Package ‘spectrolab’

January 13, 2025

Type Package

Title Class and Methods for Spectral Data

Version 0.0.19

Date 2025-01-07

Description Input/Output, processing and visualization of spectra taken with different spectrometers, including SVC (Spectra Vista), ASD and PSR (Spectral Evolution). Implements an S3 class spectra that other packages can build on. Provides methods to access, plot, manipulate, splice sensor overlap, vector normalize and smooth spectra.

License GPL-3

URL https://CRAN.R-project.org/package=spectrolab

Encoding UTF-8

LazyData true

Depends R (>= 4.0), stats

Suggests covr, tinytex, knitr (>= 1.30), rmarkdown (>= 2.5), testthat (>= 3.0.0)

VignetteBuilder knitr

Imports grDevices, parallel, RColorBrewer (>= 1.0), shiny (>= 1.5.0), shinyjs (>= 1.1)

RoxygenNote 7.3.2

NeedsCompilation no

Author Jose Eduardo Meireles [aut, cre],
Anna K. Schweiger [aut],
Jeannine Cavender-Bares [aut]

Maintainer Jose Eduardo Meireles <`jemeireles@gmail.com`>

Repository CRAN

Date/Publication 2025-01-13 17:50:02 UTC

Contents

`aggregate.spectra` *Aggregate spectra*

Description

Applies FUN (and FUN_meta) over spectra aggregating by factor ‘by‘.

Usage

```
## S3 method for class 'spectra'
aggregate(x, by, FUN, FUN_meta = NULL, ...)
```

Arguments

`x`	spectra object
`by`	vector of factors to guide the aggregation
`FUN`	function to be applied to value (and meta if FUN_meta is NULL)
`FUN_meta`	function to be applied to metadata. If NULL (default), same FUN applied to value is used.
`...`	extra args to FUN

Details

Argument FUN_meta is useful if you want to apply a different function to metadata and value. If you want to aggregate spectra and metadata using ‘mean‘, ‘sd‘, ‘median‘ etc. but try to keep the text values, wrap your function in `try_keep_txt(f)`.

Value

spectra object

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
spec_mean = aggregate(spec, by = names(spec), mean, try_keep_txt(mean))
```

`apply_by_band` *Apply numeric function by band*

Description

`apply_by_band` is conceptually similar to apply(as.matrix(x), 2, fun), but returns a spectra object while dealing with metadata and attributes. Applying a function that does not act on numeric values may crash the function or render all values NA.

Usage

```
apply_by_band(x, fun, na.rm = TRUE, keep_txt_meta = TRUE, name = NULL, ...)

## S3 method for class 'spectra'
apply_by_band(x, fun, na.rm = TRUE, keep_txt_meta = TRUE, name = NULL, ...)
```

Arguments

`x`	spectra
`fun`	numeric function to be applied to each band.
`na.rm`	boolean. remove NAs?
`keep_txt_meta`	boolean. try to keep text in the metadata?
`name`	name for each sample in the output spectra. The default (NULL) will give samples sequential numeric names. Recycled if necessary.
`...`	extra arguments passed to fun

Value

spectra

Methods (by class)

- `apply_by_band(spectra)`: Apply a numeric function by band

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
spec_mean = apply_by_band(spec, mean)
```

`as.data.frame.spectra` *Convert spectra to data.frame*

Description

Returns a data.frame that includes sample names, metadata (if present) and value data. One advantage over as.matrix, is that the metadata are returned.

Usage

```
## S3 method for class 'spectra'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  fix_names = "none",
  metadata = TRUE,
  ...
)
```

Arguments

`x`	spectra object
`row.names`	does nothing. Here for compatibility with S3 generics
`optional`	does nothing. Here for compatibility with S3 generics
`fix_names`	Use make.names to normalize names? Pick one: "none" "row" "col" "both".
`metadata`	boolean. Include spectral metadata? Defaults to TRUE
`...`	extra parameters passed to the generic as_spectra

Value

data.frame with: sample_name, metadata (if any) and value.

