Package ‘ggspectra’

July 22, 2025

Type Package

Title Extensions to 'ggplot2' for Radiation Spectra

Version 0.3.16

Date 2025-06-23

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 'r4photobiology' suite, Aphalo P. J. (2015) <doi:10.19232/uv4pb.2015.1.14>.

License GPL (>= 2)

LazyLoad TRUE

ByteCompile TRUE

Depends R (>= 4.1.0), photobiology (>= 0.13.1), ggplot2 (>= 3.5.0)

Imports stats, grid, photobiologyWavebands (>= 0.5.2), scales (>= 1.2.0), ggrepel (>= 0.9.2), lubridate (>= 1.9.0), rlang (>= 1.0.2), tibble (>= 3.1.5)

Suggests knitr (>= 1.38), rmarkdown (>= 2.13), magrittr (>= 2.0.3)

URL https://docs.r4photobiology.info/ggspectra/, https://github.com/aphalo/ggspectra/

BugReports https://github.com/aphalo/ggspectra/issues/

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation no

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

Repository CRAN

Date/Publication 2025-06-24 05:20:02 UTC

Contents

`ggspectra-package` *ggspectra: Extensions to 'ggplot2' for Radiation Spectra*

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 'r4photobiology' suite, Aphalo P. J. (2015) doi:10.19232/uv4pb.2015.1.14.

Details

Package ‘ggspectra‘ provides a set of layer functions and `autoplot()` methods extending packages ‘ggplot2‘ and ‘photobiology‘. The `autoplot()` methods specialised for objects of classes defined in package 'photobiology' facilitate in many respects the plotting of spectral data. The `ggplot()` methods specialised for objects of classes defined in package 'photobiology' combined with the new layer functions and scales easy the task of flexibly plotting radiation-related spectra and of annotating the resulting plots.

These methods, layer functions and scales are specialized and work only with certain types of data and ways of expressing physical quantities. Most importantly, all statistics expect the values mapped to the `x` aesthetic to be wavelengths expressed in nanometres (nm), which is ensured when the data are stored in data objects of classes defined in package 'photobiology'. The support for scale transforms is manual and only partial. Flipping is not supported.

Although originally aimed at plots relevant to photobiology, many of the functions in the package are also useful for plotting other 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), minima (valleys) and spikes present in spectral data can also be annotated with statistics from 'ggspectra'.

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 many other extensions to package 'ggplot2' such as packages 'ggrepel', 'gganimate' 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)

Other contributors:

- Titta K. Kotilainen (ORCID) [contributor]

References

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

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

See Also

Useful links:

- `https://docs.r4photobiology.info/ggspectra/`
- `https://github.com/aphalo/ggspectra/`
- Report bugs at `https://github.com/aphalo/ggspectra/issues/`

Examples

```
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()

autoplot(sun.spct)

autoplot(polyester.spct,
         UV_bands(),
         range = UV(),
         annotations = c("=", "segments", "labels"))
```

`Afr_label` *Absorptance 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

```
Afr_label(
  unit.exponent = ifelse(pc.out, -2, 0),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["s.Afr"]],
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  pc.out = getOption("ggspectra.pc.out", default = FALSE)
)

Rfr_total_label(
  unit.exponent = ifelse(pc.out, -2, 0),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = NULL,
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  pc.out = getOption("ggspectra.pc.out", default = FALSE)
)
```

Arguments

`unit.exponent`	integer
`format`	character string, "R", "R.expresion", "R.character", or "LaTeX".
`label.text`	character Textual portion of the labels.
`scaled`	logical If `TRUE` relative units are assumed.
`normalized`	logical (`FALSE`) or numeric Normalization wavelength in manometers (nm).
`axis.symbols`	logical If `TRUE` symbols of the quantities are added to the `name`. Supported only by `format = "R.expression"`.
`pc.out`	logical, if TRUE use percent as default instead of fraction of one.

Value

a character string or an R expression.

Examples

```
Afr_label()
Afr_label(format = "R.expression", axis.symbols = FALSE)
Afr_label(-2)
Afr_label(-3)
Afr_label(format = "R.expression")
Afr_label(format = "LaTeX")
Afr_label(-2, format = "LaTeX")


Rfr_total_label()
Rfr_total_label(axis.symbols = FALSE)
Rfr_total_label(-2)
Rfr_total_label(-3)
Rfr_total_label(format = "R.expression")
Rfr_total_label(format = "LaTeX")
Rfr_total_label(-3, format = "LaTeX")
```

`autoplot.calibration_spct` *Plot one or more irradiance-calibration spectra.*

Description

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

Usage

```
## S3 method for class 'calibration_spct'
autoplot(
  object,
  ...,
  w.band = getOption("photobiology.plot.bands", default = list(UVC(), UVB(), UVA(),
    PhR())),
  range = getOption("ggspectra.wlrange", default = NULL),
  unit.out = "ignored",
  pc.out = getOption("ggspectra.pc.out", default = FALSE),
  label.qty = "mean",
  span = NULL,
  wls.target = "HM",
  annotations = NULL,
  by.group = FALSE,
  geom = "line",
  time.format = "",
  tz = "UTC",
  norm = NA,
  text.size = 2.5,
  idfactor = NULL,
  facets = FALSE,
  plot.data = "as.is",
  ylim = c(NA, NA),
  object.label = deparse(substitute(object)),
  na.rm = TRUE
)

## S3 method for class 'calibration_mspct'
autoplot(
  object,
  ...,
  range = getOption("ggspectra.wlrange", default = NULL),
  unit.out = "ignored",
  norm = NA,
  pc.out = getOption("ggspectra.pc.out", default = FALSE),
  by.group = FALSE,
  plot.data = "as.is",
  idfactor = TRUE,
  facets = FALSE,
  object.label = deparse(substitute(object)),
  na.rm = TRUE
)
```

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 minimum and maximum wavelengths (nm).
`unit.out`	character IGNORED.
`pc.out`	logical, if `TRUE` use percent instead of fraction of one for normalized spectral data.
`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 centred 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. For details please see sections Plot **Annotations** and **Title Annotations**.
`by.group`	logical flag If TRUE repeated identical annotation layers are added for each group within a plot panel as needed for animation. If `FALSE`, the default, single layers are added per panel.
`geom`	character The name of a ggplot geometry, currently only `"area"`, `"spct"` and `"line"`. The default `NULL` selects between them based on `stacked`.
`time.format`	character Format as accepted by `strptime`.
`tz`	character Time zone to use for title and/or subtitle.
`norm`	numeric or character. Normalization to apply before plotting, If `object` is already normalized, the normalization is updated when a unit conversion applied.
`text.size`	numeric size of text in the plot decorations.
`idfactor`	character Name of an index `factor` used to identify each spectrum when multiple spectra are included in a plot. It is used as title to the guide in the plot and can include embedded spaces and new lines.
`facets`	logical or integer Indicating if facets are to be created for the levels of `idfactor` when `spct` contain multiple spectra in long form.
`plot.data`	character Data to plot. Default is `"as.is"` plotting one line per spectrum. When passing `"mean"`, `"median"`, `"sum"`, `"prod"`, `"var"`, `"sd"`, `"se"` as argument all the spectra must contain data at the same wavelength values.
`ylim`	numeric y axis limits,
`object.label`	character The name of the object being plotted.
`na.rm`	logical.

Details

The `autoplot()` methods from 'ggspectra' are convenience wrapper functions that easy the creation of plots from spectral objects at the cost of lacking the flexibility of the grammar of graphics. The plot object returned is a ggplot (an object of class `"gg"`) and it can be added to or modified as any other ggplot. The axis labels are encoded as *plotmath* expressions as they contain superscripts and special characters. In 'ggplot2', plotmath expressions do not obey theme settings related to text fonts, except for `size`.

Scale limits are expanded so as to make space for the annotations. If annotations are disabled, limits are not expanded unless `reserve.space` is passed to parameter `annotations`.

The generic of the `autoplot` method is defined in package 'ggplot2'. Package 'ggspectra' defines specializations for the different classes for storage of spectral data defined in package `photobiology`.

For details about normalization and arguments to parameter `norm`, please, see `normalize`. If `norm = NA`, the default, `normalize()` is not called. All other values passed as argument to `norm` result in a call to `normalize()` with this value as its argument. In the case of objects created with 'photobiology' (<= 0.10.9) `norm = "undo"` is not supported. Be aware that calls to `normalize()` remove any scaling previously applied with `fscale` methods.

For multiple spectra in long form spectral objects, with `idfactor = NULL`, the default, the name of the factor is retrieved from metadata. If the character string passed as argument to `idfactor` does not match the one retrieved from the object, results in renaming of the pre-existing factor. The default for collections of spectra is to create a factor named `"spct.idx"`, but if a different name is passed, it will be used instead.

Value

A `ggplot` object with a number of layers that depends on the data and annotations. The `data` member retains its original class and metadata attributes.

Plot Annotations

The recognized annotation names are: `"summaries"`, `"peaks"`, `"peak.labels"`, `"valleys"`, `"valley.labels"`, `"wls"`, `"wls.labels"`, `"colour.guide"`, `"color.guide"`, `"boxes"`, `"segments"`, `"labels"`. In addition, `"+"` is interpreted as a request to add to the already present default annotations, `"-"` as request to remove annotations and `"="` or missing`"+"` and `"-"` as a request to reset annotations to those requested. If used, `"+"`, `"-"` or `"="` must be the first member of a character vector, and followed by one or more of the names given above. To simultaneously add and remove annotations one can pass a `list` containing `character` vectors each assembled as described. The vectors are applied in the order they appear in the list. To disable all annotations pass `""` or `c("=", "")` as argument. Adding a variation of an annotation already present, replaces the existing one automatically: e.g., adding `"peak.labels"` replaces`"peaks"` if present.

The annotation layers are added to the plot using statistics defined in 'ggspectra': `stat_peaks`, `stat_valleys`, `stat_label_peaks`, `stat_label_valleys`, `stat_find_wls`, `stat_spikes`, `stat_wb_total`, `stat_wb_mean`, `stat_wb_irrad`, `stat_wb_sirrad`, `stat_wb_contribution`, `stat_wb_relative`, and `stat_wl_strip`. However, only some of their parameters can be passed arguments through `autoplot` methods. In some cases the defaults used by `autoplot` methods are not the defaults of the statistics.

Title Annotations

metadata retrieved from object `object` is paased 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 metatdata 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. Default is `"title:objt"` or no title depending on the context.

See Also

`normalize`, `calibration_spct`, `waveband`, `photobiologyWavebands-package` and `autoplot`

Other autoplot methods: `autoplot.cps_spct()`, `autoplot.filter_spct()`, `autoplot.object_spct()`, `autoplot.raw_spct()`, `autoplot.reflector_spct()`, `autoplot.response_spct()`, `autoplot.source_spct()`, `autoplot.waveband()`

Examples

```
# to be added
```

`autoplot.cps_spct` *Plot one or more detector-counts-per-second spectra.*

Description

These methods return a ggplot object with an annotated plot of a `cps_spct` or a `cps_mspct` object.

Usage

```
## S3 method for class 'cps_spct'
autoplot(
  object,
  ...,
  w.band = getOption("photobiology.plot.bands", default =
    list(photobiologyWavebands::UVC(), photobiologyWavebands::UVB(),
    photobiologyWavebands::UVA(), photobiologyWavebands::PhR())),
  range = getOption("ggspectra.wlrange", default = NULL),
  norm = NA,
  unit.out = NULL,
  pc.out = getOption("ggspectra.pc.out", default = FALSE),
  label.qty = "mean",
  span = NULL,
  wls.target = "HM",
  annotations = NULL,
  by.group = FALSE,
  geom = "line",
```

```
  time.format = "",
  tz = "UTC",
  text.size = 2.5,
  idfactor = NULL,
  facets = FALSE,
  plot.data = "as.is",
  ylim = c(NA, NA),
  object.label = deparse(substitute(object)),
  na.rm = TRUE
)

## S3 method for class 'cps_mspct'
autoplot(
  object,
  ...,
  range = getOption("ggspectra.wlrange", default = NULL),
  norm = NA,
  unit.out = NULL,
  pc.out = getOption("ggspectra.pc.out", default = FALSE),
  by.group = FALSE,
  idfactor = TRUE,
  facets = FALSE,
  plot.data = "as.is",
  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 minimum and maximum wavelengths (nm).
`norm`	numeric or character. Normalization to apply before plotting, If `object` is already normalized, the normalization is updated when a unit conversion applied.
`unit.out`	character IGNORED.
`pc.out`	logical, if `TRUE` use percent instead of fraction of one for normalized spectral data.
`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 centred 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. For details please see sections Plot **Annotations** and **Title Annotations**.
`by.group`	logical flag If TRUE repeated identical annotation layers are added for each group within a plot panel as needed for animation. If `FALSE`, the default, single layers are added per panel.
`geom`	character The name of a ggplot geometry, currently only `"area"`, `"spct"` and `"line"`. The default `NULL` selects between them based on `stacked`.
`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.
`idfactor`	character Name of an index `factor` used to identify each spectrum when multiple spectra are included in a plot. It is used as title to the guide in the plot and can include embedded spaces and new lines.
`facets`	logical or integer Indicating if facets are to be created for the levels of `idfactor` when `spct` contain multiple spectra in long form.
`plot.data`	character Data to plot. Default is `"as.is"` plotting one line per spectrum. When passing `"mean"`, `"median"`, `"sum"`, `"prod"`, `"var"`, `"sd"`, `"se"` as argument all the spectra must contain data at the same wavelength values.
`ylim`	numeric y axis limits,
`object.label`	character The name of the object being plotted.
`na.rm`	logical.

Details

The `autoplot()` methods from 'ggspectra' are convenience wrapper functions that easy the creation of plots from spectral objects at the cost of lacking the flexibility of the grammar of graphics. The plot object returned is a ggplot (an object of class `"gg"`) and it can be added to or modified as any other ggplot. The axis labels are encoded as *plotmath* expressions as they contain superscripts and special characters. In 'ggplot2', plotmath expressions do not obey theme settings related to text fonts, except for `size`.

Scale limits are expanded so as to make space for the annotations. If annotations are disabled, limits are not expanded unless `reserve.space` is passed to parameter `annotations`.

The generic of the `autoplot` method is defined in package 'ggplot2'. Package 'ggspectra' defines specializations for the different classes for storage of spectral data defined in package `photobiology`.

For details about normalization and arguments to parameter `norm`, please, see `normalize`. If `norm = NA`, the default, `normalize()` is not called. All other values passed as argument to `norm` result in a call to `normalize()` with this value as its argument. In the case of objects created with 'photobiology' (<= 0.10.9) `norm = "undo"` is not supported. Be aware that calls to `normalize()` remove any scaling previously applied with `fscale` methods.

For multiple spectra in long form spectral objects, with `idfactor = NULL`, the default, the name of the factor is retrieved from metadata. If the character string passed as argument to `idfactor` does not match the one retrieved from the object, results in renaming of the pre-existing factor. The default for collections of spectra is to create a factor named `"spct.idx"`, but if a different name is passed, it will be used instead.

Value

A `ggplot` object with a number of layers that depends on the data and annotations. The `data` member retains its original class and metadata attributes.

Plot Annotations

The recognized annotation names are: `"summaries"`, `"peaks"`, `"peak.labels"`, `"valleys"`, `"valley.labels"`, `"wls"`, `"wls.labels"`, `"colour.guide"`, `"color.guide"`, `"boxes"`, `"segments"`, `"labels"`. In addition, `"+"` is interpreted as a request to add to the already present default annotations, `"-"` as request to remove annotations and `"="` or missing`"+"` and `"-"` as a request to reset annotations to those requested. If used, `"+"`, `"-"` or `"="` must be the first member of a character vector, and followed by one or more of the names given above. To simultaneously add and remove annotations one can pass a `list` containing `character` vectors each assembled as described. The vectors are applied in the order they appear in the list. To disable all annotations pass `""` or `c("=", "")` as argument. Adding a variation of an annotation already present, replaces the existing one automatically: e.g., adding `"peak.labels"` replaces`"peaks"` if present.

The annotation layers are added to the plot using statistics defined in 'ggspectra': `stat_peaks`, `stat_valleys`, `stat_label_peaks`, `stat_label_valleys`, `stat_find_wls`, `stat_spikes`, `stat_wb_total`, `stat_wb_mean`, `stat_wb_irrad`, `stat_wb_sirrad`, `stat_wb_contribution`, `stat_wb_relative`, and `stat_wl_strip`. However, only some of their parameters can be passed arguments through `autoplot` methods. In some cases the defaults used by `autoplot` methods are not the defaults of the statistics.

Title Annotations

metadata retrieved from object `object` is paased 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 metatdata 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. Default is `"title:objt"` or no title depending on the context.

See Also

`normalize`, `cps_spct`, `waveband`, `photobiologyWavebands-package` and `autoplot`

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()`

Examples

```
autoplot(white_led.cps_spct)
autoplot(white_led.cps_spct, geom = "spct")
autoplot(normalize(white_led.cps_spct, norm = "max"))

two_leds.mspct <-
  cps_mspct(list("LED 1" = white_led.cps_spct,
                 "LED 2" = white_led.cps_spct / 2))
```

```
autoplot(two_leds.mspct)
autoplot(two_leds.mspct, idfactor = "Spectra")
autoplot(two_leds.mspct, plot.data = "mean")
```

`autoplot.filter_spct` *Plot one or more "filter" spectra.*

Description

These methods return a ggplot object of an annotated plot from spectral data contained in a `filter_spct` or a `filter_mspct` object. Data can be expressed as absorbance, absorptance or transmittance.

Usage

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

## S3 method for class 'filter_mspct'
autoplot(
  object,
  ...,
```

```
  range = getOption("ggspectra.wlrange", default = NULL),
  norm = NA,
  plot.qty = getOption("photobiology.filter.qty", default = "transmittance"),
  pc.out = getOption("ggspectra.pc.out", default = FALSE),
  by.group = FALSE,
  plot.data = "as.is",
  idfactor = TRUE,
  facets = FALSE,
  object.label = deparse(substitute(object)),
  na.rm = TRUE
)
```

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 minimum and maximum wavelengths (nm).
`norm`	numeric or character. Normalization to apply before plotting, If `object` is already normalized, the normalization is updated when a unit conversion applied.
`plot.qty`	character string one of "transmittance" or "absorbance".
`pc.out`	logical, if `TRUE` use percent instead of fraction of one for normalized spectral data.
`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 centred 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. For details please see sections Plot **Annotations** and **Title Annotations**.
`by.group`	logical flag If TRUE repeated identical annotation layers are added for each group within a plot panel as needed for animation. If `FALSE`, the default, single layers are added per panel.
`geom`	character The name of a ggplot geometry, currently only `"area"`, `"spct"` and `"line"`. The default `NULL` selects between them based on `stacked`.
`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 `factor` used to identify each spectrum when multiple spectra are included in a plot. It is used as title to the guide in the plot and can include embedded spaces and new lines.
`facets`	logical or integer Indicating if facets are to be created for the levels of `idfactor` when `spct` contain multiple spectra in long form.
`plot.data`	character Data to plot. Default is `"as.is"` plotting one line per spectrum. When passing `"mean"`, `"median"`, `"sum"`, `"prod"`, `"var"`, `"sd"`, `"se"` as argument all the spectra must contain data at the same wavelength values.
`ylim`	numeric y axis limits,
`object.label`	character The name of the object being plotted.
`na.rm`	logical.

