Package 'pavo'

May 9, 2026

Title Perceptual Analysis, Visualization and Organization of Spectral Colour Data

Version 2.9.0

Description A cohesive framework for the spectral and spatial analysis of colour described in Maia, Eliason, Bitton, Doucet & Shawkey (2013) <doi:10.1111/2041-210X.12069> and Maia, Gruson, Endler & White (2019) <doi:10.1111/2041-210X.13174>.

License GPL (>= 2)

URL http://pavo.colrverse.com, https://github.com/rmaia/pavo/

BugReports https://github.com/rmaia/pavo/issues

Depends R (>= 3.5.0)

Imports cluster, future.apply, geometry(>= 0.4.0), lightr(>= 1.0), magick, farver, plot3D, progressr, sf, viridisLite

Suggests alphashape3d, imager, knitr, mapproj, rgl, rmarkdown, testthat(>= 2.99.0), vdiffr, digest

VignetteBuilder knitr

Config/Needs/website pkgdown

Encoding UTF-8

Language en-GB

RoxygenNote 7.2.3

Config/testthat/edition 3

NeedsCompilation no

Author Thomas White [aut, cre] (ORCID: <https://orcid.org/0000-0002-3976-1734>), Rafael Maia [aut] (ORCID: <https://orcid.org/0000-0002-7563-9795>), Hugo Gruson [aut] (ORCID: <https://orcid.org/0000-0002-4094-1476>), John Endler [aut], Chad Eliason [aut], Pierre-Paul Bitton [aut] (ORCID: <https://orcid.org/0000-0001-5984-2331>)

Maintainer Thomas White <thomas.white026@gmail.com>

Repository CRAN

Date/Publication 2023-09-24 10:10:02 UTC

Contents

`adjacent` *Run an adjacency and boundary strength analysis*

Description

Calculate summary variables from the adjacency (Endler 2012) and boundary-strength (Endler et al. 2018) analyses, along with overall pattern contrast (Endler & Mielke 2005).

Usage

```
adjacent(
  classimg,
  xpts = NULL,
  xscale = NULL,
  bkgID = NULL,
  polygon = NULL,
  exclude = c("none", "background", "object"),
  coldists = NULL,
  hsl = NULL
)
```

Arguments

`classimg`	(required) an xyz matrix, or list of matrices, in which x and y correspond to spatial (e.g. pixel) coordinates, and z is a numeric code specifying a colour-class. Preferably the result of `classify()`, or constructed from grid-sampled spectra that have been visually modelled and clustered (as per Endler 2012).
`xpts`	an integer specifying the number of sample points along the x axis, from which the evenly-spaced sampling grid is constructed (if required). Defaults to the smallest dimension of `classimg`, though this should be carefully considered.
`xscale`	(required) an integer or list of integers equal in length to classimg() specifying the true length of the x-axis, in preferred units. Not required, and ignored, only if image scales have been set via `procimg()`.
`bkgID`	an integer or vector specifying the colour-class ID number(s) of pertaining to the background alone, for relatively homogeneous and uniquely-identified backgrounds (e.g. the matte background of pinned specimens). Examine the attributes of, or call `summary` on, the result of `classify()` to visualise the RGB values corresponding to colour-class ID numbers for classified images. Ignored if the focal object and background has been identified using `procimg()`.

`polygon` a data.frame of x-y coordinates delineating a closed polygon that separates the focal object from the background. Not required, and ignored, if the focal object outline is specified using `procimg()`.

`exclude` the portion of the scene to be excluded from the analysis, if any.

- `'none'`: default
- `'background'`: exclude everything *outside* the closed polygon specified using `procimg()`, or the argument `polygon`. Alternatively, if the background is relatively homogeneous the colour-class ID(s) uniquely corresponding to the background can be specified via `bkgID`, and subsequently excluded.
- `'object'`: exclude everything *inside* the closed polygon specified using `procimg()`, or the argument `polygon`.

`coldists` a data.frame specifying the visually-modelled chromatic (dS) and/or achromatic (dL) distances between colour-categories. The first two columns should be named 'c1' and 'c2', and specify all possible combinations of numeric colour-class ID's (viewable by calling `summary(image, plot = TRUE)` on a colour classified image), with the remaining columns named dS (for chromatic distances) and/or dL (for achromatic distances). See `vismodel()` and `colspace()` for visual modelling with spectral data.

`hsl` data.frame specifying the hue, saturation, and luminance of color patch elements, as might be estimated via `vismodel()` and `colspace()`. The first column, named 'patch', should contain numeric color category IDs, with the remaining columns specifying one or more of 'hue' (angle, in radians), 'sat', and/or 'lum'.

Details

You can customise the type of parallel processing used by this function with the `future::plan()` function. This works on all operating systems, as well as high performance computing (HPC) environment. Similarly, you can customise the way progress is shown with the `progressr::handlers()` functions (progress bar, acoustic feedback, nothing, etc.)

Value

a data frame of summary variables:

- `'k'`: The number of user-specified colour and/or luminance classes.
- `'N'`: The grand total (sum of diagonal and off-diagonal) transitions.
- `'n_off'`: The total off-diagonal transitions.
- `'p_i'`: The overall frequency of colour class *i*.
- `'q_i_j'`: The frequency of transitions between *all* colour classes *i* and *j*, such that `sum(q_i_j) = 1`.
- `'t_i_j'`: The frequency of off-diagonal (i.e. class-change transitions) transitions *i* and *j*, such that `sum(t_i_j) = 1`.
- `'m'`: The overall transition density (mean transitions), in units specified in the argument `xscale`.

