methods presented in teh details section.

Details

The baseline package implements various algorithms for the baseline correction. The following methods are available:

- 'als': Baseline correction by 2nd derivative constrained weighted regression
- 'fillPeaks': An iterative algorithm using suppression of baseline by means in local windows
- 'irls' (default): An algorithm with primary smoothing and repeated baseline suppressions and regressions with 2nd derivative constraint
- 'lowpass': An algorithm for removing baselines based on Fast Fourier Transform filtering
- 'medianWindow': An implementation and extention of Mark S. Friedrichs' model-free algorithm
- 'modpolyfit': An implementation of Chad A. Lieber and Anita Mahadevan-Jansen's algorithm for polynomial fiting
- 'peakDetection': A translation from Kevin R. Coombes et al.'s MATLAB code for detecting peaks and removing baselines
- 'rfbaseline': Wrapper for Andreas F. Ruckstuhl, Matthew P. Jacobson, Robert W. Field, James A. Dodd's algorithm based on LOWESS and weighted regression
- 'rollingBall': Ideas from Rolling Ball algorithm for X-ray spectra by M.A.Kneen and H.J. Annegarn. Variable window width has been left out

See baseline package documentation for more information and references.

Additionally, the baseline package provides a nice GUI that helps choosing the good baseline method and the good parametrisation. This GUI can be used with the inspectr package. This is demonstrate in the Examples section.

Value

An object of the same class as `obj` with the continuum removed from its spectra.

Author(s)

Interface to the baseline package by Pierre Roudier <pierre.roudier@gmail.com>, baseline package authored by Kristian Hovde Liland and Bjorn-Helge Mevik

References

Kristian Hovde Liland and Bjorn-Helge Mevik (2011). baseline: Baseline Correction of Spectra. R package version 1.0-1. http://CRAN.R-project.org/package=baseline

See Also

`continuum_removal`, `snv`, `rnv`

Examples

```
# Loading example data
data(australia)
spectra(australia) <- sr_no ~ ... ~ 350:2500

# Subsample for demo purposes
australia <- australia[1:10,]

# Correction using the default method (irls)
bl <- base_line(australia)
plot(bl)

# Specifying another method for baseline calculation
bl2 <- base_line(australia, method = "modpolyfit")
plot(bl2)

# Using the baseline package independently
# (useful to plot the corrections)
## Not run:
library(baseline)
bl3 <- baseline(spectra(australia), method = 'irls')
class(bl3) # this is a baseline object
plot(bl3)
# Affecting the baseline-corrected spectra back
# to the SpectraDataFrame object
spectra(australia) <- getCorrected(bl3)
plot(australia)

# Using the baselineGUI with inspectr
baselineGUI(spectra(australia))
## When happy with a configuration, clik "Apply to all" and
## save the results under a name, e.g. "corrected.spectra"
spectra(australia) <- getCorrected(corrected.spectra)
plot(australia)
```

```
## End(Not run)
```

---

big.head *Return the First or Last Part of an Object*

Description

Return the First or Last Part of an Object

Usage

```
big.head(x, n = 5, l = 5, r = 5)
```

Arguments

x	a "data.frame" or a "matrix" object
n	a single, positive integer, number of rows for the object to return
l	a single, positive integer, the number of columns to include on the left
r	a single, positive integer, the number of columns to include on the right

Details

Returns the first or last rows of a data frame like head() and tail(), but also only returns the first and last columns. This has been implemented to check big data frames.

Value

An object (usually) like 'x' but generally smaller.

Author(s)

Pierre Roudier <pierre.roudier@gmail.com>

See Also

head, tail

Examples

```
big.head(mtcars)
big.tail(mtcars)
big.tail(mtcars, 10)
big.head(mtcars, 10, 2, 4)
big.head(mtcars, , , 1)

data(australia)
big.head(australia)
```

`continuum_removal` *Continuum removal*

Description

Operates a continuum removal on a vector.

Usage

```
continuum_removal(x, wl = as.numeric(names(x)), upper = TRUE)
```

Arguments

`x`	a numeric vector
`wl`	the wavelengths of the spectra
`upper`	if TRUE, removes the upper convex hull from the spectra, if FALSE, takes the lower convex hull

Details

This operation is commonly done to normalize reflectance spectra and allow comparison of individual absorption features from a common baseline. The removal is based on the upper convex hull of the spectra.

This function is working on vectors. It may applied on matrix or data.frames using the `apply` function, or on `Spectra*` objects using the `apply_spectra` function.

Value

A numeric vector with its continuum removed.

Author(s)

Pierre Roudier <`pierre.roudier@gmail.com`>, based on code from Raphael Viscarra-Rossel.

References

Clark, R.N., and Roush, T.L. 1984. Reflectance spectroscopy: Quantitative analysis techniques for remote sensing applications. Journal of Geophysical Research 89, 6329–6340.

See Also

`baseline`, `snv`, `rnv`

Examples

```
# Loading example data
data(australia)
spectra(australia) <- sr_no ~ ... ~ 350:2500

s <- apply_spectra(australia, continuum_removal)
plot(s)

s <- apply_spectra(australia, continuum_removal, upper = FALSE)
plot(s)
```

---

cut *Manipulating the wavelength range of a* Spectra *object*

Description

This methods allows to either select or remove a specific range of wavelengths from a Spectra object.

