Package 'ggspectra'

October 4, 2020

Type Package

Title Extensions to 'ggplot2' for Radiation Spectra

Version 0.3.7

Date 2020-10-03

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)

LazyData TRUE

LazyLoad TRUE

ByteCompile TRUE

Depends R (>= 3.6.0), photobiology (>= 0.10.5), ggplot2 (>= 3.3.2)

Imports photobiologyWavebands (>= 0.4.3), scales (>= 1.1.1), ggrepel (>= 0.8.2), dplyr (>= 1.0.2), lubridate (>= 1.7.9), tidyr (>= 1.0.2), tibble (>= 3.0.3)

Suggests rlang (>= 0.4.7), knitr (>= 1.29), rmarkdown (>= 2.3)

URL https://docs.r4photobiology.info/ggspectra/, https://bitbucket.org/aphalo/ggspectra/

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

Encoding UTF-8

RoxygenNote 7.1.1

VignetteBuilder knitr

NeedsCompilation no

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

Repository CRAN

Date/Publication 2020-10-04 04:20:02 UTC

R topics documented:

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

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

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

minima (valleys) present in spectral data can also be annotated with statistics made available by the 'ggspectra' package.

Package 'ggspectra' is useful solely for plotting spectral data as most functions depend on the x aesthetic being mapped to a variable containing wavelength values expressed in nanometres. It works well together with some other extensions to package 'ggplot2' such as packages 'ggrepel' and 'cowplot'.

This package is part of a suite of R packages for photobiological calculations described at the [r4photobiology](https://www.r4photobiology.info) web site.

Note

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

Author(s)

Maintainer: Pedro J. Aphalo <pedro.aphalo@helsinki.fi> (ORCID)

Other contributors:

- Titta K. Kotilainen (ORCID) [contributor]

References

Aphalo, Pedro J. (2015) The r4photobiology suite. UV4Plants Bulletin, 2015:1, 21-29. https://doi.org/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://bitbucket.org/aphalo/ggspectra/
- Report bugs at https://bitbucket.org/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()

plot(sun.spct)

plot(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 = 0,
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels()[["Afr"]],
  scaled = FALSE,
  normalized = FALSE
)

Rfr_total_label(
  unit.exponent = 0,
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = NULL,
  scaled = FALSE,
  normalized = 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).

Value

a character string or an R expression.

Examples

```
Afr_label()
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(-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` *Create a complete ggplot for an irradiation calibration spectrum.*

Description

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

Usage

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

```
  object.label = deparse(substitute(object)),
  na.rm = TRUE
)

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

Arguments

object	a calibration_spct object or a calibration_mspct object.
...	in the case of collections of spectra, additional arguments passed to the plot methods for individual spectra, otherwise currently ignored.
w.band	a single waveband object or a list of waveband objects.
range	an R object on which range() returns a vector of length 2, with min annd max wavelengths (nm).
unit.out	character IGNORED.
pc.out	logical, if TRUE use percents instead of fraction of one.
label.qty	character string giving the type of summary quantity to use for labels, one of "mean", "total", "contribution", and "relative".
span	a peak is defined as an element in a sequence which is greater than all other elements within a window of width span centered at that element.
wls.target	numeric vector indicating the spectral quantity values for which wavelengths are to be searched and interpolated if need. The `character` strings "half.maximum" and "half.range" are also accepted as arguments. A list with `numeric` and/or `character` values is also accepted.
annotations	a character vector ("summaries" is ignored).
time.format	character Format as accepted by `strptime`.
tz	character Time zone to use for title and/or subtitle.
norm	numeric normalization wavelength (nm) or character string "max" for normalization at the wavelength of highest peak.
text.size	numeric size of text in the plot decorations.
idfactor	character Name of an index column in data holding a `factor` with each spectrum in a long-form multispectrum object corresponding to a distinct spectrum. If `idfactor=NULL` the name of the factor is retrieved from metadata or if no metadata found, the default "spct.idx" is tried. If `idfactor=NA` no aesthetic is mapped to the spectra and the user needs to use 'ggplot2' functions to manually map an aesthetic or use facets for the spectra.
ylim	numeric y axis limits,
object.label	character The name of the object being plotted.
na.rm	logical.
plot.data	character Data to plot. Default is "as.is" plotting one line per spectrum. When passing "mean" or "median" as argument all the spectra must contain data at the same wavelength values.

Value

a ggplot object.

Note

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

See Also

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

---

autoplot.cps_spct *Create a complete ggplot for detector-counts per second spectra.*

Description

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

Usage

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

Arguments

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

Value

a ggplot object.

Note

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

See Also

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

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

Description

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

Usage

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

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

Arguments

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

w.band	a single waveband object or a list of waveband objects.
range	an R object on which range() returns a vector of length 2, with min annd max wavelengths (nm).
plot.qty	character string one of "transmittance" or "absorbance".
pc.out	logical, if TRUE use percents instead of fraction of one.
label.qty	character string giving the type of summary quantity to use for labels, one of "mean", "total", "contribution", and "relative".
span	a peak is defined as an element in a sequence which is greater than all other elements within a window of width span centered at that element.
wls.target	numeric vector indicating the spectral quantity values for which wavelengths are to be searched and interpolated if need. The character strings "half.maximum" and "half.range" are also accepted as arguments. A list with numeric and/or character values is also accepted.
annotations	a character vector.
time.format	character Format as accepted by strptime.
tz	character Time zone to use for title and/or subtitle.
text.size	numeric size of text in the plot decorations.
chroma.type	character one of "CMF" (color matching function) or "CC" (color coordinates) or a chroma_spct object.
idfactor	character Name of an index column in data holding a factor with each spectrum in a long-form multispectrum object corresponding to a distinct spectrum. If idfactor=NULL the name of the factor is retrieved from metadata or if no metadata found, the default "spct.idx" is tried. If idfactor=NA no aesthetic is mapped to the spectra and the user needs to use 'ggplot2' functions to manually map an aesthetic or use facets for the spectra.
ylim	numeric y axis limits,
object.label	character The name of the object being plotted.
na.rm	logical.
plot.data	character Data to plot. Default is "as.is" plotting one line per spectrum. When passing "mean" or "median" as argument all the spectra must contain data at the same wavelength values.