Details

The `autoplot()` methods from 'ggspectra' are convenience wrapper functions that easy the creation of plots from spectral objects at the cost of lacking the flexibility of the grammar of graphics. The plot object returned is a ggplot (an object of class `"gg"`) and it can be added to or modified as any other ggplot. The axis labels are encoded as *plotmath* expressions as they contain superscripts and special characters. In 'ggplot2', plotmath expressions do not obey theme settings related to text fonts, except for `size`.

Scale limits are expanded so as to make space for the annotations. If annotations are disabled, limits are not expanded unless `reserve.space` is passed to parameter `annotations`.

The generic of the `autoplot` method is defined in package 'ggplot2'. Package 'ggspectra' defines specializations for the different classes for storage of spectral data defined in package `photobiology`.

For details about normalization and arguments to parameter `norm`, please, see `normalize`. If `norm = NA`, the default, `normalize()` is not called. All other values passed as argument to `norm` result in a call to `normalize()` with this value as its argument. In the case of objects created with 'photobiology' (<= 0.10.9) `norm = "undo"` is not supported. Be aware that calls to `normalize()` remove any scaling previously applied with `fscale` methods.

For multiple spectra in long form spectral objects, with `idfactor = NULL`, the default, the name of the factor is retrieved from metadata. If the character string passed as argument to `idfactor` does not match the one retrieved from the object, results in renaming of the pre-existing factor. The default for collections of spectra is to create a factor named `"spct.idx"`, but if a different name is passed, it will be used instead.

Value

A `ggplot` object with a number of layers that depends on the data and annotations. The `data` member retains its original class and metadata attributes.

Plot Annotations

The recognized annotation names are: `"summaries"`, `"peaks"`, `"peak.labels"`, `"valleys"`, `"valley.labels"`, `"wls"`, `"wls.labels"`, `"colour.guide"`, `"color.guide"`, `"boxes"`, `"segments"`, `"labels"`. In addition, `"+"` is interpreted as a request to add to the already present default annotations, `"-"` as request to remove annotations and `"="` or missing`"+"` and `"-"` as a request to reset annotations to those requested. If used, `"+"`, `"-"` or `"="` must be the first member of a character vector, and followed by one or more of the names given above. To simultaneously add and remove annotations one can pass a `list` containing `character` vectors each assembled as described. The vectors are applied in the order they appear in the list. To disable all annotations pass `""` or `c("=", "")` as argument. Adding a variation of an annotation already present, replaces the existing one automatically: e.g., adding `"peak.labels"` replaces`"peaks"` if present.

The annotation layers are added to the plot using statistics defined in 'ggspectra': `stat_peaks`, `stat_valleys`, `stat_label_peaks`, `stat_label_valleys`, `stat_find_wls`, `stat_spikes`, `stat_wb_total`, `stat_wb_mean`, `stat_wb_irrad`, `stat_wb_sirrad`, `stat_wb_contribution`, `stat_wb_relative`, and `stat_wl_strip`. However, only some of their parameters can be passed arguments through `autoplot` methods. In some cases the defaults used by `autoplot` methods are not the defaults of the statistics.

Title Annotations

metadata retrieved from object `object` is paased 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 metatdata 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. Default is `"title:objt"` or no title depending on the context.

Note

The plotting of absorbance is an exception to scale limits 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.

If `idfactor = NULL`, the default for single spectra, the name of the factor is retrieved from metadata or if no metadata found, the default `"spct.idx"` is tried. The default for multiple spectra is to create a factor named `"spct.idx"`, but if a different name is passed, it will be used instead, possibly renaminig a pre-existing one.

See Also

`normalize`, `filter_spct`, `waveband`, `photobiologyWavebands-package` and `autoplot`

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()`

Examples

```
# one spectrum
autoplot(yellow_gel.spct)
autoplot(yellow_gel.spct, geom = "spct")
```

```
autoplot(yellow_gel.spct, plot.qty = "transmittance")
autoplot(yellow_gel.spct, plot.qty = "absorbance")
autoplot(yellow_gel.spct, pc.out = TRUE)
autoplot(yellow_gel.spct, annotations = c("+", "wls"))

# spectra for two filters in long form
autoplot(two_filters.spct)
autoplot(two_filters.spct, idfactor = TRUE)
autoplot(two_filters.spct, idfactor = "Spectra")
autoplot(two_filters.spct, facets = TRUE)

# spectra for two filters as a collection
autoplot(two_filters.mspct)
autoplot(two_filters.mspct, idfactor = "Spectra")
autoplot(two_filters.mspct, facets = TRUE)
```

`autoplot.object_spct` *Plot one or more "object" spectra.*

Description

These methods return a ggplot object with an annotated plot of an `object_spct` or an `object_spct` object. This objects contain spectral transmittance, reflectance and possibly absorptance data. As these quantities add up to one, only two are needed.

Usage

```
## S3 method for class 'object_spct'
autoplot(
  object,
  ...,
  w.band = getOption("photobiology.plot.bands", default = list(UVC(), UVB(), UVA(),
    PhR())),
  range = getOption("ggspectra.wlrange", default = NULL),
  norm = NA,
  plot.qty = "all",
  pc.out = getOption("ggspectra.pc.out", default = FALSE),
  label.qty = NULL,
  span = NULL,
  wls.target = "HM",
  annotations = NULL,
  by.group = FALSE,
  geom = NULL,
  time.format = "",
  tz = "UTC",
  stacked = plot.qty == "all",
  text.size = 2.5,
```

```
  chroma.type = "CMF",
  idfactor = NULL,
  facets = NULL,
  plot.data = "as.is",
  ylim = c(NA, NA),
  object.label = deparse(substitute(object)),
  na.rm = TRUE
)

## S3 method for class 'object_mspct'
autoplot(
  object,
  ...,
  range = getOption("ggspectra.wlrange", default = NULL),
  norm = NA,
  plot.qty = getOption("photobiology.filter.qty", default = "all"),
  pc.out = getOption("ggspectra.pc.out", default = FALSE),
  by.group = FALSE,
  plot.data = "as.is",
  idfactor = TRUE,
  facets = plot.qty == "all",
  object.label = deparse(substitute(object)),
  na.rm = TRUE
)
```

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 minimum and maximum wavelengths (nm).
`norm`	numeric or character. Normalization to apply before plotting, If `object` is already normalized, the normalization is updated when a unit conversion applied.
`plot.qty`	character string, one of "all", "transmittance", "absorbance", "absorptance", or "reflectance".
`pc.out`	logical, if `TRUE` use percent instead of fraction of one for normalized spectral data.
`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 centred 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. For details please see sections Plot **Annotations** and **Title Annotations**.
`by.group`	logical flag If TRUE repeated identical annotation layers are added for each group within a plot panel as needed for animation. If `FALSE`, the default, single layers are added per panel.
`geom`	character The name of a ggplot geometry, currently only `"area"`, `"spct"` and `"line"`. The default `NULL` selects between them based on `stacked`.
`time.format`	character Format as accepted by `strptime`.
`tz`	character Time zone to use for title and/or subtitle.
`stacked`	logical Whether to use `position_stack()` or `position_identity()`.
`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 `factor` used to identify each spectrum when multiple spectra are included in a plot. It is used as title to the guide in the plot and can include embedded spaces and new lines.
`facets`	logical or integer Indicating if facets are to be created for the levels of `idfactor` when `spct` contain multiple spectra in long form.
`plot.data`	character Data to plot. Default is `"as.is"` plotting one line per spectrum. When passing `"mean"`, `"median"`, `"sum"`, `"prod"`, `"var"`, `"sd"`, `"se"` as argument all the spectra must contain data at the same wavelength values.
`ylim`	numeric y axis limits,
`object.label`	character The name of the object being plotted.
`na.rm`	logical.

Details

The `autoplot()` methods from 'ggspectra' are convenience wrapper functions that easy the creation of plots from spectral objects at the cost of lacking the flexibility of the grammar of graphics. The plot object returned is a ggplot (an object of class `"gg"`) and it can be added to or modified as any other ggplot. The axis labels are encoded as *plotmath* expressions as they contain superscripts and special characters. In 'ggplot2', plotmath expressions do not obey theme settings related to text fonts, except for `size`.

Scale limits are expanded so as to make space for the annotations. If annotations are disabled, limits are not expanded unless `reserve.space` is passed to parameter `annotations`.

The generic of the `autoplot` method is defined in package 'ggplot2'. Package 'ggspectra' defines specializations for the different classes for storage of spectral data defined in package `photobiology`.

For details about normalization and arguments to parameter `norm`, please, see `normalize`. If `norm = NA`, the default, `normalize()` is not called. All other values passed as argument to `norm` result in a call to `normalize()` with this value as its argument. In the case of objects created with 'photobiology' (<= 0.10.9) `norm = "undo"` is not supported. Be aware that calls to `normalize()` remove any scaling previously applied with `fscale` methods.

For multiple spectra in long form spectral objects, with `idfactor = NULL`, the default, the name of the factor is retrieved from metadata. If the character string passed as argument to `idfactor` does not match the one retrieved from the object, results in renaming of the pre-existing factor. The default for collections of spectra is to create a factor named `"spct.idx"`, but if a different name is passed, it will be used instead.

Value

A `ggplot` object with a number of layers that depends on the data and annotations. The `data` member retains its original class and metadata attributes.

Plot Annotations

The recognized annotation names are: `"summaries"`, `"peaks"`, `"peak.labels"`, `"valleys"`, `"valley.labels"`, `"wls"`, `"wls.labels"`, `"colour.guide"`, `"color.guide"`, `"boxes"`, `"segments"`, `"labels"`. In addition, `"+"` is interpreted as a request to add to the already present default annotations, `"-"` as request to remove annotations and `"="` or missing`"+"` and `"-"` as a request to reset annotations to those requested. If used, `"+"`, `"-"` or `"="` must be the first member of a character vector, and followed by one or more of the names given above. To simultaneously add and remove annotations one can pass a `list` containing `character` vectors each assembled as described. The vectors are applied in the order they appear in the list. To disable all annotations pass `""` or `c("=", "")` as argument. Adding a variation of an annotation already present, replaces the existing one automatically: e.g., adding `"peak.labels"` replaces`"peaks"` if present.

The annotation layers are added to the plot using statistics defined in 'ggspectra': `stat_peaks`, `stat_valleys`, `stat_label_peaks`, `stat_label_valleys`, `stat_find_wls`, `stat_spikes`, `stat_wb_total`, `stat_wb_mean`, `stat_wb_irrad`, `stat_wb_sirrad`, `stat_wb_contribution`, `stat_wb_relative`, and `stat_wl_strip`. However, only some of their parameters can be passed arguments through `autoplot` methods. In some cases the defaults used by `autoplot` methods are not the defaults of the statistics.

Title Annotations

metadata retrieved from object `object` is paased 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 metatdata 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. Default is `"title:objt"` or no title depending on the context.

Note

In the case of multiple spectra contained in the argument to `object` plotting is for `plot.qty = "all"` is always done using facets. Other plot quantities are handled by the methods for `filter_spct` and `reflector_spct` objects after on-the-fly conversion and the use of facets is possible but not the default.

If `idfactor = NULL`, the default for single spectra, the name of the factor is retrieved from metadata or if no metadata found, the default "spct.idx" is tried. The default for multiple spectra is to create a factor named "spct.idx", but if a different name is passed, it will be used instead, possibly renaminig a pre-existing one.

See Also

`normalize`, `object_spct`, `waveband`, `photobiologyWavebands-package` and `autoplot`

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()`

Examples

```
low_res.spct <- thin_wl(Ler_leaf.spct,
                        max.wl.step = 20,
                        max.slope.delta = 0.01,
                        col.names = "Tfr")
autoplot(low_res.spct)
autoplot(low_res.spct, geom = "line")

two_leaves.mspct <-
  object_mspct(list("Arabidopsis leaf 1" = low_res.spct,
                    "Arabidopsis leaf 2" = low_res.spct))
autoplot(two_leaves.mspct, idfactor = "Spectra")
```

`autoplot.raw_spct` *Plot one or more raw-detector-counts spectra.*

Description

These methods construct a ggplot object with an annotated plot of a `raw_spct` or a `raw_mspct` object.

Usage

```
## S3 method for class 'raw_spct'
autoplot(
  object,
  ...,
  w.band = getOption("photobiology.plot.bands", default =
    list(photobiologyWavebands::UVC(), photobiologyWavebands::UVB(),
    photobiologyWavebands::UVA(), photobiologyWavebands::PhR())),
  range = getOption("ggspectra.wlrange", default = NULL),
  norm = NA,
  unit.out = "counts",
  pc.out = getOption("ggspectra.pc.out", default = FALSE),
  by.group = FALSE,
  label.qty = "mean",
  span = NULL,
  wls.target = "HM",
  annotations = NULL,
```

```
  geom = "line",
  time.format = "",
  tz = "UTC",
  text.size = 2.5,
  idfactor = NULL,
  facets = FALSE,
  plot.data = "as.is",
  ylim = c(NA, NA),
  object.label = deparse(substitute(object)),
  na.rm = TRUE
)

## S3 method for class 'raw_mspct'
autoplot(
  object,
  ...,
  range = getOption("ggspectra.wlrange", default = NULL),
  norm = NA,
  unit.out = "counts",
  pc.out = getOption("ggspectra.pc.out", default = FALSE),
  by.group = FALSE,
  idfactor = TRUE,
  facets = FALSE,
  plot.data = "as.is",
  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 minimum and maximum wavelengths (nm).
`norm`	numeric or character. Normalization to apply before plotting, If `object` is already normalized, the normalization is updated when a unit conversion applied.
`unit.out`	character IGNORED.
`pc.out`	logical, if `TRUE` use percent instead of fraction of one for normalized spectral data.
`by.group`	logical flag If TRUE repeated identical annotation layers are added for each group within a plot panel as needed for animation. If `FALSE`, the default, single layers are added per panel.
`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 centred 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. For details please see sections Plot **Annotations** and **Title Annotations**.
`geom`	character The name of a ggplot geometry, currently only `"area"`, `"spct"` and `"line"`. The default `NULL` selects between them based on `stacked`.
`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.
`idfactor`	character Name of an index `factor` used to identify each spectrum when multiple spectra are included in a plot. It is used as title to the guide in the plot and can include embedded spaces and new lines.
`facets`	logical or integer Indicating if facets are to be created for the levels of `idfactor` when `spct` contain multiple spectra in long form.
`plot.data`	character Data to plot. Default is `"as.is"` plotting one line per spectrum. When passing `"mean"`, `"median"`, `"sum"`, `"prod"`, `"var"`, `"sd"`, `"se"` as argument all the spectra must contain data at the same wavelength values.
`ylim`	numeric y axis limits,
`object.label`	character The name of the object being plotted.
`na.rm`	logical.

Details

The `autoplot()` methods from 'ggspectra' are convenience wrapper functions that easy the creation of plots from spectral objects at the cost of lacking the flexibility of the grammar of graphics. The plot object returned is a ggplot (an object of class `"gg"`) and it can be added to or modified as any other ggplot. The axis labels are encoded as *plotmath* expressions as they contain superscripts and special characters. In 'ggplot2', plotmath expressions do not obey theme settings related to text fonts, except for `size`.

Scale limits are expanded so as to make space for the annotations. If annotations are disabled, limits are not expanded unless `reserve.space` is passed to parameter `annotations`.

The generic of the `autoplot` method is defined in package 'ggplot2'. Package 'ggspectra' defines specializations for the different classes for storage of spectral data defined in package `photobiology`.

For details about normalization and arguments to parameter `norm`, please, see `normalize`. If `norm = NA`, the default, `normalize()` is not called. All other values passed as argument to `norm` result in a call to `normalize()` with this value as its argument. In the case of objects created with 'photobiology' (<= 0.10.9) `norm = "undo"` is not supported. Be aware that calls to `normalize()` remove any scaling previously applied with `fscale` methods.

For multiple spectra in long form spectral objects, with `idfactor = NULL`, the default, the name of the factor is retrieved from metadata. If the character string passed as argument to `idfactor` does not match the one retrieved from the object, results in renaming of the pre-existing factor. The default for collections of spectra is to create a factor named `"spct.idx"`, but if a different name is passed, it will be used instead.

Value

A `ggplot` object with a number of layers that depends on the data and annotations. The `data` member retains its original class and metadata attributes.

Plot Annotations

The recognized annotation names are: `"summaries"`, `"peaks"`, `"peak.labels"`, `"valleys"`, `"valley.labels"`, `"wls"`, `"wls.labels"`, `"colour.guide"`, `"color.guide"`, `"boxes"`, `"segments"`, `"labels"`. In addition, `"+"` is interpreted as a request to add to the already present default annotations, `"-"` as request to remove annotations and `"="` or missing`"+"` and `"-"` as a request to reset annotations to those requested. If used, `"+"`, `"-"` or `"="` must be the first member of a character vector, and followed by one or more of the names given above. To simultaneously add and remove annotations one can pass a `list` containing `character` vectors each assembled as described. The vectors are applied in the order they appear in the list. To disable all annotations pass `""` or `c("=", "")` as argument. Adding a variation of an annotation already present, replaces the existing one automatically: e.g., adding `"peak.labels"` replaces`"peaks"` if present.

The annotation layers are added to the plot using statistics defined in 'ggspectra': `stat_peaks`, `stat_valleys`, `stat_label_peaks`, `stat_label_valleys`, `stat_find_wls`, `stat_spikes`, `stat_wb_total`, `stat_wb_mean`, `stat_wb_irrad`, `stat_wb_sirrad`, `stat_wb_contribution`, `stat_wb_relative`, and `stat_wl_strip`. However, only some of their parameters can be passed arguments through `autoplot` methods. In some cases the defaults used by `autoplot` methods are not the defaults of the statistics.

Title Annotations

metadata retrieved from object `object` is paased 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 metatdata 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. Default is `"title:objt"` or no title depending on the context.

See Also

`normalize`, `raw_spct`, `waveband`, `photobiologyWavebands-package` and `autoplot`

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()`

Examples

```
low_res.raw_spct <- thin_wl(white_led.raw_spct,
                            max.wl.step = 20,
```

```
                            max.slope.delta = 0.05,
                            col.names = "counts_3")
autoplot(low_res.raw_spct)
autoplot(low_res.raw_spct, annotations = "")

two_leds.mspct <-
  raw_mspct(list("LED 1" = low_res.raw_spct,
                 "LED 2" = low_res.raw_spct))
autoplot(two_leds.mspct)
autoplot(two_leds.mspct, facets = 1) # one column
```

---

autoplot.reflector_spct

Plot one or more reflector spectra.

Description

These methods return a ggplot object for an annotated plot from spectral data stored in a `reflector_spct` or a `reflector_mspct` object.

Usage

```
## S3 method for class 'reflector_spct'
autoplot(
  object,
  ...,
 w.band = getOption("photobiology.plot.bands", default = list(UVC(), UVB(), UVA(),
    PhR())),
  range = getOption("ggspectra.wlrange", default = NULL),
  norm = NA,
  plot.qty = getOption("photobiology.reflector.qty", default = "reflectance"),
  pc.out = getOption("ggspectra.pc.out", default = FALSE),
  label.qty = NULL,
  span = NULL,
  wls.target = "HM",
  annotations = NULL,
  by.group = FALSE,
  geom = "line",
  time.format = "",
  tz = "UTC",
  text.size = 2.5,
  chroma.type = "CMF",
  idfactor = NULL,
  facets = FALSE,
  plot.data = "as.is",
  ylim = c(NA, NA),
  object.label = deparse(substitute(object)),
```