- 'm_r': The row-wise transition density (mean row transitions), in user-specified units.
- 'm_c': The column-wise transition density (mean column transitions), in user-specified units.
- 'A': The transition aspect ratio (< 1 = wide, > 1 = tall).
- 'Sc': Simpson colour class diversity, Sc = 1/(sum(p_i^2)). If all colour and luminance classes are equal in relative area, then Sc = k.
- 'St': Simpson transition diversity, St = 1/sum(t_i_j^2).
- 'Jc': Simpson colour class diversity relative to its achievable maximum. Jc = Sc/k.
- 'Jt': Simpson transition diversity relative to its achievable maximum. Jt = St/(k*(k-1)/2).
- 'B': The animal/background transition ratio, or the ratio of class-change transitions entirely within the focal object and those involving the object and background, B = sum(O_a_a / O_a_b).
- 'Rt': Ratio of animal-animal and animal-background transition diversities, Rt = St_a_a / St_a_b.
- 'Rab': Ratio of animal-animal and background-background transition diversities, Rt = St_a_a / St_b_b.
- 'm_dS', 's_dS', 'cv_dS': weighted mean, sd, and coefficient of variation of the chromatic boundary strength.
- 'm_dL', 's_dL', 'cv_dL': weighted mean, sd, and coefficient of variation of the achromatic boundary strength.
- 'm_hue', 's_hue', 'var_hue': circular mean, sd, and variance of overall pattern hue (in radians).
- 'm_sat', 's_sat', 'cv_sat': weighted mean, sd, and coefficient variation of overall pattern saturation.
- 'm_lum', 's_lum', 'cv_lum': weighted mean, sd, and coefficient variation of overall pattern luminance.

Author(s)

Thomas E. White <thomas.white026@gmail.com>

References

Endler, J. A. (2012). A framework for analysing colour pattern geometry: adjacent colours. Biological Journal Of The Linnean Society, 107(2), 233-253.

Endler, J. A., Cole G., Kranz A. (2018). Boundary Strength Analysis: Combining color pattern geometry and coloured patch visual properties for use in predicting behaviour and fitness. Methods in Ecology and Evolution, 9(12), 2334-2348.

Endler, J. A., & Mielke, P. (2005). Comparing entire colour patterns as birds see them. Biological Journal Of The Linnean Society, 86(4), 405-431.

See Also

classify(), summary.rimg(), procimg()

Examples

```
# Set a seed, for reproducibility
set.seed(153)

# Run the adjacency analysis on a single image of a butterfly
papilio <- getimg(system.file("testdata/images/butterflies/papilio.png", package = "pavo"))
papilio_class <- classify(papilio, kcols = 4)
papilio_adj <- adjacent(papilio_class, xscale = 100)

# Expand on the above, by including (fake) color distances and hsl values
# of colour elements in the image

# Generate fake color distances
distances <- data.frame(
  c1 = c(1, 1, 1, 2, 2, 3),
  c2 = c(2, 3, 4, 3, 4, 4),
  dS = c(5.3, 3.5, 5.7, 2.9, 6.1, 3.2),
  dL = c(5.5, 6.6, 3.3, 2.2, 4.4, 6.6)
)

# Generate some fake hue, saturation, luminance values
hsl_vals <- data.frame(
  patch = seq_len(4),
  hue = c(1.5, 2.2, 1.0, 0.5),
  lum = c(10, 5, 7, 3),
  sat = c(3.5, 1.1, 6.3, 1.3)
)

# Run the full analysis, including the white background's ID
papilio_adj <- adjacent(papilio_class,
  xscale = 100, bkgID = 1,
  coldists = distances, hsl = hsl_vals
)

# Run an adjacency analysis on multiple images.
# First load some images of coral snake colour patterns
snakes <- getimg(system.file("testdata/images/snakes", package = "pavo"))

# Automatically colour-classify the coral snake patterns
snakes_class <- classify(snakes, kcols = 3)

# Run the adjacency analysis, with varying real-world scales for each image
snakes_adj <- adjacent(snakes_class, xpts = 120, xscale = c(50, 55))
```

---

aggplot *Plot aggregated reflectance spectra*

Description

Combines and plots spectra (by taking the average and the standard deviation, for example) according to an index or a vector of identities.

Usage

```
aggplot(
  rspecdata,
  by = NULL,
  FUN.center = mean,
  FUN.error = sd,
  lcol = NULL,
  shadecol = NULL,
  alpha = 0.2,
  legend = FALSE,
  ...
)
```

Arguments

`rspecdata`	(required) a data frame, possibly of class `rspec`, which contains a column containing a wavelength range, named 'wl', and spectra data in remaining columns.
`by`	(required) either a single value specifying the range of spectra within the data frame to be combined (for example, by = 3 indicates the function will be applied to groups of 3 consecutive columns in the spectra data frame); a vector containing identifications for the columns in the spectra data frame (in which case the function will be applied to each group of spectra sharing the same identification); or a list of vectors, e.g., by = `list(sex, species)`.
`FUN.center`	the function to be applied to the groups of spectra, calculating a measure of central tendency (defaults to `base::mean()`).
`FUN.error`	the function to be applied to the groups of spectra, calculating a measure of variation (defaults to `stats::sd()`).
`lcol`	colour of plotted lines indicating central tendency.
`shadecol`	colour of shaded areas indicating variance measure.
`alpha`	transparency of the shaded areas.
`legend`	automatically add a legend.
`...`	additional graphical parameters to be passed to plot.

Value

Plot containing the lines and shaded areas of the groups of spectra.

