Package 'photobiology'

June 25, 2017

Type Package

Title Photobiological Calculations

Version 0.9.16

Date 2017-06-25

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.

License GPL (>= 2)

Depends R (>= 3.3.0)

Imports stats, polynom (>= 1.3-9), tibble (>= 1.3.3), lubridate (>= 1.6.0), plyr (>= 1.8.4), dplyr (>= 0.7.1), splus2R (>= 1.2-2), caTools (>= 1.17.1)

Suggests knitr (>= 1.16), rmarkdown (>= 1.6), testthat (>= 1.0.2), roxygen2 (>= 6.0.1), ggmap (>= 2.6.1)

LazyLoad yes

LazyData yes

ByteCompile true

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

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

Encoding UTF-8

RoxygenNote 6.0.1

VignetteBuilder knitr

NeedsCompilation no

Author Pedro J. Aphalo [aut, cre],
Titta K. Kotilainen [ctb]

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

Repository CRAN

Date/Publication 2017-06-25 15:40:12 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.

Details

Package 'photobiology' is at the core of a suite of packages for analysis and plotting of data relevant to photobiology (described at http://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 reponses. 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 contributins 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>

Other contributors:

- Titta K. Kotilainen [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. xxx + 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 http://hdl.handle.net/10138/37558

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. doi: 10.1111/2041-210X.12069

See Also

Useful links:

- http://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, 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 coverts absorbance 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, ...)
```

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

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: T2A, as_energy, as_quantum_mol, 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), ...,
  idx = !is.null(names(spct)))

## 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), ...,
  idx = !is.null(names(spct)))
```

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 use hinges to reduce interpolation errors
...	other arguments (possibly ignored)
idx	logical whether to add a column with the names of the elements of spct

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), ..., idx = !is.null(names(spct)))

## 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), ..., idx = !is.null(names(spct)))
```

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 if wavebands crossing spectral data boundaries are trimmed or ignored
use.hinges	logical Flag indicating whether to use hinges to reduce interpolation errors
...	other arguments (possibly ignored)
idx	logical whether to add a column with the names of the elements of spct

Value

A single numeric value with no change in scale factor, except in the case of percentages (absorptance is the fraction absorbed)

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")
```

`as.generic_mspct` *Collection-of-spectra copy-constructor*

Description

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

Usage

```
as.generic_mspct(x, force.spct.class = FALSE)

as.raw_mspct(x)

as.cps_mspct(x)

as.source_mspct(x, time.unit = c("second", "day", "exposure"),
  bswf.used = c("none", "unknown"),
  strict.range = getOption("photobiology.strict.range", default = FALSE))

as.response_mspct(x, time.unit = "second")

as.filter_mspct(x, Tfr.type = c("total", "internal"), strict.range = TRUE)

as.reflector_mspct(x, Rfr.type = c("total", "specular"),
  strict.range = TRUE)

as.object_mspct(x, Tfr.type = c("total", "internal"), Rfr.type = c("total",
  "specular"), strict.range = TRUE)

as.chroma_mspct(x)
```

Arguments

`x`	a list of spectral objects or a list of objects such as data frames that can be converted into spectral objects.
`force.spct.class`	logical indicating whether to change the class of members to `generic_spct` or retain the existing class.
`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
`Tfr.type`	a character string, either "total" or "internal"
`Rfr.type`	a character string, either "total" or "specular"

Value

These functions return a copy of x converted into a given class of spectral collection object, if x is a valid argument to the corresponding set function.

Note

Members of generic_mspct objects can be heterogeneous: they can belong any class derived from generic_spct and class is not enforced. In this case when x is a list of data frames, force.spct.class = TRUE needs to be supplied.

See Also

Other creation of spectral objects functions: as.generic_spct, source_spct

---

as.generic_spct *Spectral-object copy constructor*

Description

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

Usage

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

as.raw_spct(x, ...)

as.cps_spct(x, ...)

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

as.response_spct(x, time.unit = "second", ...)

as.filter_spct(x, Tfr.type = c("total", "internal"),
  strict.range = getOption("photobiology.strict.range", default = FALSE), ...)

as.reflector_spct(x, Rfr.type = c("total", "specular"),
  strict.range = getOption("photobiology.strict.range", default = FALSE), ...)

as.object_spct(x, Tfr.type = c("total", "internal"), Rfr.type = c("total",
  "specular"), strict.range = getOption("photobiology.strict.range", default =
  FALSE), ...)

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

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
Tfr.type	a character string, either "total" or "internal"
Rfr.type	a character string, either "total" or "specular"

Value

These functions return a copy of x converted into a given class of spectral object, if x is a valid argument to the correcponding set function.

See Also

Other creation of spectral objects functions: as.generic_mspct, 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 formating and printing.

as_energy *Convert spectral photon irradiance into spectral energy irradiance*

Description

For example an spectrum [mol s-1 m-2 nm-1] is converted into a spectrum [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 Corresponding vector of spectral photon irradiances

Value

A numeric vector of spectral (energy) irradiances

See Also

Other quantity conversion functions: A2T, T2A, as_quantum_mol, as_quantum, e2qmol_multipliers, e2quantum_multipliers, e2q, q2e

Examples

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

as_quantum *Convert spectral energy irradiance into spectral photon irradiance*

Description

For example an spectrum [W m-2 nm-1] is converted into a spectrum [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 Corresponding vector of spectral (energy) irradiances

Value

A numeric array of spectral photon irradiances

See Also

Other quantity conversion functions: A2T, T2A, as_energy, as_quantum_mol, 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

For example an spectrum [W m-2 nm-1] is converted into a spectrum [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 Corresponding vector of spectral (energy) irradiances

Value

a numeric array of spectral photon irradiances

See Also

Other quantity conversion functions: A2T, T2A, as_energy, as_quantum, e2qmol_multipliers, e2quantum_multipliers, e2q, q2e

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	charater 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 wavelenght.

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 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, 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.generic_mspct *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
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 array of multipliers of the same length as w.length

Examples

```
with(sun.data, calc_multipliers(w.length, new_waveband(400,700),"photon"))
```

---

calc_source_output *Light-source spectral output*

Description

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

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 arrays are frequently used as sensors in high-UV-sensitivity array 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/resources/pdf/ssd/s11071-1004_etc_kmpd1120e.pdf. Visited 2016-02-01.