```
  na.rm = TRUE
)

## S3 method for class 'reflector_mspct'
autoplot(
  object,
  ...,
  range = getOption("ggspectra.wlrange", default = NULL),
  norm = NA,
  plot.qty = getOption("photobiology.reflector.qty", default = "reflectance"),
  pc.out = getOption("ggspectra.pc.out", default = FALSE),
  by.group = FALSE,
  plot.data = "as.is",
  idfactor = TRUE,
  facets = FALSE,
  object.label = deparse(substitute(object)),
  na.rm = TRUE
)
```

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 minimum and maximum wavelengths (nm).
`norm`	numeric or character. Normalization to apply before plotting, If `object` is already normalized, the normalization is updated when a unit conversion applied.
`plot.qty`	character string (currently ignored).
`pc.out`	logical, if `TRUE` use percent instead of fraction of one for normalized spectral data.
`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 centred 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. For details please see sections Plot **Annotations** and **Title Annotations**.
`by.group`	logical flag If TRUE repeated identical annotation layers are added for each group within a plot panel as needed for animation. If `FALSE`, the default, single layers are added per panel.

`geom`	character The name of a ggplot geometry, currently only `"area"`, `"spct"` and `"line"`. The default `NULL` selects between them based on `stacked`.
`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 `factor` used to identify each spectrum when multiple spectra are included in a plot. It is used as title to the guide in the plot and can include embedded spaces and new lines.
`facets`	logical or integer Indicating if facets are to be created for the levels of `idfactor` when `spct` contain multiple spectra in long form.
`plot.data`	character Data to plot. Default is `"as.is"` plotting one line per spectrum. When passing `"mean"`, `"median"`, `"sum"`, `"prod"`, `"var"`, `"sd"`, `"se"` as argument all the spectra must contain data at the same wavelength values.
`ylim`	numeric y axis limits,
`object.label`	character The name of the object being plotted.
`na.rm`	logical.

Details

The `autoplot()` methods from 'ggspectra' are convenience wrapper functions that easy the creation of plots from spectral objects at the cost of lacking the flexibility of the grammar of graphics. The plot object returned is a ggplot (an object of class `"gg"`) and it can be added to or modified as any other ggplot. The axis labels are encoded as *plotmath* expressions as they contain superscripts and special characters. In 'ggplot2', plotmath expressions do not obey theme settings related to text fonts, except for `size`.

Scale limits are expanded so as to make space for the annotations. If annotations are disabled, limits are not expanded unless `reserve.space` is passed to parameter `annotations`.

The generic of the `autoplot` method is defined in package 'ggplot2'. Package 'ggspectra' defines specializations for the different classes for storage of spectral data defined in package `photobiology`.

For details about normalization and arguments to parameter `norm`, please, see `normalize`. If `norm = NA`, the default, `normalize()` is not called. All other values passed as argument to `norm` result in a call to `normalize()` with this value as its argument. In the case of objects created with 'photobiology' (<= 0.10.9) `norm = "undo"` is not supported. Be aware that calls to `normalize()` remove any scaling previously applied with `fscale` methods.

For multiple spectra in long form spectral objects, with `idfactor = NULL`, the default, the name of the factor is retrieved from metadata. If the character string passed as argument to `idfactor` does not match the one retrieved from the object, results in renaming of the pre-existing factor. The default for collections of spectra is to create a factor named `"spct.idx"`, but if a different name is passed, it will be used instead.

Value

A `ggplot` object with a number of layers that depends on the data and annotations. The `data` member retains its original class and metadata attributes.

Plot Annotations

The recognized annotation names are: `"summaries"`, `"peaks"`, `"peak.labels"`, `"valleys"`, `"valley.labels"`, `"wls"`, `"wls.labels"`, `"colour.guide"`, `"color.guide"`, `"boxes"`, `"segments"`, `"labels"`. In addition, `"+"` is interpreted as a request to add to the already present default annotations, `"-"` as request to remove annotations and `"="` or missing`"+"` and `"-"` as a request to reset annotations to those requested. If used, `"+"`, `"-"` or `"="` must be the first member of a character vector, and followed by one or more of the names given above. To simultaneously add and remove annotations one can pass a `list` containing `character` vectors each assembled as described. The vectors are applied in the order they appear in the list. To disable all annotations pass `""` or `c("=", "")` as argument. Adding a variation of an annotation already present, replaces the existing one automatically: e.g., adding `"peak.labels"` replaces`"peaks"` if present.

The annotation layers are added to the plot using statistics defined in 'ggspectra': `stat_peaks`, `stat_valleys`, `stat_label_peaks`, `stat_label_valleys`, `stat_find_wls`, `stat_spikes`, `stat_wb_total`, `stat_wb_mean`, `stat_wb_irrad`, `stat_wb_sirrad`, `stat_wb_contribution`, `stat_wb_relative`, and `stat_wl_strip`. However, only some of their parameters can be passed arguments through `autoplot` methods. In some cases the defaults used by `autoplot` methods are not the defaults of the statistics.

Title Annotations

metadata retrieved from object `object` is paased 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 metatdata 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. Default is `"title:objt"` or no title depending on the context.

See Also

`normalize`, `reflector_spct`, `waveband`, `photobiologyWavebands-package` and `autoplot`

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()`

Examples

```
autoplot(Ler_leaf_rflt.spct)
autoplot(Ler_leaf_rflt.spct, geom = "spct")
autoplot(Ler_leaf_rflt.spct, annotations = c("+", "valleys"))

two_leaves.mspct <-
 reflector_mspct(list("Arabidopsis leaf 1" = Ler_leaf_rflt.spct,
                      "Arabidopsis leaf 2" = Ler_leaf_rflt.spct / 2))
autoplot(two_leaves.mspct)
autoplot(two_leaves.mspct, idfactor = "Spectra")
autoplot(two_leaves.mspct, facets = 2)
```

---

autoplot.response_spct

Plot one or more response spectra.

Description

These methods return a ggplot object with an annotated plot of the spectral data contained in a `response_spct` or a `response_mspct` object. Spectral responsitivity can be expressed either on an energy basis or a photon or quantum basis.

Usage

```
## S3 method for class 'response_spct'
autoplot(
  object,
  ...,
  w.band = getOption("photobiology.plot.bands", default =
    list(photobiologyWavebands::UVC(), photobiologyWavebands::UVB(),
    photobiologyWavebands::UVA(), photobiologyWavebands::PhR())),
  range = getOption("ggspectra.wlrange", default = NULL),
  norm = NA,
  unit.out = getOption("photobiology.radiation.unit", default = "energy"),
  pc.out = getOption("ggspectra.pc.out", default = FALSE),
  label.qty = NULL,
  span = NULL,
  wls.target = "HM",
  annotations = NULL,
  by.group = FALSE,
  geom = "line",
  time.format = "",
  tz = "UTC",
  text.size = 2.5,
  idfactor = NULL,
  facets = FALSE,
  plot.data = "as.is",
  ylim = c(NA, NA),
  object.label = deparse(substitute(object)),
  na.rm = TRUE
)

## S3 method for class 'response_mspct'
autoplot(
  object,
  ...,
  range = getOption("ggspectra.wlrange", default = NULL),
  norm = NA,
  unit.out = getOption("photobiology.radiation.unit", default = "energy"),
```

```
  pc.out = getOption("ggspectra.pc.out", default = FALSE),
  by.group = FALSE,
  plot.data = "as.is",
  facets = FALSE,
  idfactor = TRUE,
  object.label = deparse(substitute(object)),
  na.rm = TRUE
)
```

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 minimum and maximum wavelengths (nm).
`norm`	numeric or character. Normalization to apply before plotting, If `object` is already normalized, the normalization is updated when a unit conversion applied.
`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 percent instead of fraction of one for normalized spectral data.
`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 centred 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. For details please see sections Plot **Annotations** and **Title Annotations**.
`by.group`	logical flag If TRUE repeated identical annotation layers are added for each group within a plot panel as needed for animation. If `FALSE`, the default, single layers are added per panel.
`geom`	character The name of a ggplot geometry, currently only `"area"`, `"spct"` and `"line"`. The default `NULL` selects between them based on `stacked`.
`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.
`idfactor`	character Name of an index `factor` used to identify each spectrum when multiple spectra are included in a plot. It is used as title to the guide in the plot and can include embedded spaces and new lines.

`facets`	logical or integer Indicating if facets are to be created for the levels of `idfactor` when `spct` contain multiple spectra in long form.
`plot.data`	character Data to plot. Default is `"as.is"` plotting one line per spectrum. When passing `"mean"`, `"median"`, `"sum"`, `"prod"`, `"var"`, `"sd"`, `"se"` as argument all the spectra must contain data at the same wavelength values.
`ylim`	numeric y axis limits,
`object.label`	character The name of the object being plotted.
`na.rm`	logical.

Details

The `autoplot()` methods from 'ggspectra' are convenience wrapper functions that easy the creation of plots from spectral objects at the cost of lacking the flexibility of the grammar of graphics. The plot object returned is a ggplot (an object of class `"gg"`) and it can be added to or modified as any other ggplot. The axis labels are encoded as *plotmath* expressions as they contain superscripts and special characters. In 'ggplot2', plotmath expressions do not obey theme settings related to text fonts, except for `size`.

Scale limits are expanded so as to make space for the annotations. If annotations are disabled, limits are not expanded unless `reserve.space` is passed to parameter `annotations`.

The generic of the `autoplot` method is defined in package 'ggplot2'. Package 'ggspectra' defines specializations for the different classes for storage of spectral data defined in package `photobiology`.

For details about normalization and arguments to parameter `norm`, please, see `normalize`. If `norm = NA`, the default, `normalize()` is not called. All other values passed as argument to `norm` result in a call to `normalize()` with this value as its argument. In the case of objects created with 'photobiology' (<= 0.10.9) `norm = "undo"` is not supported. Be aware that calls to `normalize()` remove any scaling previously applied with `fscale` methods.

For multiple spectra in long form spectral objects, with `idfactor = NULL`, the default, the name of the factor is retrieved from metadata. If the character string passed as argument to `idfactor` does not match the one retrieved from the object, results in renaming of the pre-existing factor. The default for collections of spectra is to create a factor named `"spct.idx"`, but if a different name is passed, it will be used instead.

Value

A `ggplot` object with a number of layers that depends on the data and annotations. The `data` member retains its original class and metadata attributes.

Plot Annotations

The recognized annotation names are: `"summaries"`, `"peaks"`, `"peak.labels"`, `"valleys"`, `"valley.labels"`, `"wls"`, `"wls.labels"`, `"colour.guide"`, `"color.guide"`, `"boxes"`, `"segments"`, `"labels"`. In addition, `"+"` is interpreted as a request to add to the already present default annotations, `"-"` as request to remove annotations and `"="` or missing`"+"` and `"-"` as a request to reset annotations to those requested. If used, `"+"`, `"-"` or `"="` must be the first member of a character vector, and followed by one or more of the names given above. To simultaneously add and remove annotations one can pass a `list` containing `character` vectors each assembled as described. The vectors are applied in the order they appear in the list. To disable all annotations pass `""` or `c("=", "")` as argument. Adding a variation of an annotation already present, replaces the existing one automatically: e.g., adding `"peak.labels"` replaces`"peaks"` if present.

The annotation layers are added to the plot using statistics defined in 'ggspectra': `stat_peaks`, `stat_valleys`, `stat_label_peaks`, `stat_label_valleys`, `stat_find_wls`, `stat_spikes`, `stat_wb_total`, `stat_wb_mean`, `stat_wb_irrad`, `stat_wb_sirrad`, `stat_wb_contribution`, `stat_wb_relative`, and `stat_wl_strip`. However, only some of their parameters can be passed arguments through `autoplot` methods. In some cases the defaults used by `autoplot` methods are not the defaults of the statistics.

Title Annotations

metadata retrieved from object `object` is paased 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 metatdata 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. Default is `"title:objt"` or no title depending on the context.

See Also

`normalize`, `response_spct`, `waveband`, `photobiologyWavebands-package` and `autoplot`

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()`

Examples

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

two_sensors.mspct <-
 response_mspct(list("Photodiode" = photodiode.spct,
                     "Coupled charge device" = ccd.spct)) |>
 normalize()

autoplot(two_sensors.mspct, unit.out = "photon")
autoplot(two_sensors.mspct, idfactor = "Spectra")
autoplot(two_sensors.mspct, facets = 2)
autoplot(two_sensors.mspct, geom = "spct")
```

`autoplot.source_spct` *Plot one or more light-source spectra.*

Description

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

Usage

```
## S3 method for class 'source_spct'
autoplot(
  object,
  ...,
  w.band = getOption("photobiology.plot.bands", default =
    list(photobiologyWavebands::UVC(), photobiologyWavebands::UVB(),
    photobiologyWavebands::UVA(), photobiologyWavebands::PhR())),
  range = getOption("ggspectra.wlrange", default = NULL),
  norm = NA,
  unit.out = getOption("photobiology.radiation.unit", default = "energy"),
  pc.out = getOption("ggspectra.pc.out", default = FALSE),
  label.qty = NULL,
  span = NULL,
  wls.target = "HM",
  annotations = NULL,
  by.group = FALSE,
  geom = "line",
  time.format = "",
  tz = "UTC",
  text.size = 2.5,
  chroma.type = "CMF",
  idfactor = NULL,
  facets = FALSE,
  plot.data = "as.is",
  ylim = c(NA, NA),
  object.label = deparse(substitute(object)),
  na.rm = TRUE
)

## S3 method for class 'source_mspct'
autoplot(
  object,
  ...,
  range = getOption("ggspectra.wlrange", default = NULL),
  norm = NA,
  unit.out = getOption("photobiology.radiation.unit", default = "energy"),
  pc.out = getOption("ggspectra.pc.out", default = FALSE),
```

```
  by.group = FALSE,
  idfactor = TRUE,
  facets = FALSE,
  plot.data = "as.is",
  object.label = deparse(substitute(object)),
  na.rm = TRUE
)
```

Arguments

`object`	a source_spct 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 minimum and maximum wavelengths (nm).
`norm`	numeric or character. Normalization to apply before plotting, If `object` is already normalized, the normalization is updated when a unit conversion applied.
`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 percent instead of fraction of one for normalized spectral data.
`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 centred 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. For details please see sections Plot **Annotations** and **Title Annotations**.
`by.group`	logical flag If TRUE repeated identical annotation layers are added for each group within a plot panel as needed for animation. If `FALSE`, the default, single layers are added per panel.
`geom`	character The name of a ggplot geometry, currently only `"area"`, `"spct"` and `"line"`. The default `NULL` selects between them based on `stacked`.
`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 `factor` used to identify each spectrum when multiple spectra are included in a plot. It is used as title to the guide in the plot and can include embedded spaces and new lines.
`facets`	logical or integer Indicating if facets are to be created for the levels of `idfactor` when `spct` contain multiple spectra in long form.
`plot.data`	character Data to plot. Default is `"as.is"` plotting one line per spectrum. When passing `"mean"`, `"median"`, `"sum"`, `"prod"`, `"var"`, `"sd"`, `"se"` as argument all the spectra must contain data at the same wavelength values.
`ylim`	numeric y axis limits,
`object.label`	character The name of the object being plotted.
`na.rm`	logical.

Details

The `autoplot()` methods from 'ggspectra' are convenience wrapper functions that easy the creation of plots from spectral objects at the cost of lacking the flexibility of the grammar of graphics. The plot object returned is a ggplot (an object of class `"gg"`) and it can be added to or modified as any other ggplot. The axis labels are encoded as *plotmath* expressions as they contain superscripts and special characters. In 'ggplot2', plotmath expressions do not obey theme settings related to text fonts, except for `size`.

Scale limits are expanded so as to make space for the annotations. If annotations are disabled, limits are not expanded unless `reserve.space` is passed to parameter `annotations`.

The generic of the `autoplot` method is defined in package 'ggplot2'. Package 'ggspectra' defines specializations for the different classes for storage of spectral data defined in package `photobiology`.

For details about normalization and arguments to parameter `norm`, please, see `normalize`. If `norm = NA`, the default, `normalize()` is not called. All other values passed as argument to `norm` result in a call to `normalize()` with this value as its argument. In the case of objects created with 'photobiology' (<= 0.10.9) `norm = "undo"` is not supported. Be aware that calls to `normalize()` remove any scaling previously applied with `fscale` methods.

For multiple spectra in long form spectral objects, with `idfactor = NULL`, the default, the name of the factor is retrieved from metadata. If the character string passed as argument to `idfactor` does not match the one retrieved from the object, results in renaming of the pre-existing factor. The default for collections of spectra is to create a factor named `"spct.idx"`, but if a different name is passed, it will be used instead.

Value

A `ggplot` object with a number of layers that depends on the data and annotations. The `data` member retains its original class and metadata attributes.

Plot Annotations

The recognized annotation names are: `"summaries"`, `"peaks"`, `"peak.labels"`, `"valleys"`, `"valley.labels"`, `"wls"`, `"wls.labels"`, `"colour.guide"`, `"color.guide"`, `"boxes"`, `"segments"`, `"labels"`. In addition, `"+"` is interpreted as a request to add to the already present default annotations, `"-"` as request to remove annotations and `"="` or missing`"+"` and `"-"` as a request to reset annotations to those requested. If used, `"+"`, `"-"` or `"="` must be the first member of a character vector, and followed by one or more of the names given above. To simultaneously add and remove annotations one can pass a `list` containing `character` vectors each assembled as described. The vectors are applied in the order they appear in the list. To disable all annotations pass `""` or `c("=", "")` as argument. Adding a variation of an annotation already present, replaces the existing one automatically: e.g., adding `"peak.labels"` replaces`"peaks"` if present.

The annotation layers are added to the plot using statistics defined in 'ggspectra': `stat_peaks`, `stat_valleys`, `stat_label_peaks`, `stat_label_valleys`, `stat_find_wls`, `stat_spikes`, `stat_wb_total`, `stat_wb_mean`, `stat_wb_irrad`, `stat_wb_sirrad`, `stat_wb_contribution`, `stat_wb_relative`, and `stat_wl_strip`. However, only some of their parameters can be passed arguments through `autoplot` methods. In some cases the defaults used by `autoplot` methods are not the defaults of the statistics.

Title Annotations

metadata retrieved from object `object` is paased 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 metatdata 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. Default is `"title:objt"` or no title depending on the context.