Usage

```
## S4 method for signature 'Spectra'
cut(x, ..., wl)
```

Arguments

x	an object inheriting from class Spectra
...	ignored
wl	a vector of the wavelengths to either select or remove from x

Details

The wavelengths are extracted if wl > 0, or removed if wl < 0. You can't mix negative and positive index in wl.

Value

An object of the same class as x.

Author(s)

Pierre Roudier <pierre.roudier@gmail.com>

Examples

```
# Loading example data
data(australia)
spectra(australia) <- sr_no ~ ... ~ 350:2500

# Extracting a specific wavelengths range
s <- cut(australia, wl = 450:550)
plot(s)

s <- cut(australia, wl = c(450:550, 1850:2050))
plot(s)

# Removing a specific wavelengths range
s <- cut(australia, wl = -1*450:550)
plot(s)

s <- cut(australia, wl = -1*c(450:550, 1850:2050))
plot(s)
```

---

dimensions *Retrieve dimensions of Spectra* objects*

Description

Retrieves the wavelengths units and the spectral resolution from `Spectra*` objects.

Usage

```
## S3 method for class 'Spectra'
length(x)

## S4 method for signature 'Spectra'
nrow(x)

## S4 method for signature 'Spectra'
ncol(x)

## S3 method for class 'Spectra'
dim(x)
```

Arguments

`x`	For `nrow`, `length`, `dim`, a `Spectra` object. For `ncol`, a `SpectraDataFrame` object.

Details

* Methods for Spectra objects

nrow returns the number of individuals in the collection length returns the number of wavelengths in the collection ncol returns NULL dim returns a vector containing (1) the number of individuals and (2) in the number of wavelengths in the collection

* Methods for Spectra objects

nrow returns the number of individuals in the collection length returns the number of wavelengths in the collection ncol returns the number of attributes in the collection dim returns a vector containing (1) the number of individuals, (2) in the number of wavelengths, and (3) the number of attributes in the collection

Value

nrow, ncol, nwl, and length, return an integer, dim returns a vector of length 2 or 3 (see Details section).

Author(s)

Pierre Roudier <pierre.roudier@gmail.com>

Examples

```
# Loading example data
data(australia)
spectra(australia) <- sr_no ~ ... ~ 350:2500

nrow(australia)
ncol(australia)
length(australia)
dim(australia)
```

---

extraction-methods *Extracting and replacing parts of Spectra* objects*

Description

These methods emulates classic base methods '[', '[[' and '$' to extract or replace parts of Spectra* objects.

Usage

```
\S4method{[}{Spectra}(x,i,j,...,drop=FALSE)
\S4method{[[}{Spectra}(x,i,j,...)
\S4method{$}{SpectraDataFrame}(x,name)
\S4method{$}{Spectra}(x,name) <- value
\S4method{[[}{Spectra}(x,i,j,...) <- value
```

Arguments

x	an object of class Spectra or SpectraDataFrame
i, j, ...	indices specifying elements to extract or replace
drop	currently ignored
name	A literal character string or a name (possibly backtick quoted)
value	typically an array-like R object of a similar class as x

Value

These methods either return an object of the same class as x, or can promote a Spectra object to a SpectraDataFrame object by adding data ("[[<-" and "$<-" methods).

Methods

x=Spectra

```
x[i, j, ..., drop = FALSE]
x$name <- value
x[[name]] <- value
```

x	A Spectra object
i	Row index of the selected individuals
j	Selected wavelengths
name	A literal character string or a name
...	Ignored
drop	Ignored

x=SpectraDataFrame

```
x[i, j, k, ..., drop = FALSE]
x[[name]]
x[[name]] <- value
x$name
x$name <- value
```

x	A SpectraDataFrame object
i	Row index of the selected individuals
j	Selected wavelengths
k	Selected columns in the @data slot
name	A literal character string or a name
...	Ignored
drop	Ignored

Author(s)

Pierre Roudier <pierre.roudier@gmail.com>

Examples

```
# Loading example data
data(australia)
spectra(australia) <- sr_no ~ ... ~ 350:2500

# Getting features information from SpectraDataFrame
australia$carbon
australia[['carbon']]

# Creating new features
australia$foo <- runif(nrow(australia))
australia[['bar']] <- runif(nrow(australia))

# Replacing values
australia$foo <- sample(
  LETTERS[1:5],
  size = nrow(australia),
  replace = TRUE
)
australia[['bar']] <- sample(
  c(TRUE, FALSE),
  size = nrow(australia),
  replace = TRUE
)

# Promote Spectra to SpectraDataFrame
s <- as(australia, 'Spectra')
class(s)
s$foo <- runif(nrow(s))
```

`features` *Retrieves or sets the data slot of a SpectraDataFrame object.*

Description

Either retrieves the attributes values from the data slot of a SpectraDataFrame object, or upgrades a Spectra object to a SpectraDataFrame object by initialising its data slot by a suitable `"data.frame"` object.

Usage

```
## S4 method for signature 'SpectraDataFrame'
features(object,exclude_id)
## S4 replacement method for signature 'Spectra'
features(object,safe,exclude_id,key,append) <- value
```

Arguments

object	a Spectra object
exclude_id	see below
value	see below
safe	see below
key	see below
append	see below

Value

The features methods return a data.frame object, while the "features<-" methods return a SpectraDataFrame object.