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
df   = as.data.frame(spec, fix_names = "none")
```

`as.matrix.spectra` *Convert spectra to matrix*

Description

Convert spectra to matrix

Usage

```
## S3 method for class 'spectra'
as.matrix(x, fix_names = "none", ...)
```

Arguments

`x`	spectra object
`fix_names`	Use make.names to normalize names? Pick one: "none" "row" "col" "both".
`...`	does nothing. Here for compatibility with S3 generics

Value

matrix of spectral value. columns are bands and rows are samples

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
mat  = as.matrix(spec)
```

`as_spectra` *Convert matrix or data frame to spectra*

Description

Convert matrix or data frame to spectra

Usage

```
as_spectra(x, name_idx = NULL, meta_idxs = NULL)
```

Arguments

`x`	matrix or dataframe. Samples are in rows and bands in columns. Any data that are not the spectra themselves (labels or metadata) must have their column index included in ‘name_idx‘ or ‘meta_idxs‘.
`name_idx`	column index with sample names. Defaults to NULL. If NULL or 0, rownames(x) or a sequence of integers will be assigned as names.
`meta_idxs`	column indices with metadata (not name and not value). Defaults to NULL

Value

spectra object

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
as_spectra(spec_matrix_example, name_idx = 1)
```

`as_spectra.data.frame` *Convert data.frame to spectra*

Description

Convert data.frame to spectra

Usage

```
## S3 method for class 'data.frame'
as_spectra(x, name_idx = NULL, meta_idxs = NULL)
```

Arguments

`x`	data.frame
`name_idx`	column index with sample names. Defaults to NULL.
`meta_idxs`	column indices with metadata (not name and not value). Defaults to NULL

Value

spectra object

Author(s)

Jose Eduardo Meireles

`as_spectra.matrix` *Convert matrix to spectra*

Description

Convert matrix to spectra

Usage

```
## S3 method for class 'matrix'
as_spectra(x, name_idx = NULL, meta_idxs = NULL)
```

Arguments

`x`	matrix
`name_idx`	column index with sample names. Defaults to NULL
`meta_idxs`	column indices with metadata (not name and not value). Defaults to NULL

Value

spectra object

Author(s)

Jose Eduardo Meireles

`bands` *Get spectra band labels*

Description

bands returns a vector of band labels from spectra

Usage

```
bands(x, min = NULL, max = NULL, return_num = TRUE)

## S3 method for class 'spectra'
bands(x, min = NULL, max = NULL, return_num = TRUE)
```

Arguments

`x`	spectra object
`min`	= NULL
`max`	= NULL
`return_num`	boolean. return vector of numeric values (default). otherwise, a vector of strings is returned

Value

vector of bands. numeric if ‘return_num‘ = TRUE (default).

Methods (by class)

- `bands(spectra)`: Get spectra band labels

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
head(bands(spec))
```

`bands<-` *Set band labels*

Description

`bands` sets band labels of lhs to the rhs values

Usage

```
bands(x) <- value
```

Arguments

`x`	spectra object (lhs)
`value`	rhs

Value

nothing. called for its side effect.

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
bands(spec) = bands(spec) / 1000
```

`combine` *Combine spectral datasets*

Description

`combine` binds two spectral datasets. Both spectra must have the very same band labels, but different metadata are acceptable

Usage

```
combine(s1, s2)

## S3 method for class 'spectra'
combine(s1, s2)
```

Arguments

`s1`	spectra object 1
`s2`	spectra object 2

Value

combined spectra object

Methods (by class)

- `combine(spectra)`: Combines two spectral datasets

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)

# Create dummy spectra datasets. Pretend that these are all different...
s1 = as_spectra(spec_matrix_example, name_idx = 1)
s2 = as_spectra(spec_matrix_example, name_idx = 1)
s3 = as_spectra(spec_matrix_example, name_idx = 1)

# combine 2 spectra objects
s_1and2 = combine(s1, s2)

# combine n spectra objects using the `Reduce` function
s_n = Reduce(combine, list(s1, s2, s3))
```

`default_spec_regions` *Return default spectral regions matrix*

Description

Return default spectral regions matrix

Usage

```
default_spec_regions()
```

Value

matrix with default_spec_regions

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
# matrix that defines regions on the spectra
# Useful for plotting w/ plot_regions()
```

`dim.spectra` *Get dimension of spectra*

Description

`dim` returns a vector with number of samples and bands (bands)

Usage

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

Arguments

`x`	spectra object

Value

tuple of integers: c("n_samples", "n_bands")

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
dim(spec)
```

`guess_splice_at` *Guess splice bands (bounds between senors)*

Description

Guess splice bands (bounds between senors)

Usage

```
guess_splice_at(x)

## S3 method for class 'spectra'
guess_splice_at(x)
```

Arguments

`x`	spectra object

Value

vector of band values

Methods (by class)

- `guess_splice_at(spectra)`: Guess splice bands (bounds between senors)

Author(s)

Jose Eduardo Meireles

`is_spectra` *Is it a spectra object?*

Description

`is_spectra` tests if the argument is a spectra class object

Usage

```
is_spectra(x)
```

Arguments

`x`	any object

Value

boolean

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec  = as_spectra(spec_matrix_example, name_idx = 1)
spec1 = unclass(spec)
is_spectra(spec)
is_spectra(spec1)
```

`make_fwhm` *Resample the FWHM to a new set of bands using a gaussian model*

Description

Resample the FWHM to a new set of bands using a gaussian model

Usage

