Package ‘ggspectra’

Type Package
Title Extensions to 'ggplot2' for Radiation Spectra
Version 0.3.6
Date 2020-04-20
Maintainer Pedro J. Aphalo <pedro.aphalo@helsinki.fi>
Description Additional annotations, stats, geoms and scales for plotting
``light'' spectra with 'ggplot2', together with specializations of ggplot()
and autoplot() methods for spectral data and waveband definitions
stored in objects of classes defined in package 'photobiology'. Part of the
License GPL (>= 2)
LazyData TRUE
LazyLoad TRUE
ByteCompile TRUE
Depends R (>= 3.6.0), photobiology (>= 0.10.1), ggplot2 (>= 3.3.0)
Imports photobiologyWavebands (>= 0.4.3), scales (>= 1.1.0), ggrepel
(>= 0.8.2), dplyr (>= 0.8.1), lubridate (>= 1.7.4), tidyr (>=
1.0.2)
Suggests rlang (>= 0.4.5), knitr (>= 1.28), rmarkdown (>= 2.1)
URL https://docs.r4photobiology.info/ggspectra/,
https://bitbucket.org/aphalo/ggspectra/
BugReports https://bitbucket.org/aphalo/ggspectra/issues/
Encoding UTF-8
RoxygenNote 7.1.0
VignetteBuilder knitr
NeedsCompilation no
Author Pedro J. Aphalo [aut, cre] (<https://orcid.org/0000-0003-3385-972X>),
Titta K. Kotilainen [ctb] (<https://orcid.org/0000-0002-2822-9734>)
Repository CRAN
Date/Publication 2020-04-28 15:30:02 UTC
R topics documented:

- ggspectra-package ........................................... 3
- autoplot.calibration_spect ................................. 5
- autoplot.cps_spect ........................................... 7
- autoplot.filter_spect ....................................... 8
- autoplot.object_spect ..................................... 11
- autoplot.raw_spect ......................................... 13
- autoplot.reflector_spect ................................... 15
- autoplot.response_spect ................................... 17
- autoplot.source_spect ..................................... 19
- autoplot.waveband ......................................... 21
- autotitle ..................................................... 23
- A_internal_label ............................................ 24
- black_or_white ............................................. 25
- color_chart .................................................. 26
- counts_label ............................................... 27
- cps_label ................................................... 27
- exponent2prefix ............................................ 28
- geom_spect .................................................. 29
- ggplot ....................................................... 31
- multiplot .................................................... 34
- plot.generic_spect ......................................... 36
- Rfr_specular_label ......................................... 37
- s.e.irrad_label ............................................. 38
- s.e.response_label ......................................... 39
- scale_x_wl_continuous .................................... 39
- scale_y_A_internal_continuous ......................... 41
- scale_y_counts_continuous ............................... 42
- scale_y_cps_continuous .................................. 43
- scale_y_Rfr_specular_continuous ....................... 44
- scale_y_s.e.irrad_continuous ......................... 45
- scale_y_s.e.response_continuous ....................... 46
- scale_y_Tfr_internal_continuous ....................... 47
- sec_axis_w_number ....................................... 49
- set_annotations_default ................................ 50
- SI_pl_format ............................................... 51
- SI_tg_format ............................................... 52
- stat_color .................................................. 53
- stat_find_qtys ............................................. 54
- stat_find_wls .............................................. 57
- stat_label_peaks .......................................... 60
- stat_peaks ................................................. 63
- stat_spikes ................................................ 67
- stat_wb_box ............................................... 70
- stat_wb_column .......................................... 73
- stat_wb_contribution .................................... 75
- stat_wb_hbar .............................................. 78
Description


Details

Package 'ggspectra' provides a set of stats, geoms and methods extending packages 'ggplot2' and 'photobiology'. They easy the task of plotting radiation-related spectra and of annotating the resulting plots with labels and summary quantities derived from the spectral data.

Plot methods automate in many respects the plotting of spectral data. 'ggplot2' compatible statistics make the addition of labels or plotting of subject-area specific summaries possible as well as the addition of labels and wavelength-based colour to plots easy. Available summaries are most of those relevant to photobiology. However, many of the functions in the package are more generally useful for plotting UV, VIS and NIR spectra of light emission, transmittance, reflectance, absorptance, and responses.

The available summary quantities are both simple statistical summaries and response-weighted summaries. Simple derived quantities represent summaries of a given range of wavelengths, and can be expressed either in energy or photon based units. Derived biologically effective quantities are used to quantify the effect of radiation on different organisms or processes within organisms. These effects can range from damage to perception of informational light signals. Additional features of spectra may be important and worthwhile annotating in plots. Of these, local maxima (peaks) and minima (valleys) present in spectral data can also be annotated with statistics made available by the 'ggspectra' package.

Package 'ggspectra' is useful solely for plotting spectral data as most functions depend on the x aesthetic being mapped to a variable containing wavelength values expressed in nanometres. It works well together with some other extensions to package 'ggplot2' such as packages 'ggrepel' and 'cowplot'.
This package is part of a suite of R packages for photobiological calculations described at the [r4photobiology](https://www.r4photobiology.info) web site.

**Note**

This package makes use of the new features of ’ggplot2’ >= 2.0.0 that make writing this kind of extensions easy and is consequently not compatible with earlier versions of ’ggplot2’.

**Author(s)**

**Maintainer**: Pedro J. Aphalo <pedro.aphalo@helsinki.fi> ([ORCID](https://orcid.org/0000-0002-7174-6969))

Other contributors:

- Titta K. Kotilainen ([ORCID](https://orcid.org/0000-0002-7174-6969)) [contributor]

**References**


ggplot2 web site at [https://ggplot2.tidyverse.org/](https://ggplot2.tidyverse.org/)
ggplot2 source code at [https://github.com/hadley/ggplot2](https://github.com/hadley/ggplot2)
Function `multiplot` from [http://www.cookbook-r.com/](http://www.cookbook-r.com/)

**See Also**

Useful links:

- [https://docs.r4photobiology.info/ggspectra/](https://docs.r4photobiology.info/ggspectra/)
- [https://bitbucket.org/aphalo/ggspectra/](https://bitbucket.org/aphalo/ggspectra/)
- Report bugs at [https://bitbucket.org/aphalo/ggspectra/issues/](https://bitbucket.org/aphalo/ggspectra/issues/)

**Examples**

```r
library(photobiologyWavebands)

ggplot(sun.spct) + geom_line() + stat_peaks(span = NULL)

ggplot(sun.spct, aes(w.length, s.e.irrad)) + geom_line() +
  stat_peaks(span = 21, geom = "point", colour = "red") +
  stat_peaks(span = 51, geom = "text", colour = "red", vjust = -0.3,
             label.fmt = "%3.0f nm")

ggplot(polyester.spct, range = UV()) + geom_line()

plot(sun.spct)

plot(polyester.spct, UV_bands(), range = UV(),
     annotations = c("="", "segments", "labels"))
```
Create a complete ggplot for an irradiation calibration spectrum.

Description

These methods return a ggplot object with an annotated plot of a calibration_spct object or of the spectra contained in a calibration_mspct object.

Usage

```r
## S3 method for class 'calibration_spct'
aplot(calibration_spct, 
  object,  
  ...,  
  w.band = getOption("photobiology.plot.bands", default = list(UVC(), UVB(), UVA(), PAR())),  
  range = NULL,  
  unit.out = "counts",  
  pc.out = FALSE,  
  label.qty = "mean",  
  span = NULL,  
  wls.target = "HM",  
  annotations = NULL,  
  time.format = "",  
  tz = "UTC",  
  norm = NULL,  
  text.size = 2.5,  
  idfactor = NULL,  
  ylim = c(NA, NA),  
  object.label = deparse(substitute(object)),  
  na.rm = TRUE)

## S3 method for class 'calibration_mspct'
aplot(calibration_mspct, ..., range = NULL, plot.data = "as.is")
```

Arguments

- `object`: a calibration_spct object or a calibration_mspct object.
- `...`: in the case of collections of spectra, additional arguments passed to the plot methods for individual spectra, otherwise currently ignored.
- `w.band`: a single waveband object or a list of waveband objects.
- `range`: an R object on which range() returns a vector of length 2, with min and max wavelengths (nm).
- `unit.out`: character IGNORED.
pc.out logical, if TRUE use percents instead of fraction of one.
label_qty character string giving the type of summary quantity to use for labels, one of
"mean", "total", "contribution", and "relative".
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.
wls.target numeric vector indicating the spectral quantity values for which wavelengths are
to be searched and interpolated if need. The character strings "half.maximum" and
"half.range" are also accepted as arguments. A list with numeric and/or
character values is also accepted.
annotations a character vector ("summaries" is ignored).
time.format character Format as accepted by \texttt{strptime}.

\texttt{tz} character Time zone to use for title and/or subtitle.
norm numeric normalization wavelength (nm) or character string "max" for normal-
ization at the wavelength of highest peak.
text.size numeric size of text in the plot decorations.

\texttt{idfactor} character Name of an index column in data holding a factor with each spec-
trum in a long-form multispectrum object corresponding to a distinct spectrum.
If \texttt{idfactor}=NULL the name of the factor is retrieved from metadata or if no
metadata found, the default "spct.idx" is tried. If \texttt{idfactor}=NA no aesthetic is
mapped to the spectra and the user needs to use \texttt{ggplot2} functions to manually
map an aesthetic or use facets for the spectra.

\texttt{ylim} numeric y axis limits,

\texttt{object.label} character The name of the object being plotted.

\texttt{na.rm} logical.

\texttt{plot.data} character Data to plot. Default is "as.is" plotting one line per spectrum. When
passing "mean" or "median" as argument all the spectra must contain data at the
same wavelength values.

\texttt{Value}

a \texttt{ggplot} object.

\texttt{Note}

Note that scales are expanded so as to make space for the annotations. The object returned is a
\texttt{ggplot} objects, and can be further manipulated.

\texttt{See Also}

Other autoplot methods: \texttt{autoplot.cps_spct()}, \texttt{autoplot.filter_spct()}, \texttt{autoplot.object_spct()},
\texttt{autoplot.raw_spct()}, \texttt{autoplot.reflector_spct()}, \texttt{autoplot.response_spct()}, \texttt{autoplot.source_spct()},
\texttt{autoplot.waveband()}, \texttt{set_annotations_default()}

\texttt{Value}

a \texttt{ggplot} object.

\texttt{Note}

Note that scales are expanded so as to make space for the annotations. The object returned is a
\texttt{ggplot} objects, and can be further manipulated.

\texttt{See Also}

Other autoplot methods: \texttt{autoplot.cps_spct()}, \texttt{autoplot.filter_spct()}, \texttt{autoplot.object_spct()},
\texttt{autoplot.raw_spct()}, \texttt{autoplot.reflector_spct()}, \texttt{autoplot.response_spct()}, \texttt{autoplot.source_spct()},
\texttt{autoplot.waveband()}, \texttt{set_annotations_default()}


**Description**

This function returns a ggplot object with an annotated plot of a response_spct object.

**Usage**

```r
## S3 method for class 'cps_spct'
autoplot(
  object,
  ..., 
  w.band = getOption("photobiology.plot.bands", default = list(UVC(), UVB(), UVA(), PAR())),
  range = NULL,
  unit.out = "cps",
  pc.out = FALSE,
  label.qty = "mean",
  span = NULL,
  wls.target = "HM",
  annotations = NULL,
  time.format = "",
  tz = "UTC",
  norm = NULL,
  text.size = 2.5,
  idfactor = NULL,
  ylim = c(NA, NA),
  object.label = deparse(substitute(object)),
  na.rm = TRUE
)
```

**Arguments**

- **object**
  - a cps_spct object.

- **...**
  - in the case of collections of spectra, additional arguments passed to the plot methods for individual spectra, otherwise currently ignored.

- **w.band**
  - a single waveband object or a list of waveband objects.

- **range**
  - an R object on which range() returns a vector of length 2, with min and max wavelengths (nm).

- **unit.out**
  - character, ignored.

- **pc.out**
  - logical, if TRUE use percents instead of fraction of one

- **label.qty**
  - character string giving the type of summary quantity to use for labels, one of "mean", "total", "contribution", and "relative".
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.

wls.target
numeric vector indicating the spectral quantity values for which wavelengths are to be searched and interpolated if need. The character strings "half.maximum" and "half.range" are also accepted as arguments. A list with numeric and/or character values is also accepted.

annotations
a character vector ("summarizes" is ignored).

time.format
character Format as accepted by `strptime`.

tz
character Time zone to use for title and/or subtitle.

norm
numeric normalization wavelength (nm) or character string "max" for normalization at the wavelength of highest peak.

text.size
numeric size of text in the plot decorations.

idfactor
character Name of an index column in data holding a factor with each spectrum in a long-form multispectrum object corresponding to a distinct spectrum. If idfactor=NULL the name of the factor is retrieved from metadata or if no metadata found, the default "spct.idx" is tried.

ylim
numeric y axis limits,

object.label
character The name of the object being plotted.

na.rm
logical.

Value
a `ggplot` object.

Note
Note that scales are expanded so as to make space for the annotations. The object returned is a `ggplot` objects, and can be further manipulated.

See Also
Other autoplot methods: `autoplot.calibration_spct()`, `autoplot.filter_spct()`, `autoplot.object_spct()`, `autoplot.raw_spct()`, `autoplot.reflector_spct()`, `autoplot.response_spct()`, `autoplot.source_spct()`, `autoplot.waveband()`, `set_annotations_default()`

'autoplot.filter_spct' Create a complete `ggplot` for a filter spectrum.

Description
These methods return a ggplot object with an annotated plot of a filter_spct object or of the spectra contained in a filter_mspct object.
Usage

## S3 method for class 'filter_spct'
autoplot(
  object,
  ...,
  w.band = getOption("photobiology.plot.bands", default = list(UVC(), UVB(), UVA(),
                                                  PAR())),
  range = NULL,
  plot.qty = getOption("photobiology.filter.qty", default = "transmittance"),
  pc.out = FALSE,
  label.qty = NULL,
  span = NULL,
  wls.target = "HM",
  annotations = NULL,
  time.format = "",
  tz = "UTC",
  text.size = 2.5,
  chroma.type = "CMF",
  idfactor = NULL,
  ylim = c(NA, NA),
  object.label = deparse(substitute(object)),
  na.rm = TRUE
)

## S3 method for class 'filter_mspct'
autoplot(
  object,
  ...,
  range = NULL,
  plot.qty = getOption("photobiology.filter.qty", default = "transmittance"),
  plot.data = "as.is"
)

Arguments

object        a filter_spct object or a filter_mspct object.
...           in the case of collections of spectra, additional arguments passed to the plot
              methods for individual spectra, otherwise currently ignored.

w.band        a single waveband object or a list of waveband objects.
range         an R object on which range() returns a vector of length 2, with min annd max
              wavelengths (nm).

plot.qty      character string one of "transmittance" or "absorbance".
pc.out        logical, if TRUE use percents instead of fraction of one.
label.qty     character string giving the type of summary quantity to use for labels, one of
              "mean", "total", "contribution", and "relative".
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.
wls.target numeric vector indicating the spectral quantity values for which wavelengths are to be searched and interpolated if need. The character strings "half.maximum" and "half.range" are also accepted as arguments. A list with numeric and/or character values is also accepted.

annotations a character vector.

time.format character Format as accepted by strptime.

tz character Time zone to use for title and/or subtitle.

text.size numeric size of text in the plot decorations.

chroma.type character one of "CMF" (color matching function) or "CC" (color coordinates) or a chroma_spct object.

idfactor character Name of an index column in data holding a factor with each spectrum in a long-form multispectrum object corresponding to a distinct spectrum. If idfactor=NULL the name of the factor is retrieved from metadata or if no metadata found, the default "spct.idx" is tried. If idfactor=NA no aesthetic is mapped to the spectra and the user needs to use 'ggplot2' functions to manually map an aesthetic or use facets for the spectra.

ylim numeric y axis limits,

object.label character The name of the object being plotted.

na.rm logical.

plot.data character Data to plot. Default is "as.is" plotting one line per spectrum. When passing "mean" or "median" as argument all the spectra must contain data at the same wavelength values.

Value

a ggplot object.

Note

The ggplot object returned can be further manipulated and added to. Except when no annotations are added, limits are set for the x-axis and y-axis scales. The y scale limits are expanded to include all data, or at least to the range of expected values. The plotting of absorbance is an exception as the y-axis is not extended past 6 a.u. In the case of absorbance, values larger than 6 a.u. are rarely meaningful due to stray light during measurement. However, when transmittance values below the detection limit are rounded to zero, and later converted into absorbance, values Inf a.u. result, disrupting the plot. Scales are further expanded so as to make space for the annotations.

See Also

Other autoplot methods: autoplot.calibration_spct(), autoplot.cps_spct(), autoplot.object_spct(), autoplot.raw_spct(), autoplot.reflector_spct(), autoplot.response_spct(), autoplot.source_spct(), autoplot.waveband(), set_annotations_default()
Example

```r
autoplot(yellow_gel.spct)
autoplot(yellow_gel.spct, pc.out = TRUE)
```

Description

This function returns a ggplot object with an annotated plot of an object_spct object.

Usage

```r
## S3 method for class 'object_spct'
autoplot(
  object,
  ..., w.band = getOption("photobiology.plot.bands", default = list(UVC(), UVB(), UVA(), PAR())),
  range = NULL,
  plot.qty = "all",
  pc.out = FALSE,
  label.qty = NULL,
  span = 61,
  wls.target = "HM",
  annotations = NULL,
  time.format = "",
  tz = "UTC",
  stacked = TRUE,
  text.size = 2.5,
  chroma.type = "CMF",
  idfactor = NULL,
  ylim = c(NA, NA),
  object.label = deparse(substitute(object)),
  na.rm = TRUE
)

## S3 method for class 'object_mspct'
autoplot(object, ..., range = NULL)
```

Arguments

- `object` an object_spct object
- `...` in the case of collections of spectra, additional arguments passed to the plot methods for individual spectra, otherwise currently ignored.
w.band  
a single waveband object or a list of waveband objects

range  
an R object on which range() returns a vector of length 2, with min and max wavelengths (nm)

plot.qty  
character string, one of "all", "transmittance", "absorbance", "absorptance", or "reflectance".

pc.out  
logical, if TRUE use percents instead of fraction of one

label.qty  
character string giving the type of summary quantity to use for labels, one of "mean", "total", "contribution", and "relative".

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.

wls.target  
numeric vector indicating the spectral quantity values for which wavelengths are to be searched and interpolated if need. The character strings "half.maximum" and "half.range" are also accepted as arguments. A list with numeric and/or character values is also accepted.

annotations  
a character vector

time.format  
character Format as accepted by \texttt{strptime}.

tz  
character Time zone to use for title and/or subtitle.

stacked  
logical

text.size  
numeric size of text in the plot decorations.

chroma.type  
character one of "CMF" (color matching function) or "CC" (color coordinates) or a \texttt{chroma_spct} object.

idfactor  
character Name of an index column in data holding a factor with each spectrum in a long-form multispectrum object corresponding to a distinct spectrum. If idfactor=NULL the name of the factor is retrieved from metadata or if no metadata found, the default "spct.idx" is tried. If idfactor=NA no aesthetic is mapped to the spectra and the user needs to use 'ggplot2' functions to manually map an aesthetic or use facets for the spectra.

ylim  
numeric y axis limits,

object.label  
character The name of the object being plotted.

na.rm  
logical.