Methods

x=Spectra

```
features(object, safe=TRUE, key=NULL, exclude_id=TRUE) <- value
```

object	A Spectra object
safe	Logical. If TRUE, data is being added to the object using a SQL join (using a key field given by the key opt
key	Character, name of the column of the data.frame storing the ids for the SQL join. Ignored if safe is FALSE.
exclude_id	Logical, if TRUE, ids used for the SQL join are removed from the data slot after the join.

x=SpectraDataFrame

```
features(obj, exclude_id=TRUE)
features(obj, safe=TRUE, key=NULL, exclude_id=TRUE, append=TRUE) <-value
```

object	A SpectraDataFrame object
safe	Logical. If TRUE, data is being added to the object using a SQL join (using a key field given by the key opt
key	Character, name of the column of the data.frame storing the ids for the SQL join. Ignored if safe is FALSE.
exclude_id	Logical. For the features method, if TRUE, the spectra ids are added to the data.frame that is returned. Fo
append	Logical, if TRUE, the data is appended to any existing data. if FALSE, the data provided is erasing any existi

Author(s)

Pierre Roudier <pierre.roudier@gmail.com>

See Also

spectra, wl, SpectraDataFrame-class

Examples

```
# Loading example data
data(oz)
spectra(oz) <- sr_no ~ ... ~ 350:2500

# Printing available data
features(oz)

# Promoting a Spectra to a SpectraDataFrame object
s <- as(oz, "Spectra")

# Generating dummy data
d <- data.frame(
  id = ids(oz),
  foo = runif(nrow(oz)),
  bar = sample(LETTERS[1:5], size = nrow(oz), replace = TRUE)
)
head(d)

# Affecting data to Spectra object
features(s) <- d
features(s)

# Adding data to an existing SpectraDataFrame object

features(oz) <- d
features(oz)
```

`fill_spectra` *Fill missing wavelengths of a Spectra* object with a given value*

Description

Fill missing wavelengths of a Spectra* object with a given value. This is mostly usefull to include NA values in the spectra in order to show missing bits in plots.

Usage

```
## S4 method for signature 'Spectra'
fill_spectra(obj, ref = NULL, fill = NA, ...)
```

Arguments

`obj`	an object inheriting from class `Spectra`
`ref`	a numeric vector, giving the reference wavelengths (ie the entire range of wavelengths expected to be in the spectra before some waveleng5ths have been cut out). If NULL, the function is trying to guess it.
`fill`	values to fill gaps in the data with
`...`	ignored

Details

At this stage removing gaps does not work well with irreguarly spaced wavelengths. Results might be odd for binned spectra.

Value

An object of the same class as `obj`

Author(s)

Pierre Roudier <pierre.roudier@gmail.com>

Examples

```
# Loading example data
data(australia)
spectra(australia) <- sr_no ~ ... ~ 350:2500

# Cut wavelengths out of the collection
oz <- cut(australia, wl=-1*c(355:400, 2480:2499))
big.head(spectra(oz), , 7)

# Giving the wavelengths at which I want data
oz_filled <- fill_spectra(oz, ref = 350:2500, fill = NA)
big.head(spectra(oz_filled), , 7)
plot(oz_filled)

# Trying to guess ref values
oz_filled <- fill_spectra(oz, fill = -999)
big.head(spectra(oz_filled), , 7)
plot(oz_filled)
```

`ids` *Retrieves or sets the ids of a* `Spectra*` *object.*

Description

Either retrieves the wavelengths from a `Spectra*` object, or creates a `Spectra*` object from a `"data.frame"` object by setting some of its columns as the wavelengths. The `"ids<-"` method for `SpectraDataFrame` objects allows to use a formula interface to use a column in its `data` slot as the object IDs (see the last example provided in the Examples section).

Usage

```
ids(object, ...)
ids(object) <- value
```

```
## S4 replacement method for signature 'Spectra'
ids(object) <- value

## S4 replacement method for signature 'SpectraDataFrame'
ids(object) <- value
```

Arguments

object	an object of class "Spectra" or inheriting from this class
...	as.vector Controls whether the IDs are returned as a vector or as a data.frame (defaults to TRUE)
value	character vector for new IDs

Value

The ids methods return a vector if as.vector is TRUE, a data.frame otherwise. The "ids<-" method return a SpectraDataFrame object (or a Spectra object if the column in the data slot that has been used to initiate the IDs was the only attribute).

Methods

- ids(object, ..., as.vector = TRUE)
- ids(object) <- value

Author(s)

Pierre Roudier <pierre.roudier@gmail.com>

Examples

```
# Loading example data
data(oz)
spectra(oz) <- sr_no ~ ... ~ 350:2500

# Retrieving ids
ids(oz)

# Setting ids using a vector of values
ids(oz) <- seq_len(nrow(oz))
ids(oz)

# Setting ids using an attribute
oz$new_id <- seq_len(nrow(oz)) + 1000
ids(oz) <- ~ new_id
ids(oz)
```

kenstone *Kennard-Stone algorithm for optimal calibration set selection.*

Description

An implemnentation of the Kennard-Stone algorithm for calibration set selection.

Usage

```
kenstone(x, size, ...)
```

Arguments

x	a Spectra object
size	a positive number, the number of items to choose from
...	ignored

Value

A vector of length size giving the indices of the selected individuals in x.

Author(s)

Pierre Roudier <pierre.roudier@gmail.com>

References

Kennard, L.A. Stone, Technometrics 11 (1969) 137.

load_oz *Load the australia dataset*

Description

Loads the australia dataset as a SpectraDataFrame

Usage

```
load_oz(n = NULL)
```

Arguments

n	the number of spectra to return. By default, it returns all 100 spectra in the dataset.