```
make_fwhm(spec, new_bands, new_fwhm = NULL, return_type = "max", k = 3)
```

Arguments

`spec`	spectra object
`new_bands`	band values to resample the spectra to
`new_fwhm`	FWHM for the new bands
`return_type`	either "max" or "old". If "old" (default), it returns the fwhm inferred from the original's spectra bands. If max (default), it returns the max between the new and old fwhm.
`k`	number of unique FHWM to estimate

Value

FWHM as a numeric vector

`match_sensors` *Match spectra at sensor transitions*

Description

`match_sensors` scales values of sensors 1 (VIS) and 3 (SWIR 2)

Usage

```
match_sensors(x, splice_at, fixed_sensor = 2, interpolate_wvl = c(5, 1))

## S3 method for class 'spectra'
match_sensors(x, splice_at, fixed_sensor = 2, interpolate_wvl = c(5, 2))
```

Arguments

`x`	spectra object
`splice_at`	bands that serve as splice points, i.e the beginnings of the rightmost sensor. Must be length 1 or 2 (max 3 sensors)
`fixed_sensor`	sensor to keep fixed. Can be 1 or 2 if matching 2 sensors. If matching 3 sensors, ‘fixed_sensor‘ must be 2 (default).
`interpolate_wvl`	extent around splice_at values over which the splicing factors will be calculated. Defaults to 5

Details

Splice_at has no default because sensor transition points vary between vendors and individual instruments. The function `guess_splice_at` can help you guess what those values could be. However, `splice_at` is an important parameter though, so you should visually inspect your spectra before assigning it. Typical values in our own individual instruments were: SVC ~ c(990, 1900), ASD ~ c(1001, 1801).

If the factors used to match spectra are unreasonable, `match_sensors` will throw. Unreasonable factors (f) are defined as 0.5 > f > 3 or NaN, which happens when the value for the right sensor is 0.

Value

spectra object

Methods (by class)

- `match_sensors(spectra)`: Match sensor overlap regions

Author(s)

Jose Eduardo Meireles and Anna Schweiger

`max.spectra` *Maximum value*

Description

`max` Returns the maximum value in a spectra object

Usage

```
## S3 method for class 'spectra'
max(..., na.rm = FALSE)
```

Arguments

`...`	spectra object
`na.rm`	boolean. remove NAs? Defaults to FALSE

Value

single numeric value

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
max(spec)
```

`mean.spectra` *Mean spectrum*

Description

`mean` computes the arithmetic mean spectrum.

Usage

```
## S3 method for class 'spectra'
mean(x, na.rm = TRUE, keep_txt_meta = TRUE, ...)
```

Arguments

`x`	spectra
`na.rm`	boolean. remove NAs? Defaults to TRUE
`keep_txt_meta`	try to keep text in the metadata
`...`	nothing

Value

single spectrum

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
mean(spec)
```

`median.spectra` *Median spectrum*

Description

`median` computes the median spectrum

Usage

```
## S3 method for class 'spectra'
median(x, na.rm = TRUE, keep_txt_meta = TRUE, ...)
```

Arguments

`x`	spectra
`na.rm`	boolean. remove NAs? Defaults to TRUE
`keep_txt_meta`	try to keep text in the metadata
`...`	nothing

Value

single spectrum

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
median(spec)
```

`meta` *Get metadata*

Description

`meta` returns metadata of spectra

Usage

```
meta(x, label = NULL, sample = NULL, simplify = FALSE, quiet = TRUE)

## S3 method for class 'spectra'
meta(x, label = NULL, sample = NULL, simplify = FALSE, quiet = TRUE)
```

Arguments

`x`	spectra object
`label`	metadata column index or label
`sample`	sample index or name
`simplify`	boolean. defaults to FALSE
`quiet`	boolean. warn about non-existent metadata? defaults to TRUE

Value

data frame or vector

Methods (by class)

- `meta(spectra)`: get metadata

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
spec = normalize(spec)
meta(spec, "normalization_magnitude")
```

`meta<-` *Set metadata*

Description

`meta` sets metadata

Usage

```
meta(x, label = NULL, sample = NULL) <- value
```

Arguments

`x`	spectra object (lhs)
`label`	metadata column label
`sample`	sample name
`value`	rhs. TODO

Value

nothing. called for its side effect

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
meta(spec, "random") = rnorm(nrow(spec), mean(10), sd = 2)
```