\textbf{Value}

a \texttt{ggplot} object.

\textbf{Note}

The ggplot object returned can be further manipulated and added to. Except when no annotations are added, limits are set for the x-axis and y-axis scales. The y scale limits are expanded to include all data, or at least to the range of expected values. Scales are further expanded so as to make space for the annotations. When all "all" quantities are plotted, a single set of spectra is accepted as input.
autplot.raw_spct

See Also

Other autoplot methods: autoplot.calibration_spct(), autoplot.cps_spct(), autoplot.filter_spct(), autoplot.raw_spct(), autoplot.reflector_spct(), autoplot.response_spct(), autoplot.source_spct(), autoplot.waveband(), set_annotations_default()

Examples

autplot(Ler_leaf.spct)

autplot.raw_spct Create a complete ggplot for raw detector-counts spectra.

Description

This function returns a ggplot object with an annotated plot of a raw_spct object.

Usage

## S3 method for class 'raw_spct'
autplot(
  object,
  ..., 
  w.band = getOption("photobiology.plot.bands", default = list(UVC(), UVB(), UVA(), PAR())),
  range = NULL,
  unit.out = "counts",
  pc.out = FALSE,
  label.qty = "mean",
  span = NULL,
  wls.target = "HM",
  annotations = NULL,
  time.format = "",
  tz = "UTC",
  norm = NULL,
  text.size = 2.5,
  idfactor = NULL,
  ylim = c(NA, NA),
  object.label = deparse(substitute(object)),
  na.rm = TRUE
)
Arguments

- `object` a raw_spct object.
- `...` in the case of collections of spectra, additional arguments passed to the plot methods for individual spectra, otherwise currently ignored.
- `w.band` a single waveband object or a list of waveband objects.
- `range` an R object on which range() returns a vector of length 2, with min and max wavelengths (nm).
- `unit.out` character IGNORED.
- `pc.out` logical, if TRUE use percents instead of fraction of one.
- `label.qty` character string giving the type of summary quantity to use for labels, one of "mean", "total", "contribution", and "relative".
- `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.
- `wls.target` numeric vector indicating the spectral quantity values for which wavelengths are to be searched and interpolated if need. The character strings "half.maximum" and "half.range" are also accepted as arguments. A list with numeric and/or character values is also accepted.
- `annotations` a character vector ("summaries" is ignored).
- `time.format` character Format as accepted by `strptime`.
- `tz` character Time zone to use for title and/or subtitle.
- `norm` numeric normalization wavelength (nm) or character string "max" for normalization at the wavelength of highest peak.
- `text.size` numeric size of text in the plot decorations.
- `idfactor` character Name of an index column in data holding a factor with each spectrum in a long-form multispectrum object corresponding to a distinct spectrum. If `idfactor=NULL` the name of the factor is retrieved from metadata or if no metadata found, the default "spct.idx" is tried.
- `ylim` numeric y axis limits.
- `object.label` character The name of the object being plotted.
- `na.rm` logical.

Value

a ggplot object.

Note

Note that scales are expanded so as to make space for the annotations. The object returned is a ggplot objects, and can be further manipulated.

See Also

Other autoplot methods: `autoplot.calibration_spct()`, `autoplot.cps_spct()`, `autoplot.filter_spct()`, `autoplot.object_spct()`, `autoplot.reflector_spct()`, `autoplot.response_spct()`, `autoplot.source_spct()`, `autoplot.waveband()`, `set_annotations_default()`
Create a complete ggplot for a reflector spectrum.

**Description**

These methods return a ggplot object with an annotated plot of a reflector_spct object or of the spectra contained in a reflector_mspct object.

**Usage**

```r
## S3 method for class 'reflector_spct'
autoplot(
  object,
  ..., 
  w.band = getOption("photobiology.plot.bands", default = list(UVC(), UVB(), UVA(), PAR())),
  range = NULL,
  plot.qty = getOption("photobiology.reflector.qty", default = "reflectance"),
  pc.out = FALSE,
  label.qty = NULL,
  span = NULL,
  wls.target = "HM",
  annotations = NULL,
  time.format = ",",
  tz = "UTC",
  text.size = 2.5,
  chroma.type = "CMF",
  idfactor = NULL,
  ylim = c(NA, NA),
  object.label = deparse(substitute(object)),
  na.rm = TRUE
)

## S3 method for class 'reflector_mspct'
autoplot(
  object,
  ..., 
  range = NULL,
  plot.qty = getOption("photobiology.reflector.qty", default = "reflectance"),
  plot.data = "as.is"
)
```

**Arguments**

- `object`  a reflector_spct object or a reflector_mspct object.
... in the case of collections of spectra, additional arguments passed to the plot methods for individual spectra, otherwise currently ignored.

w.band a single waveband object or a list of waveband objects.

range an R object on which range() returns a vector of length 2, with min and max wavelengths (nm).

plot.qty character string (currently ignored).

pc.out logical, if TRUE use percents instead of fraction of one.

label.qty character string giving the type of summary quantity to use for labels, one of "mean", "total", "contribution", and "relative".

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.

wls.target numeric vector indicating the spectral quantity values for which wavelengths are to be searched and interpolated if need. The character strings "half.maximum" and "half.range" are also accepted as arguments. A list with numeric and/or character values is also accepted.

annotations a character vector.

time.format character Format as accepted by strptime.

tz character Time zone to use for title and/or subtitle.

text.size numeric size of text in the plot decorations.

chroma.type character one of "CMF" (color matching function) or "CC" (color coordinates) or a chroma_spct object.

idfactor character Name of an index column in data holding a factor with each spectrum in a long-form multispectrum object corresponding to a distinct spectrum. If idfactor=NULL the name of the factor is retrieved from metadata or if no metadata found, the default "spct.idx" is tried. If idfactor=NA no aesthetic is mapped to the spectra and the user needs to use `ggplot2` functions to manually map an aesthetic or use facets for the spectra.

ylim numeric y axis limits,

object.label character The name of the object being plotted.

na.rm logical.

plot.data character Data to plot. Default is "as.is" plotting one line per spectrum. When passing "mean" or "median" as argument all the spectra must contain data at the same wavelength values.

Value

a ggplot object.

Note

The ggplot object returned can be further manipulated and added to. Except when no annotations are added, limits are set for the x-axis and y-axis scales. The y scale limits are expanded to include all data, or at least to the range of expected values. Scales are further expanded so as to make space for the annotations.
See Also
Other autoplot methods: autoplot.calibration_spct(), autoplot.cps_spct(), autoplot.filter_spct(), autoplot.object_spct(), autoplot.raw_spct(), autoplot.response_spct(), autoplot.source_spct(), autoplot.waveband(), set_annotations_default()

Examples

autoplot(Ler_leaf_rflt.spct)

autoplott.response_spct

Create a complete ggplot for a response spectrum.

Description
These methods return a ggplot object with an annotated plot of a response_spct object or of the spectra contained in a response_mspct object.

Usage

## S3 method for class 'response_spct'
autoplot(
  object,
  ..., 
  w.band = getOption("photobiology.plot.bands", default = list(UVC(), UVB(), UVA(), PAR())),
  range = NULL,
  unit.out = getOption("photobiology.radiation.unit", default = "energy"),
  pc.out = FALSE,
  label.qty = NULL,
  span = NULL,
  wls.target = "HM",
  annotations = NULL,
  time.format = ",",
  tz = "UTC",
  norm = "max",
  text.size = 2.5,
  idfactor = NULL,
  ylim = c(NA, NA),
  object.label = deparse(substitute(object)),
  na.rm = TRUE
)

## S3 method for class 'response_mspct'
autoplot(object, ..., range = NULL, plot.data = "as.is")
Arguments

- **object** a response_spct object or a response_mspct object.

... in the case of collections of spectra, additional arguments passed to the plot methods for individual spectra, otherwise currently ignored.

- **w.band** a single waveband object or a list of waveband objects.

- **range** an R object on which range() returns a vector of length 2, with min and max wavelengths (nm).

- **unit.out** character string indicating type of radiation units to use for plotting: "photon" or its synonym "quantum", or "energy".

- **pc.out** logical, if TRUE use percents instead of fraction of one.

- **label.qty** character string giving the type of summary quantity to use for labels, one of "mean", "total", "contribution", and "relative".

- **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.

- **wls.target** numeric vector indicating the spectral quantity values for which wavelengths are to be searched and interpolated if need. The character strings "half.maximum" and "half.range" are also accepted as arguments. A list with numeric and/or character values is also accepted.

- **annotations** a character vector.

- **time.format** character Format as accepted by `strptime`.

- **tz** character Time zone to use for title and/or subtitle.

- **norm** numeric normalization wavelength (nm) or character string "max" for normalization at the wavelength of highest peak, or NULL for plotting the spectrum as is.

- **text.size** numeric size of text in the plot decorations.

- **idfactor** character Name of an index column in data holding a factor with each spectrum in a long-form multispectrum object corresponding to a distinct spectrum. If idfactor=NULL the name of the factor is retrieved from metadata or if no metadata found, the default "spct.idx" is tried.

- **ylim** numeric y axis limits.

- **object.label** character The name of the object being plotted.

- **na.rm** logical.

- **plot.data** character Data to plot. Default is "as.is" plotting one line per spectrum. When passing "mean" or "median" as argument all the spectra must contain data at the same wavelength values.

Value

- a `ggplot` object.

Note

Note that scales are expanded so as to make space for the annotations. The object returned is a `ggplot` object, and can be further manipulated and added to.
See Also

Other autoplot methods: autoplot.calibration_spct(), autoplot.cps_spct(), autoplot.filter_spct(), autoplot.object_spct(), autoplot.raw_spct(), autoplot.reflector_spct(), autoplot.source_spct(), autoplot.waveband(), set_annotations_default()

Examples

autoplot(photodiode.spct)
autoplot(photodiode.spct, unit.out = "photon")

autoplot source_spct  Create a complete ggplot for a light-source spectrum.

Description

These methods return a ggplot object with an annotated plot of a source_spct object or of the spectra contained in a source_mspct object.

Usage

## S3 method for class 'source_spct'
autoplot(
  object,
  ..., 
  w.band = getOption("photobiology.plot.bands", default = list(UVC(), UVB(), UVA(), PAR())),
  range = NULL,
  unit.out = getOption("photobiology.radiation.unit", default = "energy"),
  label.qty = NULL,
  span = NULL,
  wls.target = "HM",
  annotations = NULL,
  time.format = "",
  tz = "UTC",
  text.size = 2.5,
  chroma.type = "CMF",
  idfactor = NULL,
  ylim = c(NA, NA),
  object.label = deparse(substitute(object)),
  na.rm = TRUE
)

## S3 method for class 'source_mspct'
autoplot(
  object,
...,
  range = NULL,
  unit.out = getOption("photobiology.radiation.unit", default = "energy"),
  plot.data = "as.is"
)

Arguments

object a source_spect or a source_mspct object.

... in the case of collections of spectra, additional arguments passed to the plot
methods for individual spectra, otherwise currently ignored.

w.band a single waveband object or a list of waveband objects.

range an R object on which range() returns a vector of length 2, with min and max
wavelengths (nm).

unit.out character string indicating type of radiation units to use for plotting: "photon"
or its synonym "quantum", or "energy".

label.qty character string giving the type of summary quantity to use for labels, one of
"mean", "total", "contribution", and "relative".

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.

wls.target numeric vector indicating the spectral quantity values for which wavelengths are
to be searched and interpolated if need. The character strings "half.maximum" and
"half.range" are also accepted as arguments. A list with numeric and/or
character values is also accepted.

annotations a character vector.

time.format character Format as accepted by strptime.

tz character Time zone to use for title and/or subtitle.

text.size numeric size of text in the plot decorations.

chroma.type character one of "CMF" (color matching function) or "CC" (color coordinates)
or a chroma_spect object.

idfactor character Name of an index column in data holding a factor with each spec-
trum in a long-form multispectrum object corresponding to a distinct spectrum.
If idfactor=NULL the name of the factor is retrieved from metadata or if no
metadata found, the default "spct.idx" is tried.

ylim numeric y axis limits,

object.label character The name of the object being plotted.

na.rm logical.

plot.data character Data to plot. Default is "as.is" plotting one line per spectrum. When
passing "mean" or "median" as argument all the spectra must contain data at the
same wavelength values.

Value

a ggplot object.
Note

Note that scales are expanded so as to make space for the annotations. The object returned is a ggplot object, and can be further manipulated and added to.

See Also

Other autoplot methods: autoplot.calibration_spct(), autoplot.cps_spct(), autoplot.filter_spct(), autoplot.object_spct(), autoplot.raw_spct(), autoplot.reflector_spct(), autoplot.response_spct(), autoplot.waveband(), set_annotations_default()

Examples

autoplot(sun.spct)
autoplot(sun.spct, unit.out = "photon")

autoplot.waveband

Create a complete ggplot for a waveband descriptor.

Description

This function returns a ggplot object with an annotated plot of a waveband object.

Usage

## S3 method for class 'waveband'
autoplot(
  object,
  ...
  w.length = NULL,
  range = c(280, 800),
  fill = 0,
  span = NULL,
  wls.target = "HM",
  unit.in = getOption("photobiology.radiation.unit", default = "energy"),
  annotations = NULL,
  wb.trim = TRUE,
  norm = NULL,
  text.size = 2.5,
  ylim = c(NA, NA),
  object.label = deparse(substitute(object)),
  na.rm = TRUE
)
Arguments

- `object`: a waveband object.
- `...`: currently ignored.
- `w.length`: numeric vector of wavelengths (nm)
- `range`: an R object on which range() returns a vector of length 2, with min and max wavelengths (nm).
- `fill`: value to use as response for wavelengths outside the waveband range.
- `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.
- `wls.target`: numeric vector indicating the spectral quantity values for which wavelengths are to be searched and interpolated if need. The character strings "half.maximum" and "half.range" are also accepted as arguments. A list with numeric and/or character values is also accepted.
- `unit.in`: the type of unit we assume as reference "energy" or "photon" based.
- `annotations`: a character vector.
- `wb.trim`: logical.
- `norm`: numeric normalization wavelength (nm) or character string "max" for normalization at the wavelength of highest peak.
- `text.size`: numeric size of text in the plot decorations.
- `ylim`: numeric y axis limits.
- `object.label`: character The name of the object being plotted.
- `na.rm`: logical.

Value

a ggplot object.

Note

Note that scales are expanded so as to make space for the annotations. The object returned is a ggplot object, and can be further manipulated.

Effectiveness spectra are plotted expressing the spectral effectiveness either as $1 \text{ mol}^{-1} \text{ nm}$ photons or $1 \text{ J}^{-1} \text{ nm}$ which can selected through formal argument `unit.out`. The value of `unit.in` has no effect on the result when using BSWFs, as BSWFs are defined based on a certain base of expression, which is enforced. In contrast, for wavebands which only define a wavelength range, changing the assumed reference irradiance, changes the responsivity according to Plank's law.

This function creates a response_spct object from the waveband object and plots it. Unused arguments are passed along, which means that other plot aspects can be controlled by providing arguments for the plot method of the response_spct class.

See Also

Other autoplot methods: autoplot.calibration_spct(), autoplot.cps_spct(), autoplot.filter_spct(), autoplot.object_spct(), autoplot.raw_spct(), autoplot.reflector_spct(), autoplot.response_spct(), autoplot.source_spct(), set_annotations_default()
Examples

```r
autoplot(waveband(c(400, 500)))
```

---

**autotitle**  
*Add title, subtitle and caption to a spectral plot*

**Description**

Add a title, subtitle and caption to a spectral plot based on automatically extracted metadata stored from an spectral object.

**Usage**

```r
autotitle(
  object,
  object.label = deparse(substitute(object)),
  annotations = "title",
  time.format = "",
  tz = lubridate::tz(getWhenMeasured(object)),
  default.title = "title:objt"
)
```

```r
ggtitle_spct(
  object,
  object.label = deparse(substitute(object)),
  annotations = "title",
  time.format = "",
  tz = lubridate::tz(getWhenMeasured(object)),
  default.title = "title:objt"
)
```

**Arguments**

- `object`  
  generic_spct The spectral object plotted.

- `object.label`  
  character The name of the object being plotted.

- `annotations`  
  character vector Annotations as described for `plot()` methods, values unrelated to title are ignored.

- `time.format`  
  character Format as accepted by `strptime`.

- `tz`  
  character time zone used in labels.

- `default.title`  
  character vector The default used for `annotations = "title"`. 
Details

autotitle() retrieves from object object metadata and passes it to ggplot2::ggtitle() as arguments for title, subtitle and caption. The specification for the title is passed as argument to annotations, and consists in the keyword title with optional modifiers selecting the kind of metadata to use, separated by colons. Up to three keywords separated by colons are accepted, and correspond to title, subtitle and caption. The recognized keywords are: "objt", "class", "what", "when", "where", "how", "inst.name", "inst.sn", "comment" and "none" are recognized as modifiers to "title"; "none" is a placeholder.

Value

The return value of ggplot2::labs().

Note

Method renamed as autotitle() to better reflect its function; ggtitle_spct() is deprecated but will remain available for backwards compatibility.

Examples

```r
p <- ggplot(sun.spct) + geom_line()
p + autotitle(sun.spct)
p + autotitle(sun.spct, annotations = "title:what")
p + autotitle(sun.spct, annotations = "title:where:when")
p + autotitle(sun.spct, annotations = "title:none:none:comment")
```

A_internal_label  
Absorbance axis labels

Description

Generate cps axis labels in SI units, using SI scale factors. Output can be selected as character, expression (R default devices) or LaTeX (for tikz device).

Usage

```r
A_internal_label(
  unit.exponent = 0,
  format = getOption("photobiology.math", default = "R.expression")
)
A_total_label(
  unit.exponent = 0,
  format = getOption("photobiology.math", default = "R.expression")
)
```
**black_or_white**

**Description**

Chose black or white color based on a color to be used as background. Usefull when using `geom_text` on top of tiles or bars, or `geom_label` with a variable fill.

**Usage**

```r
black_or_white(colors, threshold = 0.45)
```

**Arguments**

- `colors` character A vector of color definitions.
- `threshold` numeric in range 0 to 1.

**Examples**

```r
black_or_white("red")
black_or_white(colors()[1:10])
```
**color_chart**

Create a color checker chart

### Description

Color-checker-chart ggplot labelled with color names or with indexes of the colors in the vector passed as first argument.

### Usage

```r
color_chart(
  colors = grDevices::colors(),
  ncol = NULL,
  use.names = NULL,
  text.size = 2,
  text.color = NULL,
  grid.color = "white"
)
```

### Arguments

- **colors** character A vector of color definitions.
- **ncol** integer Number of column in the checker grid.
- **use.names** logical Force use of names or indexes.
- **text.size** numeric Size of the text labels drawn on each color tile.
- **text.color** character Color definition, used for text on tiles.
- **grid.color** character Color definition, used for grid lines between tiles.

### Note

Default `text.color` uses `black_or_white()` to ensure enough contrast. Default for `use.names` depends on number of columns in the grid, indexes are used when columns are seven or more.

### Examples

```r
color_chart()

color_chart(grep("dark", colors(), value = TRUE), text.size = 3.5)
```
counts_label

Raw-counts axis labels

Description
Generate axis labels in SI units, using SI scale factors. Output can be selected as character, expression (R default devices) or LaTeX (for tikz device).