Author(s)

Rafael Maia <rm72@zips.uakron.edu>

Chad Eliason <cme16@zips.uakron.edu>

References

Montgomerie R (2006) Analyzing colors. In: Hill G, McGraw K (eds) Bird coloration. Harvard University Press, Cambridge, pp 90-147.

Examples

```
# Load reflectance data
data(sicalis)

# Create grouping variable based on spec names
bysic <- gsub("^ind[0-9].", "", names(sicalis)[-1])

# Plot using various error functions and options
aggplot(sicalis, bysic)
aggplot(sicalis, bysic, FUN.error = function(x) quantile(x, c(0.0275, 0.975)))
aggplot(sicalis, bysic, shadecol = spec2rgb(sicalis), lcol = 1)
aggplot(sicalis, bysic, lcol = 1, FUN.error = function(x) sd(x) / sqrt(length(x)))
```

---

aggspec *Aggregate reflectance spectra*

Description

Combines spectra (by taking the average, for example) according to an index or a vector of identities.

Usage

```
aggspec(rspecdata, by = NULL, FUN = mean, trim = TRUE)
```

Arguments

`rspecdata`	(required) a data frame, possibly of class `rspec`, which contains a column containing a wavelength range, named 'wl', and spectra data in remaining columns.
`by`	(required) either a single value specifying the range of spectra within the data frame to be combined (for example, by = 3 indicates the function will be applied to groups of 3 consecutive columns in the spectra data frame); a vector containing identifications for the columns in the spectra data frame (in which case the function will be applied to each group of spectra sharing the same identification); or a list of vectors, e.g., by = `list(sex, species)`.
`FUN`	the function to be applied to the groups of spectra. (defaults to `mean()`)
`trim`	logical. if `TRUE` (default), the function will try to identify and remove numbers at the end of the names of the columns in the new rspec object.

Value

A data frame of class `rspec` containing the spectra after applying the aggregating function.

Author(s)

Chad Eliason <cme16@zips.uakron.edu>

References

Montgomerie R (2006) Analyzing colors. In: Hill G, McGraw K (eds) Bird coloration. Harvard University Press, Cambridge, pp 90-147.

Examples

```
data(teal)

# Average every two spectra
teal.sset1 <- aggspec(teal, by = 2)
plot(teal.sset1)

# Create factor and average spectra by levels 'a' and 'b'
ind <- rep(c("a", "b"), times = 6)
teal.sset2 <- aggspec(teal, by = ind)

plot(teal.sset2)
```

---

as.rimg *Convert data to an rimg object*

Description

Converts an array of RGB values, a `cimg` object, or a `magick-image` object, to an `rimg` object.

Usage

```
as.rimg(object, name = "img")

## Default S3 method:
as.rimg(object, name = "img")

## S3 method for class 'cimg'
as.rimg(object, name = "img")

is.rimg(object)
```

Arguments

`object`	(required) a three-dimensional array containing RGB values.
`name`	the name(s) of the image(s).

Value

an object of class `rimg` for use in further `pavo` functions

a logical value indicating whether the object is of class `rimg`

Author(s)

Thomas E. White <thomas.white026@gmail.com>

Hugo Gruson <hugo.gruson+R@normalesup.org>

Examples

```
# Generate some fake image data
fake <- array(c(
  as.matrix(rep(c(0.2, 0.4, 0.6), each = 250)),
  as.matrix(rep(c(0.4, 0.7, 0.8), each = 250)),
  as.matrix(rep(c(0.6, 0.1, 0.2), each = 250))
),
dim = c(750, 750, 3)
)

# Inspect it
head(fake)

# Determine if is an rimg object
is.rimg(fake)

# Convert to rimg object and check again
fake2 <- as.rimg(fake)
is.rimg(fake2)
```

---

as.rspec *Convert data to an rspec object*

Description

Converts data frames or matrices containing spectral data to `rspec` object

Usage

```
as.rspec(
  object,
  whichwl = NULL,
  interp = TRUE,
  lim = NULL,
  exceed.range = TRUE
)

is.rspec(object)
```

Arguments

object	(required) a data frame or matrix containing spectra to process.
whichwl	a numeric or character vector specifying which column contains wavelengths. If NULL (default), function searches for column containing equally spaced numbers and sets it as wavelengths "wl". If no wavelengths are found or whichwl is not given, returns arbitrary index values.
interp	whether to interpolate wavelengths in 1-nm bins (defaults to TRUE). It is rarely recommended to turn off this option, as uninterpolated spectra are incompatible with some downstream analyses, including notably colour vision models.
lim	vector specifying wavelength range to interpolate over (e.g. c(300, 700)).
exceed.range	logical. Should data be interpolated to the limits specified by lim if lim exceeds the range of the actual data? Useful, and relatively safe, when the data range falls slightly within lim (e.g. 300.1 - 699 nm), but will produce spurious results if lim far exceeds the range of input data. Defaults to TRUE.

Value

an object of class rspec for use in further pavo functions

a logical value indicating whether the object is of class rspec

Author(s)

Chad Eliason <cme16@zips.uakron.edu>

Examples

```
# Generate some fake reflectance data
fakedat <- data.frame(wl = 300:700, refl1 = rnorm(401), refl2 = rnorm(401))
head(fakedat)

# Determine if is rspec object
is.rspec(fakedat)

# Convert to rspec object
fakedat2 <- as.rspec(fakedat)
is.rspec(fakedat2)
head(fakedat2)
```

---

axistetra *Plot reference axes in a static tetrahedral colourspace*

Description

Plots reference x, y and z arrows showing the direction of the axes in a static tetrahedral colourspace plot.