`min.spectra` *Minimum value*

Description

`min` Returns the minimum value in a spectra object

Usage

```
## S3 method for class 'spectra'
min(..., na.rm = FALSE)
```

Arguments

`...`	spectra object
`na.rm`	boolean. remove NAs? Defaults to FALSE

Value

single numeric value

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
min(spec)
```

`names.spectra` *Get spectra sample names*

Description

`names` returns a vector of sample names

Usage

```
## S3 method for class 'spectra'
names(x)
```

Arguments

`x`	spectra object

Value

vector of sample names

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
names(spec)
```

`names<-.spectra` *Set spectra sample names*

Description

`names` assigns sample names to lhs

Usage

```
## S3 replacement method for class 'spectra'
names(x) <- value
```

Arguments

`x`	spectra object (lhs)
`value`	values to be assigned (rhs)

Value

nothing. called for its side effect.

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
names(spec) = toupper(names(spec))
```

`normalize` *Vector normalize spectra*

Description

`normalize` returns a spectra obj with vector normalized values. Normalization value for each spectrum computed as sqrt(sum(x^2))

Usage

```
normalize(x, quiet = FALSE, ...)

## S3 method for class 'spectra'
normalize(x, quiet = FALSE, ...)
```

Arguments

`x`	spectra object. bands must be strictly increasing
`quiet`	boolean. Warn about change in y value units? Defaults to FALSE
`...`	nothing

Value

spectra object with normalized spectra

Methods (by class)

- `normalize(spectra)`: Vector normalize spectra

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
spec = normalize(spec)
```

`Ops.spectra` *Arithmetic operators for spectra*

Description

Overloads arithmetic operators for spectra using ‘Ops.‘

Usage

```
## S3 method for class 'spectra'
Ops(e1, e2)
```

Arguments

`e1`	lhs
`e2`	rhs

Value

Depends on the operator. math operators will return spectra and logical or comparison operators will return boolean matrices

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec  = as_spectra(spec_matrix_example, name_idx = 1)
spec1 = spec * 2
spec2 = spec + spec
all(spec1 == spec2)
```

`pairwise_indices` *Pairwise indices*

Description

`pairwise_indices` computes pairwise spectral indices. Indices are computed as (a - b) / (a + b) where a is the lower band. The column names of the resulting matrix are given as "a|b".

Usage

```
pairwise_indices(x, max_out_elements = 5e+08)
```

Arguments

`x`	spectra
`max_out_elements`	maximum number of elements in the output object

Value

list that includes the *indices* between bands a and b (column names a|b) and the pairwise *band_combinations*

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec  = as_spectra(spec_matrix_example, name_idx = 1)

# Resampling spectra since a spectral dataset with 2,001 bands
# results in 2,001,000 unique spectral indices per sample
new_bands = seq(400, 2400, 10)
spec = resample(spec, new_bands, make_fwhm(spec, new_bands) )
p_idx = pairwise_indices(spec)
```

`plot.spectra` *Plot spectra*

Description

`plot` plots spectra.

Usage

```
## S3 method for class 'spectra'
plot(x, ylab = "value", xlab = "band", col = "black", lty = 1, type = "l", ...)
```

Arguments

`x`	spectra object
`ylab`	label for y axis. Defaults to "value".
`xlab`	label for x axis. Defaults to "band".
`col`	line color. Defaults to "black".
`lty`	line type. Defaults to 1.
`type`	type of plot. Meant to take either line "l" or no plotting "n".
`...`	other arguments passed to matplot.

Value

nothing. Called for side effect.

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec  = as_spectra(spec_matrix_example, name_idx = 1)
plot(spec, lwd = 1.2)
```

`plot_interactive` *Plot spectra interactively*

Description

Interactively plots spectra with a shiny app. Useful to inspect large datasets.

Usage

```
plot_interactive(
  spec,
  colpalette = function(n) RColorBrewer::brewer.pal(n, "Dark2"),
  ...
)
```

Arguments

`spec`	spectra object
`colpalette`	a color palette function, e.g. rainbow, terrain.colors, or a function returned by colorRampPalette() or colorRamps package
`...`	Other arguments passed to plot

Details

`plot_interact` limits the number of spectra displayed at once to 600 for performance reasons. As of now, the function does not return anything and does not have side effects. This means that spectra can be selected and highlighted but not yet deleted or subset from the shiny app.

Value

interactive plot

Author(s)

Jose Eduardo Meireles and Anna K. Schweiger

Examples

```
if(interactive()){
# Create a spectra object
spec = as_spectra(spec_matrix_example, name_idx = 1)

