

# Package ‘spectrolab’

October 28, 2018

**Type** Package

**Title** Class and Methods for Hyperspectral Data

**Version** 0.0.8

**Date** 2018-10-28

**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://github.com/meireles/spectrolab>

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.1.0

**Depends** R (>= 3.4), stats

**Suggests** knitr (>= 1.16), rmarkdown (>= 1.5), testthat (>= 1.0.2)

**VignetteBuilder** knitr

**Imports** prospectr, grDevices, parallel, RColorBrewer, shiny, shinyjs, devtools, usethis

**NeedsCompilation** no

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

**Maintainer** Jose Eduardo Meireles <meireles@umn.edu>

**Repository** CRAN

**Date/Publication** 2018-10-28 15:00:03 UTC

**R topics documented:**

aggregate.spectra . . . . .	3
apply_by_band . . . . .	4
as.data.frame.spectra . . . . .	5
as.matrix.spectra . . . . .	6
as.spectra . . . . .	6
as.spectra.data.frame . . . . .	7
as.spectra.matrix . . . . .	8
combine . . . . .	8
default_spec_regions . . . . .	9
dim.spectra . . . . .	10
has_nir_dip . . . . .	10
is_spectra . . . . .	11
match_sensors . . . . .	12
max.spectra . . . . .	13
mean.spectra . . . . .	14
median.spectra . . . . .	14
meta . . . . .	15
meta<- . . . . .	16
min.spectra . . . . .	17
names.spectra . . . . .	17
names<- .spectra . . . . .	18
normalize . . . . .	19
Ops.spectra . . . . .	20
plot.spectra . . . . .	20
plot_interactive . . . . .	21
plot_quantile . . . . .	22
plot_regions . . . . .	23
print.spectra . . . . .	24
quantile.spectra . . . . .	25
range.spectra . . . . .	26
ratio . . . . .	26
read_spectra . . . . .	27
reflectance . . . . .	28
reflectance<- . . . . .	29
resample . . . . .	29
sd . . . . .	30
sd.default . . . . .	31
sd.spectra . . . . .	32
smooth . . . . .	32
smooth.default . . . . .	33
smooth.spectra . . . . .	33
spectra . . . . .	34
spectrolab . . . . .	35
spec_matrix_example . . . . .	35
spec_with_jump . . . . .	36
split.spectra . . . . .	36

str.spectra . . . . .	37
subset_by . . . . .	38
summary.spectra . . . . .	39
t.spectra . . . . .	40
try_keep_txt . . . . .	40
var . . . . .	41
var.default . . . . .	42
var.spectra . . . . .	43
wavelengths . . . . .	43
wavelengths<- . . . . .	44
[.spectra . . . . .	45
[<-.spectra . . . . .	46

**Index** **48**

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 refl (and meta if FUN\_meta is NULL)
- FUN\_meta              function to be applied to metadata. If NULL (default), same FUN applied to reflectance is used.
- ...                    extra args to FUN

**Details**

Argument FUN\_meta is useful if you want to apply a different function to metadata and reflectance. 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	<i>Apply numeric function by band</i>
---------------	---------------------------------------

---

**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 reflectance 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)**

- spectra: Apply a numeric function by band

**Author(s)**

Jose Eduardo Meireles  
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 reflectance 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 reflectance.

**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 reflectance. columns are wavelengths 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_idx = NULL)
```

**Arguments**

x	matrix or dataframe. Samples are in rows and wavelengths in columns. Any data that are not the spectra themselves (labels or metadata) must have their column index included in 'name_idx' or 'meta_idx'.
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_idx	column indices with metadata (not name and not reflectance). 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_idx = NULL)
```

**Arguments**

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

**Value**

spectra object

**Author(s)**

Jose Eduardo Meireles

---

as.spectra.matrix	<i>Convert matrix to spectra</i>
-------------------	----------------------------------