Usage

```r
counts_label(
  unit.exponent = 3,
  format = getOption("photobiology.math", default = "R.expression")
)
```

Arguments

- `unit.exponent` integer
- `format` character string, "R", "R.expression", "R.character", or "LaTeX".

Value

a character string or an R expression.

Examples

```r
counts_label()
counts_label("R.expression")
counts_label("LaTeX")
```

cps_label

Counts-per-second axis labels

Description
Generate cps axis labels in SI units, using SI scale factors. Output can be selected as character, expression (R default devices) or LaTeX (for tikz device).

Usage

```r
cps_label(
  unit.exponent = 0,
  format = getOption("photobiology.math", default = "R.expression")
)
```
Arguments

- **unit.exponent**: integer
- **format**: character string, "R", "R.expression", "R.character", or "LaTeX".

Value

A character string or an R expression.

Examples

```r
cps_label()
cps_label(3)
cps_label(format = "R.expression")
cps_label(format = "LaTeX")
cps_label(3, format = "LaTeX")
```

exponent2prefix  SI unit prefixes

Description

Convert SI unit prefixes into exponents of ten of multipliers and vice-versa.

Usage

```r
exponent2prefix(
  exponent,
  char.set = getOption("photobiology.fancy.chars", default = "utf8")
)
```

```r
exponent2factor(exponent = 0, if.zero.exponent = "1")
```

```r
exponent2prefix_name(exponent)
```

```r
prefix_name2exponent(name)
```

```r
prefix2exponent(
  prefix,
  char.set = getOption("photobiology.fancy.chars", default = "utf8")
)
```

```r
has_SI_prefix(exponent)
```

```r
nearest_SI_exponent(exponent)
```
Arguments

- **exponent**: numeric The power of 10 of the unit multiplier.
- **char.set**: character How to encode Greek letters and other fancy characters in prefixes: "utf8", "ascii", "LaTeX".
- **if.zero.exponent**: character string to return when exponent is equal to zero.
- **name**: character Long SI name of multiplier.
- **prefix**: character Unit prefix used for multiplier.

Note

To change the default char.set, set R option "photobiology.fancy.chars". Implementation is based on a table of data and extensible to any alphabet supported by R character objects by expanding the table.

Examples

```r
exponent2prefix(3)
exponent2prefix(0)
exponent2prefix(-6)
```

```r
exponent2Factor(3)
exponent2Factor(0)
exponent2Factor(0, NULL)
exponent2Factor(0,"")
exponent2Factor(-6)
```

---

**geom_spct**  
*Spectral data plots.*

Description

For each continuous x value, **geom_spct** displays a y interval. **geom_spct** is a special case of **geom_area**, where the minimum of the range is fixed to 0, but stacking is not enabled.

Usage

```r
geom_spct(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
```
Arguments

- **mapping**: The aesthetic mapping, usually constructed with `aes` or `aes_`. Only needs to be set at the layer level if you are overriding the plot defaults.
- **data**: A data frame. If specified, overrides the default data frame defined at the top level of the plot.
- **stat**: The statistical transformation to use on the data for this layer, as a string.
- **position**: Position adjustment, either as a string, or the result of a call to a position adjustment function.
- **na.rm**: If FALSE (the default), removes missing values with a warning. If TRUE silently removes missing values.
- **show.legend**: logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
- **inherit.aes**: If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. `borders`.
- **...**: other arguments passed on to `layer`. This can include aesthetics whose values you want to set, not map. See `layer` for more details.

Details

An spectrum plot is the analog of a line plot (see `geom_path`), and can be used to show how y varies over the range of x. The difference is that the area under the line is filled.

Aesthetics

See `geom_ribbon`

See Also

`geom_ribbon` for stacked areas, `geom_path` for lines (lines), `geom_point` for scatter plots.

Examples

```
# ggplot() methods for spectral objects set a default mapping for x and y.
ggplot(sun.spct) + geom_spct()
```
ggplot

Create a new ggplot plot from spectral data.

Description

`ggplot()` initializes a ggplot object. It can be used to declare the input spectral object for a graphic and to optionally specify the set of plot aesthetics intended to be common throughout all subsequent layers unless specifically overridden.

Usage

```r
## S3 method for class 'source_spct'
ggplot(
  data,
  mapping = NULL,
  ...,
  range = NULL,
  unit.out = getOption("photobiology.radiation.unit", default = "energy"),
  environment = parent.frame()
)
```

```r
## S3 method for class 'response_spct'
 ggplot(
  data,
  mapping = NULL,
  ...,
  range = NULL,
  unit.out = getOption("photobiology.radiation.unit", default = "energy"),
  environment = parent.frame()
)
```

```r
## S3 method for class 'filter_spct'
 ggplot(
  data,
  mapping = NULL,
  ...,
  range = NULL,
  plot.qty = getOption("photobiology.filter.qty", default = "transmittance"),
  environment = parent.frame()
)
```

```r
## S3 method for class 'reflector_spct'
 ggplot(
  data,
  mapping = NULL,
  ...,
  range = NULL,
```

plot.qty = NULL,
environment = parent.frame()
)

## S3 method for class 'cps_spct'
ggplot(data, mapping = NULL, ..., range = NULL, environment = parent.frame())

## S3 method for class 'calibration_spct'
ggplot(data, mapping = NULL, ..., range = NULL, environment = parent.frame())

## S3 method for class 'raw_spct'
ggplot(data, mapping = NULL, ..., range = NULL, environment = parent.frame())

## S3 method for class 'object_spct'
ggplot(
  data,
  mapping = NULL,
  ...,
  range = NULL,
  plot.qty = getOption("photobiology.object.qty", default = "all"),
  environment = parent.frame()
)

## S3 method for class 'generic_mspct'
ggplot(data, mapping = NULL, ..., range = NULL, environment = parent.frame())

## S3 method for class 'filter_mspct'
ggplot(
  data,
  mapping = NULL,
  ...,
  range = NULL,
  plot.qty = getOption("photobiology.filter.qty", default = "transmittance"),
  environment = parent.frame()
)

## S3 method for class 'source_mspct'
ggplot(
  data,
  mapping = NULL,
  ...,
  range = NULL,
  unit.out = getOption("photobiology.radiation.unit", default = "energy"),
  environment = parent.frame()
Arguments

data  Default spectrum dataset to use for plot. If not a spectrum, the methods used will be those defined in package ggplot2. See ggplot. If not specified, must be supplied in each layer added to the plot.

mapping  Default list of aesthetic mappings to use for plot. If not specified, in the case of spectral objects, a default mapping will be used.

...  Other arguments passed on to methods.
range  an R object on which range() returns a vector of length 2, with min and max wavelengths (nm).

unit.out  character string indicating type of units to use for plotting spectral irradiance or spectral response, "photon" or "energy".

environment  If a variable defined in the aesthetic mapping is not found in the data, ggplot will look for it in this environment. It defaults to using the environment in which ggplot() is called.

plot.qty  character string One of "transmittance", "absorptance" or "absorbance" for filter_spct objects, and in addition to these "reflectance", "all" or "as.is" for object_spct objects.

Details

ggplot() is typically used to construct a plot incrementally, using the + operator to add layers to the existing ggplot object. This is advantageous in that the code is explicit about which layers are added and the order in which they are added. For complex graphics with multiple layers, initialization with ggplot is recommended.

We show seven common ways to invoke ggplot for spectra and collections of spectra:

- ggplot(spct)
- ggplot(spct,unit.out = <unit.to.use>)
- ggplot(spct,plot.qty = <quantity.to.plot>)
- ggplot(spct,range = <wavelength.range>)
- ggplot(spct) + aes(<other aesthetics>)
- ggplot(spct,aes(x,y,<other aesthetics>))
- ggplot(spct,aes())

The first method is recommended if all layers use the same data and the same set of automatic default x and y aesthetics. The second, third and fourth use automatic default x and y aesthetics but first transform or trim the spectral data to be plotted. The fifth uses automatic default x and y aesthetics and adds mappings for other aesthetics. These patterns can be combined as needed. The sixth disables the use of a default automatic mapping, while the seventh delays the mapping of aesthetics and can be convenient when using different mappings for different geoms.
Object spectra

In the case of class object_spct, the arguments "all" and "as.is" if passed to plot.qty, indicate in the first case that the data are to be converted into long form, to allow stacking, while in the second case data is copied unchanged to the plot object. "reflectance" passed to plot.qty converts data into a reflector_spct object and "absorbance", "absorptance", and "reflectance", convert data into a filter_spct.

Collections of spectra

The method for collections of spectra accepts arguments for the same parameters as the corresponding methods for single spectra. Heterogeneous generic collections of spectra are not supported. When plotting collections of spectra the factor spct.idx contains as levels the names of the individual members of the collection, and can be mapped to aesthetics or used for faceting.

Note

Current implementation does not merge the default mapping with user supplied mapping. If user supplies a mapping, it is used as is, and variables should be present in the spectral object. In contrast, when using the default mapping, unit or quantity conversions are done on the fly when needed. To add to the default mapping, aes() can be used by itself to compose the ggplot. In all cases, except when an object_spct is converted into long form, the data member of the returned plot object retains its class and attributes.

plot.qty is ignored for reflectors.

Examples

```r
ggplot(sun.spct) + geom_line()
ggplot(sun.spct, unit.out = "photon") + geom_line()

# For collections of spectra

ggplot(yellow_gel.spct) + geom_line()
ggplot(yellow_gel.spct, plot.qty = "absorbance") + geom_line()

ggplot(Ler_leaf.spct) + facet_grid(~variable) + geom_line()
ggplot(Ler_leaf.spct) + aes(linetype = variable) + geom_line()```

multiplot

Multiple plot function

Description

Grid based; allows multiple plots arranged in a matrix and printed to any R device. ggplot objects can be passed in ..., or to plotlist (as a list of ggplot objects)
Usage

```r
multiplot(
  ..., 
  plotlist = NULL,
  ncol = 1,
  cols = ncol,
  layout = NULL,
  title = "",
  title.position = "left",
  title.fontsize = 12,
  title.fontfamily = "sans",
  title.fontface = "bold",
  title.colour = "black"
)
```

Arguments

... one or more ggplot objects.
plotlist list of ggplot objects.
ncol, cols numerical Number of columns in layout.
layout A numeric matrix specifying the layout. If present, 'cols' is ignored.
title character vector Title of the composite plot.
title.position numeric or character, the horizontal position of the title.
title.fontsize numeric
title.fontfamily character e.g. "sans", "serif", "mono".
title.fontface character e.g. "plain", "bold", "italic", "bold.italic".
title.colour character e.g. "black", "red".

Details

ggplot objects can be passed in ..., or to plotlist (as a list of ggplot objects) If the layout is something like matrix(c(1,2,3,3), nrow=2, byrow=TRUE), then plot 1 will go in the upper left, 2 will go in the upper right, and 3 will go all the way across the bottom.

Note

Modified from example by Winston Chang found in the Cookbook for R Licence under CC BY-SA

References

http://www.cookbook-r.com/
plot.generic_spct

Create a complete ggplot for a spectrum.

Examples

```r
multiplot(plot(sun.spct), plot(yellow_gel.spct), ncol = 1)
multiplot(plot(sun.spct), plot(yellow_gel.spct), ncol = 1,
          title = "The sun and a yellow filter")
```

Description

This method returns a ggplot object with an annotated plot of an object of a class derived from `generic_spct` or of a class derived from `generic_mspct` for which a `plot()` method exists. It is implemented as a wrapper of `autoplot()`. This function is available for backwards compatibility, but new code should call this same function using method `autoplot()` instead.

Usage

```r
## S3 method for class 'generic_spct'
plot(x, ...)

## S3 method for class 'generic_mspct'
plot(x, ...)

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

Arguments

- `x` An R object derived from class `generic_spct` or derived from class `generic_mspct`.
- `...` Named arguments passed to `plot()` methods.

Details

Support for `autoplot()` method for consistency with package `ggplot2`. Please consult the documentation of the `plot()` methods for details about use of these autoplot methods. They are implemented as simple wrappers that forward the call to `plot()`.

Value

A `ggplot` object.

Note

The generic for this method is defined in package `ggplot2` and specializations for objects of diverse classes are provided by `ggplot2` and other packages.
Rfr_specular_label

Reflectance axis labels

Description
Generate spectral reflectance labels in SI units, using SI scale factors. Output can be selected as character, expression (R default devices) or LaTeX (for tikz device).

Usage
Rfr_specular_label
  unit.exponent = 0,
  format = getOption("photobiology.math", default = "R.expression")
)

Rfr_total_label
  unit.exponent = 0,
  format = getOption("photobiology.math", default = "R.expression")
)

Arguments
  unit.exponent integer
  format character string, "R", "R.expression", "R.character", or "LaTeX".

Value
  a character string or an R expression.

Examples
Rfr_specular_label()
Rfr_specular_label(-3)
Rfr_specular_label(format = "R.expression")
Rfr_specular_label(format = "LaTeX")
Rfr_specular_label(-3, format = "LaTeX")
\begin{verbatim}
Rfr_total_label()
Rfr_total_label(-2)
Rfr_total_label(-3)
Rfr_total_label(format = "R.expression")
Rfr_total_label(format = "LaTeX")
Rfr_total_label(-2, format = "LaTeX")
Rfr_total_label(-3, format = "LaTeX")
\end{verbatim}

\textbf{s.e.irrad_label} \hspace{1cm} \textit{spectral irradiance axis labels}

\section*{Description}

Generate axis labels in SI units, using SI scale factors. Output can be selected as character, expression (R default devices) or LaTeX (for tikz device).

\section*{Usage}

\begin{verbatim}
s.e.irrad_label(
  unit.exponent = 0,
  format = getOption("photobiology.math", default = "R.expression")
)

s.q.irrad_label(
  unit.exponent = -6,
  format = getOption("photobiology.math", default = "R.expression")
)
\end{verbatim}

\section*{Arguments}

\begin{itemize}
\item \textbf{unit.exponent} \hspace{1cm} integer
\item \textbf{format} \hspace{1cm} character string, "R", "R.expression", "R.character", or "LaTeX".
\end{itemize}

\section*{Value}

a character string or an R expression.

\section*{Examples}

\begin{verbatim}
counts_label()
counts_label("R.expression")
counts_label("LaTeX")
\end{verbatim}
**s.e.response_label**  
*spectral response axis labels*

**Description**
Generate axis labels in SI units, using SI scale factors. Output can be selected as character, expression (R default devices) or LaTeX (for tikz device).

**Usage**
```r
s.e.response_label(
  unit.exponent = 0,
  format = getOption("photobiology.math", default = "R.expression")
)
```
```r
s.q.response_label(
  unit.exponent = -6,
  format = getOption("photobiology.math", default = "R.expression")
)
```

**Arguments**
- `unit.exponent` integer
- `format` character string, "R", "R.expression", "R.character", or "LaTeX".

**Value**
a character string or an R expression.

**Examples**
```r
counts_label()
counts_label("R.expression")
counts_label("LaTeX")
```

**scale_x_wl_continuous**  
*Wavelength x-scale*

**Description**
Scale x continuous with defaults suitable for wavelengths in nanometres.
scale_x_wl_continuous

Usage

scale_x_wl_continuous(
    unit.exponent = -9,
    name = w_length_label(unit.exponent = unit.exponent),
    breaks = scales::pretty_breaks(n = 7),
    labels = SI_pl_format(exponent = unit.exponent + 9),
    ...
)

Arguments

unit.exponent  integer
name            The name of the scale, used for the axis-label.
b breaks        The positions of ticks or a function to generate them.
labels         The tick labels or a function to generate them from the tick positions.
...             other named arguments passed to scale_y_continuous

Note

This function only alters two default arguments, please, see documentation for scale_continuous

Examples

ggplot(sun.spct)  +
  geom_line()  +
  scale_x_wl_continuous()

ggplot(sun.spct)  +
  geom_line()  +
  scale_x_wl_continuous(-6)

ggplot(sun.spct)  +
  geom_line()  +
  scale_x_wl_continuous(sec.axis = sec_axis_w_number())

ggplot(sun.spct)  +
  geom_line()  +
  scale_x_wl_continuous(unit.exponent = -6,
                         sec.axis = sec_axis_w_number())
scale_y_A_internal_continuous

Absorbance y-scale

Description

Scale y continuous with defaults suitable for spectral absorbance.

Usage

```r
scale_y_A_internal_continuous(
  unit.exponent = 0,
  name = A_internal_label(unit.exponent = unit.exponent),
  labels = SI_pl_format(exponent = unit.exponent),
  ...
)
```

```r
scale_y_A_total_continuous(
  unit.exponent = 0,
  name = A_total_label(unit.exponent = unit.exponent),
  labels = SI_pl_format(exponent = unit.exponent),
  ...
)
```

Arguments

- `unit.exponent` integer
- `name` The name of the scale, used for the axis-label.
- `labels` The tick labels or a function to generate them.
- `...` other named arguments passed to `scale_y_continuous`

Note

This function only alters two default arguments, please, see documentation for `scale_continuous`

Examples

```r
ggplot(yellow_gel.spct, plot.qty = "absorbance") +
  geom_line() +
  scale_y_A_internal_continuous() +
  scale_x_wl_continuous()
```

```r
ggplot(yellow_gel.spct, plot.qty = "absorbance") +
  geom_line() +
  scale_y_A_total_continuous() +
  scale_x_wl_continuous()
```
scale_y_counts_continuous

*Raw-counts y-scale*

**Description**

Scale y continuous with defaults suitable for raw detector counts.

**Usage**

```r
scale_y_counts_continuous(
  unit.exponent = 3,
  name = counts_label(unit.exponent = unit.exponent),
  labels = SI_pl_format(exponent = unit.exponent),
  ...
)
```

```r
scale_y_counts_tg_continuous(
  unit.exponent = 3,
  name = counts_label(unit.exponent = 0),
  labels = SI_tg_format(exponent = unit.exponent),
  ...
)
```

**Arguments**

- `unit.exponent` integer
- `name` The name of the scale, used for the axis-label.
- `labels` The tick labels or a function to generate them.
- `...` other named arguments passed to `scale_y_continuous`

**Note**

This function only alters two default arguments, please, see documentation for `scale_continuous`

**Examples**

```r
ggplot(white_led.raw_spct) +
  geom_line() +
  scale_y_counts_continuous() +
  scale_x_wl_continuous()
```

```r
ggplot(white_led.raw_spct) +
  geom_line() +
  scale_y_counts_continuous(0) +
  scale_x_wl_continuous()
```
scale_y_cps_continuous

Counts-per-second y-scale

Description

Scale y continuous with defaults suitable for raw detector counts.

Usage

scale_y_cps_continuous(
    unit.exponent = 0,
    name = cps_label(unit.exponent = unit.exponent),
    labels = SI_pl_format(exponent = unit.exponent),
    ...
)

Arguments

unit.exponent integer
name The name of the scale, used for the axis-label.
labels The tick labels or a function to generate them.
... other named arguments passed to scale_y_continuous

Note

This function only alters two default arguments, please, see documentation for scale_continuous

Examples

ggplot(whiteLed.raw_spct) +
  geom_line() +
  scale_y_counts_tg_continuous() +
  scale_x_wl_continuous()

ggplot(whiteLed.cps_spct) +
  geom_line() +
  scale_y_cps_continuous() +
  scale_x_wl_continuous()

ggplot(whiteLed.cps_spct) +
  geom_line() +
  scale_y_cps_continuous(3) +
  scale_x_wl_continuous()
scale_y_Rfr_specular_continuous

Reflectance y-scale

Description

Scale y continuous with defaults suitable for spectral transmittance.