See Also

`normalize`, `source_spct`, `waveband`, `photobiologyWavebands-package` and `autoplot`

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()`

Examples

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

# multiple spectra in long form
autoplot(sun_evening.spct)
autoplot(sun_evening.spct, facets = 1) # one column
autoplot(sun_evening.spct, facets = 2) # two columns
autoplot(sun_evening.spct, plot.data = "mean")
autoplot(sun_evening.spct, idfactor = "Sequence")

# multiple spectra as a collection
autoplot(sun_evening.mspct)
# other examples above using .mspct instead of .spct
```

`autoplot.waveband` *Create a complete ggplot for a waveband descriptor.*

Description

Construct 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"),
  unit.out = unit.in,
  annotations = NULL,
  by.group = FALSE,
  geom = "line",
  wb.trim = TRUE,
  norm = NA,
  text.size = 2.5,
  ylim = c(NA, NA),
  object.label = deparse(substitute(object)),
  na.rm = TRUE
)
```

Arguments

`object`	a waveband object.
`...`	arguments passed along by name to `autoplot.response_spct()`.
`w.length`	numeric vector of wavelengths (nm).
`range`	an R object on which range() returns a vector of length 2, with min annd max wavelengths (nm).
`fill`	value to use as response for wavelngths 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`, `unit.out`	the type of unit we assume as reference: "energy" or "photon" based for the waveband definition and the implicit matching response plotted.
`annotations`	a character vector. For details please see section Plot Annotations.
`by.group`	logical flag If TRUE repeated identical annotation layers are added for each group within a plot panel as needed for animation. If `FALSE`, the default, single layers are added per panel.
`geom`	character The name of a ggplot geometry, currently only `"area"`, `"spct"` and `"line"`.
`wb.trim`	logical. Passed to `trim_wl`. Relevant only when the `waveband` extends partly outside `range`.
`norm`	numeric or character Normalization wavelength (nm) or character string `"max"` or other criterion for normalization.
`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.

Details

A `response_spct` object is created based on the `waveband` object. A `waveband` object can describe either a simple wavelength range or a (biological) spectral weighting function (BSWF). See `autoplot.response_spct` for additional details.

Effectiveness spectra are plotted expressing the spectral effectiveness either as $1mol^{-1}nm$ photons of $1J^{-1}nm$ which can be selected through formal argument `unit.out`. The value of `unit.in` has no effect on the result when uisng 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 units, changes the responsivity according to Plank's law.

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.

Value

a `ggplot` object.

Plot Annotations

The recognized annotation names are: `"summaries"`, `"peaks"`, `"peak.labels"`, `"valleys"`, `"valley.labels"`, `"wls"`, `"wls.labels"`, `"colour.guide"`, `"color.guide"`, `"boxes"`, `"segments"`, `"labels"`. In addition, `"+"` is interpreted as a request to add to the already present default annotations, `"-"` as request to remove annotations and `"="` or missing`"+"` and `"-"` as a request to reset annotations to those requested. If used, `"+"`, `"-"` or `"="` must be the first member of a character vector, and followed by one or more of the names given above. To simultaneously add and remove annotations one can pass a `list` containing `character` vectors each assembled as described. The vectors are applied in the order they appear in the list. To disable all annotations pass `""` or `c("=", "")` as argument. Adding a variation of an annotation already present, replaces the existing one automatically: e.g., adding `"peak.labels"` replaces`"peaks"` if present.

The annotation layers are added to the plot using statistics defined in 'ggspectra': `stat_peaks`, `stat_valleys`, `stat_label_peaks`, `stat_label_valleys`, `stat_find_wls`, `stat_spikes`, `stat_wb_total`, `stat_wb_mean`, `stat_wb_irrad`, `stat_wb_sirrad`, `stat_wb_contribution`, `stat_wb_relative`, and `stat_wl_strip`. However, only some of their parameters can be passed arguments through `autoplot` methods. In some cases the defaults used by `autoplot` methods are not the defaults of the statistics.

See Also

`autoplot.response_spct`, `waveband`.

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()`

Examples

```
autoplot(waveband(c(400, 500)))
autoplot(waveband(c(400, 500)), geom = "spct")
```

`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 from an spectral object.

Usage

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

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

Arguments

`object`	generic_spct or generic_mspct 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"`.

Value

The return value of `ggplot2::labs()`.

Title Annotations

metadata retrieved from object `object` is paased 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 metatdata 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. Default is `"title:objt"` or no title depending on the context.

Note

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

Examples

```
p <- ggplot(sun.spct) +
  geom_line()

p + autotitle(sun.spct)
p + autotitle(sun.spct, object.label = "The terrestrial solar spectrum")
p + autotitle(sun.spct, annotations = "title:objt:class")
p + autotitle(sun.spct, annotations = "title:where:when:how")

p <- ggplot(sun_evening.spct) +
  aes(linetype = spct.idx) +
  geom_line()

p + autotitle(sun_evening.spct, annotations = "title:objt:class")
p + autotitle(sun_evening.spct, annotations = "title:where:when:how")
p + autotitle(sun_evening.spct, annotations = "title:none:none:how")

p <- ggplot(sun_evening.mspct) +
  aes(linetype = spct.idx) +
  geom_line()
```

```
p + autotitle(sun_evening.mspct, annotations = "title:objt:class")
```

`axis_labels_uk` *Default text for axis labels*

Description

Texts used by default for axis labels in plots are recalled from character vectors returned by these functions. The aim is that their default values can be easily changed or translated to other languages. They contain only the text part, but not symbols or units of expression.

Usage

```
axis_labels_uk(append = "", sep = "")

axis_labels_none()

axis_labels(append = "", sep = "")
```

Arguments

`append`	character The string to be appended to each label,
`sep`	character Passed to function `paste` as argument for parameter `sep`.

Details

By default `axis_labels()` contains a copy of `axis_labels_uk_comma()`. By assigning to this name a user function that returns a named character vector using the same names for its members as those returned by these functions, it is possible to temporarily change the default texts.

Currently only UK English label texts are predefined and `axis_labels()` is a synonym of `axis_labels_uk()`.

Value

A character vector

Examples

```
names(axis_labels())

axis_labels()[["w.length"]] # no comma
axis_labels(append = ",")[["w.length"]] # ending in a comma

axis_labels_uk()[["w.length"]] # English (same as default)
axis_labels_none()[["w.length"]] # empty label
```

`A_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

```
A_label(
  unit.exponent = 0,
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = NULL,
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  Tfr.type
)

A_internal_label(
  unit.exponent = 0,
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = NULL,
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE)
)

A_total_label(
  unit.exponent = 0,
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = NULL,
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE)
)
```

Arguments

`unit.exponent`	integer
`format`	character string, "R", "R.expresion", "R.character", or "LaTeX".
`label.text`	character Textual portion of the labels.
`scaled`	logical If `TRUE` relative units are assumed.
`normalized`	logical (`FALSE`) or numeric Normalization wavelength in manometers (nm).

`axis.symbols`	logical If `TRUE` symbols of the quantities are added to the `name`. Supported only by `format = "R.expression"`.
`Tfr.type`	character, either "total" or "internal".

Value

a character string or an R expression.

Note

Default for `label.text` depends on the value passed as argument to `Tfr.type`.

Examples

```
A_label(Tfr.type = "internal")
A_label(Tfr.type = "total")
A_label(Tfr.type = "total", axis.symbols = FALSE)


A_internal_label()
A_internal_label(format = "R.expression", axis.symbols = FALSE)
A_internal_label(-3)
A_internal_label(format = "R.expression")
A_internal_label(format = "LaTeX")
A_internal_label(-3, format = "LaTeX")


A_total_label()
A_total_label(format = "R.expression", axis.symbols = FALSE)
A_total_label(-3)
A_total_label(format = "R.expression")
A_total_label(format = "LaTeX")
A_total_label(-3, format = "LaTeX")
```

`black_or_white` *Chose black vs. white color based on weighted mean of RGB channels*

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

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

Arguments

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

Examples

```
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

```
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

```
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

```
counts_label(
  unit.exponent = 3,
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["counts"]],
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE)
)
```

Arguments

`unit.exponent`	integer
`format`	character string, "R", "R.expresion", "R.character", or "LaTeX".
`label.text`	character Textual portion of the labels.
`scaled`	logical If `TRUE` relative units are assumed.
`normalized`	logical (`FALSE`) or numeric Normalization wavelength in manometers (nm).
`axis.symbols`	logical If `TRUE` symbols of the quantities are added to the `name`. Supported only by `format = "R.expression"`.

Value

a character string or an R expression.

Examples

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

`cps_label` *Counts-per-second axis labels*

Description

Generate pixel response rate axis labels in cps units. Output can be selected as character, expression (R default devices) or LaTeX (for tikz device).

Usage

```
cps_label(
  unit.exponent = 0,
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["cps"]],
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE)
)
```

Arguments

`unit.exponent`	integer
`format`	character string, "R", "R.expresion", "R.character", or "LaTeX".
`label.text`	character Textual portion of the labels.
`scaled`	logical If `TRUE` relative units are assumed.
`normalized`	logical (`FALSE`) or numeric Normalization wavelength in manometers (nm).
`axis.symbols`	logical If `TRUE` symbols of the quantities are added to the `name`. Supported only by `format = "R.expression"`.

Value

a character string or an R expression.

Examples

```
cps_label()
cps_label(3)
cps_label(format = "R.expression")
cps_label(format = "R.character")
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

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

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

exponent2prefix_name(exponent)

prefix_name2exponent(name)

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

has_SI_prefix(exponent)

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"`. The difference between `"utf8"` and `"ascii"` is that the first uses UTF8 character "micro" (similar to Greek mu) and the second uses "u".
`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

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


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

```
geom_spct(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  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 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.
`...`	other arguments passed on to `layer`. This can include aesthetics whose values you want to set, not map. See `layer` for more details.

`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`.

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

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

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

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

## 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_spct'
ggplot(
  data,
  mapping = NULL,
  ...,
  range = NULL,
  spct_class,
  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()
)

## S3 method for class 'object_mspct'
ggplot(
  data,
  mapping = NULL,
  ...,
  range = NULL,
  plot.qty = getOption("photobiology.object.qty", default = ifelse(length(data) > 1L,
    "as.is", "all")),
  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.
`spct_class`	character Class into which a `generic_spct` object will be converted before plotting. The column names in data should match those expected by the class constructor (see `setGenericSpct`); other arguments should be passed by name).

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 `replector_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

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

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()
```

`multipliers_label` *Calibration multipliers axis labels*

Description

Calibration multipliers axis labels. Output can be selected as character, expression (R default devices) or LaTeX (for tikz device).

Usage

```
multipliers_label(
  unit.exponent = 0,
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["e.mult"]],
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE)
)
```

Arguments

`unit.exponent`	integer
`format`	character string, "R", "R.expression", "R.character", or "LaTeX".
`label.text`	character Textual portion of the labels.
`scaled`	logical If `TRUE` relative units are assumed.
`normalized`	logical (`FALSE`) or numeric Normalization wavelength in manometers (nm).
`axis.symbols`	logical If `TRUE` symbols of the quantities are added to the `name`. Supported only by `format = "R.expression"`.

Value

a character string or an R expression.

Examples

```
multipliers_label()
multipliers_label(3)
multipliers_label(format = "R.expression")
multipliers_label(format = "R.character")
multipliers_label(format = "LaTeX")
multipliers_label(3, format = "LaTeX")
```

`multiplot` *Multiple plot function*

Description

Grid based; allows multiple plots arraged 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

```
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 Licenced under CC BY-SA

References

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

Examples

```
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")
```

`plot.generic_spct` *Deprecated plot methods*

Description

These `plot()` methods return a ggplot object with an annotated plot of an object of a class derived from `generic_spct`, of a class derived from `generic_mspct` or of an object of class `waveband` for which an `autoplot()` method exists. They are implemented as wrappers of `autoplot()`. The generic for `plot()` is defined by base R and specializations for objects of diverse classes are provided various packages and R itself. The generic for `autoplot()` is defined by package 'ggplot2'.

Usage

```
## 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 `autoplot()` methods.

Value

a `ggplot` object.

Deprecation warning!

These `plot()` specializatioms are provided for backwards compatibility, but all new or updated code should call `autoplot()` instead of `plot()` on objects of spectral and waveband classes defined in package 'photobiology'.

These methods add support for `plot()` specializations as these specialization were provided by package 'ggspectra' years ago, before 'ggplot2' had an `autoplot()` generic. As these methods return ggplots autoplot is a more suitable name for them.

See Also

`autoplot.calibration_spct`, `autoplot.cps_spct`, `autoplot.filter_spct`, `autoplot.raw_spct`, `autoplot.response_spct`, `autoplot.source_spct` and `autoplot.waveband`.

Examples

```
plot(sun.spct) # deprecated syntax, to be avoided
autoplot(sun.spct) # current syntax, to be used
```

`Rfr_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_label(
  unit.exponent = ifelse(pc.out, -2, 0),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = NULL,
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  pc.out = getOption("ggspectra.pc.out", default = FALSE),
  Rfr.type
)

Rfr_specular_label(
  unit.exponent = ifelse(pc.out, -2, 0),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = NULL,
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  pc.out = getOption("ggspectra.pc.out", default = FALSE)
)
```

Arguments

`unit.exponent`	integer
`format`	character string, "R", "R.expresion", "R.character", or "LaTeX".
`label.text`	character Textual portion of the labels.

`scaled`	logical If `TRUE` relative units are assumed.
`normalized`	logical (`FALSE`) or numeric Normalization wavelength in manometers (nm).
`axis.symbols`	logical If `TRUE` symbols of the quantities are added to the `name`. Supported only by `format = "R.expression"`.
`pc.out`	logical, if TRUE use percent as default instead of fraction of one.
`Rfr.type`	character, either "total" or "specular".

Value

a character string or an R expression.

Note

Default for `label.text` depends on the value passed as argument to `Rfr.type`.

Examples

```
Rfr_label(Rfr.type = "specular")
Rfr_label(Rfr.type = "total")


Rfr_specular_label()
Rfr_specular_label(axis.symbols = FALSE)
Rfr_specular_label(-2)
Rfr_specular_label(-3)
Rfr_specular_label(format = "R.expression")
Rfr_specular_label(format = "LaTeX")
Rfr_specular_label(-3, format = "LaTeX")
```

`s.e.irrad_label` *Spectral irradiance axis labels*

Description

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

Usage

```
s.e.irrad_label(
  unit.exponent = NULL,
  markup.format = getOption("photobiology.math", default = "R.expression"),
  time.unit = "second",
  label.text = NULL,
  pc.out = FALSE,
  scaled = FALSE,
```

```
  normalised = FALSE,
  normalized = normalised,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE)
)

s.q.irrad_label(
  unit.exponent = NULL,
  markup.format = getOption("photobiology.math", default = "R.expression"),
  time.unit = "second",
  label.text = NULL,
  pc.out = FALSE,
  scaled = FALSE,
  normalised = FALSE,
  normalized = normalised,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE)
)
```

Arguments

`unit.exponent`	integer. The default is guessed from `time.unit`, `scaled` and `normalized`.
`markup.format`	character string, "R", "R.expresion", "r.character", or "LaTeX".
`time.unit`	character or duration The length of time used as base of expression.
`label.text`	character Textual portion of the labels.
`pc.out`	logical Flag to enable display of normalised data as percentages.
`scaled`	logical If `TRUE` relative units are assumed.
`normalized`, `normalised`	logical (`FALSE`) or numeric Normalization wavelength in manometers (nm).
`axis.symbols`	logical If `TRUE` symbols of the quantities are added to the `name`. Supported only by `format = "R.expression"`.

Value

a character string or an R expression.

Examples

```
str(s.e.irrad_label())
str(s.e.irrad_label(axis.symbols = FALSE))
str(s.e.irrad_label(markup.format = "R.expression"))
str(s.e.irrad_label(markup.format = "LaTeX"))
str(s.e.irrad_label(markup.format = "R.character"))

str(s.q.irrad_label())
str(s.q.irrad_label(axis.symbols = FALSE))
str(s.q.irrad_label(markup.format = "R.expression"))
str(s.q.irrad_label(markup.format = "LaTeX"))
str(s.q.irrad_label(markup.format = "R.character"))
```

---

s.e.response_label *spectral response and action axis labels*

Description

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

Usage

```
s.e.response_label(
  unit.exponent = 0,
  format = getOption("photobiology.math", default = "R.expression"),
 label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["s.e.response"]],
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE)
)

s.q.response_label(
  unit.exponent = 0,
  format = getOption("photobiology.math", default = "R.expression"),
 label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["s.q.response"]],
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE)
)

s.e.action_label(
  unit.exponent = 0,
  format = getOption("photobiology.math", default = "R.expression"),
 label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["s.e.action"]],
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE)
)

s.q.action_label(
  unit.exponent = 0,
  format = getOption("photobiology.math", default = "R.expression"),
 label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["s.q.action"]],
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE)
)
```

Arguments

`unit.exponent`	integer
`format`	character string, "R", "R.expression", "R.character", or "LaTeX".
`label.text`	character Textual portion of the labels.
`scaled`	logical If `TRUE` relative units are assumed.
`normalized`	logical (`FALSE`) or numeric Normalization wavelength in manometers (nm).
`axis.symbols`	logical If `TRUE` symbols of the quantities are added to the `name`. Supported only by `format = "R.expression"`.

Value

a character string or an R expression.

Examples

```
s.e.response_label()
s.e.response_label(format = "R.expression")
s.e.response_label(format = "R.character")
s.e.response_label(format = "LaTeX")
s.e.response_label(unit.exponent = 3, format = "R.character")
s.q.response_label(format = "R.character")
s.e.action_label(format = "R.character")
s.q.action_label(format = "R.character")
s.e.response_label(scaled = TRUE)
s.e.response_label(scaled = TRUE, format = "R.character")
s.e.response_label(scaled = TRUE, format = "LaTeX")
s.e.response_label(normalized = 300)
s.e.response_label(normalized = 300, format = "R.character")
s.e.response_label(normalized = 300, format = "LaTeX")
s.q.response_label(scaled = TRUE)
s.q.response_label(scaled = TRUE, format = "R.character")
s.q.response_label(scaled = TRUE, format = "LaTeX")
s.q.response_label(normalized = 300)
s.q.response_label(normalized = 300, format = "R.character")
s.q.response_label(normalized = 300, format = "LaTeX")
```

---

scale_x_energy_eV_continuous

Energy per photon x-scale

Description

Scale x continuous with defaults suitable for wavelengths expressed as energy per photon [eV] or [J].

Usage

```
scale_x_energy_eV_continuous(
  unit.exponent = 0,
 name = w_energy_eV_label(unit.exponent = unit.exponent, label.text = label.text,
    axis.symbols = axis.symbols),
  breaks = scales::pretty_breaks(n = 7),
  labels = SI_pl_format(exponent = unit.exponent),
  label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["energy"]],
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  ...
)

scale_x_energy_J_continuous(
  unit.exponent = -18,
 name = w_energy_J_label(unit.exponent = unit.exponent, label.text = label.text,
    axis.symbols = axis.symbols),
  breaks = scales::pretty_breaks(n = 7),
  labels = SI_pl_format(exponent = unit.exponent),
  label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["energy"]],
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  ...
)
```

Arguments

`unit.exponent`	integer
`name`	The name of the scale, used for the axis-label.
`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.
`label.text`	character Textual portion of the labels.
`axis.symbols`	logical If `TRUE` symbols of the quantities are added to the `name`. Supported only by `format = "R.expression"`.
`...`	other named arguments passed to `scale_y_continuous`

Details

This scale automates the generation of axis labels when the variable mapped to the *x* aesthetic contains numeric values for wavelengths expressed as energy per photon. This is **not** how spectral data are stored in all the packages of the R for Photobiology suite and can be used in plots built with `ggplot2()` with explicit mapping using a conversion function. If desired, a secondary axis can be added manually as described in `sec_axis`.

Note

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

Examples