Value

a ggplot object.

Note

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

See Also

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

Examples

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

---

autoplot.object_spct *Create a complete ggplot for a object spectrum.*

Description

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

Usage

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

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

Arguments

object	an object_spct object
...	in the case of collections of spectra, additional arguments passed to the plot methods for individual spectra, otherwise currently ignored.
w.band	a single waveband object or a list of waveband objects
range	an R object on which range() returns a vector of length 2, with min annd max wavelengths (nm)
plot.qty	character string, one of "all", "transmittance", "absorbance", "absorptance", or "reflectance".
pc.out	logical, if TRUE use percents instead of fraction of one
label.qty	character string giving the type of summary quantity to use for labels, one of "mean", "total", "contribution", and "relative".
span	a peak is defined as an element in a sequence which is greater than all other elements within a window of width span centered at that element.
wls.target	numeric vector indicating the spectral quantity values for which wavelengths are to be searched and interpolated if need. The character strings "half.maximum" and "half.range" are also accepted as arguments. A list with numeric and/or character values is also accepted.
annotations	a character vector
time.format	character Format as accepted by strptime.
tz	character Time zone to use for title and/or subtitle.
stacked	logical
text.size	numeric size of text in the plot decorations.
chroma.type	character one of "CMF" (color matching function) or "CC" (color coordinates) or a chroma_spct object.
idfactor	character Name of an index column in data holding a factor with each spectrum in a long-form multispectrum object corresponding to a distinct spectrum. If idfactor=NULL the name of the factor is retrieved from metadata or if no metadata found, the default "spct.idx" is tried. If idfactor=NA no aesthetic is mapped to the spectra and the user needs to use 'ggplot2' functions to manually map an aesthetic or use facets for the spectra.
ylim	numeric y axis limits,
object.label	character The name of the object being plotted.
na.rm	logical.

Value

a ggplot object.

Note

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

See Also

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

Examples

```
autoplot(Ler_leaf.spct)
```

---

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

Description

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

Usage

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

```
  object.label = deparse(substitute(object)),
  na.rm = TRUE
)
```

Arguments

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

Value

a ggplot object.

Note

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

See Also

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

`autoplot.reflector_spct` *Create a complete ggplot for a reflector spectrum.*

Description

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

Usage

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

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

Arguments

object	a reflector_spct object or a reflector_mspct object.
...	in the case of collections of spectra, additional arguments passed to the plot methods for individual spectra, otherwise currently ignored.
w.band	a single waveband object or a list of waveband objects.
range	an R object on which range() returns a vector of length 2, with min annd max wavelengths (nm).
plot.qty	character string (currently ignored).
pc.out	logical, if TRUE use percents instead of fraction of one.
label.qty	character string giving the type of summary quantity to use for labels, one of "mean", "total", "contribution", and "relative".
span	a peak is defined as an element in a sequence which is greater than all other elements within a window of width span centered at that element.
wls.target	numeric vector indicating the spectral quantity values for which wavelengths are to be searched and interpolated if need. The character strings "half.maximum" and "half.range" are also accepted as arguments. A list with numeric and/or character values is also accepted.
annotations	a character vector.
time.format	character Format as accepted by strptime.
tz	character Time zone to use for title and/or subtitle.
text.size	numeric size of text in the plot decorations.
chroma.type	character one of "CMF" (color matching function) or "CC" (color coordinates) or a chroma_spct object.
idfactor	character Name of an index column in data holding a factor with each spectrum in a long-form multispectrum object corresponding to a distinct spectrum. If idfactor=NULL the name of the factor is retrieved from metadata or if no metadata found, the default "spct.idx" is tried. If idfactor=NA no aesthetic is mapped to the spectra and the user needs to use 'ggplot2' functions to manually map an aesthetic or use facets for the spectra.
ylim	numeric y axis limits,
object.label	character The name of the object being plotted.
na.rm	logical.
plot.data	character Data to plot. Default is "as.is" plotting one line per spectrum. When passing "mean" or "median" as argument all the spectra must contain data at the same wavelength values.

Value

a ggplot object.

Note

The ggplot object returned can be further manipulated and added to. Except when no annotations are added, limits are set for the x-axis and y-axis scales. The y scale limits are expanded to include all data, or at least to the range of expected values. Scales are further expanded so as to make space for the annotations.

See Also

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

Examples

```
autoplot(Ler_leaf_rflt.spct)
```

---

autoplot.response_spct

Create a complete ggplot for a response spectrum.

