Package 'photobiology'

September 11, 2019

Type Package

Title Photobiological Calculations

Version 0.9.29

Date 2019-09-10

Description Definitions of classes, methods, operators and functions for use in photobiology and radiation meteorology and climatology. Calculation of effective (weighted) and not-weighted irradiances/doses, fluence rates, transmittance, reflectance, absorptance, absorbance and diverse ratios and other derived quantities from spectral data. Local maxima and minima. Conversion between energy- and photon-based units. Wavelength interpolation. Astronomical calculations related solar angles and day length. Colours and vision. This package is part of the 'r4photobiology' suite, Aphalo P. J. (2015) <doi:10.19232/uv4pb.2015.1.14>.

License GPL (>= 2)

Depends R (>= 3.5.0), tibble (>= 2.1.3)

Imports stats, polynom (>= 1.4-0), lubridate (>= 1.7.4), plyr (>= 1.8.4), dplyr (>= 0.8.1), splus2R (>= 1.2-2), zoo (>= 1.8-5), rlang (>= 0.3.4)

Suggests knitr (>= 1.23), rmarkdown (>= 1.13), testthat (>= 2.1.1), roxygen2 (>= 6.1.1)

LazyLoad yes

LazyData yes

ByteCompile true

URL https://www.r4photobiology.info/, https://bitbucket.org/aphalo/photobiology

BugReports https://bitbucket.org/aphalo/photobiology/issues

Encoding UTF-8

RoxygenNote 6.1.1

VignetteBuilder knitr

NeedsCompilation no

Author Pedro J. Aphalo [aut, cre] (<https://orcid.org/0000-0003-3385-972X>),
Titta K. Kotilainen [ctb] (<https://orcid.org/0000-0002-2822-9734>),
Glenn Davis [ctb]

Maintainer Pedro J. Aphalo <pedro.aphalo@helsinki.fi>

Repository CRAN

Date/Publication 2019-09-11 07:00:03 UTC

R topics documented:

photobiology-package *photobiology: Photobiological Calculations*

Description

Definitions of classes, methods, operators and functions for use in photobiology and radiation meteorology and climatology. Calculation of effective (weighted) and not-weighted irradiances/doses, fluence rates, transmittance, reflectance, absorptance, absorbance and diverse ratios and other derived quantities from spectral data. Local maxima and minima. Conversion between energy- and photon-based units. Wavelength interpolation. Astronomical calculations related solar angles and day length. Colours and vision. This package is part of the 'r4photobiology' suite, Aphalo P. J. (2015) <doi:10.19232/uv4pb.2015.1.14>.

Details

Package 'photobiology' is at the core of a suite of packages for analysis and plotting of data relevant to photobiology (described at https://www.r4photobiology.info/). The accompanying packages (under development) provide data and definitions that are to a large extent application-area specific while the functions in the present package are widely useful in photobiology and radiation quantification in geophysics and meteorology. Package 'photobiology' has its main focus in the characterization of the light environment in a biologically relevant manner and in the manipulation of spectral data to simulate photo-physical, photo-chemical and photo-biological interactions and responses. The focus of package 'pavo' (Maia et al., 2003) is in colour perception by animals and assessment of animal coloration. In spite of the different focus, there is some degree of overlap.

Acknowledgements

This work was funded by the Academy of Finland (decision 252548). COST Action FA9604 'UV4Growth' facilitated discussions and exchanges of ideas that lead to the development of this package. The contributions of Andy McLeod, Lars Olof Björn, Nigel Paul, Lasse Ylianttila, T. Matthew Robson and Titta Kotilainen were specially significant. Tutorials by Hadley Wickham and comments on my presentation at UseR!2015 allowed me to significantly improve the coding and functionality.

Note

Code for some of the astronomical calculations has been adapted from that in package 'pavo'.

Author(s)

Maintainer: Pedro J. Aphalo <pedro.aphalo@helsinki.fi> (0000-0003-3385-972X)

Other contributors:

- Titta K. Kotilainen (0000-0002-2822-9734) [contributor]
- Glenn Davis <gdavis@gluonics.com> [contributor]

References

Aphalo, P. J., Albert, A., Björn, L. O., McLeod, A. R., Robson, T. M., Rosenqvist, E. (Eds.). (2012). Beyond the Visible: A handbook of best practice in plant UV photobiology (1st ed., p. xx + 174). Helsinki: University of Helsinki, Department of Biosciences, Division of Plant Biology. ISBN 978-952-10-8363-1 (PDF), 978-952-10-8362-4 (paperback). Open access PDF download available at https://hdl.handle.net/10138/37558

Aphalo, Pedro J. (2015) The r4photobiology suite. UV4Plants Bulletin, 2015:1, 21-29. https://doi.org/10.19232/uv4pb.2015.1.14.

Maia, R., Eliason, C. M., Bitton, P. P., Doucet, S. M., Shawkey, M. D. (2013) pavo: an R package for the analysis, visualization and organization of spectral data. Methods in Ecology and Evolution, 4(10):906-913. https://doi.org/10.1111/2041-210X.12069.

See Also

Useful links:

- https://www.r4photobiology.info/
- https://bitbucket.org/aphalo/photobiology
- Report bugs at https://bitbucket.org/aphalo/photobiology/issues

Examples

```
# irradiance of the whole spectrum
irrad(sun.spct)
# photon irradiance 400 nm to 700 nm
q_irrad(sun.spct, waveband(c(400,700)))
# energy irradiance 400 nm to 700 nm
e_irrad(sun.spct, waveband(c(400,700)))
# simulating the effect of a filter on solar irradiance
e_irrad(sun.spct * yellow_gel.spct, waveband(c(400,500)))
e_irrad(sun.spct * yellow_gel.spct, waveband(c(500,700)))
# daylength
sunrise_time(lubridate::today(tzone = "EET"), tz = "EET",
             geocode = data.frame(lat = 60, lon = 25), unit.out = "hour")
day_length(lubridate::today(tzone = "EET"), tz = "EET",
           geocode = data.frame(lat = 60, lon = 25), unit.out = "hour")
# colour as seen by humans
color_of(sun.spct)
color_of(sun.spct * yellow_gel.spct)
# filter transmittance
transmittance(yellow_gel.spct)
transmittance(yellow_gel.spct, waveband(c(400,500)))
transmittance(yellow_gel.spct, waveband(c(500,700)))
```

`A.illuminant.spct` *CIE A illuminant data*

Description

A dataset containing wavelengths at a 5 nm interval (300 nm to 830 nm) and the corresponding spectral energy irradiance normalized to 1 at 560 nm. Spectrum approximates typical, domestic, tungsten-filament lighting and 'corresponds' to a black body a 2856 K. CIE standard illuminant A is intended to represent typical, domestic, tungsten-filament lighting. Original data from http://files.cie.co.at/204.xls downloaded on 2014-07-25 The variables are as follows:

Usage

```
A.illuminant.spct
```

Format

A source spectrum with 96 rows and 2 variables

Details

- w.length (nm)
- s.e.irrad (rel. units)

Author(s)

CIE

See Also

Other Spectral data examples: `D65.illuminant.spct`, `Ler_leaf.spct`, `Ler_leaf_rflt.spct`, `Ler_leaf_trns.spct`, `Ler_leaf_trns_i.spct`, `black_body.spct`, `ccd.spct`, `clear.spct`, `clear_body.spct`, `filter_cps.mspct`, `green_leaf.spct`, `opaque.spct`, `photodiode.spct`, `polyester.spct`, `sun.daily.data`, `sun.daily.spct`, `sun.data`, `sun.spct`, `white_body.spct`, `white_led.cps_spct`, `white_led.raw_spct`, `white_led.source_spct`, `yellow_gel.spct`

Examples

```
A.illuminant.spct
```

A2T *Convert absorbance into transmittance*

Description

Function that converts absorbance (a.u.) into transmittance (fraction).

Usage

```
A2T(x, action, byref, ...)

## Default S3 method:
A2T(x, action = NULL, byref = FALSE, ...)

## S3 method for class 'filter_spct'
A2T(x, action = "add", byref = FALSE, ...)

## S3 method for class 'filter_mspct'
A2T(x, action = "add", byref = FALSE, ...,
  .parallel = FALSE, .paropts = NULL)
```

Arguments

x	an R object
action	a character string
byref	logical indicating if new object will be created by reference or by copy of x
...	not used in current version
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Methods (by class)

- `default`: Default method for generic function
- `filter_spct`: Method for filter spectra
- `filter_mspct`: Method for collections of filter spectra

See Also

Other quantity conversion functions: `T2Afr`, `T2A`, `T2T`, `as_quantum`, `e2qmol_multipliers`, `e2quantum_multipliers`, `e2q`, `q2e`

absorbance *Absorbance*

Description

Function to calculate the mean, total, or other summary of absorbance for spectral data stored in a `filter_spct` or in an `object_spct`.

Usage

```
absorbance(spct, w.band, quantity, wb.trim, use.hinges, ...)

## Default S3 method:
absorbance(spct, w.band, quantity, wb.trim, use.hinges,
  ...)

## S3 method for class 'filter_spct'
absorbance(spct, w.band = NULL,
  quantity = "average",
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.hinges = getOption("photobiology.use.hinges", default = NULL), ...)

## S3 method for class 'object_spct'
absorbance(spct, w.band = NULL,
  quantity = "average",
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.hinges = getOption("photobiology.use.hinges", default = NULL), ...)

## S3 method for class 'filter_mspct'
absorbance(spct, w.band = NULL,
  quantity = "average",
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.hinges = getOption("photobiology.use.hinges", default = NULL), ...,
  attr2tb = NULL, idx = "spct.idx")

## S3 method for class 'object_mspct'
absorbance(spct, w.band = NULL,
  quantity = "average",
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.hinges = getOption("photobiology.use.hinges", default = NULL), ...,
  attr2tb = NULL, idx = "spct.idx", .parallel = FALSE,
  .paropts = NULL)
```

Arguments

spct	an R object.

w.band	waveband or list of waveband objects or a numeric vector of length two. The waveband(s) determine the region(s) of the spectrum that are summarized. If a numeric range is supplied a waveband object is constructed on the fly from it.
quantity	character string One of "average" or "mean", "total", "contribution", "contribution.pc", "relative" or "relative.pc".
wb.trim	logical Flag indicating if wavebands crossing spectral data boundaries are trimmed or ignored.
use.hinges	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.
...	other arguments (possibly used by derived methods).
attr2tb	character vector, see add_attr2tb for the syntax for attr2tb passed as is to formal parameter col.names.
idx	character Name of the column with the names of the members of the collection of spectra.
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Value

A named numeric vector in the case of methods for individual spectra, with one value for each waveband passed to parameter w.band. A data.frame in the case of collections of spectra, containing one column for each waveband object, an index column with the names of the spectra, and optionally additional columns with metadata values retrieved from the attributes of the member spectra.

By default values are only integrated, but depending on the argument passed to parameter quantity they can be re-expressed as relative fractions or percentages. In the case of vector output, names attribute is set to the name of the corresponding waveband unless a named list is supplied in which case the names of the list members are used.

Methods (by class)

- default: Default for generic function
- filter_spct: Specialization for filter spectra
- object_spct: Specialization for object spectra
- filter_mspct: Calculates absorbance from a filter_mspct
- object_mspct: Calculates absorbance from a object_mspct

Note

The use.hinges parameter controls speed optimization. The defaults should be suitable in most cases. Only the range of wavelengths in the wavebands is used and all BSWFs are ignored.

Examples

```
absorbance(polyester.spct, new_waveband(400,700))
absorbance(yellow_gel.spct, new_waveband(400,700))
absorbance(yellow_gel.spct, split_bands(c(400,700), length.out = 3))
absorbance(yellow_gel.spct, split_bands(c(400,700), length.out = 3),
        quantity = "average")
absorbance(yellow_gel.spct, split_bands(c(400,700), length.out = 3),
        quantity = "total")
absorbance(yellow_gel.spct, split_bands(c(400,700), length.out = 3),
        quantity = "relative")
absorbance(yellow_gel.spct, split_bands(c(400,700), length.out = 3),
        quantity = "relative.pc")
absorbance(yellow_gel.spct, split_bands(c(400,700), length.out = 3),
        quantity = "contribution")
absorbance(yellow_gel.spct, split_bands(c(400,700), length.out = 3),
        quantity = "contribution.pc")
```

---

absorptance *Absorptance*

Description

Function to calculate the mean, total, or other summary of absorptance for spectral data stored in a `filter_spct` or in an `object_spct`. Absorptance is a different quantity than absorbance.

Usage

```
absorptance(spct, w.band, quantity, wb.trim, use.hinges, ...)

## Default S3 method:
absorptance(spct, w.band, quantity, wb.trim, use.hinges,
  ...)

## S3 method for class 'filter_spct'
absorptance(spct, w.band = NULL,
  quantity = "average",
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.hinges = getOption("photobiology.use.hinges", default = NULL), ...)

## S3 method for class 'object_spct'
absorptance(spct, w.band = NULL,
  quantity = "average",
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.hinges = getOption("photobiology.use.hinges", default = NULL), ...)

## S3 method for class 'filter_mspct'
absorptance(spct, w.band = NULL,
```

```
  quantity = "average",
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.hinges = getOption("photobiology.use.hinges", default = NULL), ...,
  attr2tb = NULL, idx = "spct.idx")

## S3 method for class 'object_mspct'
absorptance(spct, w.band = NULL,
  quantity = "average",
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.hinges = getOption("photobiology.use.hinges", default = NULL), ...,
  attr2tb = NULL, idx = "spct.idx", .parallel = FALSE,
  .paropts = NULL)
```

Arguments

spct	an R object.
w.band	waveband or list of wavebands objects or a numeric vector of length two. The waveband(s) determine the region(s) of the spectrum that are summarized. If a numeric range is supplied a waveband object is constructed on the fly from it.
quantity	character string One of "average" or "mean", "total", "contribution", "contribution.pc", "relative" or "relative.pc".
wb.trim	logical Flag if wavebands crossing spectral data boundaries are trimmed or ignored.
use.hinges	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.
...	other arguments (possibly used by derived methods).
attr2tb	character vector, see add_attr2tb for the syntax for attr2tb passed as is to formal parameter col.names.
idx	character Name of the column with the names of the members of the collection of spectra.
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Value

A named numeric vector in the case of methods for individual spectra, with one value for each waveband passed to parameter w.band. A data.frame in the case of collections of spectra, containing one column for each waveband object, an index column with the names of the spectra, and optionally additional columns with metadata values retrieved from the attributes of the member spectra.

By default values are only integrated, but depending on the argument passed to parameter quantity they can be re-expressed as relative fractions or percentages. In the case of vector output, names

attribute is set to the name of the corresponding waveband unless a named list is supplied in which case the names of the list members are used.

Methods (by class)

- `default`: Default for generic function
- `filter_spct`: Specialization for filter spectra
- `object_spct`: Specialization for object spectra
- `filter_mspct`: Calculates absorptance from a `filter_mspct`
- `object_mspct`: Calculates absorptance from a `object_mspct`

Note

The `use.hinges` parameter controls speed optimization. The defaults should be suitable in most cases. Only the range of wavelengths in the wavebands is used and all BSWFs are ignored.

Examples

```
absorptance(black_body.spct, new_waveband(400,500))
absorptance(white_body.spct, new_waveband(300,400))
absorptance(black_body.spct, split_bands(c(400,700), length.out = 3))
absorptance(black_body.spct, split_bands(c(400,700), length.out = 3),
        quantity = "average")
absorptance(black_body.spct, split_bands(c(400,700), length.out = 3),
        quantity = "total")
absorptance(black_body.spct, split_bands(c(400,700), length.out = 3),
        quantity = "relative")
absorptance(black_body.spct, split_bands(c(400,700), length.out = 3),
        quantity = "relative.pc")
absorptance(black_body.spct, split_bands(c(400,700), length.out = 3),
        quantity = "contribution")
absorptance(black_body.spct, split_bands(c(400,700), length.out = 3),
        quantity = "contribution.pc")
```

---

add_attr2tb *Copy attributes from members of a generic_mspct*

Description

Copy the when.measured, where.measured or what.measured attribute from members of a generic_mspct object into a tibble or data.frame.

Usage

```
add_attr2tb(tb, mspct, col.names = NULL, idx = "spct.idx")

when_measured2tb(mspct, tb = NULL, col.names = "when.measured",
  idx = "spct.idx")

lonlat2tb(mspct, tb = NULL, col.names = c("lon", "lat"),
  idx = "spct.idx")

lon2tb(mspct, tb = NULL, col.names = "lon", idx = "spct.idx")

lat2tb(mspct, tb = NULL, col.names = "lat", idx = "spct.idx")

geocode2tb(mspct, tb = NULL, col.names = "geocode", idx = "spct.idx")

what_measured2tb(mspct, tb = NULL, col.names = "what.measured",
  idx = "spct.idx")
```

Arguments

tb	tibble or data.frame to which to add the data (optional).
mspct	generic_mspct Any collection of spectra.
col.names	named character vector Name(s) of column(s) to create. Values are the names of the attributes to copy, while if named, the names provide the name for the column.
idx	character Name of the column with the names of the members of the collection of spectra.

Details

The attributes are copied to a column in a tibble or data frame. If the `tb` formal parameter receives `NULL` as argument, a new `tibble` will be created. If an existing `data.frame` or `tibble` is passed as argument, new columns are added to it. However, the number of rows in the argument passed to `tb` must match the number of spectra in the argument passed to `mspct`. If the argument to `col.names` is aa named vector, with the names of members matching the names of attributes, then the values are used as names for the columns created. This permits setting any valid name for the new columns. If the vector passed to `col.names` has no names, then the values are interpreted as the names of the attributes to add, and also used as names for the new columns.

Value

A tibble With the metadata attributes in separate new variables.

Note

Currently supported attributes are `"when.measured"`, `"what.measured"` and `"where.measured"`. In the case of `"where.measured"`, which has different components the name `"where.measured"` is ignored, but instead the following names are recognized: `"lon"` and `"lat"` for creating numeric

columns of longitudes and latitudes respectively, and "geocode" for creating a column of data frames, in which case, if tb is not already a tibble it is converted into one before adding the new column. The order of the first two arguments is reversed in add_attr2tb() compared to the other functions. This is to allow its use in 'pipes', while the functions for single attributes are expected to be used mostly to create new tibbles.

Examples

```
library(dplyr)

my.mspct <- source_mspct(list(sun1 = sun.spct, sun2 = sun.spct * 2))
q_irrad(my.mspct) %>%
  add_attr2tb(my.mspct, c(lat = "latitude",
                          lon = "longitude",
                          when.measured = "time"))

when_measured2tb(my.mspct)
```

---

as.calibration_mspct *Coerce to a collection-of-spectra*

Description

Return a copy of an R object with its class set to a given type of spectrum.

Usage

```
as.calibration_mspct(x, ...)

## Default S3 method:
as.calibration_mspct(x, ...)

## S3 method for class 'data.frame'
as.calibration_mspct(x, ...)

## S3 method for class 'calibration_spct'
as.calibration_mspct(x, ...)

## S3 method for class 'list'
as.calibration_mspct(x, ..., ncol = 1, byrow = FALSE)

## S3 method for class 'matrix'
as.calibration_mspct(x, w.length,
  spct.data.var = "irrad.mult", multiplier = 1, byrow = NULL,
  spct.names = "spct_", ...)
```

Arguments

x	a list of spectral objects or a list of objects such as data frames that can be converted into spectral objects.
...	passed to individual spectrum object constructor
ncol	integer Number of 'virtual' columns in data
byrow	logical If ncol > 1 how to read in the data
w.length	numeric A vector of wavelengthvalues sorted in strictly ascending order (nm).
spct.data.var	character The name of the variable that will contain the spectral data. This indicates what physical quantity is stored in the matrix and the units of expression used.
multiplier	numeric A multiplier to be applied to the values in x to do unit or scale conversion.
spct.names	character Vector of names to be assigned to collection members, either of length 1, or with length equal to the number of spectra.

Value

A copy of x converted into a calibration_mspctt object.

Methods (by class)

- default:
- data.frame:
- calibration_spct:
- list:
- matrix:

Note

When x is a square matrix an explicit argument is needed for byrow to indicate how data in x should be read. In every case the length of the w.length vector must match one of the dimensions of x.

See Also

Other Coercion methods for collections of spectra: as.chroma_mspct, as.cps_mspct, as.filter_mspct, as.generic_mspct, as.object_mspct, as.raw_mspct, as.reflector_mspct, as.response_mspct, as.source_mspct, split2mspct, subset2mspct

as.calibration_spct *Coerce to a spectrum*

Description

Return a copy of an R object with its class set to a given type of spectrum.

Usage

```
as.calibration_spct(x, ...)
```

Arguments

x	an R object
...	other arguments passed to "set" functions

Value

A copy of x converted into a calibration_spct object.

See Also

setGenericSpct

Other constructors of spectral objects: as.chroma_spct, as.cps_spct, as.filter_spct, as.generic_spct, as.object_spct, as.raw_spct, as.reflector_spct, as.response_spct, as.source_spct, source_spct

as.chroma_mspct *Coerce to a collection-of-spectra*

Description

Return a copy of an R object with its class set to a given type of spectrum.

Usage

```
as.chroma_mspct(x, ...)

## Default S3 method:
as.chroma_mspct(x, ...)

## S3 method for class 'data.frame'
as.chroma_mspct(x, ...)

## S3 method for class 'chroma_spct'
as.chroma_mspct(x, ...)

## S3 method for class 'list'
as.chroma_mspct(x, ..., ncol = 1, byrow = FALSE)
```

Arguments

x	a list of spectral objects or a list of objects such as data frames that can be converted into spectral objects.
...	passed to individual spectrum object constructor
ncol	integer Number of 'virtual' columns in data
byrow	logical If ncol > 1 how to read in the data

Value

A copy of x converted into a chroma_mspct object.

Methods (by class)

- default:
- data.frame:
- chroma_spct:
- list:

See Also

Other Coercion methods for collections of spectra: as.calibration_mspct, as.cps_mspct, as.filter_mspct, as.generic_mspct, as.object_mspct, as.raw_mspct, as.reflector_mspct, as.response_mspct, as.source_mspct, split2mspct, subset2mspct

---

as.chroma_spct *Coerce to a spectrum*

Description

Return a copy of an R object with its class set to a given type of spectrum.

Usage

```
as.chroma_spct(x, ...)
```

Arguments

x	an R object
...	other arguments passed to "set" functions

Value

A copy of x converted into a chroma_spct object.

See Also

setGenericSpct

Other constructors of spectral objects: as.calibration_spct, as.cps_spct, as.filter_spct, as.generic_spct, as.object_spct, as.raw_spct, as.reflector_spct, as.response_spct, as.source_spct, source_spct

---

as.cps_mspct *Coerce to a collection-of-spectra*

Description

Return a copy of an R object with its class set to a given type of spectrum.

Usage

```
as.cps_mspct(x, ...)

## Default S3 method:
as.cps_mspct(x, ...)

## S3 method for class 'data.frame'
as.cps_mspct(x, ...)

## S3 method for class 'cps_spct'
as.cps_mspct(x, ...)

## S3 method for class 'list'
as.cps_mspct(x, ..., ncol = 1, byrow = FALSE)

## S3 method for class 'matrix'
as.cps_mspct(x, w.length, spct.data.var = "cps",
  multiplier = 1, byrow = NULL, spct.names = "spct_", ...)
```

Arguments

x	a list of spectral objects or a list of objects such as data frames that can be converted into spectral objects.
...	passed to individual spectrum object constructor
ncol	integer Number of 'virtual' columns in data
byrow	logical If ncol > 1 how to read in the data
w.length	numeric A vector of wavelengthvalues sorted in strictly ascending order (nm).
spct.data.var	character The name of the variable that will contain the spectral data. This indicates what physical quantity is stored in the matrix and the units of expression used.

multiplier	numeric A multiplier to be applied to the values in x to do unit or scale conversion.
spct.names	character Vector of names to be assigned to collection members, either of length 1, or with length equal to the number of spectra.

Value

A copy of x converted into a cps_mspct object.

Methods (by class)

- default:
- data.frame:
- cps_spct:
- list:
- matrix:

Note

When x is a square matrix an explicit argument is needed for byrow to indicate how data in x should be read. In every case the length of the w.length vector must match one of the dimensions of x.

See Also

Other Coercion methods for collections of spectra: as.calibration_mspct, as.chroma_mspct, as.filter_mspct, as.generic_mspct, as.object_mspct, as.raw_mspct, as.reflector_mspct, as.response_mspct, as.source_mspct, split2mspct, subset2mspct

---

as.cps_spct *Coerce to a spectrum*

Description

Return a copy of an R object with its class set to a given type of spectrum.

Usage

```
as.cps_spct(x, ...)

## Default S3 method:
as.cps_spct(x, ...)
```

Arguments

x	an R object
...	other arguments passed to "set" functions

Value

A copy of x converted into a cps_spct object.

Methods (by class)

- default:

See Also

setGenericSpct

Other constructors of spectral objects: as.calibration_spct, as.chroma_spct, as.filter_spct, as.generic_spct, as.object_spct, as.raw_spct, as.reflector_spct, as.response_spct, as.source_spct, source_spct

---

as.filter_mspct *Coerce to a collection-of-spectra*

Description

Return a copy of an R object with its class set to a given type of spectrum.

Usage

```
as.filter_mspct(x, ...)

## Default S3 method:
as.filter_mspct(x, ...)

## S3 method for class 'data.frame'
as.filter_mspct(x, Tfr.type = c("total",
  "internal"), strict.range = TRUE, ...)

## S3 method for class 'filter_spct'
as.filter_mspct(x, ...)

## S3 method for class 'list'
as.filter_mspct(x, Tfr.type = c("total", "internal"),
  strict.range = TRUE, ..., ncol = 1, byrow = FALSE)

## S3 method for class 'matrix'
as.filter_mspct(x, w.length, spct.data.var = "Tfr",
  multiplier = 1, byrow = NULL, spct.names = "spct_", ...)
```

Arguments

x	a list of spectral objects or a list of objects such as data frames that can be converted into spectral objects.
...	passed to individual spectrum object constructor
Tfr.type	a character string, either "total" or "internal"
strict.range	logical Flag indicating how off-range values are handled
ncol	integer Number of 'virtual' columns in data
byrow	logical If ncol > 1 how to read in the data
w.length	numeric A vector of wavelengthvalues sorted in strictly ascending order (nm).
spct.data.var	character The name of the variable that will contain the spectral data. This indicates what physical quantity is stored in the matrix and the units of expression used.
multiplier	numeric A multiplier to be applied to the values in x to do unit or scale conversion.
spct.names	character Vector of names to be assigned to collection members, either of length 1, or with length equal to the number of spectra.

Value

A copy of x converted into a filter_mspct object.

Methods (by class)

- default:
- data.frame:
- filter_spct:
- list:
- matrix:

Note

When x is a square matrix an explicit argument is needed for byrow to indicate how data in x should be read. In every case the length of the w.length vector must match one of the dimensions of x.

See Also

Other Coercion methods for collections of spectra: as.calibration_mspct, as.chroma_mspct, as.cps_mspct, as.generic_mspct, as.object_mspct, as.raw_mspct, as.reflector_mspct, as.response_mspct, as.source_mspct, split2mspct, subset2mspct

as.filter_spct *Coerce to a spectrum*

Description

Return a copy of an R object with its class set to a given type of spectrum.

Usage

```
as.filter_spct(x, ...)

## Default S3 method:
as.filter_spct(x, Tfr.type = c("total", "internal"),
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  ...)
```

Arguments

x	an R object
...	other arguments passed to "set" functions
Tfr.type	a character string, either "total" or "internal"
strict.range	logical Flag indicating whether off-range values result in an error instead of a warning

Value

A copy of x converted into a filter_spct object.

Methods (by class)

- default:

See Also

setGenericSpct

Other constructors of spectral objects: as.calibration_spct, as.chroma_spct, as.cps_spct, as.generic_spct, as.object_spct, as.raw_spct, as.reflector_spct, as.response_spct, as.source_spct, source_spct

as.generic_mspct *Coerce to a collection-of-spectra*

Description

Return a copy of an R object with its class set to a given type of spectrum.

Usage

```
as.generic_mspct(x, ...)

## Default S3 method:
as.generic_mspct(x, ...)

## S3 method for class 'data.frame'
as.generic_mspct(x, force.spct.class = FALSE, ...)

## S3 method for class 'generic_spct'
as.generic_mspct(x, force.spct.class = FALSE, ...)

## S3 method for class 'list'
as.generic_mspct(x, force.spct.class = FALSE, ...,
  ncol = 1, byrow = FALSE)

## S3 method for class 'matrix'
as.generic_mspct(x, w.length, member.class, spct.data.var,
  multiplier = 1, byrow = NULL, spct.names = "spct_", ...)

mat2mspct(x, w.length, member.class, spct.data.var, multiplier = 1,
  byrow = NULL, spct.names = "spct_", ...)
```

Arguments

x	a list of spectral objects or a list of objects such as data frames that can be converted into spectral objects.
...	passed to individual spectrum object constructor
force.spct.class	logical indicating whether to change the class of members to generic_spct or retain the existing class.
ncol	integer Number of 'virtual' columns in data
byrow	logical If ncol > 1 how to read in the data
w.length	numeric A vector of wavelengthvalues sorted in strictly ascending order (nm).
member.class	character The name of the class of the individual spectra to be constructed.
spct.data.var	character The name of the variable that will contain the spectral data. This indicates what physical quantity is stored in the matrix and the units of expression used.

multiplier	numeric A multiplier to be applied to the values in x to do unit or scale conversion.
spct.names	character Vector of names to be assigned to collection members, either of length 1, or with length equal to the number of spectra.

Value

A copy of x converted into a generic_mspct object.

Methods (by class)

- default:
- data.frame:
- generic_spct:
- list:
- matrix:

Note

Members of generic_mspct objects can be heterogeneous: they can belong to any class derived from generic_spct and class is not enforced. When x is a list of data frames force.spct.class = TRUE needs to be supplied. When x is a square matrix an explicit argument is needed for byrow to indicate how data in x should be read. In every case the length of the w.length vector must match one of the dimensions of x.

See Also

Other Coercion methods for collections of spectra: as.calibration_mspct, as.chroma_mspct, as.cps_mspct, as.filter_mspct, as.object_mspct, as.raw_mspct, as.reflector_mspct, as.response_mspct, as.source_mspct, split2mspct, subset2mspct

---

as.generic_spct *Coerce to a spectrum*

Description

Return a copy of an R object with its class set to a given type of spectrum.

Usage

```
as.generic_spct(x, ...)

## Default S3 method:
as.generic_spct(x, ...)

## Default S3 method:
```

```
as.calibration_spct(x, ...)

## Default S3 method:
as.raw_spct(x, ...)

## Default S3 method:
as.chroma_spct(x, ...)
```

Arguments

x	an R object
...	other arguments passed to "set" functions

Value

A copy of x converted into a generic_spct object.

Methods (by class)

- default:
- default:
- default:
- default:

See Also

setGenericSpct

Other constructors of spectral objects: as.calibration_spct, as.chroma_spct, as.cps_spct, as.filter_spct, as.object_spct, as.raw_spct, as.reflector_spct, as.response_spct, as.source_spct, source_spct

---

as.matrix-mspct *Coerce a collection of spectra into a matrix*

Description

Convert an object of class generic_mspct or a derived class into an R matrix with wavelengths saved as an attribute and spectral data in rows or columns.

Usage

```
## S3 method for class 'generic_mspct'
as.matrix(x, spct.data.var, byrow = attr(x,
  "mspct.byrow"), ...)

mspct2mat(x, spct.data.var, byrow = attr(x, "mspct.byrow"), ...)
```

Arguments

x	generic_mspct object.
spct.data.var	character The name of the variable containing the spectral data.
byrow	logical. If FALSE (the default) the matrix is filled with the spectra stored by columns, otherwise the matrix is filled by rows.
...	currently ignored.

Warning!

This conversion preserves the spectral data but discards almost all the metadata contained in the spectral objects. In other words a matrix created with this function cannot be used to recreate the original object unless the same metadata is explicitly supplied when converting the matrix into new collection of spectra.

Note

Only collections of spectra containing spectra with exactly the same `w.length` values can by converted. If needed, the spectra can be re-expressed before attempting the conversion to a matrix.

---

as.object_mspct *Coerce to a collection-of-spectra*

Description

Return a copy of an R object with its class set to a given type of spectrum.

Usage

```
as.object_mspct(x, ...)

## Default S3 method:
as.object_mspct(x, ...)

## S3 method for class 'data.frame'
as.object_mspct(x, Tfr.type = c("total",
  "internal"), Rfr.type = c("total", "specular"), strict.range = TRUE,
  ...)

## S3 method for class 'object_spct'
as.object_mspct(x, ...)

## S3 method for class 'list'
as.object_mspct(x, Tfr.type = c("total", "internal"),
  Rfr.type = c("total", "specular"), strict.range = TRUE, ...,
  ncol = 1, byrow = FALSE)
```

Arguments

x	a list of spectral objects or a list of objects such as data frames that can be converted into spectral objects.
...	passed to individual spectrum object constructor
Tfr.type	a character string, either "total" or "internal"
Rfr.type	a character string, either "total" or "specular"
strict.range	logical Flag indicating how off-range values are handled
ncol	integer Number of 'virtual' columns in data
byrow	logical If ncol > 1 how to read in the data

Value

A copy of x converted into a object_mspct object.

Methods (by class)

- default:
- data.frame:
- object_spct:
- list:

See Also

Other Coercion methods for collections of spectra: as.calibration_mspct, as.chroma_mspct, as.cps_mspct, as.filter_mspct, as.generic_mspct, as.raw_mspct, as.reflector_mspct, as.response_mspct, as.source_mspct, split2mspct, subset2mspct

---

as.object_spct *Coerce to a spectrum*

Description

Return a copy of an R object with its class set to a given type of spectrum.

Usage

```
as.object_spct(x, ...)

## Default S3 method:
as.object_spct(x, Tfr.type = c("total", "internal"),
  Rfr.type = c("total", "specular"),
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  ...)
```

Arguments

x	an R object
...	other arguments passed to "set" functions
Tfr.type	a character string, either "total" or "internal"
Rfr.type	a character string, either "total" or "specular"
strict.range	logical Flag indicating whether off-range values result in an error instead of a warning

Value

A copy of x converted into a object_spct object.

Methods (by class)

- default:

See Also

setGenericSpct

Other constructors of spectral objects: as.calibration_spct, as.chroma_spct, as.cps_spct, as.filter_spct, as.generic_spct, as.raw_spct, as.reflector_spct, as.response_spct, as.source_spct, source_spct

---

as.raw_mspct *Coerce to a collection-of-spectra*

Description

Return a copy of an R object with its class set to a given type of spectrum.

Usage

```
as.raw_mspct(x, ...)

## Default S3 method:
as.raw_mspct(x, ...)

## S3 method for class 'data.frame'
as.raw_mspct(x, ...)

## S3 method for class 'raw_spct'
as.raw_mspct(x, ...)

## S3 method for class 'list'
as.raw_mspct(x, ..., ncol = 1, byrow = FALSE)
```

```
## S3 method for class 'matrix'
as.raw_mspct(x, w.length, spct.data.var = "counts",
  multiplier = 1, byrow = NULL, spct.names = "spct_", ...)
```

Arguments

x	a list of spectral objects or a list of objects such as data frames that can be converted into spectral objects.
...	passed to individual spectrum object constructor
ncol	integer Number of 'virtual' columns in data
byrow	logical If ncol > 1 how to read in the data
w.length	numeric A vector of wavelengthvalues sorted in strictly ascending order (nm).
spct.data.var	character The name of the variable that will contain the spectral data. This indicates what physical quantity is stored in the matrix and the units of expression used.
multiplier	numeric A multiplier to be applied to the values in x to do unit or scale conversion.
spct.names	character Vector of names to be assigned to collection members, either of length 1, or with length equal to the number of spectra.

Value

A copy of x converted into a raw_mspct object.

Methods (by class)

- default:
- data.frame:
- raw_spct:
- list:
- matrix:

Note

When x is a square matrix an explicit argument is needed for byrow to indicate how data in x should be read. In every case the length of the w.length vector must match one of the dimensions of x.

See Also

Other Coercion methods for collections of spectra: as.calibration_mspct, as.chroma_mspct, as.cps_mspct, as.filter_mspct, as.generic_mspct, as.object_mspct, as.reflector_mspct, as.response_mspct, as.source_mspct, split2mspct, subset2mspct

as.raw_spct *Coerce to a spectrum*

Description

Return a copy of an R object with its class set to a given type of spectrum.

Usage

```
as.raw_spct(x, ...)
```

Arguments

x	an R object
...	other arguments passed to "set" functions

Value

A copy of x converted into a raw_spct object.

See Also

setGenericSpct

Other constructors of spectral objects: as.calibration_spct, as.chroma_spct, as.cps_spct, as.filter_spct, as.generic_spct, as.object_spct, as.reflector_spct, as.response_spct, as.source_spct, source_spct

as.reflector_mspct *Coerce to a collection-of-spectra*

Description

Return a copy of an R object with its class set to a given type of spectrum.

Usage

```
as.reflector_mspct(x, ...)

## Default S3 method:
as.reflector_mspct(x, ...)

## S3 method for class 'data.frame'
as.reflector_mspct(x, Rfr.type = c("total",
  "specular"), strict.range = TRUE, ...)
```

```
## S3 method for class 'reflector_spct'
as.reflector_mspct(x, ...)

## S3 method for class 'list'
as.reflector_mspct(x, Rfr.type = c("total", "specular"),
  strict.range = TRUE, ..., ncol = 1, byrow = FALSE)

## S3 method for class 'matrix'
as.reflector_mspct(x, w.length, spct.data.var = "Rfr",
  multiplier = 1, byrow = NULL, spct.names = "spct_", ...)
```

Arguments

x	a list of spectral objects or a list of objects such as data frames that can be converted into spectral objects.
...	passed to individual spectrum object constructor
Rfr.type	a character string, either "total" or "specular"
strict.range	logical Flag indicating how off-range values are handled
ncol	integer Number of 'virtual' columns in data
byrow	logical If ncol > 1 how to read in the data
w.length	numeric A vector of wavelengthvalues sorted in strictly ascending order (nm).
spct.data.var	character The name of the variable that will contain the spectral data. This indicates what physical quantity is stored in the matrix and the units of expression used.
multiplier	numeric A multiplier to be applied to the values in x to do unit or scale conversion.
spct.names	character Vector of names to be assigned to collection members, either of length 1, or with length equal to the number of spectra.

Value

A copy of x converted into a reflector_mspct object.

Methods (by class)

- default:
- data.frame:
- reflector_spct:
- list:
- matrix:

Note

When x is a square matrix an explicit argument is needed for byrow to indicate how data in x should be read. In every case the length of the w.length vector must match one of the dimensions of x.

See Also

Other Coercion methods for collections of spectra: as.calibration_mspct, as.chroma_mspct, as.cps_mspct, as.filter_mspct, as.generic_mspct, as.object_mspct, as.raw_mspct, as.response_mspct, as.source_mspct, split2mspct, subset2mspct

---

as.reflector_spct *Coerce to a spectrum*

Description

Return a copy of an R object with its class set to a given type of spectrum.

Usage

```
as.reflector_spct(x, ...)

## Default S3 method:
as.reflector_spct(x, Rfr.type = c("total", "specular"),
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  ...)
```

Arguments

x	an R object
...	other arguments passed to "set" functions
Rfr.type	a character string, either "total" or "specular"
strict.range	logical Flag indicating whether off-range values result in an error instead of a warning

Value

A copy of x converted into a reflector_spct object.

Methods (by class)

- default:

See Also

setGenericSpct

Other constructors of spectral objects: as.calibration_spct, as.chroma_spct, as.cps_spct, as.filter_spct, as.generic_spct, as.object_spct, as.raw_spct, as.response_spct, as.source_spct, source_spct

as.response_mspct *Coerce to a collection-of-spectra*

Description

Return a copy of an R object with its class set to a given type of spectrum.

Usage

```
as.response_mspct(x, ...)

## Default S3 method:
as.response_mspct(x, ...)

## S3 method for class 'data.frame'
as.response_mspct(x, time.unit = "second", ...)

## S3 method for class 'response_spct'
as.response_mspct(x, ...)

## S3 method for class 'list'
as.response_mspct(x, time.unit = "second", ...,
  ncol = 1, byrow = FALSE)

## S3 method for class 'matrix'
as.response_mspct(x, w.length,
  spct.data.var = "s.e.response", multiplier = 1, byrow = NULL,
  spct.names = "spct_", ...)
```

Arguments

x	a list of spectral objects or a list of objects such as data frames that can be converted into spectral objects.
...	passed to individual spectrum object constructor
time.unit	character A string, "second", "day" or "exposure"
ncol	integer Number of 'virtual' columns in data
byrow	logical If ncol > 1 how to read in the data
w.length	numeric A vector of wavelengthvalues sorted in strictly ascending order (nm).
spct.data.var	character The name of the variable that will contain the spectral data. This indicates what physical quantity is stored in the matrix and the units of expression used.
multiplier	numeric A multiplier to be applied to the values in x to do unit or scale conversion.
spct.names	character Vector of names to be assigned to collection members, either of length 1, or with length equal to the number of spectra.

Value

A copy of x converted into a response_mspct object.

Methods (by class)

- default:
- data.frame:
- response_spct:
- list:
- matrix:

Note

When x is a square matrix an explicit argument is needed for byrow to indicate how data in x should be read. In every case the length of the w.length vector must match one of the dimensions of x.

See Also

Other Coercion methods for collections of spectra: as.calibration_mspct, as.chroma_mspct, as.cps_mspct, as.filter_mspct, as.generic_mspct, as.object_mspct, as.raw_mspct, as.reflector_mspct, as.source_mspct, split2mspct, subset2mspct

---

as.response_spct *Coerce to a spectrum*

Description

Return a copy of an R object with its class set to a given type of spectrum.

Usage

```
as.response_spct(x, ...)

## Default S3 method:
as.response_spct(x, time.unit = "second", ...)
```

Arguments

x	an R object
...	other arguments passed to "set" functions
time.unit	character A string, "second", "day" or "exposure"

Value

A copy of x converted into a response_spct object.

Methods (by class)

- default:

See Also

setGenericSpct

Other constructors of spectral objects: as.calibration_spct, as.chroma_spct, as.cps_spct, as.filter_spct, as.generic_spct, as.object_spct, as.raw_spct, as.reflector_spct, as.source_spct, source_spct

---

as.solar_date *Convert a solar_time object into solar_date object*

Description

Convert a solar_time object into solar_date object

Usage

```
as.solar_date(x, time)
```

Arguments

x	solar_time object.
time	an R date time object

Value

For method as.solar_date() a date-time object with the class attr set to "solar.time". This is needed only for unambiguous formatting and printing.

---

as.source_mspct *Coerce to a collection-of-spectra*

Description

Return a copy of an R object with its class set to a given type of spectrum.

Usage

```
as.source_mspct(x, ...)

## Default S3 method:
as.source_mspct(x, ...)

## S3 method for class 'data.frame'
as.source_mspct(x, time.unit = c("second", "day",
  "exposure"), bswf.used = c("none", "unknown"),
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  ...)

## S3 method for class 'source_spct'
as.source_mspct(x, ...)

## S3 method for class 'list'
as.source_mspct(x, time.unit = c("second", "day",
  "exposure"), bswf.used = c("none", "unknown"),
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  ..., ncol = 1, byrow = FALSE)

## S3 method for class 'matrix'
as.source_mspct(x, w.length,
  spct.data.var = "s.e.irrad", multiplier = 1, byrow = NULL,
  spct.names = "spct_", ...)
```

Arguments

x	a list of spectral objects or a list of objects such as data frames that can be converted into spectral objects.
...	passed to individual spectrum object constructor
time.unit	character A string, "second", "day" or "exposure"
bswf.used	character
strict.range	logical Flag indicating how off-range values are handled
ncol	integer Number of 'virtual' columns in data
byrow	logical If ncol > 1 how to read in the data
w.length	numeric A vector of wavelengthvalues sorted in strictly ascending order (nm).
spct.data.var	character The name of the variable that will contain the spectral data. This indicates what physical quantity is stored in the matrix and the units of expression used.
multiplier	numeric A multiplier to be applied to the values in x to do unit or scale conversion.
spct.names	character Vector of names to be assigned to collection members, either of length 1, or with length equal to the number of spectra.

Value

A copy of x converted into a source_mspct object.

Methods (by class)

- default:
- data.frame:
- source_spct:
- list:
- matrix:

Note

When x is a square matrix an explicit argument is needed for byrow to indicate how data in x should be read. In every case the length of the w.length vector must match one of the dimensions of x.

See Also

Other Coercion methods for collections of spectra: as.calibration_mspct, as.chroma_mspct, as.cps_mspct, as.filter_mspct, as.generic_mspct, as.object_mspct, as.raw_mspct, as.reflector_mspct, as.response_mspct, split2mspct, subset2mspct

---

as.source_spct *Coerce to a spectrum*

Description

Return a copy of an R object with its class set to a given type of spectrum.

Usage

```
as.source_spct(x, ...)

## Default S3 method:
as.source_spct(x, time.unit = c("second", "day",
  "exposure"), bswf.used = c("none", "unknown"),
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  ...)
```

Arguments

x	an R object
...	other arguments passed to "set" functions
time.unit	character A string, "second", "day" or "exposure"
bswf.used	character
strict.range	logical Flag indicating whether off-range values result in an error instead of a warning

Value

A copy of x converted into a source_spct object.

Methods (by class)

- default:

See Also

setGenericSpct

Other constructors of spectral objects: as.calibration_spct, as.chroma_spct, as.cps_spct, as.filter_spct, as.generic_spct, as.object_spct, as.raw_spct, as.reflector_spct, as.response_spct, source_spct

---

as_energy *Convert spectral photon irradiance into spectral energy irradiance*

Description

Convert a spectral photon irradiance [mol s-1 m-2 nm-1] into a spectral energy irradiance [W m-2 nm-1].

Usage

```
as_energy(w.length, s.qmol.irrad)
```

Arguments

w.length	numeric vector of wavelengths (nm).
s.qmol.irrad	numeric vector of spectral photon irradiance values.

Value

A numeric vector of spectral (energy) irradiances.

See Also

Other low-level functions operating on numeric vectors.: as_quantum_mol, calc_multipliers, div_spectra, energy_irradiance, energy_ratio, insert_hinges, integrate_xy, interpolate_spectrum, irradiance, l_insert_hinges, oper_spectra, photon_irradiance, photon_ratio, photons_energy_ratio, prod_spectra, s_e_irrad2rgb, split_energy_irradiance, split_photon_irradiance, subt_spectra, sum_spectra, trim_tails, v_insert_hinges, v_replace_hinges

Examples

```
with(sun.spct, as_energy(w.length, s.q.irrad))
```

as_quantum *Convert spectral energy irradiance into spectral photon irradiance*

Description

Convert spectral energy irradiance [W m-2 nm-1] into spectral photon irradiance expressed as number of photons [s-1 m-2 nm-1]

Usage

```
as_quantum(w.length, s.e.irrad)
```

Arguments

w.length	numeric vector of wavelengths (nm).
s.e.irrad	numeric vector of spectral (energy) irradiance values.

Value

A numeric vector of spectral photon irradiances.

See Also

Other quantity conversion functions: A2T, T2Afr, T2A, T2T, e2qmol_multipliers, e2quantum_multipliers, e2q, q2e

Examples

```
with(sun.data, as_quantum(w.length, s.e.irrad))
```

as_quantum_mol *Convert spectral energy irradiance into spectral photon irradiance*

Description

Convert spectral energy irradiance [W m-2 nm-1] into a spectral photon irradiance expressed in number of molds of photons [mol s-1 m-2 nm-1].