```
ggplot(sun.spct, aes(x = wl2energy(w.length, unit = "joule"), y = s.e.irrad)) +
  geom_line() +
  scale_x_energy_J_continuous()

ggplot(sun.spct, aes(x = wl2energy(w.length, unit = "joule"), y = s.e.irrad)) +
  geom_line() +
  scale_x_energy_J_continuous(unit.exponent = -19)

ggplot(sun.spct, aes(x = wl2energy(w.length, unit = "eV"), y = s.e.irrad)) +
  geom_line() +
  scale_x_energy_eV_continuous()

ggplot(sun.spct, aes(x = wl2energy(w.length, unit = "eV"), y = s.e.irrad)) +
  geom_line() +
  scale_x_energy_eV_continuous(unit.exponent = -3)
```

`scale_x_frequency_continuous`

Frequency x-scale

Description

Scale x continuous with defaults suitable for wavelengths expressed as frequencies [Hz].

Usage

```
scale_x_frequency_continuous(
  unit.exponent = 12,
 name = w_frequency_label(unit.exponent = unit.exponent, label.text = label.text,
    axis.symbols = axis.symbols),
  breaks = scales::pretty_breaks(n = 7),
  labels = SI_pl_format(exponent = unit.exponent),
  label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["freq"]],
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  ...
)
```

Arguments

`unit.exponent`	integer
`name`	The name of the scale, used for the axis-label.
`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.
`label.text`	character Textual portion of the labels.

`axis.symbols`	logical If `TRUE` symbols of the quantities are added to the `name`. Supported only by `format = "R.expression"`.
`...`	other named arguments passed to `scale_y_continuous`

Details

This scale automates the generation of axis labels when the variable mapped to the *x* aesthetic contains numeric values for wavelengths expressed as frequency. This is **not** how spectral data are stored in the packages of the R for Photobiology suite and can be only used in plots built with `ggplot2()` with explicit mapping using a conversion function. If desired, a secondary axis can be added manually as described in `sec_axis`.

Note

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

Examples

```
ggplot(sun.spct, aes(x = wl2frequency(w.length), y = s.e.irrad)) +
  geom_line() +
  scale_x_frequency_continuous()

ggplot(sun.spct, aes(x = wl2frequency(w.length), y = s.e.irrad)) +
  geom_line() +
  scale_x_frequency_continuous(14)
```

---

scale_x_wavenumber_continuous

Wavenumber x-scale

Description

Scale x continuous with defaults suitable for wavelengths expressed as wavenumbers $[m^{-2}]$.

Usage

```
scale_x_wavenumber_continuous(
  unit.exponent = -6,
  name = w_number_label(unit.exponent = unit.exponent, label.text = label.text,
    axis.symbols = axis.symbols),
  breaks = scales::pretty_breaks(n = 7),
  labels = SI_pl_format(exponent = -unit.exponent),
 label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["w.number"]],
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  ...
)
```

Arguments

`unit.exponent`	integer
`name`	The name of the scale, used for the axis-label.
`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.
`label.text`	character Textual portion of the labels.
`axis.symbols`	logical If `TRUE` symbols of the quantities are added to the `name`. Supported only by `format = "R.expression"`.
`...`	other named arguments passed to `scale_y_continuous`

Details

This scale automates the generation of axis labels when the variable mapped to the *x* aesthetic contains numeric values for wavelengths expressed wavenumbers. This is **not** how spectral data are stored in all the packages of the R for Photobiology suite and can be used in plots built with `ggplot2()` with explicit mapping using a conversion function. If desired, a secondary axis can be added manually as described in `sec_axis`.

Note

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

Examples

```
ggplot(sun.spct, aes(x = wl2wavenumber(w.length), y = s.e.irrad)) +
  geom_line() +
  scale_x_wavenumber_continuous()

ggplot(sun.spct, aes(x = wl2wavenumber(w.length), y = s.e.irrad)) +
  geom_line() +
  scale_x_wavenumber_continuous(unit.exponent = -5)
```

`scale_x_wl_continuous` *Wavelength x-scale*

Description

Scale x continuous with defaults suitable for wavelengths in nanometres.

Usage

```
scale_x_wl_continuous(
  unit.exponent = -9,
  name = w_length_label(unit.exponent = unit.exponent, label.text = label.text,
    axis.symbols = axis.symbols),
  breaks = scales::pretty_breaks(n = 7),
  labels = SI_pl_format(exponent = unit.exponent + 9),
 label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["w.length"]],
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  ...
)
```

Arguments

`unit.exponent`	integer
`name`	The name of the scale, used for the axis-label.
`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.
`label.text`	character Textual portion of the labels.
`axis.symbols`	logical If `TRUE` symbols of the quantities are added to the `name`. Supported only by `format = "R.expression"`.
`...`	other named arguments passed to `scale_y_continuous`

Details

This scale automates the generation of axis labels when the variable mapped to the *x* aesthetic contains numeric values for wavelengths expressed in nanometres. This is how spectral data are stored in all the packages of the R for Photobiology suite, inlcuding the the expected data by the `autoplot()` methods defined in 'ggspectra'.

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(unit.exponent = -6)

ggplot(sun.spct) +
  geom_line() +
  scale_x_wl_continuous(label.text = "Longitud de onda,")
```

```
autoplot(sun.spct) +
  scale_x_wl_continuous(label.text = "Longitud de onda,",
                        unit.exponent = -6)
```

---

scale_y_Afr_continuous
Absorptance y-scale

Description

Scale y continuous with defaults suitable for spectral absorptance.

Usage

```
scale_y_Afr_continuous(
  unit.exponent = ifelse(pc.out, -2, 0),
  name = Afr_label(unit.exponent = unit.exponent, format = format, label.text =
   label.text, scaled = scaled, normalized = round(normalized, 1), axis.symbols =
    axis.symbols),
  labels = SI_pl_format(exponent = unit.exponent),
  limits = c(0, 1),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["s.Afr"]],
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  pc.out = getOption("ggspectra.pc.out", default = FALSE),
  ...
)
```

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.
`limits`	One of `NULL` for default based on data range, a numeric vector of length two (`NA` allowed) or a function that accepts the data-based limits as argument and returns new limits.
`format`	character string, "R", "R.expression", "R.character", or "LaTeX".
`label.text`	character Textual portion of the labels.
`scaled`	logical If `TRUE` relative units are assumed.
`normalized`	logical (`FALSE`) or numeric Normalization wavelength in manometers (nm).
`axis.symbols`	logical If `TRUE` symbols of the quantities are added to the `name`. Supported only by `format = "R.expression"`.
`pc.out`	logical, if TRUE use percent as default instead of fraction of one.
`...`	other named arguments passed to `scale_y_continuous`

Note

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

Examples

```
Afr_as_default()

ggplot(yellow_gel.spct) +
  geom_line() +
  scale_y_Afr_continuous() +
  scale_x_wl_continuous()

ggplot(yellow_gel.spct) +
  geom_line() +
  scale_y_Afr_continuous(unit.exponent = -2) +
  scale_x_wl_continuous()

ggplot(yellow_gel.spct) +
  geom_line() +
  scale_y_Afr_continuous(unit.exponent = -3) +
  scale_x_wl_continuous()

ggplot(yellow_gel.spct) +
  geom_line() +
  scale_y_Afr_continuous(axis.symbols = FALSE) +
  scale_x_wl_continuous(axis.symbols = FALSE)

unset_filter_qty_default()
```

`scale_y_A_continuous` *Absorbance y-scale*

Description

Scale y continuous with defaults suitable for spectral absorbance.

Usage

```
scale_y_A_continuous(
  unit.exponent = 0,
 name = A_label(unit.exponent = unit.exponent, format = format, label.text = label.text,
   scaled = scaled, normalized = ifelse(is.numeric(normalized), round(normalized, 1),
     unique(normalized)), axis.symbols = axis.symbols, Tfr.type = Tfr.type),
  labels = SI_pl_format(exponent = unit.exponent),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = NULL,
  scaled = FALSE,
  normalized = FALSE,
```

```
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  Tfr.type,
  ...
)

scale_y_A_internal_continuous(
  unit.exponent = 0,
 name = A_label(unit.exponent = unit.exponent, format = format, label.text = label.text,
   scaled = scaled, normalized = ifelse(is.numeric(normalized), round(normalized, 1),
    unique(normalized)), axis.symbols = axis.symbols, Tfr.type = "internal"),
  labels = SI_pl_format(exponent = unit.exponent),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = NULL,
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  ...
)

scale_y_A_total_continuous(
  unit.exponent = 0,
 name = A_label(unit.exponent = unit.exponent, format = format, label.text = label.text,
   scaled = scaled, normalized = ifelse(is.numeric(normalized), round(normalized, 1),
    unique(normalized)), axis.symbols = axis.symbols, Tfr.type = "total"),
  labels = SI_pl_format(exponent = unit.exponent),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = NULL,
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  ...
)
```

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.
`format`	character string, "R", "R.expression", "R.character", or "LaTeX".
`label.text`	character Textual portion of the labels.
`scaled`	logical If `TRUE` relative units are assumed.
`normalized`	logical (`FALSE`) or numeric Normalization wavelength in manometers (nm).
`axis.symbols`	logical If `TRUE` symbols of the quantities are added to the `name`. Supported only by `format = "R.expression"`.
`Tfr.type`	character, either "total" or "internal".
`...`	other named arguments passed to `scale_y_continuous`

Note

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

Examples

```
ggplot(yellow_gel.spct, plot.qty = "absorbance") +
  geom_line() +
  scale_y_A_continuous(Tfr.type = getTfrType(yellow_gel.spct)) +
  scale_x_wl_continuous()

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

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

ggplot(yellow_gel.spct, plot.qty = "absorbance") +
  geom_line() +
  scale_y_A_total_continuous(axis.symbols = FALSE) +
  scale_x_wl_continuous(axis.symbols = FALSE)

ggplot(yellow_gel.spct, plot.qty = "absorbance") +
  geom_line() +
  scale_y_A_internal_continuous(normalized = "none") +
  scale_x_wl_continuous()
```

`scale_y_counts_continuous`

Raw-counts y-scale

Description

Scale y continuous with defaults suitable for raw detector counts.

Usage

```
scale_y_counts_continuous(
  unit.exponent = ifelse(normalized, 0, 3),
  name = counts_label(unit.exponent = unit.exponent, format = format, label.text =
    label.text, scaled = scaled, normalized = ifelse(is.numeric(normalized),
    round(normalized, 1), unique(normalized)), axis.symbols = axis.symbols),
  labels = SI_pl_format(exponent = unit.exponent),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["counts"]],
```

```
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  ...
)

scale_y_counts_tg_continuous(
  unit.exponent = ifelse(normalized, 0, 3),
 name = counts_label(unit.exponent = 0, format = format, label.text = label.text, scaled
    = scaled, normalized = ifelse(is.numeric(normalized), round(normalized, 1),
    unique(normalized)), axis.symbols = axis.symbols),
  labels = SI_tg_format(exponent = unit.exponent),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["counts"]],
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  ...
)
```

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.
`format`	character string, "R", "R.expression", "R.character", or "LaTeX".
`label.text`	character Textual portion of the labels.
`scaled`	logical If `TRUE` relative units are assumed.
`normalized`	logical (`FALSE`) or numeric Normalization wavelength in manometers (nm).
`axis.symbols`	logical If `TRUE` symbols of the quantities are added to the `name`. Supported only by `format = "R.expression"`.
`...`	other named arguments passed to `scale_y_continuous`

Note

This function only alters default arguments values for `name` and `labels`, please, see documentation for `scale_continuous` for other parameters.

Examples

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

ggplot(white_led.raw_spct) +
  geom_line() +
```

```
  scale_y_counts_continuous(unit.exponent = 0) +
  scale_x_wl_continuous()

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

ggplot(white_led.raw_spct) +
  geom_line() +
  scale_y_counts_tg_continuous(unit.exponent = 0) +
  scale_x_wl_continuous()

if (packageVersion("photobiology") > "0.11.4") {
  norm_led.raw_spct <- normalize(white_led.raw_spct, norm = "max")

  ggplot(norm_led.raw_spct) +
    geom_line() +
    scale_y_counts_continuous(unit.exponent = 0, normalized = "max") +
    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, format = format, label.text =
    label.text, scaled = scaled, normalized = ifelse(is.numeric(normalized),
    round(normalized, 1), unique(normalized)), axis.symbols = axis.symbols),
  labels = SI_pl_format(exponent = unit.exponent),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["cps"]],
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  ...
)
```

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.
`format`	character string, "R", "R.expression", "R.character", or "LaTeX".
`label.text`	character Textual portion of the labels.
`scaled`	logical If `TRUE` relative units are assumed.
`normalized`	logical (`FALSE`) or numeric Normalization wavelength in manometers (nm).
`axis.symbols`	logical If `TRUE` symbols of the quantities are added to the `name`. Supported only by `format = "R.expression"`.
`...`	other named arguments passed to `scale_y_continuous`

Note

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

Examples

```
ggplot(white_led.cps_spct) +
  geom_line() +
  scale_y_cps_continuous() +
  scale_x_wl_continuous()

ggplot(white_led.cps_spct) +
  geom_line() +
  scale_y_cps_continuous(3) +
  scale_x_wl_continuous()

ggplot(white_led.cps_spct * 1e-4) +
  geom_line() +
  scale_y_cps_continuous(scaled = TRUE) +
  scale_x_wl_continuous()

if (packageVersion("photobiology") > "0.11.4") {
  norm_led.cps_spct <- normalize(white_led.cps_spct, norm = "max")

  ggplot(norm_led.cps_spct) +
    geom_line() +
    scale_y_cps_continuous(normalized = is_normalized(norm_led.cps_spct)) +
    scale_x_wl_continuous()

  ggplot(norm_led.cps_spct) +
    geom_line() +
    scale_y_cps_continuous(normalized = getNormalized(norm_led.cps_spct)) +
    scale_x_wl_continuous()

  ggplot(norm_led.cps_spct) +
    geom_line() +
    scale_y_cps_continuous(normalized =
```

```
        normalization(norm_led.cps_spct)$norm.type) +
      scale_x_wl_continuous()
  }
```

---

scale_y_multipliers_continuous

Calibration multipliers y-scale

Description

Scale y continuous with defaults suitable for raw the calibration multipliers used to convert pixel response rate (counts per second) into energy irradiance units.

Usage

```
scale_y_multipliers_continuous(
  unit.exponent = 0,
 name = multipliers_label(unit.exponent = unit.exponent, format = format, label.text =
    label.text, scaled = scaled, normalized = ifelse(is.numeric(normalized),
    round(normalized, 1), unique(normalized)), axis.symbols = axis.symbols),
  labels = SI_pl_format(exponent = unit.exponent),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["e.mult"]],
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  ...
)
```

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.
`format`	character string, "R", "R.expression", "R.character", or "LaTeX".
`label.text`	character Textual portion of the labels.
`scaled`	logical If `TRUE` relative units are assumed.
`normalized`	logical (`FALSE`) or numeric Normalization wavelength in manometers (nm).
`axis.symbols`	logical If `TRUE` symbols of the quantities are added to the `name`. Supported only by `format = "R.expression"`.
`...`	other named arguments passed to `scale_y_continuous`

Note

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

`scale_y_Rfr_continuous` *Reflectance y-scale*

Description

Scale y continuous with defaults suitable for spectral reflectance.

Usage

```
scale_y_Rfr_continuous(
  unit.exponent = ifelse(pc.out, -2, 0),
  name = Rfr_label(unit.exponent = unit.exponent, format = format, label.text =
    label.text, scaled = scaled, normalized = ifelse(is.numeric(normalized),
   round(normalized, 1), unique(normalized)), axis.symbols = axis.symbols, Rfr.type =
    Rfr.type),
  labels = SI_pl_format(exponent = unit.exponent),
  limits = c(0, 1),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = NULL,
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  pc.out = getOption("ggspectra.pc.out", default = FALSE),
  Rfr.type,
  ...
)

scale_y_Rfr_specular_continuous(
  unit.exponent = ifelse(pc.out, -2, 0),
  name = Rfr_label(unit.exponent = unit.exponent, format = format, label.text =
    label.text, scaled = scaled, normalized = ifelse(is.numeric(normalized),
   round(normalized, 1), unique(normalized)), axis.symbols = axis.symbols, Rfr.type =
    "specular"),
  labels = SI_pl_format(exponent = unit.exponent),
  limits = c(0, 1),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = NULL,
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  pc.out = getOption("ggspectra.pc.out", default = FALSE),
  ...
)

scale_y_Rfr_total_continuous(
  unit.exponent = ifelse(pc.out, -2, 0),
```

```
  name = Rfr_label(unit.exponent = unit.exponent, format = format, label.text =
    label.text, scaled = scaled, normalized = ifelse(is.numeric(normalized),
   round(normalized, 1), unique(normalized)), axis.symbols = axis.symbols, Rfr.type =
    "total"),
  labels = SI_pl_format(exponent = unit.exponent),
  limits = c(0, 1),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = NULL,
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  pc.out = getOption("ggspectra.pc.out", default = FALSE),
  ...
)
```

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.
`limits`	One of `NULL` for default based on data range, a numeric vector of length two (`NA` allowed) or a function that accepts the data-based limits as argument and returns new limits.
`format`	character string, "R", "R.expression", "R.character", or "LaTeX".
`label.text`	character Textual portion of the labels.
`scaled`	logical If `TRUE` relative units are assumed.
`normalized`	logical (`FALSE`) or numeric Normalization wavelength in manometers (nm).
`axis.symbols`	logical If `TRUE` symbols of the quantities are added to the `name`. Supported only by `format = "R.expression"`.
`pc.out`	logical, if TRUE use percent as default instead of fraction of one.
`Rfr.type`	character, either "total" or "spcular".
`...`	other named arguments passed to `scale_y_continuous`

Note

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

Examples