See Also

Other Spectral data examples: A.illuminant.spct, D65.illuminant.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

---

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 '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 warning, 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.
- raw_spct: Specialization for cps_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)
check_spct(-sun.spct)
try(check_spct((sun.spct[1, "w.length"] <- 1000)))
```

---

check_spectrum *Sanity check of a spectrum.*

Description

This function checks a spectral radiation 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 '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"), ...)

## 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"), ...)

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

## 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"), ...)

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

## S3 method for class 'raw_mspct'
```

```
clean(x, range = NULL, range.s.data = c(0, NA),
  fill = range.s.data, ...)

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

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

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
- 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 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`, `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, 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

Clipping of head and tail of a spectrum based on wavelength limits, no interpolation used.

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 walengths expressed in nanometres.
...	not used

Value

an R object of same class as input, 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_tails`, `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

A function that returns the equivalent RGB color of an object such as a spectrum or wavelength.

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 = !is.null(names(x)))

colour_of(x, ...)

color(x, ...)
```

Arguments

x	an R object
...	not used in current version
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	logical whether to add a column with the names of the elements of spct

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. 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 = `*`, ...)
```

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 necesarily an operator with two arguments.
...	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 from one R object to another*

Description

Copy attributes from x to y.

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,
  copy.class = FALSE, ...)
```

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

Arguments

x, y	R objects
which	character
...	not used
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:
- 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 converison.
...	Additional arguments passed to pre.fun.
dyn.range	numeric The effective dynamic range of the instrument, if NULL it is automatically set based on integartion 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 deuterim 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 calibartion 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`, `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 by means of function sun_angles, 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 = data.frame(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 = Sys.timezone(),
  geocode = data.frame(lon = 0, lat = 51.5, address = "Greenwich"),
  twilight = "none", unit.out = "datetime")

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

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

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

night_length(date = lubridate::now(), tz = "UTC", geocode = data.frame(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
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", "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` charater string, One of "datetime", "day", "hour", "minute", or "second".

Details

Twilight names are interpreted as follows. "none": solar elevation = 0 degrees. "refraction": solar elevation = 0 degrees + refraction correction. "sunlight": upper rim of solar disk corrected for refraction. "civil": -6 degrees, "naval": -12 degrees, and "astronomical": -18 degrees. Unit names for output are as follows: "hours" times for sunrise and sunset are returned as times-of-day in hours since middnight. "date" or "datetime" return the same times as datetime objects with TZ set (this is much slower than the "hours"). Day length and night length are returned as numeric values expressed in hours when '"datetime"' is paseed 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 data.frame with variables day, tz, twilight.rise, twilight.set, longitude, latitude, address, sunrise, noon, sunset, daylength, nightlength.

`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 as 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 calcualtion into two separate functions, one returning angles at given instants in time, and a separate one returning the timing of events for given dates.

`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 consequtive days.

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_mscpt has been renamed msdply().

Function mutate_mscpt 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_mscpt 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 added.

Usage

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

Arguments

`w.length1`	numeric array of wavelength (nm) of denominator
`w.length2`	numeric array of wavelength (nm) of divisor
`s.irrad1`	a numeric array of spectral values of denominator
`s.irrad2`	a numeric array 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-overlaping 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.

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 coverts 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, ...)

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

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

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 for collections of response spectra

See Also

Other quantity conversion functions: A2T, T2A, as_energy, as_quantum_mol, as_quantum, e2qmol_multipliers, e2quantum_multipliers, q2e

e2qmol_multipliers *Calculate energy to quantum (mol) multipliers*

Description

Gives 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 array of multipliers

See Also

Other quantity conversion functions: A2T, T2A, as_energy, as_quantum_mol, 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 array of multipliers

See Also

Other quantity conversion functions: A2T, T2A, as_energy, as_quantum_mol, as_quantum, e2qmol_multipliers, e2q, q2e

Examples

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

---

energy_irradiance *Calculate (energy) irradiance from spectral irradiance*

Description

This function gives the energy irradiance for a given waveband of a radiation spectrum, optionally applies a 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 array of wavelength (nm)
s.irrad	numeric array 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 use hinges to reduce interpolation errors

Value

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

See Also

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

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

This function gives the energy ratio between two given wavebands of 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 Corresponding of spectral (energy) irradiances (W m-2 nm-1)
`w.band.num`	waveband
`w.band.denom`	waveband
`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 use hinges to reduce interpolation errors

Value

a single numeric value giving the unitless ratio

See Also

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

Examples

```
# energy:energy ratio
with(sun.data,
     energy_ratio(w.length, s.e.irrad, new_waveband(400,500), new_waveband(400,700)))
# energy:energy ratio waveband : whole spectrum
with(sun.data, energy_ratio(w.length, s.e.irrad, new_waveband(400,500)))
# energy:energy ratio of whole spectrum should be equal to 1.0
with(sun.data, energy_ratio(w.length, s.e.irrad))
```

`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"), ...,
  idx = !is.null(names(spct)))
```

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 telling whether to use hinges to reduce interpolation errors
...	other arguments (possibly ignored)
idx	logical whether to add a column with the names of the elements of spct

Value

a numeric value giving number of Joule per mol of photons for each waveband, with name attribute set to the name of each waveband unless a named list of wavebands is supplied in which case the names of the list elements are used, with "e:q" prepended..

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 mosts 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, energy_ratio, photon_ratio, photons_energy_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 extrated 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.

See Also

`subset.data.frame` 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

This function returns the energy fluence for a given waveband of a light source spectrum given the duration of the exposure.

Usage

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

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

```
## S3 method for class 'source_spct'
e_fluence(spct, w.band = NULL, exposure.time,
  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,
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.cached.mult = getOption("photobiology.use.cached.mult", default =
  FALSE), use.hinges = NULL, allow.scaled = FALSE, ...,
  idx = !is.null(names(spct)))
```

Arguments

spct	an R object
w.band	a list of waveband objects or a waveband object
exposure.time	lubridate::duration
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 indicating whether to use hinges to reduce interpolation errors
allow.scaled	logical indicating whether scaled or normalized spectra as argument to spct are flagged as an error
...	other arguments (possibly ignored)
idx	logical whether to add a column with the names of the elements of spct

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 mosts 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`, `energy_irradiance`, `fluence`, `irradiance`, `irrad`, `photon_irradiance`, `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

This function returns the energy irradiance for a given waveband of a light source spectrum.

Usage

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

## Default S3 method:
e_irrad(spct, w.band, quantity, time.unit, 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, 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, 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"), ..., idx = !is.null(names(spct)))
```

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
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 indicating whether to use hinges to reduce interpolation errors
allow.scaled	logical indicating whether scaled or normalized spectra as argument to spct are flagged as an error
...	other arguments (possibly ignored)
idx	logical whether to add a column with the names of the elements of spct