Value

a SpectraDataFrame

Author(s)

Pierre Roudier

Examples

```
oz <- load_oz()
```

`melt_spectra` *Melts the spectra data of a Spectra object and returns it as wide format.*

Description

This function is very useful when wanting to plot spectra using the lattice or ggplot2 packages

Usage

```
melt_spectra(obj, attr=NULL, ...)

## S4 method for signature 'SpectraDataFrame'
melt_spectra(obj, attr = NULL, ...)
```

Arguments

`obj`	an object of class "`Spectra`" or inheriting from this class
`attr`	vector of id variables against which the spectra will be melted (see `melt`)
`...`	further arguments passed to or from other methods

Methods

x=Spectra

```
melt_spectra(obj, ...)
```

`obj`	A `Spectra` object
`...`	Ignored

x=SpectraDataFrame

```
melt_spectra(obj, attr=NULL, ...)
```

`obj`	A `SpectraDataFrame` object

attr Character, the name of an attribute in the object data to split the spectra against.
... Ignored

Author(s)

Pierre Roudier <pierre.roudier@gmail.com>

References

Hadley Wickham (2011). The Split-Apply-Combine Strategy for Data Analysis. Journal of Statistical Software, 40(1), 1-29. URL http://www.jstatsoft.org/v40/i01/.

Examples

```
# Loading example data
data(australia)
spectra(australia) <- sr_no ~ ... ~ 350:2500

# Simple melt
r <- melt_spectra(australia)
head(r)

## Not run:
# Melt against some factor (or continuous data), and plot
# using ggplot2

# Create some factor
australia$fact <- sample(
  LETTERS[1:3],
  size = nrow(australia),
  replace = TRUE
)
r <- melt_spectra(australia, attr = 'fact')

# Create plot
library(ggplot2)
p <- ggplot(r) +
  geom_line(aes(x=wl, y=nir, group=id, colour=fact)) +
  theme_bw()
print(p)

## End(Not run)
```

---

mutate *Mutate a Spectra* object by transforming the spectra values, and/or adding new or replacing existing attributes.*

Description

This function is very similar to `transform` but it executes the transformations iteratively so that later transformations can use the columns created by earlier transformations. Like transform, unnamed components are silently dropped.

Either the spectra, and/or the attributes (if the `.data` inherits from the `SpectraDataFrame` class) can be affected:

- To affect the spectra, one should use the `nir` placeholder, eg `nir = log(1/nir)`
- To affect the attributes of the object, the definitions of new columns are simply given using attributes names, `newAttribute = 1/sqrt(attribute)`
- Both spectra and attrbutes can be transformed in one command.

Usage

```
## S4 method for signature 'Spectra'
mutate(.data, ...)
```

Arguments

`.data`	an object inheriting from the `Spectra` class
`...`	named parameters giving definitions of new columns

Author(s)

Pierre Roudier <pierre.roudier@gmail.com>, from code from Hadley Wickham

References

Hadley Wickham (2011). The Split-Apply-Combine Strategy for Data Analysis. Journal of Statistical Software, 40(1), 1-29. URL http://www.jstatsoft.org/v40/i01/.

Examples

```
# Loading example data
data(australia)
spectra(australia) <- sr_no ~ ... ~ 350:2500

# Modifying spectra
m <- mutate(australia, nir = log1p(1/nir))
plot(m)

# Modifying and creating attributes
m <- mutate(
  australia,
  sqrt_carbon = sqrt(carbon),
  foo = clay + ph,
  nir = log1p(1/nir)
)
plot(m)
```

plot-Spectra *Plots an object inheriting from the Spectra class*

Description

The philosophy of this plotting routine is to provide a "quick'n'dirty" way to plot your spectra collection. For advanced visualisations, the use of `melt_spectra` alongside with ggplot2 or lattice is encouraged.

Usage

```
## S3 method for class 'Spectra'
plot(x,gg,gaps,attr,...)

## S3 method for class 'Spectra'
plot(x, gg = FALSE, gaps = TRUE, attr = NULL, ...)
```

Arguments

x	an object of class "Spectra" or inheriting from this class
gg	if TRUE, uses the ggplot2 package to plot the data, if FALSE uses matplot from base graphics (much faster)
gaps	if TRUE, gaps in the spectra are not plotted
attr	attribute against which lines are coloured (only for gg = TRUE
...	additional parameters passed to matplot

Author(s)

Pierre Roudier <pierre.roudier@gmail.com>

Examples

```
# Loading example data
data(australia)
spectra(australia) <- sr_no ~ ... ~ 350:2500

# Default plotting method
plot(australia[1:5,])

# Default plot using ggplot2
plot(australia[1:5,], gg = TRUE)

## Not run:

# Managing gaps in the spectra
s <- cut(australia, wl =c(-1*450:500, -1*1800:2050))
plot(s, gaps = TRUE)
```

```
plot(s, gaps = FALSE)

# passing various options to matplot
plot(
  australia,
  lty = 1:5,
  col = 'blue',
  xlab = 'foo', ylab = 'bar',
  ylim = c(0.4,0.6),
  main = 'my plot'
)

# Using colour ramps
plot(
  australia,
  lty = 1,
  col = rainbow(10),
  main = "It is possible to create really ugly visualisations"
)

# Example using colours given by ColorBrewer (http://colorbrewer2.org/)
library(RColorBrewer)
plot(australia, lty = 1, col = brewer.pal(n = 8, name = "Set2"))

# Using an attribute to group spectra

# Generate some kind of factor
australia$fact <- sample(
  LETTERS[1:3],
  size = nrow(australia),
  replace = TRUE
)

s <- aggregate_spectra(australia, fun = mean, id = 'fact')
plot(s, gg = TRUE, attr = 'fact')

## End(Not run)
```