Description

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

Usage

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

```
  ylim = c(NA, NA),
  object.label = deparse(substitute(object)),
  na.rm = TRUE
)

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

Arguments

object	a response_spct object or a response_mspct object.
...	in the case of collections of spectra, additional arguments passed to the plot methods for individual spectra, otherwise currently ignored.
w.band	a single waveband object or a list of waveband objects.
range	an R object on which range() returns a vector of length 2, with min annd max wavelengths (nm).
unit.out	character string indicating type of radiation units to use for plotting: "photon" or its synomin "quantum", or "energy".
pc.out	logical, if TRUE use percents instead of fraction of one
label.qty	character string giving the type of summary quantity to use for labels, one of "mean", "total", "contribution", and "relative".
span	a peak is defined as an element in a sequence which is greater than all other elements within a window of width span centered at that element.
wls.target	numeric vector indicating the spectral quantity values for which wavelengths are to be searched and interpolated if need. The character strings "half.maximum" and "half.range" are also accepted as arguments. A list with numeric and/or character values is also accepted.
annotations	a character vector.
time.format	character Format as accepted by strptime.
tz	character Time zone to use for title and/or subtitle.
norm	numeric normalization wavelength (nm) or character string "max" for normalization at the wavelength of highest peak, or NULL for plotting the spectrum as is.
text.size	numeric size of text in the plot decorations.
idfactor	character Name of an index column in data holding a factor with each spectrum in a long-form multispectrum object corresponding to a distinct spectrum. If idfactor=NULL the name of the factor is retrieved from metadata or if no metadata found, the default "spct.idx" is tried.
ylim	numeric y axis limits,
object.label	character The name of the object being plotted.
na.rm	logical.
plot.data	character Data to plot. Default is "as.is" plotting one line per spectrum. When passing "mean" or "median" as argument all the spectra must contain data at the same wavelength values.

Value

a ggplot object.

Note

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

See Also

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

Examples

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

---

autoplot.source_spct *Create a complete ggplot for light-source spectra.*

Description

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

Usage

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

```
  idfactor = NULL,
  facets = FALSE,
  ylim = c(NA, NA),
  object.label = deparse(substitute(object)),
  na.rm = TRUE
)

## S3 method for class 'source_mspct'
autoplot(
  object,
  ...,
  range = NULL,
  unit.out = getOption("photobiology.radiation.unit", default = "energy"),
  plot.data = "as.is",
  idfactor = 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 min annd max wavelengths (nm).
unit.out	character string indicating type of radiation units to use for plotting: "photon" or its synomin "quantum", or "energy".
label.qty	character string giving the type of summary quantity to use for labels, one of "mean", "total", "contribution", and "relative".
span	a peak is defined as an element in a sequence which is greater than all other elements within a window of width span centered at that element.
wls.target	numeric vector indicating the spectral quantity values for which wavelengths are to be searched and interpolated if need. The `character` strings "half.maximum" and "half.range" are also accepted as arguments. A list with `numeric` and/or `character` values is also accepted.
annotations	a character vector.
time.format	character Format as accepted by `strptime`.
tz	character Time zone to use for title and/or subtitle.
text.size	numeric size of text in the plot decorations.
chroma.type	character one of "CMF" (color matching function) or "CC" (color coordinates) or a `chroma_spct` object.
idfactor	character Name of an index column in data holding a `factor` with each spectrum in a long-form multispectrum object corresponding to a distinct level of the factor.

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

Value

a ggplot object.

Note

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

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 collections of spectra is to create a factor named "spct.idx", but if a different name is passed, it will be used instead.

See Also

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

Examples

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

two_suns.mspct <- source_mspct(list(sun1 = sun.spct, sun2 = sun.spct / 2))
autoplot(two_suns.mspct)
autoplot(two_suns.mspct, idfactor = "Spectra")
autoplot(two_suns.mspct, facets = TRUE) # uses ggplot2's default
autoplot(two_suns.mspct, facets = 1) # one column
autoplot(two_suns.mspct, facets = 2) # two columns
```

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

Description

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

Usage

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

Arguments

object	a waveband object.
...	currently ignored.
w.length	numeric vector of wavelengths (nm)
range	an R object on which range() returns a vector of length 2, with min 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	the type of unit we assume as reference "energy" or "photon" based.
annotations	a character vector.

wb.trim	logical.
norm	numeric normalization wavelength (nm) or character string "max" for normalization at the wavelength of highest peak.
text.size	numeric size of text in the plot decorations.
ylim	numeric y axis limits,
object.label	character The name of the object being plotted.
na.rm	logical.

Value

a ggplot object.

Note

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

Effectiveness spectra are plotted expressing the spectral effectiveness either as $1 mol^-1 nm$ photons of $1 J^-1 nm$ which can 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, changes the responsivity according to Plank's law.

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

See Also

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

Examples

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

---

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

Description

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

Usage

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

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

Arguments

`object`	generic_spct The spectral object plotted.
`object.label`	character The name of the object being plotted.
`annotations`	character vector Annotations as described for `plot()` methods, values unrelated to title are ignored.
`time.format`	character Format as accepted by `strptime`.
`tz`	character time zone used in labels.
`default.title`	character vector The default used for `annotations = "title"`.

Details

`autotitle()` retrieves from object `object` metadata and passes it to `ggplot2::ggtitle()` as arguments for `title`, `subtitle` and `caption`. The specification for the tittle 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.

Value

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

Note

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

Examples

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

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

---

axis_labels *Default text for axis labels*

Description

Obtain texts used by default for axis labels in plots. They contain only the text part, but not symbols or units of expression. Can be used to change the language or to suppress the text.

Usage

```
axis_labels()

axis_labels_uk()

axis_labels_uk_comma()

axis_labels_none()
```

Value

A character vector

Examples

```
axis_labels()[["w.length"]]
```

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,
  Tfr.type
)

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

A_total_label(
  unit.exponent = 0,
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = NULL,
  scaled = FALSE,
  normalized = 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).
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_internal_label()
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(-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()[["counts"]],
  scaled = FALSE,
  normalized = 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).

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()[["cps"]],
  scaled = FALSE,
  normalized = 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).

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".
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.
`na.rm`	If FALSE (the default), removes missing values with a warning. If TRUE silently removes missing values.

show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders.
...	other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.

Details

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

Aesthetics

See geom_ribbon

See Also

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

Examples

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

---

ggplot *Create a new ggplot plot from spectral data.*

Description

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

Usage