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 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 mosts 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, energy_irradiance, fluence, irradiance, irrad, photon_irradiance, 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"), ...,
  idx = !is.null(names(spct)))
```

Arguments

spct	asource_spct
w.band.num	waveband or list of waveband objects
w.band.denom	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 telling whether to use hinges to reduce interpolation errors
...	other arguments (possibly ignored)
idx	logical whether to add a column with the names of the elements of spct

Value

A single numeric nondimensional value giving an energy ratio between pairs of wavebands, with name attribute set to the name of each waveband unless a named list of wavebands is supplied in which case the names of the list elements are used, with "(e:e)" appended.

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 wavebans takes place when the number of denominator and denominator wavebands differ. The last two parameters control speed optimizations. The defaults should be suitable in mosts 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: energy_ratio, eq_ratio, photon_ratio, photons_energy_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, wb.trim, use.hinges, ...)

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

## S3 method for class 'response_spct'
e_response(spct, w.band = NULL, quantity = "total",
  time.unit = NULL, 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, wb.trim = getOption("photobiology.waveband.trim",
  default = TRUE), use.hinges = getOption("photobiology.use.hinges", default =
  NULL), ..., idx = !is.null(names(spct)))
```

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
wb.trim	logical if TRUE wavebands crossing spectral data boundaries are trimmed, if FALSE, they are discarded
use.hinges	logical indicating whether to use hinges to reduce interpolation errors
...	other arguments
idx	logical whether to add a column with the names of the elements of spct

Value

A single numeric value expressed either as a fraction of one or a percentage, or a vector of the same length as the list of wave.bands. The quantity returned, although always on energy-based units, depends on the value of `quantity`.

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 mosts 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(200)
FEL_spectrum(170:220)
```

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)
```

Arguments

x	numeric array
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	ogical flag: if TRUE, an element must be strictly greater than all other values in its window to be considered a peak. Default: TRUE.

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

Examples

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

---

fluence *Fluence*

Description

This function returns the energy or photon fluence for a given waveband of a light source spectrum and the duration of the exposure.

Usage

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

## Default S3 method:
fluence(spct, w.band, unit.out, exposure.time, 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, 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, wb.trim = getOption("photobiology.waveband.trim", default =
  TRUE), use.cached.mult = getOption("photobiology.use.cached.mult", default =
  FALSE), use.hinges = NULL, allow.scaled = FALSE, ...,
  idx = !is.null(names(spct)))
```

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
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 indicating whether to use hinges to reduce interpolation errors
allow.scaled	logical indicating whether scaled or normalized spectra as argument to spct are flagged as an error
...	other arguments (possibly ignored)
idx	logical whether to add a column with the names of the elements of spct

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 mosts 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`, `energy_irradiance`, `irradiance`, `irrad`, `photon_irradiance`, `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

---

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"),
  ...)

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

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

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

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

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

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

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

Arguments

x	An R object
...	additonal 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 annd max wavelengths (nm)
f	character string "mean" or "total" for scaling so taht 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 Alowed 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.

Value

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

---

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 shift scale.

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"), ...)

## 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"),
  ...)
```

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

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

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

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

Arguments

x	An R object
...	additonal 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 annd 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.

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 attibute 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))

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 Array of dimensions
...	ignored

Functions

- 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

See Also

Other collections of spectra classes family: split2mspct, subset2mspct

Examples

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

See Also

Other BSWF attribute functions: setBSWFUsed

Examples

```
getBSWFUsed(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: getInstrSettings, getWhatMeasured, getWhenMeasured, getWhereMeasured, isValidInstrDesc, isValidInstrSettings, 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: `getInstrDesc`, `getWhatMeasured`, `getWhenMeasured`, `getWhereMeasured`, `isValidInstrDesc`, `isValidInstrSettings`, `setInstrDesc`, `setInstrSettings`, `setWhatMeasured`, `setWhenMeasured`, `setWhereMeasured`, `trimInstrDesc`, `trimInstrSettings`

getMspctVersion *Get the "mspct.version" attribute*

Description

Funtion 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

Funtion 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`

getScaled *Get the "scaled" attribute*

Description

Funtion 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`

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

numeric 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 object.

Usage

```
getWhatMeasured(x)
```

Arguments

x	a generic_spct object

Value

list

See Also

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

---

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 = !is.null(names(x)))
```

Arguments

x	a generic_spct object
...	Allows use of additional arguments in methods for other classes.
idx	logical whether to add a column with the names of the elements of spct

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: getInstrDesc, getInstrSettings, getWhatMeasured, getWhereMeasured, isValidInstrDesc, isValidInstrSettings, 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 = !is.null(names(x)))
```

Arguments

x	a generic_spct object
...	Allows use of additional arguments in methods for other classes.
idx	logical whether to add a column with the names of the elements of spct

Value

a data.frane 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: getInstrDesc, getInstrSettings, getWhatMeasured, getWhenMeasured, isValidInstrDesc, isValidInstrSettings, setInstrDesc, setInstrSettings, setWhatMeasured, setWhenMeasured, setWhereMeasured, trimInstrDesc, trimInstrSettings

Examples

```
getWhereMeasured(sun.spct)
```

---

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)

get_valleys(x, y, ignore_threshold = 0, span = 5, strict = TRUE,
  x_unit = "", x_digits = 3)
```

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.

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

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, 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
```

---

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 array (sorted in increasing order)
y	numeric array
h	a numeric array giving the wavelengths at which the y values should be inserted by interpolation, no interpolation is indicated by an empty array (numeric(0))

Value

a data.frame with variables x and y

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.

Examples

```
with(sun.data,
    insert_hinges(w.length, s.e.irrad,
       c(399.99, 400.00, 699.99, 700.00)))
with(sun.data,
    insert_hinges(w.length, s.e.irrad,
      c(100, 399.50, 399.99, 400.00, 699.99, 700.00, 799.99, 1000)))
```

---

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 array (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 wavelenght.

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 array
y	numeric array

Value

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

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)
```

Arguments

spct	generic_spct
w.length.out	numeric array 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"

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 extrpolated 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

For example interpolate spectral irradiance (or spectral transmittance) values at new wavelengths values.

Usage

```
interpolate_spectrum(w.length.in, s.irrad, w.length.out, fill = NA)
```

Arguments

`w.length.in`	numeric array of wavelengths (nm)
`s.irrad`	a numeric array of spectral values
`w.length.out`	numeric array of wavelengths (nm)
`fill`	a value to be assigned to out of range wavelengths

Value

a numeric array 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.