Usage

```r
scale_y_Rfr_specular_continuous(
  unit.exponent = 0,
  name = Rfr_specular_label(unit.exponent = unit.exponent),
  labels = SI_pl_format(exponent = unit.exponent),
  ...
)
```

```r
scale_y_Rfr_total_continuous(
  unit.exponent = 0,
  name = Rfr_total_label(unit.exponent = unit.exponent),
  labels = SI_pl_format(exponent = unit.exponent),
  ...
)
```

Arguments

- `unit.exponent` integer
- `name` The name of the scale, used for the axis-label.
- `labels` The tick labels or a function to generate them.
- `...` other named arguments passed to `scale_y_continuous`

Note

This function only alters two default arguments, please, see documentation for `scale_continuous`

Examples

```r
ggplot(yellow_gel.spct) +
  geom_line() +
  scale_y_Rfr_total_continuous() +
  scale_x_wl_continuous()
```

```r
ggplot(yellow_gel.spct) +
  geom_line() +
  scale_y_Rfr_total_continuous(-2) +
  scale_x_wl_continuous()
```
**scale_y.s.e.irrad_continuous**

*Spectral irradiance y-scale*

**Description**

Scale y continuous with defaults suitable for raw detector counts.

**Usage**

```r
scale_y.s.e.irrad_continuous(
  unit.exponent = 0,
  name = s.e.irrad_label(unit.exponent),
  labels = SI_pl_format(exponent = unit.exponent),
  ...
)
```

```r
scale_y.s.q.irrad_continuous(
  unit.exponent = -6,
  name = s.q.irrad_label(unit.exponent = unit.exponent),
  labels = SI_pl_format(exponent = unit.exponent),
  ...
)
```

```r
scale_y.s.q.irrad_log10(
  unit.exponent = -6,
  name = s.q.irrad_label(unit.exponent = unit.exponent),
  labels = SI_pl_format(exponent = unit.exponent),
  ...
)
```

```r
scale_y.s.e.irrad_log10(
  unit.exponent = 0,
  name = s.e.irrad_label(unit.exponent),
  labels = SI_pl_format(exponent = unit.exponent),
  ...
)
```
scale_y_s.e.response_continuous

**Arguments**

- **unit.exponent** integer
- **name** The name of the scale, used for the axis-label.
- **labels** The tick labels or a function to generate them.
  
  ... other named arguments passed to scale_y_continuous

**Note**

This function only alters two default arguments, please, see documentation for `scale_continuous`.

**Examples**

```r
ggplot(sun.spct) +
  geom_line() +
  scale_y_s.e.irrad_continuous() +
  scale_x_wl_continuous()

ggplot(sun.spct) +
  geom_line() +
  scale_y_s.e.irrad_continuous(-1) +
  scale_x_wl_continuous()

ggplot(sun.spct, unit.out = "photon") +
  geom_line() +
  scale_y_s.q.irrad_continuous() +
  scale_x_wl_continuous()

ggplot(clip_wl(sun.spct, c(295, NA))) +
  geom_line() +
  scale_y_s.e.irrad_log10() +
  scale_x_wl_continuous()

ggplot(clip_wl(sun.spct, c(295, NA)),
  unit.out = "photon") +
  geom_line(na.rm = TRUE) +
  scale_y_s.q.irrad_log10() +
  scale_x_wl_continuous()
```

**scale_y_s.e.response_continuous**

*Spectral response y-scale*

**Description**

Scale y continuous with defaults suitable for raw detector counts.
Usage

scale_y_s.e.response_continuous(
  unit.exponent = 0,
  name = s.e.response_label(unit.exponent),
  labels = SI_pl_format(exponent = -unit.exponent),
  ...)

scale_y_s.q.response_continuous(
  unit.exponent = 0,
  name = s.q.response_label(unit.exponent = unit.exponent),
  labels = SI_pl_format(exponent = -unit.exponent),
  ...)

Arguments

unit.exponent integer
name The name of the scale, used for the axis-label.
labels The tick labels or a function to generate them.
... other named arguments passed to scale_y_continuous

Note

This function only alters two default arguments, please, see documentation for scale_continuous

Examples

ggplot(ccd.spct) +
  geom_line() +
  scale_y_s.e.response_continuous(unit.exponent = 6) +
  scale_x_wl_continuous()

ggplot(ccd.spct, unit.out = "photon") +
  geom_line() +
  scale_y_s.q.response_continuous() +
  scale_x_wl_continuous()

scale_y_Tfr_internal_continuous

Transmittance y-scale

Description

Scale y continuous with defaults suitable for spectral transmittance.
Usage

\[
\text{scale\_y\_Tfr\_internal\_continuous}(\text{unit.exponent = 0,} \\
\text{name = Tfr\_internal\_label(unit.exponent = unit.exponent),} \\
\text{labels = SI\_pl\_format(exponent = unit.exponent),} \\
\text{...})
\]

\[
\text{scale\_y\_Tfr\_total\_continuous}(\text{unit.exponent = 0,} \\
\text{name = Tfr\_total\_label(unit.exponent = unit.exponent),} \\
\text{labels = SI\_pl\_format(exponent = unit.exponent),} \\
\text{...})
\]

Arguments

- unit.exponent: integer
- name: The name of the scale, used for the axis-label.
- labels: The tick labels or a function to generate them.
- ... other named arguments passed to scale\_y\_continuous

Note

This function only alters two default arguments, please, see documentation for scale\_continuous

Examples

```r
ggplot(yellow_gel.spct) +
  geom_line() +
  scale_y_Tfr_internal_continuous() +
  scale_x_wl_continuous()
```

```r
ggplot(yellow_gel.spct) +
  geom_line() +
  scale_y_Tfr_internal_continuous(-2) +
  scale_x_wl_continuous()
```

```r
ggplot(yellow_gel.spct) +
  geom_line() +
  scale_y_Tfr_internal_continuous(-3) +
  scale_x_wl_continuous()
```

```r
ggplot(yellow_gel.spct) +
  geom_line() +
  scale_y_Tfr_total_continuous() +
  scale_x_wl_continuous()
```
sec_axis_w_number

Secondary axes for wavelength data in nanometres. With suitable scaling and name (axis label) for frequency and wavenumber.

Usage

sec_axis_w_number(unit.exponent = -6)

sec_axis_w_frequency(unit.exponent = 12)

Arguments

unit.exponent integer

Examples

ggplot(sun.spct) +
  geom_line() +
  scale_x_continuous(name = w_length_label(),
                     sec.axis = sec_axis_w_number())

ggplot(sun.spct) +
  geom_line() +
  scale_x_continuous(name = w_length_label(),
                     sec.axis = sec_axis_w_number(-4))

ggplot(sun.spct) +
  geom_line() +
  scale_x_continuous(name = w_length_label(),
                     sec.axis = sec_axis_w_number(nearest_SI_exponent(-4)))

ggplot(sun.spct) +
  geom_line() +
  scale_x_continuous(name = w_length_label(),
                     sec.axis = sec_axis_w_number(-3))

ggplot(sun.spct) +
  geom_line() +
  scale_x_continuous(name = w_length_label(),
                     sec.axis = sec_axis_w_frequency())
set_annotations_default

Set defaults for autoplot annotations

Description

Set R options used when plotting spectra. Option "photobiology.plot.annotations" is used as default argument to formal parameter annotations and option "photobiology.plot.bands" is used as default argument to formal parameter w.band in all the autoplot() methods exported from package 'ggspectra'. These convenience functions makes it easier to edit these two option which are stored as a vector of characters strings and a list of waveband objects, respectively.

Usage

```r
set_annotations_default(annotations = NULL)
set_w.band_default(w.band = NULL)
```

Arguments

- **annotations** character vector Annotations to add or remove from defaults used by the autoplot() methods defined in this package..
- **w.band** a single waveband object or a list of waveband objects.

Details

Vectors of character strings passed as argument to annotations are parsed so that if the first member string is "+", the remaining members are added to the current default for annotations; if it is "-" the remaining members are removed from the current default for annotations; and if it is "=" the remaining members become the new default. If the first member is none of these three strings, the whole vector becomes the new default. If annotations is NULL the annotations are reset to the package defaults. When removing annotations "title*", "peaks*" and "valleys*" will remove any variation of these annotations. The string "" means no annotations while "reserve.space" means no annotations but expand y scale to reserve space for annotations. These two values take precedence over any other values in the character vector. The order of the names of annotations has no meaning: the vector is interpreted as a set except for the three possible "operators" at position 1.

Value

Previous value of option "photobiology.plot.annotations" returned invisibly.

Note

The syntax used and behaviour are the same as for the annotations parameter of the autoplot() methods for spectra, but instead of affecting a single plot, set_annotations_default() changes the default used for subsequent calls to autoplot().
**SI_pl_format**

*Formatter for plain labels discounting for SI multipliers*

**Description**

The labels generated represent numbers rescaled to compensate for a change in unit’s by a factor of ten or by a power of ten.

**Usage**

```
SI_pl_format(exponent = 0, digits = 3, ...)  
SI_plain(x, exponent = 0, digits = 3, ...)  
```

**Arguments**

- **exponent**: numeric Power of 10 to use as multiplier
- **digits**: number of significant digits to show
- **...**: other arguments passed on to *format*
- **x**: a numeric vector to format

**Value**

a function with single parameter *x*, a numeric vector, that returns a character vector

**Examples**

```
SI_pl_format()(1:10)  
SI_pl_format()(runif(10))  
SI_pl_format(exponent = 2)(runif(10))  
SI_plain(1:10)  
SI_plain(runif(10))  
SI_plain(runif(10), digits = 2)  
```
**SI_tg_format**  
*Formatter for tagged labels using SI multipliers*

**Description**

The labels generated represent the same numbers, but with trailing zeros removed/added and compensated by attaching to each label an SI multiplier "prefix".

**Usage**

```r
SI_tg_format(exponent = 0, digits = 3, ...)
SI_tagged(x, exponent = 0, digits = 3, ...)
```

**Arguments**

- `exponent` numeric Power of 10 to use as multiplier
- `digits` number of significant digits to show
- `...` other arguments passed on to `format`
- `x` a numeric vector to format

**Value**

a function with single parameter `x`, a numeric vector, that returns a character vector

**Note**

If the exponent passed has no SI prefix defined, the exponent will be adjusted to match one.

**Examples**

```r
SI_tg_format()(1:10)
SI_tg_format()(runif(10))
SI_tg_format(exponent = 2)(runif(10))
SI_tagged(1:10)
SI_tagged(runif(10))
SI_tagged(runif(10), digits = 2)
```
Calculate colours from wavelength.

Description

`stat_color` computes color definitions according to human vision.

Usage

```r
stat_color(
  mapping = NULL,
  data = NULL,
  geom = "point",
  chroma.type = "CMF",
  position = "identity",
  na.rm = FALSE,
  show.legend = FALSE,
  inherit.aes = TRUE,
  ...
)
```

Arguments

- **mapping**: The aesthetic mapping, usually constructed with `aes` or `aes_`. Only needs to be set at the layer level if you are overriding the plot defaults.
- **data**: A layer specific dataset - only needed if you want to override the plot defaults.
- **geom**: The geometric object to use display the data.
- **chroma.type**: character one of "CMF" (color matching function) or "CC" (color coordinates) or a `chroma_spct` object.
- **position**: The position adjustment to use for overlapping points on this layer.
- **na.rm**: a logical value indicating whether NA values should be stripped before the computation proceeds.
- **show.legend**: logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
- **inherit.aes**: If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. `borders`.
- **...**: other arguments passed on to `layer`. This can include aesthetics whose values you want to set, not map. See `layer` for more details.

Value

The original data frame with a variable with color definitions added.
Computed variable

\textbf{wl.color}  color corresponding to x-value giving wavelength in nanometres.

Default aesthetics

Set by the statistic and available to geoms.

\begin{itemize}
  \item \textbf{color}  \textit{..wl.color..}
  \item \textbf{fill}  \textit{..wl.color..}
\end{itemize}

Required aesthetics

Required by the statistic and need to be set with \texttt{aes()}.

\begin{itemize}
  \item \textbf{x}  numeric, wavelength in nanometres
  \item \textbf{y}  numeric, a spectral quantity
\end{itemize}

See Also

\texttt{color_of}, which is used internally.

Other stats functions: \texttt{stat_find_qtys()}, \texttt{stat_find_wls()}, \texttt{stat_label_peaks()}, \texttt{stat_peaks()}, \texttt{stat_spikes()}, \texttt{stat_wb_box()}, \texttt{stat_wb_column()}, \texttt{stat_wb_contribution()}, \texttt{stat_wb_hbar()}, \texttt{stat_wb_irrad()}, \texttt{stat_wb_label()}, \texttt{stat_wb_mean()}, \texttt{stat_wb_relative()}, \texttt{stat_wb_sirrad()}, \texttt{stat_wb_total()}, \texttt{stat_wl_strip()}, \texttt{stat_wl_summary()}

Examples

\begin{verbatim}
ggplot(sun.spect) + geom_line() +
  stat_color() + scale_color_identity()
\end{verbatim}

\begin{verbatim}
stat_find_qtys  Find quantity value for target wavelength value.
\end{verbatim}

Description

\texttt{stat_find_qtys} finds at which y positions values equal to an x target are located.

Usage

\begin{verbatim}
stat_find_qtys(
  mapping = NULL,
  data = NULL,
  geom = "point",
  target = "half.maximum",
  interpolate = TRUE,
)\end{verbatim}
stat_find_qtys

chroma.type = "CMF",
label.fmt = "%3g",
x.label.fmt = label.fmt,
y.label.fmt = label.fmt,
position = "identity",
na.rm = FALSE,
show.legend = FALSE,
inherit.aes = TRUE,
...
)

Arguments

mapping
The aesthetic mapping, usually constructed with aes or aes_. Only needs to be
set at the layer level if you are overriding the plot defaults.
data
A layer specific dataset - only needed if you want to override the plot defaults.
geom
The geometric object to use display the data
target
numeric value indicating the spectral quantity value for which wavelengths are
to be searched and interpolated if need. The character string "half.maximum" is
also accepted as argument.
interpolate
logical Indicating whether the nearest wavelength value in x should be returned
or a value calculated by linear interpolation between wavelength values straddling the target.
chroma.type
character one of "CMF" (color matching function) or "CC" (color coordinates)
or a chroma_spct object.
label.fmt
character string giving a format definition for converting values into character
strings by means of function sprintf.
x.label.fmt
character string giving a format definition for converting $x$-values into character
strings by means of function sprintf.
y.label.fmt
character string giving a format definition for converting $y$-values into character
strings by means of function sprintf.
position
The position adjustment to use for overlapping points on this layer
na.rm
a logical value indicating whether NA values should be stripped before the compu-
tation proceeds.
show.legend
logical. Should this layer be included in the legends? NA, the default, includes if
any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes
If FALSE, overrides the default aesthetics, rather than combining with them.
This is most useful for helper functions that define both data and aesthetics and
shouldn't inherit behaviour from the default plot specification, e.g. borders.
...
other arguments passed on to layer. This can include aesthetics whose values
you want to set, not map. See layer for more details.
Details

These stats use `geom_point` by default as it is the geom most likely to work well in almost any situation without need of tweaking. The default aesthetics set by these stats allow their direct use with `geom_text`, `geom_label`, `geom_line`, `geom_rug`, `geom_hline` and `geom_vline`. The formatting of the labels returned can be controlled by the user.

Value

A data frame with one row for each match to the target subset from the data or interpolated. As spectra are monotonic in wavelength, this statistic will never return more than one row when used with spectra.

Computed variables

- **x**: x-value at or nearest to the match to the target as numeric
- **y**: target value or y-value nearest to the target as numeric
- **x.label**: x-value at or nearest to the match formatted as character
- **y.label**: target value or y-value nearest to the target formatted as character
- **color**: color definition calculated by assuming that x-values are wavelengths expressed in nanometres.

Default aesthetics

Set by the statistic and available to geoms.

- **label**: `..x.label..`
- **xintercept**: `..x..`
- **yintercept**: `..y..`
- **fill**: `..color..`

Required aesthetics

Required by the statistic and need to be set with `aes()`.

- **x**: numeric, wavelength in nanometres
- **y**: numeric, a spectral quantity

Note

These stats work nicely together with geoms `geom_text_repel` and `geom_label_repel` from package `ggrepel` to solve the problem of overlapping labels by displacing them. To discard overlapping labels use `check_overlap = TRUE` as argument to `geom_text`. By default the labels are character values suitable to be plotted as is, but with a suitable `label.fmt` labels suitable for parsing by the geoms (e.g. into expressions containing greek letters or super or subscripts) can be also easily obtained.
See Also

find_peaks.

Other stats functions: stat_color(), stat_find_wls(), stat_label_peaks(), stat_peaks(),
stat_spikes(), stat_wb_box(), stat_wb_column(), stat_wb_contribution(), stat_wb_hbar(),
stat_wb_irrad(), stat_wb_label(), stat_wb_mean(), stat_wb_relative(), stat_wb_sirrad(),
stat_wb_total(), stat_wl_strip(), stat_wl_summary()

Examples

# ggplot() methods for spectral objects set a default mapping for x and y.
ggplot(yellow_gel.spct) + geom_line() +
  stat_find_wls(target = "half.range")
ggplot(yellow_gel.spct) + geom_line() +
  stat_find_wls(target = c(490, 500, 510))
ggplot(yellow_gel.spct) + geom_line() +
  stat_find_wls(target = 500, geom = "point", colour = "red") +
  stat_find_wls(target = 500, geom = "text", colour = "red",
                hjust = 1.1, label.fmt = "Tfr = %1.2f")

stat_find_wls  Find wavelength for target quantity value.

Description

stat_find_wls finds at which x positions values equal to a target are located.

Usage

stat_find_wls(
  mapping = NULL,
  data = NULL,
  geom = "point",
  target = "half.maximum",
  interpolate = TRUE,
  chroma.type = "CMF",
  label.fmt = "%3g",
  x.label.fmt = label.fmt,
  y.label.fmt = label.fmt,
  position = "identity",
  na.rm = FALSE,
  show.legend = FALSE,
  inherit.aes = TRUE,
  ...
Arguments

mapping  The aesthetic mapping, usually constructed with `aes` or `aes_`. Only needs to be set at the layer level if you are overriding the plot defaults.
data     A layer specific dataset - only needed if you want to override the plot defaults.
geom      The geometric object to use display the data
target   numeric vector indicating the spectral quantity values for which wavelengths are to be searched and interpolated if need. The character strings "half.maximum" and "half.range" are also accepted as arguments. A list with numeric and/or character values is also accepted.
interpolate logical Indicating whether the nearest wavelength value in x should be returned or a value calculated by linear interpolation between wavelength values straddling the target.
chroma.type character one of "CMF" (color matching function) or "CC" (color coordinates) or a `chroma_spct` object.
label.fmt character string giving a format definition for converting values into character strings by means of function `sprintf`.
x.label.fmt character string giving a format definition for converting $x$-values into character strings by means of function `sprintf`.
y.label.fmt character string giving a format definition for converting $y$-values into character strings by means of function `sprintf`.
position The position adjustment to use for overlapping points on this layer
na.rm     a logical value indicating whether NA values should be stripped before the computation proceeds.
show.legend logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. `borders`.
...     other arguments passed on to `layer`. This can include aesthetics whose values you want to set, not map. See `layer` for more details.