Usage

```
axistetra(
  x = 0,
  y = 1.3,
  size = 0.1,
  arrowhead = 0.05,
  col = par("fg"),
  lty = par("lty"),
  lwd = par("lwd"),
  label = TRUE,
  adj.label = list(x = c(0.003, 0), y = c(0.003, 0.003), z = c(0, 0.003)),
  label.cex = 1,
  label.col = NULL
)
```

Arguments

x, y	position of the legend relative to plot limits (usually a value between 0 and 1, but because of the perspective distortion, values greater than 1 are possible)
size	length of the arrows. Can be either a single value (applied for x, y and z) or a vector of 3 separate values for each axis.
arrowhead	size of the arrowhead.
col, lty, lwd	graphical parameters for the arrows.
label	logical, include x, y and z labels (defaults to TRUE).
adj.label	position adjustment for the labels. a list of 3 named objects for x, y and z arrows, each with 2 values for x and y adjustment.
label.cex, label.col	graphical parameters for the labels.

Value

axistetra adds reference arrows showing the direction of the 3-dimensional axes in a static tetrahedral colourspace plot.

Author(s)

Rafael Maia <rm72@zips.uakron.edu>

Examples

```
data(sicalis)
vis.sicalis <- vismodel(sicalis, visual = "avg.uv")
tcs.sicalis <- colspace(vis.sicalis, space = "tcs")
plot(tcs.sicalis)
axistetra()
```

`bootcoldist` *Bootstrap colour distance confidence intervals*

Description

Uses a bootstrap procedure to generate confidence intervals for the mean colour distance between two or more samples of colours

Usage

```
bootcoldist(vismodeldata, by, boot.n = 1000, alpha = 0.95, raw = FALSE, ...)
```

Arguments

`vismodeldata`	(required) quantum catch colour data. Can be the result from `vismodel()`, or `colspace()`. Data may also be independently calculated quantum catches, in the form of a data frame with columns representing photoreceptors.
`by`	(required) a numeric or character vector indicating the group to which each row from the object belongs to.
`boot.n`	number of bootstrap replicates (defaults to 1000)
`alpha`	the confidence level for the confidence intervals (defaults to 0.95)
`raw`	should the full set of bootstrapped distances (equal in length to boot.n) be returned, instead of the summary distances and CI's? Defaults to FALSE.
`...`	other arguments to be passed to `coldist()`. Must at minimum include `n` and `weber`. See `coldist()` for details.

Details

You can customise the type of parallel processing used by this function with the `future::plan()` function. This works on all operating systems, as well as high performance computing (HPC) environment. Similarly, you can customise the way progress is shown with the `progressr::handlers()` functions (progress bar, acoustic feedback, nothing, etc.)

Value

a matrix including the empirical mean and bootstrapped confidence limits for dS (and dL if `achromatic = TRUE`), or a data.frame of raw bootstraped dS (and dL if `achromatic = TRUE`) values equal in length to boot.n.

References

Maia, R., White, T. E., (2018) Comparing colors using visual models. Behavioral Ecology, ary017 doi:10.1093/beheco/ary017

Examples

```
# Run the receptor-noise limited model, using the visual phenotype
# of the blue tit
data(sicalis)
vm <- vismodel(sicalis, achromatic = "bt.dc", relative = FALSE)
gr <- gsub("ind..", "", rownames(vm))
bootcoldist(vm, by = gr, n = c(1, 2, 2, 4), weber = 0.1, weber.achro = 0.1)

# Run the same again, though as a simple colourspace model
data(sicalis)
vm <- vismodel(sicalis, achromatic = "bt.dc")
space <- colspace(vm)
gr <- gsub("ind..", "", rownames(space))
bootcoldist(space, by = gr)

# Estimate bootstrapped colour-distances for a more 'specialised' model,
# like the colour hexagon
data(flowers)
vis.flowers <- vismodel(flowers,
  visual = "apis", qcatch = "Ei", relative = FALSE,
  vonkries = TRUE, achromatic = "l", bkg = "green"
)
flowers.hex <- colspace(vis.flowers, space = "hexagon")
pop_group <- c(rep("pop_1", nrow(flowers.hex) / 2), rep("pop_2", nrow(flowers.hex) / 2))
bootcoldist(flowers.hex, by = pop_group)
```

---

classify *Identify colour classes in an image for adjacency analyses*

Description

Classify image pixels into discrete colour classes.

Usage

```
classify(
  imgdat,
  method = c("kMeans", "kMedoids"),
  kcols = NULL,
  refID = NULL,
  interactive = FALSE,
  plotnew = FALSE,
  col = "red",
  ...
)
```

Arguments

imgdat	(required) image data. Either a single image, or a series of images stored in a list. Preferably the result of getimg().
method	methods for image segmentation/classification. • 'kMeans': k-means clustering (default) • 'kMedoids': k-medoids clustering, using the partitioning-around-medoids ('pam') algorithm for large datasets.
kcols	the number of discrete colour classes present in the input image(s). Can be a single integer when only a single image is present, or if kcols is identical for all images. When passing a list of images, kcols can also be a vector the same length as imgdat, or a data.frame with two columns specifying image file names and corresponding kcols. This argument can optionally be disregarded when interactive = TRUE, and kcols will be inferred from the number of selections.
refID	either the numeric index or name of a 'reference' image, for use when passing a list of images. Other images will be k-means classified using centres identified in the single reference image, thus helping to ensure that homologous pattern elements will be reliably classified between images, if so desired.
interactive	interactively specify the colour-category 'centers', for k-means clustering. When TRUE, the user is asked to click a number of points (equal to kcols, if specified, otherwise user-determined) that represent the distinct colours of interest. If a reference image is specified, it will be the only image presented.
plotnew	Should plots be opened in a new window when interactive = TRUE? Defaults to FALSE.
col	the color of the marker points, when interactive = TRUE.
...	additional graphical parameters when interactive = TRUE. Also see graphics::par().