Examples

```
my.w.length <- 300:700
my.s.e.irrad <-
  with(sun.data, interpolate_spectrum(w.length, s.e.irrad, my.w.length))
plot(my.s.e.irrad ~ my.w.length)
lines(s.e.irrad ~ w.length, data=sun.data)
```

---

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, ...)
```

Arguments

x	an R object
w.length.out	numeric array of wavelengths (nm)
fill	a value to be assigned to out of range wavelengths
length.out	numeric value
...	not used

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 extrpolated 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, wb.trim, use.cached.mult,
  use.hinges, allow.scaled, ...)

## Default S3 method:
irrad(spct, w.band, unit.out, quantity, time.unit, 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,
  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,
  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"), ..., idx = !is.null(names(spct)))
```

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
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 indicating whether to use hinges to reduce interpolation errors
allow.scaled	logical indicating whether scaled or normalized spectra as argument to spct are flagged as an error
...	other arguments (possibly ignored)
idx	logical whether to add a column with the names of the elements of spct

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] 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 mosts 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`, `energy_irradiance`, `fluence`, `irradiance`, `photon_irradiance`, `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 (quantum) irradiance from spectral (energy) irradiance.*

Description

This function returns the energy irradiance for a given waveband 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 Corresponding 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 use hinges to reduce interpolation errors

Value

a single numeric value 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 mosts 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 irradiance functions: `e_fluence`, `e_irrad`, `energy_irradiance`, `fluence`, `irrad`, `photon_irradiance`, `q_fluence`, `q_irrad`

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.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.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.any_spct(sun.spct)

is.source_spct(sun.spct)
is.filter_spct(sun.spct)
is.generic_spct(sun.spct)
is.any_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: getInstrDesc, getInstrSettings, getWhatMeasured, getWhenMeasured, getWhereMeasured, isValidInstrSettings, 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: getInstrDesc, getInstrSettings, getWhatMeasured, getWhenMeasured, getWhereMeasured, isValidInstrDesc, 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 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 quering 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 calcualtaing 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 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 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 rescled 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 spectrun, 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)
```

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)
```

---

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 appled 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 appled 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

---

max *Wavelength maximum*

Description

A function that returns the wavelength maximum from objects of class "waveband".

Usage

```
## 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 = NULL)
```

Arguments

...	not used in current version
na.rm	ignored
idx	logical whether to add a column with the names of the elements of spct

Methods (by class)

- generic_spct:
- generic_mspct:

Examples

```
max(sun.spct)
```

---

merge.generic_spct *Merge two generic_spct objects*

Description

Merge of two spct objects based on w.length.

Usage

```
## S3 method for class 'generic_spct'
merge(x, y, by = "w.length", ...)
```

Arguments

x	generic_spct (or derived) objects to be merged
y	generic_spct (or derived) objects to be merged
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()

Note

If the class of x and y is the same, it is preserved, but if it differs `generic_spct` is used for the returned value, except when x and y, are one each of classes reflector_spct and filter_spct in which case an object_spct is returned. In the current implementation only wavelengths values shared by x and y are preserved.

See Also

`join`

`midpoint` *Central wavelength value*

Description

A function that returns the wavelength at the center of the wavelength range.

Usage

```
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 = !is.null(names(x)))
```

Arguments

`x`	an R object
`...`	not used in current version
`idx`	logical whether to add a column with the names of the elements of spct

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: min, range, stepsize

Other wavelength summaries: min, range, stepsize

Other wavelength summaries: min, range, stepsize

Examples

```
midpoint(sun.spct)
```

---

min *Wavelength minimum*

Description

A function that returns the wavelength minimum.

Usage

```
## 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 = NULL)
```

Arguments

...	not used in current version
na.rm	ignored
idx	logical whether to add a column with the names of the elements of spct

Methods (by class)

- generic_spct:
- generic_mspct:

See Also

Other wavelength summaries: midpoint, range, stepsize

Examples

```
min(sun.spct)
```

---

minus-.generic_spct *Arithmetic Operators*

Description

Substraction 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, ...)

msdply(mspct, .fun, ..., idx = NULL, col.names = NULL)

mslply(mspct, .fun, ...)

msaply(mspct, .fun, ..., .drop = TRUE)
```

Arguments

`mspct`	an object of class generic_mspct or a derived class
`.fun`	a function
`...`	other arguments passed to .fun
`idx`	logical whether to add a column with the names of the elements of mspct, if `NULL`, the default, a column is added only if all members of `mscpt` are named.
`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 array 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.source_spct *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 'source_spct'
na.omit(object, na.action = "omit", ...)

## S3 method for class 'response_spct'
na.omit(object, na.action = "omit", ...)

## S3 method for class 'filter_spct'
na.omit(object, na.action = "omit", ...)

## S3 method for class 'reflector_spct'
na.omit(object, na.action = "omit", ...)

## S3 method for class 'cps_spct'
na.omit(object, na.action = "omit", ...)
```

```
## S3 method for class 'raw_spct'
na.omit(object, na.action = "omit", ...)

## S3 method for class 'chroma_spct'
na.omit(object, na.action = "omit", ...)

## S3 method for class 'generic_spct'
na.exclude(object, na.action = "exclude", ...)
```

Arguments

object	an R object
na.action	character One of "omit" or "exclude"
...	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 objetcs. R's definitions work as expected with no need to override them.

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
na.omit(my_sun.spct)
na.action(na.omit(my_sun.spct))
```

---

normalization *Normalization of an R object*

Description

Normalization wavelength of an R object, retireved 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"))

## S3 method for class 'response_spct'
normalize(x, ..., range = NULL, norm = "max",
  unit.out = getOption("photobiology.radiation.unit", default = "energy"))

## S3 method for class 'filter_spct'
normalize(x, ..., range = NULL, norm = "max",
```

```
  qty.out = getOption("photobiology.filter.qty", default = "transmittance"))

## S3 method for class 'reflector_spct'
normalize(x, ..., range = NULL, norm = "max",
  qty.out = NULL)

## S3 method for class 'raw_spct'
normalize(x, ..., range = NULL, norm = "max")

## S3 method for class 'cps_spct'
normalize(x, ..., range = NULL, norm = "max")

## S3 method for class 'generic_spct'
normalize(x, ..., range = NULL, norm = "max",
  col.names)

## S3 method for class 'source_mspct'
normalize(x, ..., range = NULL, norm = "max",
  unit.out = getOption("photobiology.radiation.unit", default = "energy"))

## S3 method for class 'response_mspct'
normalize(x, ..., range = NULL, norm = "max",
  unit.out = getOption("photobiology.radiation.unit", default = "energy"))

## S3 method for class 'filter_mspct'
normalize(x, ..., range = NULL, norm = "max",
  qty.out = getOption("photobiology.filter.qty", default = "transmittance"))

## S3 method for class 'reflector_mspct'
normalize(x, ..., range = x, norm = "max",
  qty.out = NULL)

## S3 method for class 'raw_mspct'
normalize(x, ..., range = x, norm = "max")

## S3 method for class 'cps_mspct'
normalize(x, ..., range = x, norm = "max")
```

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 annd max wavelengths (nm)
norm	numeric Normalization wavelength (nm) or character string "max", or "min" for normalization at the corresponding wavelngth, or "integral" or "mean" for rescaling by dividing by these values.
unit.out	character Allowed values "energy", and "photon", or its alias "quantum"

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.