Usage

```
as_quantum_mol(w.length, s.e.irrad)
```

Arguments

w.length	numeric vector of wavelengths (nm).
s.e.irrad	numeric vector of spectral (energy) irradiance values.

Value

a numeric vector of spectral photon irradiances.

See Also

Other low-level functions operating on numeric vectors.: as_energy, calc_multipliers, div_spectra, energy_irradiance, energy_ratio, insert_hinges, integrate_xy, interpolate_spectrum, irradiance, l_insert_hinges, oper_spectra, photon_irradiance, photon_ratio, photons_energy_ratio, prod_spectra, s_e_irrad2rgb, split_energy_irradiance, split_photon_irradiance, subt_spectra, sum_spectra, trim_tails, v_insert_hinges, v_replace_hinges

Examples

```
with(sun.data, as_quantum_mol(w.length, s.e.irrad))
```

---

as_tod *Convert date to time-of-day in hours, minutes or seconds*

Description

Convert date to time-of-day in hours, minutes or seconds

Usage

```
as_tod(x, unit.out = "hours", tz = NULL)
```

Arguments

x	a datetime object accepted by lubridate functions
unit.out	character string, One of "datetime", "hour", "minute", or "second".
tz	character string indicating time zone to be used in output.

---

average_spct *Average spectral data.*

Description

This function gives the result of integrating spectral data over wavelengths and dividing the result by the spread or span of the wavelengths.

Usage

```
average_spct(spct)
```

Arguments

spct	generic_spct

Value

One or more numeric values with no change in scale factor: e.g. [W m-2 nm-1] -> [W m-2 nm-1]. Each value in the returned vector corresponds to a variable in the spectral object, except for wavelength.

Examples

```
average_spct(sun.spct)
```

---

beesxyzCMF.spct *Honeybee xyz chromaticity colour matching function data*

Description

A dataset containing wavelengths at a 5 nm interval (300 nm to 700 nm) and the corresponding x, y, and z chromaticity coordinates. Original data from XXX.

A chroma_spct object with variables as follows:

Usage

```
beesxyzCMF.spct
```

Format

A data frame with 81 rows and 4 variables

Details

- w.length (nm)
- x
- y
- z

See Also

Other Visual response data examples: ciev10.spct, ciev2.spct, ciexyzCC10.spct, ciexyzCC2.spct, ciexyzCMF10.spct, ciexyzCMF2.spct

black_body.spct *Theoretical black body*

Description

A dataset for a hypothetical object with transmittance 0/1 (0%), reflectance 0/1 (0%)

Format

A `object_spct` object with 4 rows and 3 variables

Details

- w.length (nm)
- Tfr (0..1)
- Rfr (0..1)

See Also

Other Spectral data examples: `A.illuminant.spct`, `D65.illuminant.spct`, `Ler_leaf.spct`, `Ler_leaf_rflt.spct`, `Ler_leaf_trns.spct`, `Ler_leaf_trns_i.spct`, `ccd.spct`, `clear.spct`, `clear_body.spct`, `filter_cps.mspct`, `green_leaf.spct`, `opaque.spct`, `photodiode.spct`, `polyester.spct`, `sun.daily.data`, `sun.daily.spct`, `sun.data`, `sun.spct`, `white_body.spct`, `white_led.cps_spct`, `white_led.raw_spct`, `white_led.source_spct`, `yellow_gel.spct`

c *Combine collections of spectra*

Description

Combine two or more generic_mspct objects into a single object.

Usage

```
## S3 method for class 'generic_mspct'
c(..., recursive = FALSE, ncol = 1,
  byrow = FALSE)
```

Arguments

`...`	one or more generic_mspct objects to combine.
`recursive`	logical ignored as nesting of collections of spectra is not supported.
`ncol`	numeric Virtual number of columns
`byrow`	logical When object has two dimensions, how to map member objects to columns and rows.

Value

A collection of spectra object belonging to the most derived class shared among the combined objects.

`calc_multipliers` *Spectral weights*

Description

Calculate multipliers for selecting a range of wavelengths and optionally applying a biological spectral weighting function (BSWF) and wavelength normalization. This function returns numeric multipliers that can be used to select a waveband and apply a weight.

Usage

```
calc_multipliers(w.length, w.band, unit.out = "energy",
  unit.in = "energy", use.cached.mult = FALSE, fill = 0)
```

Arguments

`w.length`	numeric vector of wavelengths (nm).
`w.band`	waveband object.
`unit.out`	character A string: "photon" or "energy", default is "energy".
`unit.in`	character A string: "photon" or "energy", default is "energy".
`use.cached.mult`	logical Flag indicating whether multiplier values should be cached between calls.
`fill`	numeric If `fill` == `NA` then values returned for wavelengths outside the range of the waveband are set to `NA`.

Value

a numeric vector of multipliers of the same length as `w.length`.

See Also

Other low-level functions operating on numeric vectors.: `as_energy`, `as_quantum_mol`, `div_spectra`, `energy_irradiance`, `energy_ratio`, `insert_hinges`, `integrate_xy`, `interpolate_spectrum`, `irradiance`, `l_insert_hinges`, `oper_spectra`, `photon_irradiance`, `photon_ratio`, `photons_energy_ratio`, `prod_spectra`, `s_e_irrad2rgb`, `split_energy_irradiance`, `split_photon_irradiance`, `subt_spectra`, `sum_spectra`, `trim_tails`, `v_insert_hinges`, `v_replace_hinges`

Examples

```
with(sun.data, calc_multipliers(w.length, new_waveband(400,700),"photon"))
with(sun.data, calc_multipliers(w.length, new_waveband(400,700),"photon"), use.cached.mult = TRUE)
```

`calc_source_output` *Scaled and/or interpolated light-source spectral output*

Description

Values calculated by interpolation from user-supplied spectral emission data or by name for light source data included in the packages photobiologySun, photobiologyLamps, or photobiologyLEDs, optionally re-scaling the spectral data values.

Usage

```
calc_source_output(w.length.out, w.length.in, s.irrad.in,
  unit.in = "energy", scaled = NULL, fill = NA, ...)
```

Arguments

`w.length.out`	numeric vector of wavelengths (nm) for output.
`w.length.in`	numeric vector of wavelengths (nm) for input.
`s.irrad.in`	numeric vector of spectral transmittance value (fractions or percent).
`unit.in`	a character string "energy" or "photon".
`scaled`	NULL, "peak", "area"; div ignored if !is.null(scaled).
`fill`	if NA, no extrapolation is done, and NA is returned for wavelengths outside the range of the input. If NULL then the tails are deleted. If 0 then the tails are set to zero.
`...`	Additional arguments passed to `spline` if called.

Value

a source_spct with three numeric vectors with wavelength values (w.length), scaled and interpolated spectral energy irradiance (s.e.irrad), scaled and interpolated spectral photon irradiance values (s.q.irrad).

Note

This is a convenience function that adds no new functionality but makes it a little easier to plot lamp spectral emission data consistently. It automates interpolation, extrapolation/trimming and scaling.

Examples

```
with(sun.data,
     calc_source_output(290:1100,
                        w.length.in = w.length,
                        s.irrad.in = s.e.irrad)
    )
```

ccd.spct *Spectral response of a back-thinned CCD image sensor.*

Description

A dataset containing wavelengths at a 1 nm interval and spectral response as quantum efficiency for CCD sensor type S11071/S10420 from Hamamatsu (measured without a quartz window). These vectors are frequently used as sensors in high-UV-sensitivity vector spectrometers. Data digitized from manufacturer's data sheet. The original data is expressed as percent quantum efficiency with a value of 77% at the peak. The data have been re-expressed as fractions of one.

Usage

```
ccd.spct
```

Format

A `response_spct` object with 186 rows and 2 variables

Details

- w.length (nm).
- s.q.response (fractional quantum efficiency)

References

Hamamatsu (2014) Datasheet: CCD Image Sensors S11071/S10420-01 Series. Hamamatsu Photonics KK, Hamamatsu, City. http://www.hamamatsu.com/jp/en/S11071-1004.html. Visited 2017-12-15.

See Also

Other Spectral data examples: `A.illuminant.spct`, `D65.illuminant.spct`, `Ler_leaf.spct`, `Ler_leaf_rflt.spct`, `Ler_leaf_trns.spct`, `Ler_leaf_trns_i.spct`, `black_body.spct`, `clear.spct`, `clear_body.spct`, `filter_cps.mspct`, `green_leaf.spct`, `opaque.spct`, `photodiode.spct`, `polyester.spct`, `sun.daily.data`, `sun.daily.spct`, `sun.data`, `sun.spct`, `white_body.spct`, `white_led.cps_spct`, `white_led.raw_spct`, `white_led.source_spct`, `yellow_gel.spct`

Examples

```
ccd.spct
```

checkTimeUnit *Check the "time.unit" attribute of an existing source_spct object*

Description

Function to read the "time.unit" attribute

Usage

```
checkTimeUnit(x)
```

Arguments

x	a source_spct object

Value

x possibly with the `time.unit` attribute modified

Note

if x is not a `source_spct` or a `response_spct` object, NA is returned

See Also

Other time attribute functions: `convertTimeUnit`, `getTimeUnit`, `setTimeUnit`

check_spct *Check validity of spectral objects*

Description

Check that an R object contains the expected data members.

Usage

```
check_spct(x, byref, strict.range, ...)

## Default S3 method:
check_spct(x, byref = FALSE, strict.range = NA, ...)

## S3 method for class 'generic_spct'
check_spct(x, byref = TRUE, strict.range = NA,
  multiple.wl = getMultipleWl(x), ...)

## S3 method for class 'calibration_spct'
```

```
check_spct(x, byref = TRUE,
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  multiple.wl = getMultipleWl(x), ...)

## S3 method for class 'raw_spct'
check_spct(x, byref = TRUE,
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  multiple.wl = getMultipleWl(x), ...)

## S3 method for class 'cps_spct'
check_spct(x, byref = TRUE,
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  multiple.wl = getMultipleWl(x), ...)

## S3 method for class 'filter_spct'
check_spct(x, byref = TRUE,
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  multiple.wl = getMultipleWl(x), ...)

## S3 method for class 'reflector_spct'
check_spct(x, byref = TRUE,
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  multiple.wl = getMultipleWl(x), ...)

## S3 method for class 'object_spct'
check_spct(x, byref = TRUE,
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  multiple.wl = getMultipleWl(x), ...)

## S3 method for class 'response_spct'
check_spct(x, byref = TRUE, strict.range = NA,
  multiple.wl = getMultipleWl(x), ...)

## S3 method for class 'source_spct'
check_spct(x, byref = TRUE,
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  multiple.wl = getMultipleWl(x), ...)

## S3 method for class 'chroma_spct'
check_spct(x, byref = TRUE,
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  multiple.wl = getMultipleWl(x), ...)
```

Arguments

x	An R object
byref	logical indicating if new object will be created by reference or by copy of x
strict.range	logical indicating whether off-range values result in an error instead of a warn-

	ing, NA disables the test.
...	additional param possible in derived methods
multiple.wl	numeric Maximum number of repeated w.length entries with same value.

Methods (by class)

- default: Default for generic function.
- generic_spct: Specialization for generic_spct.
- calibration_spct: Specialization for calibration_spct.
- raw_spct: Specialization for raw_spct.
- cps_spct: Specialization for cps_spct.
- filter_spct: Specialization for filter_spct.
- reflector_spct: Specialization for reflector_spct.
- object_spct: Specialization for object_spct.
- response_spct: Specialization for response_spct.
- source_spct: Specialization for source_spct.
- chroma_spct: Specialization for chroma_spct.

See Also

Other data validity check functions: check_spectrum, check_w.length

Examples

```
check_spct(sun.spct)

check_spct(sun.spct)
# try(check_spct(-sun.spct))
# try(check_spct((sun.spct[1, "w.length"] <- 1000)))
```

---

check_spectrum *Sanity check a spectrum*

Description

Checks spectral irradiance data in numeric vectors for compliance with assumptions used in calculations.

Usage

```
check_spectrum(w.length, s.irrad)
```

Arguments

w.length	numeric vector of wavelengths (nm).
s.irrad	numeric Corresponding vector of spectral (energy) irradiances (W m-2 nm-1).

Value

A single logical value indicating whether test was passed or not

See Also

Other data validity check functions: check_spct, check_w.length

Examples

```
with(sun.data, check_spectrum(w.length, s.e.irrad))
```

---

check_w.length *Sanity check of wavelengths (internal function).*

Description

This function checks a w.length vector for compliance with assumptions used in calculations.

Usage

```
check_w.length(w.length)
```

Arguments

w.length	numeric array of wavelength (nm)

Value

a single logical value indicating whether test was passed or not

See Also

Other data validity check functions: check_spct, check_spectrum

Examples

```
with(sun.data, photobiology:::check_w.length(w.length))
```

`ciev10.spct` *Linear energy CIE 2008 luminous efficiency function 10 deg data*

Description

A dataset containing wavelengths at a 1 nm interval (390 nm to 830 nm) and the corresponding response values for a 10 degrees target. Original data from http://www.cvrl.org/ downloaded on 2014-04-29 The variables are as follows:

- w.length (nm)
- x
- y
- z

Usage

```
ciev10.spct
```

Format

A chroma_spct object with 441 rows and 4 variables

Author(s)

CIE

See Also

Other Visual response data examples: `beesxyzCMF.spct`, `ciev2.spct`, `ciexyzCC10.spct`, `ciexyzCC2.spct`, `ciexyzCMF10.spct`, `ciexyzCMF2.spct`

Examples

```
ciev10.spct
```

`ciev2.spct` *Linear energy CIE 2008 luminous efficiency function 2 deg data*

Description

A dataset containing wavelengths at a 1 nm interval (390 nm to 830 nm) and the corresponding response values for a 2 degrees target. Original data from http://www.cvrl.org/ downloaded on 2014-04-29 The variables are as follows:

Usage

```
ciev2.spct
```

Format

A chroma_spct object with 441 rows and 4 variables

Details

- w.length (nm)
- x
- y
- z

Author(s)

CIE

See Also

Other Visual response data examples: `beesxyzCMF.spct`, `ciev10.spct`, `ciexyzCC10.spct`, `ciexyzCC2.spct`, `ciexyzCMF10.spct`, `ciexyzCMF2.spct`

Examples

```
ciev2.spct
```

`ciexyzCC10.spct` *CIE xyz chromaticity coordinates (CC) 10 deg data*

Description

A dataset containing wavelengths at a 1 nm interval (390 nm to 830 nm) and the corresponding x, y, and z chromaticity coordinates. Derived from proposed CIE 2006 standard. Original data from http://www.cvrl.org/ downloaded on 2014-04-29 The variables are as follows:

- w.length (nm)
- x
- y
- z

Usage

```
ciexyzCC10.spct
```

Format

A chroma_spct object with 441 rows and 4 variables

Author(s)

CIE

See Also

Other Visual response data examples: `beesxyzCMF.spct`, `ciev10.spct`, `ciev2.spct`, `ciexyzCC2.spct`, `ciexyzCMF10.spct`, `ciexyzCMF2.spct`

Examples

```
ciexyzCC10.spct
```

ciexyzCC2.spct *CIE xyz chromaticity coordinates 2 deg data*

Description

A dataset containing wavelengths at a 1 nm interval (390 nm to 830 nm) and the corresponding x, y, and z chromaticity coordinates. According to proposed CIE 2006 standard. Original data from http://www.cvrl.org/ downloaded on 2014-04-28 The variables are as follows:

- w.length (nm)
- x
- y
- z

Usage

```
ciexyzCC2.spct
```

Format

A chroma_spct object with 441 rows and 4 variables

Author(s)

CIE

See Also

Other Visual response data examples: beesxyzCMF.spct, ciev10.spct, ciev2.spct, ciexyzCC10.spct, ciexyzCMF10.spct, ciexyzCMF2.spct

Examples

```
ciexyzCC2.spct
```

`ciexyzCMF10.spct` *Linear energy CIE xyz colour matching function (CMF) 10 deg data*

Description

A dataset containing wavelengths at a 1 nm interval (390 nm to 830 nm) and the corresponding x, y, and z 10 degrees CMF values. Derived from proposed CIE 2006 standard. Original data from http://www.cvrl.org/ downloaded on 2014-04-29 The variables are as follows:

- w.length (nm)
- x
- y
- z

Usage

```
ciexyzCMF10.spct
```

Format

A chroma_spct object with 441 rows and 4 variables

Author(s)

CIE

See Also

Other Visual response data examples: `beesxyzCMF.spct`, `ciev10.spct`, `ciev2.spct`, `ciexyzCC10.spct`, `ciexyzCC2.spct`, `ciexyzCMF2.spct`

Examples

```
ciexyzCMF10.spct
```

ciexyzCMF2.spct *Linear energy CIE xyz colour matching function (CMF) 2 deg data*

Description

A dataset containing wavelengths at a 1 nm interval (390 nm to 830 nm) and the corresponding x, y, and z 2 degrees CMF values. Derived from proposed CIE 2006 standard. Original data from http://www.cvrl.org/ downloaded on 2014-04-29 The variables are as follows:

- w.length (nm)
- x
- y
- z

Usage

```
ciexyzCMF2.spct
```

Format

A chroma_spct object with 441 rows and 4 variables

Author(s)

CIE

See Also

Other Visual response data examples: beesxyzCMF.spct, ciev10.spct, ciev2.spct, ciexyzCC10.spct, ciexyzCC2.spct, ciexyzCMF10.spct

Examples

```
ciexyzCMF2.spct
```

class_spct *Query which is the class of an spectrum*

Description

Functions to check if an object is a generic spectrum, or coerce it if possible.

Usage

```
class_spct(x)
```

Arguments

x	any R object

Value

class_spct returns a vector containing all matching xxxx.spct classes.

Examples

```
class_spct(sun.spct)
class(sun.spct)
```

clean *Clean (=replace) off-range values in a spectrum*

Description

These functions implement the equivalent of replace() but for spectral objects instead of vectors.

Usage

```
clean(x, range, range.s.data, fill, ...)

## Default S3 method:
clean(x, range, range.s.data, fill, ...)

## S3 method for class 'source_spct'
clean(x, range = x, range.s.data = c(0, NA),
  fill = range.s.data,
  unit.out = getOption("photobiology.radiation.unit", default =
  "energy"), ...)

## S3 method for class 'filter_spct'
```

```
clean(x, range = x, range.s.data = NULL,
  fill = range.s.data, qty.out = getOption("photobiology.filter.qty",
  default = "transmittance"), ...)

## S3 method for class 'reflector_spct'
clean(x, range = x, range.s.data = c(0, 1),
  fill = range.s.data, ...)

## S3 method for class 'object_spct'
clean(x, range = x, range.s.data = c(0, 1),
  fill = range.s.data, ...)

## S3 method for class 'response_spct'
clean(x, range = x, range.s.data = c(0, NA),
  fill = range.s.data,
  unit.out = getOption("photobiology.radiation.unit", default =
  "energy"), ...)

## S3 method for class 'cps_spct'
clean(x, range = x, range.s.data = c(0, NA),
  fill = range.s.data, ...)

## S3 method for class 'raw_spct'
clean(x, range = x, range.s.data = c(NA_real_,
  NA_real_), fill = range.s.data, ...)

## S3 method for class 'generic_spct'
clean(x, range = x, range.s.data = c(NA_real_,
  NA_real_), fill = range.s.data, col.names, ...)

## S3 method for class 'source_mspct'
clean(x, range = NULL, range.s.data = c(0, NA),
  fill = range.s.data,
  unit.out = getOption("photobiology.radiation.unit", default =
  "energy"), ..., .parallel = FALSE, .paropts = NULL)

## S3 method for class 'filter_mspct'
clean(x, range = NULL, range.s.data = NULL,
  fill = range.s.data, qty.out = getOption("photobiology.filter.qty",
  default = "transmittance"), ..., .parallel = FALSE, .paropts = NULL)

## S3 method for class 'reflector_mspct'
clean(x, range = NULL, range.s.data = c(0,
  1), fill = range.s.data, ..., .parallel = FALSE, .paropts = NULL)

## S3 method for class 'response_mspct'
clean(x, range = NULL, range.s.data = c(0,
  NA), fill = range.s.data,
```

```
  unit.out = getOption("photobiology.radiation.unit", default =
  "energy"), ..., .parallel = FALSE, .paropts = NULL)

## S3 method for class 'cps_mspct'
clean(x, range = NULL, range.s.data = c(0, NA),
  fill = range.s.data, ..., .parallel = FALSE, .paropts = NULL)

## S3 method for class 'raw_mspct'
clean(x, range = NULL, range.s.data = c(0, NA),
  fill = range.s.data, ..., .parallel = FALSE, .paropts = NULL)

## S3 method for class 'generic_mspct'
clean(x, range = x, range.s.data = c(NA_real_,
  NA_real_), fill = range.s.data, col.names, ..., .parallel = FALSE,
  .paropts = NULL)
```

Arguments

x	an R object
range	numeric vector of wavelengths
range.s.data	numeric vector of length two giving the allowable range for the spectral data.
fill	numeric vector of length 1 or 2, giving the replacement values to use at each extreme of the range.
...	currently ignored
unit.out	character string with allowed values "energy", and "photon", or its alias "quantum"
qty.out	character string with allowed values "energy", and "photon", or its alias "quantum"
col.names	character The name of the variable to clean
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Value

A copy of x, possibly with some of the spectral data values replaced by the value passed to fill.

Methods (by class)

- default: Default for generic function
- source_spct: Replace off-range values in a source spectrum
- filter_spct: Replace off-range values in a filter spectrum
- reflector_spct: Replace off-range values in a reflector spectrum

- `object_spct`: Replace off-range values in an object spectrum
- `response_spct`: Replace off-range values in a response spectrum
- `cps_spct`: Replace off-range values in a counts per second spectrum
- `raw_spct`: Replace off-range values in a raw counts spectrum
- `generic_spct`: Replace off-range values in a generic spectrum
- `source_mspct`:
- `filter_mspct`:
- `reflector_mspct`:
- `response_mspct`:
- `cps_mspct`:
- `raw_mspct`:
- `generic_mspct`:

`clear.spct` *Theoretical spectrum of a clear material*

Description

A dataset for a hypothetical object with transmittance 1/1 (100%)

Usage

```
clear.spct
```

Format

A `filter_spct` object with 4 rows and 2 variables

Details

- w.length (nm).
- Tfr (0..1)

See Also

Other Spectral data examples: `A.illuminant.spct`, `D65.illuminant.spct`, `Ler_leaf.spct`, `Ler_leaf_rflt.spct`, `Ler_leaf_trns.spct`, `Ler_leaf_trns_i.spct`, `black_body.spct`, `ccd.spct`, `clear_body.spct`, `filter_cps.mspct`, `green_leaf.spct`, `opaque.spct`, `photodiode.spct`, `polyester.spct`, `sun.daily.data`, `sun.daily.spct`, `sun.data`, `sun.spct`, `white_body.spct`, `white_led.cps_spct`, `white_led.raw_spct`, `white_led.source_spct`, `yellow_gel.spct`

Examples

```
clear.spct
```

clear_body.spct *Theoretical clear body*

Description

A dataset for a hypothetical object with transmittance 1/1 (100%), reflectance 0/1 (0%)

Format

A `object_spct` object with 4 rows and 3 variables

Details

- w.length (nm)
- Tfr (0..1)
- Rfr (0..1)

See Also

Other Spectral data examples: `A.illuminant.spct`, `D65.illuminant.spct`, `Ler_leaf.spct`, `Ler_leaf_rflt.spct`, `Ler_leaf_trns.spct`, `Ler_leaf_trns_i.spct`, `black_body.spct`, `ccd.spct`, `clear.spct`, `filter_cps.mspct`, `green_leaf.spct`, `opaque.spct`, `photodiode.spct`, `polyester.spct`, `sun.daily.data`, `sun.daily.spct`, `sun.data`, `sun.spct`, `white_body.spct`, `white_led.cps_spct`, `white_led.raw_spct`, `white_led.source_spct`, `yellow_gel.spct`

clip_wl *Clip head and/or tail of a spectrum*

Description

Clip head and tail of a spectrum based on wavelength limits, no interpolation used at range boundaries.

Usage

```
clip_wl(x, range, ...)

## Default S3 method:
clip_wl(x, range, ...)

## S3 method for class 'generic_spct'
clip_wl(x, range = NULL, ...)

## S3 method for class 'generic_mspct'
clip_wl(x, range = NULL, ...)
```

```
## S3 method for class 'waveband'
clip_wl(x, range = NULL, ...)

## S3 method for class 'list'
clip_wl(x, range = NULL, ...)
```

Arguments

x	an R object.
range	a numeric vector of length two, or any other object for which function range() will return range of wavelengths expressed in nanometres.
...	ignored (possibly used by derived methods).

Value

A copy of x, most frequently of a shorter length, and never longer.

Methods (by class)

- default: Default for generic function
- generic_spct: Clip an object of class "generic_spct" or derived.
- generic_mspct: Clip an object of class "generic_mspct" or derived.
- waveband: Clip an object of class "waveband".
- list: Clip a list (of objects of class "waveband").

Note

The condition tested is wl >= range[1] & wl < (range[2] + 1e-13).

See Also

Other trim functions: trim_spct, trim_waveband, trim_wl

Examples

```
clip_wl(sun.spct, range = c(400, 500))
clip_wl(sun.spct, range = c(NA, 500))
clip_wl(sun.spct, range = c(400, NA))
```

color_of *Color of an object*

Description

Equivalent RGB color of an object such as a spectrum, wavelength or waveband.

Usage

```
color_of(x, ...)

## Default S3 method:
color_of(x, ...)

## S3 method for class 'numeric'
color_of(x, type = "CMF", ...)

## S3 method for class 'list'
color_of(x, short.names = TRUE, type = "CMF", ...)

## S3 method for class 'waveband'
color_of(x, short.names = TRUE, type = "CMF", ...)

## S3 method for class 'source_spct'
color_of(x, type = "CMF", ...)

## S3 method for class 'source_mspct'
color_of(x, ..., idx = "spct.idx")

colour_of(x, ...)

color(x, ...)
```

Arguments

x	an R object.
...	ignored (possibly used by derived methods).
type	character telling whether "CMF", "CC", or "both" should be returned.
short.names	logical indicating whether to use short or long names for wavebands
idx	character Name of the column with the names of the members of the collection of spectra.

Value

A color definition in hexadecimal format as a `character` string of 7 characters, "#" followed by the red, blue, and green values in hexadecimal (scaled to 0 ... 255). In the case of the specialization for

`list`, a list of such definitions is returned. In the case of a collection of spectra, a `data.frame` with one column with such definitions and by default an additional column with names of the spectra as index.

Methods (by class)

- `default`: Default method (returns always "black").
- `numeric`: Method that returns Color definitions corresponding to numeric values representing a wavelengths in nm.
- `list`: Method that returns Color of elements in a list.
- `waveband`: Color at midpoint of a `waveband` object.
- `source_spct`:
- `source_mspct`:

Deprecated

Use of color() is deprecated as this wrapper function may be removed in future versions of the package because of name clashes. Use color_of() instead.

Note

When `x` is a list but not a waveband, if a method `color_of` is not available for the class of each element of the list, then `color_of.default` will be called.

Examples

```
wavelengths <- c(300, 420, 500, 600, NA) # nanometres
color_of(wavelengths)
color_of(waveband(c(300,400)))
color_of(list(blue = waveband(c(400,480)), red = waveband(c(600,700))))
color_of(numeric())
color_of(NA_real_)

color_of(sun.spct)
```

---

convertTimeUnit *Convert the "time.unit" attribute of an existing source_spct object*

Description

Function to set the "time.unit" attribute and simultaneously rescaling the spectral data to be expressed in the new time unit. The change is done by reference ('in place')

Usage

```
convertTimeUnit(x, time.unit = NULL, byref = FALSE)
```

Arguments

x	a source_spct object
time.unit	a character string, either "second", "hour", "day", "exposure" or "none", or a lubridate::duration
byref	logical indicating if new object will be created by reference or by copy of x (currently ignored)

Value

x possibly with the time.unit attribute modified

Note

if x is not a source_spct or a response_spct object, or time.unit is NULL x is returned unchanged, if the existing or new time.unit cannot be converted to a duration, then the returned spectrum will contain NAs.

See Also

Other time attribute functions: checkTimeUnit, getTimeUnit, setTimeUnit

Examples

```
my.spct <- sun.spct
my.spct
convertTimeUnit(my.spct, "day")
```

---

convolve_each *Convolve function for collections of spectra*

Description

Convolve function for collections of spectra which applies an operation on all the individual members of the collection(s) of spectra.

Usage

```
convolve_each(e1, e2, oper = `*`, sep = "_", ...)
```

Arguments

e1	an object of class generic_mspct or generic_scpt or numeric
e2	an object of class generic_mspct or generic_scpt or numeric
oper	function, usually but not necessarily an operator with two arguments.
sep	character Used when pasting the names of members of e1 and e2 to form the names of members of the returned collection of spectra.
...	additional arguments passed to oper if present.

Note

At least one of e1 and e2 must be a `generic_mspct` object or derived.

See Also

Other math operators and functions: `MathFun`, `^.generic_spct`, `div-.generic_spct`, `log`, `minus-.generic_spct`, `mod-.generic_spct`, `plus-.generic_spct`, `round`, `sign`, `slash-.generic_spct`, `times-.generic_spct`

---

copy_attributes *Copy attributes*

Description

Copy attributes from `x` to `y`. Methods defined for spectral and waveband objects of classes from package 'photobiology'.

Usage

```
copy_attributes(x, y, which, ...)

## Default S3 method:
copy_attributes(x, y, which = NULL, ...)

## S3 method for class 'generic_spct'
copy_attributes(x, y, which = NULL,
  which.not = NULL, copy.class = FALSE, ...)

## S3 method for class 'generic_mspct'
copy_attributes(x, y, which = NULL,
  which.not = NULL, copy.class = FALSE, ...)

## S3 method for class 'waveband'
copy_attributes(x, y, which = NULL, ...)
```

Arguments

`x, y`	R objects
`which`	character Names of attributes to copy, if NULL all those relevant according to the class of `x` is used as defaul,
`...`	not used
`which.not`	character Names of attributes not to be copied. The names passed here are removed from the list for `which`, which is most useful when we want to modify the default.
`copy.class`	logical If TRUE class attributes are also copied.

Value

A copy of y with additional attributes set.

Methods (by class)

- default: Default for generic function
- generic_spct:
- generic_mspct:
- waveband:

---

cps2irrad *Conversion from counts per second to physical quantities*

Description

Conversion of spectral data expressed as cps into irradiance, transmittance or reflectance.

Usage

```
cps2irrad(x.sample, pre.fun = NULL, ...)

cps2Rfr(x.sample, x.white, x.black = NULL, dyn.range = NULL)

cps2Tfr(x.sample, x.clear, x.opaque = NULL, dyn.range = NULL)
```

Arguments

x.sample, x.clear, x.opaque, x.white, x.black	cps_spct objects.
pre.fun	function A function applied to x.sample before conversion.
...	Additional arguments passed to pre.fun.
dyn.range	numeric The effective dynamic range of the instrument, if NULL it is automatically set based on integration time bracketing.

Value

A source_spct, filter_spct or reflector_spct object containing the spectral values expressed in physical units.

Note

In contrast to other classes defined in package 'photobiology', class "cps_spct" can have more than one column of cps counts in cases where the intention is to merge these values as part of the processing at the time the calibration is applied. However, being these functions the final step in the conversion to physical units, they accept as input only objects with a single "cps" column, as merging is expected to have been already done.

D2.UV586 *Data for typical calibration lamps*

Description

A dataset containing fitted constants to be used as input for function `D2_spectrum`.

Format

A `polynom::polynomial` object with 6 constants.

Details

An object of class `polynom::polynomial`.

Author(s)

Lasse Ylianttila (data)

D2.UV653 *Data for typical calibration lamps*

Description

A dataset containing fitted constants to be used as input for function `D2_spectrum`.

Format

A `polynom::polynomial` object with 6 constants.

Details

An object of class `polynom::polynomial`.

Author(s)

Lasse Ylianttila (data)

D2.UV654 *Data for typical calibration lamps*

Description

A dataset containing fitted constants to be used as input for function `D2_spectrum`.

Format

A `polynom::polynomial` object with 6 constants.

Details

An object of class `polynom::polynomial`.

Author(s)

Lasse Ylianttila (data)

D2_spectrum *Calculate deuterium lamp output spectrum from fitted constants*

Description

Calculate values by means of a nth degree polynomial from user-supplied constants (for example from a lamp calibration certificate).

Usage

```
D2_spectrum(w.length, k = photobiology::D2.UV653, fill = NA_real_)
```

Arguments

`w.length`	numeric vector of wavelengths (nm) for output
`k`	a polynom:polynomial object with n constants for the polynomial
`fill`	if NA, no extrapolation is done, and NA is returned for wavelengths outside the range 190 nm to 450 nm. If NULL then the tails are deleted. If 0 then the tails are set to zero, etc. NA is default.

Value

a dataframe with four numeric vectors with wavelength values (w.length), energy and photon irradiance (s.e.irrad, s.q.irrad) depending on the argument passed to unit.out (s.irrad).

Note

This is function is valid for wavelengths in the range 180 nm to 495 nm, for wavelengths outside this range NAs are returned.

Examples

```
D2_spectrum(200)
D2_spectrum(170:220)
```

D65.illuminant.spct *CIE D65 illuminant data*

Description

A dataset containing wavelengths at a 5 nm interval (300 nm to 830 nm) and the corresponding spectral energy irradiance normalized to 1 at 560 nm. Spectrum approximates the midday solar spectrum at middle latitude as 'corresponds' to the white point of a black body a 6504 K. Original data from http://files.cie.co.at/204.xls downloaded on 2014-07-25 The variables are as follows:

Usage

```
D65.illuminant.spct
```

Format

A source spectrum with 107 rows and 2 variables

Details

- w.length (nm)
- s.e.irrad (rel. units)

Author(s)

CIE

See Also

Other Spectral data examples: A.illuminant.spct, Ler_leaf.spct, Ler_leaf_rflt.spct, Ler_leaf_trns.spct, Ler_leaf_trns_i.spct, black_body.spct, ccd.spct, clear.spct, clear_body.spct, filter_cps.mspct, green_leaf.spct, opaque.spct, photodiode.spct, polyester.spct, sun.daily.data, sun.daily.spct, sun.data, sun.spct, white_body.spct, white_led.cps_spct, white_led.raw_spct, white_led.source_spct, yellow_gel.spct

Examples

```
D65.illuminant.spct
```

---

day_night *Times for sun positions*

Description

Functions for calculating the timing of solar positions, given geographical coordinates and dates. They can be also used to find the time for an arbitrary solar elevation between 90 and -90 degrees by supplying "twilight" angle(s) as argument.

Usage

```
day_night(date = lubridate::now(tzone = "UTC"),
  tz = lubridate::tz(date), geocode = tibble::tibble(lon = 0, lat =
  51.5, address = "Greenwich"), twilight = "none", unit.out = "hours")

day_night_fast(date, tz, geocode, twilight, unit.out)

noon_time(date = lubridate::today(), tz = lubridate::tz(date),
  geocode = tibble::tibble(lon = 0, lat = 51.5, address = "Greenwich"),
  twilight = "none", unit.out = "datetime")

sunrise_time(date = lubridate::today(), tz = lubridate::tz(date),
  geocode = tibble::tibble(lon = 0, lat = 51.5, address = "Greenwich"),
  twilight = "sunlight", unit.out = "datetime")

sunset_time(date = lubridate::today(), tz = lubridate::tz(date),
  geocode = tibble::tibble(lon = 0, lat = 51.5, address = "Greenwich"),
  twilight = "sunlight", unit.out = "datetime")

day_length(date = lubridate::now(), tz = "UTC",
  geocode = tibble::tibble(lon = 0, lat = 51.5, address = "Greenwich"),
  twilight = "sunlight", unit.out = "hours")

night_length(date = lubridate::now(), tz = "UTC",
  geocode = tibble::tibble(lon = 0, lat = 51.5, address = "Greenwich"),
  twilight = "sunlight", unit.out = "hours")
```

Arguments

date	"vector" of POSIXct times or Date objects, any valid TZ is allowed, default is current date at Greenwich.
tz	character vector indicating time zone to be used in output.

geocode	data frame with one or more rows and variables lon and lat as numeric values (degrees). If present, address will be copied to the output.
twilight	character string, one of "none", "rim", "refraction", "sunlight", "civil", "nautical", "astronomical", or a numeric vector of length one, or two, giving solar elevation angle(s) in degrees (negative if below the horizon).
unit.out	character string, One of "datetime", "day", "hour", "minute", or "second".

Details

Twilight names are interpreted as follows. "none": solar elevation = 0 degrees. "rim": upper rim of solar disk at the horizon or solar elevation = -0.53 / 2. "refraction": solar elevation = 0 degrees + refraction correction. "sunlight": upper rim of solar disk corrected for refraction, which is close to the value used by the online NOAA Solar Calculator. "civil": -6 degrees, "naval": -12 degrees, and "astronomical": -18 degrees. Unit names for output are as follows: "day", "hours", "minutes" and "seconds" times for sunrise and sunset are returned as times-of-day since midnight expressed in the chosen unit. "date" or "datetime" return the same times as datetime objects with TZ set (this is much slower than "hours"). Day length and night length are returned as numeric values expressed in hours when ‘"datetime"’ is passed as argument to unit.out. If twilight is a numeric vector of length two, the element with index 1 is used for sunrise and that with index 2 for sunset.

Value

A tibble with variables day, tz, twilight.rise, twilight.set, longitude, latitude, address, sunrise, noon, sunset, daylength, nightlength or the corresponding individual vectors.

noon_time, sunrise_time and sunset_time return a vector of POSIXct times

day_length and night_length return numeric a vector giving the length in hours

Warning

Be aware that R's Date class does not save time zone metadata. This can lead to ambiguities in the current implementation based on time instants. The argument passed to date should be of class POSIXct, in other words an instant in time, from which the correct date will be computed based on the tz argument.

Note

This function is an implementation of Meeus equations as used in NOAAs on-line web calculator, which are very precise and valid for a very broad range of dates. For sunrise and sunset the times are affected by refraction in the atmosphere, which does in turn depend on weather conditions. The effect of refraction on the apparent position of the sun is only an estimate based on "typical" conditions. The more tangential to the horizon is the path of the sun, the larger the effect of refraction is on the times of visual occlusion of the sun behind the horizon—i.e. the largest timing errors occur at high latitudes. The computation is not defined for latitudes 90 and -90 degrees, i.e. at the poles.

There exists a different R implementation of the same algorithms called "AstroCalcPureR" available as function astrocalc4r in package 'fishmethods'. Although the equations used are almost all the same, the function signatures and which values are returned differ. In particular, the present implementation splits the calculation into two separate functions, one returning angles at given instants in time, and a separate one returning the timing of events for given dates.

In the current implementation functions sunrise_time, noon_time, sunset_time and day_length are wrappers on day_night, so if more than one quantity is needed it is preferable to directly call day_night as it will be faster.

night_length returns the length of night-time conditions in one day (00:00:00 to 23:59:59), rather than the length of the night between two consecutive days.

References

The primary source for the algorithm used is the book: Meeus, J. (1998) Astronomical Algorithms, 2 ed., Willmann-Bell, Richmond, VA, USA. ISBN 978-0943396613.

A different implementation is available at https://www.nefsc.noaa.gov/AstroCalc4R/ and in R paclage astrocalc4r. In 'fishmethods' (= 1.11-0) there is a bug in function astrocalc4r() that affects sunrise and sunset times.

An interactive web page using the same algorithms is available at https://www.esrl.noaa.gov/gmd/grad/solcalc/. There are small differences in the returned times compared to our function that seem to be related to the estimation of atmospheric refraction (about 0.1 degrees).

See Also

sun_angles.

Other astronomy related functions: format.solar_time, is.solar_time, print.solar_time, solar_time, sun_angles

Examples

```
library(lubridate)
my.geocode <- data.frame(lat = 60, lon = 25)
day_night(ymd("2015-05-30"), geocode = my.geocode)
day_night(ymd("2015-05-30") + days(1:10), geocode = my.geocode, twilight = "civil")
sunrise_time(ymd("2015-05-30"), geocode = my.geocode)
noon_time(ymd("2015-05-30"), geocode = my.geocode)
sunset_time(ymd("2015-05-30"), geocode = my.geocode)
day_length(ymd("2015-05-30"), geocode = my.geocode)
day_length(ymd("2015-05-30"), geocode = my.geocode, unit.out = "day")
```

---

defunct *Defunct functions and methods*

Description

Functions listed here have been removed or deleted, and temporarily replaced by stubs that report this when they are called.

Usage

```
f_mspct(...)

mutate_mspct(...)

calc_filter_multipliers(...)
```

Arguments

...	ignored

Note

Function `f_mspct` has been renamed `msdply()`.

Function `mutate_mspct` has been renamed `msmsply()`.

Function `calc_filter_multipliers` has been removed.

---

dim.generic_mspct *Dimensions of an Object*

Description

Retrieve or set the dimension of an object.

Usage

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

## S3 replacement method for class 'generic_mspct'
dim(x) <- value
```

Arguments

x	A `generic_mspct` object or of a derived class.
value	Either NULL or a numeric vector, which is coerced to integer (by truncation).

Value

Either NULL or a numeric vector, which is coerced to integer (by truncation).

div-.generic_spct *Arithmetic Operators*

Description

Integer-division operator for generic spectra.

Usage

```
## S3 method for class 'generic_spct'
e1 %/% e2
```

Arguments

e1	an object of class "generic_spct"
e2	an object of class "generic_spct"

See Also

Other math operators and functions: MathFun, ^.generic_spct, convolve_each, log, minus-.generic_spct, mod-.generic_spct, plus-.generic_spct, round, sign, slash-.generic_spct, times-.generic_spct

div_spectra *Divide two spectra, even if the wavelengths values differ*

Description

The wavelength vectors of the two spectra are merged, and the missing spectral values are calculated by interpolation. After this, the two spectral values at each wavelength are operated upon.

Usage

```
div_spectra(w.length1, w.length2 = NULL, s.irrad1, s.irrad2,
  trim = "union", na.rm = FALSE)
```

Arguments

w.length1	numeric vector of wavelength (nm) of denominator.
w.length2	numeric vector of wavelength (nm) of divisor.
s.irrad1	a numeric vector of spectral values of denominator.
s.irrad2	a numeric vector of spectral values of divisor.
trim	a character string with value "union" or "intersection".
na.rm	a logical value, if TRUE, not the default, NAs in the input are replaced with zeros.

Details

If trim=="union" spectral values are calculated for the whole range of wavelengths covered by at least one of the input spectra, and missing values are set in each input spectrum to zero before addition. If trim=="intersection" then the range of wavelengths covered by both input spectra is returned, and the non-overlapping regions discarded. If w.length2==NULL, it is assumed that both spectra are measured at the same wavelengths, and a simple addition is used, ensuring fast calculation.

Value

a dataframe with two numeric variables.

- w.lengthA numeric vector with the wavelengths (nm) obtained by "fusing" w.length1 and w.length2. w.length contains all the unique vales, sorted in ascending order.
- s.irradA numeric vector with the sum of the two spectral values at each wavelength.

See Also

Other low-level functions operating on numeric vectors.: as_energy, as_quantum_mol, calc_multipliers, energy_irradiance, energy_ratio, insert_hinges, integrate_xy, interpolate_spectrum, irradiance, l_insert_hinges, oper_spectra, photon_irradiance, photon_ratio, photons_energy_ratio, prod_spectra, s_e_irrad2rgb, split_energy_irradiance, split_photon_irradiance, subt_spectra, sum_spectra, trim_tails, v_insert_hinges, v_replace_hinges

Examples

```
head(sun.data)
one.data <- with(sun.data, div_spectra(w.length, w.length, s.e.irrad, s.e.irrad))
head(one.data)
tail(one.data)
```

---

e2q *Convert energy-based quantities into photon-based quantities.*

Description

Function that converts spectral energy irradiance into spectral photon irradiance (molar).

Usage

```
e2q(x, action, byref, ...)

## Default S3 method:
e2q(x, action = "add", byref = FALSE, ...)

## S3 method for class 'source_spct'
```

```
e2q(x, action = "add", byref = FALSE, ...)

## S3 method for class 'response_spct'
e2q(x, action = "add", byref = FALSE, ...)

## S3 method for class 'source_mspct'
e2q(x, action = "add", byref = FALSE, ...,
  .parallel = FALSE, .paropts = NULL)

## S3 method for class 'response_mspct'
e2q(x, action = "add", byref = FALSE, ...,
  .parallel = FALSE, .paropts = NULL)
```

Arguments

x	an R object
action	a character string
byref	logical indicating if new object will be created by reference or by copy of x
...	not used in current version
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Methods (by class)

- `default`: Default method
- `source_spct`: Method for spectral irradiance
- `response_spct`: Method for spectral responsiveness
- `source_mspct`: Method for collections of (light) source spectra
- `response_mspct`: Method for collections of response spectra

See Also

Other quantity conversion functions: `A2T`, `T2Afr`, `T2A`, `T2T`, `as_quantum`, `e2qmol_multipliers`, `e2quantum_multipliers`, `q2e`

e2qmol_multipliers *Calculate energy to quantum (mol) multipliers*

Description

Multipliers as a function of wavelength, for converting from energy to photon (quantum) molar units.

Usage

```
e2qmol_multipliers(w.length)
```

Arguments

w.length	numeric Vector of wavelengths (nm)

Value

A numeric vector of multipliers

See Also

Other quantity conversion functions: A2T, T2Afr, T2A, T2T, as_quantum, e2quantum_multipliers, e2q, q2e

Examples

```
with(sun.data, e2qmol_multipliers(w.length))
```

e2quantum_multipliers *Calculate energy to quantum multipliers*

Description

Gives multipliers as a function of wavelength, for converting from energy to photon (quantum) units (number of photons as default, or moles of photons).

Usage

```
e2quantum_multipliers(w.length, molar = FALSE)
```

Arguments

w.length	numeric Vector of wavelengths (nm)
molar	logical Flag indicating whether output should be in moles or numbers

Value

A numeric vector of multipliers

See Also

Other quantity conversion functions: A2T, T2Afr, T2A, T2T, as_quantum, e2qmol_multipliers, e2q, q2e

Examples

```
with(sun.data, e2quantum_multipliers(w.length))
with(sun.data, e2quantum_multipliers(w.length, molar = TRUE))
```

---

energy_as_default *Set spectral-data options*

Description

Set spectral-data related options easily.

Usage

```
energy_as_default()

photon_as_default()

quantum_as_default()

Tfr_as_default()

Afr_as_default()

A_as_default()

unset_radiation_unit_default()

unset_filter_qty_default()

unset_user_defaults()
```

Value

Previous value of the modified option.

energy_irradiance *Calculate (energy) irradiance from spectral irradiance*

Description

Energy irradiance for a waveband from a radiation spectrum, optionally applying a "biological spectral weighting function" or BSWF.

Usage

```
energy_irradiance(w.length, s.irrad, w.band = NULL, unit.in = "energy",
  check.spectrum = TRUE, use.cached.mult = FALSE,
  use.hinges = getOption("photobiology.use.hinges", default = NULL))
```

Arguments

w.length	numeric vector of wavelength (nm).
s.irrad	numeric vector of spectral irradiances, by default as energy (W m-2 nm-1).
w.band	waveband.
unit.in	a character Allowed values "photon" or "energy", default is "energy".
check.spectrum	logical Flag indicating whether to sanity check input data, default is TRUE.
use.cached.mult	logical Flag indicating whether multiplier values should be cached between calls.
use.hinges	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.

Value

A single numeric value with no change in scale factor: [W m-2 nm-1] -> [W m-2].

See Also

Other low-level functions operating on numeric vectors.: as_energy, as_quantum_mol, calc_multipliers, div_spectra, energy_ratio, insert_hinges, integrate_xy, interpolate_spectrum, irradiance, l_insert_hinges, oper_spectra, photon_irradiance, photon_ratio, photons_energy_ratio, prod_spectra, s_e_irrad2rgb, split_energy_irradiance, split_photon_irradiance, subt_spectra, sum_spectra, trim_tails, v_insert_hinges, v_replace_hinges

Examples

```
with(sun.data, energy_irradiance(w.length, s.e.irrad))
with(sun.data, energy_irradiance(w.length, s.e.irrad, new_waveband(400,700)))
```

energy_ratio *Energy:energy ratio*

Description

Energy irradiance ratio between two wavebands for a radiation spectrum.

Usage

```
energy_ratio(w.length, s.irrad, w.band.num = NULL, w.band.denom = NULL,
  unit.in = "energy", check.spectrum = TRUE, use.cached.mult = FALSE,
  use.hinges = NULL)
```

Arguments

w.length	numeric vector of wavelengths (nm).
s.irrad	numeric vector of spectral (energy) irradiances (W m-2 nm-1).
w.band.num	waveband object used to compute the numerator of the ratio.
w.band.denom	waveband object used to compute the denominator of the ratio.
unit.in	character Allowed values "energy", and "photon", or its alias "quantum".
check.spectrum	logical Flag indicating whether to sanity check input data, default is TRUE.
use.cached.mult	logical Flag indicating whether multiplier values should be cached between calls.
use.hinges	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.

Value

a single numeric value giving the unitless ratio.

Note

The default for both w.band parameters is a waveband covering the whole range of w.length.

See Also

Other low-level functions operating on numeric vectors.: as_energy, as_quantum_mol, calc_multipliers, div_spectra, energy_irradiance, insert_hinges, integrate_xy, interpolate_spectrum, irradiance, l_insert_hinges, oper_spectra, photon_irradiance, photon_ratio, photons_energy_ratio, prod_spectra, s_e_irrad2rgb, split_energy_irradiance, split_photon_irradiance, subt_spectra, sum_spectra, trim_tails, v_insert_hinges, v_replace_hinges

Examples

```
with(sun.data,
     energy_ratio(w.length, s.e.irrad, new_waveband(400,500), new_waveband(400,700)))
```

---

eq_ratio *Energy:photon ratio*

Description

This function returns the energy to mole of photons ratio for each waveband and a light source spectrum.

Usage

```
eq_ratio(spct, w.band, wb.trim, use.cached.mult, use.hinges, ...)