Details

You can customise the type of parallel processing used by this function with the future::plan() function. This works on all operating systems, as well as high performance computing (HPC) environment. Similarly, you can customise the way progress is shown with the progressr::handlers() functions (progress bar, acoustic feedback, nothing, etc.)

Value

A matrix, or list of matrices, of class rimg containing the colour class classifications ID at each pixel location. The RGB values corresponding to cluster centres (i.e. colour classes) are stored as object attributes.

Note

Since the kmeans process draws on random numbers to find initial cluster centres when interactive = FALSE, use set.seed() if reproducible cluster ID's are desired between runs.

Author(s)

Thomas E. White <thomas.white026@gmail.com>

See Also

stats::kmeans

Examples

```
# Single image
papilio <- getimg(system.file("testdata/images/butterflies/papilio.png", package = "pavo"))
papilio_class <- classify(papilio, kcols = 4)

# Multiple images, with interactive classification and a reference image
snakes <- getimg(system.file("testdata/images/snakes", package = "pavo"))
if (interactive()) {
  snakes_class <- classify(snakes, refID = "snake_01", interactive = TRUE)
}
```

---

coldist *Colour distances*

Description

Calculates colour distances. When data are the result of `vismodel()`, it applies the receptor-noise model of Vorobyev et al. (1998) to calculate colour distances with noise based on relative photoreceptor densities. It also accepts `colspace()` data in which case unweighted Euclidean distances, CIE2000 distances (cielab and cielch only), or Manhattan distances (coc model only) are returned.

Usage

```
coldist(
  modeldata,
  noise = c("neural", "quantum"),
  subset = NULL,
  achromatic = FALSE,
  qcatch = NULL,
  n = c(1, 2, 2, 4),
  weber = 0.1,
  weber.ref = "longest",
  weber.achro = 0.1
)
```

Arguments

`modeldata`	(required) quantum catch colour data. Can be the result from `vismodel()` for noise-weighted Euclidean distances, or `colspace()` for unweighted (typically) Euclidean distances. Data may also be independently calculated quantum catches, in the form of a data frame with columns representing photoreceptors.
`noise`	how the noise will be calculated (ignored for `colspace` objects):

- `neural` (default): noise is proportional to the Weber fraction and is independent of the intensity of the signal received (i.e. assumes bright conditions).
- `quantum`: noise is the sum of the neural noise and receptor noise, and is thus proportional to the Weber fraction and inversely proportional to the intensity of the signal received (the quantum catches). Note that the `quantum` option will only work with objects of class `vismodel`.

`subset` If only some of the comparisons should be returned, a character vector of length 1 or 2 can be provided, indicating which samples are desired. The subset vector must match the labels of the input samples, but partial matching (and regular expressions) are supported.

`achromatic` Logical. If `TRUE`, last column of the data frame is used to calculate the achromatic contrast, the form of which will depend on the input data and will be indicated by a message during execution. For noise-weighted distances, noise is based on the Weber fraction given by the argument `weber.achro`.

`qcatch` if the object is of class `vismodel` or `colspace`, this argument is ignored. If the object is a data frame of quantal catches from another source, this argument is used to specify what type of quantum catch is being used, so that the noise can be calculated accordingly: * `Qi`: Quantum catch for each photoreceptor * `fi`: Quantum catch according to Fechner's law (the signal of the receptor channel is proportional to the logarithm of the quantum catch)

`n` photoreceptor densities for the cones used in visual modeling. must have same length as number of columns (excluding achromatic receptor if used; defaults to the Pekin robin *Leiothrix lutea* densities: `c(1,2,2,4)`). Ignored for `colspace` objects.

`weber` The Weber fraction(s) to be used (often also referred to as receptor noise, or *e*). The noise-to-signal ratio `v` is unknown, and therefore must be calculated based on the empirically estimated Weber fraction of one or (more rarely) all the cone classes. When noise is only known for one receptor, as is typical, `v` is then applied to estimate the Weber fraction of the other cones. By default, the value of 0.1 is used (the empirically estimated value for the LWS cone from *Leiothrix lutea*). See Olsson et al. 2017 for a review of published values in the literature. Ignored for `colspace` objects.

`weber.ref` the cone class used to obtain the empirical estimate of the Weber fraction used for the `weber` argument, if a single value is specified. By default, `n4` is used, representing the LWS cone for *Leiothrix lutea*. Ignored for `colspace` objects.

`weber.achro` the Weber fraction to be used to calculate achromatic contrast, when `achromatic = TRUE`. Defaults to 0.1. Ignored for `colspace` objects.

Value

A data frame containing up to 4 columns. The first two (`patch1, patch2`) refer to the two colors being contrasted; `dS` is the chromatic contrast (delta S) and `dL` is the achromatic contrast (delta L). Units of `dS` JND's in the receptor-noise model, unweighted Euclidean distances in colorspace models, and Manhattan distances in the colour-opponent-coding space. Units of `dL` vary, and are either simple contrast, Weber contrast, or Michelson contrast, as indicated by the output message.