Details

These stats use `geom_point` by default as it is the geom most likely to work well in almost any situation without need of tweaking. The default aesthetics set by these stats allow their direct use with `geom_text`, `geom_label`, `geom_line`, `geom_rug`, `geom_hline` and `geom_vline`. The formatting of the labels returned can be controlled by the user.

Value

A data frame with one row for each match to `target` found in the data.
Computed variables

- **x**  x-value at or nearest to the match to the target as numeric
- **y**  target value or y-value nearest to the target as numeric
- **x.label**  x-value at or nearest to the match formatted as character
- **y.label**  target value or y-value nearest to the target formatted as character
- **wl.color**  color definition calculated by assuming that x-values are wavelengths expressed in nanometers.

Default aesthetics

Set by the statistic and available to geoms.

- **label**  ..x.label..
- **xintercept**  ..x..
- **yintercept**  ..y..
- **fill**  ..wl.color..

Required aesthetics

Required by the statistic and need to be set with `aes()`.

- **x**  numeric, wavelength in nanometres
- **y**  numeric, a spectral quantity

Note

These stats work nicely together with geoms `geom_text_repel` and `geom_label_repel` from package `ggrepel` to solve the problem of overlapping labels by displacing them. To discard overlapping labels use `check_overlap = TRUE` as argument to `geom_text`. By default the labels are character values suitable to be plotted as is, but with a suitable `label.fmt` labels suitable for parsing by the geoms (e.g. into expressions containing greek letters or super or subscripts) can be also easily obtained.

See Also

- `find_peaks`

Other stats functions: `stat_color()`, `stat_find_qtys()`, `stat_label_peaks()`, `stat_peaks()`, `stat_spikes()`, `stat_wb_box()`, `stat_wb_column()`, `stat_wb_contribution()`, `stat_wb_hbar()`, `stat_wb_irrad()`, `stat_wb_label()`, `stat_wb_mean()`, `stat_wb_relative()`, `stat_wb_sirrad()`, `stat_wb_total()`, `stat_wl_strip()`, `stat_wl_summary()`
Examples

```r
# ggplot() methods for spectral objects set a default mapping for x and y.
ggplot(yellow_gel.spct) + geom_line() +
  stat_find_wls(target = c(0.25, 0.5, 0.75))

ggplot(yellow_gel.spct) + geom_line() +
  stat_find_wls(target = "half.maximum", geom = "point", colour = "red") +
  stat_find_wls(target = "half.maximum", geom = "text", colour = "red",
                hjust = 1.1, label.fmt = "%3.0f nm")
```

### stat_label_peaks

Label peaks and valleys.

Description

`stat_labels_peaks` finds at which x positions local maxima are located, and adds labels and colors to the data without subsetting. To find local minima, you can use `stat_labels_valleys` instead.

Usage

```r
stat_label_peaks(
  mapping = NULL,
  data = NULL,
  geom = "text",
  span = 5,
  ignore_threshold = 0,
  strict = TRUE,
  chroma.type = "CMF",
  label.fmt = "%.3g",
  x.label.fmt = label.fmt,
  y.label.fmt = label.fmt,
  label.fill = "",
  position = "identity",
  na.rm = TRUE,
  show.legend = FALSE,
  inherit.aes = TRUE,
  ...
)
```

```r
stat_label_valleys(
  mapping = NULL,
  data = NULL,
  geom = "text",
  span = 5,
  ignore_threshold = 0,
  strict = TRUE,
  chroma.type = "CMF",
  label.fmt = "%.3g",
  x.label.fmt = label.fmt,
  y.label.fmt = label.fmt,
  label.fill = "",
  position = "identity",
  na.rm = TRUE,
  show.legend = FALSE,
  inherit.aes = TRUE,
  ...
)
```
label.fmt = "%.3g",
x.label.fmt = label.fmt,
y.label.fmt = label.fmt,
label.fill = "",
position = "identity",
na.rm = TRUE,
show.legend = FALSE,
inherit.aes = TRUE,
...)

Arguments

mapping The aesthetic mapping, usually constructed with \texttt{aes} or \texttt{aes_}. Only needs to be set at the layer level if you are overriding the plot defaults.
data A layer specific dataset - only needed if you want to override the plot defaults.
geom The geometric object to use display the data
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 5, meaning that a peak is bigger than two consecutive neighbors on each side. Default: 5.
ignore_threshold numeric value between 0.0 and 1.0 indicating the size threshold below which peaks will be ignored.
strict logical flag: if TRUE, an element must be strictly greater than all other values in its window to be considered a peak. Default: FALSE.
chroma.type character one of "CMF" (color matching function) or "CC" (color coordinates) or a \texttt{chromaспект} object.
label.fmt character string giving a format definition for converting values into character strings by means of function \texttt{sprintf}.
x.label.fmt character string giving a format definition for converting $x$-values into character strings by means of function \texttt{sprintf}.
y.label.fmt character string giving a format definition for converting $y$-values into character strings by means of function \texttt{sprintf}.
label.fill character string ot use for labels not at peaks or valleys being highlighted.
position The position adjustment to use for overlapping points on this layer
na.rm a logical value indicating whether NA values should be stripped before the computation proceeds.
show.legend logical. Should this layer be included in the legends? \texttt{NA}, the default, includes if any aesthetics are mapped. \texttt{FALSE} never includes, and \texttt{TRUE} always includes.
inherit.aes If \texttt{FALSE}, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. \texttt{borders}.
... other arguments passed on to \texttt{layer}. This can include aesthetics whose values you want to set, not map. See \texttt{layer} for more details.
Details

These stats use `geom_text` by default as it is the geom most likely to work well in almost any situation without need of tweaking. These statistics work best with `geom_text_repel` and `geom_label_repel` from package 'ggrepel' as they are designed so that peak or valley labels will not overlap any observation in the whole data set. Default aesthetics set by these stats allow their direct use with `geom_text`, `geom_label`, `geom_line`, `geom_rug`, `geom_hline` and `geom_vline`. The formatting of the labels returned can be controlled by the user.

Value

The original data with additional computed variables added.

Computed variables

- **x.label** x-value at a peak (or valley) formatted as character or otherwise the value passed to `label.fill` which defaults to an empty string ("").
- **y.label** y-value at the peak (or valley) formatted as character or otherwise the value passed to `label.fill` which defaults to an empty string ("").
- **wl.color** At peaks and valleys, color definition calculated by assuming that x-values are wavelengths expressed in nanometres, otherwise, `rgb(1,1,1,0)` (transparent white).

Default aesthetics

Set by the statistic and available to geoms.

- **label** `.x.label.`
- **xintercept** `.x.`
- **yintercept** `.y.`
- **color** `black_or_white(.wl.color.)`
- **fill** `.wl.color.`

Required aesthetics

Required by the statistic and need to be set with `aes()`.

- **x** numeric, wavelength in nanometres
- **y** numeric, a spectral quantity

Note

These stats work nicely together with geoms `geom_text_repel` and `geom_label_repel` from package `ggrepel` to solve the problem of overlapping labels by displacing them. To discard overlapping labels use `check_overlap = TRUE` as argument to `geom_text`. By default the labels are character values suitable to be plotted as is, but with a suitable `label.fmt` labels suitable for parsing by the geoms (e.g. into expressions containing greek letters or super or subscripts) can be also easily obtained.
stat_peaks

Find peaks and valleys.

Description

stat_peaks finds at which x positions local maxima are located. If you want find local minima, you can use stat_valleys instead.

See Also

stat_peaks, stat_valleys and find_peaks, which is used internally.

Other stats functions: stat_color(), stat_find_qtys(), stat_find_wls(), stat_peaks(), stat_spikes(), stat_wb_box(), stat_wb_column(), stat_wb_contribution(), stat_wb_hbar(), stat_wb_irrad(), stat_wb_label(), stat_wb_mean(), stat_wb_relative(), stat_wb_sirrad(), stat_wb_total(), stat wl_strip(), stat wl_summary()

Examples

```r
# ggplot() methods for spectral objects set a default mapping for x and y.
ggplot(sun.spc) + geom_line() +
  stat_label_peaks(hjust = "left", span = 31, angle = 90, color = "red")
ggplot(sun.spc) + geom_line() +
  stat_label_valleys(hjust = "right", span = 21, angle = 90, color = "blue")

ggplot(sun.spc) + geom_line() +
  stat_peaks(span = 41, shape = 21, size = 3) +
  stat_label_peaks(span = 41, geom = "label", label.fmt = "%3.0f nm") +
  scale_fill_identity() +
  scale_color_identity() +
  expand_limits(y = c(NA, 1))

# using 'ggrepel' to avoid overlaps
# too slow for CRAN checks
## Not run:
library(ggrepel)

## End(Not run)
```
Usage

```r
stat_peaks(
  mapping = NULL,
  data = NULL,
  geom = "point",
  position = "identity",
  ..., 
  span = 5,
  ignore_threshold = 0.01,
  strict = is.null(span),
  refine.wl = FALSE,
  method = "spline",
  chroma.type = "CMF",
  label.fmt = "%3g",
  x.label.fmt = label.fmt,
  y.label.fmt = label.fmt,
  na.rm = FALSE,
  show.legend = FALSE,
  inherit.aes = TRUE
)
```

```r
stat_valleys(
  mapping = NULL,
  data = NULL,
  geom = "point",
  span = 5,
  ignore_threshold = -0.01,
  strict = is.null(span),
  refine.wl = FALSE,
  method = "spline",
  chroma.type = "CMF",
  label.fmt = "%3g",
  x.label.fmt = label.fmt,
  y.label.fmt = label.fmt,
  na.rm = FALSE,
  show.legend = FALSE,
  inherit.aes = TRUE,
  ...
)
```

Arguments

- **mapping** The aesthetic mapping, usually constructed with `aes` or `aes_`. Only needs to be set at the layer level if you are overriding the plot defaults.
- **data** A layer specific dataset - only needed if you want to override the plot defaults.
- **geom** The geometric object to use display the data
stat_peaks

position
The position adjustment to use for overlapping points on this layer. Other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.

span
integer 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. Use NULL for the global peak. Valleys are the reverse.

ignore_threshold
numeric For peaks, value between 0.0 and 1.0 indicating the relative size of peaks compared to tallest peak threshold below which peaks will be ignored, while negative values between 0.0 and -1.0 set a threshold so that the tallest peaks are ignored, instead of the shortest. For valleys, value between 0.0 and 1.0 indicating the relative depth of valleys below which valleys will be ignored, while negative values between 0.0 and -1.0 set a threshold so that the deeper valleys are ignored, instead of the shallower ones.

strict
logical If TRUE, an element must be strictly greater than all other values in its window to be considered a peak.

refine.wl
logical Flag indicating if peak or valleys locations should be refined by fitting a function.

method
character String with the name of a method used for peak fitting. Currently only spline interpolation is implemented.

chroma.type
character one of "CMF" (color matching function) or "CC" (color coordinates) or a chroma_spct object.

label.fmt
character string giving a format definition for converting values into character strings by means of function sprintf.

x.label.fmt
character string giving a format definition for converting $x$-values into character strings by means of function sprintf.

y.label.fmt
character string giving a format definition for converting $y$-values into character strings by means of function sprintf.

na.rm
a logical value indicating whether NA values should be stripped before the computation proceeds.

show.legend
logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.

inherit.aes
If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. borders.

Details
These stats use geom_point by default as it is the geom most likely to work well in almost any situation without need of tweaking. The default aesthetics set by these stats allow their direct use with geom_text, geom_label, geom_line, geom_rug, geom_hline and geom_vline. The formatting of the labels returned can be controlled by the user.

Value
A data frame with one row for each peak (or valley) found in the data.
**Computed variables**

- **x** x-value at the peak (or valley) as numeric
- **y** y-value at the peak (or valley) as numeric
- **x.label** x-value at the peak (or valley) formatted as character
- **y.label** y-value at the peak (or valley) formatted as character
- **wl.color** color definition calculated by assuming that x-values are wavelengths expressed in nanometres.
- **BW.color** color definition, either "black" or "white", as needed to ensure high contrast to \( w_l.\)color.

**Default aesthetics**

Set by the statistic and available to geoms.

- **label** stat(x.label)
- **xintercept** stat(x)
- **yintercept** stat(y)
- **fill** stat(wl.color)

**Required aesthetics**

Required by the statistic and need to be set with \texttt{aes()}.

- **x** numeric, wavelength in nanometres
- **y** numeric, a spectral quantity

**Note**

These stats work nicely together with geoms \texttt{geom_text_repel} and \texttt{geom_label_repel} from package \texttt{ggrepel} to solve the problem of overlapping labels by displacing them. To discard overlapping labels use \texttt{check_overlap = TRUE} as argument to \texttt{geom_text}. By default the labels are character values suitable to be plotted as is, but with a suitable \texttt{label.fmt} labels suitable for parsing by the geoms (e.g. into expressions containing greek letters or super or subscripts) can be also easily obtained.

**See Also**

- \texttt{find_peaks}, which is used internally.
- Other stats functions: \texttt{stat_color()}, \texttt{stat_find_qtys()}, \texttt{stat_find_wls()}, \texttt{stat_label_peaks()}, \texttt{stat_spikes()}, \texttt{stat_wb_box()}, \texttt{stat_wb_column()}, \texttt{stat_wb_contribution()}, \texttt{stat_wb_box()}, \texttt{stat_wb_irrad()}, \texttt{stat_wb_label()}, \texttt{stat_wb_mean()}, \texttt{stat_wb_relative()}, \texttt{stat_wb_sirrad()}, \texttt{stat_wb_total()}, \texttt{stat wl_strip()}, \texttt{stat wl_summary()}
Examples

# ggplot() methods for spectral objects set a default mapping for x and y.
ggplot(sun.spct) +
  geom_line() +
  stat_peaks()

ggplot(sun.spct) +
  geom_line() +
  stat_valleys()

ggplot(sun.spct) +
  geom_line() +
  stat_peaks(span = 51, geom = "point", colour = "red") +
  stat_peaks(span = 51, geom = "text", colour = "red",
            vjust = -0.4, label.fmt = "%3.2f nm")

ggplot(sun.spct) +
  geom_line() +
  stat_peaks(span = 51, geom = "point", colour = "red", refine.wl = TRUE) +
  stat_peaks(span = 51, geom = "text", colour = "red",
            vjust = -0.4, label.fmt = "%3.2f nm",
            refine.wl = TRUE)

ggplot(sun.spct) +
  geom_line() +
  stat_peaks(span = 51, geom = "point", colour = "red", refine.wl = TRUE) +
  stat_peaks(mapping = aes(fill = stat(wl.colour), color = stat(BW.colour)),
            span = 51, geom = "label",
            vjust = -0.3, hjust = c(1, 0, 0.5), label.fmt = "%.6g nm",
            refine.wl = TRUE) +
  stat_valleys(span = 71, geom = "point", colour = "blue", refine.wl = TRUE) +
  stat_valleys(mapping = aes(fill = stat(wl.colour), color = stat(BW.colour)),
            span = 71, geom = "label",
            vjust = 1.3, hjust = 2/3, label.fmt = "%.6g nm",
            refine.wl = TRUE) +
  expand_limits(y = 1) +
  scale_fill_identity() +
  scale_color_identity()  

stat_spikes  Find spikes

Description

stat_spikes finds at which x positions spikes are located. Spikes can be either upwards or downwards from the baseline.
stat_spikes

Usage

stat_spikes(
mapping = NULL,
data = NULL,
geom = "point",
position = "identity",
..., 
z.threshold = 9,
max.spike.width = 8,
chroma.type = "CMF",
label.fmt = "%.3g",
x.label.fmt = label.fmt,
y.label.fmt = label.fmt,
na.rm = FALSE,
show.legend = FALSE,
inherit.aes = TRUE
)

Arguments

mapping The aesthetic mapping, usually constructed with aes or aes_. Only needs to be set at the layer level if you are overriding the plot defaults.
data A layer specific dataset - only needed if you want to override the plot defaults.
geom The geometric object to use display the data
position The position adjustment to use for overlapping points on this layer
... other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.
z.threshold numeric Modified Z values larger than z.threshold are considered to be spikes.
max.spike.width integer Wider regions with high Z values are not detected as spikes.
chroma.type character one of "CMF" (color matching function) or "CC" (color coordinates) or a chroma_spct object.
label.fmt character string giving a format definition for converting values into character strings by means of function sprintf.
x.label.fmt character string giving a format definition for converting $x$-values into character strings by means of function sprintf.
y.label.fmt character string giving a format definition for converting $y$-values into character strings by means of function sprintf.
na.rm a logical value indicating whether NA values should be stripped before the computation proceeds.
show.legend logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. borders.
Details

This stat uses `geom_point` by default as it is the geom most likely to work well in almost any situation without need of tweaking. The default aesthetics set by this stat allows its direct use with `geom_text`, `geom_label`, `geom_line`, `geom_rug`, `geom_hline` and `geom_vline`. The formatting of the labels returned can be controlled by the user.

Value

A data frame with one row for each peak (or valley) found in the data.

Computed variables

- **x** x-value at the peak (or valley) as numeric
- **y** y-value at the peak (or valley) as numeric
- **x.label** x-value at the peak (or valley) formatted as character
- **y.label** y-value at the peak (or valley) formatted as character
- **wl.color** color definition calculated by assuming that x-values are wavelengths expressed in nanometres.
- **BW.color** color definition that either "black" or "white", to ensure high contrast to `wl.color`.

Default aesthetics

Set by the statistic and available to geoms.

- **label** `stat(x.label)`
- **xintercept** `stat(x)`
- **yintercept** `stat(y)`
- **fill** `stat(wl.color)`

Required aesthetics

Required by the statistic and need to be set with `aes()`.

- **x** numeric, wavelength in nanometres
- **y** numeric, a spectral quantity

Note

This stat works nicely together with geoms `geom_text_repel` and `geom_label_repel` from package `ggrepel` to solve the problem of overlapping labels by displacing them. To discard overlapping labels use `check_overlap = TRUE` as argument to `geom_text`. By default the labels are character values suitable to be plotted as is, but with a suitable `label.fmt` labels suitable for parsing by the geoms (e.g. into expressions containing greek letters or super or subscripts) can be also easily obtained.
See Also

find_spikes, which is used internally, for a description of the algorithm used.