```
## 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_mspct'
ggplot(data, mapping = NULL, ..., range = NULL, environment = parent.frame())

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

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

Arguments

data	Default spectrum dataset to use for plot. If not a spectrum, the methods used will be those defined in package ggplot2. See ggplot. If not specified, must be supplied in each layer added to the plot.
mapping	Default list of aesthetic mappings to use for plot. If not specified, in the case of spectral objects, a default mapping will be used.
...	Other arguments passed on to methods.
range	an R object on which range() returns a vector of length 2, with min and max wavelengths (nm).
unit.out	character string indicating type of units to use for plotting spectral irradiance or spectral response, "photon" or "energy".
environment	If a variable defined in the aesthetic mapping is not found in the data, ggplot will look for it in this environment. It defaults to using the environment in which ggplot() is called.
plot.qty	character string One of "transmittance", "absorptance" or "absorbance" for filter_spct objects, and in addition to these "reflectance", "all" or "as.is" for object_spct objects.

Details

ggplot() is typically used to construct a plot incrementally, using the + operator to add layers to the

existing ggplot object. This is advantageous in that the code is explicit about which layers are added and the order in which they are added. For complex graphics with multiple layers, initialization with `ggplot` is recommended.

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

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

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

Object spectra

In the case of class object_spct, the arguments `"all"` and `"as.is"` if passed to `plot.qty`, indicate in the first case that the data are to be converted into long form, to allow stacking, while in the second case `data` is copied unchanged to the plot object. `"reflectance"` passed to `plot.qty` converts `data` into a `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()[["e.mult"]],
  scaled = FALSE,
  normalized = 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).

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 *Create a complete ggplot for a spectrum.*

Description

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

Usage

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

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

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

Arguments

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

Details

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

Value

a `ggplot` object.

Note

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

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, annotations = "") # deprecated syntax
autoplot(sun.spct, annotations = "") # preferred syntax
```

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 = 0,
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = NULL,
  scaled = FALSE,
  normalized = FALSE,
  Rfr.type
)

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

```
  label.text = NULL,
  scaled = FALSE,
  normalized = 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).
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(-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 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 = 0,
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels()[["s.e.irrad"]],
  scaled = FALSE,
  normalized = FALSE
)

s.q.irrad_label(
  unit.exponent = ifelse(normalized, 0, -6),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels()[["s.q.irrad"]],
  scaled = FALSE,
  normalized = 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).

Value

a character string or an R expression.

Examples

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

---

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()[["s.e.response"]],
  scaled = FALSE,
  normalized = FALSE
)

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

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

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

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).

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_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),
  breaks = scales::pretty_breaks(n = 7),
  labels = SI_pl_format(exponent = unit.exponent + 9),
  label.text = axis_labels()[["w.length"]],
  ...
)
```

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.
`...`	other named arguments passed to `scale_y_continuous`

Note

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

Examples

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

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

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

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

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

---

scale_y_Afr_continuous

Absorptance y-scale

Description

Scale y continuous with defaults suitable for spectral absorptance.

Usage

```
scale_y_Afr_continuous(
  unit.exponent = 0,
  name = Afr_label(unit.exponent = unit.exponent, format = format, label.text =
    label.text, scaled = scaled, normalized = round(normalized, 1)),
  labels = SI_pl_format(exponent = unit.exponent),
  limits = c(0, 1),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels()[["Afr"]],
  scaled = FALSE,
  normalized = 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).
...	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()

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 = round(normalized, 1), 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,
  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 = round(normalized, 1), Tfr.type =
    "internal"),
  labels = SI_pl_format(exponent = unit.exponent),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = NULL,
  scaled = FALSE,
  normalized = FALSE,
  ...
)

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

---

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 = round(normalized, 1)),
  labels = SI_pl_format(exponent = unit.exponent),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels()[["counts"]],
  scaled = FALSE,
  normalized = FALSE,
  ...
)
```

```
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 = round(normalized, 1)),
  labels = SI_tg_format(exponent = unit.exponent),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels()[["counts"]],
  scaled = FALSE,
  normalized = 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.
`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).
`...`	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()

norm_led.raw_spct <- normalize(white_led.raw_spct[ , 1:2], norm = "max")

ggplot(norm_led.raw_spct) +
  geom_line() +
  scale_y_counts_continuous(normalized = getNormalized(norm_led.raw_spct)) +
  scale_x_wl_continuous()

ggplot(norm_led.raw_spct) +
  geom_line() +
  scale_y_counts_tg_continuous(normalized = getNormalized(norm_led.raw_spct)) +
  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 = round(normalized, 1)),
  labels = SI_pl_format(exponent = unit.exponent),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels()[["cps"]],
  scaled = FALSE,
  normalized = 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.
`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).
`...`	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()

norm_led.cps_spct <- normalize(white_led.cps_spct, norm = "max")

ggplot(norm_led.cps_spct) +
  geom_line() +
  scale_y_cps_continuous(normalized = getNormalized(norm_led.cps_spct)) +
  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 = round(normalized, 1)),
  labels = SI_pl_format(exponent = unit.exponent),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels()[["e.mult"]],
  scaled = FALSE,
```