Value

A new object of the same class as 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

Accepted values for norm vary depending on the class of x

See Also

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

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, 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 array of wavelength (nm)
w.length2	numeric array of wavelength (nm)
s.irrad1	a numeric array of spectral values
s.irrad2	a numeric array 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-overlaping 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.

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, ...)

## Default S3 method:
peaks(x, span, ignore_threshold, strict, ...)

## S3 method for class 'numeric'
```

```
peaks(x, span = 5, ignore_threshold, strict = TRUE, ...)

## S3 method for class 'generic_spct'
peaks(x, span, ignore_threshold, strict, ...)

## S3 method for class 'source_spct'
peaks(x, span = 5, ignore_threshold = 0,
  strict = TRUE, unit.out = getOption("photobiology.radiation.unit", default
  = "energy"), ...)

## S3 method for class 'response_spct'
peaks(x, span = 5, ignore_threshold = 0,
  strict = TRUE, unit.out = getOption("photobiology.radiation.unit", default
  = "energy"), ...)

## S3 method for class 'filter_spct'
peaks(x, span = 5, ignore_threshold = 0,
  strict = TRUE, filter.qty = getOption("photobiology.filter.qty", default =
  "transmittance"), ...)

## S3 method for class 'reflector_spct'
peaks(x, span = 5, ignore_threshold = 0,
  strict = TRUE, ...)

## S3 method for class 'cps_spct'
peaks(x, span = 5, ignore_threshold = 0, strict = TRUE,
  ...)

## S3 method for class 'generic_mspct'
peaks(x, span = 5, ignore_threshold = 0,
  strict = TRUE, ...)
```

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 talelst peakthreshold 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.
...	ignored
unit.out	character One of "energy" or "photon"
filter.qty	character One of "transmittance" or "absorbance"

Value

a subset of x with rows corresponding to local maxima.

Methods (by class)

- `default`: Default function usable on numeric vectors.
- `numeric`: Default function usable on numeric vectors.
- `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`

Examples

```
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/resources/pdf/ssd/g5645_etc_kgpd1004e.pdf. Visited 2016-02-01.

See Also

Other Spectral data examples: `A.illuminant.spct`, `D65.illuminant.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`

---

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 array of wavelength (nm)
`s.irrad`	numeric array of spectral (energy) irradiances (W m-2 nm-1)
`w.band`	waveband
`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 telling whether to use hinges to reduce interpolation errors

Value

A single numeric value giving the ratio moles-photons per Joule.

See Also

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

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 array of wavelength (nm)
`s.irrad`	numeric array 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 telling whether to use hinges to reduce interpolation errors

Value

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

See Also

Other irradiance functions: `e_fluence`, `e_irrad`, `energy_irradiance`, `fluence`, `irradiance`, `irrad`, `q_fluence`, `q_irrad`

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 array of wavelength (nm)
s.irrad	numeric array of spectral (energy) irradiances (W m-2 nm-1)
w.band.num	waveband
w.band.denom	waveband
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 telling whether to use hinges to reduce interpolation errors

Value

a single numeric value giving the unitless ratio

See Also

Other photon and energy ratio functions: e_ratio, energy_ratio, eq_ratio, photons_energy_ratio, q_ratio, qe_ratio

Examples

```
# photon:photon ratio
with(sun.data,
     photon_ratio(w.length, s.e.irrad, new_waveband(400,500), new_waveband(400,700)))
# photon:photon ratio waveband : whole spectrum
with(sun.data, photon_ratio(w.length, s.e.irrad, new_waveband(400,500)))
# photon:photon ratio of whole spectrum should be equal to 1.0
with(sun.data, photon_ratio(w.length, s.e.irrad))
```

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, 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 array of wavelength (nm)
w.length2	numeric array of wavelength (nm)
s.irrad1	a numeric array of spectral values
s.irrad2	a numeric array 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-overlaping 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.

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 coverts 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, ...)

## S3 method for class 'response_mspct'
q2e(x, action = "add", byref = FALSE, ...)
```

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

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`, `T2A`, `as_energy`, `as_quantum_mol`, `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"), ...,
  idx = !is.null(names(spct)))
```

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 telling whether to use hinges to reduce interpolation errors
...	other arguments (possibly ignored)
idx	logical whether to add a column with the names of the elements of spct

Value

A vector of numeric values giving number of moles of photons per Joule for each waveband, with name attribute set to the name of each waveband unless a named list of wavebands is supplied in which case the names of the list elements are used, with "q:e" prepended..

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 mosts 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, energy_ratio, eq_ratio, photon_ratio, photons_energy_ratio, q_ratio

Examples

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

---

q_fluence *Photon fluence*

Description

This function returns the photon irradiance (or quantum irradiance) for a given waveband of a light source spectrum.

Usage

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

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

## S3 method for class 'source_spct'
q_fluence(spct, w.band = NULL, exposure.time,
  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,
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.cached.mult = getOption("photobiology.use.cached.mult", default =
  FALSE), use.hinges = NULL, allow.scaled = FALSE, ...,
  idx = !is.null(names(spct)))
```

Arguments

spct	an R object
w.band	a list of waveband objects or a waveband object
exposure.time	lubridate::duration
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 indicating whether to use hinges to reduce interpolation errors
allow.scaled	logical indicating whether scaled or normalized spectra as argument to spct are flagged as an error
...	other arguments (possibly ignored)
idx	logical whether to add a column with the names of the elements of spct

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 attibute. 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 mosts 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, energy_irradiance, fluence, irradiance, irrad, photon_irradiance, 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

This function returns the photon irradiance (or quantum irradiance) for a given waveband of a light source spectrum.

Usage

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

## Default S3 method:
q_irrad(spct, w.band, quantity, time.unit, 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, 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, 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"), ..., idx = !is.null(names(spct)))
```