Other stats functions: stat_color(), stat_find_qtys(), stat_find_wls(), stat_label_peaks(),
stat_peaks(), stat_wb_box(), stat_wb_column(), stat_wb_contribution(), stat_wb_hbar(),
stat_wb_irrad(), stat_wb_label(), stat_wb_mean(), stat_wb_relative(), stat_wb_sirrad(),
stat_wb_total(), stat_wl_strip(), stat_wl_summary()

Examples

# ggplot() methods for spectral objects set a default mapping for x and y.

# two spurious(?) spikes
ggplot(sun.spekt) +
  geom_line() +
  stat_spikes(colour = "red", alpha = 0.3)

# no spikes detected
ggplot(sun.spekt) +
  geom_line() +
  stat_spikes(colour = "red", alpha = 0.3,
              max.spike.width = 3,
              z.threshold = 12)

# small noise spikes detected
ggplot(white_led.raw_spct) +
  geom_line() +
  stat_spikes(colour = "red", alpha = 0.3)

    ggplot(white_led.raw_spct) +
      geom_line() +
      stat_spikes(colour = "red", alpha = 0.3) +
      stat_spikes(geom = "text", colour = "red", check_overlap = TRUE,
                   vjust = -0.5, label.fmt = "%3.0f nm")

    ggplot(white_led.raw_spct, aes(w.length, counts_2)) +
      geom_line() +
      stat_spikes(colour = "red", alpha = 0.3,
                   max.spike.width = 3,
                   z.threshold = 12)

stat_wb_box

Draw colour boxes for wavebands

Description

stat_wb_box plots boxes corresponding to wavebands, by default located slightly above the peak
of the spectrum. Sets suitable default aesthetics for "rect" geom.
Usage

```r
stat_wb_box(
  mapping = NULL,
  data = NULL,
  geom = "rect",
  w.band = NULL,
  chroma.type = "CMF",
  ypos.mult = 1.07,
  ypos.fixed = NULL,
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)
```

Arguments

- **mapping**: The aesthetic mapping, usually constructed with `aes` or `aes_`. Only needs to be set at the layer level if you are overriding the plot defaults.
- **data**: A layer specific dataset - only needed if you want to override the plot defaults.
- **geom**: The geometric object to use display the data
- **w.band**: a waveband object or a list of waveband objects or numeric vector of at least length two.
- **chroma.type**: character one of "CMF" (color matching function) or "CC" (color coordinates) or a `chroma_spct` object.
- **ypos.mult**: numeric Multiplier constant used to scale returned y values.
- **ypos.fixed**: numeric If not NULL used a constant value returned in y.
- **position**: The position adjustment to use for overlapping points on this layer
- **na.rm**: a logical value indicating whether NA values should be stripped before the computation proceeds.
- **show.legend**: logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
- **inherit.aes**: If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. `borders`.
- **...**: other arguments passed on to `layer`. This can include aesthetics whose values you want to set, not map. See `layer` for more details.

Value

A data frame with one row for each waveband object in the argument to `w.band`. Wavebands outside the range of the spectral data are trimmed or discarded.
Computed variables

What it is named integral below is the result of applying `integral.fun` to the data, with default `integrate_xy`.

- **x**  w.band-midpoint
- **wb.xmin** w.band minimum
- **wb.xmax** w.band maximum
- **wb.ymin** data$y minimum
- **wbymax** data$y maximum
- **y**  ypos.fixed or top of data, adjusted by `ypos.mult`
- **wb.color** color of the w.band
- **wb.name** label of w.band
- **BW.color** `black_or_white(wb.color)`

Default aesthetics

Set by the statistic and available to geoms.

- **xmin** ..wb.xmin..
- **xmax** ..wb.xmax..
- **ymin** ..y.. - (..wb.ymax.. - ..wb.ymin..) * 0.03
- **ymax** ..y.. + (..wb.ymax.. - ..wb.ymin..) * 0.03
- **fill** ..wb.color..

Required aesthetics

Required by the statistic and need to be set with `aes()`.

- **x** numeric, wavelength in nanometres
- **y** numeric, a spectral quantity

Note

This stat uses a panel function and ignores grouping as it is meant to be used for annotations. The value returned as default value for y is based on the y-range of spectral values for the whole data set.

See Also

Other stats functions: `stat_color()`, `stat_find_qtys()`, `stat_find_wls()`, `stat_label_peaks()`, `stat_peaks()`, `stat_spikes()`, `stat_wb_column()`, `stat_wb_contribution()`, `stat_wb_hbar()`, `stat_wb_irrad()`, `stat_wb_label()`, `stat_wb_mean()`, `stat_wb_relative()`, `stat_wb_sirrad()`, `stat_wb_total()`, `stat_wl_strip()`, `stat_wl_summary()`
Examples

```r
library(photobiologyWavebands)
# ggplot() methods for spectral objects set a default mapping for x and y.
ggplot(sun.spct) +
  stat_wb_box(w.band = VIS_bands()) +
  geom_line() +
  scale_fill_identity()
ggplot(sun.spct) +
  stat_wb_box(w.band = VIS_bands(), color = "white") +
  geom_line() +
  scale_fill_identity()
```

---

**stat_wb_column**

*Integrate ranges under curve.*

**Description**

`stat_wb_column` computes means under a curve. It first integrates the area under a spectral curve and also the mean expressed per nanometre of wavelength for each waveband in the input. Sets suitable default aesthetics for "rect" geom.

**Usage**

```r
stat_wb_column(
  mapping = NULL,
  data = NULL,
  geom = "rect",
  w.band = NULL,
  integral.fun = integrate_xy,
  chroma.type = "CMF",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)
```

**Arguments**

- **mapping**: The aesthetic mapping, usually constructed with `aes` or `aes_*`. Only needs to be set at the layer level if you are overriding the plot defaults.
- **data**: A layer specific dataset - only needed if you want to override the plot defaults.
- **geom**: The geometric object to use display the data
- **w.band**: a waveband object or a list of waveband objects or numeric vector of at least length two.
integral.fun  
function on $x$ and $y$.

chroma.type  
character one of "CMF" (color matching function) or "CC" (color coordinates) or a chroma_spct object.

position  
The position adjustment to use for overlapping points on this layer

na.rm  
a logical value indicating whether NA values should be stripped before the computation proceeds.

show.legend  
logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.

inherit.aes  
If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. borders.

other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.

Value
A data frame with one row for each waveband object in the argument to w.band. Wavebands outside the range of the spectral data are trimmed or discarded.

Computed variables
What it is named integral below is the result of applying integral.fun, with default integrate_xy.

x  
w.band-midpoint

wb.xmin  
w.band minimum

wb.xmax  
w.band maximum

wb.ymin  
data_y minimum

wbymax  
data_y maximum

wb.ymean  
yint divided by wl_expanse(w.band)

y  
wb.ymean

wb.color  
color of the w.band

wb.name  
label of w.band

BW.color  
black_or_white(wb.color)

Default aesthetics
Set by the statistic and available to geoms.

xmin  
..wb.xmin..

xmax  
..wb.xmax..

ymin  
0

ymax  
..wb.ymean..

fill  
..wb.color..
Required aesthetics

Required by the statistic and need to be set with `aes()`.

- **x** numeric, wavelength in nanometres
- **y** numeric, a spectral quantity

Note

If the argument passed to `w.band` is a BSWF it is silently converted to a wavelength range and the average of spectral values without weighting is returned as default value for `ymax` while the default value for `ymin` is zero.

See Also

Other stats functions: `stat_color()`, `stat_find_qtys()`, `stat_find_wls()`, `stat_label_peaks()`, `stat_peaks()`, `stat_spikes()`, `stat_wb_box()`, `stat_wb_contribution()`, `stat_wb_hbar()`, `stat_wb_irrad()`, `stat_wb_label()`, `stat_wb_mean()`, `stat_wb_relative()`, `stat_wb_sirrad()`, `stat_wb_total()`, `stat_wl_strip()`, `stat_wl_summary()`

Examples

```r
library(photobiologyWavebands)
# ggplot() methods for spectral objects set a default mapping for x and y.
ggplot(sun.spct) +
  stat_wb_column(w.band = VIS_bands()) +
  geom_line() +
  scale_fill_identity()

ggplot(sun.spct) +
  stat_wb_column(w.band = VIS_bands(), alpha = 0.5) +
  geom_line() +
  scale_fill_identity()
```

Description

`stat_wb_contribution` computes means under a curve. It first integrates the area under a spectral curve and also the mean expressed per nanometre of wavelength for each waveband in the input. Sets suitable default aesthetics for "rect", "hline", "vline", "text" and "label" geoms displaying "contributions" per waveband to the total of the spectral integral.
stat_wb_contribution

Usage

stat_wb_contribution(
  mapping = NULL,
  data = NULL,
  geom = "text",
  w.band = NULL,
  integral.fun = integrate_xy,
  label.mult = 1,
  chroma.type = "CMF",
  label.fmt = "%.1f",
  ypos.mult = 1.07,
  ypos.fixed = NULL,
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)

Arguments

mapping  The aesthetic mapping, usually constructed with aes or aes_. Only needs to be set at the layer level if you are overriding the plot defaults.
data      A layer specific dataset - only needed if you want to override the plot defaults.
geom      The geometric object to use display the data
w.band    a waveband object or a list of waveband objects or numeric vector of at least length two.
integral.fun function on y and z.
label.mult numeric Scaling factor applied to y-integral values before conversion into character strings.
chroma.type character one of "CMF" (color matching function) or "CC" (color coordinates) or a chroma_spct object.
label.fmt character string giving a format definition for converting y-integral values into character strings by means of function sprintf.
ypos.mult numeric Multiplier constant used to scale returned y values.
ypos.fixed numeric If not NULL used a constant value returned in y.
position  The position adjustment to use for overlapping points on this layer
na.rm     a logical value indicating whether NA values should be stripped before the computation proceeds.
show.legend logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. borders.
...
other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.
Value

A data frame with one row for each waveband object in the argument to w.band. Wavebands outside the range of the spectral data are trimmed or discarded.

Computed variables

What it is named integral below is the result of applying `integral.fun` to the data, with default `integrate.xy`.

- **y.label**: yint multiplied by `label.mult` and formatted according to `label.fmt`
- **x**: w.band-midpoint
- **xmin**: w.band minimum
- **xmax**: w.band maximum
- **ymin**: `data$y` minimum
- **ymax**: `data$y` maximum
- **yint**: `data$y` integral for w.band / `data$y` integral for whole range of `data$x`
- **xmean**: yint divided by `wl_expanse(w.band)`
- **y**: `ypos.fixed` or top of data, adjusted by `ypos.mult`
- **wb.color**: color of the w.band
- **wb.name**: label of w.band

Default aesthetics

Set by the statistic and available to geoms.

- **label**: ..y.label..
- **x**: ..x..
- **xmin**: ..xmin..
- **xmax**: ..xmax..
- **ymin**: y.. - (..ymax.. - ..ymin..) * 0.03
- **ymax**: y.. + (..ymax.. - ..ymin..) * 0.03
- **yintercept**: ..ymean..
- **fill**: ..wb.color..

Required aesthetics

Required by the statistic and need to be set with `aes()`.

- **x**: numeric, wavelength in nanometres
- **y**: numeric, a spectral quantity
See Also

Other stats functions: `stat_color()`, `stat_find_qtys()`, `stat_find_wls()`, `stat_label_peaks()`, `stat_peaks()`, `stat_spikes()`, `stat_wb_box()`, `stat_wb_column()`, `stat_wb_hbar()`, `stat_wb_irrad()`, `stat_wb_label()`, `stat_wb_mean()`, `stat_wb_relative()`, `stat_wb_sirrad()`, `stat_wb_total()`, `stat_wl_strip()`, `stat_wl_summary()`

Examples

```r
library(photobiologyWavebands)
# ggplot() methods for spectral objects set a default mapping for x and y.
ggplot(sun.sptc) +
  geom_line() +
  stat_wb_box(w.band = VIS()) +
  stat_wb_contribution(w.band = VIS()) +
  scale_fill_identity() + scale_color_identity()

  ggplot(sun.sptc) +
  geom_line() +
  stat_wb_box(w.band = VIS_bands()) +
  stat_wb_contribution(w.band = VIS_bands(), angle = 90, size = 2.5) +
  scale_fill_identity() + scale_color_identity()

  ggplot(sun.sptc) +
  geom_line() +
  stat_wb_box(w.band = VIS_bands()) +
  stat_wb_contribution(w.band = VIS_bands(), angle = 90, size = 2.5,
    label.mult = 100, label.fmt = "%3.0f%%") +
  scale_fill_identity() + scale_color_identity()
```

```r
stat_wb_hbar
Integrate ranges under curve.
```

Description

`stat_wb_hbar` computes means under a curve. It first integrates the area under a spectral curve and also the mean expressed per nanometre of wavelength for each waveband in the input. Sets suitable default aesthetics for geoms "errorbarh" and "hline" from 'ggplot', and "linerangeh", and "errorbarh" from 'ggstance'.

Usage

```r
stat_wb_hbar(
  mapping = NULL,
  data = NULL,
  geom = "errorbarh",
  w.band = NULL,
```
integral.fun = integrate_xy,
chroma.type = "CMF",
ypos.fixed = NULL,
position = "identity",
na.rm = FALSE,
show.legend = NA,
inherit.aes = TRUE,
...
)

Arguments

mapping The aesthetic mapping, usually constructed with \code{aes} or \code{aes_.} Only needs to be set at the layer level if you are overriding the plot defaults.
data A layer specific dataset - only needed if you want to override the plot defaults.
geom The geometric object to use display the data
w.band a waveband object or a list of waveband objects or numeric vector of at least length two.
integral.fun function on $x$ and $y$.
chroma.type character one of "CMF" (color matching function) or "CC" (color coordinates) or a \code{chroma_spct} object.
ypos.fixed numeric If not NULL used a constant value returned in y.
position The position adjustment to use for overlapping points on this layer
na.rm a logical value indicating whether NA values should be stripped before the computation proceeds.
show.legend logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. \code{borders}.
...
other arguments passed on to \code{layer}. This can include aesthetics whose values you want to set, not map. See \code{layer} for more details.

Value

A data frame with one row for each waveband object in the argument to \code{w.band}. Waveband outside the range of the spectral data are trimmed or discarded.

Computed variables

What it is named integral below is the result of appying \code{integral.fun}, with default \code{integrate_xy}.

\code{x} w.band-midpoint
\code{xmin} w.band minimum
\code{xmax} w.band maximum
\( \text{ymin} \) data$y minimum
\( \text{ymax} \) data$y maximum
\( \text{yint} \) data$y integral for the range of \( w.\text{band} \)
\( \text{ymean} \) yint divided by wl_expanse(\( w.\text{band} \))
\( y \) ypos.fixed or mean of data
\( \text{wb.color} \) color of the \( w.\text{band} \)
\( \text{wb.name} \) label of \( w.\text{band} \)

**Default aesthetics**

Set by the statistic and available to geoms.

- \( \text{xmin} \) \( .\text{xmin}.. \)
- \( \text{xmax} \) \( .\text{xmax}.. \)
- \( \text{yintercept} \) \( .\text{ymean}.. \)
- \( \text{height} \) \( (.\text{ymax}.. - .\text{ymin}..) \times 10^{-2} \)
- \( \text{color} \) \( .\text{wb.color}.. \)

**Required aesthetics**

Required by the statistic and need to be set with \text{aes()}.

- \( x \) numeric, wavelength in nanometres
- \( y \) numeric, a spectral quantity

**Note**

If the argument passed to \( w.\text{band} \) is a BSWF it is silently converted to a wavelength range and the average of spectral values without any weighting is returned as default value for \( y \).

**See Also**

Other stats functions: \text{stat_color()}, \text{stat_find_qtys()}, \text{stat_find_wls()}, \text{stat_label_peaks()}, \text{stat_peaks()}, \text{stat_spikes()}, \text{stat_wb_box()}, \text{stat_wb_column()}, \text{stat_wb_contribution()}, \text{stat_wb_irrad()}, \text{stat_wb_label()}, \text{stat_wb_mean()}, \text{stat_wb_relative()}, \text{stat_wb_sirrad()}, \text{stat_wb_total()}, \text{stat_wl_strip()}, \text{stat_wl_summary()}

**Examples**

```r
library(photobiologyWavebands)
# ggplot() methods for spectral objects set a default mapping for x and y.
ggplot(sun.spct) +
  geom_line() +
  stat_wb_hbar(w.band = VIS_bands(), size = 1) +
  scale_color_identity() +
  theme_bw()
```
**stat_wb_irrad**

Integrate irradiance for wavebands.

**Description**

stat_wb_irrad computes areas under a curve.

**Usage**

```r
stat_wb_irrad(
  mapping = NULL,
  data = NULL,
  geom = "text",
  w.band = NULL,
  time.unit,
  unit.in,
  label.qty = "total",
  label.mult = 1,
  chroma.type = "CMF",
  label.fmt = "%.3g",
  ypos.mult = 1.07,
  ypos.fixed = NULL,
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = NA,
  ...
)
```
Arguments

mapping The aesthetic mapping, usually constructed with `aes` or `aes_`. Only needs to be set at the layer level if you are overriding the plot defaults.
data A layer specific dataset - only needed if you want to override the plot defaults.
geom The geometric object to use display the data
w.band a waveband object or a list of waveband objects or numeric vector of at least length two.


time.unit character or lubridate::duration

unit.in character One of "photon","quantum" or "energy"

label.qty character

label.mult numeric Scaling factor applied to y-integral values before conversion into character strings.

chroma.type character one of "CMF" (color matching function) or "CC" (color coordinates) or a chroma_spct object.

label.fmt character string giving a format definition for converting y-integral values into character strings by means of function sprintf.

ypos.mult numeric Multiplier constant used to scale returned y values.

ypos.fixed numeric If not NULL used a constant value returned in y.

position The position adjustment to use for overlapping points on this layer

na.rm a logical value indicating whether NA values should be stripped before the computation proceeds.

show.legend logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.

inherit.aes If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. borders.

... other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.

Value

A data frame with one row for each waveband object in the argument to w.band. Wavebands outside the range of the spectral data are trimmed or discarded.

Computed variables

What it is named integral below is the result of applying irr ad, e_irrad or q_irrad to the data.

y.label yeff multiplied by label.mult and formatted according to label.fmt

x w.band-midpoint

wb.xmin w.band minimum

wb.xmax w.band maximum

wb.ymin data$y minimum

wb ymax data$y maximum

wb.yeff weighted irradiance if w.band describes a BSWF

wb.yint not weighted irradiance for the range of w.band

wb.xmean yint divided by wl_expanse(w.band)

y ypos.fixed or top of data, adjusted by ypos.mult

wb.color color of the w.band

wb.name label of w.band

BW.color black_or_white(wb.color)
Default aesthetics

Set by the statistic and available to geoms.

- **label** .y.label..
- **x** ..x..
- **xmin** ..wb.xmin..
- **xmax** ..wbxmax..
- **ymin** ..y.. - (..wb.ymax.. - ..wb.ymin..) * 0.03
- **ymax** ..y.. + (..wb.ymax.. - ..wb.ymin..) * 0.03
- **yintercept** ..wb.ymean..
- **fill** ..wb.color..

Required aesthetics

Required by the statistic and need to be set with `aes()`.

- **x** numeric, wavelength in nanometres
- **y** numeric, a spectral quantity

See Also

Other stats functions: `stat_color()`, `stat_find_qtys()`, `stat_find_wls()`, `stat_label_peaks()`, `stat_peaks()`, `stat_spikes()`, `stat_wb_box()`, `stat_wb_column()`, `stat_wb_contribution()`, `stat_wb_hbar()`, `stat_wb_label()`, `stat_wb_mean()`, `stat_wb_relative()`, `stat_wb_sirrad()`, `stat_wb_total()`, `stat_wl_strip()`, `stat_wl_summary()`

Examples

```r
library(photobiologyWavebands)
# ggplot() methods for spectral objects set a default mapping for x and y.
ggplot(sun.spct) +
  stat_wb_column(w.band = VIS_bands()) +
  stat_wb_e_irrad(w.band = VIS_bands(),
    angle = 90, size = 4,
    label.fmt = "%2.0f", ypos.fixed = 0.1) +
  geom_line() +
  scale_fill_identity() + scale_color_identity()

ggplot(sun.spct) +
  stat_wb_box(w.band = VIS_bands()) +
  stat_wb_e_irrad(w.band = VIS_bands(),
    angle = 90, size = 4,
    label.fmt = "%2.0f") +
  geom_line() +
  scale_fill_identity() + scale_color_identity()
```
stat_wb_label

Label ranges under spectral curve.