---

**Description**

Convert matrix to spectra

**Usage**

```
## S3 method for class 'matrix'  
as.spectra(x, name_idx = NULL, meta_idx = NULL)
```

**Arguments**

x	matrix
name_idx	column index with sample names. Defaults to NULL
meta_idx	column indices with metadata (not name and not reflectance). Defaults to NULL

**Value**

spectra object

**Author(s)**

Jose Eduardo Meireles

---

combine	<i>Combine spectral datasets</i>
---------	----------------------------------

---

**Description**

combine binds two spectral datasets. Both spectra must have the very same wavelength 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)**

- 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	<i>Get dimension of spectra</i>
-------------	---------------------------------

---

**Description**

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

**Usage**

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

**Arguments**

x                    spectra object

**Value**

tuple of integers: c("n\_samples", "n\_wavelengths")

**Author(s)**

Jose Eduardo Meireles

**Examples**

```
library(spectrolab)
spec = as.spectra(spec_matrix_example, name_idx = 1)
dim(spec)
```

---

has_nir_dip	<i>Has a dip in the NIR region?</i>
-------------	-------------------------------------

---

**Description**

has\_nir\_dip tests if spectra have a NIR dip

**Usage**

```
has_nir_dip(x, smooth = TRUE, ...)
```

**Arguments**

x                    spectra object  
smooth                boolean. Smooth spectra internally? Defaults to TRUE  
...                    additional parameters passed to 'spectrolab::smooth' (method spline)

**Details**

Because the NIR dip is detected using features on the second derivative, high frequency noise in the spectra is likely to spoil the method and give false positives. Ideally, spectra should have been smoothed ('spectrolab::smooth') before trying to detect an NIR dip. If param 'smooth' is set to TRUE (default), 'has\_nir\_dip' does some basic smoothing internally.

**Value**

boolean vector

**Author(s)**

Jose Eduardo Meireles

**Examples**

```
library(spectrolab)
spec = as.spectra(spec_matrix_example, name_idx = 1)

# Find spec with NIR dip
# Set parallel = TRUE if many samples need testing.
dip = has_nir_dip(spec, parallel = FALSE)

# remove NIR dip
spec = spec[ ! dip, ]
```

---

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)
```

---

match_sensors	<i>Match spectra at sensor transitions</i>
---------------	--

---

**Description**

match\_sensors scales reflectance values of sensors 1 (vis) and 3 (swir2)

**Usage**

```
match_sensors(x, splice_at, fixed_sensor = 2, interpolate_wvl = 5,
  factor_range = c(0.5, 3))
```

```
## S3 method for class 'spectra'
match_sensors(x, splice_at, fixed_sensor = 2,
  interpolate_wvl = 5, factor_range = c(0.5, 3))
```

**Arguments**

x	spectra object
splice_at	wavelengths 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
factor_range	range of acceptable correction factors (min, max). Defaults to c(0.5, 3)

**Details**

Splice\_at has no default because sensor transition points vary between vendors and individual instruments. It 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 reflectance value for the right sensor is 0.

**Value**

spectra object

**Methods (by class)**

- spectra: Match sensor overlap regions

**Author(s)**

Jose Eduardo Meireles and Anna Schweiger

**Examples**

```
library(spectrolab)
data(spec_with_jump)
spec = spec_with_jump
spec = match_sensors(spec, splice_at = c(971, 1910), fixed_sensor = 2)
```

---

max.spectra	<i>Maximum reflectance</i>
-------------	----------------------------

---

**Description**

max Returns the maximum reflectance 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	<i>Mean spectrum</i>
--------------	----------------------

---

**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	<i>Median spectrum</i>
----------------	------------------------

---

**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, sample, 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)**

- 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, sample) <- 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	<i>Minimum reflectance</i>
-------------	----------------------------

---

**Description**

min Returns the minimum reflectance 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	<i>Get spectra sample names</i>
---------------	---------------------------------

---

**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)

**Details**

Sample names must not be coercible to numeric. That is, names such as "1" and "153.44" are invalid even if they are encoded as character. names will add the prefix "spec\_" to any element of value that is coercible to numeric.

**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	<i>Vector normalize spectra</i>
-----------	---------------------------------

---

**Description**

normalize returns a spectra obj with vector normalized reflectances

**Usage**

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

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

**Arguments**

x	spectra object. Wavelengths 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)**

- 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)
```

---

plot.spectra

*Plot spectra*

---

### Description

plot plots spectra.