```
ggplot(Ler_leaf_rflt.spct) +
  geom_line() +
  scale_y_Rfr_continuous(Rfr.type = getRfrType(Ler_leaf_rflt.spct)) +
  scale_x_wl_continuous()

ggplot(Ler_leaf_rflt.spct) +
  geom_line() +
```

```
  scale_y_Rfr_continuous(unit.exponent = -2,
                         Rfr.type = getRfrType(Ler_leaf_rflt.spct)) +
  scale_x_wl_continuous()

ggplot(Ler_leaf_rflt.spct) +
  geom_line() +
  scale_y_Rfr_continuous(unit.exponent = -3,
                         Rfr.type = getRfrType(Ler_leaf_rflt.spct)) +
  scale_x_wl_continuous()

ggplot(Ler_leaf_rflt.spct) +
  geom_line() +
  scale_y_Rfr_specular_continuous() +
  scale_x_wl_continuous()

ggplot(Ler_leaf_rflt.spct) +
  geom_line() +
  scale_y_Rfr_specular_continuous(axis.symbols = FALSE) +
  scale_x_wl_continuous(axis.symbols = FALSE)

ggplot(normalize(Ler_leaf_rflt.spct)) +
  geom_line() +
  scale_y_Rfr_continuous(Rfr.type = getRfrType(Ler_leaf_rflt.spct),
     normalized = "max") +
  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

```
scale_y_s.e.irrad_continuous(
  unit.exponent = 0,
 name = s.e.irrad_label(unit.exponent = unit.exponent, markup.format = markup.format,
   time.unit = "second", label.text = label.text, pc.out = pc.out, scaled = scaled,
    normalized = normalized, axis.symbols = axis.symbols),
  labels = SI_pl_format(exponent = unit.exponent - pc.out * 2),
  markup.format = getOption("photobiology.math", default = "R.expression"),
 label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["s.e.irrad"]],
  pc.out = FALSE,
  scaled = FALSE,
  normalised = FALSE,
  normalized = normalised,
```

```
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  ...
)

scale_y_s.q.irrad_continuous(
  unit.exponent = -6,
  name = s.q.irrad_label(unit.exponent = unit.exponent, markup.format = markup.format,
    time.unit = "second", label.text = label.text, pc.out = pc.out, scaled = scaled,
    normalized = normalized, axis.symbols = axis.symbols),
  labels = SI_pl_format(exponent = unit.exponent - pc.out * 2),
  markup.format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["s.q.irrad"]],
  pc.out = FALSE,
  scaled = FALSE,
  normalised = FALSE,
  normalized = normalised,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  ...
)

scale_y_s.e.irrad_log10(
  unit.exponent = 0,
  name = s.e.irrad_label(unit.exponent = unit.exponent, markup.format = markup.format,
    time.unit = "second", label.text = label.text, pc.out = pc.out, scaled = scaled,
    normalized = normalized, axis.symbols = axis.symbols),
  labels = SI_pl_format(exponent = unit.exponent - pc.out * 2),
  markup.format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["s.e.irrad"]],
  pc.out = FALSE,
  scaled = FALSE,
  normalised = FALSE,
  normalized = normalised,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  ...
)

scale_y_s.q.irrad_log10(
  unit.exponent = -6,
  name = s.q.irrad_label(unit.exponent = unit.exponent, markup.format = markup.format,
    time.unit = "second", label.text = label.text, pc.out = pc.out, scaled = scaled,
    normalized = normalized, axis.symbols = axis.symbols),
  labels = SI_pl_format(exponent = unit.exponent - pc.out * 2),
  markup.format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["s.q.irrad"]],
  pc.out = FALSE,
  scaled = FALSE,
  normalised = FALSE,
  normalized = normalised,
```

```
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  ...
)
```

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.
`markup.format`	character string, "R", "R.expression", "r.character", or "LaTeX".
`label.text`	character Textual portion of the labels.
`pc.out`	logical, if `TRUE` use percent instead of fraction of one for normalized spectral data.
`scaled`	logical If `TRUE` relative units are assumed.
`normalized`, `normalised`	logical (`FALSE`) or numeric Normalization wavelength in manometers (nm).
`axis.symbols`	logical If `TRUE` symbols of the quantities are added to the default `name`.
`...`	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, unit.out = "energy") +
  geom_line() +
  scale_y_s.e.irrad_continuous() +
  scale_x_wl_continuous()

ggplot(sun.spct) +
  geom_line() +
  scale_y_s.e.irrad_continuous(unit.exponent = -3) +
  scale_x_wl_continuous()

ggplot(fscale(sun.spct)) +
  geom_line() +
  scale_y_s.e.irrad_continuous(scaled = TRUE) +
  scale_x_wl_continuous()

ggplot(normalize(sun.spct, norm = "max")) +
  geom_line() +
  scale_y_s.e.irrad_continuous(normalized = "max") +
  scale_x_wl_continuous()

my.spct <- normalize(q2e(sun.spct, action = "replace"), norm = "max")
ggplot(my.spct) +
  geom_line() +
```

```
    scale_y_s.e.irrad_continuous(normalized = normalization(my.spct)$norm.type,
                                 pc.out = TRUE) +
    scale_x_wl_continuous()

  ggplot(my.spct) +
    geom_line() +
    scale_y_s.e.irrad_continuous(normalized = normalization(my.spct)$norm.wl,
                                 pc.out = TRUE) +
    scale_x_wl_continuous()

  ggplot(sun.spct) +
    geom_line() +
    scale_y_s.e.irrad_continuous(axis.symbols = FALSE) +
    scale_x_wl_continuous()

  ggplot(sun.spct) +
    geom_line() +
    scale_y_s.e.irrad_continuous(label.text = "") +
    scale_x_wl_continuous()

  ggplot(sun.spct) +
    geom_line() +
    scale_y_s.e.irrad_continuous(label.text = "Irradiancia espectral,") +
    scale_x_wl_continuous(label.text = "Longitud de onda,")

  ggplot(sun.spct) +
    geom_line() +
    scale_y_s.e.irrad_continuous(unit.exponent = -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 and action y-scales

Description

Scale y continuous with defaults suitable for response and action spectra.

Usage

```
scale_y_s.e.response_continuous(
  unit.exponent = 0,
  name = s.e.response_label(unit.exponent = unit.exponent, format = format, label.text =
    label.text, scaled = scaled, normalized = ifelse(is.numeric(normalized),
    round(normalized, 1), unique(normalized)), axis.symbols = axis.symbols),
  labels = SI_pl_format(exponent = -unit.exponent),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["s.e.response"]],
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  ...
)

scale_y_s.q.response_continuous(
  unit.exponent = 0,
  name = s.q.response_label(unit.exponent = unit.exponent, format = format, label.text =
    label.text, scaled = scaled, normalized = ifelse(is.numeric(normalized),
    round(normalized, 1), unique(normalized)), axis.symbols = axis.symbols),
  labels = SI_pl_format(exponent = -unit.exponent),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["s.q.response"]],
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  ...
)

scale_y_s.e.action_continuous(
  unit.exponent = 0,
  name = s.e.action_label(unit.exponent = unit.exponent, format = format, label.text =
    label.text, scaled = scaled, normalized = ifelse(is.numeric(normalized),
    round(normalized, 1), unique(normalized)), axis.symbols = axis.symbols),
  labels = SI_pl_format(exponent = -unit.exponent),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["s.e.action"]],
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  ...
)

scale_y_s.q.action_continuous(
```

```
  unit.exponent = 0,
  name = s.q.action_label(unit.exponent = unit.exponent, format = format, label.text =
    label.text, scaled = scaled, normalized = ifelse(is.numeric(normalized),
    round(normalized, 1), unique(normalized)), axis.symbols = axis.symbols),
  labels = SI_pl_format(exponent = -unit.exponent),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["s.q.action"]],
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  ...
)
```

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.
`format`	character string, "R", "R.expression", "R.character", or "LaTeX".
`label.text`	character Textual portion of the labels.
`scaled`	logical If `TRUE` relative units are assumed.
`normalized`	logical (`FALSE`) or numeric Normalization wavelength in manometers (nm).
`axis.symbols`	logical If `TRUE` symbols of the quantities are added to the `name`. Supported only by `format = "R.expression"`.
`...`	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.action_continuous() + #  per joule
  scale_x_wl_continuous()

ggplot(ccd.spct) +
  geom_line() +
  scale_y_s.e.response_continuous() + #  per joule
  scale_x_wl_continuous()

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

ggplot(ccd.spct, unit.out = "photon") +
```

```
  geom_line() +
  scale_y_s.q.response_continuous() + # per mol
  scale_x_wl_continuous()

ggplot(ccd.spct, unit.out = "photon") +
  geom_line() +
  scale_y_s.q.response_continuous(unit.exponent = 3) + # per 1000 moles
  scale_x_wl_continuous()

norm_ccd.spct <- normalize(ccd.spct, norm = "max")
ggplot(norm_ccd.spct) +
  geom_line() +
  scale_y_s.e.response_continuous(normalized = getNormalized(norm_ccd.spct)) +
  scale_x_wl_continuous()

if (packageVersion("photobiology") > "0.11.4") {
  ggplot(norm_ccd.spct) +
    geom_line() +
    scale_y_s.e.response_continuous(normalized =
      normalization(norm_ccd.spct)$norm.type) +
    scale_x_wl_continuous()
}
photon_as_default()

norm_ccd.spct <- normalize(ccd.spct, norm = "max")
ggplot(norm_ccd.spct) +
  geom_line() +
  scale_y_s.q.response_continuous(normalized = getNormalized(norm_ccd.spct)) +
  scale_x_wl_continuous()

ggplot(norm_ccd.spct) +
  geom_line() +
  scale_y_s.q.response_continuous(unit.exponent = 2,
                                  normalized = getNormalized(norm_ccd.spct)) +
  scale_x_wl_continuous()

unset_radiation_unit_default()
```

`scale_y_Tfr_continuous` *Transmittance y-scale*

Description

Scale y continuous with defaults suitable for spectral transmittance.

Usage

```
scale_y_Tfr_continuous(
```

```
  unit.exponent = ifelse(pc.out, -2, 0),
  name = Tfr_label(unit.exponent = unit.exponent, format = format, label.text =
    label.text, scaled = scaled, normalized = ifelse(is.numeric(normalized),
   round(normalized, 1), unique(normalized)), axis.symbols = axis.symbols, Tfr.type =
    Tfr.type),
  labels = SI_pl_format(exponent = unit.exponent),
  limits = c(0, 1),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = NULL,
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  pc.out = getOption("ggspectra.pc.out", default = FALSE),
  Tfr.type,
  ...
)

scale_y_Tfr_internal_continuous(
  unit.exponent = ifelse(pc.out, -2, 0),
  name = Tfr_label(unit.exponent = unit.exponent, format = format, label.text =
    label.text, scaled = scaled, normalized = ifelse(is.numeric(normalized),
   round(normalized, 1), unique(normalized)), axis.symbols = axis.symbols, Tfr.type =
    "internal"),
  labels = SI_pl_format(exponent = unit.exponent),
  limits = c(0, 1),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = NULL,
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  pc.out = getOption("ggspectra.pc.out", default = FALSE),
  ...
)

scale_y_Tfr_total_continuous(
  unit.exponent = ifelse(pc.out, -2, 0),
  name = Tfr_label(unit.exponent = unit.exponent, format = format, label.text =
    label.text, scaled = scaled, normalized = ifelse(is.numeric(normalized),
   round(normalized, 1), unique(normalized)), axis.symbols = axis.symbols, Tfr.type =
    "total"),
  labels = SI_pl_format(exponent = unit.exponent),
  limits = c(0, 1),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = NULL,
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  pc.out = getOption("ggspectra.pc.out", default = FALSE),
```

```
  ...
)
```

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.
`limits`	One of `NULL` for default based on data range, a numeric vector of length two (`NA` allowed) or a function that accepts the data-based limits as argument and returns new limits.
`format`	character string, "R", "R.expression", "R.character", or "LaTeX".
`label.text`	character Textual portion of the labels.
`scaled`	logical If `TRUE` relative units are assumed.
`normalized`	logical (`FALSE`) or numeric Normalization wavelength in manometers (nm).
`axis.symbols`	logical If `TRUE` symbols of the quantities are added to the `name`. Supported only by `format = "R.expression"`.
`pc.out`	logical, if TRUE use percent as default instead of fraction of one.
`Tfr.type`	character, either "total" or "internal".
`...`	other named arguments passed to `scale_y_continuous`

Note

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

Examples

```
Tfr_as_default()

ggplot(yellow_gel.spct) +
  geom_line() +
  scale_y_Tfr_continuous(Tfr.type = getTfrType(yellow_gel.spct)) +
  scale_x_wl_continuous()

ggplot(yellow_gel.spct) +
  geom_line() +
  scale_y_Tfr_continuous(unit.exponent = -2,
                         Tfr.type = getTfrType(yellow_gel.spct)) +
  scale_x_wl_continuous()

ggplot(yellow_gel.spct) +
  geom_line() +
  scale_y_Tfr_continuous(unit.exponent = -3,
                         Tfr.type = getTfrType(yellow_gel.spct)) +
  scale_x_wl_continuous()

ggplot(yellow_gel.spct) +
```

```
  geom_line() +
  scale_y_Tfr_total_continuous() +
  scale_x_wl_continuous()

ggplot(yellow_gel.spct) +
  geom_line() +
  scale_y_Tfr_total_continuous(axis.symbols = FALSE) +
  scale_x_wl_continuous(axis.symbols = FALSE)

ggplot(normalize(yellow_gel.spct)) +
  geom_line() +
  scale_y_Tfr_total_continuous(normalized = "max") +
  scale_x_wl_continuous()

unset_filter_qty_default()
```

`sec_axis_w_number` *Secondary axes for wavelengths*

Description

Secondary axes for wavelength data in nanometres. With suitable scaling and name (axis label) for frequency, wave number, photon energy and wavelength.

Usage

```
sec_axis_w_number(
  unit.exponent = -6,
 label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["w.number"]],
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE)
)

sec_axis_w_frequency(
  unit.exponent = 12,
  label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["freq"]],
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE)
)

sec_axis_energy_eV(
  unit.exponent = 0,
  label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["energy"]],
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE)
)

sec_axis_energy_J(
  unit.exponent = -18,
  label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["energy"]],
```

```
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE)
)

sec_axis_wl(
  unit.exponent = -9,
  label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["w.length"]],
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE)
)
```

Arguments

`unit.exponent`	integer The exponent on base 10 of the scale multiplier used for the axis labels, e.g., 3 for 10^3 or k.
`label.text`	character Textual portion of the labels.
`axis.symbols`	logical If `TRUE` symbols of the quantities are added to the `name`. Supported only by `format = "R.expression"`.

Details

These secondary axis functions can be used only when the *x* aesthetic is mapped to a numerical variable containing wavelength values expressed in nanometres. They can be used to add a secondary x axis to plots created using `ggplot()` or `autoplot()`.

See Also

the default text used for quantity names are most easily changed by resetting all the defaults once as explained in `axis_labels_uk`, even if it is possible to override them also in each call.

Examples

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

# Secondary axes can be added to plots built with autoplot() methods
autoplot(sun.spct) +
  scale_x_wl_continuous(sec.axis = sec_axis_w_number())

# Using 'ggplot2' scale
ggplot(sun.spct) +
  geom_line() +
  scale_x_continuous(name = w_length_label(),
                     sec.axis = sec_axis_w_number())

# change scale multipliers, SI defined
ggplot(sun.spct) +
  geom_line() +
  scale_x_wl_continuous(-6, sec.axis = sec_axis_w_number(-3))

# change scale multipliers, not SI defined (best avoided)
```

```
ggplot(sun.spct) +
  geom_line() +
  scale_x_wl_continuous(-8, sec.axis = sec_axis_w_number(-4))

# Change quantity name to Spanish
ggplot(sun.spct) +
  geom_line() +
  scale_x_wl_continuous(label.text = "Longitud de onda,",
                        sec.axis = sec_axis_w_frequency(label.text = "Frecuencia,"))

# Frequency in secondary axis
ggplot(sun.spct) +
  geom_line() +
  scale_x_wl_continuous(sec.axis = sec_axis_w_frequency())

# Energy (per photon) in atto joules
ggplot(sun.spct) +
  geom_line() +
  scale_x_wl_continuous(sec.axis = sec_axis_energy_J())

# Energy (per photon) in electron volts
ggplot(sun.spct) +
  geom_line() +
  scale_x_wl_continuous(sec.axis = sec_axis_energy_eV())

# Secondary axis with wavelength using a different scale factor
ggplot(sun.spct) +
  geom_line() +
  scale_x_wl_continuous(sec.axis = sec_axis_wl(-6))

# Secondary axes can be added to plots built with autoplot() methods
autoplot(sun.spct) +
  scale_x_wl_continuous(sec.axis = sec_axis_wl(-6))
```

`set_annotations_default`

Set defaults for plotting

Description

Set R options controlling default arguments for some formal parameters in methods and functions from package 'ggspectra'.

Usage

```
set_annotations_default(annotations = NULL)

set_w.band_default(w.band = NULL)
```

```
set_markup_format_default(markup.format = NULL)

set_plot_range_default(range = NULL)

set_pc_out_default(pc.out = TRUE)

set_axis_symbols_default(axis.symbols = TRUE)
```

Arguments

`annotations`	a character vector. For details please see sections Plot **Annotations** and **Title Annotations**.
`w.band`	a single waveband object or a list of waveband objects.
`markup.format`	character string, "R", "R.expression", "r.character", or "LaTeX".
`range`	an R object on which `range()` returns a vector of length 2, with minimum and maximum wavelengths (nm).
`pc.out`	logical, if `TRUE` use percent instead of fraction of one for normalized spectral data.
`axis.symbols`	logical If `TRUE` symbols of the quantities are added to the default `name`.

Details

The values accepted, syntax used and behaviour are the same as when passing arguments to formal parameters in function and methods calls, except that `NULL` as argument clears the R option. To restore the previous state of an option, save the value returned and pass it as argument in a later call to the same function.

Changing the defaults with options, instead of affecting a single function call (e.g., affecting a single plot or layer in a plot), changes the default used for all subsequent function calls calls when when no argument is passed explicitly. This makes it possible to easily change in one place in a script the appearance of all/multiple plots. Using these functions functions instead of `options` to set the defaults adds a validation step that protects from errors triggered in subsequent function calls.

R option `photobiology.plot.annotations` controls the default for formal parameter `annotations` in `autoplot` methods and in function `decoration`.

R option `photobiology.plot.bands` controls the default for formal parameter `w.band` in `autoplot` methods and in function `decoration`.

R option `photobiology.math` controls the default for formal parameter `markup.format` or `format` in different *axis label* and *scale* functions.

R option `ggspectra.wlrange` controls the default for formal parameter `range` in `autoplot` methods.

R option `ggspectra.pc.out` controls the default for formal parameter `pc.out` in `autoplot` methods that have this formal parameter.

R option `ggspectra.axis.symbols` controls the default for formal parameter `axis.symbols` in different *axis label* and *scale* functions.

Value

Previous value of the option, returned invisibly. This is a named list of length one as returned by `options`, that can be passed unchanged as argument to R function `options` or in a new call to the same function that returned it.

Plot Annotations

The recognized annotation names are: `"summaries"`, `"peaks"`, `"peak.labels"`, `"valleys"`, `"valley.labels"`, `"wls"`, `"wls.labels"`, `"colour.guide"`, `"color.guide"`, `"boxes"`, `"segments"`, `"labels"`. In addition, `"+"` is interpreted as a request to add to the already present default annotations, `"-"` as request to remove annotations and `"="` or missing`"+"` and `"-"` as a request to reset annotations to those requested. If used, `"+"`, `"-"` or `"="` must be the first member of a character vector, and followed by one or more of the names given above. To simultaneously add and remove annotations one can pass a `list` containing `character` vectors each assembled as described. The vectors are applied in the order they appear in the list. To disable all annotations pass `""` or `c("=", "")` as argument. Adding a variation of an annotation already present, replaces the existing one automatically: e.g., adding `"peak.labels"` replaces`"peaks"` if present.

The annotation layers are added to the plot using statistics defined in 'ggspectra': `stat_peaks`, `stat_valleys`, `stat_label_peaks`, `stat_label_valleys`, `stat_find_wls`, `stat_spikes`, `stat_wb_total`, `stat_wb_mean`, `stat_wb_irrad`, `stat_wb_sirrad`, `stat_wb_contribution`, `stat_wb_relative`, and `stat_wl_strip`. However, only some of their parameters can be passed arguments through `autoplot` methods. In some cases the defaults used by `autoplot` methods are not the defaults of the statistics.

Title Annotations

metadata retrieved from object `object` is paased 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 metatdata 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. Default is `"title:objt"` or no title depending on the context.

See Also

Additional argument defaults are controlled by options also used in package 'photobiology'. See `energy_as_default`, `using_Tfr` and `options`.