Description

stat_wb_label computes computes the center of a waveband. Sets suitable default aesthetics for "text" and "label" geoms displaying "boundaries" and "names" of wavebands.

Usage

stat_wb_label(
  mapping = NULL,
  data = NULL,
  geom = "text",
  w.band = NULL,
  chroma.type = "CMF",
  label.fmt = "%s",
  ypos.fixed = 0,
  position = "identity",
  na.rm = TRUE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)

Arguments

mapping  The aesthetic mapping, usually constructed with aes or aes_. Only needs to be set at the layer level if you are overriding the plot defaults.

data A layer specific dataset - only needed if you want to override the plot defaults.

geom The geometric object to use display the data

w.band a waveband object or a list of waveband objects or numeric vector of at least length two.

chroma.type character one of "CMF" (color matching function) or "CC" (color coordinates) or a chroma_spct object.

label.fmt character string giving a format definition for formating the name of the waveband. sprintf.

ypos.fixed numeric If not NULL used a constant value returned in y.

position The position adjustment to use for overlapping points on this layer

na.rm a logical value indicating whether NA values should be stripped before the computation proceeds.

show.legend logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. `borders`.

... other arguments passed on to `layer`. This can include aesthetics whose values you want to set, not map. See `layer` for more details.

Value

A data frame with one row for each waveband object in the argument to `w.band`. Wavebeand outside the range of the spectral data are trimmed or discarded.

Computed variables

- `x` w.band-midpoint
- `wb.xmin` w.band minimum
- `wb.xmax` w.band maximum
- `y` ypos.fixed or zero
- `wb.color` color of the w.band
- `wb.name` label of w.band
- `wb.label` formatted wb.name

Default aesthetics

Set by the statistic and available to geoms.

- `label` ..wb.label..
- `x` ..x..
- `xmin` ..wb.xmin..
- `xmax` ..wb.xmax..
- `fill` ..wb.color..

Required aesthetics

Required by the statistic and need to be set with `aes()`.

- `x` numeric, wavelength in nanometres

Note

This stat uses a panel function and ignores grouping as it is meant to be used for annotations.

See Also

Other stats functions: `stat_color()`, `stat_find_qtys()`, `stat_find_wls()`, `stat_label_peaks()`, `stat_peaks()`, `stat_spikes()`, `stat_wb_box()`, `stat_wb_column()`, `stat_wb_contribution()`, `stat_wb_hbar()`, `stat_wb_irrad()`, `stat_wb_mean()`, `stat_wb_relative()`, `stat_wb_sirrad()`, `stat_wb_total()`, `stat_wl_strip()`, `stat_wl_summary()`
Examples

library(photobiologyWavebands)
# ggplot() methods for spectral objects set a default mapping for x and y.
ggplot(sun.spct) +
  geom_line() +
  stat_wb_box(w.band = VIS(), ymin = -0.04, ymax = 0,
              color = "black", fill = "white") +
  stat_wb_label(w.band = VIS(), ypos.fixed = -0.02, color = "black")

  stat_wb_hbar(w.band = PAR(), ypos.fixed = 0, size = 1) +
  stat_wb_label(aes(color = ..wb.color..),
                w.band = PAR(), ypos.fixed = +0.025) +
  scale_color_identity()

stat_wb_mean

Integrate ranges under curve.

Description

stat_wb_mean computes means under a curve. It first integrates the area under a spectral curve
and also the mean expressed per nanometre of wavelength for each waveband in the input. Sets
suitable default aesthetics for "rect", "hline", "vline", "text" and "label" geoms.

Usage

stat_wb_mean(
  mapping = NULL,
  data = NULL,
  geom = "text",
  w.band = NULL,
  integral.fun = integrate_xy,
  label.mult = 1,
  chroma.type = "CMF",
  label.fmt = "%3g",
  ypos.mult = 1.07,
  xpos.fixed = NULL,
  ypos.fixed = NULL,
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)
Arguments

- **mapping**: The aesthetic mapping, usually constructed with `aes` or `aes_`. Only needs to be set at the layer level if you are overriding the plot defaults.
- **data**: A layer specific dataset - only needed if you want to override the plot defaults.
- **geom**: The geometric object to use display the data.
- **w.band**: a waveband object or a list of waveband objects or numeric vector of at least length two.
- **integral.fun**: function on $x$ and $y$.
- **label.mult**: numeric Scaling factor applied to y-integral values before conversion into character strings.
- **chroma.type**: character one of "CMF" (color matching function) or "CC" (color coordinates) or a `chroma_spct` object.
- **label.fmt**: character string giving a format definition for converting y-integral values into character strings by means of function `sprintf`.
- **ypos.mult**: numeric Multiplier constant used to scale returned y values.
- **xpos.fixed, ypos.fixed**: numeric If not NULL used as constant value returned in x or y.
- **position**: The position adjustment to use for overlapping points on this layer.
- **na.rm**: a logical value indicating whether NA values should be stripped before the computation proceeds.
- **show.legend**: logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
- **inherit.aes**: If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. `borders`.
- **...**: other arguments passed on to `layer`. This can include aesthetics whose values you want to set, not map. See `layer` for more details.

Value

A data frame with one row for each waveband object in the argument to w.band. Waveband outside the range of the spectral data are trimmed or discarded.

Computed variables

What it is named integral below is the result of applying `integral.fun`, with default `integrate_xy`.

- **y.label**: ymean multiplied by `label.mult` and formatted according to `label.fmt`
- **x**: w.band-midpoint
- **wb.xmin**: w.band minimum
- **wb.xmax**: w.band maximum
- **wb.ymin**: data$y minimum
- **wb.ymax**: data$y maximum
stat_wb_mean

wb.yint  data$y integral for the range of w.band
wb.xmean  yint divided by wl_expanse(w.band)
y  ypos.fixed or top of data, adjusted by ypos.mult
wb.color  color of the w.band
wb.name  label of w.band
BW.color  black_or_white(wb.color)

Default aesthetics
Set by the statistic and available to geoms.

label  .y.label..
x  ..x..
xmin  ..wb.xmin..
xmax  ..wb.xmax..
ymin  0
ymax  ..wb.ymean..
yintercept  ..wb.ymean..
fill  ..wb.color..

Required aesthetics
Required by the statistic and need to be set with aes().

x  numeric, wavelength in nanometres
y  numeric, a spectral quantity

See Also
Other stats functions: stat_color(), stat_find_qtys(), stat_find_wls(), stat_label_peaks(),
stat_peaks(), stat_spikes(), stat_wb_box(), stat_wb_column(), stat_wb_contribution(),
stat_wb_hbar(), stat_wb_irrad(), stat_wb_label(), stat_wb_relative(), stat_wb_sirrad(),
stat_wb_total(), stat_wl_strip(), stat_wl_summary()

Examples

library(photobiologyWavebands)
# ggplot() methods for spectral objects set a default mapping for x and y.
ggplot(sun.spct) +
  stat_wb_column(w.band = VIS_bands()) +
  stat_wb_mean(w.band = VIS_bands(), angle = 90, color = "black") +
  geom_line() +
  scale_fill_identity() + scale_color_identity()

## Not run:
# example takes long to run
library(ggrepel)
ggplot(sun.spct) +
  geom_line() +
  stat_wb_hbar(w.band = VIS_bands(), size = 1.5) +
  stat_wb_mean(w.band = VIS_bands(),
    geom = "label_repel", nudge_y = +.03,
    segment.colour = NA) +
  scale_fill_identity() + scale_color_identity()

## End(Not run)

stat_wb_relative

Integrate ranges under spectral curve.

Description

`stat_wb_relative` computes means under a curve. It first integrates the area under a spectral curve and also the mean expressed per nanometre of wavelength for each waveband in the input. Sets suitable default aesthetics for "rect", "hline", "vline", "text" and "label" geoms displaying values per waveband "relative" to the sum of the wavebands.

Usage

```r
stat_wb_relative(
  mapping = NULL,
  data = NULL,
  geom = "text",
  w.band = NULL,
  integral.fun = integrate_xy,
  label.mult = 1,
  chroma.type = "CMF",
  label.fmt = "%1.2f",
  ypos.mult = 1.07,
  ypos.fixed = NULL,
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...)
```

Arguments

- `mapping` The aesthetic mapping, usually constructed with `aes` or `aes_`. Only needs to be set at the layer level if you are overriding the plot defaults.
- `data` A layer specific dataset - only needed if you want to override the plot defaults.
geom

The geometric object to use display the data.

w.band

A waveband object or a list of waveband objects or numeric vector of at least length two.

integral.fun

Function on $x$s and $y$s.

label.mult

Numeric scaling factor applied to y-integral values before conversion into character strings.

chroma.type

Character one of "CMF" (color matching function) or "CC" (color coordinates) or a chroma_spct object.

label.fmt

Character string giving a format definition for converting y-integral values into character strings by means of function sprintf.

ypos.mult

Numeric multiplier constant used to scale returned y values.

ypos.fixed

Numeric If not NULL used a constant value returned in y.

position

The position adjustment to use for overlapping points on this layer.

na.rm

A logical value indicating whether NA values should be stripped before the computation proceeds.

show.legend

Logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.

inherit.aes

If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. borders.

... other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.

Value

A data frame with one row for each waveband object in the argument to w.band. Wavebands outside the range of the spectral data are trimmed or discarded.

Computed variables

What it is named integral below is the result of applying integral.fun to the data, with default integrate_xy.

y.label yint multiplied by label.mult and formatted according to label.fmt

x w.band-midpoint

wb.xmin w.band minimum

wb.xmax w.band maximum

wb.ymin data$y minimum

wb.ymax data$y maximum

wb.yint data$y integral for each member of w.band / sum of data$y integrals for all wavebands in w.band

wb.xmean yint divided by wl_expanse(w.band)

y ypos.fixed or top of data, adjusted by ypos.mult

wb.color color of the w.band

wb.name label of w.band

BW.color black_or_white(wb.color)
Default aesthetics

Set by the statistic and available to geoms.

- **label**: `.y.label..`
- **x**: `..x..`
- **xmin**: `..wb.xmin..`
- **xmax**: `..wb.xmax..`
- **ymin**: `.y. - (..wb.ymax.. - ..wb.ymin..) * 0.03`
- **ymax**: `.y. + (..wb.ymax.. - ..wb.ymin..) * 0.03`
- **yintercept**: `..wb.ymean..`
- **fill**: `..wb.color..`

Required aesthetics

Required by the statistic and need to be set with `aes()`.

- **x**: numeric, wavelength in nanometres
- **y**: numeric, a spectral quantity

See Also

Other stats functions: `stat_color()`, `stat_find_qtys()`, `stat_find_wls()`, `stat_label_peaks()`, `stat_peaks()`, `stat_spikes()`, `stat_wb_box()`, `stat_wb_column()`, `stat_wb_contribution()`, `stat_wb_hbar()`, `stat_wb_irrad()`, `stat_wb_label()`, `stat_wb_mean()`, `stat_wb_sirrad()`, `stat_wb_total()`, `stat wl_strip()`, `stat wl_summary()`

Examples

```r
library(photobiologyWavebands)
# ggplot() methods for spectral objects set a default mapping for x and y.
ggplot(sun.spct) +
  geom_line() +
  stat_wb_box(w.band = VIS()) +
  stat_wb_relative(w.band = VIS()) +
  scale_fill_identity() + scale_color_identity()

ggplot(sun.spct) +
  geom_line() +
  stat_wb_box(w.band = VIS_bands()) +
  stat_wb_relative(w.band = VIS_bands(), angle = 90, size = 2.5) +
  scale_fill_identity() + scale_color_identity()

ggplot(sun.spct) +
  geom_line() +
  stat_wb_box(w.band = VIS_bands()) +
  stat_wb_relative(w.band = VIS_bands(), angle = 90, size = 2.5,
                   label.mult = 100, label.fmt = "%3.0f\%") +
  scale_fill_identity() + scale_color_identity()
```


**stat_wb_sirrad**

Integrate spectral irradiance for wavebands.

**Description**

`stat_wb_sirrad` computes areas under a curve.

**Usage**

```r
stat_wb_sirrad(
  mapping = NULL,
  data = NULL,
  geom = "text",
  w.band = NULL,
  time.unit = NULL,
  unit.in = NULL,
  label.qty = "mean",
  label.mult = 1,
  chroma.type = "CMF",
  label.fmt = "%g",
  ypos.mult = 0.55,
  xpos.fixed = NULL,
  ypos.fixed = NULL,
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)
```

```r
statwb.e_sirrad(
  mapping = NULL,
  data = NULL,
  geom = "text",
  w.band = NULL,
  time.unit = "second",
  unit.in = "energy",
  label.qty = "mean",
  label.mult = 1,
  chroma.type = "CMF",
  label.fmt = "%g",
  ypos.mult = 0.55,
  xpos.fixed = NULL,
  ypos.fixed = NULL,
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  ...
)
```
stat_wb_sirrad

mapping = NULL,
data = NULL,
geom = "text",
w.band = NULL,
time.unit = "second",
unit.in = "photon",
label.qty = "mean",
label.mult = 1,
chroma.type = "CMF",
label.fmt = "%3g",
ypos.mult = 1.07,
xpos.fixed = NULL,
ypos.fixed = NULL,
position = "identity",
na.rm = FALSE,
show.legend = NA,
inherit.aes = TRUE,
...)

Arguments

mapping The aesthetic mapping, usually constructed with \texttt{aes} or \texttt{aes_}. Only needs to be set at the layer level if you are overriding the plot defaults.
data A layer specific dataset - only needed if you want to override the plot defaults.
geom The geometric object to use display the data
w.band a waveband object or a list of waveband objects or numeric vector of at least length two.
time.unit character or lubridate::duration
unit.in character One of "photon","quantum" or "energy"
label.qty character
label.mult numeric Scaling factor applied to y-integral values before conversion into character strings.
chroma.type character one of "CMF" (color matching function) or "CC" (color coordinates) or a \texttt{chroma_spct} object.
label.fmt character string giving a format definition for converting y-integral values into character strings by means of function \texttt{sprintf}.
ypos.mult numeric Multiplier constant used to scale returned y values.
xpos.fixed, ypos.fixed numeric If not \texttt{NULL} used a constant value returned in x or y.
position The position adjustment to use for overlapping points on this layer
na.rm a logical value indicating whether NA values should be stripped before the com-
putation proceeds.
show.legend logical. Should this layer be included in the legends? NA, the default, includes if
any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes If FALSE, overrides the default aesthetics, rather than combining with them.
This is most useful for helper functions that define both data and aesthetics and
shouldn’t inherit behaviour from the default plot specification, e.g. borders.
...
other arguments passed on to layer. This can include aesthetics whose values
you want to set, not map. See layer for more details.

Value
A data frame with one row for each waveband object in the argument to w.band. Wavebands outside
the range of the spectral data are trimmed or discarded.

Computed variables
What it is named integral below is the result of applying irrad, e_irrad or q_irrad to the data.

y.label yeff multiplied by label.mult and formatted according to label.fmt
x w.band-midpoint
wb.xmin w.band minimum
wb.xmax w.band maximum
wb.ymin data$y minimum
wb.ymax data$y maximum
wb.yeff weighted irradiance if w.band describes a BSWF
wb.yint not weighted irradiance for the range of w.band
wb.xmean yint divided by wl_expanse(w.band)
y ypos.fixed or top of data, adjusted by ypos.mult
wb.color color of the w.band
wb.name label of w.band
BW.color black_or_white(wb.color)

Default aesthetics
Set by the statistic and available to geoms.

label ..y.label..
x ..x..
xmin ..wb.xmin..
xmax ..wb.xmax..
ymin 0
ymax ..wb.ymean..
yintercept ..wb.ymean..
fill ..wb.color..
Required aesthetics

Required by the statistic and need to be set with `aes()`.

- **x** numeric, wavelength in nanometres
- **y** numeric, a spectral quantity

See Also

Other stats functions: `stat_color()`, `stat_find_qtys()`, `stat_find_wls()`, `stat_label_peaks()`, `stat_peaks()`, `stat_spikes()`, `stat_wb_box()`, `stat_wb_column()`, `stat_wb_contribution()`, `stat_wb_hbar()`, `stat_wb_irrad()`, `stat_wb_label()`, `stat_wb_mean()`, `stat_wb_relative()`, `stat_wb_total()`, `stat_wl_strip()`, `stat_wl_summary()`

Examples

```r
library(photobiologyWavebands)
# ggplot() methods for spectral objects set a default mapping for x and y.
ggplot(sun.spct) +
  stat_wb_column(w.band = VIS_bands()) +
  stat_wb_e_sirrad(w.band = VIS_bands(), angle = 90, size = 4,
    label.fmt = "%1.2f", ypos.fixed = 0.1) +
  geom_line() +
  scale_fill_identity() + scale_color_identity()

ggplot(sun.spct, unit.out = "photon") +
  geom_line() +
  stat_wb_hbar(w.band = PAR(), size = 1) +
  stat_wb_q_sirrad(aes(color = ..wb.color..),
    w.band = PAR(), label.fmt = "mean = %.3g",
    ypos.mult = 1, xpos.fixed = 390, hjust = 1) +
  scale_color_identity()
```

---

**stat_wb_total**

Integrate ranges under spectral curve.

Description

`stat_wb_total` computes integral under a curve. It first integrates the area under a spectral curve and also the mean expressed per nanometre of wavelength for each waveband in the input. Sets suitable default aesthetics for "rect", "hline", "vline", "text" and "label" geoms displaying "totals" per waveband.
Usage

stat_wb_total(
  mapping = NULL,
  data = NULL,
  geom = "text",
  w.band = NULL,
  integral.fun = integrate_xy,
  label.mult = 1,
  chroma.type = "CMF",
  label.fmt = "%3g",
  ypos.mult = 1.07,
  ypos.fixed = NULL,
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)

Arguments

mapping The aesthetic mapping, usually constructed with aes or aes_. Only needs to be set at the layer level if you are overriding the plot defaults.
data A layer specific dataset - only needed if you want to override the plot defaults.
geom The geometric object to use display the data
w.band a waveband object or a list of waveband objects or numeric vector of at least length two.
integral.fun function on $x$ and $y$.
label.mult numeric Scaling factor applied to y-integral values before conversion into character strings.
chroma.type character one of "CMF" (color matching function) or "CC" (color coordinates) or a chroma_spct object.
label.fmt character string giving a format definition for converting y-integral values into character strings by means of function sprintf.
ypos.mult numeric Multiplier constant used to scale returned y values.
ypos.fixed numeric If not NULL used a constant value returned in y.
position The position adjustment to use for overlapping points on this layer
na.rm a logical value indicating whether NA values should be stripped before the computation proceeds.
show.legend logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. borders.
... other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.
Value

A data frame with one row for each waveband object in the argument to `w.band`. Wavebands outside the range of the spectral data are trimmed or discarded.