Note on previous versions

Generic di- tri- and tetra-chromatic `colspace` objects were previously passed through the receptor-noise limited model to return noise-weighted Euclidean distances. This behaviour has been amended, and generic spaces now return unweighted Euclidean distances. Equivalent results to the former behaviour can be attained by sending the results of `vismodel()` directly to `coldist()` , as previously, which also offers greater flexibility and reliability. Thus `coldist()` now returns unweighted Euclidean distances for `colspace` objects (with the exception of Manhattan distances for the coc space, and CIE2000, distances for CIELab and CIELCh spaces), and noise-weighted Euclidean distances for `vismodel` objects.

Author(s)

Thomas E. White <thomas.white026@gmail.com>

Rafael Maia <rm72@zips.uakron.edu>

References

Vorobyev, M., Osorio, D., Bennett, A., Marshall, N., & Cuthill, I. (1998). Tetrachromacy, oil droplets and bird plumage colours. Journal Of Comparative Physiology A-Neuroethology Sensory Neural And Behavioral Physiology, 183(5), 621-633.

Hart, N. S. (2001). The visual ecology of avian photoreceptors. Progress In Retinal And Eye Research, 20(5), 675-703.

Endler, J. A., & Mielke, P. (2005). Comparing entire colour patterns as birds see them. Biological Journal Of The Linnean Society, 86(4), 405-431.

Olsson, P., Lind, O., & Kelber, A. (2015) Bird colour vision: behavioural thresholds reveal receptor noise. Journal of Experimental Biology, 218, 184-193.

Lind, O. (2016) Colour vision and background adaptation in a passerine bird, the zebra finch (Taeniopygia guttata). Royal Society Open Science, 3, 160383.

Olsson, P., Lind, O., & Kelber, A. (2017) Chromatic and achromatic vision: parameter choice and limitations for reliable model predictions. Behavioral Ecology, doi:10.1093/beheco/arx133

Examples

```
# Dichromat
data(flowers)
vis.flowers <- vismodel(flowers, visual = "canis", relative = FALSE)
didist.flowers <- coldist(vis.flowers, n = c(1, 2))

# Trichromat
vis.flowers <- vismodel(flowers, visual = "apis", relative = FALSE)
tridist.flowers <- coldist(vis.flowers, n = c(1, 2, 1))

# Trichromat, colour-hexagon model (euclidean distances)
vis.flowers <- vismodel(flowers,
  visual = "apis", qcatch = "Ei",
  relative = FALSE, vonkries = TRUE, achromatic = "l", bkg = "green"
)
hex.flowers <- colspace(vis.flowers, space = "hexagon")
```

```
hexdist.flowers <- coldist(hex.flowers)

# Trichromat, colour-opponent-coding model (manhattan distances)
vis.flowers <- vismodel(flowers, visual = "apis", qcatch = "Ei", relative = FALSE, vonkries = TRUE)
coc.flowers <- colspace(vis.flowers, space = "coc")
hexdist.flowers <- coldist(coc.flowers)

# Tetrachromat
data(sicalis)
vis.sicalis <- vismodel(sicalis, visual = "avg.uv", relative = FALSE)
tetradist.sicalis.n <- coldist(vis.sicalis)
```

`colspace` *Model spectra in a colorspace*

Description

Models reflectance spectra in a colorspace. For information on plotting arguments and graphical parameters, see `plot.colspace()`.

Usage

```
colspace(
  vismodeldata,
  space = c("auto", "di", "tri", "tcs", "hexagon", "coc", "categorical", "ciexyz",
    "cielab", "cielch", "segment"),
  qcatch = NULL,
  ...
)
```

Arguments

`vismodeldata`	(required) quantum catch color data. Can be either the result from `vismodel()` or independently calculated data (in the form of a data frame with columns representing quantum catches).
`space`	Which colorspace/model to use. Options are: • `auto`: if data is a result from `vismodel()`, applies `di`, `tri` or `tcs` if input visual model had two, three or four cones, respectively. • `di`: dichromatic colourspace. See `dispace()` for details. (plotting arguments) • `tri`: trichromatic colourspace (i.e. Maxwell triangle). See `trispace()` for details. (plotting arguments) • `tcs`: tetrahedral colourspace. See `tcspace()` for details. (plotting arguments)

- hexagon: the trichromatic colour-hexagon of Chittka (1992). See hexagon() for details. (plotting arguments)
- coc: the trichromatic colour-opponent-coding model of Backhaus (1991). See coc() for details. (plotting arguments)
- categorical: the tetrachromatic categorical fly-model of Troje (1993). See categorical() for details. (plotting arguments)
- ciexyz: CIEXYZ space. See cie() for details. (plotting arguments)
- cielab: CIELAB space. See cie() for details. (plotting arguments)
- cielch: CIELCh space. See cie() for details. (plotting arguments)
- segment: segment analysis of Endler (1990). See segspace() for details. (plotting arguments)

qcatch — Which quantal catch metric is being inputted. Only used when input data is NOT an output from vismodel(). Must be Qi, fi or Ei.

... — additional arguments passed to cie() for non vismodel() data.

Author(s)

Rafael Maia <rm72@zips.uakron.edu>

Thomas White <thomas.white026@gmail.com>

References

Smith T, Guild J. (1932) The CIE colorimetric standards and their use. Transactions of the Optical Society, 33(3), 73-134.

Westland S, Ripamonti C, Cheung V. (2012). Computational colour science using MATLAB. John Wiley & Sons.

Chittka L. (1992). The colour hexagon: a chromaticity diagram based on photoreceptor excitations as a generalized representation of colour opponency. Journal of Comparative Physiology A, 170(5), 533-543.