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
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 indicating whether to use hinges to reduce interpolation errors
allow.scaled	logical indicating whether scaled or normalized spectra as argument to spct are flagged as an error
...	other arguments (possibly ignored)
idx	logical whether to add a column with the names of the elements of spct

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: 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 mosts 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, energy_irradiance, fluence, irradiance, irrad, photon_irradiance, 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"), ...,
  idx = !is.null(names(spct)))
```

Arguments

spct	an object of class "source_spct"
w.band.num	waveband definition created with new_waveband()
w.band.denom	waveband definition created with new_waveband()
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 indicating whether to use hinges to reduce interpolation errors
...	other arguments (possibly ignored)
idx	logical whether to add a column with the names of the elements of spct

Value

a single numeric nondimensional value giving a photon ratio between 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.

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

Recycling for wavebans takes place when the number of denominator and denominator wavebands differ. The last two parameters control speed optimizations. The defaults should be suitable in mosts 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, energy_ratio, eq_ratio, photon_ratio, photons_energy_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, wb.trim, use.hinges, ...)

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

## S3 method for class 'response_spct'
q_response(spct, w.band = NULL, quantity = "total",
  time.unit = NULL, 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, wb.trim = getOption("photobiology.waveband.trim",
  default = TRUE), use.hinges = getOption("photobiology.use.hinges", default =
  NULL), ..., idx = !is.null(names(spct)))
```

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
wb.trim	logical if TRUE wavebands crossing spectral data boundaries are trimmed, if FALSE, they are discarded
use.hinges	logical indicating whether to use hinges to reduce interpolation errors
...	other arguments
idx	logical whether to add a column with the names of the elements of spct

Value

A single numeric value expressed either as a fraction of one or a percentage, or a vector of the same length as the list of wave.bands. The quantity returned, although always on photon-based units, depends on the value of `quantity`.

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 mosts 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)
```

---

range *Wavelength range*

Description

A function that returns the wavelength range.

Usage

```
## 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 = NULL)
```

Arguments

...	not used in current version
na.rm	ignored
idx	logical whether to add a column with the names of the elements of spct

Methods (by class)

- generic_spct:
- generic_mspct:

See Also

Other wavelength summaries: midpoint, min, stepsize

Examples

```
range(sun.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 accross the bound spectral objetcs, a warning is issued.

`dplyr::rbind_fill` is called internally and the result returned is the highest class in the inheritance hierachy 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), ..., idx = !is.null(names(spct)))

## 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), ..., idx = !is.null(names(spct)))
```

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 use hinges to reduce interpolation errors
...	other arguments
idx	logical whether to add a column with the names of the elements of spct

Value

A single numeric value with no change in scale factor

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)))
```

---

response *Integrated response*

Description

Calculate average photon- or energy-based photo-response.

Usage

```
response(spct, w.band, unit.out, quantity, time.unit, wb.trim, use.hinges, ...)

## Default S3 method:
response(spct, w.band, unit.out, quantity, time.unit, 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,
  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,
  wb.trim = getOption("photobiology.waveband.trim", default = TRUE),
  use.hinges = getOption("photobiology.use.hinges", default = NULL), ...,
  idx = !is.null(names(spct)))
```

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
wb.trim	logical Flag telling if wavebands crossing spectral data boundaries are trimmed or ignored
use.hinges	logical indicating whether to use hinges to reduce interpolation errors
...	other arguments
idx	logical whether to add a column with the names of the elements of spct

Value

A single numeric value expressed either as a fraction of one or a percentage, or a vector of the same length as the list of wave.bands. The quantity returned depends on the value of `quantity`. Whether it is expressed in energy-based or photon-based units depends on `unit.out`.

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 mosts 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: s_e_irrad2rgb, 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 attibutes "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 lis 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 attibutes "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 appled 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

---

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 attibute of a data.frame or an object of a derived class to "generic_spct".

Usage

```
setGenericSpct(x, multiple.wl = 1L)

setRawSpct(x, strict.range = getOption("photobiology.strict.range", default =
  FALSE), multiple.wl = 1L)

setCpsSpct(x, strict.range = getOption("photobiology.strict.range", default =
  FALSE), multiple.wl = 1L)

setFilterSpct(x, Tfr.type = c("total", "internal"),
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  multiple.wl = 1L)

setReflectorSpct(x, Rfr.type = c("total", "specular"),
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  multiple.wl = 1L)

setObjectSpct(x, Tfr.type = c("total", "internal"), Rfr.type = c("total",
  "specular"), strict.range = getOption("photobiology.strict.range", default =
  FALSE), multiple.wl = 1L)

setResponseSpct(x, time.unit = "second", multiple.wl = 1L)

setSourceSpct(x, time.unit = "second", bswf.used = c("none", "unknown"),
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  multiple.wl = 1L)

setChromaSpct(x, multiple.wl = 1L)
```

Arguments

x	data.frame, list or generic_spct and derived classes
multiple.wl	numeric Maximum number of repeated w.length entries with same value.
strict.range	logical Flag indicating whether off-range values result in an error instead of a warning.

`Tfr.type`	character A string, either "total" or "internal".
`Rfr.type`	character A string, either "total" or "specular".
`time.unit`	character A string "second", "day" or "exposure".
`bswf.used`	character A string, either "none" or the name of a BSWF.

Value

x

Functions

- `setRawSpct`: Set class of a an object to "cps_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)
```

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: getInstrDesc, getInstrSettings, getWhatMeasured, getWhenMeasured, getWhereMeasured, isValidInstrDesc, isValidInstrSettings, 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: getInstrDesc, getInstrSettings, getWhatMeasured, getWhenMeasured, getWhereMeasured, isValidInstrDesc, isValidInstrSettings, 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

Funtion 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.

---

setScaled *Set the "scaled" attribute*

Description

Funtion 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 overwritting 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: `getInstrDesc`, `getInstrSettings`, `getWhatMeasured`, `getWhenMeasured`, `getWhereMeasured`, `isValidInstrDesc`, `isValidInstrSettings`, `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: getInstrDesc, getInstrSettings, getWhatMeasured, getWhenMeasured, getWhereMeasured, isValidInstrDesc, isValidInstrSettings, 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 geocode 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: getInstrDesc, getInstrSettings, getWhatMeasured, getWhenMeasured, getWhereMeasured, isValidInstrDesc, isValidInstrSettings, 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_mscpt 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, ...)

## S3 method for class 'filter_spct'
smooth_spct(x, method = "custom", strength = 1, ...)

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

```
smooth_spct(x, method = "custom", strength = 1,
  ...)