Computed variables

What it is named integral below is the result of applying `integral.fun`, with default `integrate.xy`.

- **y.label**: `ymean` multiplied by `label.mult` and formatted according to `label.fmt`
- **x**: `w.band`-midpoint
- **wb.xmin**: `w.band` minimum
- **wb.xmax**: `w.band` maximum
- **wb.ymin**: data$y` minimum
- **wb.ymax**: data$y` maximum
- **wb.yint**: data$y` integral for the range of `w.band`
- **wb.xmean**: `yint` divided by `wl_expanse(w.band)`
- **y**: `ypos.fixed` or top of data, adjusted by `ypos.mult`
- **wb.color**: color of the `w.band`
- **wb.name**: label of `w.band`
- **BW.color**: `black_or_white(wb.color)`

Default aesthetics

Set by the statistic and available to geoms.

- **label**: `..y.label..`
- **x**: `..x..`
- **xmin**: `..wb.xmin..`
- **xmax**: `..wb.xmax..`
- **ymin**: `..y.. - (..wb.ymax.. - ..wb.ymin..) * 0.03`
- **ymax**: `..y.. + (..wb.ymax.. - ..wb.ymin..) * 0.03`
- **yintercept**: `..wb.ymean..`
- **fill**: `..wb.color..`

Required aesthetics

Required by the statistic and need to be set with `aes()`.

- **x**: numeric, wavelength in nanometres
- **y**: numeric, a spectral quantity
See Also

Other stats functions: `stat_color()`, `stat_find_qtys()`, `stat_find_wls()`, `stat_label_peaks()`, `stat_peaks()`, `stat_spikes()`, `stat_wb_box()`, `stat_wb_column()`, `stat_wb_contribution()`, `stat_wb_hbar()`, `stat_wb_irrad()`, `stat_wb_label()`, `stat_wb_mean()`, `stat_wb_relative()`, `stat_wb_sirrad()`, `stat_wl_strip()`, `stat_wl_summary()`

Examples

```r
library(photobiologyWavebands)
# ggplot() methods for spectral objects set a default mapping for x and y.
ggplot(sun.spct) +
  geom_line() +
  stat_wb_box(w.band = VIS()) +
  stat_wb_total(w.band = VIS()) +
  scale_fill_identity() + scale_color_identity()

ggplot(sun.spct) +
  geom_line() +
  stat_wb_box(w.band = UV_bands(), color = "white") +
  stat_wb_total(w.band = UV_bands()) +
  scale_fill_identity() + scale_color_identity()
```

---

**stat_wl_strip**  
Calculate colours from wavelength.

**Description**

*stat_wl_strip* computes color definitions according to human vision.

**Usage**

```r
stat_wl_strip(
  mapping = NULL,
  data = NULL,
  geom = "rect",
  w.band = NULL,
  length.out = 150,
  chroma.type = "CMF",
  position = "identity",
  na.rm = TRUE,
  show.legend = FALSE,
  inherit.aes = TRUE,
  ...)
```

wl_guide(
mapping = NULL,
data = NULL,
chroma.type = "CMF",
w.band = NULL,
length.out = 150,
ymin = -Inf,
ymax = Inf,
position = "identity",
na.rm = FALSE,
show.legend = FALSE,
inherit.aes = TRUE,

)

Arguments

mapping The aesthetic mapping, usually constructed with \texttt{aes} or \texttt{aes_}. Only needs to be set at the layer level if you are overriding the plot defaults.
data A layer specific dataset - only needed if you want to override the plot defaults.
geom The geometric object to use display the data
w.band waveband object or a list of such objects or NULL.
length.out The number of steps to use to simulate a continuous range of colours when w.band == NULL.
chroma.type character one of "CMF" (color matching function) or "CC" (color coordinates) or a \texttt{chroma_spct} object.
position The position adjustment to use for overlapping points on this layer
na.rm a logical value indicating whether NA values should be stripped before the computation proceeds.
show.legend logical. Should this layer be included in the legends? \texttt{NA}, the default, includes if any aesthetics are mapped. \texttt{FALSE} never includes, and \texttt{TRUE} always includes.
inherit.aes If \texttt{FALSE}, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. \texttt{borders}.

... other arguments passed on to \texttt{layer}. This can include aesthetics whose values you want to set, not map. See \texttt{layer} for more details.

ymin, ymax numeric used as aesthetics for plotting the guide.

Value

\texttt{generic_spect} object with new \texttt{x} values plus other computed variables described below.

Computed variables

\texttt{x} (w.low + w1.high) / 2
\texttt{wl.low} boundary of waveband
**stat_wl_strip**

*wl.high*  boundary of waveband

*wl.color*  color corresponding to wavelength

*wb.color*  color corresponding to waveband

*wb.name*  label of w.band

**Default aesthetics**

Set by the statistic and available to geoms.

- **x**  ..x..
- **label**  as.character(..wb.f..)
- **xmin**  ..wl.low..
- **xmax**  ..wl.high..
- **fill**  ..wb.color..

**Required aesthetics**

Required by the statistic and need to be set with `aes()`.

- **x**  numeric, wavelength in nanometres

**Note**

This stat uses a panel function and ignores grouping as it is meant to be used for annotations.

**See Also**

- `color_of`, which is used internally.

Other stats functions: `stat_color()`, `stat_find_qtys()`, `stat_find_wls()`, `stat_label_peaks()`, `stat_peaks()`, `stat_spikes()`, `stat_wb_box()`, `stat_wb_column()`, `stat_wb_contribution()`, `stat_wb_hbar()`, `stat_wb_irrad()`, `stat_wb_label()`, `stat_wb_mean()`, `stat_wb_relative()`, `stat_wb_sirrad()`, `stat_wb_total()`, `stat_wl_summary()`

**Examples**

```r
# ggplot() methods for spectral objects set a default mapping for x and y.
ggplot(sun.spct) + geom_line() +
  stat_wl_strip(ymax = -0.02, ymin = -0.04) +
  scale_fill_identity()

# on some graphic devices the output may show spurious vertical lines
ggplot(sun.spct) + wl_guide(alpha = 0.33, color = NA) + geom_line()
```
stat_wl_summary  

Average area under curve for regions.

Description

stat_wl_summary computes the area under a curve.

Usage

stat_wl_summary(
  mapping = NULL,
  data = NULL,
  geom = "text",
  range = NULL,
  integral.fun = integrate_xy,
  label.fmt = "%.3g",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)

Arguments

mapping  
The aesthetic mapping, usually constructed with \texttt{aes} or \texttt{aes_}. Only needs to be set at the layer level if you are overriding the plot defaults.

data  
A layer specific dataset - only needed if you want to override the plot defaults.

geom  
The geometric object to use display the data

range  
a numeric vector of at least length two.

integral.fun  
function on $x$ and $y$.

label.fmt  
character string giving a format definition for converting y-integral values into character strings by means of function \texttt{sprintf}.

position  
The position adjustment to use for overlapping points on this layer

na.rm  
a logical value indicating whether NA values should be stripped before the computation proceeds.

show.legend  
logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.

inherit.aes  
If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. \texttt{borders}.

...  
other arguments passed on to \texttt{layer}. This can include aesthetics whose values you want to set, not map. See \texttt{layer} for more details.
**Value**

A data frame with one row.

**Computed variables**

What it is named integral below is the result of applying integral.fun, with default integrate_xy.

- `y.label`: y formatted according to label.fmt
- `x`: range-midpoint
- `wb.xmin`: range minimum
- `wb.xmax`: range maximum
- `y`: data$y integral for the range by the expanse of the range

**Default aesthetics**

Set by the statistic and available to geoms.

- `label`: ..label..
- `x`: ..x..
- `xmin`: ..wb.xmin..
- `xmax`: ..wb.xmax..
- `y`: ..y..
- `ymin`: 0
- `ymax`: ..y..
- `yintercept`: ..y..

**Required aesthetics**

Required by the statistic and need to be set with `aes()`.

- `x`: numeric, wavelength in nanometres
- `y`: numeric, a spectral quantity

**See Also**

Other stats functions: `stat_color()`, `stat_find_qtys()`, `stat_find_wls()`, `stat_label_peaks()`, `stat_peaks()`, `stat_spikes()`, `stat_wb_box()`, `stat_wb_column()`, `stat_wb_contribution()`, `stat_wb_hbar()`, `stat_wb_irrad()`, `stat_wb_label()`, `stat_wb_mean()`, `stat_wb_relative()`, `stat_wb_sirrad()`, `stat_wb_total()`, `stat_wl_strip()`
Examples

```r
# ggplot() methods for spectral objects set a default mapping for x and y.
ggplot(sun.spct) + geom_line() +
    stat_wl_summary(geom = "hline")

ggplot(sun.spct) + geom_line() +
    stat_wl_summary(label.fmt = "mean = %.3f", color = "red", vjust = -0.3) +
    stat_wl_summary(geom = "hline", color = "red")
```

---

**Tfr_internal_label**  
*Transmittance axis labels*

**Description**

Generate cps axis labels in SI units, using SI scale factors. Output can be selected as character, expression (R default devices) or LaTeX (for tikz device).

**Usage**

```r
Tfr_internal_label(
    unit.exponent = 0,
    format = getOption("photobiology.math", default = "R.expression")
)
```

**Arguments**

- `unit.exponent` integer
- `format` character string, "R", "R.expression", "R.character", or "LaTeX".

**Value**

a character string or an R expression.

**Examples**

```r
Tfr_internal_label()
Tfr_internal_label(-3)
Tfr_internal_label(format = "R.expression")
Tfr_internal_label(format = "LaTeX")
```
Description

Generate wavelength, wavenumber and wave frequency axis labels in SI units, using SI scale factors. Output can be selected as character, expression (R default devices) or LaTeX (for tikz device).

Usage

```r
w_length_label(
  unit.exponent = -9,
  format = getOption("photobiology.math", default = "R.expression")
)

w_number_label(
  unit.exponent = 0,
  format = getOption("photobiology.math", default = "R.expression")
)

w_frequency_label(
  unit.exponent = 9,
  format = getOption("photobiology.math", default = "R.expression")
)
```

Arguments

- `unit.exponent` integer
- `format` character string, "R", "R.expression", "R.character", or "LaTeX".

Value

a character string or an R expression.
Examples

w_length_label()
w_length_label("R.expression")
w_length_label("LaTeX")
w_number_label()
w_number_label("R.expression")
w_frequency_label()
w_frequency_label("R.expression")

Description

Convert wavelength into wavenumber or into frequency.

Usage

w_number(w.length, unit.exponent = 0)

w_frequency(w.length, unit.exponent = 0)

Arguments

w.length numeric wavelength (nm)
unit.exponent integer

Examples

w_number(600)
w_frequency(600)
Index

*Topic hplot
  autoplot.calibration_spct, 5
  autoplot.filter_spct, 8
  autoplot.object_spct, 11
  autoplot.raw_spct, 13
  autoplot.reflector_spct, 15
  autoplot.response_spct, 17
  autoplot.source_spct, 19
  autoplot.waveband, 21

A_internal_label, 24
A_total_label (A_internal_label), 24
aes, 30, 53, 55, 58, 61, 64, 68, 71, 73, 76, 79, 82, 85, 88, 90, 94, 97, 100, 102
aes_, 30, 53, 55, 58, 61, 64, 68, 71, 73, 76, 79, 82, 85, 88, 90, 94, 97, 100, 102
autoplot.calibration_mspct
  (autoplot.calibration_spct), 5
autoplot.calibration_spct, 5, 8, 10, 13, 14, 17, 19, 21, 22, 37, 51
autoplot.cps_spct, 6, 7, 10, 13, 14, 17, 19, 21, 22, 37, 51
autoplot.filter_mspct
  (autoplot.filter_spct), 8
autoplot.filter_spct, 6, 8, 8, 13, 14, 17, 19, 21, 22, 37, 51
autoplot.object_mspct
  (autoplot.object_spct), 11
autoplot.object_spct, 6, 8, 10, 11, 14, 17, 19, 21, 22, 51
autoplot.raw_spct, 6, 8, 10, 13, 13, 17, 19, 21, 22, 37, 51
autoplot.reflector_mspct
  (autoplot.reflector_spct), 15
autoplot.reflector_spct, 6, 8, 10, 13, 14, 15, 19, 21, 22, 51
autoplot.response_mspct
  (autoplot.response_spct), 17
autoplot.response_spct, 6, 8, 10, 13, 14, 17, 17, 21, 22, 37, 51
autoplot.source_mspct
  (autoplot.source_spct), 19
autoplot.source_spct, 6, 8, 10, 13, 14, 17, 19, 19, 22, 37, 51
autoplot.waveband, 6, 8, 10, 13, 14, 17, 19, 21, 21, 37, 51
autotitle, 23

black_or_white, 25
borders, 30, 53, 55, 58, 61, 65, 68, 71, 74, 76, 79, 83, 86, 88, 91, 95, 97, 100, 102

chroma_spct, 10, 12, 16, 20, 53, 55, 58, 61, 65, 68, 71, 74, 76, 79, 83, 85, 88, 91, 94, 97, 100

color_chart, 26
color_of, 54, 101
counts_label, 27
cps_label, 27

exponent2factor (exponent2prefix), 28
exponent2prefix, 28
exponent2prefix_name (exponent2prefix), 28

find_peaks, 57, 59, 63, 66
find_spikes, 70
format, 51, 52

geom_path, 30
gem_point, 30
gem_ribbon, 30
gem_spect, 29
ggplot, 31, 33
ggplot, 56, 59, 62, 66, 69
ggspectra (ggspectra-package), 3

ggspectra-package, 3
ggtitle_spct (autotitle), 23

has_SI_prefix (exponent2prefix), 28
layer, 30, 53, 55, 58, 61, 65, 68, 71, 74, 76, 79, 83, 86, 88, 91, 95, 97, 100, 102

multiplot, 34

nearest_SI_exponent (exponent2prefix), 28

plot.generic_mspct (plot.generic_spct), 36
plot.generic_spct, 36
plot.waveband (plot.generic_spct), 36
prefix2exponent (exponent2prefix), 28
prefix_name2exponent (exponent2prefix), 28

Rfr_specular_label, 37
Rfr_total_label (Rfr_specular_label), 37

s.e.irrad_label, 38
s.e.response_label, 39
s.q.irrad_label (s.e.irrad_label), 38
s.q.response_label
  (s.e.response_label), 39
scale_continuous, 40–44, 46–48
scale_x wl_continuous, 39
scale_y A_internal_continuous, 41
scale_y A_total_continuous
  (scale_y A_internal_continuous), 41
scale_y counts_continuous, 42
scale_y counts tg_continuous
  (scale_y counts_continuous), 42
scale_y cps_continuous, 43
scale_y Rfr_specular_continuous, 44
scale_y Rfr_total_continuous
  (scale_y Rfr_specular_continuous), 44
scale_y s.e.irrad_continuous, 45
scale_y s.e.irrad_log10
  (scale_y s.e.irrad_continuous), 45
scale_y s.e.response_continuous, 46
scale_y s.q.irrad_continuous
  (scale_y s.e.irrad_continuous), 45
scale_y s.q.irrad_log10
  (scale_y s.e.irrad_continuous), 45

scale_y s.q.response_continuous
  (scale_y s.e.response_continuous), 46
scale_y Tfr_internal_continuous, 47
scale_y Tfr_total_continuous
  (scale_y Tfr_internal_continuous), 47
sec_axis w_frequency
  (sec_axis w_number), 49
sec_axis w_number, 49
set_annotations_default, 6, 8, 10, 13, 14, 17, 19, 21, 22, 50
set_w.band_default
  (set_annotations_default), 50
SI_pl_format, 51
SI_plain (SI_pl_format), 51
SI_tagged (SI_tg_format), 52
SI_tg_format, 52
sprintf, 53, 55, 58, 61, 65, 68, 76, 83, 85, 88, 91, 94, 97, 102
stat_color, 53, 57, 59, 63, 66, 70, 72, 75, 78, 80, 84, 86, 89, 92, 96, 99, 101, 103
stat_find qtys, 54, 59, 63, 66, 70, 72, 75, 78, 80, 84, 86, 89, 92, 96, 99, 101, 103
stat_find_wls, 54, 57, 59, 63, 66, 70, 72, 75, 78, 80, 84, 86, 89, 92, 96, 99, 101, 103
stat_label_peaks, 54, 57, 59, 60, 66, 70, 72, 75, 78, 80, 84, 86, 89, 92, 96, 99, 101, 103
stat_label_valleys (stat_label_peaks), 60
stat_peaks, 54, 57, 59, 63, 67, 70, 72, 75, 78, 80, 84, 86, 89, 92, 96, 99, 101, 103
stat_spikes, 54, 57, 59, 63, 67, 72, 75, 78, 80, 84, 86, 89, 92, 96, 99, 101, 103
stat_valleys, 63
stat_valleys (stat_peaks), 63
stat_wb_box, 54, 57, 59, 63, 66, 70, 72, 75, 78, 80, 84, 86, 89, 92, 96, 99, 101, 103
stat_wb_column, 54, 57, 59, 63, 66, 70, 72, 73, 78, 80, 84, 86, 89, 92, 96, 99, 101, 103
stat_wb_contribution, 54, 57, 59, 63, 66, 70, 72, 75, 78, 80, 84, 86, 89, 92, 96,
99, 101, 103
stat_wb_e_irrad(stat_wb Irrad), 81
stat_wb_e_sirrad(stat_wb sirrad), 93
stat_wb_hbar, 54, 57, 59, 63, 66, 70, 72, 75, 78, 84, 86, 89, 92, 96, 99, 101, 103
stat_wb_irrad, 54, 57, 59, 63, 66, 70, 72, 75, 78, 80, 81, 86, 89, 92, 96, 99, 101, 103
stat_wb_label, 54, 57, 59, 63, 66, 70, 72, 75, 78, 80, 84, 85, 89, 92, 96, 99, 101, 103
stat_wb_mean, 54, 57, 59, 63, 66, 70, 72, 75, 78, 80, 84, 86, 87, 92, 96, 99, 101, 103
stat_wb_q_irrad(stat_wb Irrad), 81
stat_wb_q_sirrad(stat_wb sirrad), 93
stat_wb_relative, 54, 57, 59, 63, 66, 70, 72, 75, 78, 80, 84, 86, 89, 92, 96, 99, 101, 103
stat_wb_sirrad, 54, 57, 59, 63, 66, 70, 72, 75, 78, 80, 84, 86, 89, 92, 93, 99, 101, 103
stat_wb_total, 54, 57, 59, 63, 66, 70, 72, 75, 78, 80, 84, 86, 89, 92, 96, 96, 101, 103
stat_wl_strip, 54, 57, 59, 63, 66, 70, 72, 75, 78, 80, 84, 86, 89, 92, 96, 99, 103
stat_wl_summary, 54, 57, 59, 63, 66, 70, 72, 75, 78, 80, 84, 86, 89, 92, 96, 99, 101, 102
strptime, 6, 8, 10, 12, 14, 16, 18, 20, 23
Tfr_internal_label, 104
Tfr_total_label (Tfr_internal_label), 104
w_frequency (w_number), 106
w_frequency_label (w_length_label), 105
w_length_label, 105
w_number, 106
w_number_label (w_length_label), 105
wl_guide (stat_wl_strip), 99