`plot_offset` *Offset plot of a collection of spectra*

Description

Creates an offset plot of a collection of Spectra

Usage

```
plot_offset(x, offset = 1)
```

Arguments

x	an object of class "Spectra" or inheriting from this class
offset	Offset between spectra

Author(s)

Pierre Roudier

Examples

```
oz <- load_oz(3)
plot_offset(oz)
plot_offset(oz, 0.3)
plot_offset(oz, 2)
```

---

plot_stack *Stacked plot of a collection of spectra*

Description

Creates a stacked plot of a collection of Spectra

Usage

```
plot_stack(x)
```

Arguments

x	an object of class "Spectra" or inheriting from this class

Author(s)

Pierre Roudier

Examples

```
oz <- load_oz(3)
plot_stack(oz)
```

plot_summary *Summary plot of a collection of spectra*

Description

Creates a summary plot of a collection of Spectra

Usage

```
plot_summary(x, fun = mean, se = TRUE, ...)
```

Arguments

x	an object of class "Spectra" or inheriting from this class
fun	an aggregation function
se	if TRUE, plots the standard deviation around the summary spectra (computed by function as given by fun). If FALSE, does not plot dispersion information. If a function, uses this function instead of sd.
...	additional parameters, currently ignored

Author(s)

Pierre Roudier

Examples

```
oz <- load_oz()
plot_summary(oz)
```

postResampleSpectro *Calculates performance indictors across resamples*

Description

Given two numeric vectors of data, the root mean squared error, the R-squared, the bias, the RPD, the RPIQ, the CCC and the standard error are calculated. For two factors, the overall agreement rate and Kappa are determined.

Usage

```
postResampleSpectro(pred, obs)
spectroSummary(data, lev = NULL, model = NULL)
```

Arguments

data	a data frame or matrix with columns obs and pred for the observed and predicted outcomes
lev	a character vector of factors levels for the response. In regression cases, this would be NULL.
model	a character string for the model name
pred	A vector of numeric data
obs	A vector of numeric data

Details

This function extends postResample in the caret package.

Author(s)

Pierre Roudier, adapted from code from Max Kuhn

Examples

```
predicted <-  matrix(rnorm(50), ncol = 5)
observed <- rnorm(10)
apply(predicted, 2, postResampleSpectro, obs = observed)
```

---

rbind *Stacking* Spectra *objects together*

Description

This method stacks two or more Spectra* objects together.

Usage

```
## S3 method for class 'Spectra'
rbind(..., create_new_ids = FALSE, new_ids = NULL)

## S3 method for class 'SpectraDataFrame'
rbind(..., create_new_ids = FALSE, new_ids = NULL)
```

Arguments

...	The Spectra objects to be combined.
create_new_ids	allows creation of new ids if the ids of the Spectra* objects you are trying to stack are redundant
new_ids	vector of new ids to be given to the new object

Value

a Spectra* object.

Examples

```
# Loading example data
data(australia)
spectra(australia) <- sr_no ~ ... ~ 350:2500

s <- rbind(australia, australia, create_new_ids = TRUE)

l <- separate(australia, calibration = 0.6)
s <- rbind(l$validation, l$calibration)
```

---

res,numeric-method *Spectral resolution*

Description

Returns the spectral resolution of an object

Usage

```
## S4 method for signature 'numeric'
res(x)

## S4 method for signature 'integer'
res(x)

## S4 method for signature 'Spectra'
res(x)
```

Arguments

x	object an object inheriting from Spectra

Value

a vector

Author(s)

Pierre Roudier <pierre.roudier@gmail.com>

separate *Separates a* Spectra* *object into a calibration and a validation set.*

Description

Separates a Spectra* object into a calibration and a validation set.

Usage

```
## S4 method for signature 'Spectra'
separate(obj, calibration)
```

Arguments

obj	an object inheriting from class SpectraDataFrame
calibration	The fraction of the dataset to be put in the calibration set

Value

An list with two SpectraDataFrame objects, one for the calibration, and the other for the validation.

Author(s)

Pierre Roudier <pierre.roudier@gmail.com>

Examples

```
# Loading example data
data(australia)
spectra(australia) <- sr_no ~ ... ~ 350:2500

l <- separate(australia, calibration=0.7)
# The result is a list of two Spectra* objects
str(l)
lapply(l, nrow)
```

snv *Standard and Robust Normal Variate transformations*

Description

Standard and Robust Normal Variate transformations are often used in chemometrics to normalise a spectra collection and remove the baseline effect.

The Standard Normal Variate transformation (SNV, Barnes et al., 1989) is a common method to reduce within-class variance.

The Robust Normal Variate transformation (RNV, Guo et al., 1999) is a modification of the SNV to make it more robust to closure problems.

These function are to be used in conjonction with apply_spectra.