## Default S3 method:
eq_ratio(spct, w.band, wb.trim, use.cached.mult,
  use.hinges, ...)

## S3 method for class 'source_spct'
eq_ratio(spct, w.band = NULL,
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.cached.mult = FALSE,
  use.hinges = getOption("photobiology.use.hinges"), ...)

## S3 method for class 'source_mspct'
eq_ratio(spct, w.band = NULL,
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.cached.mult = FALSE,
  use.hinges = getOption("photobiology.use.hinges"), ...,
  attr2tb = NULL, idx = "spct.idx", .parallel = FALSE,
  .paropts = NULL)
```

Arguments

spct	source_spct.
w.band	waveband or list of waveband objects.
wb.trim	logical if TRUE wavebands crossing spectral data boundaries are trimmed, if FALSE, they are discarded.
use.cached.mult	logical Flag telling whether multiplier values should be cached between calls.
use.hinges	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.

...	other arguments (possibly used by derived methods).
attr2tb	character vector, see add_attr2tb for the syntax for attr2tb passed as is to formal parameter col.names.
idx	character Name of the column with the names of the members of the collection of spectra.
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Value

Computed values are ratios between energy irradiance and photon irradiance for a given waveband. A named numeric vector in the case of methods for individual spectra, with one value for each waveband passed to parameter w.band. A data.frame in the case of collections of spectra, containing one column for each waveband object, an index column with the names of the spectra, and optionally additional columns with metadata values retrieved from the attributes of the member spectra.

By default values are only integrated, but depending on the argument passed to parameter quantity they can be re-expressed as relative fractions or percentages. In the case of vector output, names attribute is set to the name of the corresponding waveband unless a named list is supplied in which case the names of the list members are used, with "e:q" prepended. Units [J mol-1].

Methods (by class)

- default: Default for generic function
- source_spct: Method for source_spct objects
- source_mspct: Calculates energy:photon from a source_mspct object.

Note

The last two parameters control speed optimizations. The defaults should be suitable in most cases. If you will use repeatedly the same SWFs on many spectra measured at exactly the same wavelengths you may obtain some speed up by setting use.cached.mult=TRUE. However, be aware that you are responsible for ensuring that the wavelengths are the same in each call, as the only test done is for the length of the w.length vector.

See Also

Other photon and energy ratio functions: e_ratio, q_ratio, qe_ratio

Examples

```
eq_ratio(sun.spct, new_waveband(400,700))
```

Extract *Extract or replace parts of a spectrum*

Description

Just like extraction and replacement with indexes in base R, but preserving the special attributes used in spectral classes and checking for validity of remaining spectral data.

Usage

```
## S3 method for class 'generic_spct'
x[i, j, drop = NULL]

## S3 method for class 'raw_spct'
x[i, j, drop = NULL]

## S3 method for class 'cps_spct'
x[i, j, drop = NULL]

## S3 method for class 'source_spct'
x[i, j, drop = NULL]

## S3 method for class 'response_spct'
x[i, j, drop = NULL]

## S3 method for class 'filter_spct'
x[i, j, drop = NULL]

## S3 method for class 'reflector_spct'
x[i, j, drop = NULL]

## S3 method for class 'object_spct'
x[i, j, drop = NULL]

## S3 method for class 'chroma_spct'
x[i, j, drop = NULL]

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

## S3 replacement method for class 'generic_spct'
x$name <- value
```

Arguments

x	spectral object from which to extract element(s) or in which to replace element(s)

i	index for rows,
j	index for columns, specifying elements to extract or replace. Indices are numeric or character vectors or empty (missing) or NULL. Please, see Extract for more details.
drop	logical. If TRUE the result is coerced to the lowest possible dimension. The default is FALSE unless the result is a single column.
value	A suitable replacement value: it will be repeated a whole number of times if necessary and it may be coerced: see the Coercion section. If NULL, deletes the column if a single column is selected.
name	A literal character string or a name (possibly backtick quoted). For extraction, this is normally (see under 'Environments') partially matched to the names of the object.

Details

These methods are just wrappers on the method for data.frame objects which copy the additional attributes used by these classes, and validate the extracted object as a spectral object. When drop is TRUE and the returned object has only one column, then a vector is returned. If the extracted columns are more than one but do not include `w.length`, a data frame is returned instead of a spectral object.

Value

An object of the same class as `x` but containing only the subset of rows and columns that are selected. See details for special cases.

Note

If any argument is passed to `j`, even `TRUE`, some metadata attributes are removed from the returned object. This is how the extraction operator works with `data.frames` in R. For the time being we retain this behaviour for spectra, but it may change in the future.

See Also

subset and trim_spct

Examples

```
sun.spct[sun.spct$w.length > 400, ]
subset(sun.spct, w.length > 400)

tmp.spct <- sun.spct
tmp.spct[tmp.spct$s.e.irrad < 1e-5 , "s.e.irrad"] <- 0
e2q(tmp.spct[ , c("w.length", "s.e.irrad")]) # restore data consistency!
```

`Extract_mspct` *Extract or replace members of a collection of spectra*

Description

Just like extraction and replacement with indexes for base R lists, but preserving the special attributes used in spectral classes.

Usage

```
## S3 method for class 'generic_mspct'
x[i, drop = NULL]

## S3 replacement method for class 'generic_mspct'
x[i] <- value

## S3 replacement method for class 'generic_mspct'
x$name <- value

## S3 replacement method for class 'generic_mspct'
x[[name]] <- value
```

Arguments

`x`	Collection of spectra object from which to extract member(s) or in which to replace member(s)
`i`	Index specifying elements to extract or replace. Indices are numeric or character vectors. Please, see `Extract` for more details.
`drop`	If TRUE the result is coerced to the lowest possible dimension (see the examples). This only works for extracting elements, not for the replacement.
`value`	A suitable replacement value: it will be repeated a whole number of times if necessary and it may be coerced: see the Coercion section. If NULL, deletes the column if a single column is selected.
`name`	A literal character string or a name (possibly backtick quoted). For extraction, this is normally (see under 'Environments') partially matched to the names of the object.

Details

This method is a wrapper on base R's extract method for lists that sets additional attributes used by these classes.

Value

An object of the same class as `x` but containing only the subset of members that are selected.

e_fluence *Energy fluence*

Description

Energy fluence for one or more wavebands of a light source spectrum and a duration of the exposure.

Usage

```
e_fluence(spct, w.band, exposure.time, scale.factor, wb.trim,
  use.cached.mult, use.hinges, allow.scaled, ...)

## Default S3 method:
e_fluence(spct, w.band, exposure.time, scale.factor,
  wb.trim, use.cached.mult, use.hinges, allow.scaled, ...)

## S3 method for class 'source_spct'
e_fluence(spct, w.band = NULL, exposure.time,
  scale.factor = 1, wb.trim = getOption("photobiology.waveband.trim",
  default = TRUE),
  use.cached.mult = getOption("photobiology.use.cached.mult", default =
  FALSE), use.hinges = NULL, allow.scaled = FALSE, ...)

## S3 method for class 'source_mspct'
e_fluence(spct, w.band = NULL, exposure.time,
  scale.factor = 1, wb.trim = getOption("photobiology.waveband.trim",
  default = TRUE),
  use.cached.mult = getOption("photobiology.use.cached.mult", default =
  FALSE), use.hinges = NULL, allow.scaled = FALSE, ...,
  attr2tb = NULL, idx = "spct.idx", .parallel = FALSE,
  .paropts = NULL)
```

Arguments

spct	an R object
w.band	a list of waveband objects or a waveband object
exposure.time	lubridate::duration object.
scale.factor	numeric vector of length 1, or length equal to that of w.band. Numeric multiplier applied to returned values.
wb.trim	logical if TRUE wavebands crossing spectral data boundaries are trimmed, if FALSE, they are discarded
use.cached.mult	logical indicating whether multiplier values should be cached between calls
use.hinges	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.

allow.scaled	logical indicating whether scaled or normalized spectra as argument to spct are flagged as an error
...	other arguments (possibly ignored)
attr2tb	character vector, see add_attr2tb for the syntax for attr2tb passed as is to formal parameter col.names.
idx	character Name of the column with the names of the members of the collection of spectra.
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Value

One numeric value for each waveband with no change in scale factor, with name attribute set to the name of each waveband unless a named list is supplied in which case the names of the list elements are used. The exposure.time is copied to the output as an attribute. Units are as follows: (J) joules per exposure.

Methods (by class)

- default: Default for generic function
- source_spct: Calculate energy fluence from a source_spct object and the duration of the exposure.
- source_mspct: Calculates energy fluence from a source_mspct object.

Note

The last two parameters control speed optimizations. The defaults should be suitable in most cases. If you will use repeatedly the same SWFs on many spectra measured at exactly the same wavelengths you may obtain some speed up by setting use.cached.mult=TRUE. However, be aware that you are responsible for ensuring that the wavelengths are the same in each call, as the only test done is for the length of the w.length vector.

See Also

Other irradiance functions: e_irrad, fluence, irrad, q_fluence, q_irrad

Examples

```
library(lubridate)
e_fluence(sun.spct, w.band = waveband(c(400,700)),
          exposure.time = lubridate::duration(3, "minutes") )
```

e_irrad *Energy irradiance*

Description

Energy irradiance for one or more wavebands of a light source spectrum.

Usage

```
e_irrad(spct, w.band, quantity, time.unit, scale.factor, wb.trim,
  use.cached.mult, use.hinges, allow.scaled, ...)

## Default S3 method:
e_irrad(spct, w.band, quantity, time.unit, scale.factor,
  wb.trim, use.cached.mult, use.hinges, allow.scaled, ...)

## S3 method for class 'source_spct'
e_irrad(spct, w.band = NULL, quantity = "total",
  time.unit = NULL, scale.factor = 1,
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.cached.mult = getOption("photobiology.use.cached.mult", default =
  FALSE), use.hinges = NULL, allow.scaled = !quantity %in%
  c("average", "mean", "total"), ...)

## S3 method for class 'source_mspct'
e_irrad(spct, w.band = NULL, quantity = "total",
  time.unit = NULL, scale.factor = 1,
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.cached.mult = getOption("photobiology.use.cached.mult", default =
  FALSE), use.hinges = NULL, allow.scaled = !quantity %in%
  c("average", "mean", "total"), ..., attr2tb = NULL, idx = "spct.idx",
  .parallel = FALSE, .paropts = NULL)
```

Arguments

spct	an R object.
w.band	a list of waveband objects or a waveband object.
quantity	character string One of "total", "average" or "mean", "contribution", "contribution.pc", "relative" or "relative.pc".
time.unit	character or lubridate::duration object.
scale.factor	numeric vector of length 1, or length equal to that of w.band. Numeric multiplier applied to returned values.
wb.trim	logical if TRUE wavebands crossing spectral data boundaries are trimmed, if FALSE, they are discarded.
use.cached.mult	logical indicating whether multiplier values should be cached between calls.

use.hinges	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.
allow.scaled	logical indicating whether scaled or normalized spectra as argument to spct are flagged as an error.
...	other arguments (possibly used by derived methods).
attr2tb	character vector, see add_attr2tb for the syntax for attr2tb passed as is to formal parameter col.names.
idx	character Name of the column with the names of the members of the collection of spectra.
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Value

A named numeric vector in the case of methods for individual spectra, with one value for each waveband passed to parameter w.band. A data.frame in the case of collections of spectra, containing one column for each waveband object, an index column with the names of the spectra, and optionally additional columns with metadata values retrieved from the attributes of the member spectra.

By default values are only integrated, but depending on the argument passed to parameter quantity they can be re-expressed as relative fractions or percentages. In the case of vector output, names attribute is set to the name of the corresponding waveband unless a named list is supplied in which case the names of the list members are used. The time.unit attribute is copied from the spectrum object to the output. Units are as follows: If units are absolute and time.unit is second, [W m-2 nm-1] -> [W m-2] If time.unit is day, [J d-1 m-2 nm-1] -> [J m-2]; if units are relative, fraction of one or percent.

Methods (by class)

- default: Default for generic function
- source_spct: Calculates energy irradiance from a source_spct object.
- source_mspct: Calculates energy irradiance from a source_mspct object.

Note

The last two parameters control speed optimizations. The defaults should be suitable in most cases. If you will use repeatedly the same SWFs on many spectra measured at exactly the same wavelengths you may obtain some speed up by setting use.cached.mult=TRUE. However, be aware that you are responsible for ensuring that the wavelengths are the same in each call, as the only test done is for the length of the w.length vector.

See Also

Other irradiance functions: e_fluence, fluence, irrad, q_fluence, q_irrad

Examples

```
e_irrad(sun.spct, waveband(c(400,700)))
e_irrad(sun.spct, split_bands(c(400,700), length.out = 3))
e_irrad(sun.spct, split_bands(c(400,700), length.out = 3),
        quantity = "total")
e_irrad(sun.spct, split_bands(c(400,700), length.out = 3),
        quantity = "average")
e_irrad(sun.spct, split_bands(c(400,700), length.out = 3),
        quantity = "relative")
e_irrad(sun.spct, split_bands(c(400,700), length.out = 3),
        quantity = "relative.pc")
e_irrad(sun.spct, split_bands(c(400,700), length.out = 3),
        quantity = "contribution")
e_irrad(sun.spct, split_bands(c(400,700), length.out = 3),
        quantity = "contribution.pc")
```

---

e_ratio *Energy:energy ratio*

Description

This function returns the photon ratio for a given pair of wavebands of a light source spectrum.

Usage

```
e_ratio(spct, w.band.num, w.band.denom, wb.trim, use.cached.mult,
  use.hinges, ...)

## Default S3 method:
e_ratio(spct, w.band.num, w.band.denom, wb.trim,
  use.cached.mult, use.hinges, ...)

## S3 method for class 'source_spct'
e_ratio(spct, w.band.num = NULL,
  w.band.denom = NULL,
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.cached.mult = FALSE,
  use.hinges = getOption("photobiology.use.hinges"), ...)

## S3 method for class 'source_mspct'
e_ratio(spct, w.band.num = NULL,
  w.band.denom = NULL,
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.cached.mult = FALSE,
  use.hinges = getOption("photobiology.use.hinges"), ...,
  attr2tb = NULL, idx = "spct.idx", .parallel = FALSE,
  .paropts = NULL)
```

Arguments

spct	source_spct
w.band.num	waveband object or a list of waveband objects used to compute the numerator(s) of the ratio(s).
w.band.denom	waveband object or a list of waveband objects used to compute the denominator(s) of the ratio(s).
wb.trim	logical if TRUE wavebands crossing spectral data boundaries are trimmed, if FALSE, they are discarded
use.cached.mult	logical Flag telling whether multiplier values should be cached between calls.
use.hinges	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.
...	other arguments (possibly used by derived methods).
attr2tb	character vector, see add_attr2tb for the syntax for attr2tb passed as is to formal parameter col.names.
idx	character Name of the column with the names of the members of the collection of spectra.
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Value

In the case of methods for individual spectra, a numeric vector of adimensional values giving a energy ratio between integrated energy irradiances for pairs of wavebands, with name attribute set to the name of the wavebands unless a named list of wavebands is supplied in which case the names of the list elements are used, with "(e:e)" appended. A data.frame in the case of collections of spectra, containing one column for each ratio definition, an index column with the names of the spectra, and optionally additional columns with metadata values retrieved from the attributes of the member spectra.

Ratio definitions are "assembled" from the arguments passed to w.band.num and w.band.denom. If both arguments are of equal length, then the wavebands are paired to obtain as many ratios as the number of wavebands in each list. Recycling for wavebands takes place when the number of denominator and numerator wavebands differ.

Methods (by class)

- default: Default for generic function
- source_spct: Method for source_spct objects
- source_mspct: Calculates energy:energy ratio from a source_mspct object.

Note

Recycling for wavebands takes place when the number of denominator and denominator wavebands differ. The last two parameters control speed optimizations. The defaults should be suitable in most cases. If you will use repeatedly the same SWFs on many spectra measured at exactly the same wavelengths you may obtain some speed up by setting `use.cached.mult=TRUE`. However, be aware that you are responsible for ensuring that the wavelengths are the same in each call, as the only test done is for the length of the `w.length` vector.

See Also

Other photon and energy ratio functions: `eq_ratio`, `q_ratio`, `qe_ratio`

Examples

```
e_ratio(sun.spct, new_waveband(400,500), new_waveband(400,700))
```

---

e_response *Energy-based photo-response*

Description

This function returns the mean, total, or contribution of response for each waveband and a response spectrum.

Usage

```
e_response(spct, w.band, quantity, time.unit, scale.factor, wb.trim,
  use.hinges, ...)

## Default S3 method:
e_response(spct, w.band, quantity, time.unit,
  scale.factor, wb.trim, use.hinges, ...)

## S3 method for class 'response_spct'
e_response(spct, w.band = NULL,
  quantity = "total", time.unit = NULL, scale.factor = 1,
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.hinges = getOption("photobiology.use.hinges", default = NULL), ...)

## S3 method for class 'response_mspct'
e_response(spct, w.band = NULL,
  quantity = "total", time.unit = NULL, scale.factor = 1,
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.hinges = getOption("photobiology.use.hinges", default = NULL), ...,
  attr2tb = NULL, idx = "spct.idx", .parallel = FALSE,
  .paropts = NULL)
```

Arguments

spct	an R object.
w.band	waveband or list of waveband objects or a numeric vector of length two. The waveband(s) determine the region(s) of the spectrum that are summarized. If a numeric range is supplied a waveband object is constructed on the fly from it.
quantity	character string One of "total", "average" or "mean", "contribution", "contribution.pc", "relative" or "relative.pc".
time.unit	character or lubridate::duration object.
scale.factor	numeric vector of length 1, or length equal to that of w.band. Numeric multiplier applied to returned values.
wb.trim	logical if TRUE wavebands crossing spectral data boundaries are trimmed, if FALSE, they are discarded.
use.hinges	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.
...	other arguments (possibly used by derived methods).
attr2tb	character vector, see add_attr2tb for the syntax for attr2tb passed as is to formal parameter col.names.
idx	character Name of the column with the names of the members of the collection of spectra.
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Value

A named numeric vector in the case of methods for individual spectra, with one value for each waveband passed to parameter w.band. A data.frame in the case of collections of spectra, containing one column for each waveband object, an index column with the names of the spectra, and optionally additional columns with metadata values retrieved from the attributes of the member spectra.

By default values are only integrated, but depending on the argument passed to parameter quantity they can be re-expressed as relative fractions or percentages. In the case of vector output, names attribute is set to the name of the corresponding waveband unless a named list is supplied in which case the names of the list members are used.

Methods (by class)

- default: Default method for generic function
- response_spct: Method for response spectra.
- response_mspct: Calculates energy response from a response_mspct

Note

The parameter `use.hinges` controls speed optimization. The defaults should be suitable in most cases. Only the range of wavelengths in the wavebands is used and all BSWFs are ignored.

See Also

Other response functions: `q_response`, `response`

Examples

```
e_response(ccd.spct, new_waveband(200,300))
e_response(photodiode.spct)
```

FEL.BN.9101.165 *Data for typical calibration lamps*

Description

A dataset containing fitted constants to be used as input for function `FEL_spectrum`.

Format

A `numeric vector`.

Author(s)

Lasse Ylianttila (data)

FEL_spectrum *Incandescent "FEL" lamp emission spectrum*

Description

Calculate values by means of a nth degree polynomial from user-supplied constants (for example from a lamp calibration certificate).

Usage

```
FEL_spectrum(w.length, k = photobiology::FEL.BN.9101.165,
  fill = NA_real_)
```

Arguments

w.length	numeric vector of wavelengths (nm) for output
k	a numeric vector with n constants for the function
fill	if NA, no extrapolation is done, and NA is returned for wavelengths outside the range 250 nm to 900 nm. If NULL then the tails are deleted. If 0 then the tails are set to zero, etc. NA is default.

Value

a dataframe with four numeric vectors with wavelength values (w.length), energy and photon irradiance (s.e.irrad, s.q.irrad) depending on the argument passed to unit.out (s.irrad).

Note

This is function is valid for wavelengths in the range 250 nm to 900 nm, for wavelengths outside this range NAs are returned.

Examples

```
FEL_spectrum(400)
FEL_spectrum(250:900)
```

---

findMultipleWl *Find repeated w.length values*

Description

Find repeated w.length values

Usage

```
findMultipleWl(x, same.wls = TRUE)
```

Arguments

x	a generic_spct object
same.wls	logical If TRUE all spectra spected to share same w.length values.

Value

integer Number of spectra, guessed from the number of copies of each individual w.length value.

`find_peaks` *Find peaks in a spectrum*

Description

This function finds all peaks (local maxima) in a spectrum, using a user selectable size threshold relative to the tallest peak (global maximum). This a wrapper built on top of function peaks from package splus2R.

Usage

```
find_peaks(x, ignore_threshold = 0, span = 3, strict = TRUE,
  na.rm = FALSE)
```

Arguments

`x`	numeric vector
`ignore_threshold`	numeric value between 0.0 and 1.0 indicating the size threshold below which peaks will be ignored.
`span`	a peak is defined as an element in a sequence which is greater than all other elements within a window of width span centered at that element. The default value is 3, meaning that a peak is bigger than both of its neighbors. Default: 3.
`strict`	logical flag: if TRUE, an element must be strictly greater than all other values in its window to be considered a peak. Default: TRUE.
`na.rm`	logical indicating whether `NA` values should be stripped before searching for peaks.

Value

an object like s.irrad of logical values. Values that are TRUE correspond to local peaks in the data.

Note

This function is a wrapper built on function `peaks` from **splus2R** and handles non-finite (including NA) values differently than `peaks`, instead of giving an error they are replaced with the smallest finite value in `x`.

See Also

`peaks`

Other peaks and valleys functions: `get_peaks`, `peaks`, `valleys`, `wls_at_target`

Examples

```
with(sun.data, w.length[find_peaks(s.e.irrad)])
```

find_wls *Find wavelength values in a spectrum*

Description

Find wavelength values corresponding to a target y value in any spectrum. The name of the column of the spectral data to be used to match the target needs to be passed as argument unless the spectrum contains a single numerical variable in addition to "w.length".

Usage

```
find_wls(x, target = NULL, col.name.x = NULL, col.name = NULL,
  .fun = `<=`, interpolate = FALSE, na.rm = FALSE)
```

Arguments

x	an R object
target	numeric value indicating the spectral quantity value for which wavelengths are to be searched and interpolated if need. The character strings "half.maximum" and "half.range" are also accepted as arguments.
col.name.x	character The name of the column in which to the independent variable is stored. Defaults to "w.length" for objects of class "generic_spct" or derived.
col.name	character The name of the column in which to search for the target value.
.fun	function A binary comparison function or operator.
interpolate	logical Indicating whether the nearest wavelength value in x should be returned or a value calculated by linear interpolation between wavelength values stradling the target.
na.rm	logical indicating whether NA values should be stripped before searching for the target.

Value

A spectrum object of the same class as x with fewer rows, possibly even no rows. If FALSE is passed to interpolate a subset of x is returned, otherwise a new object of the same class containing interpolated wavelenths for the target value is returned.

Note

This function is used internally by method wls_at_target(), and these methods should be preferred in user code and scripts.

fluence *Fluence*

Description

Energy or photon fluence for one or more wavebands of a light source spectrum and a duration of exposure.

Usage

```
fluence(spct, w.band, unit.out, exposure.time, scale.factor, wb.trim,
  use.cached.mult, use.hinges, allow.scaled, ...)

## Default S3 method:
fluence(spct, w.band, unit.out, exposure.time,
  scale.factor, wb.trim, use.cached.mult, use.hinges, allow.scaled, ...)

## S3 method for class 'source_spct'
fluence(spct, w.band = NULL,
  unit.out = getOption("photobiology.radiation.unit", default =
  "energy"), exposure.time, scale.factor = 1,
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.cached.mult = getOption("photobiology.use.cached.mult", default =
  FALSE), use.hinges = NULL, allow.scaled = FALSE, ...)

## S3 method for class 'source_mspct'
fluence(spct, w.band = NULL,
  unit.out = getOption("photobiology.radiation.unit", default =
  "energy"), exposure.time, scale.factor = 1,
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.cached.mult = getOption("photobiology.use.cached.mult", default =
  FALSE), use.hinges = NULL, allow.scaled = FALSE, ...,
  attr2tb = NULL, idx = "spct.idx", .parallel = FALSE,
  .paropts = NULL)
```

Arguments

spct	an R object.
w.band	a list of waveband objects or a waveband object.
unit.out	character string with allowed values "energy", and "photon", or its alias "quantum".
exposure.time	lubridate::duration object.
scale.factor	numeric vector of length 1, or length equal to that of w.band. Numeric multiplier applied to returned values.
wb.trim	logical if TRUE wavebands crossing spectral data boundaries are trimmed, if FALSE, they are discarded.

use.cached.mult
: logical indicating whether multiplier values should be cached between calls.

use.hinges
: logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.

allow.scaled
: logical indicating whether scaled or normalized spectra as argument to spct are flagged as an error.

...
: other arguments (possibly used by derived methods).

attr2tb
: character vector, see add_attr2tb for the syntax for attr2tb passed as is to formal parameter col.names.

idx
: character Name of the column with the names of the members of the collection of spectra.

.parallel
: if TRUE, apply function in parallel, using parallel backend provided by foreach

.paropts
: a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Value

One numeric value for each waveband with no change in scale factor, with name attribute set to the name of each waveband unless a named list is supplied in which case the names of the list elements are used. The time.unit attribute is copied from the spectrum object to the output. Units are as follows: If time.unit is second, [W m-2 nm-1] -> [mol s-1 m-2] If time.unit is day, [J d-1 m-2 nm-1] -> [mol d-1 m-2]

Methods (by class)

- default: Default for generic function
- source_spct: Calculate photon fluence from a source_spct object and the duration of the exposure
- source_mspct: Calculates fluence from a source_mspct object.

Note

The last two parameters control speed optimizations. The defaults should be suitable in most cases. If you will use repeatedly the same SWFs on many spectra measured at exactly the same wavelengths you may obtain some speed up by setting use.cached.mult=TRUE. However, be aware that you are responsible for ensuring that the wavelengths are the same in each call, as the only test done is for the length of the w.length vector.

See Also

Other irradiance functions: e_fluence, e_irrad, irrad, q_fluence, q_irrad

Examples

```
library(lubridate)
fluence(sun.spct,
        w.band = waveband(c(400,700)),
        exposure.time = lubridate::duration(3, "minutes") )
```

---

format.solar_time *Encode in a Common Format*

Description

Format a solar_time object for pretty printing

Usage

```
## S3 method for class 'solar_time'
format(x, ..., sep = ":")
```

Arguments

x	an R object
...	ignored
sep	character used as separator

See Also

Other astronomy related functions: day_night, is.solar_time, print.solar_time, solar_time, sun_angles

---

formatted_range *Compute range and format it*

Description

Compute the range of an R object, and format it as string suitable for printing.

Usage

```
formatted_range(x, na.rm = TRUE, digits = 3, nsmall = 2,
  collapse = "..")
```

Arguments

x	an R object
na.rm	logical, indicating if NA's should be omitted.
digits, nsmall	numeric, passed to same name parameters of format().
collapse	character, passed to same name parameter of paste().

See Also

range, format and paste.

Examples

```
formatted_range(c(1, 3.5, -0.01))
```

---

fscale *Rescale a spectrum using a summary function*

Description

These functions return a spectral object of the same class as the one supplied as argument but with the spectral data rescaled.

Usage

```
fscale(x, ...)

## Default S3 method:
fscale(x, ...)

## S3 method for class 'source_spct'
fscale(x, range = NULL, f = "mean", target = 1,
  unit.out = getOption("photobiology.radiation.unit", default =
  "energy"), ...)

## S3 method for class 'response_spct'
fscale(x, range = NULL, f = "mean",
  target = 1, unit.out = getOption("photobiology.radiation.unit",
  default = "energy"), ...)

## S3 method for class 'filter_spct'
fscale(x, range = NULL, f = "mean", target = 1,
  qty.out = getOption("photobiology.filter.qty", default =
  "transmittance"), ...)

## S3 method for class 'reflector_spct'
```

```
fscale(x, range = NULL, f = "mean",
  target = 1, qty.out = NULL, ...)

## S3 method for class 'raw_spct'
fscale(x, range = NULL, f = "mean", target = 1,
  ...)

## S3 method for class 'cps_spct'
fscale(x, range = NULL, f = "mean", target = 1,
  ...)

## S3 method for class 'generic_spct'
fscale(x, range = NULL, f = "mean",
  target = 1, col.names, ...)

## S3 method for class 'source_mspct'
fscale(x, range = NULL, f = "mean",
  target = 1, unit.out = getOption("photobiology.radiation.unit",
  default = "energy"), ..., .parallel = FALSE, .paropts = NULL)

## S3 method for class 'response_mspct'
fscale(x, range = NULL, f = "mean",
  target = 1, unit.out = getOption("photobiology.radiation.unit",
  default = "energy"), ..., .parallel = FALSE, .paropts = NULL)

## S3 method for class 'filter_mspct'
fscale(x, range = NULL, f = "mean",
  target = 1, qty.out = getOption("photobiology.filter.qty", default =
  "transmittance"), ..., .parallel = FALSE, .paropts = NULL)

## S3 method for class 'reflector_mspct'
fscale(x, range = NULL, f = "mean",
  target = 1, qty.out = NULL, ..., .parallel = FALSE,
  .paropts = NULL)

## S3 method for class 'raw_mspct'
fscale(x, range = NULL, f = "mean", target = 1,
  ..., .parallel = FALSE, .paropts = NULL)

## S3 method for class 'cps_mspct'
fscale(x, range = NULL, f = "mean", target = 1,
  ..., .parallel = FALSE, .paropts = NULL)

## S3 method for class 'generic_mspct'
fscale(x, range = NULL, f = "mean",
  target = 1, col.names, ..., .parallel = FALSE, .paropts = NULL)

## Default S3 method:
```

```
fshift(x, ...)
```

Arguments

x	An R object
...	additional named arguments passed down to f.
range	An R object on which range() returns a numeric vector of length 2 with the limits of a range of wavelengths in nm, with min and max wavelengths (nm)
f	character string "mean" or "total" for scaling so that this summary value becomes 1 for the returned object, or the name of a function taking x as first argument and returning a numeric value.
target	numeric A constant used as target value for scaling.
unit.out	character Allowed values "energy", and "photon", or its alias "quantum"
qty.out	character Allowed values "transmittance", and "absorbance"
col.names	character vector containing the names of columns or variables to which to apply the scaling.
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Value

A copy of x with the original spectral data values replaced with rescaled values, and the "scaled" attribute set to a list describing the scaling applied.

a new object of the same class as x.

a new object of the same class as x.

Methods (by class)

- default: Default for generic function
- source_spct:
- response_spct:
- filter_spct:
- reflector_spct:
- raw_spct:
- cps_spct:
- generic_spct:
- source_mspct:
- response_mspct:
- filter_mspct:

- reflector_mspct:
- raw_mspct:
- cps_mspct:
- generic_mspct:
- default: Default for generic function

See Also

Other rescaling functions: fshift, getNormalized, is_normalized, is_scaled, normalize, setNormalized, setScaled

Examples

```
fscale(sun.spct, f = "mean")
fscale(sun.spct, f = "mean", na.rm = TRUE)
fscale(sun.spct, f = sum)
fscale(sun.spct, f = function(x) {sum(x) / length(x)})
```

---

fshift *Shift the scale of a spectrum using a summary function*

Description

These functions return a spectral object of the same class as the one supplied as argument but with the spectral data on a shifted scale. A range of wavelengths is taken a reference (zero or another numeric constant) and a summary is calculated for this waveband. The difference between the computed and reference value are used to shift the scale so that these two values match in the returned object.

Usage

```
fshift(x, ...)

## S3 method for class 'source_spct'
fshift(x, range = c(min(x), min(x) + 10),
  f = "mean", unit.out = getOption("photobiology.radiation.unit",
  default = "energy"), ...)

## S3 method for class 'response_spct'
fshift(x, range = c(min(x), min(x) + 10),
  f = "mean", unit.out = getOption("photobiology.radiation.unit",
  default = "energy"), ...)

## S3 method for class 'filter_spct'
fshift(x, range = c(min(x), min(x) + 10),
  f = "min", qty.out = getOption("photobiology.filter.qty", default =
```

```
  "transmittance"), ...)

## S3 method for class 'reflector_spct'
fshift(x, range = c(min(x), min(x) + 10),
  f = "min", qty.out = NULL, ...)

## S3 method for class 'source_mspct'
fshift(x, range = c(min(x), min(x) + 10),
  f = "mean", unit.out = getOption("photobiology.radiation.unit",
  default = "energy"), ...)

## S3 method for class 'raw_spct'
fshift(x, range = c(min(x), min(x) + 10),
  f = "mean", qty.out = NULL, ...)

## S3 method for class 'cps_spct'
fshift(x, range = c(min(x), min(x) + 10),
  f = "mean", qty.out = NULL, ...)

## S3 method for class 'generic_spct'
fshift(x, range = c(min(x), min(x) + 10),
  f = "mean", col.names, ...)

## S3 method for class 'response_mspct'
fshift(x, range = c(min(x), min(x) + 10),
  f = "mean", unit.out = getOption("photobiology.radiation.unit",
  default = "energy"), ..., .parallel = FALSE, .paropts = NULL)

## S3 method for class 'filter_mspct'
fshift(x, range = c(min(x), min(x) + 10),
  f = "min", qty.out = getOption("photobiology.filter.qty", default =
  "transmittance"), ..., .parallel = FALSE, .paropts = NULL)

## S3 method for class 'reflector_mspct'
fshift(x, range = c(min(x), min(x) + 10),
  f = "min", qty.out = NULL, ..., .parallel = FALSE,
  .paropts = NULL)

## S3 method for class 'raw_mspct'
fshift(x, range = c(min(x), min(x) + 10),
  f = "min", ..., .parallel = FALSE, .paropts = NULL)

## S3 method for class 'cps_mspct'
fshift(x, range = c(min(x), min(x) + 10),
  f = "min", ..., .parallel = FALSE, .paropts = NULL)

## S3 method for class 'generic_mspct'
fshift(x, range = c(min(x), min(x) + 10),
```

```
f = "min", col.names, ..., .parallel = FALSE, .paropts = NULL)
```

Arguments

x	An R object
...	additional named arguments passed down to f.
range	An R object on which range() returns a numeric vector of length 2 with the limits of a range of wavelengths in nm, with min and max wavelengths (nm)
f	character string "mean", "min" or "max" for scaling so that this summary value becomes the origin of the spectral data scale in the returned object, or the name of a function taking x as first argument and returning a numeric value.
unit.out	character Allowed values "energy", and "photon", or its alias "quantum"
qty.out	character Allowed values "transmittance", and "absorbance"
col.names	character vector containing the names of columns or variables to which to apply the scale shift.
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Value

A copy of x with the spectral data values replaced with values zero-shifted.

Methods (by class)

- source_spct:
- response_spct:
- filter_spct:
- reflector_spct:
- source_mspct:
- raw_spct:
- cps_spct:
- generic_spct:
- response_mspct:
- filter_mspct:
- reflector_mspct:
- raw_mspct:
- cps_mspct:
- generic_mspct:

See Also

Other rescaling functions: fscale, getNormalized, is_normalized, is_scaled, normalize, setNormalized, setScaled

---

generic_mspct *Collection-of-spectra constructor*

Description

Converts a list of spectral objects into a "multi spectrum" object by setting the class attribute of the list of spectra to the corresponding multi-spct class, check that components of the list belong to the expected class.

Usage

```
generic_mspct(l = NULL, class = "generic_spct", ncol = 1,
  byrow = FALSE, dim = c(length(l)%/%ncol, ncol))

calibration_mspct(l = NULL, ncol = 1, byrow = FALSE, ...)

raw_mspct(l = NULL, ncol = 1, byrow = FALSE, ...)

cps_mspct(l = NULL, ncol = 1, byrow = FALSE, ...)

source_mspct(l = NULL, ncol = 1, byrow = FALSE, ...)

filter_mspct(l = NULL, ncol = 1, byrow = FALSE, ...)

reflector_mspct(l = NULL, ncol = 1, byrow = FALSE, ...)

object_mspct(l = NULL, ncol = 1, byrow = FALSE, ...)

response_mspct(l = NULL, ncol = 1, byrow = FALSE, ...)

chroma_mspct(l = NULL, ncol = 1, byrow = FALSE, ...)
```

Arguments

l	list of generic_spct or derived classes
class	character The multi spectrum object class or the expected class for the elements of l
ncol	integer Number of 'virtual' columns in data
byrow	logical If ncol > 1 how to read in the data
dim	integer vector of dimensions
...	ignored

Functions

- `calibration_mspct`: Specialization for collections of `calibration_spct` objects.
- `raw_mspct`: Specialization for collections of `raw_spct` objects.
- `cps_mspct`: Specialization for collections of `cps_spct` objects.
- `source_mspct`: Specialization for collections of `source_spct` objects.
- `filter_mspct`: Specialization for collections of `filter_spct` objects.
- `reflector_mspct`: Specialization for collections of `reflector_spct` objects.
- `object_mspct`: Specialization for collections of `object_spct` objects.
- `response_mspct`: Specialization for collections of `response_spct` objects.
- `chroma_mspct`: Specialization for collections of `chroma_spct` objects.

Note

Setting class = source_spct or class = source_mspct makes no difference

Examples

```
filter_mspct(list(polyester.spct, yellow_gel.spct))
```

`getAfrType` *Get the "Afr.type" attribute*

Description

Function to read the "Afr.type" attribute of an existing filter_spct or object_spct object.

Usage

```
getAfrType(x)
```

Arguments

`x`	a filter_spct or object_spct object

Value

character string

Note

If x is not a `filter_spct` or an `object_spct` object, `NA` is returned.

See Also

Other Afr attribute functions: `setAfrType`

Examples

```
getAfrType(polyester.spct)
```

`getBSWFUsed` *Get the "bswf.used" attribute*

Description

Function to read the "time.unit" attribute of an existing source_spct object

Usage

```
getBSWFUsed(x)
```

Arguments

x	a source_spct object

Value

character string

Note

if x is not a `source_spct` object, NA is returned

See Also

Other BSWF attribute functions: `setBSWFUsed`

Examples

```
getBSWFUsed(sun.spct)
```

getHowMeasured *Get the "how.measured" attribute*

Description

Function to read the "how.measured" attribute of an existing generic_spct or a generic_mspct.

Usage

```
getHowMeasured(x, ...)

## Default S3 method:
getHowMeasured(x, ...)

## S3 method for class 'generic_spct'
getHowMeasured(x, ...)

## S3 method for class 'summary_generic_spct'
getHowMeasured(x, ...)

## S3 method for class 'generic_mspct'
getHowMeasured(x, ..., idx = "spct.idx")
```

Arguments

x	a generic_spct object
...	Allows use of additional arguments in methods for other classes.
idx	character Name of the column with the names of the members of the collection of spectra.

Value

character vector An object containing a description of the data.

Methods (by class)

- default: default
- generic_spct: generic_spct
- summary_generic_spct: summary_generic_spct
- generic_mspct: generic_mspct

Note

The method for collections of spectra returns the a tibble with a column of character strings.

See Also

Other measurement metadata functions: getInstrDesc, getInstrSettings, getWhatMeasured, getWhenMeasured, getWhereMeasured, get_attributes, isValidInstrDesc, isValidInstrSettings, setHowMeasured, setInstrDesc, setInstrSettings, setWhatMeasured, setWhenMeasured, setWhereMeasured, trimInstrDesc, trimInstrSettings

Examples

```
getHowMeasured(sun.spct)
```

---

getIdFactor *Get the "idfactor" attribute*

Description

Function to read the "idfactor" attribute of an existing generic_spct.

Usage

```
getIdFactor(x)
```

Arguments

x	a generic_spct object

Value

character

Note

If x is not a `generic_spct` or an object of a derived class `NA` is returned.

See Also

Other idfactor attribute functions: setIdFactor

Examples

```
getMultipleWl(sun.spct)
```

getInstrDesc *Get the "instr.desc" attribute*

Description

Function to read the "instr.desc" attribute of an existing generic_spct object.

Usage

```
getInstrDesc(x)
```

Arguments

x	a generic_spct object

Value

list (depends on instrument type)

See Also

Other measurement metadata functions: getHowMeasured, getInstrSettings, getWhatMeasured, getWhenMeasured, getWhereMeasured, get_attributes, isValidInstrDesc, isValidInstrSettings, setHowMeasured, setInstrDesc, setInstrSettings, setWhatMeasured, setWhenMeasured, setWhereMeasured, trimInstrDesc, trimInstrSettings

getInstrSettings *Get the "instr.settings" attribute*

Description

Function to read the "instr.settings" attribute of an existing generic_spct object.

Usage

```
getInstrSettings(x)
```

Arguments

x	a generic_spct object

Value

list

See Also

Other measurement metadata functions: getHowMeasured, getInstrDesc, getWhatMeasured, getWhenMeasured, getWhereMeasured, get_attributes, isValidInstrDesc, isValidInstrSettings, setHowMeasured, setInstrDesc, setInstrSettings, setWhatMeasured, setWhenMeasured, setWhereMeasured, trimInstrDesc, trimInstrSettings

---

getMspctVersion *Get the "mspct.version" attribute*

Description

Function to read the "mspct.version" attribute of an existing generic_mspct object.

Usage

```
getMspctVersion(x)
```

Arguments

x	a generic_mspct object

Value

numeric value

Note

if x is not a generic_mspct object, NA is returned, and if it the attribute is missing, zero is returned with a warning.

---

getMultipleWl *Get the "multiple.wl" attribute*

Description

Function to read the "multiple.wl" attribute of an existing generic_spct.

Usage

```
getMultipleWl(x)
```

Arguments

x	a generic_spct object

Value

integer

Note

If x is not a `generic_spct` or an object of a derived class `NA` is returned.

See Also

Other multiple.wl attribute functions: `setMultipleWl`

Examples

```
getMultipleWl(sun.spct)
```

`getNormalized` *Get the "normalized" attribute*

Description

Function to read the "normalized" attribute of an existing generic_spct object.

Usage

```
getNormalized(x)
```

Arguments

x	a generic_spct object

Value

character or numeric or logical

Note

if x is not a `generic_spct` object, `NA` is returned

See Also

Other rescaling functions: `fscale`, `fshift`, `is_normalized`, `is_scaled`, `normalize`, `setNormalized`, `setScaled`

getRfrType *Get the "Rfr.type" attribute*

Description

Function to read the "Rfr.type" attribute of an existing reflector_spct object or object_spct object.

Usage

```
getRfrType(x)
```

Arguments

x	a source_spct object

Value

character string

Note

if x is not a `filter_spct` object, `NA` is returned

See Also

Other Rfr attribute functions: `getScaled`, `setRfrType`

getScaled *Get the "scaled" attribute*

Description

Function to read the "scaled" attribute of an existing generic_spct object.

Usage

```
getScaled(x)
```

Arguments

x	a generic_spct object

Value

logical

Note

if x is not a `filter_spct` object, `NA` is returned

See Also

Other Rfr attribute functions: `getRfrType`, `setRfrType`

`getSpctVersion` *Get the "spct.version" attribute*

Description

Function to read the "spct.version" attribute of an existing generic_spct object.

Usage

```
getSpctVersion(x)
```

Arguments

`x`	a generic_spct object

Value

integer value

Note

if x is not a `generic_spct` object, `NA` is returned, and if it the attribute is missing, zero is returned with a warning.

`getTfrType` *Get the "Tfr.type" attribute*

Description

Function to read the "Tfr.type" attribute of an existing filter_spct or object_spct object.

Usage

```
getTfrType(x)
```

Arguments

`x`	a filter_spct or object_spct object

Value

character string

Note

If x is not a `filter_spct` or an `object_spct` object, `NA` is returned.

See Also

Other Tfr attribute functions: `setTfrType`

Examples

```
getTfrType(polyester.spct)
```

`getTimeUnit` *Get the "time.unit" attribute of an existing source_spct object*

Description

Function to read the "time.unit" attribute

Usage

```
getTimeUnit(x, force.duration = FALSE)
```

Arguments

`x`	a source_spct object
`force.duration`	logical If TRUE a lubridate::duration is returned even if the object attribute is a character string, if no conversion is possible NA is returned.

Value

character string or a lubridate::duration

Note

if x is not a `source_spct` or a `response_spct` object, NA is returned

See Also

Other time attribute functions: `checkTimeUnit`, `convertTimeUnit`, `setTimeUnit`

Examples

```
getTimeUnit(sun.spct)
```

getWhatMeasured *Get the "what.measured" attribute*

Description

Function to read the "what.measured" attribute of an existing generic_spct or a generic_mspct.

Usage

```
getWhatMeasured(x, ...)