# Start interactive plot
plot_interactive(spec)
}
```

`plot_quantile` *Plot spectra quantiles*

Description

`plot_quantile` plots polygons for the quantiles of spectra per band.

Usage

```
plot_quantile(
  spec,
  total_prob = 0.95,
  col = rgb(0, 0, 0, 0.1),
  border = TRUE,
  add = FALSE,
  na.rm = TRUE,
  ...
)
```

Arguments

`spec`	spectra object
`total_prob`	total probability mass to encompass. Single number between 0.0 and 1.0. Defaults to 0.95.
`col`	polygon color
`border`	boolean. Draw border?
`add`	if add = FALSE (default), a new plot is created. Otherwise (add = TRUE), the quantile is added to the current plot.
`na.rm`	boolean. remove NAs to compute quantiles? Defaults to TRUE
`...`	other parameters passed to polygon() or to plot.

Value

nothing. Called for its side effect.

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec  = as_spectra(spec_matrix_example, name_idx = 1)
plot_quantile(spec, total_prob = 0.5)
```

`plot_regions` *Plot polygons for spectral regions*

Description

`plot_regions` plots polygons for default (VIS, NIR, SWIR 1, SWIR 2) or customized regions of the spectrum.

Usage

```
plot_regions(
  spec,
  regions = default_spec_regions(),
  col = grDevices::rgb(0.7, 0.7, 0.7, 0.3),
  border = FALSE,
  add = TRUE,
  add_label = TRUE,
  cex_label = 1,
  ...
)
```

Arguments

`spec`	spectra object
`regions`	matrix with spectral regions in columns and only two rows named "begin" and "end". Values are the bands where a spectral regions begins and ends. See details for how the default regions are defined.
`col`	color for regions. Single value or vector of length ncol (regions).
`border`	color for region borders. Defaults to FALSE (no border).
`add`	boolean. If TRUE (default) adds polygons to current plot (if a plot exists) or throws an error if a plot does not exist. If FALSE, a new plot is created **without** any spectra.
`add_label`	boolean. Add region column names on top of the polygons?
`cex_label`	label scale
`...`	additional parameters passed to polygon().

Details

Default regions: spec_regions = cbind("VIS" = c(begin = 400, end = 700), "NIR" = c(begin = 800, end = 1300), "SWIR1" = c(begin = 1550, end = 1800), "SWIR2" = c(begin = 2000, end = 2400)).

Value

nothing. Called for its side effect.

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
plot_regions(spec, default_spec_regions())
plot(spec, add = TRUE)

# Alternatively, if you want to get fancy...
col_fun = colorRampPalette(c(rgb(1, 1, 0, 0.7),rgb(1, 0, 0, 0.7)), alpha = TRUE)
colors = col_fun(4)

plot_regions(spec,default_spec_regions(), col = colors)
plot(spec, add = TRUE)
```

`print.spectra` *Print spectra*

Description

`print` prints basic information about the spectra obj to the console

Usage

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

Arguments

`x`	spectra object
`...`	other arguments passed to print. not implemented for spectra

Value

nothing. called for side effect

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
print(spec)
## or simply
spec
```

`quantile.spectra` *Compute spectra quantiles*

Description

`quantile` computes quantiles by band and returns them as ‘spectra‘.

Usage

```
## S3 method for class 'spectra'
quantile(
  x,
  probs = c(0.025, 0.25, 0.5, 0.75, 0.975),
  na.rm = TRUE,
  names = NULL,
  ...
)
```

Arguments

`x`	spectra object. Must have at least the same number of sample that length(probs) has.
`probs`	Probabilities to compute quantiles. Must be a vector of numerics between 0.0 and 1.0. Defaults to c(0.025, 0.25, 0.5, 0.75, 0.975). Duplicated probs will be removed.
`na.rm`	remove NAs before computing quantiles? Defaults to TRUE
`names`	names for each quantile spectrum. If NULL (default), names are set to ‘probs‘. A char vector should otherwise be given. Recycled.
`...`	other arguments passed to quantile.