Usage

```
snv(x)

rnv(x, r)
```

Arguments

x	a vector of numeric values
r	the percentile to use in the RNV computation

Value

A vector of numeric values

Author(s)

Pierre Roudier <pierre.roudier@gmail.com>

References

- Barnes, R.J., Dhanoa, M.S., Lister, S.J. 1989. Standard normal variate transformation and detrending of near-infra-red diffuse reflectance spectra. Applied Spectroscopy 43, 772–777.
- Guo, Q., Wu, W., Massar, D.L. 1999. The robust normal variate transform for pattern recognition with near-infrared data. Analytica Chimica Acta 382:1–2, 87–103.

Examples

```
# Loading example data
data(australia)
spectra(australia) <- sr_no ~ ... ~ 350:2500

# Standard Normal Variate transform
s <- apply_spectra(australia[1:10,], snv)
plot(s)

# The scale function in the base package is actually doing
# the same thing!
s <- apply_spectra(australia[1:10,], scale, center = TRUE, scale = TRUE)
plot(s)

# Robust Normal Variate transform
s <- apply_spectra(australia[1:10,], rnv, r = 0.25)
plot(s)
```

Spectra *Constructor for the Spectra class.*

Description

Constructor for the Spectra class. Creates a Spectra object from scratch.

Usage

```
Spectra(wl = numeric(), nir = matrix(), id = as.character(NA), units = "nm")
```

Arguments

wl	a numeric vector giving the wavelengths at with the spectra have been measured
nir	a "matrix" or a "data.frame" object giving the spectra values for each sample
id	a vector giving the unique id of each sample in the collection
units	a character giving the unit in which the wavelengths values are expressed

Value

a new "Spectra" object

Author(s)

Pierre Roudier <pierre.roudier@gmail.com>

See Also

spectra, wl, Spectra-class, SpectraDataFrame

Examples

```
wls <- 350:2500
id <- c("A", "B")
nir <- matrix(runif(2*length(wls)), nrow = 2)
s <- Spectra(wl = wls, nir = nir, id = id, units = "nm")
```

Spectra-class *Spectra* classes*

Description

The spectacles package provides the user with S4 classes that have been developped to store and manipulate spectroscopy data.

The `Spectra` class is storing the spectra matrix, along with the wavelengths at which those have been measured, the units in whioch those wavelengths are expressed, and a unique id for each sample in the collection.

The `SpectraDataFrame` class is extending the `Spectra` class by giving the opportunity to store attribute data along with the spectra - this is mostly the case when we want to predict physical or chemical properties from the spectra set.

The `SpatialSpectra` and `SpatialSpectraDataFrame` classes are extending the `Spectra` and `SpectraDataFrame` classes using the `SpatialPoints` class from package sp. This allows to store spatial information on the dataset: coordinates, coordinate reference system, bounding box, etc.

Common generic methods implemented for these classes include:

`summary`, `show`, `nrow`, `length`, `plot`, `[`, `[[`, `$`.

`SpatialPoints` methods from the sp package can be applied to `SpatialSpectra` and `SpatialSpectraDataFrame` objects are they inherit from this class.

Slots

`wl` object of class "`numeric`"; the wavelengths at which the spectra has been measured

`nir` object of class "`matrix`"; the spectra, with as many columns as wavelengths, and as many rows as samples

`id` object of class "`data.frame`" with one attribute; the identification strings for each sample in the collection

`units` object of class "`character`"; units in which the wavelengths are expressed

`data` object of class data.frame containing the attribute data

Objects from the Class

Objects can be created by calls of the form `new("Spectra", ...)`, with the constructor functions like `Spectra(...)`, or with the helper functions such as `wl` and `spectra`.

Author(s)

Pierre Roudier <pierre.roudier@gmail.com>

Examples

```
showClass("Spectra")
showClass("SpectraDataFrame")
```

spectra-methods *Retrieves or sets the spectra of a* Spectra* *objects.*

Description

Either retrieves the spectra matrix from a Spectra* object, or creates a Spectra* object from a "data.frame" object different interfaces detailed below.When applied to a Spectra* object, this functions simply returns the spectra it is storing.

Usage

```
## S4 method for signature 'Spectra'
spectra(object)

## S4 replacement method for signature 'data.frame'
spectra(object, ...) <- value

## S4 replacement method for signature 'Spectra'
spectra(object) <- value
```

Arguments

object	an object of class "Spectra" or inheriting from this class
...	see details below
value	see details below

Details

If applied on a "data.frame" object, it is an helper function to create a Spectra* object. Two kind of interfaces are then available. value can be:

a vector: Similarly to wl, the wavelengths of the spectra can be passed by a "numeric" vector. Alternatively, the names of the columns that contain the spectra information can be passed as a "character" vector.

a formula: This interface is specific to inspectr. It follows a scheme where differents parts can be observed, the id column, the attributes columns, and the spectra columns, described by the wavelengths at which it has been measured:

- **Placeholders:**
  - ... placeholder for all the columns of your data.frame object except those that have been already used in other parts of the formula. This can lead to errors. E.g. if object has data one every wavelength between 350 and 2500 nm, spectra(object) <- id_field ~ ... ~ 500:2500 will stores the columns corresponding to the wavelengths 350-499 nm in its data slot!
  - id For the creation of a SpectraDataFrame, it is important to always specify an id field in the formula. If no id column is present, the id placeholder will create one for you.