`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

```
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

```
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)
```

`stat_color` *Calculate colours from wavelength.*

Description

`stat_color` computes colour definitions according to human vision.

Usage

```
stat_color(
  mapping = NULL,
  data = NULL,
  geom = "point",
  position = "identity",
  ...,
  chroma.type = "CMF",
  x.colour.transform = function(x) {
     x
 },
  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.
`chroma.type`	character one of "CMF" (color matching function) or "CC" (color coordinates) or a `chroma_spct` object.
`x.colour.transform`	function Applied to `x` values before computing matching colours.
`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

For each row in `data` a colour definition is computed assuming that after transformation with `x.colour.transform()` the values in `x` are wavelengths expressed in nanometres.

Value

The original data frame with variable `wl.color` containing colour definitions added.

Computed variable

wl.color color corresponding to x-value giving wavelength in nanometres.

Default aesthetics

Set by the statistic and available to geoms.

color after_stat(wl.color)

fill after_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

See Also

`color_of`, which is used internally.

Other stats functions: `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()`, `stat_wl_summary()`

Examples

```
ggplot(sun.spct) +
  geom_line() +
  stat_color() +
  scale_color_identity()

ggplot(sun.spct) +
  geom_line() +
  stat_color(x.colour.transform = function(x) {-x}) +
  scale_color_identity() +
  scale_x_reverse()

ggplot(sun.spct) +
  geom_line() +
  stat_color(x.colour.transform = function(x) {10^x}) +
  scale_color_identity() +
  scale_x_log10()
```

`stat_find_qtys` *Find quantity value for target wavelength value.*

Description

`stat_find_qtys` finds at which y positions values equal to an x target are located. **Axis flipping is currently not supported.**

Usage

```
stat_find_qtys(
  mapping = NULL,
  data = NULL,
  geom = "point",
  position = "identity",
  ...,
  target = "half.maximum",
  interpolate = TRUE,
  chroma.type = "CMF",
  label.fmt = "%.3g",
  x.label.fmt = label.fmt,
  y.label.fmt = label.fmt,
  x.label.transform = function(x) {
     x
 },
  y.label.transform = function(x) {
     x
 },
  x.colour.transform = x.label.transform,
```

```
  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.
`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`, `x.label.fmt`, `y.label.fmt`	character strings giving a format definition for construction of character strings labels with function `sprintf` from `x` and/or `y` values.
`x.label.transform`, `y.label.transform`, `x.colour.transform`	function Applied to `x` or `y` values when constructing the character labels or computing matching colours.
`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 match to the target subset from the data or linearly interpolated between the two nearest values available. As spectra are monotonic in wavelength, this statistic will never return more than one row in `data` per target value.

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_qtys(target = "half.range")

ggplot(yellow_gel.spct) +
  geom_line() +
  stat_find_qtys(target = c(490, 500, 510))

ggplot(yellow_gel.spct) +
  geom_line() +
  stat_find_qtys(target = 500, geom = "point", colour = "red") +
  stat_find_qtys(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. **Axis flipping is currently not supported.**

Usage

```
stat_find_wls(
  mapping = NULL,
  data = NULL,
  geom = "point",
  position = "identity",
  ...,
  target = "half.maximum",
  interpolate = TRUE,
  chroma.type = "CMF",
  label.fmt = "%.3g",
  x.label.fmt = label.fmt,
  y.label.fmt = label.fmt,
  x.label.transform = function(x) {
     x
 },
  y.label.transform = function(x) {
     x
 },
  x.colour.transform = x.label.transform,
  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.
`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 stradling the target.
`chroma.type`	character one of "CMF" (color matching function) or "CC" (color coordinates) or a `chroma_spct` object.
`label.fmt`, `x.label.fmt`, `y.label.fmt`	character strings giving a format definition for construction of character strings labels with function `sprintf` from `x` and/or `y` values.
`x.label.transform`, `y.label.transform`, `x.colour.transform`	function Applied to `x` or `y` values when constructing the character labels or computing matching colours.
`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

For each row in the subset of `data` matching `target` a colour definition is computed assuming that after transformation with `x.colour.transform()` the values in `x` are wavelengths expressed in nanometres. Labels are constructed from `x` and `y` values after applying to them `x.label.transform` and `y.label.transform`, respectively. In most cases the `x.label.transform` is used to back-transform the values in `data` to make them agree with those displayed on the axis guides.

These stats use `geom_point` by default as it is a geometry likely to work well in almost any situation. The additional default aesthetic mappings set by these statistics allow their direct use with `geom_text`, `geom_label`, `geom_line`, `geom_rug`, `geom_hline` and `geom_vline`. The format 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 linearly interpolated between the two nearest values available. As spectra are not monotonic in the spectral quantity, this statistic can return more than one row in `data` per target value.

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 nanometres.

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

```
# 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 the global maximum or local maxima are located, and adds labels and color definitions to the data without subsetting. `stat_labels_valleys` finds instead minima. The variable mapped to the `x` aesthetic is expected to contain wavelength values expressed in nanometres. **Axis flipping is currently not supported.**

Usage

```
stat_label_peaks(
  mapping = NULL,
  data = NULL,
  geom = "text",
  position = "identity",
  ...,
  span = 5,
  ignore_threshold = 0,
  global.threshold = ignore_threshold,
  strict = TRUE,
  chroma.type = "CMF",
  label.fmt = "%.3g",
  x.label.fmt = label.fmt,
  y.label.fmt = label.fmt,
  x.label.transform = function(x) {
     x
 },
  y.label.transform = function(x) {
     x
 },
  x.colour.transform = x.label.transform,
  label.fill = "",
  na.rm = TRUE,
```

```
  show.legend = FALSE,
  inherit.aes = TRUE
)

stat_label_valleys(
  mapping = NULL,
  data = NULL,
  geom = "text",
  position = "identity",
  ...,
  span = 5,
  ignore_threshold = 0,
  global.threshold = ignore_threshold,
  strict = TRUE,
  chroma.type = "CMF",
  label.fmt = "%.3g",
  x.label.fmt = label.fmt,
  y.label.fmt = label.fmt,
  x.label.transform = function(x) {
     x
 },
  y.label.transform = function(x) {
     x
 },
  x.colour.transform = x.label.transform,
  label.fill = "",
  na.rm = TRUE,
  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.
`span`	odd positive integer A peak is defined as an element in a sequence which is greater than all other elements within a moving window of width span centred at that element. The default value is 5, meaning that a peak is taller than its four nearest neighbours. `span = NULL` extends the span to the whole length of `x`.
`ignore_threshold`	Deprecated synonym for `global.threshold`.

`global.threshold`	numeric A value belonging to class `"AsIs"` is interpreted as an absolute minimum height or depth expressed in data units. A bare `numeric` value (normally between 0.0 and 1.0), is interpreted as relative to the range of the data. In both cases it sets a *global* height (depth) threshold below which peaks (valleys) are ignored. A bare negative `numeric` value indicates the *global* height (depth) threshold below which peaks (valleys) are be ignored. If `global.threshold = NULL`, no threshold is applied and all peaks are returned.
`strict`	logical flag: if `TRUE`, an element must be strictly greater than all other values in its window to be considered a peak.
`chroma.type`	character one of "CMF" (color matching function) or "CC" (color coordinates) or a `chroma_spct` object.
`label.fmt`, `x.label.fmt`, `y.label.fmt`	character strings giving a format definition for construction of character strings labels with function `sprintf` from `x` and/or `y` values.
`x.label.transform`, `y.label.transform`, `x.colour.transform`	function Applied to `x` or `y` values when constructing the character labels or computing matching colours.
`label.fill`	character string to use for labels not at peaks or valleys being highlighted.
`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 statistics assemble text labels for each peak or valley and compute the colour corresponding to the wavelength of the peaks and valleys. Defaults work as long as the variable mapped to the `x` aesthetic contains wavelengths expressed in nanometres and the plot has an x-scale that does not apply a transformation. The three `transform` parameters can be used to back-transform the values when scales apply transformations so that peak/valley labels and axis labels match. Of course, `x.label.transform` and `y.label.transform` make also possible to scale the values in the labels.

Both statistics 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 statistics 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 after_stat(x.label)

xintercept after_stat(x)

yintercept after_stat(y)

color black_or_white(after_stat(wl.color))

fill after_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

When using `geom_text`, to discard overlapping labels pass `check_overlap = TRUE` in the call to the statistic.

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, without discarding any of them. The difference between `stat_peaks` and `stat_label_peaks`, and between `stat_valleys` and `stat_label_valleys`, is that while the first only returns the rows in data matching peaks or valleys, the second return all rows, but set the labels to the value passed as argument to `label.fill`. In the "label" stats the default `label.fill = ""` ensures that when using repulsive geoms the labels do not overlap any observations, labelled or not.

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

`stat_peaks`, `stat_valleys` and `find_peaks`, which is used in the implementation.

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

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

ggplot(sun.spct) +
  geom_line() +
  stat_label_valleys(hjust = "right", span = 21, angle = 90, color = "blue")

# using transformed scales requires the user to pass functions as arguments
ggplot(sun.spct) +
  geom_line() +
  stat_label_peaks(hjust = "left", span = 31, angle = 90, color = "red",
                   x.label.transform = abs) +
  scale_x_reverse()

ggplot(sun.spct) +
  geom_line() +
  stat_label_peaks(hjust = "left", span = 31, angle = 90, color = "red",
                   x.label.transform = function(x) {10^x}) +
  scale_x_log10()

# geom_label
ggplot(sun.spct) +
  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)

ggplot(sun.spct) + geom_line() +
  stat_peaks(span = 41, shape = 21, size = 2) +
  stat_label_peaks(span = 41, geom = "label_repel", segment.colour = "red",
                   nudge_y = 0.12, label.fmt = "%3.0f nm",
                   max.overlaps = Inf, min.segment.length = 0) +
  scale_fill_identity() +
  scale_color_identity() +
  expand_limits(y = c(NA, 1))

## End(Not run)
```

`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. **Axis flipping is currently not supported.**

Usage

```
stat_peaks(
  mapping = NULL,
  data = NULL,
  geom = "point",
  position = "identity",
  ...,
  span = 5,
  ignore_threshold = 0.01,
  global.threshold = ignore_threshold,
  local.threshold = NULL,
  local.reference = "median",
  strict = FALSE,
  refine.wl = FALSE,
  method = "spline",
  chroma.type = "CMF",
  label.fmt = "%.3g",
  x.label.fmt = label.fmt,
  y.label.fmt = label.fmt,
  x.label.transform = function(x) {
     x
 },
  y.label.transform = function(x) {
     x
 },
  x.colour.transform = x.label.transform,
  na.rm = FALSE,
  show.legend = FALSE,
  inherit.aes = TRUE
)

stat_valleys(
  mapping = NULL,
  data = NULL,
  geom = "point",
  position = "identity",
  ...,
  span = 5,
  ignore_threshold = 0.01,
  global.threshold = ignore_threshold,
  local.threshold = NULL,
  local.reference = "median",
  strict = FALSE,
```

```
  refine.wl = FALSE,
  method = "spline",
  chroma.type = "CMF",
  label.fmt = "%.3g",
  x.label.fmt = label.fmt,
  y.label.fmt = label.fmt,
  x.label.transform = function(x) {
     x
 },
  y.label.transform = function(x) {
     x
 },
  x.colour.transform = x.label.transform,
  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.
`span`	odd positive integer A peak is defined as an element in a sequence which is greater than all other elements within a moving window of width `span` centred at that element. The default value is 5, meaning that a peak is taller than its four nearest neighbours. `span = NULL` extends the span to the whole length of `x`.
`ignore_threshold`	Deprecated synonym for `global.threshold`.
`global.threshold`	numeric A value belonging to class `"AsIs"` is interpreted as an absolute minimum height or depth expressed in data units. A bare `numeric` value (normally between 0.0 and 1.0), is interpreted as relative to the range of the data. In both cases it sets a *global* height (depth) threshold below which peaks (valleys) are ignored. A bare negative `numeric` value indicates the *global* height (depth) threshold below which peaks (valleys) are be ignored. If `global.threshold = NULL`, no threshold is applied and all peaks are returned.
`local.threshold`	numeric A value belonging to class `"AsIs"` is interpreted as an absolute minimum height (depth) expressed in data units relative to the within-window computed minimum (maximum) value. A bare `numeric` value (normally between 0.0 and 1.0), is interpreted as expressed in units relative to the range of the

	data. In both cases `local.threshold` sets a *local* height (depth) threshold below which peaks (valleys) are ignored. If `local.threshold = NULL` or if `span` spans the whole of `x`, no threshold is applied.
`local.reference`	character One of `"minimum"/maximum` or `"median"`. The reference used to assess the height of the peak, either the minimum value within the window or the median of all values in the window.
`strict`	logical flag: 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`, `x.label.fmt`, `y.label.fmt`	character strings giving a format definition for construction of character strings labels with function `sprintf` from `x` and/or `y` values.
`x.label.transform`, `y.label.transform`, `x.colour.transform`	function Applied to `x` or `y` values when constructing the character labels or computing matching colours.
`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.

Two tests make it possible to ignore irrelevant peaks or valleys. One test controlled by (`global.threshold`) is based on the absolute height/depth of peaks/valleys and can be used in all cases to ignore globally low peaks and shallow valleys. A second test controlled by (`local.threshold`) is available when the window defined by 'span' does not include all observations and can be used to ignore peaks/valleys that are not locally prominent. In this second approach the height/depth of each peak/valley is compared to a summary computed from other values within the window where it was found. In this second case, the reference value used is the summary indicated by `local.reference`. The values `global.threshold` and `local.threshold` if bare numeric are relative to the range of *y*. Thresholds for ignoring too small peaks are applied after peaks are searched for, and threshold values can in some cases result in no peaks being displayed.

Value

A data frame with one row for each peak (or valley) found in the data. If `refine.wl = FALSE`, the returned rows have `x` and `y` matching those in a row in the input `data`. If `refine.wl = TRUE`, interpolation based on a fitted spline is used to compute new `x` and `y` values.

Computed and copied variables in the returned data frame

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 `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

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`, which is used internally.

Other stats functions: `stat_color()`, `stat_find_qtys()`, `stat_find_wls()`, `stat_label_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.

# PEAKS

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

# threshold relative to data range [0..1]
ggplot(sun.spct) +
  geom_line() +
  stat_peaks(global.threshold = 0.6) # 0.6 * range of data

# threshold in data units
ggplot(sun.spct) +
  geom_line() +
  stat_peaks(global.threshold = I(0.4))

# threshold in data units
ggplot(sun.spct, unit.out = "photon") +
  geom_line() +
  stat_peaks(global.threshold = I(2e-6)) # Q in mol m-2 s-1

# VALLEYS

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

# discard multiple maxima or minima
ggplot(sun.spct) +
  geom_line() +
  stat_valleys(strict = TRUE)

# threshold relative to data range [0..1]
ggplot(sun.spct) +
  geom_line() +
  stat_valleys(global.threshold = 0.6)

# reverse threshold relative to data range [-1..0]
ggplot(sun.spct) +
  geom_line() +
  stat_valleys(global.threshold = -0.9)

# threshold in data units using I()
ggplot(sun.spct) +
  geom_line() +
  stat_valleys(global.threshold = I(0.6), strict = TRUE)

# USING OTHER COMPUTED VALUES
```

```
# colours matching the wavelength at peaks
ggplot(sun.spct) +
  geom_line() +
  stat_peaks(span = 51, size = 2.7,
             mapping = aes(colour = after_stat(wl.colour))) +
  scale_color_identity()

# labels for local maxima
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")

# labels for local fitted peaks
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)

# fitted peaks and valleys
ggplot(sun.spct) +
  geom_line() +
  stat_peaks(span = 31, geom = "point", colour = "red", refine.wl = TRUE) +
  stat_peaks(mapping = aes(fill = after_stat(wl.colour), color = after_stat(BW.colour)),
             span = 31, geom = "label",
             size = 3, vjust = -0.2, label.fmt = "%.4g nm",
             refine.wl = TRUE) +
  stat_valleys(span = 51, geom = "point", colour = "blue", refine.wl = TRUE) +
 stat_valleys(mapping = aes(fill = after_stat(wl.colour), color = after_stat(BW.colour)),
              span = 51, geom = "label",
              size = 3, vjust = 1.2, label.fmt = "%.4g nm",
              refine.wl = TRUE) +
  expand_limits(y = 0.85) + # make room for label
  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. **Axis flipping is currently not supported.**

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,
  x.label.transform = function(x) {
     x
 },
  y.label.transform = function(x) {
     x
 },
  x.colour.transform = x.label.transform,
  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`, `x.label.fmt`, `y.label.fmt`	character strings giving a format definition for construction of character strings labels with function `sprintf` from `x` and/or `y` values.
`x.label.transform`, `y.label.transform`, `x.colour.transform`	function Applied to `x` or `y` values when constructing the character labels or computing matching colours.

`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 allow 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 of observations found in the data matching the criterion of being part of a spike. That is to say, the returned data frame not only includes the observation at the tip of the spike but also those on its shoulders.

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 of observations in the spike formatted as character

y.label y-value of observations in the spike formatted as character

wl.color color definition calculated by assuming that x-values are wavelengths expressed in nanometres.

BW.color color definition that is 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` argument 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.spct) +
  geom_line() +
  stat_spikes(colour = "red", alpha = 0.3)

# no spikes detected
ggplot(sun.spct) +
  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 `geom_rect()`. **`x`-scale transformations and axis flipping are currently not supported**.

Usage

```
stat_wb_box(
  mapping = NULL,
  data = NULL,
  geom = "rect",
  position = "identity",
  ...,
  by.group = FALSE,
  w.band = NULL,
  chroma.type = "CMF",
  ypos.mult = 1.07,
  ypos.fixed = NULL,
  box.height = 0.06,
  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
`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.
`by.group`	logical flag If TRUE repeated identical layers are added for each group within a plot panel as needed for animation. If `FALSE`, the default, a single layer is added per panel.
`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 compute returned `y` values. This is numerically similar to using npc units, but values larger than one expand the plotting area.
`ypos.fixed`	numeric If not `NULL` used a constant value returned in `y`.
`box.height`	numeric The height of the box as a fraction of the range of \$y\$. This is similar to using npc units.
`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

By default `stat_wb_box()` uses a panel function and ignores grouping as needed for annotation of layers supporting free axis scales. Passing `by.group = TRUE` as argument changes this behaviour adding the same layer repeatedly for each group as needed for constructing animated plots with functions from package 'gganimate'.

The value returned as default value for `y` is based on the y-range of spectral values for the whole data set.

As colours are returned as RGB colour definitions, depending on the geometry used the use of `scale_fill_identity` and/or `scale_colour_identity` can be necessary for the correct colours to be displayed in the plot.

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

What it is named integral below is the result of appying `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

wb.ymax data\$y maximum

ymin box bottom

ymax box top

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 stat(wb.xmin)

xmax stat(wb.xmax)

ymin stat(ymin)

ymax stat(ymax)

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

As only one colour scale can exist within a `"gg"` object, using this scale prevents the mapping to the colour aesthetic of factors in `data` to create a grouping.

See Also

`fast_color_of_wb`, which is used in the implementation.

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

```
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 nanaometre of wavelength for each waveband in the input. Sets suitable default aesthetics for `geom_rect()`. **`x`-scale transformations and axis flipping are currently not supported**.