Value

spectra object with one spectrum for each prob

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
quantile(spec, probs = c(0.25, 0.75))
```

`range.spectra` *Range of spectral values*

Description

`range` Returns the range of (min, max) values in spectra

Usage

```
## S3 method for class 'spectra'
range(..., na.rm = FALSE)
```

Arguments

`...`	spectra object
`na.rm`	boolean. remove NAs? Defaults to FALSE

Value

tuple of numeric values (min, max)

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
range(spec)
```

`read_spectra` *Read files from various formats into ‘spectra‘*

Description

Read files from various formats into ‘spectra‘

Usage

```
read_spectra(
  path,
  format = NULL,
  type = "target_reflectance",
  extract_metadata = FALSE,
  exclude_if_matches = NULL,
  ignore_extension = FALSE
)
```

Arguments

`path`	Path to directory or input files.
`format`	File format. Defaults to NULL so spectrolab tries to guess it from the file name. Alternatively, use "asd" for ASD; "sig" for SVC (Spectra Vista); or "sed" for PSR (Spectral Evolution)
`type`	Data type to read. "target_reflectance", "target_radiance", or "reference_radiance". Defaults to "target_reflectance".
`extract_metadata`	Boolean. Defaults to FALSE. Only implemented for the Spectra Vista (.sig) and Spectral Evolution (.sed) file types.
`exclude_if_matches`	excludes files that match this regular expression. Example: "BAD"
`ignore_extension`	Boolean. If TRUE, the parser will try to read every file in path regardless of the expected extension.

Value

a single ‘spectra‘ or a list of ‘spectra‘ (in case files have incompatible band number or bands values)

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
dir_path = system.file("extdata", "Acer_example", package = "spectrolab")

spec     = read_spectra(path = dir_path, format = "sig")
```

`resample` *Resample spectra*

Description

Resample spectra

Usage

```
resample(spec, new_bands, fwhm)
```

Arguments

`spec`	spectra object
`new_bands`	band values to resample the spectra to
`fwhm`	FWHM for the new bands

Value

resampled spectra

`sd` *Standard deviation*

Description

`sd` computes the standard deviation spectrum. Note that values will not reflect value anymore, but the sd of the value instead.

Usage

```
sd(x, na.rm = FALSE)
```

Arguments

`x`	a numeric vector or an R object which is coercible to one by as.double(x)
`na.rm`	logical. Should missing values be removed?

Value

standard deviation

`sd.default` *Default standard deviation*

Description

`sd` computes the standard deviation of the values in x. If na.rm is TRUE then missing values are removed before computation proceeds.

Usage

```
## Default S3 method:
sd(x, na.rm = FALSE)
```

Arguments

`x`	a numeric vector or an R object which is coercible to one by as.double(x)
`na.rm`	logical. Should missing values be removed?

Value

standard deviation of x

Examples

```
x = rnorm(n = 200, mean = 0, sd = 1)
sd(x)
```

`sd.spectra` *Standard deviation spectrum*

Description

Forces keep_txt_meta = TRUE

Usage

```
## S3 method for class 'spectra'
sd(x, na.rm = TRUE)
```

Arguments

`x`	spectra
`na.rm`	boolean. remove NAs?

Value

single spectrum

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
sd(spec)
```

`smooth` *Generic Smoothing function*

Description

Generic Smoothing function

Usage

```
smooth(x, ...)
```

Arguments

`x`	data to smooth over
`...`	additional arguments

Value

smoothed data

`smooth.default` *Default smoothing function*

Description

Default smoothing function

Usage

```
## Default S3 method:
smooth(x, ...)
```

Arguments

`x`	data to smooth over
`...`	additional arguments

Value

smoothed data

`smooth.spectra` *Smooth spectra*

Description

`smooth` runs each spectrum by a smoothing and returns the spectra

Usage

```
## S3 method for class 'spectra'
smooth(x, method = "gaussian", ...)
```

Arguments

`x`	spectra object. bands must be strictly increasing
`method`	Choose smoothing method: "gaussian" (default), "spline", or "moving_average"
`...`	additional parameters passed to methods `smooth_fwhm`, `smooth_spline`, `smooth_moving_avg`

Value

a spectra object of with smoothed spectra

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)

spec = as_spectra(spec_matrix_example, name_idx = 1)
spec = smooth(spec)
```

`smooth_fwhm` *Smooth spectra with a gaussian model*

Description

Smooth spectra with a gaussian model

Usage

```
smooth_fwhm(x, fwhm = NULL)
```

Arguments

`x`	spectra
`fwhm`	Full Width at Half Maximum.

Value

smoothed spectra

`smooth_moving_avg` *Smooth moving average for spectra*

Description

Smooth moving average for spectra

Usage

```
smooth_moving_avg(x, n = NULL, save_bands_to_meta = TRUE)
```

Arguments

`x`	spectra object
`n`	= NULL
`save_bands_to_meta`	boolean. keep lost ends of original wvls in metadata

Value

spectra object

Author(s)

Jose Eduardo Meireles

`smooth_spline` *Smooth spline functions for spectra*

Description

Gets spline functions for each spectrum in a spectra object.

Usage