```
  normalized = 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.
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).
...	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 = 0,
  name = Rfr_label(unit.exponent = unit.exponent, format = format, label.text =
   label.text, scaled = scaled, normalized = round(normalized, 1), 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,
  Rfr.type,
  ...
)

scale_y_Rfr_specular_continuous(
  unit.exponent = 0,
```

```
  name = Rfr_label(unit.exponent = unit.exponent, format = format, label.text =
    label.text, scaled = scaled, normalized = round(normalized, 1), 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,
  ...
)

scale_y_Rfr_total_continuous(
  unit.exponent = 0,
  name = Rfr_label(unit.exponent = unit.exponent, format = format, label.text =
   label.text, scaled = scaled, normalized = round(normalized, 1), 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,
  ...
)
```

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).
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()
```

---

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, format = format, label.text =
    label.text, scaled = scaled, normalized = round(normalized, 1)),
  labels = SI_pl_format(exponent = unit.exponent),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels()[["s.e.irrad"]],
  scaled = FALSE,
  normalized = FALSE,
  ...
)

scale_y_s.q.irrad_continuous(
```

```
  unit.exponent = ifelse(normalized, 0, -6),
 name = s.q.irrad_label(unit.exponent = unit.exponent, format = format, label.text =
    label.text, scaled = scaled, normalized = round(normalized, 1)),
  labels = SI_pl_format(exponent = unit.exponent),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels()[["s.q.irrad"]],
  scaled = FALSE,
  normalized = FALSE,
  ...
)

scale_y_s.e.irrad_log10(
  unit.exponent = 0,
 name = s.e.irrad_label(unit.exponent = unit.exponent, format = format, label.text =
    label.text, scaled = scaled, normalized = round(normalized, 1)),
  labels = SI_pl_format(exponent = unit.exponent),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels()[["s.e.irrad"]],
  scaled = FALSE,
  normalized = FALSE,
  ...
)

scale_y_s.q.irrad_log10(
  unit.exponent = ifelse(normalized, 0, -6),
 name = s.q.irrad_label(unit.exponent = unit.exponent, format = format, label.text =
    label.text, scaled = scaled, normalized = round(normalized, 1)),
  labels = SI_pl_format(exponent = unit.exponent),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels()[["s.q.irrad"]],
  scaled = FALSE,
  normalized = 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.
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).
...	other named arguments passed to scale_y_continuous

Note

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

Examples

```
ggplot(sun.spct) +
  geom_line() +
  scale_y_s.e.irrad_continuous() +
  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 spectral,") +
  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()

photon_as_default()
normalized_sun.spct <- normalize(sun.spct)
ggplot(normalized_sun.spct) +
  geom_line(na.rm = TRUE) +
  scale_y_s.q.irrad_continuous(normalized =
                               getNormalized(normalized_sun.spct)) +
  scale_x_wl_continuous()

unset_radiation_unit_default()
```

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 = round(normalized, 1)),
  labels = SI_pl_format(exponent = -unit.exponent),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels()[["s.e.response"]],
  scaled = FALSE,
  normalized = FALSE,
  ...
)

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 = round(normalized, 1)),
  labels = SI_pl_format(exponent = -unit.exponent),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels()[["s.q.response"]],
  scaled = FALSE,
  normalized = FALSE,
  ...
)

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 = round(normalized, 1)),
  labels = SI_pl_format(exponent = -unit.exponent),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels()[["s.e.action"]],
  scaled = FALSE,
  normalized = FALSE,
  ...
)

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 = round(normalized, 1)),
  labels = SI_pl_format(exponent = -unit.exponent),
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels()[["s.q.action"]],
  scaled = FALSE,
  normalized = 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.
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).
...	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()

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 = 0,
  name = Tfr_label(unit.exponent = unit.exponent, format = format, label.text =
   label.text, scaled = scaled, normalized = round(normalized, 1), 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,
  Tfr.type,
```

```
  ...
)

scale_y_Tfr_internal_continuous(
  unit.exponent = 0,
  name = Tfr_label(unit.exponent = unit.exponent, format = format, label.text =
    label.text, scaled = scaled, normalized = round(normalized, 1), 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,
  ...
)

scale_y_Tfr_total_continuous(
  unit.exponent = 0,
  name = Tfr_label(unit.exponent = unit.exponent, format = format, label.text =
   label.text, scaled = scaled, normalized = round(normalized, 1), 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,
  ...
)
```

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).
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()

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 and wavenumber.

Usage

```
sec_axis_w_number(unit.exponent = -6, label.text = axis_labels()[["w.number"]])

sec_axis_w_frequency(unit.exponent = 12, label.text = axis_labels()[["freq"]])
```

Arguments

unit.exponent	integer
label.text	character Textual portion of the labels.

Examples

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

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

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

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

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

---

set_annotations_default

Set defaults for autoplot annotations

Description

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

Usage

```
set_annotations_default(annotations = NULL)

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

Arguments

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

Details

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

Value

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

Note

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

See Also

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

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 color definitions according to human vision.

Usage

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

Arguments

`mapping`	The aesthetic mapping, usually constructed with `aes` or `aes_`. Only needs to be set at the layer level if you are overriding the plot defaults.
`data`	A layer specific dataset - only needed if you want to override the plot defaults.
`geom`	The geometric object to use display the data
`chroma.type`	character one of "CMF" (color matching function) or "CC" (color coordinates) or a `chroma_spct` object.
`position`	The position adjustment to use for overlapping points on this layer
`na.rm`	a logical value indicating whether NA values should be stripped before the computation proceeds.

show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders.
...	other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.

Value

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

Computed variable

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

Default aesthetics

Set by the statistic and available to geoms.

color ..wl.color..

fill ..wl.color..