## S3 method for class 'response_spct'
smooth_spct(x, method = "custom", strength = 1, ...)
```

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

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 array 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.

---

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 streching for several days using the local solar time but distinguishing between days.

Usage

```
solar_time(time = lubridate::now(), geocode = data.frame(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	charater 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 be off by one day, if the time zone does not match the coordinates of the goecode 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 succesive 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 <- ggmap::geocode("Buenos Aires, Argentina")
BA.geocode <- data.frame(lon = -58.38156, lat = -34.60368)
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 fucntions 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, ...)

raw_spct(w.length = NULL, counts = NA_real_, comment = NULL,
  instr.desc = NA, instr.settings = NA, multiple.wl = 1L, ...)

cps_spct(w.length = NULL, cps = NA_real_, comment = NULL,
  instr.desc = NA, instr.settings = NA, multiple.wl = 1L, ...)

generic_spct(w.length = NULL, comment = NULL, multiple.wl = 1L, ...)

response_spct(w.length = NULL, s.e.response = NULL, s.q.response = NULL,
  time.unit = c("second", "day", "exposure"), comment = NULL,
  multiple.wl = 1L, ...)

filter_spct(w.length = NULL, Tfr = NULL, Tpc = NULL, A = NULL,
  Tfr.type = c("total", "internal"), comment = NULL,
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  multiple.wl = 1L, ...)

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, ...)

object_spct(w.length = NULL, Rfr = NULL, Tfr = NULL,
  Tfr.type = c("total", "internal"), Rfr.type = c("total", "specular"),
  comment = NULL, strict.range = getOption("photobiology.strict.range",
  default = FALSE), multiple.wl = 1L, ...)

chroma_spct(w.length = NULL, x, y, z, comment = NULL,
  strict.range = getOption("photobiology.strict.range", default = FALSE),
  multiple.wl = 1L, ...)
```

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.
...	other arguments passed to tibble()
counts	numeric vector with raw counts expressed per scan
instr.desc	a list
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
A	numeric vector of absorbance values (log10 based)
Tfr.type	character string indicating whether transmittance values are "total" or "internal" values
Rfr	numeric vector with spectral refletance 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 ammount of light transmitted.

See Also

Other creation of spectral objects functions: `as.generic_mspct`, `as.generic_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, ...)
```

Arguments

x	data frame
member.class	character Class of the collection members
spct.data.var	character Name of the spctral 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 collections of spectra classes family: generic_mspct, subset2mspct

split_bands *List-of-wavebands constructor*

Description

Build a list of unweighted "waveband" objects that can be used as imput when calculating irradiances.

Usage

```
split_bands(x, list.names = NULL, short.names = is.null(list.names),
  length.out = NULL)
```

Arguments

x	a numeric array 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 wabeband 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 Corresponding vector of spectral (energy) irradiances (W m-2 nm-1)
cut.w.length	numeric Vector of wavelengths (nm)
unit.in	character A string with allowed values "energy", and "photon", or its alias "quantum"
scale	character A 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 indicating whether multiplier values should be cached between calls
use.hinges	logical indicating whether to use hinges to reduce interpolation errors

Value

a numeric array of irradiances with no change in scale factor: [W m-2 nm-1] -> [mol s-1 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 mosts 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 split a spectrum into regions functions: split_irradiance, split_photon_irradiance

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 Corresponding vector of spectral (energy) irradiances (W 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 use hinges to reduce interpolation errors

Value

a numeric array of irradiances with no change in scale factor: [W m-2 nm-1] -> [mol s-1 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 mosts 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 split a spectrum into regions functions: `split_energy_irradiance`, `split_photon_irradiance`

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 Corresponding vector of spectral (energy) irradiances (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 Falg indicating whether to use hinges to reduce interpolation errors

Value

a numeric array of photon irradiances with no change in scale factor: [W m-2 nm-1] -> [mol s-1 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 mosts 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 split a spectrum into regions functions: `split_energy_irradiance`, `split_irradiance`

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,
             cut.w.length = c(200, 400, 500, 600, 900)))
with(sun.data,
     split_photon_irradiance(w.length, s.e.irrad,
             cut.w.length = c(300, 400, 500)))
with(sun.data,
     split_photon_irradiance(w.length, s.e.irrad))
```

---

spread *Length of object in wavelength units*

Description

A function that returns the spread (max(x) - min(x)) for R objects.

Usage

```
spread(x, ...)

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

## S3 method for class 'numeric'
spread(x, ...)
```

```
## S3 method for class 'waveband'
spread(x, ...)

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

## S3 method for class 'generic_mspct'
spread(x, ..., idx = !is.null(names(x)))
```

Arguments

x	an R object
...	not used in current version
idx	logical whether to add a column with the names of the elements of spct

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

```
spread(sun.spct)
```

`stepsize` *Stepsize*

Description

Function that returns the range of step sizes in an object. Range of differences between succesive sorted values.

Usage

```
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 = !is.null(names(x)))
```

Arguments

x	an R object
...	not used in current version
idx	logical whether to add a column with the names of the elements of spct

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: `midpoint`, `min`, `range`

Examples

```
stepsize(sun.spct)

stepsize(sun.spct)
```

`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 soectra 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 collections of spectra classes family: `generic_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 array of wavelength (nm)
w.length2	numeric array of wavelength (nm)
s.irrad1	a numeric array of spectral values
s.irrad2	a numeric array 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-overlaping 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.

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

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

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

Arguments

`w.length1`	numeric array of wavelength (nm)
`w.length2`	numeric array of wavelength (nm)
`s.irrad1`	a numeric array of spectral values
`s.irrad2`	a numeric array 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-overlaping 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.

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`, `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 simualtions 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`, `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`, `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, 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 for a given time and location.

Usage

```
sun_angles(time = lubridate::now(tzone = "UTC"), tz = lubridate::tz(time),
  geocode = data.frame(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 = data.frame(lon = 0, lat = 51.5, address = "Greenwich"),
  use.refraction = FALSE)

sun_zenith_angle(time = lubridate::now(), tz = lubridate::tz(time),
```

```
  geocode = data.frame(lon = 0, lat = 51.5, address = "Greenwich"),
  use.refraction = FALSE)

sun_azimuth(time = lubridate::now(), tz = lubridate::tz(time),
  geocode = data.frame(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.

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, require that either a single time instant or a single location are supplied—i.e. only one of these two arguments can be vectorized in a given call.

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 the times of 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 occusion 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 calcualtion into two separate functions, one returning angles at given instants in time, and a separate one returning the timing of events for given dates.

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` *Spectrum 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 array of wavelengths (nm)
`s.e.irrad`	numeric array 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 respository at http://www.cvrl.org/.