Usage

```
stat_wb_column(
  mapping = NULL,
  data = NULL,
  geom = "rect",
  position = "identity",
  ...,
  w.band = NULL,
  integral.fun = integrate_xy,
  chroma.type = "CMF",
  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
`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.
`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.
`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`.

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

What it is named integral below is the result of appying `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

wb.ymax data$y maximum

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

y wb.ymeam

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

```
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()
```

`stat_wb_contribution` *Integrate ranges under spectral curve.*

Description

`stat_wb_contribution` integrates the area under a spectral curve. It first integrates the area under the curve for each waveband and for the whole curve and then expresses the integral for each band as a relative contribution to the area under the whole spectral curve. Sets suitable default aesthetics for "rect", "hline", "vline", "text" and "label" geoms displaying "contributions" per waveband to the total of the spectral integral. **`x`-scale transformations and axis flipping are currently not supported**.

Usage

```
stat_wb_contribution(
  mapping = NULL,
  data = NULL,
  geom = "text",
  position = "identity",
  ...,
  w.band = NULL,
  integral.fun = integrate_xy,
  label.mult = 1,
  chroma.type = "CMF",
  label.fmt = "%1.2f",
  ypos.mult = 1.07,
  ypos.fixed = NULL,
```

```
  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
`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.
`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`.
`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`.

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

What it is named integral below is the result of appying `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

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

# Using defaults
ggplot(sun.spct) +
  geom_line() +
  stat_wb_box(w.band = VIS()) +
```

```
  stat_wb_contribution(w.band = VIS()) +
  scale_fill_identity() + scale_color_identity()

# Setting position and angle of the text
ggplot(sun.spct) +
  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()

# Showing percentages, i.e., using a different format for numbers
ggplot(sun.spct) +
  geom_line() +
  stat_wb_box(w.band = VIS_bands()) +
  stat_wb_contribution(w.band = VIS_bands(), size = 2.5,
                       label.mult = 100, label.fmt = "%3.0f%%") +
  scale_fill_identity() + scale_color_identity()

# Including the name of the waveband, i.e., changing the mapping for label
ggplot(sun.spct, range = c(NA, 410)) +
  geom_line() +
  stat_wb_box(w.band = UV_bands(), color = "white") +
  stat_wb_contribution(w.band = UV_bands(), size = 2.5,
                       label.mult = 100, label.fmt = "%3.0f%%",
                       mapping = aes(label = after_stat(paste(wb.name, y.label)))) +
  scale_fill_identity() + scale_color_identity()
```

`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'. **`x`-scale transformations and axis flipping are currently not supported**.

Usage

```
stat_wb_hbar(
  mapping = NULL,
  data = NULL,
  geom = "linerange",
  position = "identity",
  ...,
  w.band = NULL,
  integral.fun = integrate_xy,
  chroma.type = "CMF",
```

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

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

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

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 the range of `w.band`

ymean yint divided by wl_expanse(w.band)

y ypos.fixed or mean of data

wb.color color of the w.band

wb.name label of w.band

Default aesthetics

Set by the statistic and available to geoms.

xmin ..xmin..

xmax ..xmax..

yintercept ..ymean..

height (..ymax.. - ..ymin..) * 2e-2

color ..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 any weighting is returned as default value for `y`.

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_irrad()`, `stat_wb_label()`, `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_hbar(w.band = VIS_bands(), size = 1) +
  scale_color_identity() +
  theme_bw()

ggplot(sun.spct) +
  geom_line() +
  stat_wb_hbar(w.band = PAR(), size = 1) +
  scale_color_identity() +
  theme_bw()
```

```
ggplot(sun.spct) +
  geom_line() +
  stat_wb_hbar(w.band = PAR(), size = 1, ypos.fixed = 0) +
  scale_color_identity() +
  theme_bw()

ggplot(sun.spct) +
  geom_line() +
  stat_wb_hbar(w.band = CIE(), size = 1) +
  scale_color_identity() +
  theme_bw()
```

`stat_wb_irrad` *Integrate irradiance for wavebands.*

Description

`stat_wb_irrad` integrates the area under a spectral irradiance curve, yielding energy or photon irradiance. The range(s) of wavelengths to integrate are set with a list of waveband objects. `x`**-scale transformations and axis flipping are currently not supported**.

Usage

```
stat_wb_irrad(
  mapping = NULL,
  data = NULL,
  geom = "text",
  position = "identity",
  ...,
  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,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_wb_e_irrad(
  mapping = NULL,
  data = NULL,
```

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

  stat_wb_q_irrad(
    mapping = NULL,
    data = NULL,
    geom = "text",
    position = "identity",
    ...,
    w.band = NULL,
    time.unit = "second",
    unit.in = "photon",
    label.qty = "total",
    label.mult = 1,
    chroma.type = "CMF",
    label.fmt = "%.3g",
    ypos.mult = 1.07,
    ypos.fixed = NULL,
    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
`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.
`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`.
`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`.

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

What it is named integral below is the result of appying `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 ..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

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

# using defaults for energy irradiance in W m-2
ggplot(sun.spct) +
  stat_wb_column(w.band = PAR(), alpha = 0.5) +
  stat_wb_e_irrad(w.band = PAR(), ypos.fixed = 0.32) +
  geom_line() +
  scale_fill_identity() + scale_color_identity()

# using defaults for photon irradiance in umol m-2 s-1
ggplot(sun.spct, unit.out = "photon") +
  stat_wb_column(w.band = PAR(), alpha = 0.5) +
  stat_wb_q_irrad(w.band = PAR(), ypos.fixed = 1.5e-6, label.mult = 1e6) +
  geom_line() +
  scale_fill_identity() + scale_color_identity()

# modify label format and position
ggplot(sun.spct) +
  stat_wb_column(w.band = VIS_bands(), alpha = 0.7) +
  stat_wb_e_irrad(w.band = VIS_bands(),
```

```
                angle = 90, size = 3, hjust = "left",
                label.fmt = "%2.0f~~W~m^{-2}", parse = TRUE,
                ypos.fixed = 0.1) +
  geom_line() +
  scale_fill_identity() + scale_color_identity()

# Changing label mapping
ggplot(sun.spct) +
  stat_wb_column(w.band = VIS_bands(), alpha = 0.5) +
  stat_wb_e_irrad(w.band = VIS_bands(),
             label.fmt = "%.2f",
             angle = 90, color = "black", ypos.fixed = 0.1,
             hjust = "left", size = 3,
             mapping = aes(label = after_stat(paste(wb.name, ": ",
                                                  signif(wb.yint, 3),
                                                  sep = "")))) +
  geom_line() +
  scale_fill_identity() + scale_color_identity() +
  theme_bw()
```

`stat_wb_label` *Label ranges under spectral curve.*

Description

`stat_wb_label` computes the center of a waveband. Sets suitable default aesthetics for "text" and "label" geoms displaying "boundaries" and "names" of wavebands. **`x`-scale transformations and axis flipping are currently not supported**.

Usage

```
stat_wb_label(
  mapping = NULL,
  data = NULL,
  geom = "text",
  position = "identity",
  ...,
  by.group = FALSE,
  w.band = NULL,
  chroma.type = "CMF",
  label.fmt = "%s",
  ypos.fixed = 0,
  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
`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.
`by.group`	logical flag If TRUE repeated identical layers are added for each group within a plot panel as needed for animation. If `FALSE`, the default, a single layer is added per panel.
`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`.
`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

By default `stat_wb_label()` uses a panel function and ignores grouping as needed for annotation of layers supporting free axis scales. Passing `by.group = TRUE` as argument changes this behaviour adding the same layer repeatedly for each group as needed for constructing animated plots with functions from package 'gganimate'.

As colours are returned as RGB colour definitions, depending on the geometry used the use of `scale_fill_identity` and/or `scale_colour_identity` will be necessary for the correct colours to be displayed in the plot.

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

As only one colour scale can exist within a "gg" object, using this scale prevents the mapping to the colour aesthetic of factors in `data` to create a grouping.

See Also

`fast_color_of_wb`, which is used in the implementation.

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")
```

```
ggplot(sun.spct) +
  geom_line() +
  stat_wb_hbar(w.band = PAR(), ypos.fixed = 0, linewidth = 1) +
  stat_wb_label(aes(color = after_stat(wb.color)),
                w.band = PAR(), ypos.fixed = +0.025) +
  scale_color_identity()
```

`stat_wb_mean` *Integrate ranges under curve.*

Description

`stat_wb_mean` computes mean spectral irradiance under a curve for each waveband in the input. Sets suitable default aesthetics for "rect", "hline", "vline", "text" and "label" geoms. **`x`-scale transformations and axis flipping are currently not supported**.

Usage

```
stat_wb_mean(
  mapping = NULL,
  data = NULL,
  geom = "text",
  position = "identity",
  ...,
  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,
  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
`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.
`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`.
`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`.

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

What it is named integral below is the result of appying `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 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.

# Using defaults
ggplot(sun.spct) +
  stat_wb_column(w.band = VIS_bands()) +
  stat_wb_mean(w.band = VIS_bands(),
               color = "black") +
  scale_fill_identity() + scale_color_identity()

# Setting format for numbers, position, angle, and color
ggplot(sun.spct) +
  stat_wb_column(w.band = VIS_bands(), alpha = 0.5) +
  stat_wb_mean(w.band = VIS_bands(),
               label.fmt = "%.2f",
               angle = 90, color = "black", ypos.fixed = 0.1) +
  geom_line() +
  scale_fill_identity() + scale_color_identity() +
  theme_bw()

# Changing label mapping
```

```
ggplot(sun.spct) +
  stat_wb_column(w.band = VIS_bands(), alpha = 0.5) +
  stat_wb_mean(w.band = VIS_bands(),
               label.fmt = "%.2f",
               angle = 90, color = "black", ypos.fixed = 0.1,
               hjust = "left", size = 3,
            mapping = aes(label = after_stat(paste(wb.name, ": ", y.label, sep = "")))) +
  geom_line() +
  scale_fill_identity() + scale_color_identity() +
  theme_bw()

# example using repulsion
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 = +0.04, size = 3,
               segment.colour = NA, label.size = NA) +
  expand_limits(y = 0.9) +
  scale_fill_identity() + scale_color_identity() +
  theme_bw()
```

`stat_wb_relative` *Integrate ranges under spectral curve.*

Description

`stat_wb_relative` computes relative-irradiances under a curve. It first integrates the area under the spectral curve for each waveband in the input, and expresses these irradianses relative to their sum. Sets suitable default aesthetics for "rect", "hline", "vline", "text" and "label" geoms. **`x`-scale transformations and axis flipping are currently not supported**.

Usage

```
stat_wb_relative(
  mapping = NULL,
  data = NULL,
  geom = "text",
  position = "identity",
  ...,
  w.band = NULL,
  integral.fun = integrate_xy,
  label.mult = 1,
  chroma.type = "CMF",
  label.fmt = "%1.2f",
  ypos.mult = 1.07,
  ypos.fixed = NULL,
```

```
    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
`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.
`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`.
`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`.

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

What it is named integral below is the result of appying `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$y integral for each mebmer of w.band / sum of data$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

```
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 the mean spectral irradiance under a curve, yielding energy or photon spectral irradiance. The range(s) of wavelengths to integrate are set with a list of `waveband` objects. **`x`-scale transformations and axis flipping are currently not supported**.

Usage

```
stat_wb_sirrad(
  mapping = NULL,
  data = NULL,
  geom = "text",
  position = "identity",
  ...,
  w.band = NULL,
  time.unit,
  unit.in,
  label.qty = "mean",
  label.mult = 1,
  chroma.type = "CMF",
  label.fmt = "%.3g",
  ypos.mult = 0.55,
```

```
  xpos.fixed = NULL,
  ypos.fixed = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_wb_e_sirrad(
  mapping = NULL,
  data = NULL,
  geom = "text",
  position = "identity",
  ...,
  w.band = NULL,
  time.unit = "second",
  unit.in = "energy",
  label.qty = "mean",
  label.mult = 1,
  chroma.type = "CMF",
  label.fmt = "%.3g",
  ypos.mult = 0.55,
  xpos.fixed = NULL,
  ypos.fixed = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_wb_q_sirrad(
  mapping = NULL,
  data = NULL,
  geom = "text",
  position = "identity",
  ...,
  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,
  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
`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.
`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.
`xpos.fixed`, `ypos.fixed`	numeric If not `NULL` used a constant value returned in `x` or `y`.
`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`.

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

What it is named integral below is the result of appying `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

```
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. Sets suitable default aesthetics for "rect", "hline", "vline", "text" and "label" geoms displaying "totals" per waveband. `x`**-scale transformations and axis flipping are currently not supported**.

Usage

```
stat_wb_total(
  mapping = NULL,
  data = NULL,
  geom = "text",
  position = "identity",
  ...,
  w.band = NULL,
  integral.fun = integrate_xy,
  label.mult = 1,
  chroma.type = "CMF",
  label.fmt = "%.3g",
  ypos.mult = 1.07,
  ypos.fixed = NULL,
  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
`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.
`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`.
`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`.

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

What it is named integral below is the result of appying `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

```
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 and by default plots a narrow, guide-like colour gradient strip based on wavelength. **`x`-scale transformations and axis flipping are currently not supported**.

Usage

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

wl_guide(
  mapping = NULL,
  data = NULL,
  position = "identity",
  ...,
  by.group = FALSE,
  chroma.type = "CMF",
  w.band = NULL,
  length.out = 150,
  ymin = -Inf,
  ymax = Inf,
  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.
`by.group`	logical flag If TRUE repeated identical layers are added for each group within a plot panel as needed for animation. If `FALSE`, the default, a single layer is added per panel.
`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 `chroma_spct` object.
`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`.
`ymin`, `ymax`	numeric used as aesthetics for plotting the guide.

Details

By default `stat_wl_strip()` uses a panel function and ignores grouping as needed for annotation of layers supporting free axis scales. Passing `by.group = TRUE` as argument changes this behaviour adding the same layer repeatedly for each group as needed for constructing animated plots with functions from package 'gganimate'.

Function `wl_guide()` is a conveneince wrapper on `stat_wl_strip()` that also adds the required `scale_fill_identity()`.

As colours are returned as RGB colour definitions, depending on the geometry used the use of `scale_fill_identity` and/or `scale_colour_identity` will be necessary for the correct colours to be displayed in the plot.

Value

generic_spect object with new `x` values plus other computed variables described below.

Computed variables

x (w.low + wl.high) / 2

wl.low boundary of waveband

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

As only one colour scale can exist within a `"gg"` object, using this scale prevents the mapping to the colour aesthetic of factors in `data` to create a grouping.

See Also

`color_of` and `fast_color_of_wl`, which are used in the implementation.

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

```
# 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",
  position = "identity",
  ...,
  range = NULL,
  integral.fun = integrate_xy,
  label.fmt = "%.3g",
  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
`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.
`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 `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`.

Value

A data frame with one row.

Computed variables

What it is named integral below is the result of appying `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

```
# 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_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

```
Tfr_label(
  unit.exponent = ifelse(pc.out, -2, 0),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = NULL,
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE),
  pc.out = getOption("ggspectra.pc.out", default = FALSE),
  Tfr.type
)

Tfr_internal_label(
  unit.exponent = 0,
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = NULL,
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE)
)

Tfr_total_label(
  unit.exponent = 0,
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = NULL,
  scaled = FALSE,
  normalized = FALSE,
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE)
)
```

Arguments

`unit.exponent`	integer
`format`	character string, "R", "R.expresion", "R.character", or "LaTeX".
`label.text`	character Textual portion of the labels.
`scaled`	logical If `TRUE` relative units are assumed.
`normalized`	logical (`FALSE`) or numeric Normalization wavelength in manometers (nm).

`axis.symbols`	logical If `TRUE` symbols of the quantities are added to the `name`. Supported only by `format = "R.expression"`.
`pc.out`	logical, if TRUE use percent as default instead of fraction of one.
`Tfr.type`	character, either "total" or "internal".

Value

a character string or an R expression.

Note

Default for `label.text` depends on the value passed as argument to `Tfr.type`.

Examples

```
Tfr_label(Tfr.type = "internal")
Tfr_label(Tfr.type = "total")
Tfr_label(Tfr.type = "internal", axis.symbols = FALSE)


Tfr_internal_label()
Tfr_internal_label(format = "R.expression", axis.symbols = FALSE)
Tfr_internal_label(-2)
Tfr_internal_label(-3)
Tfr_internal_label(format = "R.expression")
Tfr_internal_label(format = "LaTeX")
Tfr_internal_label(-3, format = "LaTeX")


Tfr_total_label()
Tfr_total_label(format = "R.expression", axis.symbols = FALSE)
Tfr_total_label(-2)
Tfr_total_label(-3)
Tfr_total_label(format = "R.expression")
Tfr_total_label(format = "LaTeX")
Tfr_total_label(-3, format = "LaTeX")
```

`w_length_label` *Wave- axis labels*

Description

Generate wavelength, wavenumber, wave frequency, and energy per photon 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

```
w_length_label(
  unit.exponent = -9,
  format = getOption("photobiology.math", default = "R.expression"),
 label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["w.length"]],
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE)
)

w_number_label(
  unit.exponent = 0,
  format = getOption("photobiology.math", default = "R.expression"),
 label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["w.number"]],
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE)
)

w_frequency_label(
  unit.exponent = 9,
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["freq"]],
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE)
)

w_energy_eV_label(
  unit.exponent = 0,
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["energy"]],
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE)
)

w_energy_J_label(
  unit.exponent = -18,
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels(append = ifelse(axis.symbols, ",", ""))[["energy"]],
  axis.symbols = getOption("ggspectra.axis.symbols", default = TRUE)
)
```

Arguments

`unit.exponent`	integer The exponent in base 10 of the scale multiplier to use.
`format`	character string, "R", "R.expresion", "R.character", or "LaTeX".
`label.text`	character Textual portion of the labels.
`axis.symbols`	logical If `TRUE` symbols of the quantities are added to the `name`. Supported only by `format = "R.expression"`.

Details

By default labels consist in a textual name for the quantity, a symbol separated by a comma and units with scale factor in parenthesis. The textual names are by default in English but this default can be overridden for example with translations to a different language. To change or translate the default texts please see `axis_labels_uk`. The markup language used for the labels can be selected through a parameter argument, with character strings ready to be parsed into R expressions as default.

Wavelengths are assumed to be expressed in nanometres in the data. The `unit.exponent` corresponds to that desired for the tick labels with the corresponding axis label automatically set to an SI scale factor if possible, and otherwise shown as a power of 10.

These functions are used internally by *x* scales; see `sec_axis_w_number` and `scale_x_wl_continuous`. The scales and secondary axis functions should be used except when defining new scale functions.

Value

a character string or an R expression.

Examples

```
w_length_label()
w_length_label(axis.symbols = FALSE)
w_length_label(format = "R.expression")
w_length_label(format = "LaTeX")
w_number_label()
w_number_label(format = "R.expression")
w_frequency_label()
w_frequency_label(format = "R.expression")
w_energy_J_label()
w_energy_eV_label()
```

`w_number` *Deprecated functions*

Description

To convert wavelength into wavenumber or into frequency, please, use the conversion functions from package 'photobiology' in place of the deprecated functions `w_number()` and `w_frequency()` from this package.

Usage

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

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

Arguments

`w.length`	numeric wavelength (nm)
`unit.exponent`	integer Exponent of the scale multiplier implicit in result, e.g., use 3 for kJ.

Deprecated

These functions will be removed from package 'ggpmisc' in the near future.

See Also

See `wl2wavenumber` for the functions to be used in all new code.

Examples

```
library(photobiology)

wl2wavenumber(600)
wl2frequency(600)
```

Index