Required aesthetics

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

x numeric, wavelength in nanometres

y numeric, a spectral quantity

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()
```

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.

Usage

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

Arguments

mapping	The aesthetic mapping, usually constructed with aes or aes_. Only needs to be set at the layer level if you are overriding the plot defaults.
data	A layer specific dataset - only needed if you want to override the plot defaults.
geom	The geometric object to use display the data
target	numeric value indicating the spectral quantity value for which wavelengths are to be searched and interpolated if need. The character string "half.maximum" is also accepted as argument.
interpolate	logical Indicating whether the nearest wavelength value in x should be returned or a value calculated by linear interpolation between wavelength values straddling the target.
chroma.type	character one of "CMF" (color matching function) or "CC" (color coordinates) or a chroma_spct object.
label.fmt	character string giving a format definition for converting values into character strings by means of function sprintf.
x.label.fmt	character string giving a format definition for converting x-values into character strings by means of function sprintf.

y.label.fmt	character string giving a format definition for converting y-values into character strings by means of function sprintf.
position	The position adjustment to use for overlapping points on this layer
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders.
...	other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.

Details

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

Value

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

Computed variables

x x-value at or nearest to the match to the target as numeric

y target value or y-value nearest to the target as numeric

x.label x-value at or nearest to the match formatted as character

y.label target value or y-value nearest to the target formatted as character

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

Default aesthetics

Set by the statistic and available to geoms.

label ..x.label..

xintercept ..x..

yintercept ..y..

fill ..color..

Required aesthetics

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

x numeric, wavelength in nanometres

y numeric, a spectral quantity

Note

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

See Also

`find_peaks`.

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

Examples

```
# ggplot() methods for spectral objects set a default mapping for x and y.
ggplot(yellow_gel.spct) + geom_line() +
  stat_find_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.

Usage

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

Arguments

mapping	The aesthetic mapping, usually constructed with aes or aes_. Only needs to be set at the layer level if you are overriding the plot defaults.
data	A layer specific dataset - only needed if you want to override the plot defaults.
geom	The geometric object to use display the data
target	numeric vector indicating the spectral quantity values for which wavelengths are to be searched and interpolated if need. The character strings "half.maximum" and "half.range" are also accepted as arguments. A list with numeric and/or character values is also accepted.
interpolate	logical Indicating whether the nearest wavelength value in x should be returned or a value calculated by linear interpolation between wavelength values stradling the target.
chroma.type	character one of "CMF" (color matching function) or "CC" (color coordinates) or a chroma_spct object.
label.fmt	character string giving a format definition for converting values into character strings by means of function sprintf.
x.label.fmt	character string giving a format definition for converting x-values into character strings by means of function sprintf.
y.label.fmt	character string giving a format definition for converting y-values into character strings by means of function sprintf.
position	The position adjustment to use for overlapping points on this layer
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.

inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders.
...	other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.

Details

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

Value

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

Computed variables

x x-value at or nearest to the match to the target as numeric

y target value or y-value nearest to the target as numeric

x.label x-value at or nearest to the match formatted as character

y.label target value or y-value nearest to the target formatted as character

wl.color color definition calculated by assuming that x-values are wavelengths expressed in 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 local maxima are located, and adds labels and colors to the data wihtout subsetting. To find local minima, you can use stat_labels_valleys instead.

Usage

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

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

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
span	a peak is defined as an element in a sequence which is greater than all other elements within a window of width span centered at that element. The default value is 5, meaning that a peak is bigger than two consequtive neighbors on each side. Default: 5.
ignore_threshold	numeric value between 0.0 and 1.0 indicating the size threshold below which peaks will be ignored.
strict	logical flag: if TRUE, an element must be strictly greater than all other values in its window to be considered a peak. Default: FALSE.
chroma.type	character one of "CMF" (color matching function) or "CC" (color coordinates) or a chroma_spct object.
label.fmt	character string giving a format definition for converting values into character strings by means of function sprintf.
x.label.fmt	character string giving a format definition for converting x-values into character strings by means of function sprintf.
y.label.fmt	character string giving a format definition for converting y-values into character strings by means of function sprintf.
label.fill	character string ot use for labels not at peaks or valleys being highlighted.
position	The position adjustment to use for overlapping points on this layer
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.

show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders.
...	other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.

Details

These stats use geom_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 overlapT any observation in the whole data set. Default aesthetics set by these stats allow their direct use with geom_text, geom_label, geom_line, geom_rug, geom_hline and geom_vline. The formatting of the labels returned can be controlled by the user.

Value

The original data with additional computed variables added.

Computed variables

x.label x-value at a peak (or valley) formatted as character or otherwise the value passed to label.fill which defaults to an empty string ("").

y.label y-value at the peak (or valley) formatted as character or otherwise the value passed to label.fill which defaults to an empty string ("").

wl.color At peaks and valleys, color definition calculated by assuming that x-values are wavelengths expressed in nanometres, otherwise, rgb(1,1,1,0) (transparent white).

Default aesthetics

Set by the statistic and available to geoms.

label ..x.label..

xintercept ..x..

yintercept ..y..

color black_or_white(..wl.color..)

fill ..wl.color..

Required aesthetics

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

x numeric, wavelength in nanometres

y numeric, a spectral quantity

Note

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

See Also

stat_peaks, stat_valleys and find_peaks, which is used internally.