### Usage

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

**Arguments**

x	spectra object
ylab	label for y axis. Defaults to "Reflectance".
xlab	label for x axis. Defaults to "Wavelength".
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)**

Anna K. Schweiger and Jose Eduardo Meireles

**Examples**

```
## Not run:
# Create a spectra object
spec = as.spectra(spec_matrix_example, name_idx = 1)

# Start interactive plot
plot_interactive(spec)

## End(Not run)
```

---

plot_quantile	<i>Plot spectra quantiles</i>
---------------	-------------------------------

---

**Description**

plot\_quantile plots polygons for the quantiles of spectra per wavelength.

**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	<i>Plot polygons for spectral regions</i>
--------------	---

---

**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 wavelengths 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 <b>**without**</b> 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...
## Not run:
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)

## End(Not run)
```

---

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 wavelength 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).
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	<i>Range of reflectance values</i>
---------------	------------------------------------

---

**Description**

range Returns the range of (min, max) reflectance 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)
```

---

ratio	<i>Pairwise reflectance ratios</i>
-------	------------------------------------

---

**Description**

ratio computes pairwise ratios between bands

**Usage**

```
ratio(x, simplify = FALSE)

## S3 method for class 'spectra'
ratio(x, simplify = FALSE)
```

**Arguments**

x	spectra
simplify	coerce to matrix or keep result as list

**Value**

list or matrix

**Methods (by class)**

- spectra: Compute pairwise reflectance ratios

**Author(s)**

Jose Eduardo Meireles

**Examples**

```
library(spectrolab)

# Ratios of visible part of the spectrum
spec = as.spectra(spec_matrix_example, name_idx = 1)[ , 400:700 ]
spec_ratio_mat = ratio(spec)
```

---

read_spectra	<i>Read files from various formats into 'spectra'</i>
--------------	---

---

**Description**

Read files from various formats into 'spectra'

**Usage**

```
read_spectra(path, format, recursive = FALSE,
             exclude_if_matches = NULL, ignore_extension = FALSE, ...)
```

**Arguments**

path	Path to directory or input files
format	file formats. "asd" (for ASD); "sig" or "svc" (for SVC); "sed" or "psr" (for SpecEvo PSR).
recursive	read files recursively
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.
...	nothing yet

**Value**

a single 'spectra' or a list of 'spectra' (in case files have incompatible band number or wavelengths 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")
```

---

reflectance

*Get spectra reflectance*

---

**Description**

reflectance returns the reflectance matrix from spectra

**Usage**

```
reflectance(x)
```

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

**Arguments**

x                    spectra object

**Value**

matrix with samples in rows and wavelengths in columns

**Methods (by class)**

- spectra: Get spectra reflectance

**Author(s)**

Jose Eduardo Meireles

**Examples**

```
library(spectrolab)
spec = as.spectra(spec_matrix_example, name_idx = 1)
is.matrix(reflectance(spec))
```

---

reflectance<-            *Set spectra reflectance*

---

**Description**

reflectance Assigns the rhs to the reflectance of the lhs spectra obj

**Usage**

```
reflectance(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 reflectance values by 2
reflectance(spec) = reflectance(spec) * 2
```

---

resample                *Resample spectra*

---

**Description**

resample returns spectra resampled to new wavelengths using smoothing. Possible to increase or decrease the spectral resolution.

**Usage**

```
resample(x, new_wvls, ...)

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

**Arguments**

x spectra object. Wavelengths must be strictly increasing  
 new\_wvls numeric vector of wavelengths to sample from spectra  
 ... additional parameters passed to the `smooth.spline` function.