## Default S3 method:
getWhatMeasured(x, ...)

## S3 method for class 'generic_spct'
getWhatMeasured(x, ...)

## S3 method for class 'summary_generic_spct'
getWhatMeasured(x, ...)

## S3 method for class 'generic_mspct'
getWhatMeasured(x, ..., idx = "spct.idx")
```

Arguments

x	a generic_spct object
...	Allows use of additional arguments in methods for other classes.
idx	character Name of the column with the names of the members of the collection of spectra.

Value

character vector An object containing a description of the data.

Methods (by class)

- default: default
- generic_spct: generic_spct
- summary_generic_spct: summary_generic_spct
- generic_mspct: generic_mspct

Note

The method for collections of spectra returns the a tibble with a column of character strings.

See Also

Other measurement metadata functions: getHowMeasured, getInstrDesc, getInstrSettings, getWhenMeasured, getWhereMeasured, get_attributes, isValidInstrDesc, isValidInstrSettings, setHowMeasured, setInstrDesc, setInstrSettings, setWhatMeasured, setWhenMeasured, setWhereMeasured, trimInstrDesc, trimInstrSettings

Examples

```
getWhatMeasured(sun.spct)
```

---

getWhenMeasured *Get the "when.measured" attribute*

Description

Function to read the "when.measured" attribute of an existing generic_spct or a generic_mspct.

Usage

```
getWhenMeasured(x, ...)

## Default S3 method:
getWhenMeasured(x, ...)

## S3 method for class 'generic_spct'
getWhenMeasured(x, ...)

## S3 method for class 'summary_generic_spct'
getWhenMeasured(x, ...)

## S3 method for class 'generic_mspct'
getWhenMeasured(x, ..., idx = "spct.idx")
```

Arguments

x	a generic_spct object
...	Allows use of additional arguments in methods for other classes.
idx	character Name of the column with the names of the members of the collection of spectra.

Value

POSIXct An object with date and time.

Methods (by class)

- default: default
- generic_spct: generic_spct
- summary_generic_spct: summary_generic_spct
- generic_mspct: generic_mspct

Note

If x is not a generic_spct or an object of a derived class NA is returned.

The method for collections of spectra returns the a tibble with the correct times in TZ = "UTC".

See Also

Other measurement metadata functions: getHowMeasured, getInstrDesc, getInstrSettings, getWhatMeasured, getWhereMeasured, get_attributes, isValidInstrDesc, isValidInstrSettings, setHowMeasured, setInstrDesc, setInstrSettings, setWhatMeasured, setWhenMeasured, setWhereMeasured, trimInstrDesc, trimInstrSettings

Examples

```
getWhenMeasured(sun.spct)
```

---

getWhereMeasured *Get the "where.measured" attribute*

Description

Function to read the "where.measured" attribute of an existing generic_spct.

Usage

```
getWhereMeasured(x, ...)

## Default S3 method:
getWhereMeasured(x, ...)

## S3 method for class 'generic_spct'
getWhereMeasured(x, ...)

## S3 method for class 'summary_generic_spct'
getWhereMeasured(x, ...)

## S3 method for class 'generic_mspct'
getWhereMeasured(x, ..., idx = "spct.idx")
```

Arguments

x	a generic_spct object
...	Allows use of additional arguments in methods for other classes.
idx	character Name of the column with the names of the members of the collection of spectra.

Value

a data.frame with a single row and at least columns "lon" and "lat".

Methods (by class)

- default: default
- generic_spct: generic_spct
- summary_generic_spct: summary_generic_spct
- generic_mspct: generic_mspct

Note

If x is not a generic_spct or an object of a derived class NA is returned.

See Also

Other measurement metadata functions: getHowMeasured, getInstrDesc, getInstrSettings, getWhatMeasured, getWhenMeasured, get_attributes, isValidInstrDesc, isValidInstrSettings, setHowMeasured, setInstrDesc, setInstrSettings, setWhatMeasured, setWhenMeasured, setWhereMeasured, trimInstrDesc, trimInstrSettings

Examples

```
getWhereMeasured(sun.spct)
```

---

get_attributes *Get the metadata attributes*

Description

Method returning attributes of an object of class generic_spct or derived, or of class waveband. Only attributes defined and/or set by package 'photobiology' for objects of the corresponding class are returned.

Usage

```
get_attributes(x, which, ...)

## S3 method for class 'generic_spct'
get_attributes(x, which = NULL,
  allowed = all.attributes, ...)

## S3 method for class 'source_spct'
get_attributes(x, which = NULL, ...)

## S3 method for class 'filter_spct'
get_attributes(x, which = NULL, ...)

## S3 method for class 'reflector_spct'
get_attributes(x, which = NULL, ...)

## S3 method for class 'object_spct'
get_attributes(x, which = NULL, ...)

## S3 method for class 'waveband'
get_attributes(x, which = NULL, ...)
```

Arguments

x	a generic_spct object.
which	character vector Names of attributes to retrieve.
...	currently ignored
allowed	character vector Names of attributes accepted by which.

Value

Named list of attribute values.

Methods (by class)

- generic_spct: generic_spct
- source_spct: source_spct
- filter_spct: filter_spct
- reflector_spct: reflector_spct
- object_spct: object_spct
- waveband: waveband

See Also

Other measurement metadata functions: getHowMeasured, getInstrDesc, getInstrSettings, getWhatMeasured, getWhenMeasured, getWhereMeasured, isValidInstrDesc, isValidInstrSettings,

setHowMeasured, setInstrDesc, setInstrSettings, setWhatMeasured, setWhenMeasured, setWhereMeasured, trimInstrDesc, trimInstrSettings

---

get_peaks *Get peaks and valleys in a spectrum*

Description

These functions find peaks (local maxima) or valleys (local minima) in a spectrum, using a user selectable size threshold relative to the tallest peak (global maximum). This a wrapper built on top of function peaks from package splus2R.

Usage

```
get_peaks(x, y, ignore_threshold = 0, span = 5, strict = TRUE,
  x_unit = "", x_digits = 3, na.rm = FALSE)

get_valleys(x, y, ignore_threshold = 0, span = 5, strict = TRUE,
  x_unit = "", x_digits = 3, na.rm = FALSE)
```

Arguments

x	numeric
y	numeric
ignore_threshold	numeric Value between 0.0 and 1.0 indicating the relative size compared to tallest peak or deepest valley of the peaks to return.
span	numeric A peak is defined as an element in a sequence which is greater than all other elements within a window of width span centered at that element. For example, a value of 3 means that a peak is bigger than both of its neighbors.
strict	logical Flag: if TRUE, an element must be strictly greater than all other values in its window to be considered a peak. Default: TRUE.
x_unit	character Vector of texts to be pasted at end of labels built from x value at peaks.
x_digits	numeric Number of significant digits in wavelength label.
na.rm	logical indicating whether NA values should be stripped before searching for peaks.

Value

A data frame with variables w.length and s.irrad with their values at the peaks or valleys plus a character variable of labels.

See Also

Other peaks and valleys functions: find_peaks, peaks, valleys, wls_at_target

Examples

```
with(sun.spct, get_peaks(w.length, s.e.irrad))
with(sun.spct, get_valleys(w.length, s.e.irrad))
```

`green_leaf.spct` *Green birch leaf reflectance.*

Description

A dataset of spectral reflectance expressed as a fraction of one.

Usage

```
green_leaf.spct
```

Format

A `reflector_spct` object with 226 rows and 2 variables

Details

- w.length (nm)
- Rfr (0..1)

References

Aphalo, P. J. & Lehto, T. Effects of light quality on growth and N accumulation in birch seedlings Tree Physiology, 1997, 17, 125-132

See Also

Other Spectral data examples: `A.illuminant.spct`, `D65.illuminant.spct`, `Ler_leaf.spct`, `Ler_leaf_rflt.spct`, `Ler_leaf_trns.spct`, `Ler_leaf_trns_i.spct`, `black_body.spct`, `ccd.spct`, `clear.spct`, `clear_body.spct`, `filter_cps.mspct`, `opaque.spct`, `photodiode.spct`, `polyester.spct`, `sun.daily.data`, `sun.daily.spct`, `sun.data`, `sun.spct`, `white_body.spct`, `white_led.cps_spct`, `white_led.raw_spct`, `white_led.source_spct`, `yellow_gel.spct`

Examples

```
green_leaf.spct
```

head_tail *Return the First and Last Part of an Object*

Description

Returns the first and last "parts" (rows or members) of a spectrum, dataframe, vector, function, table or ftable. In other words, the combined output from methods head and tail.

Usage

```
head_tail(x, n, ...)

## Default S3 method:
head_tail(x, n = 3L, ...)

## S3 method for class 'data.frame'
head_tail(x, n = 3L, ...)

## S3 method for class 'matrix'
head_tail(x, n = 3L, ...)

## S3 method for class 'function'
head_tail(x, n = 6L, ...)

## S3 method for class 'table'
head_tail(x, n = 6L, ...)

## S3 method for class 'ftable'
head_tail(x, n = 6L, ...)
```

Arguments

x	an R object.
n	integer. If positive, n rows or members in the returned object are copied from each of "head" and "tail" of x. If negative, all except n elements of x from each of "head" and "tail" are returned.
...	arguments to be passed to or from other methods.

Details

The value returned by head_tail() is equivalent to row binding the the values returned by head() and tail(), although not implemented in this way. The same specializations as defined in package 'utils' for head() and tail() have been implemented.

Value

An object (usually) like x but smaller, except when n = 0. For ftable objects x, a transformed format(x).

Methods (by class)

- `default`:
- `data.frame`:
- `matrix`:
- `function`:
- `table`:
- `ftable`:

Note

For some types of input, like functions, the output may be confusing, however, we have opted for consistency with existing functions. The code is in part a revision of that of head() and tail() from package 'utils'. I have been missing this method especially when checking spectral data, as both ends are of interest.

`head_tail()` methods for function, table and ftable classes, are wrappers for head() method.

See Also

head, and compare the examples and the values returned to the examples below.

Examples

```
head_tail(letters)
head_tail(letters, n = -6L)
head_tail(freeny.x, n = 10L)
head_tail(freeny.y)

head_tail(stats::ftable(Titanic))
```

`insert_hinges` *Insert wavelength values into spectral data.*

Description

Inserting wavelengths values immediately before and after a discontinuity in the SWF, greatly reduces the errors caused by interpolating the weighted irradiance during integration of the effective spectral irradiance. This is specially true when data have a large wavelength step size.

Usage

```
insert_hinges(x, y, h)
```

Arguments

x	numeric vector (sorted in increasing order)
y	numeric vector
h	a numeric vector giving the wavelengths at which the y values should be inserted by interpolation, no interpolation is indicated by an empty vector (numeric(0))

Value

a data.frame with variables x and y. Unless the hinge values were already present in y, each inserted hinge, expands the vectors returned in the data frame by one value.

Note

Insertion is a costly operation but I have tried to optimize this function as much as possible by avoiding loops. Earlier this function was implemented in C++, but a bug was discovered and I have now rewritten it using R.

See Also

Other low-level functions operating on numeric vectors.: as_energy, as_quantum_mol, calc_multipliers, div_spectra, energy_irradiance, energy_ratio, integrate_xy, interpolate_spectrum, irradiance, l_insert_hinges, oper_spectra, photon_irradiance, photon_ratio, photons_energy_ratio, prod_spectra, s_e_irrad2rgb, split_energy_irradiance, split_photon_irradiance, subt_spectra, sum_spectra, trim_tails, v_insert_hinges, v_replace_hinges

Examples

```
with(sun.data,
    insert_hinges(w.length, s.e.irrad,
       c(399.99, 400.00, 699.99, 700.00)))
```

---

insert_spct_hinges *Insert new wavelength values into a spectrum*

Description

Insert new wavelength values into a spectrum interpolating the corresponding spectral data values.

Usage

```
insert_spct_hinges(spct, hinges = NULL, byref = FALSE)
```

Arguments

spct	an object of class "generic_spct"
hinges	numeric vector of wavelengths (nm) at which the s.irrad should be inserted by interpolation, no interpolation is indicated by an empty vector (numeric(0))
byref	logical indicating if new object will be created by reference or by copy of spct

Value

a generic_spct or a derived type with variables `w.length` and other numeric variables.

Note

Inserting wavelengths values "hinges" immediately before and after a discontinuity in the SWF, greatly reduces the errors caused by interpolating the weighted irradiance during integration of the effective spectral irradiance. This is specially true when data has a large wavelength step size.

Examples

```
insert_spct_hinges(sun.spct, c(399.99,400.00,699.99,700.00))
insert_spct_hinges(sun.spct,
                   c(199.99,200.00,399.50,399.99,400.00,699.99,
                         700.00,799.99,1000.00))
```

---

integrate_spct *Integrate spectral data.*

Description

This function gives the result of integrating spectral data over wavelengths.

Usage

```
integrate_spct(spct)
```

Arguments

spct	generic_spct

Value

One or more numeric values with no change in scale factor: e.g. [W m-2 nm-1] -> [W m-2]. Each value in the returned vector corresponds to a variable in the spectral object, except for wavelength.

Examples

```
integrate_spct(sun.spct)
```

integrate_xy *Gives irradiance from spectral irradiance.*

Description

This function gives the result of integrating spectral irradiance over wavelengths.

Usage

```
integrate_xy(x, y)
```

Arguments

x	numeric vector.
y	numeric vector.

Value

a single numeric value with no change in scale factor: e.g. [W m-2 nm-1] -> [W m-2]

See Also

Other low-level functions operating on numeric vectors.: as_energy, as_quantum_mol, calc_multipliers, div_spectra, energy_irradiance, energy_ratio, insert_hinges, interpolate_spectrum, irradiance, l_insert_hinges, oper_spectra, photon_irradiance, photon_ratio, photons_energy_ratio, prod_spectra, s_e_irrad2rgb, split_energy_irradiance, split_photon_irradiance, subt_spectra, sum_spectra, trim_tails, v_insert_hinges, v_replace_hinges

Examples

```
with(sun.data, integrate_xy(w.length, s.e.irrad))
```

interpolate_spct *Map a spectrum to new wavelength values.*

Description

This function gives the result of interpolating spectral data from the original set of wavelengths to a new one.

Usage

```
interpolate_spct(spct, w.length.out = NULL, fill = NA,
  length.out = NULL)

interpolate_mspct(mspct, w.length.out = NULL, fill = NA,
  length.out = NULL, .parallel = FALSE, .paropts = NULL)
```

Arguments

spct	generic_spct
w.length.out	numeric vector of wavelengths (nm)
fill	a value to be assigned to out of range wavelengths
length.out	numeric value
mspct	an object of class "generic_mspct"
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Details

If length.out it is a numeric value, then gives the number of rows in the output, if it is NULL, the values in the numeric vector w.length.out are used. If both are not NULL then the range of w.length.out and length.out are used to generate a vector of wavelength. A value of NULL for fill prevents extrapolation. If both w.length.out and length.out are NULL the input is returned as is. If w.length.out has length equal to zero, zero rows from the input are returned.

Value

A new spectral object of the same class as argument spct.

Note

The default fill = NA fills extrapolated values with NA. Giving NULL as argument for fill deletes wavelengths outside the input data range from the returned spectrum. A numerical value can be also be provided as fill. This function calls interpolate_spectrum for each non-wavelength column in the input spectra object.

Examples

```
interpolate_spct(sun.spct, 400:500, NA)
interpolate_spct(sun.spct, 400:500, NULL)
interpolate_spct(sun.spct, seq(200, 1000, by=0.1), 0)
interpolate_spct(sun.spct, c(400,500), length.out=201)
```

`interpolate_spectrum` *Calculate spectral values at a different set of wavelengths*

Description

Interpolate/re-express spectral irradiance (or other spectral quantity) values at new wavelengths values. This is a low-level function operating on numeric vectors and called by higher level functions in the package, such as mathematical operators for classes for spectral data.

Usage

```
interpolate_spectrum(w.length.in, s.irrad, w.length.out, fill = NA, ...)
```

Arguments

`w.length.in`	numeric vector of wavelengths (nm).
`s.irrad`	a numeric vector of spectral values.
`w.length.out`	numeric vector of wavelengths (nm).
`fill`	a value to be assigned to out of range wavelengths.
`...`	additional arguments passed to `spline()`.

Value

a numeric vector of interpolated spectral values.

Note

The current version of interpolate uses `spline` if fewer than 25 data points are available. Otherwise it uses `approx`. In the first case a cubic spline is used, in the second case linear interpolation, which should be faster.

See Also

`splinefun`.

Other low-level functions operating on numeric vectors.: `as_energy`, `as_quantum_mol`, `calc_multipliers`, `div_spectra`, `energy_irradiance`, `energy_ratio`, `insert_hinges`, `integrate_xy`, `irradiance`, `l_insert_hinges`, `oper_spectra`, `photon_irradiance`, `photon_ratio`, `photons_energy_ratio`, `prod_spectra`, `s_e_irrad2rgb`, `split_energy_irradiance`, `split_photon_irradiance`, `subt_spectra`, `sum_spectra`, `trim_tails`, `v_insert_hinges`, `v_replace_hinges`

Examples

```
my.w.length <- 300:700
with(sun.data, interpolate_spectrum(w.length, s.e.irrad, my.w.length))
```

`interpolate_wl` *Map spectra to new wavelength values.*

Description

This function returns the result of interpolating spectral data from the original set of wavelengths to a new one.

Usage

```
interpolate_wl(x, w.length.out, fill, length.out, ...)

## Default S3 method:
interpolate_wl(x, w.length.out, fill, length.out, ...)

## S3 method for class 'generic_spct'
interpolate_wl(x, w.length.out = NULL,
  fill = NA, length.out = NULL, ...)

## S3 method for class 'generic_mspct'
interpolate_wl(x, w.length.out = NULL,
  fill = NA, length.out = NULL, ..., .parallel = FALSE,
  .paropts = NULL)
```

Arguments

`x`	an R object
`w.length.out`	numeric vector of wavelengths (nm)
`fill`	a value to be assigned to out of range wavelengths
`length.out`	numeric value
`...`	not used
`.parallel`	if TRUE, apply function in parallel, using parallel backend provided by foreach
`.paropts`	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Details

If `length.out` it is a numeric value, then gives the number of rows in the output, if it is `NULL`, the values in the numeric vector `w.length.out` are used. If both are not `NULL` then the range of `w.length.out` and `length.out` are used to generate a vector of wavelength. A value of `NULL` for `fill` prevents extrapolation.

Value

A new spectral object of the same class as argument `spct`.

Methods (by class)

- `default`: Default for generic function
- `generic_spct`: Interpolate wavelength in an object of class "generic_spct" or derived.
- `generic_mspct`: Interpolate wavelength in an object of class "generic_mspct" or derived.

Note

The default `fill = NA` fills extrapolated values with NA. Giving NULL as argument for `fill` deletes wavelengths outside the input data range from the returned spectrum. A numerical value can be also be provided as fill. This function calls `interpolate_spectrum` for each non-wavelength column in the input spectra object.

Examples

```
interpolate_wl(sun.spct, 400:500, NA)
interpolate_wl(sun.spct, 400:500, NULL)
interpolate_wl(sun.spct, seq(200, 1000, by=0.1), 0)
interpolate_wl(sun.spct, c(400,500), length.out=201)
```

`irrad` *Irradiance*

Description

This function returns the irradiance for a given waveband of a light source spectrum.

Usage

```
irrad(spct, w.band, unit.out, quantity, time.unit, scale.factor, wb.trim,
  use.cached.mult, use.hinges, allow.scaled, ...)

## Default S3 method:
irrad(spct, w.band, unit.out, quantity, time.unit,
  scale.factor, wb.trim, use.cached.mult, use.hinges, allow.scaled, ...)

## S3 method for class 'source_spct'
irrad(spct, w.band = NULL,
  unit.out = getOption("photobiology.radiation.unit", default =
  "energy"), quantity = "total", time.unit = NULL, scale.factor = 1,
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.cached.mult = getOption("photobiology.use.cached.mult", default =
  FALSE), use.hinges = getOption("photobiology.use.hinges"),
```

```
  allow.scaled = !quantity %in% c("average", "mean", "total"), ...)

## S3 method for class 'source_mspct'
irrad(spct, w.band = NULL,
  unit.out = getOption("photobiology.radiation.unit", default =
  "energy"), quantity = "total", time.unit = NULL, scale.factor = 1,
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.cached.mult = getOption("photobiology.use.cached.mult", default =
  FALSE), use.hinges = NULL, allow.scaled = !quantity %in%
  c("average", "mean", "total"), ..., attr2tb = NULL, idx = "spct.idx",
  .parallel = FALSE, .paropts = NULL)
```

Arguments

spct	an R object.
w.band	waveband or list of waveband objects The waveband(s) determine the region(s) of the spectrum that are summarized.
unit.out	character string with allowed values "energy", and "photon", or its alias "quantum".
quantity	character string One of "total", "average" or "mean", "contribution", "contribution.pc", "relative" or "relative.pc".
time.unit	character or lubridate::duration object.
scale.factor	numeric vector of length 1, or length equal to that of w.band. Numeric multiplier applied to returned values.
wb.trim	logical if TRUE wavebands crossing spectral data boundaries are trimmed, if FALSE, they are discarded.
use.cached.mult	logical indicating whether multiplier values should be cached between calls.
use.hinges	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.
allow.scaled	logical indicating whether scaled or normalized spectra as argument to spct are flagged as an error.
...	other arguments (possibly ignored)
attr2tb	character vector, see add_attr2tb for the syntax for attr2tb passed as is to formal parameter col.names.
idx	character Name of the column with the names of the members of the collection of spectra.
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach.
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Value

A named numeric vector in the case of methods for individual spectra, with one value for each waveband passed to parameter w.band. A data.frame in the case of collections of spectra, containing one column for each waveband object, an index column with the names of the spectra, and optionally additional columns with metadata values retrieved from the attributes of the member spectra.

By default values are only integrated, but depending on the argument passed to parameter quantity they can be re-expressed as relative fractions or percentages. In the case of vector output, names attribute is set to the name of the corresponding waveband unless a named list is supplied in which case the names of the list members are used. The time.unit attribute is copied from the spectrum object to the output. Units are as follows: If time.unit is second, [W m-2 nm-1] -> [mol s-1 m-2] or [W m-2 nm-1] -> [W m-2] If time.unit is day, [J d-1 m-2 nm-1] -> [mol d-1 m-2] or [J d-1 m-2 nm-1] -> [J m-2]

Methods (by class)

- default: Default for generic function
- source_spct: Calculates irradiance from a source_spct object.
- source_mspct: Calculates irradiance from a source_mspct object.

Note

Formal parameter allow.scaled is used internally for calculation of ratios, as rescaling and normalization do not invalidate the calculation of ratios.

The last two parameters control speed optimizations. The defaults should be suitable in most cases. If you will use repeatedly the same SWFs on many spectra measured at exactly the same wavelengths you may obtain some speed up by setting use.cached.mult=TRUE. However, be aware that you are responsible for ensuring that the wavelengths are the same in each call, as the only test done is for the length of the w.length vector.

See Also

Other irradiance functions: e_fluence, e_irrad, fluence, q_fluence, q_irrad

Examples

```
irrad(sun.spct, waveband(c(400,700)))
irrad(sun.spct, waveband(c(400,700)), "energy")
irrad(sun.spct, waveband(c(400,700)), "photon")
irrad(sun.spct, split_bands(c(400,700), length.out = 3))
irrad(sun.spct, split_bands(c(400,700), length.out = 3), quantity = "total")
irrad(sun.spct, split_bands(c(400,700), length.out = 3), quantity = "average")
irrad(sun.spct, split_bands(c(400,700), length.out = 3), quantity = "relative")
irrad(sun.spct, split_bands(c(400,700), length.out = 3), quantity = "relative.pc")
irrad(sun.spct, split_bands(c(400,700), length.out = 3), quantity = "contribution")
irrad(sun.spct, split_bands(c(400,700), length.out = 3), quantity = "contribution.pc")
```

`irradiance` *Photon or energy irradiance from spectral energy or photon irradiance.*

Description

Energy or photon irradiance for one or more wavebands of a radiation spectrum.

Usage

```
irradiance(w.length, s.irrad, w.band = NULL, unit.out = NULL,
  unit.in = "energy", check.spectrum = TRUE, use.cached.mult = FALSE,
  use.hinges = getOption("photobiology.use.hinges", default = NULL))
```

Arguments

`w.length`	numeric Vector of wavelength (nm).
`s.irrad`	numeric vector of spectral (energy) irradiances (W m-2 nm-1).
`w.band`	waveband or list of waveband objects The waveband(s) determine the region(s) of the spectrum that are summarized.
`unit.out`	character Allowed values "energy", and "photon", or its alias "quantum".
`unit.in`	character Allowed values "energy", and "photon", or its alias "quantum".
`check.spectrum`	logical Flag indicating whether to sanity check input data, default is TRUE.
`use.cached.mult`	logical Flag indicating whether multiplier values should be cached between calls.
`use.hinges`	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.

Value

A single numeric value or a vector of numeric values with no change in scale factor: [W m-2 nm-1] -> [mol s-1 m-2]

Note

The last three parameters control speed optimizations. The defaults should be suitable in most cases. If you set `check.spectrum=FALSE` then you should call `check_spectrum()` at least once for your spectrum before using any of the other functions. If you will use repeatedly the same SWFs on many spectra measured at exactly the same wavelengths you may obtain some speed up by setting `use.cached.mult=TRUE`. However, be aware that you are responsible for ensuring that the wavelengths are the same in each call, as the only test done is for the length of the `w.length` vector. The is no reason for setting `use.cpp.code=FALSE` other than for testing the improvement in speed, or in cases where there is no suitable C++ compiler for building the package.

See Also

Other low-level functions operating on numeric vectors.: as_energy, as_quantum_mol, calc_multipliers, div_spectra, energy_irradiance, energy_ratio, insert_hinges, integrate_xy, interpolate_spectrum, l_insert_hinges, oper_spectra, photon_irradiance, photon_ratio, photons_energy_ratio, prod_spectra, s_e_irrad2rgb, split_energy_irradiance, split_photon_irradiance, subt_spectra, sum_spectra, trim_tails, v_insert_hinges, v_replace_hinges

Examples

```
with(sun.data, irradiance(w.length, s.e.irrad, new_waveband(400,700), "photon"))
```

---

is.generic_mspct *Query class of spectrum objects*

Description

Functions to check if an object is of a given type of spectrum, or coerce it if possible.

Usage

```
is.generic_mspct(x)

is.calibration_mspct(x)

is.raw_mspct(x)

is.cps_mspct(x)

is.source_mspct(x)

is.response_mspct(x)

is.filter_mspct(x)

is.reflector_mspct(x)

is.object_mspct(x)

is.chroma_mspct(x)

is.any_mspct(x)
```

Arguments

x	an R object.

Value

These functions return TRUE if its argument is a of the queried type of spectrum and FALSE otherwise.

Note

Derived types also return TRUE for a query for a base type such as generic_mspct.

Examples

```
my.mspct <- filter_mspct(list(polyester.spct, yellow_gel.spct))
is.any_mspct(my.mspct)
is.filter_mspct(my.mspct)
is.source_mspct(my.mspct)
```

---

is.generic_spct *Query class of spectrum objects*

Description

Functions to check if an object is of a given type of spectrum, or coerce it if possible.

Usage

```
is.generic_spct(x)

is.raw_spct(x)

is.calibration_spct(x)

is.cps_spct(x)

is.source_spct(x)

is.response_spct(x)

is.filter_spct(x)

is.reflector_spct(x)

is.object_spct(x)

is.chroma_spct(x)

is.any_spct(x)
```

Arguments

x	an R object.

Value

These functions return TRUE if its argument is a of the queried type of spectrum and FALSE otherwise.

Note

Derived types also return TRUE for a query for a base type such as generic_spct.

Examples

```
is.source_spct(sun.spct)
is.filter_spct(sun.spct)
is.generic_spct(sun.spct)
is.generic_spct(sun.spct)

is.source_spct(sun.spct)
is.filter_spct(sun.spct)
is.generic_spct(sun.spct)
is.generic_spct(sun.spct)
```

---

is.old_spct *Query if an object has old class names*

Description

Query if an object has old class names Query if an object has old class names as used in photobiology (>= 0.6.0).

Usage

```
is.old_spct(object)
```

Arguments

object	an R object

Value

logical

See Also

Other upgrade from earlier versions: upgrade_spct, upgrade_spectra

is.solar_time *Query class*

Description

Query class

Usage

```
is.solar_time(x)

is.solar_date(x)
```

Arguments

x	an R object.

See Also

Other astronomy related functions: day_night, format.solar_time, print.solar_time, solar_time, sun_angles

is.summary_generic_spct *Query class of spectrum summary objects*

Description

Functions to check if an object is of a given type of spectrum, or coerce it if possible.

Usage

```
is.summary_generic_spct(x)

is.summary_raw_spct(x)

is.summary_cps_spct(x)

is.summary_source_spct(x)

is.summary_response_spct(x)

is.summary_filter_spct(x)

is.summary_reflector_spct(x)
```

```
is.summary_object_spct(x)

is.summary_chroma_spct(x)

is.any_summary_spct(x)
```

Arguments

x	an R object.

Value

These functions return TRUE if its argument is a of the queried type of spectrum and FALSE otherwise.

Note

Derived types also return TRUE for a query for a base type such as generic_spct.

Examples

```
sm <- summary(sun.spct)
is.summary_source_spct(sm)
```

---

is.waveband *Query if it is a waveband*

Description

Functions to check if an object is waveband.

Usage

```
is.waveband(x)
```

Arguments

x	any R object

Value

is.waveband returns TRUE if its argument is a waveband and FALSE otherwise.

isValidInstrDesc *Check the "instr.desc" attribute*

Description

Function to validate the "instr.settings" attribute of an existing generic_spct object.

Usage

```
isValidInstrDesc(x)
```

Arguments

x	a generic_spct object

Value

logical TRUE if at least instrument name and serial number is found.

See Also

Other measurement metadata functions: getHowMeasured, getInstrDesc, getInstrSettings, getWhatMeasured, getWhenMeasured, getWhereMeasured, get_attributes, isValidInstrSettings, setHowMeasured, setInstrDesc, setInstrSettings, setWhatMeasured, setWhenMeasured, setWhereMeasured, trimInstrDesc, trimInstrSettings

isValidInstrSettings *Check the "instr.settings" attribute*

Description

Function to validate the "instr.settings" attribute of an existing generic_spct object.

Usage

```
isValidInstrSettings(x)
```

Arguments

x	a generic_spct object

Value

logical TRUE if at least integration time data is found.

See Also

Other measurement metadata functions: getHowMeasured, getInstrDesc, getInstrSettings, getWhatMeasured, getWhenMeasured, getWhereMeasured, get_attributes, isValidInstrDesc, setHowMeasured, setInstrDesc, setInstrSettings, setWhatMeasured, setWhenMeasured, setWhereMeasured, trimInstrDesc, trimInstrSettings

---

is_absorbance_based *Query if a spectrum contains absorbance or transmittance data*

Description

Functions to check if an filter spectrum contains spectral absorbance data or spectral transmittance data.

Usage

```
is_absorbance_based(x)

is_transmittance_based(x)
```

Arguments

x	an R object

Value

is_absorbance_based returns TRUE if its argument is a filter_spct object that contains spectral absorbance data and FALSE if it does not contain such data, but returns NA for any other R object, including those belonging other generic_spct-derived classes.

is_transmittance_based returns TRUE if its argument is a filter_spct object that contains spectral transmittance data and FALSE if it does not contain such data, but returns NA for any other R object, including those belonging other generic_spct-derived classes.

See Also

Other query units functions: is_photon_based

Examples

```
is_absorbance_based(polyester.spct)
my.spct <- T2A(polyester.spct)
is.filter_spct(my.spct)
is_absorbance_based(my.spct)

is_transmittance_based(polyester.spct)
```

`is_effective` *Is an R object "effective"*

Description

A generic function for querying if a biological spectral weighting function (BSWF) has been applied to an object or is included in its definition.

Usage

```
is_effective(x)

## Default S3 method:
is_effective(x)

## S3 method for class 'waveband'
is_effective(x)

## S3 method for class 'generic_spct'
is_effective(x)

## S3 method for class 'source_spct'
is_effective(x)

## S3 method for class 'summary_generic_spct'
is_effective(x)

## S3 method for class 'summary_source_spct'
is_effective(x)
```

Arguments

`x`	an R object

Value

A `logical`.

Methods (by class)

- `default`: Default method.
- `waveband`: Is a `waveband` object defining a method for calculating effective irradiance.
- `generic_spct`: Does a `source_spct` object contain effective spectral irradiance values.
- `source_spct`: Does a `source_spct` object contain effective spectral irradiance values.
- `summary_generic_spct`: Method for "summary_generic_spct".
- `summary_source_spct`: Method for "summary_source_spct".

See Also

Other waveband attributes: labels, normalization

Examples

```
is_effective(summary(sun.spct))
```

`is_normalized` *Query whether a generic spectrum has been normalized.*

Description

This function tests a `generic_spct` object for an attribute that signals whether the spectral data has been normalized or not after the object was created.

Usage

```
is_normalized(x)
```

Arguments

`x`	An R object.

Value

A `logical` value. If `x` is not normalized or `x` is not a `generic_spct` object the value returned is `FALSE`.

See Also

Other rescaling functions: fscale, fshift, getNormalized, is_scaled, normalize, setNormalized, setScaled

---

`is_photon_based` *Query if a spectrum contains photon- or energy-based data.*

Description

Functions to check if `source_spct` and `response_spct` objects contains photon-based or energy-based data.

Usage

```
is_photon_based(x)

is_energy_based(x)
```

Arguments

x	any R object

Value

is_photon_based returns TRUE if its argument is a source_spct or a response_spct object that contains photon base data and FALSE if such an object does not contain such data, but returns NA for any other R object, including those belonging other generic_spct-derived classes.

is_energy_based returns TRUE if its argument is a source_spct or a response_spct object that contains energy base data and FALSE if such an object does not contain such data, but returns NA for any other R object, including those belonging other generic_spct-derived classes

See Also

Other query units functions: is_absorbance_based

Examples

```
is_photon_based(sun.spct)
my.spct <- dplyr::select(sun.spct, w.length, s.e.irrad)
is.source_spct(my.spct)
is_photon_based(my.spct)

is_energy_based(sun.spct)
my.spct <- dplyr::select(sun.spct, w.length, s.q.irrad)
is.source_spct(my.spct)
is_energy_based(my.spct)
```

---

is_scaled *Query whether a generic spectrum has been scaled*

Description

This function tests a generic_spct object for an attribute that signals whether the spectral data has been rescaled or not after the object was created.

Usage

```
is_scaled(x)
```

Arguments

x	An R object.

Value

A logical value. If x is not scaled or x is not a generic_spct object the value returned is FALSE.

See Also

Other rescaling functions: fscale, fshift, getNormalized, is_normalized, normalize, setNormalized, setScaled

---

`is_tagged` *Query if it is an spectrum is tagged*

Description

Functions to check if an spct object contains tags.

Usage

```
is_tagged(x)
```

Arguments

x	any R object

Value

is_tagged returns TRUE if its argument is a an spectrum that contains tags and FALSE if it is an untagged spectrum, but returns NA for any other R object.

See Also

Other tagging and related functions: tag, untag, wb2rect_spct, wb2spct, wb2tagged_spct

Examples

```
is_tagged(sun.spct)
```

`join_mspct` *Join all spectra in a collection*

Description

Join all the spectra contained in a homogenous collection, returning a data frame with spectral-data columns named according to the names of the spectra in the collection. By default a full join is done, filling the spectral data for missing wave lengths in individual spectra with NA.

Usage

```
join_mspct(x, type, ...)

## Default S3 method:
join_mspct(x, type = "full", ...)

## S3 method for class 'generic_mspct'
join_mspct(x, type = "full", col.name, ...)

## S3 method for class 'source_mspct'
join_mspct(x, type = "full",
  unit.out = "energy", ...)

## S3 method for class 'response_mspct'
join_mspct(x, type = "full",
  unit.out = "energy", ...)

## S3 method for class 'filter_mspct'
join_mspct(x, type = "full",
  qty.out = "transmittance", ...)

## S3 method for class 'reflector_mspct'
join_mspct(x, type = "full", ...)

## S3 method for class 'object_mspct'
join_mspct(x, type = "full", qty.out, ...)
```

Arguments

x	A generic_mspct object, or an object of a class derived from generic_mspct.
type	character Type of join: "left", "right", "inner" or "full" (default). See details for more information.
...	ignored (possibly used by derived methods).
col.name	character, name of the column in the spectra to be preserved, in addition to "w.length".
unit.out	character Allowed values "energy", and "photon", or its alias "quantum".
qty.out	character Allowed values "transmittance", and "absorbance".

Value

An object of class dataframe, with the spectra joined by wave length, with rows in addition sorted by wave length (variable `w.length`).

Methods (by class)

- `default`:
- `generic_mspct`:

- source_mspct:
- response_mspct:
- filter_mspct:
- reflector_mspct:
- object_mspct:

Note

Currently only generic_spct, source_mspct, response_mspct, filter_mspct, reflector_mspct and object_mspct classes have this method implemented.

---

labels *Find labels from "waveband" object*

Description

A function to obtain the name and label of objects of class "waveband".

Usage

```
## S3 method for class 'waveband'
labels(object, ...)

## S3 method for class 'generic_spct'
labels(object, ...)
```

Arguments

object	an object of class "waveband"
...	not used in current version

Methods (by class)

- generic_spct:

See Also

Other waveband attributes: is_effective, normalization

Examples

```
labels(sun.spct)
```

Ler_leaf.spct *Green Arabidopsis leaf reflectance and transmittance.*

Description

A dataset of total spectral reflectance and total spectral transmittance expressed as fractions of one from the upper surface of a leaf of an Arabidopsis thaliana 'Ler' rosette.

Usage

```
Ler_leaf.spct
```

Format

An `object_spct` object with 2401 rows and 3 variables

Details

- w.length (nm)
- Rfr (0..1)
- Tfr (0..1)

Note

Measured with a Jaz spectrometer from Ocean Optics (USA) configured with a PX Xenon lamp module and Spectroclip double integrating spheres.

Author(s)

Aphalo, P. J. & Wang, F (unpublished data)

See Also

Other Spectral data examples: `A.illuminant.spct`, `D65.illuminant.spct`, `Ler_leaf_rflt.spct`, `Ler_leaf_trns.spct`, `Ler_leaf_trns_i.spct`, `black_body.spct`, `ccd.spct`, `clear.spct`, `clear_body.spct`, `filter_cps.mspct`, `green_leaf.spct`, `opaque.spct`, `photodiode.spct`, `polyester.spct`, `sun.daily.data`, `sun.daily.spct`, `sun.data`, `sun.spct`, `white_body.spct`, `white_led.cps_spct`, `white_led.raw_spct`, `white_led.source_spct`, `yellow_gel.spct`

Examples

```
Ler_leaf.spct
```

`Ler_leaf_rflt.spct` *Green Arabidopsis leaf spectral reflectance.*

Description

A dataset of total spectral reflectance expressed as fractions of one from the upper surface of a leaf of an Arabidopsis thaliana 'Ler' rosette.

Usage

```
Ler_leaf_rflt.spct
```

Format

An `reflector_spct` object with 1750 rows and 2 variables

Details

- w.length (nm)
- Rfr (0..1)

Note

Measured with a Jaz spectrometer from Ocean Optics (USA) configured with a PX Xenon lamp module and Spectroclip double integrating spheres.

Author(s)

Aphalo, P. J. & Wang, F (unpublished data)

See Also

Other Spectral data examples: `A.illuminant.spct`, `D65.illuminant.spct`, `Ler_leaf.spct`, `Ler_leaf_trns.spct`, `Ler_leaf_trns_i.spct`, `black_body.spct`, `ccd.spct`, `clear.spct`, `clear_body.spct`, `filter_cps.mspct`, `green_leaf.spct`, `opaque.spct`, `photodiode.spct`, `polyester.spct`, `sun.daily.data`, `sun.daily.spct`, `sun.data`, `sun.spct`, `white_body.spct`, `white_led.cps_spct`, `white_led.raw_spct`, `white_led.source_spct`, `yellow_gel.spct`

Examples

```
Ler_leaf_rflt.spct
```

`Ler_leaf_trns.spct` *Green Arabidopsis leaf spectral transmittance.*

Description

A dataset of total spectral transmittance expressed as a fraction of one from the upper surface of a leaf of an Arabidopsis thaliana 'Ler' rosette.

Usage

```
Ler_leaf_trns.spct
```

Format

An `filter_spct` object with 1753 rows and 2 variables

Details

- w.length (nm)
- Tfr (0..1)

Note

Measured with a Jaz spectrometer from Ocean Optics (USA) configured with a PX Xenon lamp module and Spectroclip double integrating spheres.

Author(s)

Aphalo, P. J. & Wang, F (unpublished data)

See Also

Other Spectral data examples: `A.illuminant.spct`, `D65.illuminant.spct`, `Ler_leaf.spct`, `Ler_leaf_rflt.spct`, `Ler_leaf_trns_i.spct`, `black_body.spct`, `ccd.spct`, `clear.spct`, `clear_body.spct`, `filter_cps.mspct`, `green_leaf.spct`, `opaque.spct`, `photodiode.spct`, `polyester.spct`, `sun.daily.data`, `sun.daily.spct`, `sun.data`, `sun.spct`, `white_body.spct`, `white_led.cps_spct`, `white_led.raw_spct`, `white_led.source_spct`, `yellow_gel.spct`

Examples

```
Ler_leaf_trns.spct
```

`Ler_leaf_trns_i.spct` *Green Arabidopsis leaf spectral transmittance.*

Description

A dataset of internal spectral transmittance expressed as a fraction of one from the upper surface of a leaf of an Arabidopsis thaliana 'Ler' rosette.

Usage

```
Ler_leaf_trns_i.spct
```

Format

An `filter_spct` object with 2401 rows and 2 variables

Details

- w.length (nm)
- Tfr (0..1)

Note

Measured with a Jaz spectrometer from Ocean Optics (USA) configured with a PX Xenon lamp module and Spectroclip double integrating spheres.

Author(s)

Aphalo, P. J. & Wang, F (unpublished data)

See Also

Other Spectral data examples: `A.illuminant.spct`, `D65.illuminant.spct`, `Ler_leaf.spct`, `Ler_leaf_rflt.spct`, `Ler_leaf_trns.spct`, `black_body.spct`, `ccd.spct`, `clear.spct`, `clear_body.spct`, `filter_cps.mspct`, `green_leaf.spct`, `opaque.spct`, `photodiode.spct`, `polyester.spct`, `sun.daily.data`, `sun.daily.spct`, `sun.data`, `sun.spct`, `white_body.spct`, `white_led.cps_spct`, `white_led.raw_spct`, `white_led.source_spct`, `yellow_gel.spct`

Examples

```
Ler_leaf_trns_i.spct
```

log *Logarithms and Exponentials*

Description

Logarithms and Exponentials for Spectra. The functions are applied to the spectral data, not the wavelengths. The quantity in the spectrum to which the function is applied depends on the class of x and the current value of output options

Usage

```
## S3 method for class 'generic_spct'
log(x, base = exp(1))

log2.generic_spct(x)

log10.generic_spct(x)

## S3 method for class 'generic_spct'
exp(x)
```

Arguments

x	an object of class "generic_spct"
base	a positive number: the base with respect to which logarithms are computed. Defaults to e=exp(1).

See Also

Other math operators and functions: MathFun, ^.generic_spct, convolve_each, div-.generic_spct, minus-.generic_spct, mod-.generic_spct, plus-.generic_spct, round, sign, slash-.generic_spct, times-.generic_spct

MathFun *Miscellaneous Mathematical Functions*

Description

abs(x) computes the absolute value of x, sqrt(x) computes the (principal) square root of x. The functions are applied to the spectral data, not the wavelengths. The quantity in the spectrum to which the function is applied depends on the class of x and the current value of output options.

Usage

```
## S3 method for class 'generic_spct'
sqrt(x)

## S3 method for class 'generic_spct'
abs(x)
```

Arguments

x	an object of class "generic_spct"

See Also

Other math operators and functions: `^.generic_spct`, `convolve_each`, `div-.generic_spct`, `log`, `minus-.generic_spct`, `mod-.generic_spct`, `plus-.generic_spct`, `round`, `sign`, `slash-.generic_spct`, `times-.generic_spct`

---

merge2object_spct *Merge into object_spct*

Description

Merge a filter_spct with a reflector_spct returning an object_spct object, even if wavelength values are mismatched.

Usage

```
merge2object_spct(x, y, by = "w.length", ...,
  w.length.out = x[["w.length"]], Tfr.type.out = "total")
```

Arguments

x, y	a filter_spct object and a reflector_spct object.
by	a vector of shared column names in x and y to merge on; by defaults to w.length.
...	other arguments passed to dplyr::inner_join()
w.length.out	numeric vector of wavelengths to be used for the returned object (nm).
Tfr.type.out	character string indicating whether transmittance values in the returned object should be expressed as "total" or "internal". This applies only to the case when an object_spct is returned.

Value

An object_spct is returned as the result of merging a filter_spct and a reflector_spct object.

Note

If a numeric vector is supplied as argument for `w.length.out`, the two spectra are interpolated to the new wavelength values before merging. The default argument for `w.length.out` is x[[w.length]].

See Also

`join`

`merge_attributes` *Merge and copy attributes*

Description

Merge attributes from `x` and `y` and copy them to `z`. Methods defined for spectral objects of classes from package 'photobiology'.

Usage

```
merge_attributes(x, y, z, which, ...)

## Default S3 method:
merge_attributes(x, y, z, which = NULL, ...)

## S3 method for class 'generic_spct'
merge_attributes(x, y, z, which = NULL,
  copy.class = FALSE, ...)
```

Arguments

`x, y, z`	R objects
`which`	character
`...`	not used
`copy.class`	logical If TRUE class attributes are also copied.

Value

A copy of `z` with additional attributes set.

Methods (by class)

- `default`: Default for generic function
- `generic_spct`:

minus-.generic_spct *Arithmetic Operators*

Description

Subtraction operator for generic spectra.

Usage

```
## S3 method for class 'generic_spct'
e1 - e2 = NULL
```

Arguments

e1	an object of class "generic_spct"
e2	an object of class "generic_spct"

See Also

Other math operators and functions: MathFun, ^.generic_spct, convolve_each, div-.generic_spct, log, mod-.generic_spct, plus-.generic_spct, round, sign, slash-.generic_spct, times-.generic_spct

mod-.generic_spct *Arithmetic Operators*

Description

Reminder operator for generic spectra.

Usage

```
## S3 method for class 'generic_spct'
e1 %% e2
```

Arguments

e1	an object of class "generic_spct"
e2	an object of class "generic_spct"

See Also

Other math operators and functions: MathFun, ^.generic_spct, convolve_each, div-.generic_spct, log, minus-.generic_spct, plus-.generic_spct, round, sign, slash-.generic_spct, times-.generic_spct

msmsply *Multi-spct transform methods*

Description

Apply a function or operator to a collection of spectra.

Usage

```
msmsply(mspct, .fun, ..., .parallel = FALSE, .paropts = NULL)

msdply(mspct, .fun, ..., idx = NULL, col.names = NULL,
  .parallel = FALSE, .paropts = NULL)

mslply(mspct, .fun, ..., .parallel = FALSE, .paropts = NULL)

msaply(mspct, .fun, ..., .drop = TRUE, .parallel = FALSE,
  .paropts = NULL)
```

Arguments

mspct	an object of class generic_mspct or a derived class
.fun	a function
...	other arguments passed to .fun
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.
idx	character Name of the column with the names of the members of the collection of spectra.
col.names	character Names to be used for data columns.
.drop	should extra dimensions of length 1 in the output be dropped, simplifying the output. Defaults to TRUE

Value

a collection of spectra in the case of msmsply

a data frame in the case of msdply

a list in the case of mslply

an vector in the case of msaply

mspct_classes *Names of multi-spectra classes*

Description

Function that returns a vector containing the names of multi-spectra classes using for collections of spectra.

Usage

```
mspct_classes()
```

Value

A `character` vector of class names.

Examples

```
mspct_classes()
```

na.omit *Handle Missing Values in Objects*

Description

These methods are useful for dealing with NAs in e.g., `source_spct`, `response_spct`, `filter_spct` and `reflector_spct`.

Usage

```
## S3 method for class 'generic_spct'
na.omit(object, na.action = "omit", fill = NULL,
  target.colnames, ...)