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

Examples

```
# 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")

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 = 3) +
  stat_label_peaks(span = 41, geom = "label_repel", segment.colour = "red",
                   nudge_y = 0.12, label.fmt = "%3.0f nm", vjust = 1) +
  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.

Usage

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

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

```
  ...
)
```

Arguments

mapping	The aesthetic mapping, usually constructed with aes or aes_. Only needs to be set at the layer level if you are overriding the plot defaults.
data	A layer specific dataset - only needed if you want to override the plot defaults.
geom	The geometric object to use display the data
position	The position adjustment to use for overlapping points on this layer
...	other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.
span	integer A peak is defined as an element in a sequence which is greater than all other elements within a window of width span centered at that element. Use NULL for the global peak. Valleys are the reverse.
ignore_threshold	numeric For peaks, value between 0.0 and 1.0 indicating the relative size of peaks compared to tallest peak threshold below which peaks will be ignored, while negative values between 0.0 and -1.0 set a threshold so that the tallest peaks are ignored, instead of the shortest. For valleys, value between 0.0 and 1.0 indicating the relative depth of valleys below which valleys will be ignored, while negative values between 0.0 and -1.0 set a threshold so that the deeper valleys are ignored, instead of the shallower ones.
strict	logical If TRUE, an element must be strictly greater than all other values in its window to be considered a peak.
refine.wl	logical Flag indicating if peak or valleys locations should be refined by fitting a function.
method	character String with the name of a method used for peak fitting. Currently only spline interpolation is implemented.
chroma.type	character one of "CMF" (color matching function) or "CC" (color coordinates) or a chroma_spct object.
label.fmt	character string giving a format definition for converting values into character strings by means of function sprintf.
x.label.fmt	character string giving a format definition for converting x-values into character strings by means of function sprintf.
y.label.fmt	character string giving a format definition for converting y-values into character strings by means of function sprintf.
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders.

Details

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

Value

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

Computed variables

x x-value at the peak (or valley) as numeric

y y-value at the peak (or valley) as numeric

x.label x-value at the peak (or valley) formatted as character

y.label y-value at the peak (or valley) formatted as character

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

BW.color color definition, either "black" or "white", as needed to ensure high contrast to `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.
ggplot(sun.spct) +
  geom_line() +
  stat_peaks()

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

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

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

ggplot(sun.spct) +
  geom_line() +
  stat_peaks(span = 51, geom = "point", colour = "red", refine.wl = TRUE) +
  stat_peaks(mapping = aes(fill = stat(wl.colour), color = stat(BW.colour)),
             span = 51, geom = "label",
             size = 3, vjust = -0.2, label.fmt = "%.3g nm",
             refine.wl = TRUE) +
  stat_valleys(span = 71, geom = "point", colour = "blue", refine.wl = TRUE) +
  stat_valleys(mapping = aes(fill = stat(wl.colour), color = stat(BW.colour)),
               span = 71, geom = "label",
               size = 3, vjust = 1.2, label.fmt = "%.3g nm",
               refine.wl = TRUE) +
  expand_limits(y = 4e-6) +
  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.

Usage

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

Arguments

mapping	The aesthetic mapping, usually constructed with aes or aes_. Only needs to be set at the layer level if you are overriding the plot defaults.
data	A layer specific dataset - only needed if you want to override the plot defaults.
geom	The geometric object to use display the data
position	The position adjustment to use for overlapping points on this layer
...	other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.
z.threshold	numeric Modified Z values larger than z.threshold are considered to be spikes.
max.spike.width	integer Wider regions with high Z values are not detected as spikes.
chroma.type	character one of "CMF" (color matching function) or "CC" (color coordinates) or a chroma_spct object.
label.fmt	character string giving a format definition for converting values into character strings by means of function sprintf.
x.label.fmt	character string giving a format definition for converting x-values into character strings by means of function sprintf.

y.label.fmt	character string giving a format definition for converting y-values into character strings by means of function sprintf.
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders.

Details

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

Value

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

Computed variables

x x-value at the peak (or valley) as numeric

y y-value at the peak (or valley) as numeric

x.label x-value at the peak (or valley) formatted as character

y.label y-value at the peak (or valley) formatted as character

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

BW.color color definition that either "black" or "white", to ensure high contrast to wl.color.

Default aesthetics

Set by the statistic and available to geoms.

label stat(x.label)

xintercept stat(x)

yintercept stat(y)

fill stat(wl.color)

Required aesthetics

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

x numeric, wavelength in nanometres

y numeric, a spectral quantity

Note

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

See Also

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

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

Examples

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

# two spurious(?) spikes
ggplot(sun.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 "rect" geom.

Usage

```
stat_wb_box(
  mapping = NULL,
  data = NULL,
  geom = "rect",
  w.band = NULL,
  chroma.type = "CMF",
  ypos.mult = 1.07,
  ypos.fixed = NULL,
  box.height = 0.06,
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)
```

Arguments

mapping	The aesthetic mapping, usually constructed with aes or aes_. Only needs to be set at the layer level if you are overriding the plot defaults.
data	A layer specific dataset - only needed if you want to override the plot defaults.
geom	The geometric object to use display the data
w.band	a waveband object or a list of waveband objects or numeric vector of at least length two.
chroma.type	character one of "CMF" (color matching function) or "CC" (color coordinates) or a chroma_spct object.
ypos.mult	numeric Multiplier constant used to 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.
position	The position adjustment to use for overlapping points on this layer
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.

show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders.
...	other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.

Value

A data frame with one row for each waveband object in the argument to w.band. 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

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

See Also

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

Examples

```
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 "rect" geom.

Usage

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

```
  inherit.aes = TRUE,
  ...
)
```

Arguments

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

Value

A data frame with one row for each waveband object in the argument to w.band. 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 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 "rect", "hline", "vline", "text" and "label" geoms displaying "contributions" per waveband to the total of the spectral integral.