```
smooth_spline(x, parallel = TRUE, return_fn = FALSE, ...)
```

Arguments

`x`	spectra object. bands must be strictly increasing
`parallel`	boolean. Do computation in parallel? Defaults to TRUE. Unfortunately, the parallelization does not work on Windows.
`return_fn`	Boolean. If TRUE, `smooth_spline` returns the spline functions instead of the smoothed spectra. Defaults to FALSE
`...`	additional parameters passed to smooth.spline except nknots, which is computed internally

Value

Smoothed spectra or, if return_fn = TRUE, a list of spline functions.

Author(s)

Jose Eduardo Meireles

`spectra` *Spectra object constructor*

Description

`spectra` "manually" creates a spectra object

Usage

```
spectra(value, bands, names, meta = NULL, ...)
```

Arguments

`value`	N by M numeric matrix. N samples in rows and M bands in columns
`bands`	band names in vector of length M
`names`	sample names in vector of length N
`meta`	spectra metadata. defaults to NULL. Must be either of length or nrow equals to the number of samples (nrow(value) or length(names))
`...`	additional arguments to metadata creation. not implemented yet

Value

spectra object

Note

This function resorts to an ugly hack to deal with metadata assignment. Need to think a little harder to find a solution.

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
# 1. Create a value matrix.
#    In this case, by removing the first column that holds the species name
rf = spec_matrix_example[ , -1]

# (2) Create a vector with band labels that match
#     the value matrix columns.
wl = colnames(rf)

# (3) Create a vector with sample labels that match
#     the value matrix rows.
#     In this case, use the first colum of spec_matrix_example
sn = spec_matrix_example[ , 1]

# Finally, construct the spectra object using the `spectra` constructor
spec = spectra(value = rf, bands = wl, names = sn)
```

`spec_matrix_example` *Example spectral dataset*

Description

Simulated spectral dataset as a matrix. First column hold species names and the remaining ones store the spectra values. band labels are given as column names

Usage

```
spec_matrix_example
```

Format

An object of class `matrix` (inherits from `array`) with 50 rows and 2102 columns.

Author(s)

Jose Eduardo Meireles

`split.spectra` *Split spectra*

Description

`split` a spectra object into a list of spectra according to grouping f.

Usage

```
## S3 method for class 'spectra'
split(x, f, drop = FALSE, ...)
```

Arguments

`x`	spectra object
`f`	factor vector defining the grouping. Must have length nrow(x)
`drop`	NOT used
`...`	NOT used

Value

list of spectra

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
spec_list = split(spec, names(spec))
```

`str.spectra` *Structure of the spectra object*

Description

Structure of the spectra object

Usage

```
## S3 method for class 'spectra'
str(object, ...)
```

Arguments

`object`	spectra object
`...`	additional args. not implemented

Value

prints to console

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
str(spec)
```

`subset_by` *Subset spectra by factor*

Description

`subset_by` subsets spectra by a factor ‘by‘ ensuring that it appears at most ‘n_max‘ times **and** at least ‘n_min‘ times in the dataset.

Usage

```
subset_by(x, by, n_min, n_max, random = TRUE)

## S3 method for class 'spectra'
subset_by(x, by, n_min, n_max, random = TRUE)
```

Arguments

`x`	spectra object
`by`	vector coercible to factor and of same length as nrow(x)
`n_min`	int. only keep spectra with at least (inclusive) ‘n_min‘ number of samples per unique ‘by‘.
`n_max`	int. keep at most (incl) this number of spectra per unique ‘by‘
`random`	boolean. Sample randomly or keep first n_max? Defaults to TRUE

Details

Note that `subset_by` forces you to provide both a minimum and a maximum number of spectra to be kept for each unique value of ‘by‘. In case you’re interested in subsetting *only* based on ‘n_min‘, set ‘n_max‘ to ‘Inf‘.

Value

spectra

Methods (by class)

- `subset_by(spectra)`: Subset spectra by factor

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)

# remove spec of species with less than 4 samples
spec = subset_by(spec, by = names(spec), n_min = 4, n_max = Inf)
```

`summary.spectra` *Summarize spectra*

Description

Summarize spectra

Usage

```
## S3 method for class 'spectra'
summary(object, ...)
```

Arguments

`object`	spectra object
`...`	additional params to summary. not used yet

Value

nothing yet (just prints to console)

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
summary(spec)
```

`t.spectra` *Spectra Transpose*

Description

spectra are not transposable. Transpose the value instead

Usage

```
## S3 method for class 'spectra'
t(x)
```

Arguments

`x`	spectra

Value

No return value. Operation not allowed

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
s = as_spectra(spec_matrix_example, name_idx = 1)

t(value(s))
t(as.matrix(s))
```

`try_keep_txt` *Wrap function to try to keep text*

Description

Function operator returning a function f that tries to keep text.

Usage