## S3 method for class 'source_spct'
na.omit(object, na.action = "omit", fill = NULL,
  ...)

## S3 method for class 'response_spct'
na.omit(object, na.action = "omit",
  fill = NULL, ...)

## S3 method for class 'filter_spct'
na.omit(object, na.action = "omit", fill = NULL,
  ...)
```

```
## S3 method for class 'reflector_spct'
na.omit(object, na.action = "omit",
  fill = NULL, ...)

## S3 method for class 'object_spct'
na.omit(object, na.action = "omit", fill = NULL,
  ...)

## S3 method for class 'cps_spct'
na.omit(object, na.action = "omit", fill = NULL,
  ...)

## S3 method for class 'raw_spct'
na.omit(object, na.action = "omit", fill = NULL,
  ...)

## S3 method for class 'chroma_spct'
na.omit(object, na.action = "omit", fill = NULL,
  ...)

## S3 method for class 'generic_spct'
na.exclude(object, na.action = "exclude",
  fill = NULL, target.colnames, ...)

## S3 method for class 'source_spct'
na.exclude(object, na.action = "exclude",
  fill = NULL, ...)

## S3 method for class 'response_spct'
na.exclude(object, na.action = "exclude",
  fill = NULL, ...)

## S3 method for class 'filter_spct'
na.exclude(object, na.action = "exclude",
  fill = NULL, ...)

## S3 method for class 'reflector_spct'
na.exclude(object, na.action = "exclude",
  fill = NULL, ...)

## S3 method for class 'object_spct'
na.exclude(object, na.action = "exclude",
  fill = NULL, ...)

## S3 method for class 'cps_spct'
na.exclude(object, na.action = "exclude",
  fill = NULL, ...)
```

```
## S3 method for class 'raw_spct'
na.exclude(object, na.action = "exclude",
  fill = NULL, ...)

## S3 method for class 'chroma_spct'
na.exclude(object, na.action = "exclude",
  fill = NULL, ...)
```

Arguments

object	an R object
na.action	character One of "omit", "exclude" or "replace".
fill	numeric Value used to replace NAs unless NULL, in which case interpolation is attempted.
target.colnames	character Vector of names for the target columns to operate upon, if present in object.
...	further arguments other special methods could require

Details

If na.omit removes cases, the row numbers of the cases form the "na.action" attribute of the result, of class "omit".

na.exclude differs from na.omit only in the class of the "na.action" attribute of the result, which is "exclude".

Note

na.fail and na.pass do not require a specialisation for spectral objects. R's definitions work as expected with no need to override them. We do not define a method na.replace, just pass "replace" as argument. The current implementation replaces by interpolation only individual NAs which are flanked on both sides by valid data. Runs of multiple NAs con only replaced by a constant value passed through parameter fill.

See Also

na.fail and na.action

Examples

```
my_sun.spct <- sun.spct
my_sun.spct[3, "s.e.irrad"] <- NA
my_sun.spct[5, "s.q.irrad"] <- NA

head(my_sun.spct)

# rows omitted
zo <- na.omit(my_sun.spct)
```

```
head(zo)
na.action(zo)

# rows excluded
ze <- na.exclude(my_sun.spct)
head(ze)
na.action(ze)

# data in both rows replaced
zr <- na.omit(my_sun.spct, na.action = "replace")
head(zr)
na.action(zr)
```

`normalization` *Normalization of an R object*

Description

Normalization wavelength of an R object, retrieved from the object's attributes.

Usage

```
normalization(x)

## Default S3 method:
normalization(x)

## S3 method for class 'waveband'
normalization(x)
```

Arguments

`x`	an R object

Methods (by class)

- `default`: Default methods.
- `waveband`: Normalization of a `waveband` object.

See Also

Other waveband attributes: `is_effective`, `labels`

normalize *Normalize spectral data*

Description

These functions return a spectral object of the same class as the one supplied as argument but with the spectral data normalized to 1.o a certain wavelength.

Usage

```
normalize(x, ...)

## Default S3 method:
normalize(x, ...)

## S3 method for class 'source_spct'
normalize(x, ..., range = NULL, norm = "max",
  unit.out = getOption("photobiology.radiation.unit", default =
  "energy"), na.rm = FALSE)

## S3 method for class 'response_spct'
normalize(x, ..., range = NULL, norm = "max",
  unit.out = getOption("photobiology.radiation.unit", default =
  "energy"), na.rm = FALSE)

## S3 method for class 'filter_spct'
normalize(x, ..., range = NULL, norm = "max",
  qty.out = getOption("photobiology.filter.qty", default =
  "transmittance"), na.rm = FALSE)

## S3 method for class 'reflector_spct'
normalize(x, ..., range = NULL, norm = "max",
  qty.out = NULL, na.rm = FALSE)

## S3 method for class 'raw_spct'
normalize(x, ..., range = NULL, norm = "max",
  na.rm = FALSE)

## S3 method for class 'cps_spct'
normalize(x, ..., range = NULL, norm = "max",
  na.rm = FALSE)

## S3 method for class 'generic_spct'
normalize(x, ..., range = NULL, norm = "max",
  col.names, na.rm = FALSE)

## S3 method for class 'source_mspct'
```

```
normalize(x, ..., range = NULL, norm = "max",
  unit.out = getOption("photobiology.radiation.unit", default =
  "energy"), na.rm = FALSE, .parallel = FALSE, .paropts = NULL)

## S3 method for class 'response_mspct'
normalize(x, ..., range = NULL, norm = "max",
  unit.out = getOption("photobiology.radiation.unit", default =
  "energy"), na.rm = FALSE, .parallel = FALSE, .paropts = NULL)

## S3 method for class 'filter_mspct'
normalize(x, ..., range = NULL, norm = "max",
  qty.out = getOption("photobiology.filter.qty", default =
  "transmittance"), na.rm = FALSE, .parallel = FALSE,
  .paropts = NULL)

## S3 method for class 'reflector_mspct'
normalize(x, ..., range = x, norm = "max",
  qty.out = NULL, na.rm = FALSE, .parallel = FALSE,
  .paropts = NULL)

## S3 method for class 'raw_mspct'
normalize(x, ..., range = x, norm = "max",
  na.rm = FALSE, .parallel = FALSE, .paropts = NULL)

## S3 method for class 'cps_mspct'
normalize(x, ..., range = x, norm = "max",
  na.rm = FALSE, .parallel = FALSE, .paropts = NULL)
```

Arguments

x	An R object
...	not used in current version
range	An R object on which range() returns a numeric vector of length 2 with the limits of a range of wavelengths in nm, with min and max wavelengths (nm) used to set boundaries for search for normalization.
norm	numeric Normalization wavelength (nm) or character string "max", or "min" for normalization at the corresponding wavelength.
unit.out	character Allowed values "energy", and "photon", or its alias "quantum"
na.rm	logical indicating whether NA values should be stripped before calculating the summary (e.g. "max") used for normalization.
qty.out	character string Allowed values are "transmittance", and "absorbance" indicating on which quantity to apply the normalization.
col.names	character vector containing the names of columns or variables to which to apply the normalization.
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach

`.paropts` a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Value

A copy of `x`, with spectral data values normalized to one for the criterion specified by the argument passed to `norm`.

A copy of `x` with the values of the spectral quantity rescaled to 1 at the normalization wavelength. If the normalization wavelength is not already present in `x`, it is added by interpolation—i.e. the returned value may be one row longer than `x`.

Methods (by class)

- `default`: Default for generic function
- `source_spct`: Normalize a `source_spct` object.
- `response_spct`: Normalize a response spectrum.
- `filter_spct`: Normalize a filter spectrum.
- `reflector_spct`: Normalize a reflector spectrum.
- `raw_spct`: Normalize a raw spectrum.
- `cps_spct`: Normalize a cps spectrum.
- `generic_spct`: Normalize a raw spectrum.
- `source_mspct`: Normalize the members of a source_mspct object.
- `response_mspct`: Normalize the members of a response_mspct object.
- `filter_mspct`: Normalize the members of a filter_mspct object.
- `reflector_mspct`: Normalize the members of a reflector_mspct object.
- `raw_mspct`: Normalize the members of a raw_mspct object.
- `cps_mspct`: Normalize the members of a cps_mspct object.

Note

1) By default if `x` contains one or more `NA` values and the normalization is based on a summary quantity, the returned spectrum will contain only `NA` values. If `na.rm == TRUE` then the summary quantity will be calculated after striping `NA` values, and only the values that were `NA` in `x` will be NA values in the returned spectrum.

See Also

Other rescaling functions: `fscale`, `fshift`, `getNormalized`, `is_normalized`, `is_scaled`, `setNormalized`, `setScaled`

Examples

```
normalize(sun.spct)
normalize(sun.spct, norm = "max")
normalize(sun.spct, norm = 400)
```

`normalized_diff_ind` *Calculate a normalized index.*

Description

This function returns a normalized difference index value for an arbitrary pair of wavebands.

Usage

```
normalized_diff_ind(spct, plus.w.band, minus.w.band, f, ...)
```

Arguments

`spct`	an R object
`plus.w.band`	waveband objects The waveband determine the region of the spectrum used in the calculations
`minus.w.band`	waveband objects The waveband determine the region of the spectrum used in the calculations
`f`	function used for integration taking spct as first argument and a list of wavebands as second argument.
`...`	additional arguments passed to f

Value

A numeric value for the index

Note

`f` is most frequently `reflectance`, but also `transmittance`, or even `absorbance`, `response`, `irradiance` or a user-defined function can be used if there is a good reason for it. In every case `spct` should be of the class expected by `f`. When using two wavebands of different widths do consider passing to `f` a suitable `quantity` argument. Wavebands can describe weighting functions if desired.

normalize_range_arg *Normalize a range argument into a true numeric range*

Description

Several functions in this package and the suite accept a range argument with a flexible syntax. To ensure that all functions and methods behave in the same way this code has been factored out into a separate function.

Usage

```
normalize_range_arg(arg.range, wl.range, trim = TRUE)
```

Arguments

arg.range	a numeric vector of length two, or any other object for which function range() will return a range of wavelengths (nm).
wl.range	a numeric vector of length two, or any other object for which function range() will return a range of wavelengths (nm), missing values are not allowed.
trim	logical If TRUE the range returned is bound within wl.range while if FALSE it can be broader.

Details

The arg.range argument can contain NAs which are replaced by the value at the same position in wl.range. In addition a NULL argument for range is converted into wl.range. The wl.range is also the limit to which the returned value is trimmed if trim == TRUE. The idea is that the value supplied as wl.range is the wavelength range of the data.

Value

a numeric vector of length two, guaranteed not to have missing values.

Examples

```
normalize_range_arg(c(NA, 500), range(sun.spct))
normalize_range_arg(c(300, NA), range(sun.spct))
normalize_range_arg(c(100, 5000), range(sun.spct), FALSE)
normalize_range_arg(c(NA, NA), range(sun.spct))
normalize_range_arg(c(NA, NA), sun.spct)
```

opaque.spct *Theoretical spectrum of an opaque material*

Description

A dataset for a hypothetical object with transmittance 0/1 (0%)

Usage

```
opaque.spct
```

Format

A `filter_spct` object with 4 rows and 2 variables

Details

- w.length (nm).
- Tfr (0..1)

See Also

Other Spectral data examples: `A.illuminant.spct`, `D65.illuminant.spct`, `Ler_leaf.spct`, `Ler_leaf_rflt.spct`, `Ler_leaf_trns.spct`, `Ler_leaf_trns_i.spct`, `black_body.spct`, `ccd.spct`, `clear.spct`, `clear_body.spct`, `filter_cps.mspct`, `green_leaf.spct`, `photodiode.spct`, `polyester.spct`, `sun.daily.data`, `sun.daily.spct`, `sun.data`, `sun.spct`, `white_body.spct`, `white_led.cps_spct`, `white_led.raw_spct`, `white_led.source_spct`, `yellow_gel.spct`

Examples

```
opaque.spct
```

oper_spectra *Binary operation on two spectra, even if the wavelengths values differ*

Description

The wavelength vectors of the two spectra are merged, and the missing spectral values are calculated by interpolation. After this, the two spectral values at each wavelength are added.

Usage

```
oper_spectra(w.length1, w.length2 = NULL, s.irrad1, s.irrad2,
  trim = "union", na.rm = FALSE, bin.oper = NULL, ...)
```

Arguments

w.length1	numeric vector of wavelength (nm)
w.length2	numeric vector of wavelength (nm)
s.irrad1	a numeric vector of spectral values
s.irrad2	a numeric vector of spectral values
trim	a character string with value "union" or "intersection"
na.rm	a logical value, if TRUE, not the default, NAs in the input are replaced with zeros
bin.oper	a function defining a binary operator (for the usual math operators enclose argument in backticks)
...	additional arguments (by name) passed to bin.oper

Details

If trim=="union" spectral values are calculated for the whole range of wavelengths covered by at least one of the input spectra, and missing values are set in each input spectrum to zero before addition. If trim=="intersection" then the range of wavelengths covered by both input spectra is returned, and the non-overlapping regions discarded. If w.length2==NULL, it is assumed that both spectra are measured at the same wavelengths, and a simple addition is used, ensuring fast calculation.

Value

a dataframe with two numeric variables

w.length	A numeric vector with the wavelengths (nm) obtained by "fusing" w.length1 and w.length2. w.length contains all the unique vales, sorted in ascending order.
s.irrad	A numeric vector with the sum of the two spectral values at each wavelength.

See Also

Other low-level functions operating on numeric vectors.: as_energy, as_quantum_mol, calc_multipliers, div_spectra, energy_irradiance, energy_ratio, insert_hinges, integrate_xy, interpolate_spectrum, irradiance, l_insert_hinges, photon_irradiance, photon_ratio, photons_energy_ratio, prod_spectra, s_e_irrad2rgb, split_energy_irradiance, split_photon_irradiance, subt_spectra, sum_spectra, trim_tails, v_insert_hinges, v_replace_hinges

Examples

```
head(sun.data)
result.data <-
    with(sun.data,
         oper_spectra(w.length, w.length, s.e.irrad, s.e.irrad, bin.oper=`+`))
head(result.data)
tail(result.data)
my_fun <- function(e1, e2, k) {return((e1 + e2) / k)}
```

```
result.data <-
    with(sun.data,
        oper_spectra(w.length, w.length, s.e.irrad, s.e.irrad, bin.oper=my_fun, k=2))
head(result.data)
tail(result.data)
```

---

peaks *Peaks or local maxima*

Description

Function that returns a subset of an R object with observations corresponding to local maxima.

Usage

```
peaks(x, span, ignore_threshold, strict, na.rm, ...)

## Default S3 method:
peaks(x, span = NA, ignore_threshold = NA,
  strict = NA, na.rm = FALSE, ...)

## S3 method for class 'numeric'
peaks(x, span = 5, ignore_threshold = NA,
  strict = TRUE, na.rm = FALSE, ...)

## S3 method for class 'data.frame'
peaks(x, span = 5, ignore_threshold = 0,
  strict = TRUE, na.rm = FALSE, var.name, ...)

## S3 method for class 'generic_spct'
peaks(x, span = 5, ignore_threshold = 0,
  strict = TRUE, na.rm = FALSE, var.name = NULL, ...)

## S3 method for class 'source_spct'
peaks(x, span = 5, ignore_threshold = 0,
  strict = TRUE, na.rm = FALSE,
  unit.out = getOption("photobiology.radiation.unit", default =
  "energy"), ...)

## S3 method for class 'response_spct'
peaks(x, span = 5, ignore_threshold = 0,
  strict = TRUE, na.rm = FALSE,
  unit.out = getOption("photobiology.radiation.unit", default =
  "energy"), ...)

## S3 method for class 'filter_spct'
```

```
peaks(x, span = 5, ignore_threshold = 0,
  strict = TRUE, na.rm = FALSE,
  filter.qty = getOption("photobiology.filter.qty", default =
  "transmittance"), ...)

## S3 method for class 'reflector_spct'
peaks(x, span = 5, ignore_threshold = 0,
  strict = TRUE, na.rm = FALSE, ...)

## S3 method for class 'cps_spct'
peaks(x, span = 5, ignore_threshold = 0,
  strict = TRUE, na.rm = FALSE, ...)

## S3 method for class 'generic_mspct'
peaks(x, span = 5, ignore_threshold = 0,
  strict = TRUE, na.rm = FALSE, ..., .parallel = FALSE,
  .paropts = NULL)
```

Arguments

x	an R object
span	a peak is defined as an element in a sequence which is greater than all other elements within a window of width span centered at that element. The default value is 3, meaning that a peak is bigger than both of its neighbors. Default: 3.
ignore_threshold	numeric value between 0.0 and 1.0 indicating the relative size compared to tallest peak threshold below which peaks will be ignored.
strict	logical flag: if TRUE, an element must be strictly greater than all other values in its window to be considered a peak. Default: TRUE.
na.rm	logical indicating whether NA values should be stripped before searching for peaks.
...	ignored
var.name	Name of column where to look for peaks.
unit.out	character One of "energy" or "photon"
filter.qty	character One of "transmittance" or "absorbance"
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Value

A subset of x with rows corresponding to local maxima.

Methods (by class)

- `default`: Default returning always NA.
- `numeric`: Default function usable on numeric vectors.
- `data.frame`: Method for "data.frame" objects.
- `generic_spct`: Method for "generic_spct" objects.
- `source_spct`: Method for "source_spct" objects.
- `response_spct`: Method for "response_spct" objects.
- `filter_spct`: Method for "filter_spct" objects.
- `reflector_spct`: Method for "reflector_spct" objects.
- `cps_spct`: Method for "cps_spct" objects.
- `generic_mspct`: Method for "cps_spct" objects.

See Also

Other peaks and valleys functions: `find_peaks`, `get_peaks`, `valleys`, `wls_at_target`

Examples

```
peaks(sun.spct, span = 50)
peaks(sun.spct, span = NULL)

peaks(sun.spct)
```

---

photodiode.spct *Spectral response of a GaAsP photodiode*

Description

A dataset containing wavelengths at a 1 nm interval and spectral response as $A/(W/nm)$ for GaAsP photodiode type G6262 from Hamamatsu. Data digitized from manufacturer's data sheet. The value at the peak is 0.19 A/W.

Usage

```
photodiode.spct
```

Format

A `response_spct` object with 94 rows and 2 variables

Details

- w.length (nm).
- s.e.response (A/W)

References

Hamamatsu (2011) Datasheet: GaAsP Photodiodes G5645 G5842 G6262. Hamamatsu Photonics KK, Hamamatsu, City. http://www.hamamatsu.com/jp/en/G6262.html. Visited 2017-12-15.

See Also

Other Spectral data examples: A.illuminant.spct, D65.illuminant.spct, Ler_leaf.spct, Ler_leaf_rflt.spct, Ler_leaf_trns.spct, Ler_leaf_trns_i.spct, black_body.spct, ccd.spct, clear.spct, clear_body.spct, filter_cps.mspct, green_leaf.spct, opaque.spct, polyester.spct, sun.daily.data, sun.daily.spct, sun.data, sun.spct, white_body.spct, white_led.cps_spct, white_led.raw_spct, white_led.source_spct, yellow_gel.spct

Examples

```
photodiode.spct
```

---

photons_energy_ratio *Photon:energy ratio*

Description

This function gives the photons:energy ratio between for one given waveband of a radiation spectrum.

Usage

```
photons_energy_ratio(w.length, s.irrad, w.band = NULL,
  unit.in = "energy", check.spectrum = TRUE, use.cached.mult = FALSE,
  use.hinges = getOption("photobiology.use.hinges", default = NULL))
```

Arguments

w.length	numeric vector of wavelength (nm).
s.irrad	numeric vector of spectral (energy) irradiances (W m-2 nm-1).
w.band	waveband object.
unit.in	character Allowed values "energy", and "photon", or its alias "quantum".
check.spectrum	logical Flag telling whether to sanity check input data, default is TRUE.
use.cached.mult	logical Flag telling whether multiplier values should be cached between calls.
use.hinges	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.

Value

A single numeric value giving the ratio moles-photons per Joule.

Note

The default for the `w.band` parameter is a waveband covering the whole range of `w.length`.

See Also

Other low-level functions operating on numeric vectors.: `as_energy`, `as_quantum_mol`, `calc_multipliers`, `div_spectra`, `energy_irradiance`, `energy_ratio`, `insert_hinges`, `integrate_xy`, `interpolate_spectrum`, `irradiance`, `l_insert_hinges`, `oper_spectra`, `photon_irradiance`, `photon_ratio`, `prod_spectra`, `s_e_irrad2rgb`, `split_energy_irradiance`, `split_photon_irradiance`, `subt_spectra`, `sum_spectra`, `trim_tails`, `v_insert_hinges`, `v_replace_hinges`

Examples

```
# photons:energy ratio
with(sun.data, photons_energy_ratio(w.length, s.e.irrad, new_waveband(400,500)))
# photons:energy ratio for whole spectrum
with(sun.data, photons_energy_ratio(w.length, s.e.irrad))
```

`photon_irradiance` *Photon irradiance*

Description

This function returns the photon irradiance for a given waveband of a radiation spectrum, optionally applies a BSWF.

Usage

```
photon_irradiance(w.length, s.irrad, w.band = NULL, unit.in = "energy",
  check.spectrum = TRUE, use.cached.mult = FALSE,
  use.hinges = getOption("photobiology.use.hinges", default = NULL))
```

Arguments

`w.length`	numeric vector of wavelength (nm).
`s.irrad`	numeric vector of spectral irradiances, by default as energy (W m-2 nm-1).
`w.band`	waveband.
`unit.in`	character Values recognized "photon" or "energy".
`check.spectrum`	logical Flag telling whether to sanity check input data, default is TRUE.
`use.cached.mult`	logical Flag telling whether multiplier values should be cached between calls.

use.hinges	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.

Value

A single numeric value with no change in scale factor: [mol s-1 m-2 nm-1] -> [mol s-1 m-2].

See Also

Other low-level functions operating on numeric vectors.: as_energy, as_quantum_mol, calc_multipliers, div_spectra, energy_irradiance, energy_ratio, insert_hinges, integrate_xy, interpolate_spectrum, irradiance, l_insert_hinges, oper_spectra, photon_ratio, photons_energy_ratio, prod_spectra, s_e_irrad2rgb, split_energy_irradiance, split_photon_irradiance, subt_spectra, sum_spectra, trim_tails, v_insert_hinges, v_replace_hinges

Examples

```
with(sun.data, photon_irradiance(w.length, s.e.irrad))
with(sun.data, photon_irradiance(w.length, s.e.irrad, new_waveband(400,700)))
```

---

photon_ratio *Photo:photon ratio*

Description

This function gives the photon ratio between two given wavebands of a radiation spectrum.

Usage

```
photon_ratio(w.length, s.irrad, w.band.num = NULL, w.band.denom = NULL,
  unit.in = "energy", check.spectrum = TRUE, use.cached.mult = FALSE,
  use.hinges = getOption("photobiology.use.hinges", default = NULL))
```

Arguments

w.length	numeric vector of wavelength (nm).
s.irrad	numeric vector of spectral (energy or photon) irradiances (W m-2 nm-1) or (mol s-1 m-2 nm-1).
w.band.num	waveband object used to compute the numerator of the ratio.
w.band.denom	waveband object used to compute the denominator of the ratio.
unit.in	character Allowed values "energy", and "photon", or its alias "quantum".
check.spectrum	logical Flag telling whether to sanity check input data, default is TRUE.
use.cached.mult	logical Flag telling whether multiplier values should be cached between calls.
use.hinges	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.

Value

a single numeric value giving the unitless ratio.

Note

The default for both `w.band` parameters is a waveband covering the whole range of `w.length`.

See Also

Other low-level functions operating on numeric vectors.: `as_energy`, `as_quantum_mol`, `calc_multipliers`, `div_spectra`, `energy_irradiance`, `energy_ratio`, `insert_hinges`, `integrate_xy`, `interpolate_spectrum`, `irradiance`, `l_insert_hinges`, `oper_spectra`, `photon_irradiance`, `photons_energy_ratio`, `prod_spectra`, `s_e_irrad2rgb`, `split_energy_irradiance`, `split_photon_irradiance`, `subt_spectra`, `sum_spectra`, `trim_tails`, `v_insert_hinges`, `v_replace_hinges`

Examples

```
with(sun.data,
     photon_ratio(w.length, s.e.irrad, new_waveband(400,500), new_waveband(400,700)))
```

---

plus-.generic_spct *Arithmetic Operators*

Description

Division operator for generic spectra.

Usage

```
## S3 method for class 'generic_spct'
e1 + e2 = NULL
```

Arguments

`e1`	an object of class "generic_spct"
`e2`	an object of class "generic_spct"

See Also

Other math operators and functions: `MathFun`, `^.generic_spct`, `convolve_each`, `div-.generic_spct`, `log`, `minus-.generic_spct`, `mod-.generic_spct`, `round`, `sign`, `slash-.generic_spct`, `times-.generic_spct`

polyester.spct *Transmittance spectrum of clear polyester film*

Description

A dataset containing the wavelengths at a 1 nm interval and fractional total transmittance for polyester film.

Usage

```
polyester.spct
```

Format

A `filter_spct` object with 611 rows and 2 variables

Details

- w.length (nm).
- Tfr (0..1)

See Also

Other Spectral data examples: `A.illuminant.spct`, `D65.illuminant.spct`, `Ler_leaf.spct`, `Ler_leaf_rflt.spct`, `Ler_leaf_trns.spct`, `Ler_leaf_trns_i.spct`, `black_body.spct`, `ccd.spct`, `clear.spct`, `clear_body.spct`, `filter_cps.mspct`, `green_leaf.spct`, `opaque.spct`, `photodiode.spct`, `sun.daily.data`, `sun.daily.spct`, `sun.data`, `sun.spct`, `white_body.spct`, `white_led.cps_spct`, `white_led.raw_spct`, `white_led.source_spct`, `yellow_gel.spct`

Examples

```
polyester.spct
```

print *Print a spectral object*

Description

Print method for objects of spectral classes.

Usage

```
## S3 method for class 'generic_spct'
print(x, ..., n = NULL, width = NULL)

## S3 method for class 'generic_mspct'
print(x, ..., n = NULL, width = NULL,
  n.members = 10)
```

Arguments

x	An object of one of the summary classes for spectra
...	not used in current version
n	Number of rows to show. If NULL, the default, will print all rows if less than option dplyr.print_max. Otherwise, will print dplyr.print_min
width	Width of text output to generate. This defaults to NULL, which means use getOption("width") and only display the columns that fit on one screen. You can also set option(dplyr.width = Inf) to override this default and always print all columns.
n.members	numeric Number of members of the collection to print.

Value

Returns x invisibly.

Methods (by class)

- generic_mspct:

Note

At the moment just a modified copy of dplyr:::print.tbl_df.

Examples

```
print(sun.spct)
print(sun.spct, n = 5)
```

print.solar_time *Print solar time and solar date objects*

Description

Print solar time and solar date objects

Usage

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

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

Arguments

x	an R object
...	passed to format method

Note

Default is to print the underlying POSIXct as a solar time.

See Also

Other astronomy related functions: day_night, format.solar_time, is.solar_time, solar_time, sun_angles

print.summary_generic_spct *Print spectral summary*

Description

A function to nicely print objects of classes "summary...spct".

Usage

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

Arguments

x	An object of one of the summary classes for spectra
...	not used in current version

Examples

```
print(summary(sun.spct))
```

`print.waveband` *Print a "waveband" object*

Description

A function to more nicely print objects of class "waveband".

Usage

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

Arguments

`x`	an object of class "waveband"
`...`	not used in current version

`prod_spectra` *Multiply two spectra, even if the wavelengths values differ*

Description

The wavelength vectors of the two spectra are merged, and the missing spectral values are calculated by interpolation. After this, the two spectral values at each wavelength are added.

Usage

```
prod_spectra(w.length1, w.length2 = NULL, s.irrad1, s.irrad2,
  trim = "union", na.rm = FALSE)
```

Arguments

`w.length1`	numeric vector of wavelength (nm).
`w.length2`	numeric vector of wavelength (nm).
`s.irrad1`	a numeric vector of spectral values.
`s.irrad2`	a numeric vector of spectral values.
`trim`	a character string with value "union" or "intersection".
`na.rm`	a logical value, if TRUE, not the default, NAs in the input are replaced with zeros.

Details

If trim=="union" spectral values are calculated for the whole range of wavelengths covered by at least one of the input spectra, and missing values are set in each input spectrum to zero before addition. If trim=="intersection" then the range of wavelengths covered by both input spectra is returned, and the non-overlapping regions discarded. If w.length2==NULL, it is assumed that both spectra are measured at the same wavelengths, and a simple addition is used, ensuring fast calculation.

Value

a dataframe with two numeric variables

`w.length`	A numeric vector with the wavelengths (nm) obtained by "fusing" w.length1 and w.length2. w.length contains all the unique vales, sorted in ascending order.
`s.irrad`	A numeric vector with the sum of the two spectral values at each wavelength.

See Also

Other low-level functions operating on numeric vectors.: as_energy, as_quantum_mol, calc_multipliers, div_spectra, energy_irradiance, energy_ratio, insert_hinges, integrate_xy, interpolate_spectrum, irradiance, l_insert_hinges, oper_spectra, photon_irradiance, photon_ratio, photons_energy_ratio, s_e_irrad2rgb, split_energy_irradiance, split_photon_irradiance, subt_spectra, sum_spectra, trim_tails, v_insert_hinges, v_replace_hinges

Examples

```
head(sun.data)
square.sun.data <-
  with(sun.data, prod_spectra(w.length, w.length, s.e.irrad, s.e.irrad))
head(square.sun.data)
tail(square.sun.data)
```

---

q2e *Convert photon-based quantities into energy-based quantities*

Description

Function that converts spectral photon irradiance (molar) into spectral energy irradiance.

Usage

```
q2e(x, action, byref, ...)

## Default S3 method:
q2e(x, action = "add", byref = FALSE, ...)
```

```
## S3 method for class 'source_spct'
q2e(x, action = "add", byref = FALSE, ...)

## S3 method for class 'response_spct'
q2e(x, action = "add", byref = FALSE, ...)

## S3 method for class 'source_mspct'
q2e(x, action = "add", byref = FALSE, ...,
  .parallel = FALSE, .paropts = NULL)

## S3 method for class 'response_mspct'
q2e(x, action = "add", byref = FALSE, ...,
  .parallel = FALSE, .paropts = NULL)
```

Arguments

x	an R object
action	a character string
byref	logical indicating if new object will be created by reference or by copy of x
...	not used in current version
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Methods (by class)

- `default`: Default method
- `source_spct`: Method for spectral irradiance
- `response_spct`: Method for spectral responsiveness
- `source_mspct`: Method for collections of (light) source spectra
- `response_mspct`: Method for collections of response spectra

See Also

Other quantity conversion functions: `A2T`, `T2Afr`, `T2A`, `T2T`, `as_quantum`, `e2qmol_multipliers`, `e2quantum_multipliers`, `e2q`

qe_ratio *Photon:energy ratio*

Description

This function returns the photon to energy ratio for each waveband of a light source spectrum.

Usage

```
qe_ratio(spct, w.band, wb.trim, use.cached.mult, use.hinges, ...)

## Default S3 method:
qe_ratio(spct, w.band, wb.trim, use.cached.mult,
  use.hinges, ...)

## S3 method for class 'source_spct'
qe_ratio(spct, w.band = NULL,
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.cached.mult = FALSE,
  use.hinges = getOption("photobiology.use.hinges"), ...)

## S3 method for class 'source_mspct'
qe_ratio(spct, w.band = NULL,
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.cached.mult = FALSE,
  use.hinges = getOption("photobiology.use.hinges"), ...,
  attr2tb = NULL, idx = "spct.idx", .parallel = FALSE,
  .paropts = NULL)
```

Arguments

spct	source_spct.
w.band	waveband or list of waveband objects.
wb.trim	logical if TRUE wavebands crossing spectral data boundaries are trimmed, if FALSE, they are discarded.
use.cached.mult	logical Flag telling whether multiplier values should be cached between calls.
use.hinges	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.
...	other arguments (possibly used by derived methods).
attr2tb	character vector, see add_attr2tb for the syntax for attr2tb passed as is to formal parameter col.names.
idx	character Name of the column with the names of the members of the collection of spectra.

.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Value

Computed values are ratios between photon irradiance and energy irradiance for a given waveband. A named `numeric` vector in the case of methods for individual spectra, with one value for each `waveband` passed to parameter `w.band`. A `data.frame` in the case of collections of spectra, containing one column for each `waveband` object, an index column with the names of the spectra, and optionally additional columns with metadata values retrieved from the attributes of the member spectra.

By default values are only integrated, but depending on the argument passed to parameter `quantity` they can be re-expressed as relative fractions or percentages. In the case of vector output, `names` attribute is set to the name of the corresponding waveband unless a named list is supplied in which case the names of the list members are used, with "q:e" prepended. Units [mol J-1].

Methods (by class)

- `default`: Default for generic function
- `source_spct`: Method for `source_spct` objects
- `source_mspct`: Calculates photon:energy ratio from a `source_mspct` object.

Note

The last two parameters control speed optimizations. The defaults should be suitable in most cases. If you will use repeatedly the same SWFs on many spectra measured at exactly the same wavelengths you may obtain some speed up by setting `use.cached.mult=TRUE`. However, be aware that you are responsible for ensuring that the wavelengths are the same in each call, as the only test done is for the length of the `w.length` vector.

See Also

Other photon and energy ratio functions: `e_ratio`, `eq_ratio`, `q_ratio`

Examples

```
qe_ratio(sun.spct, new_waveband(400,700))
```

q_fluence *Photon fluence*

Description

Photon irradiance (i.e. quantum irradiance) for one or more waveband of a light source spectrum.

Usage

```
q_fluence(spct, w.band, exposure.time, scale.factor, wb.trim,
  use.cached.mult, use.hinges, allow.scaled, ...)

## Default S3 method:
q_fluence(spct, w.band, exposure.time, scale.factor,
  wb.trim, use.cached.mult, use.hinges, allow.scaled, ...)

## S3 method for class 'source_spct'
q_fluence(spct, w.band = NULL, exposure.time,
  scale.factor = 1, wb.trim = getOption("photobiology.waveband.trim",
  default = TRUE),
  use.cached.mult = getOption("photobiology.use.cached.mult", default =
  FALSE), use.hinges = NULL, allow.scaled = FALSE, ...)

## S3 method for class 'source_mspct'
q_fluence(spct, w.band = NULL, exposure.time,
  scale.factor = 1, wb.trim = getOption("photobiology.waveband.trim",
  default = TRUE),
  use.cached.mult = getOption("photobiology.use.cached.mult", default =
  FALSE), use.hinges = NULL, allow.scaled = FALSE, ...,
  attr2tb = NULL, idx = "spct.idx", .parallel = FALSE,
  .paropts = NULL)
```

Arguments

spct	an R object.
w.band	a list of waveband objects or a waveband object
exposure.time	lubridate::duration object.
scale.factor	numeric vector of length 1, or length equal to that of w.band. Numeric multiplier applied to returned values.
wb.trim	logical if TRUE wavebands crossing spectral data boundaries are trimmed, if FALSE, they are discarded.
use.cached.mult	logical indicating whether multiplier values should be cached between calls.
use.hinges	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.

allow.scaled	logical indicating whether scaled or normalized spectra as argument to spct are flagged as an error.
...	other arguments (possibly ignored).
attr2tb	character vector, see add_attr2tb for the syntax for attr2tb passed as is to formal parameter col.names.
idx	character Name of the column with the names of the members of the collection of spectra.
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Value

One numeric value for each waveband with no change in scale factor, with name attribute set to the name of each waveband unless a named list is supplied in which case the names of the list elements are used. The exposure.time is copied from the spectrum object to the output as an attribute. Units are as follows: moles of photons per exposure.

Methods (by class)

- default: Default for generic function
- source_spct: Calculate photon fluence from a source_spct object and the duration of the exposure
- source_mspct: Calculates photon (quantum) fluence from a source_mspct object.

Note

The last two parameters control speed optimizations. The defaults should be suitable in most cases. If you will use repeatedly the same SWFs on many spectra measured at exactly the same wavelengths you may obtain some speed up by setting use.cached.mult=TRUE. However, be aware that you are responsible for ensuring that the wavelengths are the same in each call, as the only test done is for the length of the w.length vector.

See Also

Other irradiance functions: e_fluence, e_irrad, fluence, irrad, q_irrad

Examples

```
library(lubridate)
q_fluence(sun.spct,
          w.band = waveband(c(400,700)),
          exposure.time = lubridate::duration(3, "minutes") )
```

q_irrad *Photon irradiance*

Description

Photon irradiance (i.e. quantum irradiance) for one or more wavebands of a light source spectrum.

Usage

```
q_irrad(spct, w.band, quantity, time.unit, scale.factor, wb.trim,
  use.cached.mult, use.hinges, allow.scaled, ...)

## Default S3 method:
q_irrad(spct, w.band, quantity, time.unit, scale.factor,
  wb.trim, use.cached.mult, use.hinges, allow.scaled, ...)

## S3 method for class 'source_spct'
q_irrad(spct, w.band = NULL, quantity = "total",
  time.unit = NULL, scale.factor = 1,
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.cached.mult = getOption("photobiology.use.cached.mult", default =
  FALSE), use.hinges = NULL, allow.scaled = !quantity %in%
  c("average", "mean", "total"), ...)

## S3 method for class 'source_mspct'
q_irrad(spct, w.band = NULL, quantity = "total",
  time.unit = NULL, scale.factor = 1,
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.cached.mult = getOption("photobiology.use.cached.mult", default =
  FALSE), use.hinges = NULL, allow.scaled = !quantity %in%
  c("average", "mean", "total"), ..., attr2tb = NULL, idx = "spct.idx",
  .parallel = FALSE, .paropts = NULL)
```

Arguments

spct	an R object.
w.band	a list of waveband objects or a waveband object.
quantity	character string One of "total", "average" or "mean", "contribution", "contribution.pc", "relative" or "relative.pc".
time.unit	character or lubridate::duration object.
scale.factor	numeric vector of length 1, or length equal to that of w.band. Numeric multiplier applied to returned values.
wb.trim	logical if TRUE wavebands crossing spectral data boundaries are trimmed, if FALSE, they are discarded.
use.cached.mult	logical indicating whether multiplier values should be cached between calls.

use.hinges	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.
allow.scaled	logical indicating whether scaled or normalized spectra as argument to spct are flagged as an error.
...	other arguments (possibly ignored).
attr2tb	character vector, see add_attr2tb for the syntax for attr2tb passed as is to formal parameter col.names.
idx	character Name of the column with the names of the members of the collection of spectra.
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Value

A named numeric vector in the case of methods for individual spectra, with one value for each waveband passed to parameter w.band. A data.frame in the case of collections of spectra, containing one column for each waveband object, an index column with the names of the spectra, and optionally additional columns with metadata values retrieved from the attributes of the member spectra.

By default values are only integrated, but depending on the argument passed to parameter quantity they can be re-expressed as relative fractions or percentages. In the case of vector output, names attribute is set to the name of the corresponding waveband unless a named list is supplied in which case the names of the list members are used. The time.unit attribute is copied from the spectrum object to the output. Units are as follows: If time.unit is second, [W m-2 nm-1] -> [mol s-1 m-2] If time.unit is day, [J d-1 m-2 nm-1] -> [mol d-1 m-2]

Methods (by class)

- default: Default for generic function
- source_spct: Calculates photon irradiance from a source_spct object.
- source_mspct: Calculates photon (quantum) irradiance from a source_mspct object.

Note

The last two parameters control speed optimizations. The defaults should be suitable in most cases. If you will use repeatedly the same SWFs on many spectra measured at exactly the same wavelengths you may obtain some speed up by setting use.cached.mult=TRUE. However, be aware that you are responsible for ensuring that the wavelengths are the same in each call, as the only test done is for the length of the w.length vector.

See Also

Other irradiance functions: e_fluence, e_irrad, fluence, irrad, q_fluence

Examples

```
q_irrad(sun.spct, waveband(c(400,700)))
q_irrad(sun.spct, split_bands(c(400,700), length.out = 3))
q_irrad(sun.spct, split_bands(c(400,700), length.out = 3), quantity = "total")
q_irrad(sun.spct, split_bands(c(400,700), length.out = 3), quantity = "average")
q_irrad(sun.spct, split_bands(c(400,700), length.out = 3), quantity = "relative")
q_irrad(sun.spct, split_bands(c(400,700), length.out = 3), quantity = "relative.pc")
q_irrad(sun.spct, split_bands(c(400,700), length.out = 3), quantity = "contribution")
q_irrad(sun.spct, split_bands(c(400,700), length.out = 3), quantity = "contribution.pc")
```

---

q_ratio *Photon:photon ratio*

Description

This function returns the photon ratio for a given pair of wavebands of a light source spectrum.

Usage

```
q_ratio(spct, w.band.num, w.band.denom, wb.trim, use.cached.mult,
  use.hinges, ...)

## Default S3 method:
q_ratio(spct, w.band.num, w.band.denom, wb.trim,
  use.cached.mult, use.hinges, ...)

## S3 method for class 'source_spct'
q_ratio(spct, w.band.num = NULL,
  w.band.denom = NULL,
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.cached.mult = FALSE,
  use.hinges = getOption("photobiology.use.hinges"), ...)

## S3 method for class 'source_mspct'
q_ratio(spct, w.band.num = NULL,
  w.band.denom = NULL,
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.cached.mult = FALSE,
  use.hinges = getOption("photobiology.use.hinges"), ...,
  attr2tb = NULL, idx = "spct.idx", .parallel = FALSE,
  .paropts = NULL)
```

Arguments

spct	an object of class "source_spct".

w.band.num	waveband object or a list of waveband objects used to compute the numerator(s) of the ratio(s).
w.band.denom	waveband object or a list of waveband objects used to compute the denominator(s) of the ratio(s).
wb.trim	logical if TRUE wavebands crossing spectral data boundaries are trimmed, if FALSE, they are discarded
use.cached.mult	logical indicating whether multiplier values should be cached between calls
use.hinges	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.
...	other arguments (possibly ignored)
attr2tb	character vector, see add_attr2tb for the syntax for attr2tb passed as is to formal parameter col.names.
idx	character Name of the column with the names of the members of the collection of spectra.
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Value

In the case of methods for individual spectra, a numeric vector of adimensional values giving a photon ratio between integrated photon irradiances for pairs of wavebands, with name attribute set to the name of the wavebands unless a named list of wavebands is supplied in which case the names of the list elements are used, with "(q:q)" appended. A data.frame in the case of collections of spectra, containing one column for each ratio definition, an index column with the names of the spectra, and optionally additional columns with metadata values retrieved from the attributes of the member spectra.

Ratio definitions are "assembled" from the arguments passed to w.band.num and w.band.denom. If both arguments are of equal length, then the wavebands are paired to obtain as many ratios as the number of wavebands in each list. Recycling for wavebands takes place when the number of denominator and numerator wavebands differ.

Methods (by class)

- default: Default for generic function
- source_spct: Method for source_spct objects
- source_mspct: Calculates photon:photon from a source_mspct object.

Note

The last two parameters control speed optimizations. The defaults should be suitable in most cases. If you will use repeatedly the same SWFs on many spectra measured at exactly the same wavelengths you may obtain some speed up by setting `use.cached.mult=TRUE`. However, be aware that you are responsible for ensuring that the wavelengths are the same in each call, as the only test done is for the length of the `w.length` vector.

See Also

Other photon and energy ratio functions: `e_ratio`, `eq_ratio`, `qe_ratio`

Examples

```
q_ratio(sun.spct, new_waveband(400,500), new_waveband(400,700))
```

---

q_response *Photon-based photo-response*

Description

This function returns the mean response for a given waveband and a response spectrum.

Usage

```
q_response(spct, w.band, quantity, time.unit, scale.factor, wb.trim,
  use.hinges, ...)

## Default S3 method:
q_response(spct, w.band, quantity, time.unit,
  scale.factor, wb.trim, use.hinges, ...)

## S3 method for class 'response_spct'
q_response(spct, w.band = NULL,
  quantity = "total", time.unit = NULL, scale.factor = 1,
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.hinges = getOption("photobiology.use.hinges", default = NULL), ...)