See Also

Other color functions: rgb_spct, w_length2rgb, w_length_range2rgb

Examples

```
my.color <-
    with(sun.data, s_e_irrad2rgb(w.length, s.e.irrad, color.name="sunWhite"))
col2rgb(my.color)
```

T2A *Convert transmittance into absorbance.*

Description

Function that coverts transmittance into absorbance (fraction).

Usage

```
T2A(x, action, byref, ...)

## Default S3 method:
T2A(x, action = NULL, byref = FALSE, ...)

## S3 method for class 'filter_spct'
T2A(x, action = "add", byref = FALSE, ...)

## S3 method for class 'filter_mspct'
T2A(x, action = "add", byref = FALSE, ...)
```

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
...	not used in current version

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`, `as_energy`, `as_quantum_mol`, `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`.

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, ...)
```

Arguments

`x`	an R object
`...`	not used in current version
`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 use hinges to reduce interpolation errors
`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`

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), ..., idx = !is.null(names(spct)))

## 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), ..., idx = !is.null(names(spct)))
```

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 indicating if wavebands crossing spectral data boundaries are trimmed or ignored
`use.hinges`	logical Flag indicating whether to use hinges to reduce interpolation errors
`...`	other arguments
`idx`	logical whether to add a column with the names of the elements of spct

Value

A numeric vector with no change in scale factor

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 mosts 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 appled 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: `getInstrDesc`, `getInstrSettings`, `getWhatMeasured`, `getWhenMeasured`, `getWhereMeasured`, `isValidInstrDesc`, `isValidInstrSettings`, `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: `getInstrDesc`, `getInstrSettings`, `getWhatMeasured`, `getWhenMeasured`, `getWhereMeasured`, `isValidInstrDesc`, `isValidInstrSettings`, `setInstrDesc`, `setInstrSettings`, `setWhatMeasured`, `setWhenMeasured`, `setWhereMeasured`, `trimInstrDesc`

trim_spct *Trim (or expand) head and/or tail*

Description

Trimming of head and tail 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_spct(spct, range = NULL, low.limit = NULL, high.limit = NULL,
  use.hinges = TRUE, fill = NULL, byref = FALSE,
  verbose = getOption("photobiology.verbose", default = FALSE))

trim_mspct(mspct, range = NULL, low.limit = NULL, high.limit = NULL,
  use.hinges = TRUE, fill = NULL, byref = FALSE,
  verbose = getOption("photobiology.verbose", default = TRUE))

trim2overlap(mspct, use.hinges = TRUE,
  verbose = getOption("photobiology.verbose", default = TRUE))

extend2extremes(mspct, use.hinges = TRUE, fill = NA,
  verbose = getOption("photobiology.verbose", default = TRUE))
```

Arguments

spct	an object of class "generic_spct"
range	a numeric vector of length two, or any other object for which function 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, if TRUE (the default) wavelengths in nm.
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"

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 legth two, that can be interpreted as wavelengths expressed in nm.

See Also

Other trim functions: `clip_wl`, `trim_tails`, `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

Trimming of 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)
```

Arguments

`x`	numeric array
`y`	numeric array
`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 largets x-value in input)
`use.hinges`	logical, if TRUE (the default)
`fill`	if fill==NULL then tails are deleted, otherwise tails of y are filled with the value of fill

Value

a data.frame with variables x and y

Note

When expanding an spectrum, if fill==NULL, then expansion is not performed with a warning.

See Also

Other trim functions: clip_wl, trim_spct, trim_waveband, trim_wl

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, if TRUE (the default) hinges are inserted when trimming.

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_tails`, `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

Triming of head and tail of a spectrum based on wavelength limits, interpolation 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, ...)

## 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, if TRUE (the default) wavelengths in nm.
fill	if fill == NULL then tails are deleted, otherwise tails are filled with the value of fill.
...	not used
trim	logical (default is TRUE which trims the wavebands at the boundary, while FALSE discards wavebands that are partly off-boundary).

Value

an R object of same class as input, usually of a different length, either shorter or longer.

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_tails, 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 = Sys.timezone(),
  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'

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
...	not used in current version
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" atrribute 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`

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, ignore_threshold, strict, ...)

## S3 method for class 'numeric'
valleys(x, span = 5, ignore_threshold, strict = TRUE, ...)
```

```
## S3 method for class 'generic_spct'
valleys(x, span = 5, ignore_threshold = 0,
  strict = TRUE, ...)

## S3 method for class 'source_spct'
valleys(x, span = 5, ignore_threshold = 0,
  strict = TRUE, unit.out = getOption("photobiology.radiation.unit", default
  = "energy"), ...)

## S3 method for class 'response_spct'
valleys(x, span = 5, ignore_threshold = 0,
  strict = TRUE, unit.out = getOption("photobiology.radiation.unit", default
  = "energy"), ...)

## S3 method for class 'filter_spct'
valleys(x, span = 5, ignore_threshold = 0,
  strict = TRUE, filter.qty = getOption("photobiology.filter.qty", default =
  "transmittance"), ...)

## S3 method for class 'reflector_spct'
valleys(x, span = 5, ignore_threshold = 0,
  strict = TRUE, ...)

## S3 method for class 'cps_spct'
valleys(x, span = 5, ignore_threshold = 0,
  strict = TRUE, ...)

## S3 method for class 'generic_mspct'
valleys(x, span = 5, ignore_threshold = 0,
  strict = TRUE, ...)
```

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 talelst peakthreshold 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
unit.out	character One of "energy" or "photon"
filter.qty	character One of "transmittance" or "absorbance"

Value

a subset of x with rows corresponding to local maxima.

Methods (by class)

- `default`: Default function usable on numeric vectors.
- `numeric`: Default function usable on numeric vectors.
- `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`

Examples

```
valleys(sun.spct)
```

---

waveband *Waveband constructor method*

Description

Constructor for "waveband" objects that can be used as imput 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 array of two numeric values, describimg a range of wavelengths (nm)
weight	a character string "SWF" or "BSWF", use NULL (the defalt) 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 array giving the wavelengths at which the s.irrad should be inserted by interpolation, no interpolation is indicated by an empty array (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 ploting, 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 Corresponding vector of spectral (energy) irradiances (W m-2 nm-1)
w.band.num	waveband
w.band.denom	waveband
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 use hinges to reduce interpolation errors

Value

a single numeric value giving the ratio

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 midppoint 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 use hinges to reduce interpolation errors
short.names	logical Flag indicating whether to use short or long names for wavebands
...	not used in current version

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

---

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, 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, 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, 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, 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
```

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`, `s_e_irrad2rgb`, `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

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, s_e_irrad2rgb, 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, 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