Usage

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

Arguments

mapping	The aesthetic mapping, usually constructed with aes or aes_. Only needs to be set at the layer level if you are overriding the plot defaults.
data	A layer specific dataset - only needed if you want to override the plot defaults.
geom	The geometric object to use display the data
w.band	a waveband object or a list of waveband objects or numeric vector of at least length two.
integral.fun	function on x and y.
label.mult	numeric Scaling factor applied to y-integral values before conversion into character strings.
chroma.type	character one of "CMF" (color matching function) or "CC" (color coordinates) or a chroma_spct object.
label.fmt	character string giving a format definition for converting y-integral values into character strings by means of function sprintf.

ypos.mult	numeric Multiplier constant used to scale returned y values.
ypos.fixed	numeric If not NULL used a constant value returned in y.
position	The position adjustment to use for overlapping points on this layer
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders.
...	other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.

Value

A data frame with one row for each waveband object in the argument to w.band. 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 = 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 nanaometre of wavelength for each waveband in the input. Sets suitable default aesthetics for geoms "errorbarh" and "hline" from 'ggplot', and "linerangeh", and "errorbarh" from 'ggstance'.

Usage

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

Arguments

mapping	The aesthetic mapping, usually constructed with aes or aes_. Only needs to be set at the layer level if you are overriding the plot defaults.
data	A layer specific dataset - only needed if you want to override the plot defaults.
geom	The geometric object to use display the data
w.band	a waveband object or a list of waveband objects or numeric vector of at least length two.
integral.fun	function on x and y.
chroma.type	character one of "CMF" (color matching function) or "CC" (color coordinates) or a chroma_spct object.
ypos.fixed	numeric If not NULL used a constant value returned in y.
position	The position adjustment to use for overlapping points on this layer

na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders.
...	other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.

Value

A data frame with one row for each waveband object in the argument to w.band. 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 computes areas under a curve.

Usage

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

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

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

Arguments

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

Value

A data frame with one row for each waveband object in the argument to `w.band`. 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 = 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 computes the center of a waveband. Sets suitable default aesthetics for "text" and "label" geoms displaying "boundaries" and "names" of wavebands.

Usage

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

Arguments

mapping	The aesthetic mapping, usually constructed with aes or aes_. Only needs to be set at the layer level if you are overriding the plot defaults.
data	A layer specific dataset - only needed if you want to override the plot defaults.
geom	The geometric object to use display the data
w.band	a waveband object or a list of waveband objects or numeric vector of at least length two.
chroma.type	character one of "CMF" (color matching function) or "CC" (color coordinates) or a chroma_spct object.
label.fmt	character string giving a format definition for formating the name of the waveband. sprintf.
ypos.fixed	numeric If not NULL used a constant value returned in y.
position	The position adjustment to use for overlapping points on this layer
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.

inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders.
...	other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.

Value

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

Computed variables

x w.band-midpoint

wb.xmin w.band minimum

wb.xmax w.band maximum

y ypos.fixed or zero

wb.color color of the w.band

wb.name label of w.band

wb.label formatted wb.name

Default aesthetics

Set by the statistic and available to geoms.

label ..wb.label..

x ..x..

xmin ..wb.xmin..

xmax ..wb.xmax..

fill ..wb.color..

Required aesthetics

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

x numeric, wavelength in nanometres

Note

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

See Also

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

Examples

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

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

---

stat_wb_mean *Integrate ranges under curve.*

Description

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

Usage

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

Arguments

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

Value

A data frame with one row for each waveband object in the argument to w.band. 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 = 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 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 "rect", "hline", "vline", "text" and "label" geoms displaying values per waveband "relative" to the sum of the wavebands.

Usage

```
stat_wb_relative(
  mapping = NULL,
  data = NULL,
```

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

Arguments

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

Value

A data frame with one row for each waveband object in the argument to `w.band`. 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 areas under a curve.

Usage

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

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

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

```
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)
```

Arguments

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

Value

A data frame with one row for each waveband object in the argument to w.band. 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. 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 "rect", "hline", "vline", "text" and "label" geoms displaying "totals" per waveband.

Usage

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

Arguments

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

Value

A data frame with one row for each waveband object in the argument to w.band. 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.

Usage

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

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

Arguments

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

geom	The geometric object to use display the data
w.band	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.
position	The position adjustment to use for overlapping points on this layer
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders.
...	other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.
ymin, ymax	numeric used as aesthetics for plotting the guide.

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

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

See Also

color_of, which is used internally.

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

Examples

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

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

---

stat_wl_summary *Average area under curve for regions.*

Description

stat_wl_summary computes the area under a curve.

Usage

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

Arguments

mapping	The aesthetic mapping, usually constructed with 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
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.
position	The position adjustment to use for overlapping points on this layer
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders.
...	other arguments passed on to layer. This can include aesthetics whose values you want to set, not map. See layer for more details.

Value

A data frame with one row.

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 = 0,
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = NULL,
  scaled = FALSE,
  normalized = FALSE,
  Tfr.type
)

Tfr_internal_label(
  unit.exponent = 0,
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = NULL,
  scaled = FALSE,
```

```
  normalized = FALSE
)

Tfr_total_label(
  unit.exponent = 0,
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = NULL,
  scaled = FALSE,
  normalized = 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).
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_internal_label()
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(-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 and wave frequency axis labels in SI units, using SI scale factors. Output can be selected as character, expression (R default devices) or LaTeX (for tikz device).

Usage

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

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

w_frequency_label(
  unit.exponent = 9,
  format = getOption("photobiology.math", default = "R.expression"),
  label.text = axis_labels()[["freq"]]
)
```

Arguments

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

Value

a character string or an R expression.

Examples

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

```
w_frequency_label("R.expression")
```

---

w_number *Wavelength conversions*

Description

Convert wavelength into wavenumber or into frequency.

Usage

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

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

Arguments

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

Examples

```
w_number(600)
w_frequency(600)
```

Index