## S3 method for class 'response_mspct'
q_response(spct, w.band = NULL,
  quantity = "total", time.unit = NULL, scale.factor = 1,
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.hinges = getOption("photobiology.use.hinges", default = NULL), ...,
  attr2tb = NULL, idx = "spct.idx", .parallel = FALSE,
  .paropts = NULL)
```

Arguments

spct	an R object.
w.band	waveband or list of wavebands objects or a numeric vector of length two. The waveband(s) determine the region(s) of the spectrum that are summarized. If a numeric range is supplied a waveband object is constructed on the fly from it.
quantity	character string One of "total", "average" or "mean", "contribution", "contribution.pc", "relative" or "relative.pc".
time.unit	character or lubridate::duration object.
scale.factor	numeric vector of length 1, or length equal to that of w.band. Numeric multiplier applied to returned values.
wb.trim	logical if TRUE wavebands crossing spectral data boundaries are trimmed, if FALSE, they are discarded.
use.hinges	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.
...	other arguments (possibly used by derived methods).
attr2tb	character vector, see add_attr2tb for the syntax for attr2tb passed as is to formal parameter col.names.
idx	character Name of the column with the names of the members of the collection of spectra.
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Value

A named numeric vector in the case of methods for individual spectra, with one value for each waveband passed to parameter w.band. A data.frame in the case of collections of spectra, containing one column for each waveband object, an index column with the names of the spectra, and optionally additional columns with metadata values retrieved from the attributes of the member spectra.

By default values are only integrated, but depending on the argument passed to parameter quantity they can be re-expressed as relative fractions or percentages. In the case of vector output, names attribute is set to the name of the corresponding waveband unless a named list is supplied in which case the names of the list members are used.

Methods (by class)

- default: Default method for generic function
- response_spct: Method for response spectra.
- response_mspct: Calculates photon (quantum) response from a response_mspct

Note

The parameter `use.hinges` controls speed optimization. The defaults should be suitable in most cases. Only the range of wavelengths in the wavebands is used and all BSWFs are ignored.

See Also

Other response functions: `e_response`, `response`

Examples

```
q_response(ccd.spct, new_waveband(200,300))
q_response(photodiode.spct)
```

`rbindspct` *Row-bind spectra*

Description

A wrapper on `dplyr::rbind_fill` that preserves class and other attributes of spectral objects.

Usage

```
rbindspct(l, use.names = TRUE, fill = TRUE, idfactor = TRUE)
```

Arguments

`l`	A `source_mspct`, `filter_mspct`, `reflector_mspct`, `response_mspct`, `chroma_mspct`, `cps_mspct`, `generic_mspct` object or a list containing `source_spct`, `filter_spct`, `reflector_spct`, `response_spct`, `chroma_spct`, `cps_spct`, or `generic_spct` objects.
`use.names`	logical If `TRUE` items will be bound by matching column names. By default `TRUE` for `rbindspct`. Columns with duplicate names are bound in the order of occurrence, similar to base. When TRUE, at least one item of the input list has to have non-null column names.
`fill`	logical If `TRUE` fills missing columns with NAs. By default `TRUE`. When `TRUE`, `use.names` has also to be `TRUE`, and all items of the input list have to have non-null column names.
`idfactor`	logical or character Generates an index column of `factor` type. Default (`TRUE`) is to for both lists and `_mspct` objects. If `idfactor=TRUE` then the column is auto named `spct.idx`. Alternatively the column name can be directly provided to `idfactor` as a character string.

Details

Each item of `l` should be a spectrum, including `NULL` (skipped) or an empty object (0 rows). `rbindspc` is most useful when there are a variable number of (potentially many) objects to stack. `rbindspct` always returns at least a `generic_spct` as long as all elements in l are spectra.

Value

An spectral object of a type common to all bound items containing a concatenation of all the items passed in. If the argument 'idfactor' is TRUE, then a factor 'spct.idx' will be added to the returned spectral object.

Note

Note that any additional 'user added' attributes that might exist on individual items of the input list will not be preserved in the result. The attributes used by the `photobiology` package are preserved, and if they are not consistent across the bound spectral objects, a warning is issued.

`dplyr::rbind_fill` is called internally and the result returned is the highest class in the inheritance hierarchy which is common to all elements in the list. If not all members of the list belong to one of the `_spct` classes, an error is triggered. The function sets all data in `source_spct` and `response_spct` objects supplied as arguments into energy-based quantities, and all data in `filter_spct` objects into transmittance before the row binding is done. If any member spectrum is tagged, it is untagged before row binding.

Examples

```
spct <- rbindspct(list(sun.spct, sun.spct))
spct
class(spct)

# adds factor 'spct.idx' with letters as levels
spct <- rbindspct(list(sun.spct, sun.spct), idfactor = TRUE)
head(spct)
class(spct)

# adds factor 'spct.idx' with the names given to the spectra in the list
# supplied as formal argument 'l' as levels
spct <- rbindspct(list(one = sun.spct, two = sun.spct), idfactor = TRUE)
head(spct)
class(spct)

# adds factor 'ID' with the names given to the spectra in the list
# supplied as formal argument 'l' as levels
spct <- rbindspct(list(one = sun.spct, two = sun.spct),
                  idfactor = "ID")
head(spct)
class(spct)
```

---

reflectance *Reflectance*

Description

Function to calculate the mean, total, or other summary of reflectance for spectral data stored in a `reflector_spct` or in an `object_spct`.

Usage

```
reflectance(spct, w.band, quantity, wb.trim, use.hinges, ...)

## Default S3 method:
reflectance(spct, w.band, quantity, wb.trim, use.hinges,
  ...)

## S3 method for class 'reflector_spct'
reflectance(spct, w.band = NULL,
  quantity = "average",
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.hinges = getOption("photobiology.use.hinges", default = NULL), ...)

## S3 method for class 'object_spct'
reflectance(spct, w.band = NULL,
  quantity = "average",
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.hinges = getOption("photobiology.use.hinges", default = NULL), ...)

## S3 method for class 'reflector_mspct'
reflectance(spct, w.band = NULL,
  quantity = "average",
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.hinges = getOption("photobiology.use.hinges", default = NULL), ...,
  attr2tb = NULL, idx = "spct.idx", .parallel = FALSE,
  .paropts = NULL)

## S3 method for class 'object_mspct'
reflectance(spct, w.band = NULL,
  quantity = "average",
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.hinges = getOption("photobiology.use.hinges", default = NULL), ...,
  attr2tb = NULL, idx = "spct.idx", .parallel = FALSE,
  .paropts = NULL)
```

Arguments

spct	an R object
w.band	waveband or list of waveband objects or a numeric vector of length two. The waveband(s) determine the region(s) of the spectrum that are summarized. If a numeric range is supplied a waveband object is constructed on the fly from it.
quantity	character string One of "total", "average" or "mean", "contribution", "contribution.pc", "relative" or "relative.pc"
wb.trim	logical Flag telling if wavebands crossing spectral data boundaries are trimmed or ignored
use.hinges	logical Flag indicating whether to insert "hinges" into the spectral data before

	integration so as to reduce interpolation errors at the boundaries of the wavebands.
...	other arguments
attr2tb	character vector, see add_attr2tb for the syntax for attr2tb passed as is to formal parameter col.names.
idx	character Name of the column with the names of the members of the collection of spectra.
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Value

A named numeric vector in the case of methods for individual spectra, with one value for each waveband passed to parameter w.band. A data.frame in the case of collections of spectra, containing one column for each waveband object, an index column with the names of the spectra, and optionally additional columns with metadata values retrieved from the attributes of the member spectra.

By default values are only integrated, but depending on the argument passed to parameter quantity they can be re-expressed as relative fractions or percentages. In the case of vector output, names attribute is set to the name of the corresponding waveband unless a named list is supplied in which case the names of the list members are used.

Methods (by class)

- default: Default for generic function
- reflector_spct: Specialization for reflector_spct
- object_spct: Specialization for object_spct
- reflector_mspct: Calculates reflectance from a reflector_mspct
- object_mspct: Calculates reflectance from a object_mspct

Note

The use.hinges parameter controls speed optimization. The defaults should be suitable in most cases. Only the range of wavelengths in the wavebands is used and all BSWFs are ignored.

Examples

```
reflectance(black_body.spct, waveband(c(400,700)))
reflectance(white_body.spct, waveband(c(400,700)))
```

relative_AM *Relative Air Mass (AM)*

Description

Approximate relative air mass (AM) from sun elevation or sun zenith angle.

Usage

```
relative_AM(elevation.angle = NULL, zenith.angle = NULL,
  occluded.value = NA)
```

Arguments

elevation.angle, zenith.angle	numeric vector Angle in degrees for the sun position. An argument should be passed to one and only one of elevation_angle or zenith_angle.
occluded.value	numeric Value to return when elevation angle is negative (sun below the horizon).

Details

This is an implementation of equation (3) in Kasten and Young (1989). This equation is only an approximation to the tabulated values in the same paper. Returned values are rounded to three significant digits.

Note

Although relative air mass is not defined when the sun is not visible, returning a value different from the default NA might be useful in some cases.

References

F. Kasten, A. T. Young (1989) Revised optical air mass tables and approximation formula. Applied Optics, 28, 4735-. doi:10.1364/ao.28.004735.

Examples

```
relative_AM(c(90, 60, 30, 1, -10))
relative_AM(c(90, 60, 30, 1, -10), occluded.value = Inf)
relative_AM(zenith.angle = 0)
```

response *Integrated response*

Description

Calculate average photon- or energy-based photo-response.

Usage

```
response(spct, w.band, unit.out, quantity, time.unit, scale.factor,
  wb.trim, use.hinges, ...)

## Default S3 method:
response(spct, w.band, unit.out, quantity, time.unit,
  scale.factor, wb.trim, use.hinges, ...)

## S3 method for class 'response_spct'
response(spct, w.band = NULL,
  unit.out = getOption("photobiology.radiation.unit", default =
  "energy"), quantity = "total", time.unit = NULL, scale.factor = 1,
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.hinges = getOption("photobiology.use.hinges", default = NULL), ...)

## S3 method for class 'response_mspct'
response(spct, w.band = NULL,
  unit.out = getOption("photobiology.radiation.unit", default =
  "energy"), quantity = "total", time.unit = NULL, scale.factor = 1,
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.hinges = getOption("photobiology.use.hinges", default = NULL), ...,
  attr2tb = NULL, idx = "spct.idx", .parallel = FALSE,
  .paropts = NULL)
```

Arguments

spct	an R object of class "generic_spct".
w.band	waveband or list of waveband objects or a numeric vector of length two. The waveband(s) determine the region(s) of the spectrum that are summarized. If a numeric range is supplied a waveband object is constructed on the fly from it.
unit.out	character Allowed values "energy", and "photon", or its alias "quantum".
quantity	character string One of "total", "average" or "mean", "contribution", "contribution.pc", "relative" or "relative.pc".
time.unit	character or lubridate::duration object.
scale.factor	numeric vector of length 1, or length equal to that of w.band. Numeric multiplier applied to returned values.
wb.trim	logical Flag telling if wavebands crossing spectral data boundaries are trimmed or ignored.

use.hinges	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.
...	other arguments (possibly used by derived methods).
attr2tb	character vector, see add_attr2tb for the syntax for attr2tb passed as is to formal parameter col.names.
idx	character Name of the column with the names of the members of the collection of spectra.
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Value

A named numeric vector in the case of methods for individual spectra, with one value for each waveband passed to parameter w.band. A data.frame in the case of collections of spectra, containing one column for each waveband object, an index column with the names of the spectra, and optionally additional columns with metadata values retrieved from the attributes of the member spectra.

Whether returned values are expressed in energy-based or photon-based units depends on unit.out. By default values are only integrated, but depending on the argument passed to parameter quantity they can be re-expressed as relative fractions or percentages. In the case of vector output, names attribute is set to the name of the corresponding waveband unless a named list is supplied in which case the names of the list members are used.

Methods (by class)

- default: Default for generic function
- response_spct: Method for response spectra.
- response_mspct: Calculates response from a response_mspct

Note

The parameter use.hinges controls speed optimization. The defaults should be suitable in most cases. Only the range of wavelengths in the wavebands is used and all BSWFs are ignored.

See Also

Other response functions: e_response, q_response

rgb_spct *RGB color values*

Description

This function returns the RGB values for a source spectrum.

Usage

```
rgb_spct(spct, sens = photobiology::ciexyzCMF2.spct, color.name = NULL)
```

Arguments

spct	an object of class "source_spct"
sens	a chroma_spct object with variables w.length, x, y, and z, giving the CC or CMF definition (default is the proposed human CMF according to CIE 2006.)
color.name	character string for naming the rgb color definition

Value

A color defined using `rgb()`. The numeric values of the RGB components can be obtained

See Also

Other color functions: `w_length2rgb`, `w_length_range2rgb`

Examples

```
rgb_spct(sun.spct)
```

rmDerivedMspct *Remove "generic_mspct" and derived class attributes.*

Description

Removes from an spectrum object the class attributes "generic_mspct" and any derived class attribute such as "source_mspct". **This operation is done by reference!**

Usage

```
rmDerivedMspct(x)
```

Arguments

x	an R object.

Value

A character vector containing the removed class attribute values. This is different to the behaviour of function `unlist` in base R!

Note

If `x` is an object of any of the multi spectral classes defined in this package, this function changes by reference the multi spectrum object into the underlying list object. Otherwise, it just leaves `x` unchanged. The modified `x` is also returned invisibly.

See Also

Other set and unset 'multi spectral' class functions: `shared_member_class`

`rmDerivedSpct` *Remove "generic_spct" and derived class attributes.*

Description

Removes from an spectrum object the class attributes "generic_spct" and any derived class attribute such as "source_spct". **This operation is done by reference!**

Usage

```
rmDerivedSpct(x)
```

Arguments

`x`	an R object.

Value

A character vector containing the removed class attribute values. This is different to the behaviour of function `unlist` in base R!

Note

If `x` is an object of any of the spectral classes defined in this package, this function changes by reference the spectrum object into the underlying data.frame object. Otherwise, it just leaves `x` unchanged.

This function alters x itself by reference. If x is not a generic_spct object, x is not modified.

See Also

Other set and unset spectral class functions: `setGenericSpct`

Examples

```
my.spct <- sun.spct
removed <- rmDerivedSpct(my.spct)
removed
class(sun.spct)
class(my.spct)
```

---

round *Rounding of Numbers*

Description

ceiling takes a single numeric argument x and returns a numeric vector containing the smallest integers not less than the corresponding elements of x. \ floor takes a single numeric argument x and returns a numeric vector containing the largest integers not greater than the corresponding elements of x. \ trunc takes a single numeric argument x and returns a numeric vector containing the integers formed by truncating the values in x toward 0. \ round rounds the values in its first argument to the specified number of decimal places (default 0). \ signif rounds the values in its first argument to the specified number of significant digits. \ The functions are applied to the spectral data, not the wavelengths. The quantity in the spectrum to which the function is applied depends on the class of x and the current value of output options.

Usage

```
## S3 method for class 'generic_spct'
round(x, digits = 0)

## S3 method for class 'generic_spct'
signif(x, digits = 6)

## S3 method for class 'generic_spct'
ceiling(x)

## S3 method for class 'generic_spct'
floor(x)

## S3 method for class 'generic_spct'
trunc(x, ...)
```

Arguments

x	an object of class "generic_spct" or a derived class.
digits	integer indicating the number of decimal places (round) or significant digits (signif) to be used. Negative values are allowed (see 'Details').
...	arguments to be passed to methods.

See Also

Other math operators and functions: MathFun, ^.generic_spct, convolve_each, div-.generic_spct, log, minus-.generic_spct, mod-.generic_spct, plus-.generic_spct, sign, slash-.generic_spct, times-.generic_spct

---

`setAfrType` *Set the "Afr.type" attribute*

Description

Function to set by reference the "Afr.type" attribute of an existing filter_spct or object_spct object

Usage

```
setAfrType(x, Afr.type = c("total", "internal"))
```

Arguments

`x`	a filter_spct or an object_spct object
`Afr.type`	a character string, either "total" or "internal"

Value

x

Note

This function alters x itself by reference and in addition returns x invisibly. If x is not a filter_spct or an object_spct object, x is not modified The behaviour of this function is 'unusual' in that the default for parameter `Afr.type` is used only if `x` does not already have this attribute set.

See Also

Other Afr attribute functions: getAfrType

Examples

```
my.spct <- polyester.spct
getAfrType(my.spct)
setAfrType(my.spct, "internal")
getAfrType(my.spct)
```

setBSWFUsed *Set the "bswf.used" attribute*

Description

Function to set by reference the "time.unit" attribute of an existing source_spct object

Usage

```
setBSWFUsed(x, bswf.used = c("none", "unknown"))
```

Arguments

x	a source_spct object
bswf.used	a character string, either "none" or the name of a BSWF

Value

x

Note

This function alters x itself by reference and in addition returns x invisibly. If x is not a source_spct, x is not modified. The behaviour of this function is 'unusual' in that the default for parameter `bswf.used` is used only if `x` does not already have this attribute set. `time.unit = "hour"` is currently not fully supported.

See Also

Other BSWF attribute functions: `getBSWFUsed`

setGenericSpct *Convert an R object into a spectrum object.*

Description

Sets the class attribute of a data.frame or an object of a derived class to "generic_spct".

Usage

```
setGenericSpct(x, multiple.wl = 1L, idfactor = NULL)

setCalibrationSpct(x,
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  multiple.wl = 1L, idfactor = NULL)

setRawSpct(x, strict.range = getOption("photobiology.strict.range",
  default = FALSE), multiple.wl = 1L, idfactor = NULL)

setCpsSpct(x, time.unit = "second",
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  multiple.wl = 1L, idfactor = NULL)

setFilterSpct(x, Tfr.type = c("total", "internal"),
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  multiple.wl = 1L, idfactor = NULL)

setReflectorSpct(x, Rfr.type = c("total", "specular"),
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  multiple.wl = 1L, idfactor = NULL)

setObjectSpct(x, Tfr.type = c("total", "internal"),
  Rfr.type = c("total", "specular"),
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  multiple.wl = 1L, idfactor = NULL)

setResponseSpct(x, time.unit = "second", multiple.wl = 1L,
  idfactor = NULL)

setSourceSpct(x, time.unit = "second", bswf.used = c("none",
  "unknown"), strict.range = getOption("photobiology.strict.range",
  default = FALSE), multiple.wl = 1L, idfactor = NULL)

setChromaSpct(x, multiple.wl = 1L, idfactor = NULL)
```

Arguments

x	data.frame, list or generic_spct and derived classes
multiple.wl	numeric Maximum number of repeated w.length entries with same value.
idfactor	character Name of factor distinguishing multiple spectra when stored logitudinally (required if mulitple.wl > 1).
strict.range	logical Flag indicating whether off-range values result in an error instead of a warning.
time.unit	character A string "second", "day" or "exposure".
Tfr.type	character A string, either "total" or "internal".
Rfr.type	character A string, either "total" or "specular".

bswf.used character A string, either "none" or the name of a BSWF.

Value

x

Functions

- setCalibrationSpct: Set class of a an object to "calibration_spct".
- setRawSpct: Set class of a an object to "raw_spct".
- setCpsSpct: Set class of a an object to "cps_spct".
- setFilterSpct: Set class of an object to "filter_spct".
- setReflectorSpct: Set class of a an object to "reflector_spct".
- setObjectSpct: Set class of an object to "object_spct".
- setResponseSpct: Set class of an object to "response_spct".
- setSourceSpct: Set class of an object to "source_spct".
- setChromaSpct: Set class of an object to "chroma_spct".

Note

This method alters x itself by reference and in addition returns x invisibly.

See Also

Other set and unset spectral class functions: rmDerivedSpct

Examples

```
my.df <- data.frame(w.length = 300:309, s.e.irrad = rep(100, 10))
is.source_spct(my.df)
setSourceSpct(my.df)
is.source_spct(my.df)
```

---

setHowMeasured *Set the "how.measured" attribute*

Description

Function to set by reference the "how.measured" attribute of an existing generic_spct or derived-class object.

Usage

```
setHowMeasured(x, how.measured)
```

Arguments

x	a generic_spct object
how.measured	a list

Value

x

Note

This function alters x itself by reference and in addition returns x invisibly. If x is not a generic_spct object, x is not modified.

See Also

Other measurement metadata functions: getHowMeasured, getInstrDesc, getInstrSettings, getWhatMeasured, getWhenMeasured, getWhereMeasured, get_attributes, isValidInstrDesc, isValidInstrSettings, setInstrDesc, setInstrSettings, setWhatMeasured, setWhenMeasured, setWhereMeasured, trimInstrDesc, trimInstrSettings

---

setIdFactor *Set the "idfactor" attribute*

Description

Function to set by reference the "idfactor" attribute of an existing generic_spct or an object of a class derived from generic_spct.

Usage

```
setIdFactor(x, idfactor)
```

Arguments

x	a generic_spct object
idfactor	character The name of a factor identifying multiple spectra stored longitudinally.

Value

x

Note

This function alters x itself by reference and in addition returns x invisibly. If x is not a generic_spct or an object of a class derived from generic_spct, x is not modified.

See Also

Other idfactor attribute functions: getIdFactor

setInstrDesc *Set the "instr.desc" attribute*

Description

Function to set by reference the "instr.desc" attribute of an existing generic_spct or derived-class object.

Usage

```
setInstrDesc(x, instr.desc)
```

Arguments

x	a generic_spct object
instr.desc	a list

Value

x

Note

This function alters x itself by reference and in addition returns x invisibly. If x is not a generic_spct object, x is not modified.

See Also

Other measurement metadata functions: getHowMeasured, getInstrDesc, getInstrSettings, getWhatMeasured, getWhenMeasured, getWhereMeasured, get_attributes, isValidInstrDesc, isValidInstrSettings, setHowMeasured, setInstrSettings, setWhatMeasured, setWhenMeasured, setWhereMeasured, trimInstrDesc, trimInstrSettings

setInstrSettings *Set the "instr.settings" attribute*

Description

Function to set by reference the "what.measured" attribute of an existing generic_spct or derived-class object.

Usage

```
setInstrSettings(x, instr.settings)
```

Arguments

x	a generic_spct object
instr.settings	a list

Value

x

Note

This function alters x itself by reference and in addition returns x invisibly. If x is not a generic_spct object, x is not modified.

See Also

Other measurement metadata functions: getHowMeasured, getInstrDesc, getInstrSettings, getWhatMeasured, getWhenMeasured, getWhereMeasured, get_attributes, isValidInstrDesc, isValidInstrSettings, setHowMeasured, setInstrDesc, setWhatMeasured, setWhenMeasured, setWhereMeasured, trimInstrDesc, trimInstrSettings

---

setMultipleWl *Set the "multiple.wl" attribute*

Description

Function to set by reference the "multiple.wl" attribute of an existing generic_spct or an object of a class derived from generic_spct.

Usage

```
setMultipleWl(x, multiple.wl = NULL)
```

Arguments

x	a generic_spct object
multiple.wl	numeric >= 1 If multiple.wl is NULL, the default, the attribute is not modified if it is already present and valid, and set to 1 otherwise.

Value

x

Note

This function alters x itself by reference and in addition returns x invisibly. If x is not a generic_spct or an object of a class derived from generic_spct, x is not modified. If multiple.wl

See Also

Other multiple.wl attribute functions: getMultipleWl

---

setNormalized *Set the "normalized" attribute*

Description

Function to write the "normalized" attribute of an existing generic_spct object.

Usage

```
setNormalized(x, norm = FALSE)
```

Arguments

x	a generic_spct object
norm	numeric or logical

Note

if x is not a generic_spct object, x is not modified.

See Also

Other rescaling functions: fscale, fshift, getNormalized, is_normalized, is_scaled, normalize, setScaled

---

setRfrType *Set the "Rfr.type" attribute*

Description

Function to set by reference the "Rfr.type" attribute of an existing reflector_spct or object_spct object.

Usage

```
setRfrType(x, Rfr.type = c("total", "specular"))
```

Arguments

x	a reflector_spct or an object_spct object
Rfr.type	a character string, either "total" or "specular"

Value

x

Note

This function alters x itself by reference and in addition returns x invisibly. If x is not a reflector_spct or object_spct object, x is not modified. The behaviour of this function is 'unusual' in that the default for parameter Rfr.type is used only if x does not already have this attribute set.

See Also

Other Rfr attribute functions: `getRfrType`, `getScaled`

Examples

```
my.spct <- reflector_spct(w.length = 400:409, Rfr = 0.1)
getRfrType(my.spct)
setRfrType(my.spct, "specular")
getRfrType(my.spct)
```

---

setScaled *Set the "scaled" attribute*

Description

Function to write the "scaled" attribute of an existing generic_spct object.

Usage

```
setScaled(x, scaled = FALSE)
```

Arguments

x	a generic_spct object
scaled	logical

Note

if x is not a `generic_spct` object, x is not modified. attribute set.

See Also

Other rescaling functions: `fscale`, `fshift`, `getNormalized`, `is_normalized`, `is_scaled`, `normalize`, `setNormalized`

`setTfrType` *Set the "Tfr.type" attribute*

Description

Function to set by reference the "Tfr.type" attribute of an existing filter_spct or object_spct object

Usage

```
setTfrType(x, Tfr.type = c("total", "internal"))
```

Arguments

`x`	a filter_spct or an object_spct object
`Tfr.type`	a character string, either "total" or "internal"

Value

x

Note

This function alters x itself by reference and in addition returns x invisibly. If x is not a filter_spct or an object_spct object, x is not modified The behaviour of this function is 'unusual' in that the default for parameter `Tfr.type` is used only if `x` does not already have this attribute set.

See Also

Other Tfr attribute functions: `getTfrType`

Examples

```
my.spct <- polyester.spct
getTfrType(my.spct)
setTfrType(my.spct, "internal")
getTfrType(my.spct)
```

setTimeUnit *Set the "time.unit" attribute of an existing source_spct object*

Description

Function to set by reference the "time.unit" attribute

Usage

```
setTimeUnit(x, time.unit = c("second", "hour", "day", "exposure",
  "none"), override.ok = FALSE)
```

Arguments

x	a source_spct object
time.unit	a character string, either "second", "hour", "day", "exposure" or "none", or a lubridate::duration
override.ok	logical Flag that can be used to silence warning when overwriting an existing attribute value (used internally)

Value

x

Note

This function alters x itself by reference and in addition returns x invisibly. If x is not a source_spct or response_spct object, x is not modified. The behaviour of this function is 'unusual' in that the default for parameter `time.unit` is used only if x does not already have this attribute set. `time.unit = "hour"` is currently not fully supported.

See Also

Other time attribute functions: checkTimeUnit, convertTimeUnit, getTimeUnit

Examples

```
my.spct <- sun.spct
setTimeUnit(my.spct, time.unit = "second")
setTimeUnit(my.spct, time.unit = lubridate::duration(1, "seconds"))
```

setWhatMeasured *Set the "what.measured" attribute*

Description

Function to set by reference the "what.measured" attribute of an existing generic_spct or derived-class object.

Usage

```
setWhatMeasured(x, what.measured)
```

Arguments

x	a generic_spct object
what.measured	a list

Value

x

Note

This function alters x itself by reference and in addition returns x invisibly. If x is not a generic_spct object, x is not modified.

See Also

Other measurement metadata functions: getHowMeasured, getInstrDesc, getInstrSettings, getWhatMeasured, getWhenMeasured, getWhereMeasured, get_attributes, isValidInstrDesc, isValidInstrSettings, setHowMeasured, setInstrDesc, setInstrSettings, setWhenMeasured, setWhereMeasured, trimInstrDesc, trimInstrSettings

setWhenMeasured *Set the "when.measured" attribute*

Description

Function to set by reference the "when" attribute of an existing generic_spct or an object of a class derived from generic_spct.

Usage

```
setWhenMeasured(x, when.measured, ...)

## Default S3 method:
setWhenMeasured(x, when.measured, ...)

## S3 method for class 'generic_spct'
setWhenMeasured(x,
  when.measured = lubridate::now(tzone = "UTC"), ...)

## S3 method for class 'summary_generic_spct'
setWhenMeasured(x,
  when.measured = lubridate::now(tzone = "UTC"), ...)

## S3 method for class 'generic_mspct'
setWhenMeasured(x,
  when.measured = lubridate::now(tzone = "UTC"), ...)
```

Arguments

x	a generic_spct object
when.measured	POSIXct to add as attribute, or a list of POSIXct.
...	Allows use of additional arguments in methods for other classes.

Value

x

Methods (by class)

- default: default
- generic_spct: generic_spct
- summary_generic_spct: summary_generic_spct
- generic_mspct: generic_mspct

Note

This method alters x itself by reference and in addition returns x invisibly. If x is not a generic_spct or an object of a class derived from generic_spct, x is not modified. If when is not a POSIXct object or NULL an error is triggered. A POSIXct describes an instant in time (date plus time-of-day plus time zone).

See Also

Other measurement metadata functions: getHowMeasured, getInstrDesc, getInstrSettings, getWhatMeasured, getWhenMeasured, getWhereMeasured, get_attributes, isValidInstrDesc, isValidInstrSettings, setHowMeasured, setInstrDesc, setInstrSettings, setWhatMeasured, setWhereMeasured, trimInstrDesc, trimInstrSettings

Examples

```
my.spct <- sun.spct
getWhenMeasured(my.spct)
setWhenMeasured(my.spct, lubridate::ymd_hms("2015-12-12 08:00:00"))
getWhenMeasured(my.spct)
```

`setWhereMeasured` *Set the "where.measured" attribute*

Description

Function to set by reference the "where.measured" attribute of an existing generic_spct or an object of a class derived from generic_spct.

Usage

```
setWhereMeasured(x, where.measured, lat, lon, ...)

## Default S3 method:
setWhereMeasured(x, where.measured, lat, lon, ...)

## S3 method for class 'generic_spct'
setWhereMeasured(x, where.measured = NA,
  lat = NA, lon = NA, ...)

## S3 method for class 'summary_generic_spct'
setWhereMeasured(x, where.measured = NA,
  lat = NA, lon = NA, ...)

## S3 method for class 'generic_mspct'
setWhereMeasured(x, where.measured = NA,
  lat = NA, lon = NA, ...)
```

Arguments

`x`	a generic_spct object
`where.measured`	A one row data.frame such as returned by function geocode from package 'ggmap' for a location search.
`lat`	numeric Latitude in decimal degrees North
`lon`	numeric Longitude in decimal degrees West
`...`	Allows use of additional arguments in methods for other classes.

Value

x

Methods (by class)

- `default`: default
- `generic_spct`: generic_spct
- `summary_generic_spct`: summary_generic_spct
- `generic_mspct`: generic_mspct

Note

This method alters x itself by reference and in addition returns x invisibly. If x is not a generic_spct or an object of a class derived from generic_spct, x is not modified. If `where` is not a POSIXct object or `NULL` an error is triggered. A `POSIXct` describes an instant in time (date plus time-of-day plus time zone).

Method for collections of spectra recycles the location information only if it is of length one.

See Also

Other measurement metadata functions: `getHowMeasured`, `getInstrDesc`, `getInstrSettings`, `getWhatMeasured`, `getWhenMeasured`, `getWhereMeasured`, `get_attributes`, `isValidInstrDesc`, `isValidInstrSettings`, `setHowMeasured`, `setInstrDesc`, `setInstrSettings`, `setWhatMeasured`, `setWhenMeasured`, `trimInstrDesc`, `trimInstrSettings`

`shared_member_class` *Classes common to all collection members.*

Description

Finds the set intersection among the class attributes of all collection member as a target set of class names.

Usage

```
shared_member_class(l, target.set = spct_classes())
```

Arguments

`l`	a list or a generic_mspct object or of a derived class.
`target.set`	character The target set of classes within which to search for classes common to all members.

Value

A character vector containing the class attribute values.

See Also

Other set and unset 'multi spectral' class functions: `rmDerivedMspct`

sign *Sign*

Description

sign returns a vector with the signs of the corresponding elements of x (the sign of a real number is 1, 0, or -1 if the number is positive, zero, or negative, respectively).

Usage

```
## S3 method for class 'generic_spct'
sign(x)
```

Arguments

x	an object of class "generic_spct"

See Also

Other math operators and functions: MathFun, ^.generic_spct, convolve_each, div-.generic_spct, log, minus-.generic_spct, mod-.generic_spct, plus-.generic_spct, round, slash-.generic_spct, times-.generic_spct

slash-.generic_spct *Arithmetic Operators*

Description

Division operator for generic spectra.

Usage

```
## S3 method for class 'generic_spct'
e1 / e2
```

Arguments

e1	an object of class "generic_spct"
e2	an object of class "generic_spct"

See Also

Other math operators and functions: MathFun, ^.generic_spct, convolve_each, div-.generic_spct, log, minus-.generic_spct, mod-.generic_spct, plus-.generic_spct, round, sign, times-.generic_spct

smooth_spct *Smooth a spectrum*

Description

These functions implement one original methods and acts as a wrapper for other common R smoothing functions. The advantage of using this function for smoothing spectral objects is that it simplifies the user interface and sets, when needed, defaults suitable for spectral data.

Usage

```
smooth_spct(x, method, strength, ...)

## Default S3 method:
smooth_spct(x, method, strength, ...)

## S3 method for class 'source_spct'
smooth_spct(x, method = "custom", strength = 1,
  na.rm = FALSE, ...)

## S3 method for class 'filter_spct'
smooth_spct(x, method = "custom", strength = 1,
  na.rm = FALSE, ...)

## S3 method for class 'reflector_spct'
smooth_spct(x, method = "custom",
  strength = 1, na.rm = FALSE, ...)

## S3 method for class 'response_spct'
smooth_spct(x, method = "custom", strength = 1,
  na.rm = FALSE, ...)
```

Arguments

x	an R object.
method	a character string "custom", "lowess", "supsmu".
strength	numeric value to adjust the degree of smoothing.
...	other parameters passed to the underlying smoothing functions.
na.rm	logical A value indicating whether NA values should be stripped before the computation proceeds.

Value

A copy of x with spectral data values replaced by smoothed ones.

Methods (by class)

- `default`: Default for generic function
- `source_spct`: Smooth a source spectrum
- `filter_spct`: Smooth a filter spectrum
- `reflector_spct`: Smooth a reflector spectrum
- `response_spct`: Smooth a response spectrum

Note

Method "custom" is our home-brewed method which applies strong smoothing to low signal regions of the spectral data, and weaker or no smoothing to the high signal areas. Values very close to zero are set to zero with a limit which depends on the local variation. This method is an ad-hock method suitable for smoothing spectral data obtained with spectrometers. In the cased of methods "lowess" and "supsmu" the current function behaves like a wrapper of the functions of the same names from base R.

Examples

```
my.spct <- clip_wl(sun.spct, c(400, 500))
smooth_spct(my.spct)
smooth_spct(my.spct, method = "supsmu", strength = 4)
```

`solar_time` *Local solar time*

Description

`solar_time` computes from a time and geocode, the time of day expressed in seconds since midnight. `solar_date` returns the same instant in time as a date-time object. Solar time is useful when we want to plot data according to the local solar time of day, irrespective of the date. Solar date is useful when we want to plot a time series stretching for several days using the local solar time but distinguishing between days.

Usage

```
solar_time(time = lubridate::now(), geocode = tibble::tibble(lon = 0,
  lat = 51.5, address = "Greenwich"), unit.out = "time")
```

Arguments

`time`	POSIXct Time, any valid time zone (TZ) is allowed, default is current time
`geocode`	data frame with variables lon and lat as numeric values (degrees).
`unit.out`	character string, One of "datetime", "hour", "minute", or "second".

Value

For `solar_time()` numeric value in seconds from midnight but with an additional class attribute "solar.time".

Warning!

Returned values are computed based on the time zone of the argument for parameter time. In the case of solar time, this timezone does not affect the result. However, in the case of solar dates the date part may be off by one day, if the time zone does not match the coordinates of the geocode value provided as argument.

Note

The algorithm is approximate, it calculates the difference between local solar noon and noon in the time zone of `time` and uses this value for the whole day when converting times into solar time. Days are not exactly 24 h long. Between successive days the shift is only a few seconds, and this leads to a small jump at midnight.

See Also

Other astronomy related functions: `day_night`, `format.solar_time`, `is.solar_time`, `print.solar_time`, `sun_angles`

Examples

```
BA.geocode <-
  data.frame(lon = -58.38156, lat = -34.60368, address = "Buenos Aires, Argentina")
sol_t <- solar_time(lubridate::dmy_hms("21/06/2016 10:00:00", tz = "UTC"),
                    BA.geocode)
sol_t
class(sol_t)

sol_d <- solar_time(lubridate::dmy_hms("21/06/2016 10:00:00", tz = "UTC"),
                    BA.geocode,
                    unit.out = "datetime")
sol_d
class(sol_d)
```

---

source_spct *Spectral-object constructor*

Description

These functions can be used to create spectral objects derived from `generic_spct`. They take as arguments numeric vectors for the data character scalars for attributes, and a logical flag.

Usage

```
source_spct(w.length = NULL, s.e.irrad = NULL, s.q.irrad = NULL,
  time.unit = c("second", "day", "exposure"), bswf.used = c("none",
  "unknown"), comment = NULL,
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  multiple.wl = 1L, idfactor = NULL, ...)

calibration_spct(w.length = NULL, irrad.mult = NA_real_,
  comment = NULL, instr.desc = NA, multiple.wl = 1L,
  idfactor = NULL, ...)

raw_spct(w.length = NULL, counts = NA_real_, comment = NULL,
  instr.desc = NA, instr.settings = NA, multiple.wl = 1L,
  idfactor = NULL, ...)

cps_spct(w.length = NULL, cps = NA_real_, comment = NULL,
  instr.desc = NA, instr.settings = NA, multiple.wl = 1L,
  idfactor = NULL, ...)

generic_spct(w.length = NULL, comment = NULL, multiple.wl = 1L,
  idfactor = NULL, ...)

response_spct(w.length = NULL, s.e.response = NULL,
  s.q.response = NULL, time.unit = c("second", "day", "exposure"),
  comment = NULL, multiple.wl = 1L, idfactor = NULL, ...)

filter_spct(w.length = NULL, Tfr = NULL, Tpc = NULL, Afr = NULL,
  A = NULL, Tfr.type = c("total", "internal"), Afr.type = Tfr.type,
  comment = NULL, strict.range = getOption("photobiology.strict.range",
  default = FALSE), multiple.wl = 1L, idfactor = NULL, ...)

reflector_spct(w.length = NULL, Rfr = NULL, Rpc = NULL,
  Rfr.type = c("total", "specular"), comment = NULL,
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  multiple.wl = 1L, idfactor = NULL, ...)

object_spct(w.length = NULL, Rfr = NULL, Tfr = NULL, Afr = NULL,
  Tfr.type = c("total", "internal"), Rfr.type = c("total", "specular"),
  Afr.type = c("total", "internal"), comment = NULL,
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  multiple.wl = 1L, idfactor = NULL, ...)

chroma_spct(w.length = NULL, x, y, z, comment = NULL,
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  multiple.wl = 1L, idfactor = NULL, ...)
```

Arguments

w.length	numeric vector with wavelengths in nanometres
s.e.irrad	numeric vector with spectral energy irradiance in [W m-2 nm-1] or [J d-1 m-2 nm-1]
s.q.irrad	numeric A vector with spectral photon irradiance in [mol s-1 m-2 nm-1] or [mol d-1 m-2 nm-1].
time.unit	character string indicating the time unit used for spectral irradiance or exposure ("second" , "day" or "exposure") or an object of class duration as defined in package lubridate.
bswf.used	character A string indicating the BSWF used, if any, for spectral effective irradiance or exposure ("none" or the name of the BSWF).
comment	character A string to be added as a comment attribute to the object created.
strict.range	logical Flag indicating whether off-range values result in an error instead of a warning.
multiple.wl	numeric Maximum number of repeated w.length entries with same value.
idfactor	character Name of factor distinguishing multiple spectra when stored logitudinally (required if mulitple.wl > 1).
...	other arguments passed to tibble()
irrad.mult	numeric vector with multipliers for each detector pixel.
instr.desc	a list
counts	numeric vector with raw counts expressed per scan
instr.settings	a list
cps	numeric vector with linearized raw counts expressed per second
s.e.response	numeric vector with spectral energy irradiance in W m-2 nm-1 or J d-1 m-2 nm-1
s.q.response	numeric vector with spectral photon irradiance in mol s-1 m-2 nm-1 or mol d-1 m-2 nm-1
Tfr	numeric vector with spectral transmittance as fraction of one
Tpc	numeric vector with spectral transmittance as percent values
Afr	numeric vector of absorptance as fraction of one
A	numeric vector of absorbance values (log10 based a.u.)
Tfr.type, Afr.type	character string indicating whether transmittance and absorptance values are "total" or "internal" values
Rfr	numeric vector with spectral reflectance as fraction of one
Rpc	numeric vector with spectral reflectance as percent values
Rfr.type	character A string, either "total" or "specular".
x, y, z	numeric colour coordinates

Value

A object of class generic_spct or a class derived from it, depending on the function used. In other words an object of a class with the same name as the constructor function.

Note

The functions can be used to add only one spectral quantity to a spectral object. Some of the functions have different arguments, for the same quantity expressed in different units. An actual parameter can be supplied to only one of these formal parameters in a given call to any of these functions.

"internal" transmittance is defined as the transmittance of the material body itself, while "total" transmittance includes the effects of surface reflectance on the amount of light transmitted.

See Also

Other constructors of spectral objects: `as.calibration_spct`, `as.chroma_spct`, `as.cps_spct`, `as.filter_spct`, `as.generic_spct`, `as.object_spct`, `as.raw_spct`, `as.reflector_spct`, `as.response_spct`, `as.source_spct`

---

spct_classes *Function that returns a vector containing the names of spectra classes.*

Description

Function that returns a vector containing the names of spectra classes.

Usage

```
spct_classes()
```

Value

A `character` vector of class names.

Examples

```
spct_classes()
```

split2mspct *Convert a 'wide' or untidy data frame into a collection of spectra*

Description

Convert a data frame object into a "multi spectrum" object by constructing a an object of a multi-spct class, converting numeric columns other than wavelength into individual spct objects.

Usage

```
split2mspct(x, member.class = NULL, spct.data.var = NULL,
  w.length.var = "w.length", idx.var = NULL, ncol = 1,
  byrow = FALSE, ...)

split2source_mspct(x, spct.data.var = "s.e.irrad",
  w.length.var = "w.length", idx.var = NULL, ncol = 1,
  byrow = FALSE, ...)

split2response_mspct(x, spct.data.var = "s.e.response",
  w.length.var = "w.length", idx.var = NULL, ncol = 1,
  byrow = FALSE, ...)

split2filter_mspct(x, spct.data.var = "Tfr", w.length.var = "w.length",
  idx.var = NULL, ncol = 1, byrow = FALSE, ...)

split2reflector_mspct(x, spct.data.var = "Rfr",
  w.length.var = "w.length", idx.var = NULL, ncol = 1,
  byrow = FALSE, ...)

split2cps_mspct(x, spct.data.var = "cps", w.length.var = "w.length",
  idx.var = NULL, ncol = 1, byrow = FALSE, ...)

split2raw_mspct(x, spct.data.var = "count", w.length.var = "w.length",
  idx.var = NULL, ncol = 1, byrow = FALSE, ...)

split2calibration_mspct(x, spct.data.var = "irrad.mult",
  w.length.var = "w.length", idx.var = NULL, ncol = 1,
  byrow = FALSE, ...)
```

Arguments

x	data frame
member.class	character Class of the collection members
spct.data.var	character Name of the spectral data argument in the object constructor for member.class
w.length.var	character Name of column containing wavelength data in nanometres

idx.var	character Name of column containing data to be copied unchanged to each spct object
ncol	integer Number of 'virtual' columns in data
byrow	logical If ncol > 1 how to read in the data
...	additional named arguments passed to the member constructor function.

See Also

Other Coercion methods for collections of spectra: as.calibration_mspct, as.chroma_mspct, as.cps_mspct, as.filter_mspct, as.generic_mspct, as.object_mspct, as.raw_mspct, as.reflector_mspct, as.response_mspct, as.source_mspct, subset2mspct

---

split_bands *List-of-wavebands constructor*

Description

Build a list of unweighted "waveband" objects that can be used as input when calculating irradiances.

Usage

```
split_bands(x, list.names = NULL, short.names = is.null(list.names),
  length.out = NULL)
```

Arguments

x	a numeric vector of wavelengths to split at (nm), or a range of wavelengths or a generic_spct or a waveband.
list.names	character vector with names for the component wavebands in the returned list (in order of increasing wavelength)
short.names	logical indicating whether to use short or long names for wavebands
length.out	numeric giving the number of regions to split the range into (ignored if w.length is not numeric).

Value

an un-named list of waveband objects

Note

list.names is used to assign names to the elements of the list, while the waveband objects themselves always retain their wb.label and wb.name as generated during their creation.

See Also

Other waveband constructors: waveband

Examples

```
split_bands(c(400,500,600))
split_bands(list(c(400,500),c(550,650)))
split_bands(list(A=c(400,500),B=c(550,650)))
split_bands(c(400,500,600), short.names=FALSE)
split_bands(c(400,500,600), list.names=c("a","b"))
split_bands(c(400,700), length.out=6)
split_bands(400:700, length.out=3)
split_bands(sun.spct, length.out=10)
split_bands(waveband(c(400,700)), length.out=5)
```

---

split_energy_irradiance

Energy irradiance for split spectrum regions

Description

This function returns the energy irradiance for a series of contiguous wavebands from a radiation-source spectrum. The returned values can be either absolute or relative to their sum.

Usage

```
split_energy_irradiance(w.length, s.irrad,
  cut.w.length = range(w.length), unit.in = "energy",
  scale = "absolute", check.spectrum = TRUE, use.cached.mult = FALSE,
  use.hinges = getOption("photobiology.use.hinges", default = NULL))
```

Arguments

w.length	numeric vector of wavelengths (nm).
s.irrad	numeric vector of spectral (energy or photon) irradiance values (W m-2 nm-1) or (mol s-1 m-2 nm-1).
cut.w.length	numeric vector of wavelengths (nm).
unit.in	character string with allowed values "energy", and "photon", or its alias "quantum".
scale	character string indicating the scale used for the returned values ("absolute", "relative", "percent").
check.spectrum	logical indicating whether to sanity check input data, default is TRUE.
use.cached.mult	logical Flag indicating whether multiplier values should be cached between calls.
use.hinges	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.

Value

a numeric vector of irradiances with no change in scale factor: [W m-2 nm-1] -> [W m-2] or [mol s-1 m-2] -> [W m-2] or relative values (fraction of one) if scale = "relative" or scale = "percent".

Note

The last three parameters control speed optimizations. The defaults should be suitable in most cases. If you set `check.spectrum=FALSE` then you should call `check_spectrum` at least once for your spectrum before using any of the other functions. If you will use repeatedly the same SWFs on many spectra measured at exactly the same wavelengths you may obtain some speed up by setting `use.cached.mult=TRUE`. However, be aware that you are responsible for ensuring that the wavelengths are the same in each call, as the only test done is for the length of the `w.length` vector.

See Also

Other low-level functions operating on numeric vectors.: `as_energy`, `as_quantum_mol`, `calc_multipliers`, `div_spectra`, `energy_irradiance`, `energy_ratio`, `insert_hinges`, `integrate_xy`, `interpolate_spectrum`, `irradiance`, `l_insert_hinges`, `oper_spectra`, `photon_irradiance`, `photon_ratio`, `photons_energy_ratio`, `prod_spectra`, `s_e_irrad2rgb`, `split_photon_irradiance`, `subt_spectra`, `sum_spectra`, `trim_tails`, `v_insert_hinges`, `v_replace_hinges`

Examples

```
with(sun.data,
     split_energy_irradiance(w.length, s.e.irrad,
                             cut.w.length = c(300, 400, 500, 600, 700)))
```

---

split_irradiance *Energy or photon irradiance for split spectrum regions*

Description

This function returns the energy or photon irradiance for a series of contiguous wavebands from a radiation spectrum. The returned values can be either absolute or relative to their sum.

Usage

```
split_irradiance(w.length, s.irrad, cut.w.length = range(w.length),
  unit.out = getOption("photobiology.base.unit", default = "energy"),
  unit.in = "energy", scale = "absolute", check.spectrum = TRUE,
  use.cached.mult = FALSE,
  use.hinges = getOption("photobiology.use.hinges", default = NULL))
```

Arguments

w.length	numeric Vector of wavelengths (nm).
s.irrad	numeric vector of spectral (energy or photon) irradiances (W m-2 nm-1) or (mol s-1 m-2 nm-1).
cut.w.length	numeric Vector of wavelengths (nm).
unit.out	character Allowed values "energy", and "photon", or its alias "quantum".
unit.in	character Allowed values "energy", and "photon", or its alias "quantum".
scale	a character A string indicating the scale used for the returned values ("absolute", "relative", "percent").
check.spectrum	logical Flag indicating whether to sanity check input data, default is TRUE.
use.cached.mult	logical Flag indicating whether multiplier values should be cached between calls.
use.hinges	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.

Value

A numeric vector of irradiances with no change in scale factor: [W m-2 nm-1] -> [mol s-1 m-2] or [mol s-1 m-2 nm-1] -> [mol s-1 m-2] or relative values (as fraction of one if `scale == "relative"` or percentages if `scale == "percent"`.

Note

The last three parameters control speed optimizations. The defaults should be suitable in most cases. If you set `check.spectrum=FALSE` then you should call `check_spectrum` at least once for your spectrum before using any of the other functions. If you will use repeatedly the same SWFs on many spectra measured at exactly the same wavelengths you may obtain some speed up by setting `use.cached.mult=TRUE`. However, be aware that you are responsible for ensuring that the wavelengths are the same in each call, as the only test done is for the length of the `w.length` vector.

Examples

```
with(sun.data,
     split_irradiance(w.length, s.e.irrad,
                      cut.w.length = c(300, 400, 500, 600, 700),
                      unit.out = "photon"))
```

split_photon_irradiance *Photon irradiance for split spectrum regions*

Description

This function returns the photon irradiance for a series of contiguous wavebands from a radiation spectrum. The returned values can be either absolute or relative to their sum.

Usage

```
split_photon_irradiance(w.length, s.irrad,
  cut.w.length = range(w.length), unit.in = "energy",
  scale = "absolute", check.spectrum = TRUE, use.cached.mult = FALSE,
  use.hinges = getOption("photobiology.use.hinges", default = NULL))
```

Arguments

w.length	numeric vector of wavelengths (nm).
s.irrad	numeric vector of spectral (energy or photon) irradiance values (W m-2 nm-1).
cut.w.length	numeric vector of wavelengths (nm).
unit.in	character Allowed values "energy", and "photon", or its alias "quantum".
scale	a character A string indicating the scale used for the returned values ("absolute", "relative", "percent").
check.spectrum	logical Flag indicating whether to sanity check input data, default is TRUE.
use.cached.mult	logical Flag indicating whether multiplier values should be cached between calls.
use.hinges	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.

Value

a numeric vector of photon irradiances with no change in scale factor: [W m-2 nm-1] -> [mol s-1 m-2], [mol s-1 m-2 nm-1] -> [mol s-1 m-2] or relative values (fraction of one based on photon units) if scale = "relative" or scale = "percent".

Note

The last three parameters control speed optimizations. The defaults should be suitable in most cases. If you set check.spectrum=FALSE then you should call check_spectrum at least once for your spectrum before using any of the other functions. If you will use repeatedly the same SWFs on many spectra measured at exactly the same wavelengths you may obtain some speed up by setting use.cached.mult=TRUE. However, be aware that you are responsible for ensuring that the wavelengths are the same in each call, as the only test done is for the length of the w.length vector.

See Also

Other low-level functions operating on numeric vectors.: as_energy, as_quantum_mol, calc_multipliers, div_spectra, energy_irradiance, energy_ratio, insert_hinges, integrate_xy, interpolate_spectrum, irradiance, l_insert_hinges, oper_spectra, photon_irradiance, photon_ratio, photons_energy_ratio, prod_spectra, s_e_irrad2rgb, split_energy_irradiance, subt_spectra, sum_spectra, trim_tails, v_insert_hinges, v_replace_hinges

Examples

```
with(sun.data,
     split_photon_irradiance(w.length, s.e.irrad,
             cut.w.length = c(300, 400, 500, 600, 700)))
with(sun.data,
     split_photon_irradiance(w.length, s.e.irrad))
```

---

spread *Expanse*

Description

A function that returns the expanse (max(x) - min(x)) for R objects.

Usage

```
spread(x, ...)

wl_expanse(x, ...)

expanse(x, ...)