- spectra(object) <- ~ 350:2500 will build a Spectra object from the wavelengths between 350 and 2500, based on the column names.
- spectra(object) <- ~ 350:2:2500 will build a Spectra object from the wavelengths in seq(350, 2500, by = 2)
- spectra(object) <- ~ 500:2350 will build a Spectra object from the wavelengths between 500 and 2350, even though other wavelengths are present (they will be dropped)
  In the three later cases, the id field has been dropped (it will be automatically created). If you want to use a column of "data.frame" as an id filed, you can still use the first part of the formula:
- spectra(object) <- id_field ~ 350:2500
- spectra(object) <- id_field ~ 350:5:2500
  Some data can also be added to the object, which will then be of SpectraDataFrame class:
- spectra(object) <- id_field ~ property1 ~ 500:2300 will create a SpectraDataFrame with ids from the id_field column, data from the property1 column, and spectral information between 500 and 2300 nm. That means that data property2, and all spectral information from bands < 500 nm or > 2300 nm will be dropped
  You can also combine the placeholders:
- spectra(object) <- id_field ~ ... ~ 350:2500 will create a SpectraDataFrame object with ids from the id_field column, all spectral bands between 350 and 2500 nm. The data slot is given all the remaining columns.

Author(s)

Pierre Roudier <pierre.roudier@gmail.com>

Examples

```
# Loading example data
data(oz)
class(oz) # this is a simple data.frame
# structure of the data.frame: it is rowwise-formatted
big.head(oz)

## CREATING Spectra OBJECTS
##

# Using spectra() to initiate a Spectra from
# the data.frame
spectra(oz) <- sr_no ~ 350:2500

# It is possible to select wavelengths using the formula interface
data(oz)
spectra(oz) <- sr_no ~ 350:5:2500

data(oz)
spectra(oz) <- sr_no ~ 500:1800

## CREATING SpectraDataFrame OBJECTS
##
```

```
# Using spectra() to initiate a SpectraDataFrame from
# the data.frame
data(oz)
spectra(oz) <- sr_no ~ carbon + ph + clay ~ 350:2500

# Selecting data to be included in the SpectradataFrame object
data(oz)
spectra(oz) <- sr_no ~ carbon ~ 350:2500

# Forcing the creation of new ids using the id keyword in the
# formula interface
data(oz)
spectra(oz) <- id ~ carbon ~ 350:2500
ids(oz, as.vector = TRUE)

# Using the "..." short-hand to select all the remaining columns
data(oz)
spectra(oz) <- sr_no ~ ... ~ 350:2500

## CREATING Spectra OBJECTS FROM
## BY-COLS-FORMATTED DATA
##

# For data formatted in the colwise format,
# use the "colwise" mode

# Transforming data into colwise format
# for demonstration's sake
#
m <- melt_spectra(oz)
oz_by_col <- reshape2::acast(m, ... ~ sr_no)
oz_by_col <- data.frame(
  wl = rownames(oz_by_col),
  oz_by_col,
  check.names = FALSE)

# Here's colwise-formatted data
big.head(oz_by_col)

# Convert it into Spectra object
spectra(oz_by_col, mode = "colwise") <- wl ~ ...

# Then data can be added to promote it as a SpectraDataFrame
my.data <- features(oz)
features(oz_by_col) <- my.data
```

`SpectraDataFrame` *Constructor for the SpectraDataFrame class.*

Description

Constructor for the SpectraDataFrame class. Creates a SpectraDataFrame object, either from scratch, or from an existing Spectra object.

Usage

```
SpectraDataFrame(
  ...,
  wl = numeric(),
  nir = matrix(),
  id = as.character(NA),
  units = "nm",
  data = data.frame()
)
```

Arguments

...	an object inheriting from "Spectra"
wl	a numeric vector giving the wavelengths at with the spectra have been measured
nir	a "matrix" or a "data.frame" object giving the spectra values for each sample
id	a vector giving the unique id of each sample in the collection
units	a character giving the unit in which the wavelengths values are expressed
data	object of class "data.frame" containing the attribute data

Value

a new "SpectraDataFrame" object

Author(s)

Pierre Roudier <pierre.roudier@gmail.com>

See Also

spectra, wl, Spectra-class

Examples

```
# Creating a SpectraDataFrame object from scratch
my.wl <- 350:2500
my.id <- c("A", "B")
my.nir <- matrix(runif(2*length(my.wl)), nrow=2)
my.data <- data.frame(foo = runif(2), bar = LETTERS[1:2])
my.sdf <- SpectraDataFrame(wl = my.wl, nir = my.nir, id = my.id, data = my.data)

# Creating a SpectraDataFrame object from an existing Spectra object
my.s <- Spectra(wl = my.wl, nir = my.nir, id = my.id)
my.sdf <- SpectraDataFrame(my.s, data = my.data)
```

splice *Splice correction of a spectra collected using ASD hardware*

Description

This is the correction method available in the ViewSpec Pro software from ASD, which aims at correcting steps in the data (see details).

Usage

```
## S4 method for signature 'Spectra'
splice(x, locations = list(c(750, 1000), c(1830, 1950)))
```

Arguments

x	a Spectra object
locations	the wavelengths to cut out and interpolate

Details

The SWIR1 part of the spectrum (1000-1800 nm) is taken as a reference for corrections as it is stable to the instrument sensitivity drift (Beal and Eamon, 2010)

This is based on a description of the splice correction algorithm described in:

Beal, D. and Eamon, M., 1996. Dynamic, Parabolic Linear Transformations of 'Stepped' Radiometric Data. Analytical Spectral Devices Inc., Boulder, CO.