**Details**

The function runs a couple basic checks when resampling, but they are not exhaustive, so look at the data before resampling. The implemented checks are: 1. Stop if trying to predict wavelengths outside of the original range and, 2. Warn if a gap is found in wavelengths. E.g. wvls are mostly at a 1nm resolution but go from 1530 to 1820 in the infrared. Does not check for NAs

**Value**

spectra object with resampled spectra

**Methods (by class)**

- spectra: Resample spectra

**Author(s)**

Jose Eduardo Meireles

**Examples**

```
library(spectrolab)
spec = as.spectra(spec_matrix_example, name_idx = 1)
spec = resample(spec, new_wvls = seq(400, 2400, 0.5), parallel = FALSE)
```

---

 sd

*Standard deviation*


---

**Description**

sd computes the standard deviation spectrum. Note that values will not reflect reflectance anymore, but the sd of the reflectance 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	<i>Default variance</i>
------------	-------------------------

---

**Description**

This function 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**

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

**Details**

Like `var` this uses denominator  $n - 1$ .

The standard deviation of a zero-length vector (after removal of NAs if `na.rm = TRUE`) is not defined and gives an error. The standard deviation of a length-one vector is NA.

**See Also**

`var` for its square, and `mad`, the most robust alternative.

**Examples**

```
sd(1:2) ^ 2
```

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	<i>Default smoothing function</i>
----------------	-----------------------------------

---

**Description**

Default smoothing function

**Usage**

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

**Arguments**

x	data to smooth over
...	additional arguments

**Value**

smoothed data

---

smooth.spectra	<i>Smooth spectra</i>
----------------	-----------------------

---

**Description**

smooth runs each spectrum by a smoothing and returns the spectra

**Usage**

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

**Arguments**

x	spectra object. Wavelengths must be strictly increasing
method	Choose smoothing method: "spline" (default) or "moving_average"
...	additional parameters passed to smooth.spline or parameters 'n' and 'save_wvls_to_meta' for the moving average smoothing.

**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, parallel = FALSE)
```

---

spectra

*Spectra object constructor*

---

**Description**

spectra "manually" creates a spectra object

**Usage**

```
spectra(reflectance, wavelengths, names, meta = NULL, ...)
```

**Arguments**

reflectance	N by M numeric matrix. N samples in rows and M wavelengths in columns
wavelengths	wavelength 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(reflectance) 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 reflectance matrix.
#   In this case, by removing the first column that holds the species name
rf = spec_matrix_example[ , -1]

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

# (3) Create a vector with sample labels that match
#     the reflectance 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(reflectance = rf, wavelengths = wl, names = sn)

```

---

spectrolab

*Spectrolab*


---

**Description**

Class and methods for hyperspectral data.

---

spec\_matrix\_example

*Example spectral dataset*


---

**Description**

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

**Usage**

```
spec_matrix_example
```

**Format**

An object of class `matrix` with 50 rows and 2102 columns.

**Author(s)**

Jose Eduardo Meireles

---

spec_with_jump	<i>Example spectra with jump between sensors</i>
----------------	--

---

**Description**

Spectra object of a simulated spectra with "jumps" between sensors. Used to illustrate the 'match\_sensors' function

**Usage**

```
spec_with_jump
```

**Format**

An object of class spectra with 1 rows and 2101 columns.

**Author(s)**

Jose Eduardo Meireles

---

split.spectra	<i>Split spectra</i>
---------------	----------------------

---

**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	<i>Structure of the spectra object</i>
-------------	--

---

**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	<i>Subset spectra by factor</i>
-----------	---------------------------------

---

### 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)

- 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	<i>Summarize spectra</i>
-----------------	--------------------------

---

## 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 reflectance instead

**Usage**

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

**Arguments**

x                    spectra

**Value**

nothing. operation not allowed

**Author(s)**

Jose Eduardo Meireles

**Examples**

```
library(spectrolab)  
s = as.spectra(spec_matrix_example, name_idx = 1)  
  
# This will throw an error  
## Not run:  
t(s)  
  
## End(Not run)  
# But these options should work  
t(reflectance(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"))
```