## Default S3 method:
expanse(x, ...)

## S3 method for class 'numeric'
expanse(x, ...)

## S3 method for class 'waveband'
expanse(x, ...)

## S3 method for class 'generic_spct'
expanse(x, ...)

## S3 method for class 'generic_mspct'
expanse(x, ..., idx = "spct.idx")
```

Arguments

x	an R object
...	not used in current version
idx	character Name of the column with the names of the members of the collection of spectra.

Value

A numeric value equal to max(x) - min(x). In the case of spectral objects wavelength difference in nm. For any other R object, according to available definitions of min and max.

Methods (by class)

- default: Default method for generic function
- numeric: Method for "numeric"
- waveband: Method for "waveband"
- generic_spct: Method for "generic_spct"
- generic_mspct: Method for "generic_mspct" objects.

Examples

```
expanse(10:20)
expanse(sun.spct)
wl_expanse(sun.spct)

expanse(sun.spct)
```

Subset *Subsetting spectra*

Description

Return subsets of spectra stored in class generic_spct or derived from it.

Usage

```
## S3 method for class 'generic_spct'
subset(x, subset, select, drop = FALSE, ...)
```

Arguments

x	object to be subsetted.
subset	logical expression indicating elements or rows to keep: missing values are taken as false.
select	expression, indicating columns to select from a spectrum.
drop	passed on to [indexing operator.
...	further arguments to be passed to or from other methods.

Value

An object similar to x containing just the selected rows and columns. Depending on the columns remaining after subsetting the class of the object will be simplified to the most derived parent class.

Note

This method is copied from base::subset.data.frame() but ensures that all metadata stored in attributes of spectral objects are copied to the returned value.

Examples

```
subset(sun.spct, w.length > 400)
```

---

subset2mspct *Convert 'long' or tidy spectral data into a collection of spectra*

Description

Convert a data frame object or spectral object into a collection of spectra object of the corresponding class. For data frames converting numeric columns other than wavelength into individual spct objects.

Usage

```
subset2mspct(x, member.class = NULL, idx.var = attr(x, "idfactor"),
  drop.idx = TRUE, ncol = 1, byrow = FALSE, ...)
```

Arguments

x	a generic_spct object or a derived class, or a data frame
member.class	character string
idx.var	character Name of column containing data to be copied unchanged to each spct object
drop.idx	logical Flag indicating whether to drop or keep idx.var in the collection members.
ncol	integer Number of 'virtual' columns in data
byrow	logical If ncol > 1 how to read in the data
...	additional named arguments passed to the member constructor function.

Value

A collection of spectral objects, each with attributes set if x is a spectral object in long form with metadata attributes. If this object was created by row binding with 'photobiology' 0.9.14 or later then all metadata for each individual spectrum will be preserved, except for comments which are merged.

Note

A non-null value for `member.class` is mandatory only when `x` is a data frame.

See Also

Other Coercion methods for collections of spectra: `as.calibration_mspct`, `as.chroma_mspct`, `as.cps_mspct`, `as.filter_mspct`, `as.generic_mspct`, `as.object_mspct`, `as.raw_mspct`, `as.reflector_mspct`, `as.response_mspct`, `as.source_mspct`, `split2mspct`

`subt_spectra` *Subtract two spectra*

Description

The wavelength vectors of the two spectra are merged, and the missing spectral values are calculated by interpolation. After this, the two spectral values at each wavelength are added. This is 'parallel' operation between two spectra.

Usage

```
subt_spectra(w.length1, w.length2 = NULL, s.irrad1, s.irrad2,
  trim = "union", na.rm = FALSE)
```

Arguments

`w.length1`	numeric vector of wavelength (nm).
`w.length2`	numeric vector of wavelength (nm).
`s.irrad1`	a numeric vector of spectral values.
`s.irrad2`	a numeric vector of spectral values.
`trim`	a character string with value "union" or "intersection".
`na.rm`	a logical value, if TRUE, not the default, NAs in the input are replaced with zeros.

Details

If trim=="union" spectral values are calculated for the whole range of wavelengths covered by at least one of the input spectra, and missing values are set in each input spectrum to zero before addition. If trim=="intersection" then the range of wavelengths covered by both input spectra is returned, and the non-overlapping regions discarded. If w.length2==NULL, it is assumed that both spectra are measured at the same wavelengths, and a simple addition is used, ensuring fast calculation.

Value

a data frame with two numeric variables

w.length	A numeric vector with the wavelengths (nm) obtained by "fusing" w.length1 and w.length2. w.length contains all the unique vales, sorted in ascending order.
s.irrad	A numeric vector with the sum of the two spectral values at each wavelength.

See Also

Other low-level functions operating on numeric vectors.: as_energy, as_quantum_mol, calc_multipliers, div_spectra, energy_irradiance, energy_ratio, insert_hinges, integrate_xy, interpolate_spectrum, irradiance, l_insert_hinges, oper_spectra, photon_irradiance, photon_ratio, photons_energy_ratio, prod_spectra, s_e_irrad2rgb, split_energy_irradiance, split_photon_irradiance, sum_spectra, trim_tails, v_insert_hinges, v_replace_hinges

Examples

```
head(sun.data)
zero.data <- with(sun.data, subt_spectra(w.length, w.length, s.e.irrad, s.e.irrad))
head(zero.data)
tail(zero.data)
```

---

summary *Summary of a spectral object*

Description

Methods of generic function summary for objects of spectral classes.

Usage

```
## S3 method for class 'generic_spct'
summary(object, maxsum = 7, digits = max(3,
  getOption("digits") - 3), ...)
```

Arguments

object	An object of one of the spectral classes for which a summary is desired
maxsum	integer Indicates how many levels should be shown for factors.
digits	integer Used for number formatting with format().
...	additional arguments affecting the summary produced, ignored in current version

Value

A summary object matching the class of object.

Examples

```
summary(sun.spct)
```

---

summary_spct_classes *Function that returns a vector containing the names of spectral summary classes.*

Description

Function that returns a vector containing the names of spectral summary classes.

Usage

```
summary_spct_classes()
```

Value

A `character` vector of class names.

---

sum_spectra *Add two spectra*

Description

Merge wavelength vectors of two spectra are merged, and the missing spectral values are calculated by interpolation. After this, the two spectral values at each wavelength are added. This is a 'parallel' operation between two spectra.

Usage

```
sum_spectra(w.length1, w.length2 = NULL, s.irrad1, s.irrad2,
  trim = "union", na.rm = FALSE)
```

Arguments

`w.length1`	numeric vector of wavelength (nm).
`w.length2`	numeric vector of wavelength (nm).
`s.irrad1`	a numeric vector of spectral values.
`s.irrad2`	a numeric vector of spectral values.
`trim`	a character string with value "union" or "intersection".
`na.rm`	a logical value, if TRUE, not the default, NAs in the input are replaced with zeros.

Details

If trim=="union" spectral values are calculated for the whole range of wavelengths covered by at least one of the input spectra, and missing values are set in each input spectrum to zero before addition. If trim=="intersection" then the range of wavelengths covered by both input spectra is returned, and the non-overlapping regions discarded. If w.length2==NULL, it is assumed that both spectra are measured at the same wavelengths, and a simple addition is used, ensuring fast calculation.

Value

a dataframe with two numeric variables

`w.length`	A numeric vector with the wavelengths (nm) obtained by "fusing" w.length1 and w.length2. w.length contains all the unique vales, sorted in ascending order.
`s.irrad`	A numeric vector with the sum of the two spectral values at each wavelength.

See Also

Other low-level functions operating on numeric vectors.: as_energy, as_quantum_mol, calc_multipliers, div_spectra, energy_irradiance, energy_ratio, insert_hinges, integrate_xy, interpolate_spectrum, irradiance, l_insert_hinges, oper_spectra, photon_irradiance, photon_ratio, photons_energy_ratio, prod_spectra, s_e_irrad2rgb, split_energy_irradiance, split_photon_irradiance, subt_spectra, trim_tails, v_insert_hinges, v_replace_hinges

Examples

```
head(sun.data)
twice.sun.data <- with(sun.data, sum_spectra(w.length, w.length, s.e.irrad, s.e.irrad))
head(twice.sun.data)
tail(twice.sun.data)
```

---

sun.daily.data *Daily solar spectral irradiance (simulated)*

Description

A dataset containing the wavelengths at a 1 nm interval and the corresponding spectral (energy) irradiance. Values simulated for 2 June 2012, at Helsinki, under clear sky conditions. The variables are as follows:

Usage

```
sun.daily.data
```

Format

A `data.frame` object with 511 rows and 3 variables

Details

- w.length (nm), range 290 to 800 nm.
- s.e.irrad (J d-1 m-2 nm-1)
- s.q.irrad (mol d-1 m-2 nm-1)

Author(s)

Anders K. Lindfors (data)

References

Lindfors, A.; Heikkilä, A.; Kaurola, J.; Koskela, T. & Lakkala, K. (2009) Reconstruction of Solar Spectral Surface UV Irradiances Using Radiative Transfer Simulations. Photochemistry and Photobiology, 85: 1233-1239

See Also

Other Spectral data examples: `A.illuminant.spct`, `D65.illuminant.spct`, `Ler_leaf.spct`, `Ler_leaf_rflt.spct`, `Ler_leaf_trns.spct`, `Ler_leaf_trns_i.spct`, `black_body.spct`, `ccd.spct`, `clear.spct`, `clear_body.spct`, `filter_cps.mspct`, `green_leaf.spct`, `opaque.spct`, `photodiode.spct`, `polyester.spct`, `sun.daily.spct`, `sun.data`, `sun.spct`, `white_body.spct`, `white_led.cps_spct`, `white_led.raw_spct`, `white_led.source_spct`, `yellow_gel.spct`

Examples

```
sun.daily.spct
```

---

sun.daily.spct *Daily solar spectral irradiance (simulated)*

Description

A dataset containing the wavelengths at a 1 nm interval and the corresponding spectral (energy) irradiance. Values simulated for 2 June 2012, at Helsinki, under clear sky conditions. The variables are as follows:

Usage

```
sun.daily.spct
```

Format

A `source_spct` object with 511 rows and 3 variables

Details

- w.length (nm), range 290 to 800 nm.
- s.e.irrad (J d-1 m-2 nm-1)
- s.q.irrad (mol d-1 m-2 nm-1)

Note

The simulations are based on libRadTran using hourly mean global radiation measurements to estimate cloud cover. The simulations were for each hour and the results integrated for the whole day.

Author(s)

Anders K. Lindfors (data)

References

Lindfors, A.; Heikkilä, A.; Kaurola, J.; Koskela, T. & Lakkala, K. (2009) Reconstruction of Solar Spectral Surface UV Irradiances Using Radiative Transfer Simulations. Photochemistry and Photobiology, 85: 1233-1239

See Also

Other Spectral data examples: `A.illuminant.spct`, `D65.illuminant.spct`, `Ler_leaf.spct`, `Ler_leaf_rflt.spct`, `Ler_leaf_trns.spct`, `Ler_leaf_trns_i.spct`, `black_body.spct`, `ccd.spct`, `clear.spct`, `clear_body.spct`, `filter_cps.mspct`, `green_leaf.spct`, `opaque.spct`, `photodiode.spct`, `polyester.spct`, `sun.daily.data`, `sun.data`, `sun.spct`, `white_body.spct`, `white_led.cps_spct`, `white_led.raw_spct`, `white_led.source_spct`, `yellow_gel.spct`

Examples

```
sun.daily.spct
```

---

sun.data *Solar spectral irradiance (simulated)*

Description

A dataset containing the wavelengths at a 1 nm interval and the corresponding spectral (energy) irradiance and spectral photon irradiance. Values simulated for 22 June 2010, near midday, at Helsinki, under partly cloudy conditions. The variables are as follows:

Usage

```
sun.data
```

Format

A `data.frame` object with 508 rows and 3 variables

Details

- w.length (nm), range 293 to 800 nm.
- s.e.irrad (W m-2 nm-1)
- s.q.irrad (mol m-2 nm-1)

Author(s)

Anders K. Lindfors (data)

References

Lindfors, A.; Heikkilä, A.; Kaurola, J.; Koskela, T. & Lakkala, K. (2009) Reconstruction of Solar Spectral Surface UV Irradiances Using Radiative Transfer Simulations. Photochemistry and Photobiology, 85: 1233-1239

See Also

Other Spectral data examples: `A.illuminant.spct`, `D65.illuminant.spct`, `Ler_leaf.spct`, `Ler_leaf_rflt.spct`, `Ler_leaf_trns.spct`, `Ler_leaf_trns_i.spct`, `black_body.spct`, `ccd.spct`, `clear.spct`, `clear_body.spct`, `filter_cps.mspct`, `green_leaf.spct`, `opaque.spct`, `photodiode.spct`, `polyester.spct`, `sun.daily.data`, `sun.daily.spct`, `sun.spct`, `white_body.spct`, `white_led.cps_spct`, `white_led.raw_spct`, `white_led.source_spct`, `yellow_gel.spct`

Examples

```
sun.data
```

---

sun.spct *Solar spectral irradiance (simulated)*

Description

A dataset containing the wavelengths at a 1 nm interval and the corresponding spectral (energy) irradiance and spectral photon irradiance. Values simulated for 22 June 2010, near midday, at Helsinki, under partly cloudy conditions. The variables are as follows:

Usage

```
sun.spct
```

Format

A `source_spct` object with 508 rows and 3 variables

Details

- w.length (nm), range 293 to 800 nm.
- s.e.irrad (W m-2 nm-1)
- s.q.irrad (mol m-2 nm-1)

Author(s)

Anders K. Lindfors (data)

References

Lindfors, A.; Heikkilä, A.; Kaurola, J.; Koskela, T. & Lakkala, K. (2009) Reconstruction of Solar Spectral Surface UV Irradiances Using Radiative Transfer Simulations. Photochemistry and Photobiology, 85: 1233-1239

See Also

Other Spectral data examples: A.illuminant.spct, D65.illuminant.spct, Ler_leaf.spct, Ler_leaf_rflt.spct, Ler_leaf_trns.spct, Ler_leaf_trns_i.spct, black_body.spct, ccd.spct, clear.spct, clear_body.spct, filter_cps.mspct, green_leaf.spct, opaque.spct, photodiode.spct, polyester.spct, sun.daily.data, sun.daily.spct, sun.data, white_body.spct, white_led.cps_spct, white_led.raw_spct, white_led.source_spct, yellow_gel.spct

Examples

```
sun.spct
```

---

sun_angles *Solar angles*

Description

This function returns the solar angles at a given time and location.

Usage

```
sun_angles(time = lubridate::now(tzone = "UTC"),
  tz = lubridate::tz(time), geocode = tibble::tibble(lon = 0, lat =
  51.5, address = "Greenwich"), use.refraction = FALSE)

sun_angles_fast(time, tz, geocode, use.refraction)

sun_elevation(time = lubridate::now(), tz = lubridate::tz(time),
  geocode = tibble::tibble(lon = 0, lat = 51.5, address = "Greenwich"),
  use.refraction = FALSE)
```

```
sun_zenith_angle(time = lubridate::now(), tz = lubridate::tz(time),
  geocode = tibble::tibble(lon = 0, lat = 51.5, address = "Greenwich"),
  use.refraction = FALSE)

sun_azimuth(time = lubridate::now(), tz = lubridate::tz(time),
  geocode = tibble::tibble(lon = 0, lat = 51.5, address = "Greenwich"),
  use.refraction = FALSE)
```

Arguments

time	A "vector" of POSIXct Time, with any valid time zone (TZ) is allowed, default is current time.
tz	character string indicating time zone to be used in output.
geocode	data frame with variables lon and lat as numeric values (degrees), nrow > 1, allowed.
use.refraction	logical Flag indicating whether to correct for fraction in the atmosphere.

Details

This function is an implementation of Meeus equations as used in NOAAs on-line web calculator, which are precise and valid for a very broad range of dates (years -1000 to 3000 at least). The apparent solar elevations near sunrise and sunset are affected by refraction in the atmosphere, which does in turn depend on weather conditions. The effect of refraction on the apparent position of the sun is only an estimate based on "typical" conditions for the atmosphere. The computation is not defined for latitudes 90 and -90 degrees, i.e. exactly at the poles.

In the current implementation functions `sun_azimuth`, `sun_elevation`, and `sun_zenith_angle` are wrappers on `sun_angles`, so if more than one angle is needed it is preferable to directly call `sun_angles` as it will be faster.

Value

A data.frame with variables time (in same TZ as input), TZ, solartime, longitude, latitude, address, azimuth, and elevation. If a data frame with multiple rows is passed to `geocode` and a vector of times longer than one is passed to `time`, sun position for all combinations of locations and times are returned are returned by `sun_angles`. In contrast, convenience functions returning a vector.

Note

There exists a different R implementation of the same algorithms called "AstroCalcPureR" available as function `astrocalc4r` in package 'fishmethods'. Although the equations used are almost all the same, the function signatures and which values are returned differ. In particular, the present implementation splits the calculation into two separate functions, one returning angles at given instants in time, and a separate one returning the timing of events for given dates.

References

The primary source for the algorithm used is the book: Meeus, J. (1998) Astronomical Algorithms, 2 ed., Willmann-Bell, Richmond, VA, USA. ISBN 978-0943396613.

A different implementation is available at https://www.nefsc.noaa.gov/AstroCalc4R/ and in R paclage astrocalc4r. In 'fishmethods' (= 1.11-0) there is a bug in function astrocalc4r() that affects sunrise and sunset times.

An interactive web page using the same algorithms is available at https://www.esrl.noaa.gov/gmd/grad/solcalc/. There are small differences in the returned times compared to our function that seem to be related to the estimation of atmospheric refraction (about 0.1 degrees).

See Also

Other astronomy related functions: day_night, format.solar_time, is.solar_time, print.solar_time, solar_time

Examples

```
library(lubridate)
sun_angles()
sun_azimuth()
sun_elevation()
sun_zenith_angle()
sun_angles(ymd_hms("2014-09-23 12:00:00"))
sun_angles(ymd_hms("2014-09-23 12:00:00"),
           geocode = data.frame(lat=60, lon=0))
sun_angles(ymd_hms("2014-09-23 12:00:00") + minutes((0:6) * 10))
```

---

s_e_irrad2rgb *Spectral irradiance to rgb color conversion*

Description

Calculates rgb values from spectra based on human color matching functions (CMF) or chromaticity coordinates (CC). A CMF takes into account luminous sensitivity, while a CC only the color hue. This function, in contrast to that in package pavo does not normalize the values to equal luminosity, so using a CMF as input gives the expected result. Another difference is that it allows the user to choose the chromaticity data to be used. The data used by default is different, and it corresponds to the whole range of CIE standard, rather than the reduced range 400 nm to 700 nm. The wavelength limits are not hard coded, so the function could be used to simulate vision in other organisms as long as pseudo CMF or CC data are available for the simulation.

Usage

```
s_e_irrad2rgb(w.length, s.e.irrad, sens = photobiology::ciexyzCMF2.spct,
  color.name = NULL, check = TRUE)
```

Arguments

w.length	numeric vector of wavelengths (nm).
s.e.irrad	numeric vector of spectral irradiance values.
sens	a chroma_spct object with variables w.length, x, y, and z, giving the CC or CMF definition (default is the proposed human CMF according to CIE 2006.).
color.name	character string for naming the rgb color definition.
check	logical indicating whether to check or not spectral data.

Value

A color defined using rgb. The numeric values of the RGB components can be obtained using function col2rgb.

Note

Very heavily modified from Chad Eliason's <cme16@zips.uakron.edu> spec2rgb function in package Pavo.

References

CIE(1932). Commission Internationale de l'Eclairage Proceedings, 1931. Cambridge: Cambridge University Press.

Color matching functions obtained from Colour and Vision Research Laboratory online data repository at http://www.cvrl.org/.

See Also

Other low-level functions operating on numeric vectors.: as_energy, as_quantum_mol, calc_multipliers, div_spectra, energy_irradiance, energy_ratio, insert_hinges, integrate_xy, interpolate_spectrum, irradiance, l_insert_hinges, oper_spectra, photon_irradiance, photon_ratio, photons_energy_ratio, prod_spectra, split_energy_irradiance, split_photon_irradiance, subt_spectra, sum_spectra, trim_tails, v_insert_hinges, v_replace_hinges

Examples

```
my.color <-
    with(sun.data, s_e_irrad2rgb(w.length, s.e.irrad, color.name="sunWhite"))
col2rgb(my.color)
```

s_mean *Mean from collection of spectra*

Description

A method to compute the mean of values across members of a collections of spectra. Computes the mean at each wavelength across all the spectra in the collection returning a spectral object.

Usage

```
s_mean(x, trim, na.rm, ...)

## Default S3 method:
s_mean(x, trim = 0, na.rm = FALSE, ...)

## S3 method for class 'source_mspct'
s_mean(x, trim = 0, na.rm = FALSE, ...)

## S3 method for class 'response_mspct'
s_mean(x, trim = 0, na.rm = FALSE, ...)

## S3 method for class 'filter_mspct'
s_mean(x, trim = 0, na.rm = FALSE, ...)

## S3 method for class 'reflector_mspct'
s_mean(x, trim = 0, na.rm = FALSE, ...)

## S3 method for class 'calibration_mspct'
s_mean(x, trim = 0, na.rm = FALSE, ...)
```

Arguments

x	An R object. Currently this package defines methods for collections of spectral objects.
trim	numeric. The fraction (0 to 0.5) of observations to be trimmed from each end of x before the mean is computed. Values of trim outside that range are taken as the nearest endpoint.
na.rm	logical. A value indicating whether NA values should be stripped before the computation proceeds.
...	Further arguments passed to or from other methods.

Value

If x is a collection spectral of objects, such as a "filter_mspct" object, the returned object is of same class as the members of the collection, such as "filter_spct", containing the mean spectrum.

Methods (by class)

- default:
- source_mspct:
- response_mspct:
- filter_mspct:
- reflector_mspct:
- calibration_mspct:

Note

Trimming of extreme values and omission of NAs is done separately at each wavelength. Interpolation is not applied, so all spectra in x must share the same set of wavelengths.

See Also

See mean for the mean() method used for the computations.

---

s_mean_se *Mean and standard error from collection of spectra*

Description

A method to compute the mean of values across members of a collections of spectra. Computes the mean at each wavelength across all the spectra in the collection returning a spectral object.

Usage

```
s_mean_se(x, na.rm, mult, ...)

## Default S3 method:
s_mean_se(x, na.rm = FALSE, mult = 1, ...)

## S3 method for class 'filter_mspct'
s_mean_se(x, na.rm = FALSE, mult = 1, ...)

## S3 method for class 'source_mspct'
s_mean_se(x, na.rm = FALSE, mult = 1, ...)

## S3 method for class 'response_mspct'
s_mean_se(x, na.rm = FALSE, mult = 1, ...)

## S3 method for class 'reflector_mspct'
s_mean_se(x, na.rm = FALSE, mult = 1, ...)

## S3 method for class 'calibration_mspct'
s_mean_se(x, na.rm = FALSE, mult = 1, ...)
```

Arguments

x	An R object. Currently this package defines methods for collections of spectral objects.
na.rm	logical. A value indicating whether NA values should be stripped before the computation proceeds.
mult	numeric number of multiples of standard error
...	Further arguments passed to or from other methods.

Value

If x is a collection spectral of objects, such as a "filter_mspct" object, the returned object is of same class as the members of the collection, such as "filter_spct", containing the mean spectrum.

Methods (by class)

- default:
- filter_mspct:
- source_mspct:
- response_mspct:
- reflector_mspct:
- calibration_mspct:

Note

Trimming of extreme values and omission of NAs is done separately at each wavelength. Interpolation is not applied, so all spectra in x must share the same set of wavelengths.

See Also

See mean for the mean() method used for the computations.

---

s_median *Median of a collection of spectra*

Description

A method to compute the median of values across members of a collections of spectra. Computes the median at each wavelength across all the spectra in the collection returning a spectral object.

Usage

```
s_median(x, na.rm, ...)

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

## S3 method for class 'source_mspct'
s_median(x, na.rm = FALSE, ...)

## S3 method for class 'response_mspct'
s_median(x, na.rm = FALSE, ...)

## S3 method for class 'filter_mspct'
s_median(x, na.rm = FALSE, ...)

## S3 method for class 'reflector_mspct'
s_median(x, na.rm = FALSE, ...)

## S3 method for class 'calibration_mspct'
s_median(x, na.rm = FALSE, ...)
```

Arguments

x	An R object. Currently this package defines methods for collections of spectral objects.
na.rm	logical. A value indicating whether NA values should be stripped before the computation proceeds.
...	Further arguments passed to or from other methods.

Value

If x is a collection spectral of objects, such as a "filter_mspct" object, the returned object is of same class as the members of the collection, such as "filter_spct", containing the median spectrum.

Methods (by class)

- default:
- source_mspct:
- response_mspct:
- filter_mspct:
- reflector_mspct:
- calibration_mspct:

Note

Omission of NAs is done separately at each wavelength. Interpolation is not applied, so all spectra in x must share the same set of wavelengths.

See Also

See median for the median() method used for the computations.

---

s_prod *Product from collection of spectra*

Description

A method to compute the product of values across members of a collections of spectra. Computes the product at each wavelength across all the spectra in the collection returning a spectral object.

Usage

```
s_prod(x, na.rm, ...)

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

## S3 method for class 'source_mspct'
s_prod(x, na.rm = FALSE, ...)

## S3 method for class 'response_mspct'
s_prod(x, na.rm = FALSE, ...)

## S3 method for class 'filter_mspct'
s_prod(x, na.rm = FALSE, ...)

## S3 method for class 'reflector_mspct'
s_prod(x, na.rm = FALSE, ...)

## S3 method for class 'calibration_mspct'
s_prod(x, na.rm = FALSE, ...)
```

Arguments

x	An R object. Currently this package defines methods for collections of spectral objects.
na.rm	logical. A value indicating whether NA values should be stripped before the computation proceeds.
...	Further arguments passed to or from other methods.

Value

If x is a collection spectral of objects, such as a "filter_mspct" object, the returned object is of same class as the members of the collection, such as "filter_spct", containing the product of the spectra.

Methods (by class)

- default:
- source_mspct:
- response_mspct:
- filter_mspct:
- reflector_mspct:
- calibration_mspct:

Note

Omission of NAs is done separately at each wavelength. Interpolation is not applied, so all spectra in x must share the same set of wavelengths.

A product of spectral irradiance or spectral response is no longer a well defined physical quanttiy, and these product operations return an object of class generic_spct.

See Also

See prod for the prod() method used for the computations.

---

s_range *Range of a collection of spectra*

Description

A method to compute the range of values across members of a collections of spectra. Computes the max and min at each wavelength across all the spectra in the collection returning a spectral object.

Usage

```
s_range(x, na.rm, ...)

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

## S3 method for class 'filter_mspct'
s_range(x, na.rm = FALSE, ...)

## S3 method for class 'source_mspct'
s_range(x, na.rm = FALSE, ...)

## S3 method for class 'response_mspct'
s_range(x, na.rm = FALSE, ...)

## S3 method for class 'reflector_mspct'
s_range(x, na.rm = FALSE, ...)
```

```
## S3 method for class 'calibration_mspct'
s_range(x, na.rm = FALSE, ...)
```

Arguments

x	An R object. Currently this package defines methods for collections of spectral objects.
na.rm	logical. A value indicating whether NA values should be stripped before the computation proceeds.
...	Further arguments passed to or from other methods.

Value

If x is a collection spectral of objects, such as a "filter_mspct" object, the returned object is of same class as the members of the collection, such as "filter_spct", containing the mean spectrum.

Methods (by class)

- default:
- filter_mspct:
- source_mspct:
- response_mspct:
- reflector_mspct:
- calibration_mspct:

Note

Trimming of extreme values and omission of NAs is done separately at each wavelength. Interpolation is not applied, so all spectra in x must share the same set of wavelengths.

See Also

See Extremes details on the min() and max() methods used for the computations.

---

s_sd *Standard Deviation of a collection of spectra*

Description

A method to compute the standard deviation of values across members of a collections of spectra. Computes the standard deviation at each wavelength across all the spectra in the collection returning a spectral object.

Usage

```
s_sd(x, na.rm, ...)

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

## S3 method for class 'filter_mspct'
s_sd(x, na.rm = FALSE, ...)

## S3 method for class 'source_mspct'
s_sd(x, na.rm = FALSE, ...)

## S3 method for class 'response_mspct'
s_sd(x, na.rm = FALSE, ...)

## S3 method for class 'reflector_mspct'
s_sd(x, na.rm = FALSE, ...)
```

Arguments

x	An R object. Currently this package defines methods for collections of spectral objects.
na.rm	logical. A value indicating whether NA values should be stripped before the computation proceeds.
...	Further arguments passed to or from other methods.

Value

If x is a collection spectral of objects, such as a "filter_mspct" object, the returned object is of class "generic_spct", containing the standard deviation among the spectra at each wavelength in a column with name ending in ".sd".

Methods (by class)

- default:
- filter_mspct:
- source_mspct:
- response_mspct:
- reflector_mspct:

Note

Omission of NAs is done separately at each wavelength. Interpolation is not applied, so all spectra in x must share the same set of wavelengths.

See Also

See sd for details about sd() methods for other classes.

s_se *Standard Error of a collection of spectra*

Description

A method to compute the standard error of values across members of a collections of spectra. Computes the standard error at each wavelength across all the spectra in the collection returning a spectral object.

Usage

```
s_se(x, na.rm, ...)

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

## S3 method for class 'source_mspct'
s_se(x, na.rm = FALSE, ...)

## S3 method for class 'response_mspct'
s_se(x, na.rm = FALSE, ...)

## S3 method for class 'filter_mspct'
s_se(x, na.rm = FALSE, ...)

## S3 method for class 'reflector_mspct'
s_se(x, na.rm = FALSE, ...)

## S3 method for class 'calibration_mspct'
s_se(x, na.rm = FALSE, ...)
```

Arguments

x	An R object. Currently this package defines methods for collections of spectral objects.
na.rm	logical. A value indicating whether NA values should be stripped before the computation proceeds.
...	Further arguments passed to or from other methods.

Value

If x is a collection spectral of objects, such as a "filter_mspct" object, the returned object is of class "generic_spct", containing the standard error among the spectra at each wavelength in a column with name ending in ".se".

Methods (by class)

- default:
- source_mspct:
- response_mspct:
- filter_mspct:
- reflector_mspct:
- calibration_mspct:

Note

Omission of NAs is done separately at each wavelength. Interpolation is not applied, so all spectra in x must share the same set of wavelengths.

---

s_sum *Sum from collection of spectra*

Description

A method to compute the sum of values across members of a collections of spectra. Computes the sum at each wavelength across all the spectra in the collection returning a spectral object.

Usage

```
s_sum(x, na.rm, ...)

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

## S3 method for class 'filter_mspct'
s_sum(x, na.rm = FALSE, ...)

## S3 method for class 'source_mspct'
s_sum(x, na.rm = FALSE, ...)

## S3 method for class 'response_mspct'
s_sum(x, na.rm = FALSE, ...)

## S3 method for class 'reflector_mspct'
s_sum(x, na.rm = FALSE, ...)

## S3 method for class 'calibration_mspct'
s_sum(x, na.rm = FALSE, ...)
```

Arguments

x	An R object. Currently this package defines methods for collections of spectral objects.
na.rm	logical. A value indicating whether NA values should be stripped before the computation proceeds.
...	Further arguments passed to or from other methods.

Value

If x is a collection spectral of objects, such as a "filter_mspct" object, the returned object is of same class as the members of the collection, such as "filter_spct", containing the sum of the spectra.

Methods (by class)

- default:
- filter_mspct:
- source_mspct:
- response_mspct:
- reflector_mspct:
- calibration_mspct:

Note

Omission of NAs is done separately at each wavelength. Interpolation is not applied, so all spectra in x must share the same set of wavelengths.

A sum of transmitances or reflectances is no longer a well defined physical quanttiy, and these sum operations return an object of class generic_spct.

See Also

See sum for the sum() method used for the computations.

---

s_var *Variance of a collection of spectra*

Description

A method to compute the variance of values across members of a collections of spectra. Computes the variance at each wavelength across all the spectra in the collection returning a spectral object.

Usage

```
s_var(x, na.rm, ...)

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

## S3 method for class 'filter_mspct'
s_var(x, na.rm = FALSE, ...)

## S3 method for class 'source_mspct'
s_var(x, na.rm = FALSE, ...)

## S3 method for class 'response_mspct'
s_var(x, na.rm = FALSE, ...)

## S3 method for class 'reflector_mspct'
s_var(x, na.rm = FALSE, ...)

## S3 method for class 'calibration_mspct'
s_var(x, na.rm = FALSE, ...)
```

Arguments

x	An R object. Currently this package defines methods for collections of spectral objects.
na.rm	logical. A value indicating whether NA values should be stripped before the computation proceeds.
...	Further arguments passed to or from other methods.

Details

Variance method for collections of spectra. Computes the variance at each wavelength across all the spectra in the collection.

Value

If x is a collection spectral of objects, such as a "filter_mspct" object, the returned object is of class "generic_spct", containing the variance among the spectra at each wavelength in a column with name ending in ".var".

Methods (by class)

- default:
- filter_mspct:
- source_mspct:
- response_mspct:
- reflector_mspct:
- calibration_mspct:

Note

Omission of NAs is done separately at each wavelength. Interpolation is not applied, so all spectra in x must share the same set of wavelengths.

See Also

See cor for details about var(), which is used for the computations.

T2A *Convert transmittance into absorbance.*

Description

Function that converts transmittance (fraction) into absorbance (a.u.).

Usage

```
T2A(x, action, byref, clean, ...)

## Default S3 method:
T2A(x, action = NULL, byref = FALSE, clean = TRUE,
  ...)

## S3 method for class 'filter_spct'
T2A(x, action = "add", byref = FALSE,
  clean = TRUE, ...)

## S3 method for class 'filter_mspct'
T2A(x, action = "add", byref = FALSE,
  clean = TRUE, ..., .parallel = FALSE, .paropts = NULL)
```

Arguments

x	an R object
action	character Allowed values "replace" and "add"
byref	logical indicating if new object will be created by reference or by copy of x
clean	logical replace off-boundary values before conversion
...	not used in current version
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Methods (by class)

- `default`: Default method for generic function
- `filter_spct`: Method for filter spectra
- `filter_mspct`: Method for collections of filter spectra

See Also

Other quantity conversion functions: `A2T`, `T2Afr`, `T2T`, `as_quantum`, `e2qmol_multipliers`, `e2quantum_multipliers`, `e2q`, `q2e`

`T2Afr` *Convert transmittance into absorptance.*

Description

Function that converts transmittance (fraction) into absorptance (fraction). If reflectance (fraction) is available, it allows conversions between internal and total absorptance.

Usage

```
T2Afr(x, action, byref, clean, ...)

## Default S3 method:
T2Afr(x, action = NULL, byref = FALSE,
  clean = FALSE, ...)

## S3 method for class 'filter_spct'
T2Afr(x, action = "add", byref = FALSE,
  clean = FALSE, ...)

## S3 method for class 'object_spct'
T2Afr(x, action = "add", byref = FALSE,
  clean = FALSE, ...)

## S3 method for class 'filter_mspct'
T2Afr(x, action = "add", byref = FALSE,
  clean = FALSE, ..., .parallel = FALSE, .paropts = NULL)

## S3 method for class 'object_mspct'
T2Afr(x, action = "add", byref = FALSE,
  clean = FALSE, ..., .parallel = FALSE, .paropts = NULL)
```

Arguments

x	an R object
action	character Allowed values "replace" and "add"
byref	logical indicating if new object will be created by reference or by copy of x
clean	logical replace off-boundary values before conversion
...	not used in current version
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Methods (by class)

- default: Default method for generic function
- filter_spct: Method for filter spectra
- object_spct: Method for object spectra
- filter_mspct: Method for collections of filter spectra
- object_mspct: Method for collections of object spectra

See Also

Other quantity conversion functions: A2T, T2A, T2T, as_quantum, e2qmol_multipliers, e2quantum_multipliers, e2q, q2e

Examples

```
T2Afr(Ler_leaf.spct)
```

T2T *Convert transmittance type.*

Description

Function that allows conversion between internal and total transmittance.

Usage

```
T2T(x, y, byref, Tfr.type.out, ...)

## Default S3 method:
T2T(x, y, byref = NULL, Tfr.type.out = NULL, ...)

## S3 method for class 'filter_spct'
T2T(x, y, byref = FALSE, Tfr.type.out = "total",
  ...)

## S3 method for class 'object_spct'
T2T(x, y = NULL, byref = FALSE,
  Tfr.type.out = "total", ...)

## S3 method for class 'filter_mspct'
T2T(x, y, byref = FALSE, Tfr.type.out = "total",
  ..., .parallel = FALSE, .paropts = NULL)

## S3 method for class 'object_mspct'
T2T(x, y, byref = FALSE, Tfr.type.out = "total",
  ..., .parallel = FALSE, .paropts = NULL)
```

Arguments

x, y	R objects
byref	logical indicating if new object will be created by reference or by copy of x
Tfr.type.out	character One of "total" or "internal"
...	not used in current version
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Methods (by class)

- `default`: Default method
- `filter_spct`: Method for filter spectra
- `object_spct`: Method for object spectra
- `filter_mspct`: Method for collections of filter spectra
- `object_mspct`: Method for collections of object spectra

See Also

Other quantity conversion functions: A2T, T2Afr, T2A, as_quantum, e2qmol_multipliers, e2quantum_multipliers, e2q, q2e

tag *Tag a spectrum*

Description

Spectra are tagged by adding variables and attributes containing color definitions, labels, and a factor following the wavebands given in `w.band`. This methods are most useful for plotting realistic computed colors from spectral data.

Usage

```
tag(x, ...)

## Default S3 method:
tag(x, ...)

## S3 method for class 'generic_spct'
tag(x, w.band = NULL,
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.hinges = TRUE, short.names = TRUE, byref = FALSE, ...)

## S3 method for class 'generic_mspct'
tag(x, w.band = NULL,
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.hinges = TRUE, short.names = TRUE, byref = FALSE, ...,
  .parallel = FALSE, .paropts = NULL)
```

Arguments

`x`	an R object.
`...`	ignored (possibly used by derived methods).
`w.band`	waveband or list of waveband objects. The waveband(s) determine the region(s) of the spectrum that are tagged
`wb.trim`	logical Flag telling if wavebands crossing spectral data boundaries are trimmed or ignored
`use.hinges`	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.
`short.names`	logical Flag indicating whether to use short or long names for wavebands
`byref`	logical Flag indicating if new object will be created *by reference* or *by copy* of `x`
`.parallel`	if TRUE, apply function in parallel, using parallel backend provided by foreach
`.paropts`	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Value

A copy of x expanded with additional columns with color-related information.

Methods (by class)

- default: Default method for generic
- generic_spct: Tag one of generic_spct, and derived classes including source_spct, filter_spct, reflector_spct, object_spct, and response_spct.
- generic_mspct: Tag one of generic_mspct, and derived classes including source_mspct, filter_mspct, reflector_mspct, object_mspct, and response_mspct.

Note

NULL as w.band argument does not add any new tags, instead it removes existing tags if present. NA, the default, as w.band argument removes existing waveband tags if present and sets the wl.color variable. If a waveband object or a list of wavebands is supplied as argument then tagging is based on them, and wl.color is also set.

See Also

Other tagging and related functions: is_tagged, untag, wb2rect_spct, wb2spct, wb2tagged_spct

Examples

```
tag(sun.spct)
tag(sun.spct, list(A = waveband(c(300,3005))))
```

---

times-.generic_spct *Arithmetic Operators*

Description

Multiplication operator for spectra.

Usage

```
## S3 method for class 'generic_spct'
e1 * e2
```

Arguments

e1	an object of class "generic_spct"
e2	an object of class "generic_spct"

See Also

Other math operators and functions: MathFun, ^.generic_spct, convolve_each, div-.generic_spct, log, minus-.generic_spct, mod-.generic_spct, plus-.generic_spct, round, sign, slash-.generic_spct

---

`transmittance` *Transmittance*

Description

Summary transmittance for supplied wavebands from filter or object spectrum.

Usage

```
transmittance(spct, w.band, quantity, wb.trim, use.hinges, ...)

## Default S3 method:
transmittance(spct, w.band, quantity, wb.trim,
  use.hinges, ...)

## S3 method for class 'filter_spct'
transmittance(spct, w.band = NULL,
  quantity = "average",
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.hinges = getOption("photobiology.use.hinges", default = NULL), ...)

## S3 method for class 'object_spct'
transmittance(spct, w.band = NULL,
  quantity = "average",
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.hinges = getOption("photobiology.use.hinges", default = NULL), ...)