Value

an object of same class as x

Author(s)

Pierre Roudier <pierre.roudier@gmail.com>

Examples

```
data(australia)
spectra(australia) <- sr_no ~ ... ~ 350:2500
oz_spliced <- splice(australia)
plot(oz_spliced)
```

split *Split a Spectra* object using factors*

Description

Splits a Spectra* object into groups using a factor, either a provided as a vector or as an attribute in the features of the object.

Usage

```
## S4 method for signature 'Spectra'
split(x, f, drop = FALSE, ...)
```

Arguments

x	Spectra object
f	either a vector of factors (for objects inheriting from Spectra), or the name or indice of an attribute in the data slot of an object inheriting from SpectraDataFrame
drop	ignored
...	further potential arguments passed to methods.

Details

This is an adaptation of the split function in the base package.

Value

A list of objects of same class as x.

Author(s)

Pierre Roudier <pierre.roudier@gmail.com>

Examples

```
# Loading example data
data(australia)
spectra(australia) <- sr_no ~ ... ~ 350:2500

# On a Spectra object, we need to provide a vector of factors
# to split the object
s <- as(australia, 'Spectra')
# We make up some kind of factor to split the data.
idx <- sample(letters[1:5], replace = TRUE, size = nrow(s)) # This is a vector
r <- split(s, idx)
str(r)

# On a SpectradataFrame object, we can also provide the name or index
```

```
# of an attribute

# Generate some kind of factor
australia$fact <- sample(LETTERS[1:3], size = nrow(australia), replace = TRUE)
r <- split(australia, 'fact')
str(r)

# A list is returned, and is thus ready for use with lapply, or any
# of the l*ply functions from the plyr package
lapply(r, nrow)
```

---

subset *Subset SpectraDataFrame object*

Description

Returns subsets of a SpectraDataFrame object.

Usage

```
subset(x, ...)
```

Arguments

x	SpectraDataFrame object
...	see details below

Details

Additional parameters:

subset logical expression indicating elements or rows to keep: missing values are taken as false

select expression, indicating columns to select from the data slot

drop passed on to "[" indexing operator

... Additional arguments

Value

SpectraDataFrame object

Author(s)

Pierre Roudier <pierre.roudier@gmail.com>

See Also

mutate

Examples

```
# Loading example data
data(australia)
spectra(australia) <- sr_no ~ ... ~ 350:2500

# Subset on attributes
s <- subset(australia, carbon < 5)

# Subset and selection of attributes
s <- subset(australia, carbon < 5, select = 1)
```

---

summary *Summary*

Description

Summarize a Spectra* object.

Usage

```
\method{summary}{Spectra}(object, ...)

## S3 method for class 'summary.Spectra'
print(x, ...)
```

Arguments

object	an object of class Spectra or SpectraDataFrame
...	Additional arguments passed to summary
x	a result of summary

Value

A "summary.Spectra" object

NULL

Author(s)

Pierre Roudier <pierre.roudier@gmail.com>

Examples

```
data(oz)
spectra(oz) <- sr_no ~ ... ~ 350:2500
summary(oz)
```

wl *Retrieves or sets the wavelengths of a* Spectra* *object.*

Description

Either retrieves the wavelengths from a Spectra* object, or creates a Spectra* object from a "data.frame" object by setting some of its columns as the wavelengths.

When applied to a Spectra* object, this functions simply returns the wavelengths of the spectra it is storing.

If applied on a "data.frame" object, it is an helper function to create a Spectra* object. It then needs to be indicated the wavelengths at which the spectra values are measured. The assumption is that each row of the "data.frame" is a spectra, and the column names of the "data.frame" contain the wavelengths values.

If all the columns are used to create the Spectra* object, a Spectra object is created. If some attributes are left, they will be used to generate a SpectraDataFrame object.

Usage

```
## S4 method for signature 'Spectra'
wl(object)

## S4 replacement method for signature 'data.frame'
wl(object) <- value

## S4 replacement method for signature 'Spectra'
wl(object) <- value
```

Arguments

object	a "data.frame" or an object inheriting from class Spectra
value	the wavelengths of the Spectra* object to create

Value

If applied on a "data.frame", either a Spectra or a SpectraDataFrame object. If applied on a Spectra* object, a vector.

Author(s)

Pierre Roudier <pierre.roudier@gmail.com>

See Also

spectra, Spectra-class, SpectraDataFrame-class

Examples

```
# Loading example data
data(oz)
spectra(oz) <- sr_no ~ ... ~ 350:2500

# Retrieving wavelengths from Spectra* object
wl(oz)

# Replacing wavelength values - USE WITH CAUTION!
wl(oz) <- 1:length(oz)
wl(oz)

# Use to initiate a Spectra* object from a data.frame
data(oz)
wl(oz) <- 350:2500
ids(oz) <- ~ sr_no
```

---

wl_units *Wavelengths of Spectra* objects*

Description

Retrieves the wavelengths units from Spectra* object

Usage

```
wl_units(object)

## S4 replacement method for signature 'Spectra'
wl_units(object) <- value
```

Arguments

object	an object inheriting from class Spectra
value	a character string

Value

A vector

Author(s)

Pierre Roudier <pierre.roudier@gmail.com>

Examples

```
# Loading example data
data(oz)
spectra(oz) <- sr_no ~ ... ~ 350:2500

# Print wavelength information
wl(oz)
range(wl(oz))

# Manipulate wavelength information
wl(oz) <- wl(oz) + 1000
wl(oz)
```

Index