---

var

*Variance*

---

**Description**

var computes the variance spectrum. Note that values will not reflect reflectance anymore, but the variance of the reflectance 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 reflectance anymore, but the variance of the reflectance 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	<i>Variance spectrum</i>
-------------	--------------------------

---

**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
...	further arguments passed to methods

**Value**

single spectrum

**Author(s)**

Jose Eduardo Meireles

**Examples**

```
library(spectrolab)  
spec = as.spectra(spec_matrix_example, name_idx = 1)  
var(spec)
```

---

wavelengths	<i>Get spectra wavelength labels</i>
-------------	--------------------------------------

---

**Description**

wavelengths returns a vector of wavelength labels from spectra

**Usage**

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

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

**Arguments**

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

**Value**

vector of wavelengths. numeric if 'return\_num' = TRUE (default).

**Methods (by class)**

- spectra: Set spectra wavelength labels

**Author(s)**

Jose Eduardo Meireles

**Examples**

```
library(spectrolab)
spec = as.spectra(spec_matrix_example, name_idx = 1)
head(wavelengths(spec))
```

---

*wavelengths<-*                    *Set wavelength labels*

---

**Description**

*wavelengths* sets wavelength labels of lhs to the rhs values

**Usage**

```
wavelengths(x, unsafe = FALSE) <- value
```

**Arguments**

x	spectra object (lhs)
unsafe	boolean. Skip length safety check? Defaults to FALSE
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)
wavelengths(spec) = wavelengths(spec) / 1000
```

---

[.spectra	<i>Subset spectra</i>
-----------	-----------------------

---

**Description**

``[`` Subsets spectra by sample names (rows) or (and) wavelengths (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 <code>nrow(x)</code>
j	Wavelength labels, as numeric or character or a logical vector of length <code>ncol(x)</code> . Do not use indexes!
simplify	Boolean. If TRUE (default), single band selections are returned as a named vector of reflectance 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 wavelengths (second argument) matches the wavelength labels, not indices! That is, `x[, 600]` will give you the reflectance data for the 600nm wavelength and not the 600th band. Boolean vectors of the appropriate length can be used to subset samples and wavelengths.

**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 reflectance 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	Wavelength 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

## \*Topic **datasets**

spec\_matrix\_example, 35  
spec\_with\_jump, 36  
[.spectra, 45  
[<-.spectra, 46

aggregate.spectra, 3  
apply\_by\_band, 4  
as.data.frame.spectra, 5  
as.matrix.spectra, 6  
as.spectra, 6  
as.spectra.data.frame, 7  
as.spectra.matrix, 8

combine, 8

default\_spec\_regions, 9  
dim.spectra, 10

has\_nir\_dip, 10

is\_spectra, 11

mad, 31  
match\_sensors, 12  
max.spectra, 13  
mean.spectra, 14  
median.spectra, 14  
meta, 15  
meta<-, 16  
min.spectra, 17

names.spectra, 17  
names<-.spectra, 18  
normalize, 19

Ops.spectra, 20

plot.spectra, 20  
plot\_interactive, 21  
plot\_quantile, 22

plot\_regions, 23  
print.spectra, 24  
quantile.spectra, 25

range.spectra, 26  
ratio, 26  
read\_spectra, 27  
reflectance, 28  
reflectance<-, 29  
resample, 29

sd, 30  
sd.default, 31  
sd.spectra, 32  
smooth, 32  
smooth.default, 33  
smooth.spectra, 33  
spec\_matrix\_example, 35  
spec\_with\_jump, 36  
spectra, 34  
spectrolab, 35  
spectrolab-package (spectrolab), 35  
split.spectra, 36  
str.spectra, 37  
subset\_by, 38  
summary.spectra, 39

t.spectra, 40  
try\_keep\_txt, 40

var, 31, 41  
var.default, 42  
var.spectra, 43

wavelengths, 43  
wavelengths<-, 44