## S3 method for class 'filter_mspct'
transmittance(spct, w.band = NULL,
  quantity = "average",
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.hinges = getOption("photobiology.use.hinges", default = NULL), ...,
  attr2tb = NULL, idx = "spct.idx")

## S3 method for class 'object_mspct'
transmittance(spct, w.band = NULL,
  quantity = "average",
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.hinges = getOption("photobiology.use.hinges", default = NULL), ...,
  attr2tb = NULL, idx = "spct.idx", .parallel = FALSE,
  .paropts = NULL)
```

Arguments

spct	an R object.
w.band	waveband or list of wavebands objects or a numeric vector of length two. The waveband(s) determine the region(s) of the spectrum that are summarized. If a numeric range is supplied a waveband object is constructed on the fly from it.
quantity	character string One of "total", "average" or "mean", "contribution", "contribution.pc", "relative" or "relative.pc".
wb.trim	logical Flag indicating if wavebands crossing spectral data boundaries are trimmed or ignored.
use.hinges	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.
...	ignored (possibly used by derived methods).
attr2tb	character vector, see add_attr2tb for the syntax for attr2tb passed as is to formal parameter col.names.
idx	character Name of the column with the names of the members of the collection of spectra.
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Value

A named numeric vector in the case of methods for individual spectra, with one value for each waveband passed to parameter w.band. A data.frame in the case of collections of spectra, containing one column for each waveband object, an index column with the names of the spectra, and optionally additional columns with metadata values retrieved from the attributes of the member spectra.

By default values are only integrated, but depending on the argument passed to parameter quantity they can be re-expressed as relative fractions or percentages. In the case of vector output, names attribute is set to the name of the corresponding waveband unless a named list is supplied in which case the names of the list members are used.

Methods (by class)

- default: Default method
- filter_spct: Method for filter spectra
- object_spct: Method for object spectra
- filter_mspct: Calculates transmittance from a filter_mspct
- object_mspct: Calculates transmittance from a object_mspct

Note

The `use.hinges` parameter controls speed optimization. The defaults should be suitable in most cases. Only the range of wavelengths in the wavebands is used and all BSWFs are ignored.

Examples

```
transmittance(polyester.spct, waveband(c(280, 315)))
transmittance(polyester.spct, waveband(c(315, 400)))
transmittance(polyester.spct, waveband(c(400, 700)))
```

Trig *Trigonometric Functions*

Description

Trigonometric functions for object of `generic_spct` and derived classes. \ The functions are applied to the spectral data, not the wavelengths. The quantity in the spectrum to which the function is applied depends on the class of `x` and the current value of output options.

Usage

```
## S3 method for class 'generic_spct'
cos(x)

## S3 method for class 'generic_spct'
sin(x)

## S3 method for class 'generic_spct'
tan(x)

## S3 method for class 'generic_spct'
acos(x)

## S3 method for class 'generic_spct'
asin(x)

## S3 method for class 'generic_spct'
atan(x)

cospi.generic_spct(x)

sinpi.generic_spct(x)

tanpi.generic_spct(x)
```

Arguments

x	an object of class "generic_spct" or a derived class.

`trimInstrDesc` *Trim the "instr.desc" attribute*

Description

Function to trim the "instr.desc" attribute of an existing generic_spct object, discarding all fields except for ‘spectrometer.name‘, ‘spectrometer.sn‘, ‘bench.grating‘, ‘bench.slit‘, and calibration name.

Usage

```
trimInstrDesc(x, fields = c("time", "spectrometer.name",
  "spectrometer.sn", "bench.grating", "bench.slit"))
```

Arguments

x	a generic_spct object
fields	a character vector with the names of the fields to keep, or if first member is ‘"-"‘, the names of fields to delete; "*" as first member of the vector makes the function a no-op, leaving the spectrum object unaltered.

Value

x

Note

This function alters x itself by reference and in addition returns x invisibly. If x is not a generic_spct object, x is not modified.

See Also

Other measurement metadata functions: `getHowMeasured`, `getInstrDesc`, `getInstrSettings`, `getWhatMeasured`, `getWhenMeasured`, `getWhereMeasured`, `get_attributes`, `isValidInstrDesc`, `isValidInstrSettings`, `setHowMeasured`, `setInstrDesc`, `setInstrSettings`, `setWhatMeasured`, `setWhenMeasured`, `setWhereMeasured`, `trimInstrSettings`

trimInstrSettings *Trim the "instr.settings" attribute*

Description

Function to trim the "instr.settings" attribute of an existing generic_spct object, by discarding some fields.

Usage

```
trimInstrSettings(x, fields = "*")
```

Arguments

x	a generic_spct object
fields	a character vector with the names of the fields to keep, or if first member is '"-"', the names of fields to delete; "*" as first member of the vector makes the function a no-op, leaving the spectrum object unaltered.

Value

x

Note

This function alters x itself by reference and in addition returns x invisibly. If x is not a generic_spct object, x is not modified.

See Also

Other measurement metadata functions: getHowMeasured, getInstrDesc, getInstrSettings, getWhatMeasured, getWhenMeasured, getWhereMeasured, get_attributes, isValidInstrDesc, isValidInstrSettings, setHowMeasured, setInstrDesc, setInstrSettings, setWhatMeasured, setWhenMeasured, setWhereMeasured, trimInstrDesc

trim_spct *Trim (or expand) head and/or tail of a spectrum*

Description

Trim head and tail of a spectrum based on wavelength limits, interpolating the values at the boundaries of the range. Trimming is needed for example to remove short wavelength noise when the measured spectrum extends beyond the known emission spectrum of the measured light source. Occasionally one may want also to expand the wavelength range.

Usage

```
trim_spct(spct, range = NULL, low.limit = NULL, high.limit = NULL,
  use.hinges = TRUE, fill = NULL, byref = FALSE,
  verbose = getOption("photobiology.verbose"))

trim_mspct(mspct, range = NULL, low.limit = NULL, high.limit = NULL,
  use.hinges = TRUE, fill = NULL, byref = FALSE,
  verbose = getOption("photobiology.verbose"), .parallel = FALSE,
  .paropts = NULL)

trim2overlap(mspct, use.hinges = TRUE,
  verbose = getOption("photobiology.verbose"), .parallel = FALSE,
  .paropts = NULL)

extend2extremes(mspct, use.hinges = TRUE, fill = NA,
  verbose = getOption("photobiology.verbose"), .parallel = FALSE,
  .paropts = NULL)
```

Arguments

spct	an object of class "generic_spct".
range	a numeric vector of length two, or any other object for which method range() will return a numeric vector of length two.
low.limit	shortest wavelength to be kept (defaults to shortest w.length value).
high.limit	longest wavelength to be kept (defaults to longest w.length value).
use.hinges	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.
fill	if fill==NULL then tails are deleted, otherwise tails or s.irrad are filled with the value of fill.
byref	logical indicating if new object will be created by reference or by copy of spct.
verbose	logical.
mspct	an object of class "generic_mspct"
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Value

a spectrum of same class as input with its tails trimmed or expanded.

Note

When expanding an spectrum, if fill==NULL, then expansion is not performed. Range can be "waveband" object, a numeric vector or a list of numeric vectors, or any other user-defined or built-in object for which `range()` returns a numeric vector of length two, that can be interpreted as wavelengths expressed in nm.

See Also

Other trim functions: `clip_wl`, `trim_waveband`, `trim_wl`

Examples

```
trim_spct(sun.spct, low.limit=300)
trim_spct(sun.spct, low.limit=300, fill=NULL)
trim_spct(sun.spct, low.limit=300, fill=NA)
trim_spct(sun.spct, low.limit=300, fill=0.0)
trim_spct(sun.spct, range = c(300, 400))
trim_spct(sun.spct, range = c(300, NA))
trim_spct(sun.spct, range = c(NA, 400))
```

`trim_tails` *Trim (or expand) head and/or tail*

Description

Trim tails of a spectrum based on wavelength limits, interpolating the values at the boundaries.Trimming is needed for example to remove short wavelength noise when the measured spectrum extends beyond the known emission spectrum of the measured light source. Occasionally one may want also to expand the wavelength range.

Usage

```
trim_tails(x, y, low.limit = min(x), high.limit = max(x),
  use.hinges = TRUE, fill = NULL, verbose = TRUE)
```

Arguments

`x`	numeric vector of wavelengths.
`y`	numeric vector of values for a spectral quantity.
`low.limit`	smallest x-value to be kept (defaults to smallest x-value in input).
`high.limit`	largest x-value to be kept (defaults to largest x-value in input).
`use.hinges`	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.
`fill`	if `fill == NULL` then tails are deleted, otherwise tails of y are filled with the value of `fill`.
`verbose`	logical Use to suppress warnings.

Value

A data.frame with variables x and y.

Note

When expanding an spectrum, if `fill == NULL`, expansion is not performed with a warning.

See Also

Other low-level functions operating on numeric vectors.: as_energy, as_quantum_mol, calc_multipliers, div_spectra, energy_irradiance, energy_ratio, insert_hinges, integrate_xy, interpolate_spectrum, irradiance, l_insert_hinges, oper_spectra, photon_irradiance, photon_ratio, photons_energy_ratio, prod_spectra, s_e_irrad2rgb, split_energy_irradiance, split_photon_irradiance, subt_spectra, sum_spectra, v_insert_hinges, v_replace_hinges

Examples

```
head(sun.data)
head(with(sun.data,
     trim_tails(w.length, s.e.irrad, low.limit=300)))
head(with(sun.data,
     trim_tails(w.length, s.e.irrad, low.limit=300, fill=NULL)))
```

---

trim_waveband *Trim (or expand) head and/or tail*

Description

Trimming of waveband boundaries can be required needed when the spectral data does not cover the whole waveband.

Usage

```
trim_waveband(w.band, range = NULL, low.limit = 0, high.limit = Inf,
  trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.hinges = TRUE)
```

Arguments

w.band	an object of class "waveband" or a list of such objects.
range	a numeric vector of length two, or any other object for which function range() will return a numeric vector of two wavelengths (nm).
low.limit	shortest wavelength to be kept (defaults to 0 nm).
high.limit	longest wavelength to be kept (defaults to Inf nm).
trim	logical (default is TRUE which trims the wavebands at the boundary, while FALSE discards wavebands that are partly off-boundary).

use.hinges — logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.

Value

A waveband object or a list of waveband objects trimmed or filtered depending on whether a single waveband object or a list of waveband objects was supplied as argument to formal parameter w.band.

See Also

Other trim functions: clip_wl, trim_spct, trim_wl

Examples

```
VIS <- waveband(c(380, 760)) # nanometres

trim_waveband(VIS, c(400,700))
trim_waveband(VIS, low.limit = 400)
trim_waveband(VIS, high.limit = 700)
```

---

trim_wl *Trim head and/or tail of a spectrum*

Description

Trim head and tail of a spectrum based on wavelength limits, with interpolation at range boundaries used by default. Expansion is also possible.

Usage

```
trim_wl(x, range, use.hinges, fill, ...)

## Default S3 method:
trim_wl(x, range, use.hinges, fill, ...)

## S3 method for class 'generic_spct'
trim_wl(x, range = NULL, use.hinges = TRUE,
  fill = NULL, ...)

## S3 method for class 'generic_mspct'
trim_wl(x, range = NULL, use.hinges = TRUE,
  fill = NULL, ..., .parallel = FALSE, .paropts = NULL)

## S3 method for class 'waveband'
trim_wl(x, range = NULL, use.hinges = TRUE,
```

```
  fill = NULL, trim = getOption("photobiology.waveband.trim", default =
  TRUE), ...)

## S3 method for class 'list'
trim_wl(x, range = NULL, use.hinges = TRUE,
  fill = NULL, trim = getOption("photobiology.waveband.trim", default =
  TRUE), ...)
```

Arguments

x	an R object.
range	a numeric vector of length two, or any other object for which function range() will return two.
use.hinges	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.
fill	if fill == NULL then tails are deleted, otherwise tails are filled with the value of fill.
...	ignored (possibly used by derived methods).
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.
trim	logical (default is TRUE which trims the wavebands at the boundary, while FALSE discards wavebands that are partly off-boundary).

Value

A copy of x, usually trimmed or expanded to a different length, either shorter or longer. Possibly with some of the original spectral data values replaced with fill.

Methods (by class)

- default: Default for generic function
- generic_spct: Trim an object of class "generic_spct" or derived.
- generic_mspct: Trim an object of class "generic_mspct" or derived.
- waveband: Trim an object of class "waveband".
- list: Trim a list (of "waveband" objects).

Note

By default the w.length values for the first and last rows in the returned object are the values supplied as range.

trim_wl when applied to waveband objects always inserts hinges when trimming.

trim_wl when applied to waveband objects always inserts hinges when trimming.

See Also

Other trim functions: clip_wl, trim_spct, trim_waveband

Examples

```
trim_wl(sun.spct, range = c(400, 500))
trim_wl(sun.spct, range = c(NA, 500))
trim_wl(sun.spct, range = c(400, NA))
```

---

tz_time_diff *Time difference between two time zones*

Description

Returns the time difference in hours between two time zones at a given instant in time.

Usage

```
tz_time_diff(when = lubridate::now(), tz.target = lubridate::tz(when),
  tz.reference = "UTC")
```

Arguments

when	datetime A time instant
tz.target, tz.reference	character Two time zones using names recognized by functions from package 'lubridate'

Value

A numeric value.

---

untag *Remove tags*

Description

Remove tags from an R object if present, otherwise return the object unchanged.

Usage

```
untag(x, ...)

## Default S3 method:
untag(x, ...)

## S3 method for class 'generic_spct'
untag(x, byref = FALSE, ...)

## S3 method for class 'generic_mspct'
untag(x, byref = FALSE, ...)
```

Arguments

x	an R object.
...	ignored (possibly used by derived methods).
byref	logical indicating if new object will be created by reference or by copy of x

Value

if x contains tag data they are removed and the "spct.tags" attribute is set to NA, while if x has no tags, it is not modified. In either case, the byref argument is respected: in all cases if byref = FALSE a copy of x is returned.

Methods (by class)

- default: Default for generic function
- generic_spct: Specialization for generic_spct
- generic_mspct: Specialization for generic_spct

See Also

Other tagging and related functions: is_tagged, tag, wb2rect_spct, wb2spct, wb2tagged_spct

---

upgrade_spct *Upgrade one spectral object*

Description

Update the spectral class names of objects to those used in photobiology (>= 0.6.0) and add 'version' attribute as used in photobiology (>= 0.70).

Usage

```
upgrade_spct(object)
```

Arguments

object	generic.spct A single object to upgrade

Value

The modified object (invisibly).

Note

The object is modified by reference. The class names with ending ".spct" replaced by their new equivalents ending in "_spct".

See Also

Other upgrade from earlier versions: is.old_spct, upgrade_spectra

---

upgrade_spectra *Upgrade one or more spectral objects*

Description

Update the spectral class names of objects to those used in photobiology (>= 0.6.0).

Usage

```
upgrade_spectra(obj.names = ls(parent.frame()))
```

Arguments

obj.names	char Names of objects to upgrade as a vector of character strings

Value

The modified object (invisibly).

Note

The objects are modified by reference. The class names with ending ".spct" are replaced by their new equivalents ending in "_spct". object.names can safely include names of any R object. Names of objects which do not belong to any the old .spct classes are ignored. This makes it possible to supply as argument the output from ls, the default, or its equivalent objects.

See Also

Other upgrade from earlier versions: is.old_spct, upgrade_spct

using_Tfr *Use photobiology options*

Description

Execute an R expression, possibly compound, using a certain setting for spectral data related options.

Usage

```
using_Tfr(expr)

using_Afr(expr)

using_A(expr)

using_energy(expr)

using_photon(expr)

using_quantum(expr)
```

Arguments

expr	an R expression to execute.

Value

The value returned by the execution of `expression`.

References

Based on `withOptions()` as offered by Thomas Lumley, and listed in http://www.burns-stat.com/the-options-mechanism-in-r/, section Deep End, of "The Options mechanism in R" by Patrick Burns.

valleys *Valleys or local minima*

Description

Function that returns a subset of an R object with observations corresponding to local maxima.

Usage

```
valleys(x, span, ignore_threshold, strict, ...)

## Default S3 method:
valleys(x, span = NA, ignore_threshold = NA,
  strict = NA, na.rm = FALSE, ...)

## Default S3 method:
valleys(x, span = NA, ignore_threshold = NA,
  strict = NA, na.rm = FALSE, ...)

## S3 method for class 'numeric'
valleys(x, span = 5, ignore_threshold, strict = TRUE,
  na.rm = FALSE, ...)

## S3 method for class 'data.frame'
valleys(x, span = 5, ignore_threshold = 0,
  strict = TRUE, na.rm = FALSE, var.name, ...)

## S3 method for class 'generic_spct'
valleys(x, span = 5, ignore_threshold = 0,
  strict = TRUE, na.rm = FALSE, var.name = NULL, ...)

## S3 method for class 'source_spct'
valleys(x, span = 5, ignore_threshold = 0,
  strict = TRUE, na.rm = FALSE,
  unit.out = getOption("photobiology.radiation.unit", default =
  "energy"), ...)

## S3 method for class 'response_spct'
valleys(x, span = 5, ignore_threshold = 0,
  strict = TRUE, na.rm = FALSE,
  unit.out = getOption("photobiology.radiation.unit", default =
  "energy"), ...)

## S3 method for class 'filter_spct'
valleys(x, span = 5, ignore_threshold = 0,
  strict = TRUE, na.rm = FALSE,
  filter.qty = getOption("photobiology.filter.qty", default =
  "transmittance"), ...)

## S3 method for class 'reflector_spct'
valleys(x, span = 5, ignore_threshold = 0,
  strict = TRUE, na.rm = FALSE, ...)

## S3 method for class 'cps_spct'
valleys(x, span = 5, ignore_threshold = 0,
  strict = TRUE, na.rm = FALSE, ...)
```

```
## S3 method for class 'generic_mspct'
valleys(x, span = 5, ignore_threshold = 0,
  strict = TRUE, na.rm = FALSE, ..., .parallel = FALSE,
  .paropts = NULL)
```

Arguments

x	an R object
span	a peak is defined as an element in a sequence which is greater than all other elements within a window of width span centered at that element. The default value is 3, meaning that a peak is bigger than both of its neighbors. Default: 3.
ignore_threshold	numeric value between 0.0 and 1.0 indicating the relative size compared to tallest peak threshold below which valleys will be ignored.
strict	logical flag: if TRUE, an element must be strictly greater than all other values in its window to be considered a peak. Default: TRUE.
...	ignored
na.rm	logical indicating whether NA values should be stripped before searching for peaks.
var.name	Name of column where to look for peaks.
unit.out	character One of "energy" or "photon"
filter.qty	character One of "transmittance" or "absorbance"
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Value

A subset of x with rows corresponding to local minima.

Methods (by class)

- default: Default function usable on numeric vectors.
- default: Default returning always NA.
- numeric: Default function usable on numeric vectors.
- data.frame: Method for "data.frame" objects.
- generic_spct: Method for "generic_spct" objects.
- source_spct: Method for "source_spct" objects.
- response_spct: Method for "response_spct" objects.
- filter_spct: Method for "filter_spct" objects.
- reflector_spct: Method for "reflector_spct".
- cps_spct: Method for "cps_spct" objects.
- generic_mspct: Method for "generic_mspct" objects.

See Also

Other peaks and valleys functions: find_peaks, get_peaks, peaks, wls_at_target

Examples

```
valleys(sun.spct, span = 50)

valleys(sun.spct)
```

---

verbose_as_default *Set error reporting options*

Description

Set error reporting related options easily.

Usage

```
verbose_as_default(flag = TRUE)

strict_range_as_default(flag = TRUE)
```

Arguments

flag	logical.

Value

Previous value of the modified option.

---

v_insert_hinges *Insert spectral data values at new wavelength values.*

Description

Inserting wavelengths values immediately before and after a discontinuity in the SWF, greatly reduces the errors caused by interpolating the weighted irradiance during integration of the effective spectral irradiance. This is specially true when data have a relatively large wavelength step size and/or when the weighting function used has discontinuities in its value or slope. This function differs from insert_hinges() in that it returns a vector of y values instead of a tibble.

Usage

```
v_insert_hinges(x, y, h)
```

Arguments

x	numeric vector (sorted in increasing order).
y	numeric vector.
h	a numeric vector giving the wavelengths at which the y values should be inserted by interpolation, no interpolation is indicated by an empty numeric vector (`numeric(0)`).

Value

A numeric vector with the numeric values of y, but longer. Unless the hinge values were already present in y, each inserted hinge, expands the vector by two values.

See Also

Other low-level functions operating on numeric vectors.: `as_energy`, `as_quantum_mol`, `calc_multipliers`, `div_spectra`, `energy_irradiance`, `energy_ratio`, `insert_hinges`, `integrate_xy`, `interpolate_spectrum`, `irradiance`, `l_insert_hinges`, `oper_spectra`, `photon_irradiance`, `photon_ratio`, `photons_energy_ratio`, `prod_spectra`, `s_e_irrad2rgb`, `split_energy_irradiance`, `split_photon_irradiance`, `subt_spectra`, `sum_spectra`, `trim_tails`, `v_replace_hinges`

---

v_replace_hinges *Overwrite spectral data values at existing wavelength values.*

Description

Overwriting spectral data with interpolated values at wavelengths values containing bad data is needed when cleaning spectral data. This function differs from `insert_hinges()` in that it returns a vector of y values instead of a `tibble`.

Usage

```
v_replace_hinges(x, y, h)
```

Arguments

x	numeric vector (sorted in increasing order).
y	numeric vector.
h	a numeric vector giving the wavelengths at which the y values should be replaced by interpolation, no interpolation is indicated by an empty numeric vector (`numeric(0)`).

Value

A numeric vector with the numeric values of y with values at the hinges replaced by interpolation of neighbours.

See Also

Other low-level functions operating on numeric vectors.: as_energy, as_quantum_mol, calc_multipliers, div_spectra, energy_irradiance, energy_ratio, insert_hinges, integrate_xy, interpolate_spectrum, irradiance, l_insert_hinges, oper_spectra, photon_irradiance, photon_ratio, photons_energy_ratio, prod_spectra, s_e_irrad2rgb, split_energy_irradiance, split_photon_irradiance, subt_spectra, sum_spectra, trim_tails, v_insert_hinges

---

water_vp_sat *Water vapour pressure*

Description

Approximate water pressure in air as a function of temperature, and its inverse the calculation of dewpoint.

Usage

```
water_vp_sat(temperature, over.ice = FALSE, method = "tetens",
  check.range = TRUE)

water_dp(water.vp, over.ice = FALSE, method = "tetens",
  check.range = TRUE)

water_fp(water.vp, over.ice = TRUE, method = "tetens",
  check.range = TRUE)

water_vp2mvc(water.vp, temperature)

water_mvc2vp(water.mvc, temperature)

water_vp2RH(water.vp, temperature, over.ice = FALSE, method = "tetens",
  pc = TRUE, check.range = TRUE)

water_RH2vp(relative.humidity, temperature, over.ice = FALSE,
  method = "tetens", pc = TRUE, check.range = TRUE)
```

Arguments

temperature	numeric vector of air temperatures (C).
over.ice	logical Is the estimate for equilibrium with liquid water or with ice.
method	character Currently "tetens", modified "magnus", "wexler" and "goff.gratch" equations are supported.
check.range	logical Flag indicating whether to check or not that arguments for temperature are within the range of validity of the method used.
water.vp	numeric vector of water vapour pressure in air (Pa).

`water.mvc` numeric vector of water vapour concnetration as mass per volume (g m-3).

`pc` logical flag for result returned as percent or not.

`relative.humidity` numeric Relative humedity as fraction of 1.

Details

Function `water_vp_sat()` provides implementations of several well known equations for the estimation of saturation vapor pressure in air. Functions `water_dp()` and `water_fp()` use the inverse of these equations to compute the dew point or frost point from water vapour pressure in air. The inverse functions are either analytical solutions or fitted approximations. None of these functions are solved numerically by iteration.

Method `"tetens"` implements Tetens' (1930) equation for the cases of equilibrium with a water and an ice surface. Method `"magnus"` implements the modified Magnus equations of Alduchov and Eskridge (1996, eqs. 21 and 23). Method `"wexler"` implements the equations proposed by Wexler (1976, 1977), and their inverse according to Hardy (1998). Method `"goff.gratch"` implements the equations of Groff and Gratch (1946) with the minor updates of Groff (1956).

The equations are approximations, and in spite of their different names, Tetens' and Magnus' equations have the same form with the only difference in the values of the parameters. However, the modified Magnus equation is more accurate as Tetens equation suffers from some bias errors at extreme low temperatures (< -40 C). In contrast Magnus equations with recently fitted values for the parameters are usable for temperatures from -80 C to +50 C over water and -80 C to 0 C over ice. The Groff Gratch equation is more complex and is frequently used as a reference in comparison as it is considered reliable over a broad range of temperatures. Wexler's equations are computationally simpler and fitted to relatively recent data. There is little difference at temperatures in the range -20 C to +50 C, and differences become large at extreme temperatures. Temperatures outside the range where estimations are highly reliable for each equation return `NA`, unless extrapolation is enabled by passing `FALSE` as argument to parameter `check.range`.

The switch between equations for ice or water cannot be based on air temperature, as it depends on the presence or not of a surface of liquid water. It must be set by passing an argument to parameter `over.ice` which defaults to `FALSE`.

Tetens equation is still very frequently used, and is for example the one recommended by FAO for computing potential evapotranspiration. For this reason it is used as default here.

Value

A numeric vector of partial pressures in pascal (P) for `water_vp_sat` and `water_mvc2vp`, a numeric vector of dew point temperatures (C) for `water_dp` and numeric vector of mass per volume concentrations (g m-3) for `water_vp2mvc`.

Note

The inverse of the Groff Gratch equation has yet to be implemented.

References

Tetens, O., 1930. Uber einige meteorologische Begriffe. Zeitschrift fur Geophysik, Vol. 6:297.

Goff, J. A., and S. Gratch (1946) Low-pressure properties of water from -160 to 212 F, in Transactions of the American Society of Heating and Ventilating Engineers, pp 95-122, presented at the 52nd annual meeting of the American Society of Heating and Ventilating Engineers, New York, 1946.

Wexler, A. (1976) Vapor Pressure Formulation for Water in Range 0 to 100°C. A Revision, Journal of Research ofthe National Bureau of Standards: A. Physics and Chemistry, September-December 1976, Vol. 80A, Nos.5 and 6, 775-785

Wexler, A., Vapor Pressure Formulation for Ice, Journal of Research of the National Bureau of Standards - A. Physics and Chemistry, January - February 1977, Vol. 81A, No. 1, 5-19

Alduchov, O. A., Eskridge, R. E., 1996. Improved Magnus Form Approximation of Saturation Vapor Pressure. Journal of Applied Meteorology, 35: 601-609 .

Hardy, Bob (1998) ITS-90 formulations for vapor pressure, frostpoint temperature, dewpoint temperature, andenhancement factors in the range -100 TO +100 C. The Proceedings of the Third International Symposium on Humidity & Moisture, Teddington, London, England, April 1998. http://www.decatur.de/javascript/dew/resources/its90formulas.pdf

Monteith, J., Unsworth, M. (2008) Principles of Environmental Physics. Academic Press, Amsterdam.

[Equations describing the physical properties of moist air](http://www.conservationphysics.org/atmcalc/atmoclc2.pdf)

Examples

```
water_vp_sat(20) # C -> Pa
water_vp_sat(temperature = c(0, 10, 20, 30, 40)) # C -> Pa
water_vp_sat(temperature = -10) # over water!!
water_vp_sat(temperature = -10, over.ice = TRUE)
water_vp_sat(temperature = 20) / 100 # C -> mbar

water_vp_sat(temperature = 20, method = "magnus") # C -> Pa
water_vp_sat(temperature = 20, method = "tetens") # C -> Pa
water_vp_sat(temperature = 20, method = "wexler") # C -> Pa
water_vp_sat(temperature = 20, method = "goff.gratch") # C -> Pa

water_vp_sat(temperature = -20, over.ice = TRUE, method = "magnus") # C -> Pa
water_vp_sat(temperature = -20, over.ice = TRUE, method = "tetens") # C -> Pa
water_vp_sat(temperature = -20, over.ice = TRUE, method = "wexler") # C -> Pa
water_vp_sat(temperature = -20, over.ice = TRUE, method = "goff.gratch") # C -> Pa

water_dp(water.vp = 1000) # Pa -> C
water_dp(water.vp = 1000, method = "magnus") # Pa -> C
water_dp(water.vp = 1000, method = "wexler") # Pa -> C
water_dp(water.vp = 500, over.ice = TRUE) # Pa -> C
water_dp(water.vp = 500, method = "wexler", over.ice = TRUE) # Pa -> C

water_fp(water.vp = 300) # Pa -> C
water_dp(water.vp = 300, over.ice = TRUE) # Pa -> C

water_vp2RH(water.vp = 1500, temperature = 20) # Pa, C -> RH %
water_vp2RH(water.vp = 1500, temperature = c(20, 30)) # Pa, C -> RH %
water_vp2RH(water.vp = c(600, 1500), temperature = 20) # Pa, C -> RH %
```

```
water_vp2mvc(water.vp = 1000, temperature = 20) # Pa -> g m-3

water_mvc2vp(water.mvc = 30, temperature = 40) # g m-3 -> Pa

water_dp(water.vp = water_mvc2vp(water.mvc = 10, temperature = 30)) # g m-3 -> C
```

---

waveband *Waveband constructor method*

Description

Constructor for "waveband" objects that can be used as input when calculating irradiances.

Usage

```
waveband(x = NULL, weight = NULL, SWF.e.fun = NULL,
  SWF.q.fun = NULL, norm = NULL, SWF.norm = NULL, hinges = NULL,
  wb.name = NULL, wb.label = wb.name)

new_waveband(w.low, w.high, weight = NULL, SWF.e.fun = NULL,
  SWF.q.fun = NULL, norm = NULL, SWF.norm = NULL, hinges = NULL,
  wb.name = NULL, wb.label = wb.name)
```

Arguments

x	any R object on which applying the function range yields an vector of two numeric values, describing a range of wavelengths (nm)
weight	a character string "SWF" or "BSWF", use NULL (the default) to indicate no weighting used when calculating irradiance
SWF.e.fun	a function giving multipliers for a spectral weighting function (energy) as a function of wavelength (nm)
SWF.q.fun	a function giving multipliers for a spectral weighting function (quantum) as a function of wavelength (nm)
norm	a single numeric value indicating the wavelength at which the SWF should be normalized to 1.0, in nm. "NULL" means no normalization.
SWF.norm	a numeric value giving the native normalization wavelength (nm) used by SWF.e.fun and SWF.q.fun
hinges	a numeric vector giving the wavelengths at which the s.irrad should be inserted by interpolation, no interpolation is indicated by an empty vector (numeric(0)), if NULL then interpolation will take place at both ends of the band.
wb.name	character string giving the name for the waveband defined, default is NULL
wb.label	character string giving the label of the waveband to be used for plotting, default is wb.name
w.low	numeric value, wavelength at the short end of the band (nm)
w.high	numeric value, wavelength at the long end of the band (nm)

Value

a waveband object

Functions

- new_waveband: A less flexible variant

See Also

Other waveband constructors: split_bands

Examples

```
waveband(c(400,700))

new_waveband(400,700)
```

---

waveband_ratio *Photon or energy ratio*

Description

This function gives the (energy or photon) irradiance ratio between two given wavebands of a radiation spectrum.

Usage

```
waveband_ratio(w.length, s.irrad, w.band.num = NULL,
  w.band.denom = NULL, unit.out.num = NULL,
  unit.out.denom = unit.out.num, unit.in = "energy",
  check.spectrum = TRUE, use.cached.mult = FALSE,
  use.hinges = getOption("photobiology.use.hinges", default = NULL))
```

Arguments

w.length	numeric Vector of wavelengths (nm)
s.irrad	numeric vector of spectral (energy or photon) irradiances (W m-2 nm-1) or (mol s-1 m-2 nm-1).
w.band.num	waveband object used to compute the numerator of the ratio.
w.band.denom	waveband object used to compute the denominator of the ratio.
unit.out.num	character Allowed values "energy", and "photon", or its alias "quantum".
unit.out.denom	character Allowed values "energy", and "photon", or its alias "quantum".
unit.in	character Allowed values "energy", and "photon", or its alias "quantum".
check.spectrum	logical Flag indicating whether to sanity check input data, default is TRUE.

use.cached.mult
: logical Flag indicating whether multiplier values should be cached between calls.

use.hinges
: logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.

Value

a single numeric value giving the ratio

Note

The default for both w.band parameters is a waveband covering the whole range of w.length. From version 9.19 onwards use of this default does not trigger a warning, but instead is used silently.

Examples

```
# photon:photon ratio
with(sun.data,
     waveband_ratio(w.length, s.e.irrad,
                    new_waveband(400,500),
                    new_waveband(400,700), "photon"))
# energy:energy ratio
with(sun.data,
     waveband_ratio(w.length, s.e.irrad,
                    new_waveband(400,500),
                    new_waveband(400,700), "energy"))
# energy:photon ratio
with(sun.data,
     waveband_ratio(w.length, s.e.irrad,
                    new_waveband(400,700),
                    new_waveband(400,700),
                    "energy", "photon"))
# photon:photon ratio waveband : whole spectrum
with(sun.data,
     waveband_ratio(w.length, s.e.irrad,
                    new_waveband(400,500),
                    unit.out.num="photon"))
# photon:photon ratio of whole spectrum should be equal to 1.0
with(sun.data,
     waveband_ratio(w.length, s.e.irrad,
     unit.out.num="photon"))
```

---

wb2rect_spct *Create tagged spectrum from wavebands*

Description

Create a generic_spct object with wavelengths from the range of wavebands in a list. The spectrum is suitable for plotting labels, symbols, rectangles or similar, as the midpoint of each waveband is added to the spectrum.

Usage

```
wb2rect_spct(w.band, short.names = TRUE)
```

Arguments

w.band	waveband or list of waveband objects The waveband(s) determine the wavelengths in variable w.length of the returned spectrum
short.names	logical Flag indicating whether to use short or long names for wavebands

Value

A generic.spectrum object, with columns w.length, wl.low, wl.hi, wl.color, wb.color and wb.name. The w.length values are the midpoint of the wavebands, wl.low and wl.high give the boundaries of the wavebands, wl.color the color definition corresponding to the wavelength at the center of the waveband and wb.color the color of the waveband as a whole (assuming a flat energy irradiance spectrum). Different spectral data variables are set to zero and added making the returned value compatible with classes derived from generic_spct.

See Also

Other tagging and related functions: is_tagged, tag, untag, wb2spct, wb2tagged_spct

---

wb2spct *Create spectrum from wavebands*

Description

Create a generic_spct object with wavelengths from wavebands in a list.

Usage

```
wb2spct(w.band)
```

Arguments

w.band	waveband or list of waveband objects The waveband(s) determine the wavelengths in variable w.length of the returned spectrum

Value

A generic.spectrum object, with columns w.length set to the *union* of all boundaries and hinges defined in the waveband(s). Different spectral data variables are set to zero and added making the returned value compatible with classes derived from generic_spct.

See Also

Other tagging and related functions: is_tagged, tag, untag, wb2rect_spct, wb2tagged_spct

---

wb2tagged_spct *Create tagged spectrum from wavebands*

Description

Create a tagged generic_spct object with wavelengths from the range of wavebands in a list, and names of the same bands as factor levels, and corresponding color definitions. The spectrum is not suitable for plotting labels, symbols, rectangles or similar, as the midpoint of each waveband is not added to the spectrum.

Usage

```
wb2tagged_spct(w.band, use.hinges = TRUE, short.names = TRUE, ...)
```

Arguments

w.band	waveband or list of waveband objects The waveband(s) determine the region(s) of the spectrum that are tagged and the wavelengths returned in variable w.length.
use.hinges	logical Flag indicating whether to insert "hinges" into the spectral data before integration so as to reduce interpolation errors at the boundaries of the wavebands.
short.names	logical Flag indicating whether to use short or long names for wavebands.
...	ignored (possibly used by derived methods).

Value

A spectrum as returned by wb2spct but additionally tagged using function tag

See Also

Other tagging and related functions: is_tagged, tag, untag, wb2rect_spct, wb2spct

wb_trim_as_default *Set computation options*

Description

Set computation related options easily.

Usage

```
wb_trim_as_default(flag = TRUE)

use_cached_mult_as_default(flag = TRUE)
```

Arguments

flag	logical.

Value

Previous value of the modified option.

white_body.spct *Theoretical white body*

Description

A dataset for a hypothetical object with transmittance 0/1 (0%), reflectance 1/1 (100%)

Format

A object_spct object with 4 rows and 3 variables

Details

- w.length (nm)
- Tfr (0..1)
- Rfr (0..1)

See Also

Other Spectral data examples: A.illuminant.spct, D65.illuminant.spct, Ler_leaf.spct, Ler_leaf_rflt.spct, Ler_leaf_trns.spct, Ler_leaf_trns_i.spct, black_body.spct, ccd.spct, clear.spct, clear_body.spct, filter_cps.mspct, green_leaf.spct, opaque.spct, photodiode.spct, polyester.spct, sun.daily.data, sun.daily.spct, sun.data, sun.spct, white_led.cps_spct, white_led.raw_spct, white_led.source_spct, yellow_gel.spct

white_led.cps_spct *White led bulb spectrum*

Description

A dataset containing wavelengths and the corresponding spectral data as counts per second for an Osram warm white led lamp:

Usage

```
white_led.cps_spct
```

Format

A `data.frame` object with 2068 rows and 2 variables

Details

- w.length (nm), range 188 to 1117 nm.
- cps

See Also

Other Spectral data examples: `A.illuminant.spct`, `D65.illuminant.spct`, `Ler_leaf.spct`, `Ler_leaf_rflt.spct`, `Ler_leaf_trns.spct`, `Ler_leaf_trns_i.spct`, `black_body.spct`, `ccd.spct`, `clear.spct`, `clear_body.spct`, `filter_cps.mspct`, `green_leaf.spct`, `opaque.spct`, `photodiode.spct`, `polyester.spct`, `sun.daily.data`, `sun.daily.spct`, `sun.data`, `sun.spct`, `white_body.spct`, `white_led.raw_spct`, `white_led.source_spct`, `yellow_gel.spct`

Examples

```
white_led.cps_spct
```

white_led.raw_spct *White led bulb spectrum*

Description

A dataset containing wavelengths and the corresponding spectral data as raw instrument counts for an Osram warm white led lamp, for three different integration times:

Usage

```
white_led.raw_spct
```

Format

An object of class raw_spct (inherits from generic_spct, tbl_df, tbl, data.frame) with 2068 rows and 4 columns.

Details

- w.length (nm), range 188 to 1117 nm.
- counts_1
- counts_2
- counts_3
- w.length (nm), range 188 to 1117 nm.
- cps

See Also

Other Spectral data examples: A.illuminant.spct, D65.illuminant.spct, Ler_leaf.spct, Ler_leaf_rflt.spct, Ler_leaf_trns.spct, Ler_leaf_trns_i.spct, black_body.spct, ccd.spct, clear.spct, clear_body.spct, filter_cps.mspct, green_leaf.spct, opaque.spct, photodiode.spct, polyester.spct, sun.daily.data, sun.daily.spct, sun.data, sun.spct, white_body.spct, white_led.cps_spct, white_led.source_spct, yellow_gel.spct

Examples

```
white_led.raw_spct
```

---

white_led.source_spct *White led bulb spectrum*

Description

A dataset containing wavelengths and the corresponding spectral irradiance data for an Osram warm white led lamp:

Usage

```
white_led.source_spct
```

Format

A source_spct object with 1421 rows and 2 variables

Details

- w.length (nm), range 250 to 900 nm.
- s.e.irrad (W m-2 nm-1)

See Also

Other Spectral data examples: A.illuminant.spct, D65.illuminant.spct, Ler_leaf.spct, Ler_leaf_rflt.spct, Ler_leaf_trns.spct, Ler_leaf_trns_i.spct, black_body.spct, ccd.spct, clear.spct, clear_body.spct, filter_cps.mspct, green_leaf.spct, opaque.spct, photodiode.spct, polyester.spct, sun.daily.data, sun.daily.spct, sun.data, sun.spct, white_body.spct, white_led.cps_spct, white_led.raw_spct, yellow_gel.spct

Examples

```
white_led.source_spct
```

---

wls_at_target *Find wavelengths values corresponding to a target spectral value*

Description

Find wavelength values corresponding to a target spectral value in a spectrum. The name of the column of the spectral data to be used is inferred from the class of `x` and the argument passed to `unit.out` or `filter.qty` or their defaults that depend on R options set.

Usage

```
wls_at_target(x, target = NULL, interpolate = FALSE, na.rm = FALSE,
  ...)

## Default S3 method:
wls_at_target(x, target = NULL, interpolate = FALSE,
  na.rm = FALSE, ...)

## S3 method for class 'generic_spct'
wls_at_target(x, target = "half.maximum",
  interpolate = FALSE, na.rm = FALSE, col.name = NULL, ...)

## S3 method for class 'source_spct'
wls_at_target(x, target = "half.maximum",
  interpolate = FALSE, na.rm = FALSE,
  unit.out = getOption("photobiology.radiation.unit", default =
  "energy"), ...)

## S3 method for class 'response_spct'
wls_at_target(x, target = "half.maximum",
  interpolate = FALSE, na.rm = FALSE,
  unit.out = getOption("photobiology.radiation.unit", default =
  "energy"), ...)

## S3 method for class 'filter_spct'
```

```
wls_at_target(x, target = "half.maximum",
  interpolate = FALSE, na.rm = FALSE,
  filter.qty = getOption("photobiology.filter.qty", default =
  "transmittance"), ...)

## S3 method for class 'reflector_spct'
wls_at_target(x, target = "half.maximum",
  interpolate = FALSE, na.rm = FALSE, ...)

## S3 method for class 'cps_spct'
wls_at_target(x, target = "half.maximum",
  interpolate = FALSE, na.rm = FALSE, ...)

## S3 method for class 'generic_mspct'
wls_at_target(x, target = "half.maximum",
  interpolate = FALSE, na.rm = FALSE, ..., .parallel = FALSE,
  .paropts = NULL)
```

Arguments

x	data.frame or spectrum object.
target	numeric value indicating the spectral quantity value for which wavelengths are to be searched and interpolated if need. The character string "half.maximum" is also accepted as argument.
interpolate	logical Indicating whether the nearest wavelength value in x should be returned or a value calculated by linear interpolation between wavelength values stradling the target.
na.rm	logical indicating whether NA values should be stripped before searching for the target.
...	currently ignored.
col.name	character The name of the column in which to search for the target value.
unit.out	character One of "energy" or "photon"
filter.qty	character One of "transmittance" or "absorbance"
.parallel	if TRUE, apply function in parallel, using parallel backend provided by foreach
.paropts	a list of additional options passed into the foreach function when parallel computation is enabled. This is important if (for example) your code relies on external data or packages: use the .export and .packages arguments to supply them so that all cluster nodes have the correct environment set up for computing.

Value

A spectrum object of the same class as x with fewer rows, possibly even no rows. If FALSE is passed to interpolate a subset of x is returned, otherwise a new object of the same class containing interpolated wavelenths for the target value is returned.

Methods (by class)

- default: Default returning always an empty object of the same class as x.
- generic_spct: Method for "generic_spct" objects.
- source_spct: Method for "source_spct" objects.
- response_spct: Method for "response_spct" objects.
- filter_spct: Method for "filter_spct" objects.
- reflector_spct: Method for "reflector_spct" objects.
- cps_spct: Method for "cps_spct" objects.
- generic_mspct: Method for "generic_mspct" objects.

Note

When interpolation is used, only column w.length and the column against which the target value was compared are included in the returned object, otherwise, all columns in x are returned. We implement support for data.frame to simplify the coding of 'ggplot2' stats using this function.

See Also

Other peaks and valleys functions: find_peaks, get_peaks, peaks, valleys

Examples

```
wls_at_target(sun.spct, target = 0.1)
```

---

wl_max *Wavelength maximum*

Description

A method specialization that returns the wavelength maximum from objects of classes "waveband" or of class "generic_spct" or derived.

Usage

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

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

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

## S3 method for class 'generic_mspct'
max(..., na.rm = FALSE, idx = "spct.idx")
```

Arguments

x	generic_spct, generic_mspct or waveband object.
na.rm	ignored
...	not used in current version
idx	character Name of the column with the names of the members of the collection of spectra.

Methods (by class)

- generic_spct:
- generic_mspct:

Examples

```
max(sun.spct)
wl_max(sun.spct)
```

---

wl_midpoint *Midpoint*

Description

A function that returns the wavelength (or value) at the center of the of the wavelength range of a waveband or spectrum object (or numeric vector).

Usage

```
wl_midpoint(x, ...)

midpoint(x, ...)

## Default S3 method:
midpoint(x, ...)

## S3 method for class 'numeric'
midpoint(x, ...)

## S3 method for class 'waveband'
midpoint(x, ...)

## S3 method for class 'generic_spct'
midpoint(x, ...)

## S3 method for class 'generic_mspct'
midpoint(x, ..., idx = "spct.idx")
```

Arguments

x	an R object
...	not used in current version
idx	character Name of the column with the names of the members of the collection of spectra.

Value

A numeric value equal to (max(x) - min(x)) / 2. In the case of spectral objects a wavelength in nm. For any other R object, according to available definitions of `min` and `max`.

Methods (by class)

- `default`: Default method for generic function
- `numeric`: Default method for generic function
- `waveband`: Wavelength at center of a "waveband".
- `generic_spct`: Method for "generic_spct".
- `generic_mspct`: Method for "generic_mspct" objects.

See Also

Other wavelength summaries: `wl_min`, `wl_range`, `wl_stepsize`

Other wavelength summaries: `wl_min`, `wl_range`, `wl_stepsize`

Other wavelength summaries: `wl_min`, `wl_range`, `wl_stepsize`

Examples

```
midpoint(10:20)
midpoint(sun.spct)
wl_midpoint(sun.spct)

midpoint(sun.spct)
```

---

wl_min *Wavelength minimum*

Description

A method specialization that returns the wavelength minimum from objects of classes "waveband" or of class "generic_spct" or derived.

Usage

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

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

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

## S3 method for class 'generic_mspct'
min(..., na.rm = FALSE, idx = "spct.idx")
```

Arguments

x	generic_spct, generic_mspct or waveband object.
na.rm	ignored
...	not used in current version
idx	character Name of the column with the names of the members of the collection of spectra.

Methods (by class)

- generic_spct:
- generic_mspct:

See Also

Other wavelength summaries: wl_midpoint, wl_range, wl_stepsize

Examples

```
min(sun.spct)
wl_min(sun.spct)
```

---

wl_range *Wavelength range*

Description

A method specialization that returns the wavelength range from objects of classes "waveband" or of class "generic_spct" or derived.

Usage

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

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

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

## S3 method for class 'generic_mspct'
range(..., na.rm = FALSE, idx = "spct.idx")
```

Arguments

x	generic_spct, generic_mspct or waveband object.
na.rm	ignored
...	a single R object
idx	character Name of the column with the names of the members of the collection of spectra.

Methods (by class)

- generic_spct:
- generic_mspct:

See Also

Other wavelength summaries: wl_midpoint, wl_min, wl_stepsize

Examples

```
range(sun.spct)
wl_range(sun.spct)

range(sun.spct)
```

---

wl_stepsize *Stepsize*

Description

Function that returns the range of step sizes in an object. Range of differences between successive sorted values.

Usage

```
wl_stepsize(x, ...)

stepsize(x, ...)

## Default S3 method:
stepsize(x, ...)

## S3 method for class 'numeric'
stepsize(x, ...)

## S3 method for class 'generic_spct'
stepsize(x, ...)

## S3 method for class 'generic_mspct'
stepsize(x, ..., idx = "spct.idx")
```

Arguments

x	an R object
...	not used in current version
idx	character Name of the column with the names of the members of the collection of spectra.

Value

A numeric vector of length 2 with min and maximum stepsize values.

Methods (by class)

- `default`: Default function usable on numeric vectors.
- `numeric`: Method for numeric vectors.
- `generic_spct`: Method for "generic_spct" objects.
- `generic_mspct`: Method for "generic_mspct" objects.

See Also

Other wavelength summaries: `wl_midpoint`, `wl_min`, `wl_range`

Examples

```
stepsize(sun.spct)
wl_stepsize(sun.spct)

stepsize(sun.spct)
```

w_length2rgb *Wavelength to rgb color conversion*

Description

Calculates rgb values from spectra based on human color matching functions

Usage

```
w_length2rgb(w.length, sens = photobiology::ciexyzCMF2.spct,
  color.name = NULL)
```

Arguments

w.length	numeric Vector of wavelengths (nm)
sens	chroma_spct Used as chromaticity definition
color.name	character Used for naming the rgb color definition

Value

A vector of colors defined using `rgb()`. The numeric values of the RGB components can be obtained using function `col2rgb()`.

See Also

Other color functions: `rgb_spct`, `w_length_range2rgb`

Examples

```
col2rgb(w_length2rgb(580))
col2rgb(w_length2rgb(c(400, 500, 600, 700)))
col2rgb(w_length2rgb(c(400, 500, 600, 700), color.name=c("a","b","c","d")))
col2rgb(w_length2rgb(c(400, 500, 600, 700), color.name="a"))
```

w_length_range2rgb *Wavelength range to rgb color conversion*

Description

Calculates rgb values from spectra based on human color matching functions

Usage

```
w_length_range2rgb(w.length, sens = photobiology::ciexyzCMF2.spct,
  color.name = NULL)
```

Arguments

w.length	numeric vector of wavelengths (nm) of length 2. If longer, its range is used.
sens	chroma_spct Used as the chromaticity definition.
color.name	character Used for naming the rgb color definition(s) returned.

Value

A vector of colors defined using `rgb()`. The numeric values of the RGB components can be obtained by calling function `col2rgb`.

See Also

Other color functions: `rgb_spct`, `w_length2rgb`

Examples

```
col2rgb(w_length_range2rgb(c(500,600)))
col2rgb(w_length_range2rgb(550))
col2rgb(w_length_range2rgb(500:600))
```

---

yellow_gel.spct *Transmittance spectrum of yellow theatrical gel.*

Description

A dataset containing the wavelengths at a 1 nm interval and fractional total transmittance for polyester film.

Usage

```
yellow_gel.spct
```

Format

A `filter_spct` object with 611 rows and 2 variables

Details

- w.length (nm).
- Tfr (0..1)

See Also

Other Spectral data examples: `A.illuminant.spct`, `D65.illuminant.spct`, `Ler_leaf.spct`, `Ler_leaf_rflt.spct`, `Ler_leaf_trns.spct`, `Ler_leaf_trns_i.spct`, `black_body.spct`, `ccd.spct`, `clear.spct`, `clear_body.spct`, `filter_cps.mspct`, `green_leaf.spct`, `opaque.spct`, `photodiode.spct`, `polyester.spct`, `sun.daily.data`, `sun.daily.spct`, `sun.data`, `sun.spct`, `white_body.spct`, `white_led.cps_spct`, `white_led.raw_spct`, `white_led.source_spct`

Examples

```
yellow_gel.spct
```

`^.generic_spct` *Arithmetic Operators*

Description

Power operator for spectra.

Usage

```
## S3 method for class 'generic_spct'
e1 ^ e2
```

Arguments

e1	an object of class "generic_spct"
e2	a numeric vector. possibly of length one.

See Also

Other math operators and functions: MathFun, convolve_each, div-.generic_spct, log, minus-.generic_spct, mod-.generic_spct, plus-.generic_spct, round, sign, slash-.generic_spct, times-.generic_spct

Index