Chittka L, Shmida A, Troje N, Menzel R. (1994). Ultraviolet as a component of flower reflections, and the colour perception of Hymenoptera. Vision research, 34(11), 1489-1508.

Troje N. (1993). Spectral categories in the learning behaviour of blowflies. Zeitschrift fur Naturforschung C, 48, 96-96.

Stoddard, M. C., & Prum, R. O. (2008). Evolution of avian plumage color in a tetrahedral color space: A phylogenetic analysis of new world buntings. The American Naturalist, 171(6), 755-776.

Endler, J. A., & Mielke, P. (2005). Comparing entire colour patterns as birds see them. Biological Journal Of The Linnean Society, 86(4), 405-431.

Kelber A, Vorobyev M, Osorio D. (2003). Animal colour vision - behavioural tests and physiological concepts. Biological Reviews, 78, 81 - 118.

Backhaus W. (1991). Color opponent coding in the visual system of the honeybee. Vision Research, 31, 1381-1397.

Endler, J. A. (1990) On the measurement and classification of color in studies of animal color patterns. Biological Journal of the Linnean Society, 41, 315-352.

Examples

```
data(flowers)

# Model a dichromat viewer in a segment colourspace
vis.flowers <- vismodel(flowers, visual = "canis")
di.flowers <- colspace(vis.flowers, space = "di")

# Model a honeybee viewer in the colour hexagon
vis.flowers <- vismodel(flowers,
  visual = "apis", qcatch = "Ei", relative = FALSE,
  vonkries = TRUE, achromatic = "l", bkg = "green"
)
hex.flowers <- colspace(vis.flowers, space = "hexagon")

# Model a trichromat (the honeybee) in a Maxwell triangle
vis.flowers <- vismodel(flowers, visual = "apis")
tri.flowers <- colspace(vis.flowers, space = "tri")
plot(tri.flowers)

# Model a tetrachromat (the Blue Tit) in a tetrahedral colourspace
vis.flowers <- vismodel(flowers, visual = "bluetit")
tcs.flowers <- colspace(vis.flowers, space = "tcs")

# Model a housefly in the 'categorical' colourspace
vis.flowers <- vismodel(flowers, visual = "musca", achro = "md.r1")
cat.flowers <- colspace(vis.flowers, space = "categorical")
```

---

explorespec *Plot spectral curves*

Description

Plots one or multiple spectral curves in the same graph to rapidly compare groups of spectra.

Usage

```
explorespec(
  rspecdata,
  by = NULL,
  scale = c("equal", "free"),
  legpos = "topright",
  ...
)
```

Arguments

`rspecdata`	(required) a data frame, possibly of class `rspec`, which contains a column containing a wavelength range, named 'wl', and spectra data in remaining columns.

by	number of spectra to include in each graph (defaults to 1)
scale	defines how the y-axis should be scaled. "free": panels can vary in the range of the y-axis; "equal": all panels have the y-axis with the same range.
legpos	legend position control. Either a vector containing x and y coordinates or a single keyword from the list: "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" and "center".
...	additional parameters to be passed to plot

Value

Spectral curve plots

Note

Number of plots presented per page depends on the number of graphs produced.

Author(s)

Pierre-Paul Bitton <bittonp@uwindsor.ca>

Examples

```
data(sicalis)
explorespec(sicalis, 3)
explorespec(sicalis, 3, ylim = c(0, 100), legpos = c(500, 80))
```

---

flowers *Reflectance spectra from a suite of native Australian flowers, collected around Cairns, Queensland.*

Description

Dataset containing reflectance measurements from 36 native Australian angiosperm species, indicated by column names.

Usage

```
data(flowers)
```

Format

An object of class rspec (inherits from data.frame) with 401 rows and 37 columns.

Author(s)

Thomas White <thomas.white026@gmail.com>

References

Dalrymple, R., L., Kemp, D. J., Flores-Moreno, H., Laffan, S. W., White, T. E., Hemmings, F. A., Tindall, M. L., & Moles, A. T. (2015). Birds, butterflies and flowers in the tropics are not more colourful than those at higher latitudes. Global Ecology and Biogeography, 24(12), 1424-1432. doi:10.1111/geb.12368

White, T. E., Dalrymple, R. L., Herberstein, M. E., & Kemp, D. J. (2017). The perceptual similarity of orb-spider prey lures and flowers colours. Evolutionary Ecology, 31(1), 1-20. doi:10.1007/s106820169876x

Dalrymple, R. L., Flores-Moreno, H., Kemp, D. J., White, T. E., Laffan, S. W., Hemmings, F. A., Tindall, M. L., & Moles, A. T. (2018). Abiotic and biotic predictors of macroecological patterns in bird and butterfly coloration. Ecological Monographs, 88(2), 204-224.

`getimg` *Import image data*

Description

Finds and imports PNG, JPEG, and/or BMP images.

Usage

```
getimg(imgpath = getwd(), subdir = FALSE, subdir.names = FALSE, max.size = 1)
```

Arguments

`imgpath`	(required) either the full file-path or URL to an image (including extension), or the path to a folder in which multiple image files are located. Mixed file formats within a folder are accepted.
`subdir`	should subdirectories within the `imgpath` folder be included in the search? (defaults to `FALSE`).
`subdir.names`	should subdirectory path be included in the name of the images? (defaults to `FALSE`).
`max.size`	maximum size of all images to be allowed in memory, in GB. Defaults to `1`.

Value

a image, or list of images, of class `rimg`, for use in further `pavo` functions.