```
try_keep_txt(f)
```

Arguments

`f`	function to be applied

Details

`try_keep_txt` takes a function f as argument, typically a mathematical operation such as mean, median, etc. and returns a modified version of it that will try return a string of unique values in case function f emits a warning. Useful when aggregating over spectral metadata that has both numeric values (which you want to aggregate) and text values, which you want to keep.

Value

modified function f (f').

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
g = try_keep_txt(mean)
g(c(1, 2))
g(c("a", "b"))
```

`value` *Get spectra value*

Description

`value` returns the value matrix from spectra

Usage

```
value(x)

## S3 method for class 'spectra'
value(x)
```

Arguments

`x`	spectra object

Value

matrix with samples in rows and bands in columns

Methods (by class)

- `value(spectra)`: Get spectra value

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
is.matrix(value(spec))
```

`value<-` *Set spectra value*

Description

`value<-` Assigns the rhs to the value of the lhs spectra obj

Usage

```
value(x) <- value
```

Arguments

`x`	spectra object
`value`	value to be assigned to the lhs

Value

nothing. called for its side effect

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
# scale all refletance values by 2
value(spec) = value(spec) * 2
```

`var` *Variance*

Description

`var` computes the variance spectrum. Note that values will not reflect value anymore, but the variance of the value instead.

Usage

```
var(x, y = NULL, na.rm = FALSE, use)
```

Arguments

`x`	a numeric vector, matrix or data frame
`y`	NULL (default) or a vector, matrix or data frame with compatible dimensions to x.
`na.rm`	logical. Should missing values be removed?
`use`	an optional character string giving a method for computing covariances in the presence of missing values. This must be (an abbreviation of) one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs"

Value

variance

`var.default` *Variance*

Description

`var` computes the variance spectrum. Note that values will not reflect value anymore, but the variance of the value instead.

Usage

```
## Default S3 method:
var(x, y = NULL, na.rm = FALSE, use)
```

Arguments

`x`	a numeric vector, matrix or data frame
`y`	NULL (default) or a vector, matrix or data frame with compatible dimensions to x.
`na.rm`	logical. Should missing values be removed?
`use`	an optional character string giving a method for computing covariances in the presence of missing values. This must be (an abbreviation of) one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs"

Value

variance

`var.spectra` *Variance spectrum*

Description

Forces keep_txt_meta = TRUE

Usage

```
## S3 method for class 'spectra'
var(x, y = NULL, na.rm = TRUE, use)
```

Arguments

`x`	spectra
`y`	nothing
`na.rm`	boolean. remove NAs?
`use`	nothing

Value

single spectrum

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
var(spec)
```

`[.spectra` *Subset spectra*

Description

`` `[` `` Subsets spectra by sample names (rows) or (and) bands (columns)

Usage

```
## S3 method for class 'spectra'
x[i, j, simplify = TRUE]
```

Arguments

`x`	spectra object
`i`	Sample names (preferred), index, or a logical vector of length nrow(x)
`j`	band labels, as numeric or character or a logical vector of length ncol(x). Do not use indexes!
`simplify`	Boolean. If TRUE (default), single band selections are returned as a named vector of values

Details

Subset operations based on samples (first argument) will match sample names or indexes, in that order. The spectra constructor ensures that names are not numeric nor are coercible to numeric, such that x[1:2,] will return the first and second samples in the ‘spectra‘ object. Subsetting based on bands (second argument) matches the band labels, not indices! That is, x[, 600] will give you the value data for the 600nm band and not the 600th band. Boolean vectors of the appropriate length can be used to subset samples and bands.

Value

usually a spectra object, but see param ‘simplify‘

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
head(names(spec), n = 3)
# by name
spec1 = spec[ "species_7" , ]
spec1
# by band
spec2 = spec[ , 400:700 ]
spec2
```

`[<-.spectra` *Assign values to spectra*

Description

``[<-`` assigns the rhs values to spectra

Usage

```
## S3 replacement method for class 'spectra'
x[i, j] <- value
```

Arguments

`x`	spectra object (lhs)
`i`	Sample names (preferred), index, or a logical vector of length nrow(x)
`j`	band labels, as numeric or character or a logical vector of length ncol(x). Do not use indexes!
`value`	value to be assigned (rhs). Must either data coercible to numeric or another ‘spectra‘ obj

Value

nothing. modifies spectra as side effect

Author(s)

Jose Eduardo Meireles

Examples

```
library(spectrolab)
spec = as_spectra(spec_matrix_example, name_idx = 1)
spec[ , 400:500] = spec[ , 400:500] * 1.2
spec
```

Index