Author(s)

Thomas E. White <thomas.white026@gmail.com>

Examples

```
# Single image
papilio <- getimg(system.file("testdata/images/butterflies/papilio.png", package = "pavo"))

# Multiple images
snakes <- getimg(system.file("testdata/images/snakes", package = "pavo"))
```

`getspec` *Import spectra files*

Description

Finds and imports spectra files from a folder. Currently works for reflectance files generated in Ocean Optics SpectraSuite (USB2000, USB4000 and Jaz spectrometers), CRAIC software (after exporting) and Avantes (before or after exporting).

Usage

```
getspec(
  where = getwd(),
  ext = "txt",
  lim = c(300, 700),
  decimal = ".",
  sep = NULL,
  subdir = FALSE,
  subdir.names = FALSE,
  ignore.case = TRUE
)
```

Arguments

`where`	Folder in which files are located (defaults to current working directory).
`ext`	File extension to be searched for, without the "." (defaults to `txt`). You can also use a character vector to specify multiple file extensions.
`lim`	A vector with two numbers determining the wavelength limits to be considered (defaults to `c(300, 700)`).
`decimal`	Character to be used to identify decimal plates (defaults to `.`).
`sep`	Column delimiting characters to be considered in addition to the default (which are: tab, space, and ";")
`subdir`	Should subdirectories within the `where` folder be included in the search? (defaults to `FALSE`).
`subdir.names`	Should subdirectory path be included in the name of the spectra? (defaults to `FALSE`).
`ignore.case`	Should the extension search be case insensitive? (defaults to `TRUE`)

Details

You can customise the type of parallel processing used by this function with the `future::plan()` function. This works on all operating systems, as well as high performance computing (HPC) environment. Similarly, you can customise the way progress is shown with the `progressr::handlers()` functions (progress bar, acoustic feedback, nothing, etc.)

Value

A data frame, of class `rspec`, containing individual imported spectral files as columns. Reflectance values are interpolated to the nearest wavelength integer.

Author(s)

Rafael Maia <rm72@zips.uakron.edu>

Hugo Gruson <hugo.gruson+R@normalesup.org>

References

Gruson H, White TE, Maia R (2019) lightr: import spectral data and metadata in R. Journal of Open Source Software, 4(43), 1857, doi:10.21105/joss.01857.

See Also

`lightr::lr_get_spec()` for a more flexible version of this function (e.g. uninterpolated wavelengths), and `lightr::lr_get_metadata()` for the retrieval and import of spectral metadata. See https://docs.ropensci.org/lightr/ for the complete, and up-to-date, list of supported file formats.

Examples

```
# Import and inspect example spectral data with a range of set to 400-700nm.
rspecdata <- getspec(system.file("testdata", package = "lightr"), ext = "ttt", lim = c(400, 700))
head(rspecdata)
```

---

img_conversion *Convert images between class rimg and cimg or magick-image*

Description

Conveniently convert single objects of class `rimg` to class `cimg` (from the package `imager`) or `magick-image` (from the package `magick`), both of which contains a suite of useful image-processing capabilities.

Usage

```
rimg2cimg(image)

rimg2magick(image)
```

Arguments

image	an object of class rimg

Value

an image of the specified class

Note

Attributes (e.g. scales, color-classes) will not be preserved following conversion from class rimg, so it's best to use early in the analysis workflow.

Author(s)

Thomas E. White <thomas.white026@gmail.com>

Hugo Gruson <hugo.gruson+R@normalesup.org>

Examples

```
papilio <- getimg(system.file("testdata/images/butterflies/papilio.png", package = "pavo"))

# Convert from class rimg to cimg
if (requireNamespace("imager", quiety = TRUE)) {
  papilio_cimg <- rimg2cimg(papilio)
  class(papilio_cimg)
}


# Convert from class rimg to magick-image
papilio_magick <- rimg2magick(papilio)
class(papilio_magick)
```

---

irrad2flux *Converts between irradiance and photon (quantum) flux*

Description

Some spectrometers will give illuminant values in units of irradiance ($\mu Watt.cm^{-2}$), but physiological models require illuminants in units of photon (quantum) flux ($\mu mol.s^{-1}.m^{-2}$). The functions irrad2flux() and flux2irrad() allows for easy conversion of rspec objects between these units.

Usage

```
irrad2flux(rspecdata)

flux2irrad(rspecdata)
```

Arguments

rspecdata	(required) a rspec object containing illuminant values.

Value

a converted rspec object.

Author(s)

Rafael Maia <rm72@zips.uakron.edu>

---

is.colspace *Test if object is of class 'colspace'*

Description

Test if object is of class 'colspace'

Usage

```
is.colspace(object)
```

Arguments

object	an R object

Value

a logical value indicating whether the object is of class colspace

See Also

colspace()

is.vismodel *Test if object is of class 'vismodel'*

Description

Test if object is of class 'vismodel'

Usage

```
is.vismodel(object)
```

Arguments

object	an R object

Value

a logical value indicating whether the object is of class `vismodel`.

See Also

`vismodel()`

jnd2xyz *Convert JND distances into perceptually-corrected Cartesian coordinates*

Description

Converts a `coldist()` output into Cartesian coordinates that are perceptually-corrected (i.e. noise-weighted Euclidean distances)

Usage

```
jnd2xyz(
  coldistres,
  center = TRUE,
  rotate = TRUE,
  rotcenter = c("mean", "achro"),
  ref1 = "l",
  ref2 = "u",
  axis1 = c(1, 1, 0),
  axis2 = c(0, 0, 1)
)
```

Arguments

coldistres	(required) the output from a coldist() call.
center	logical indicating if the data should be centered on its centroid (defaults to TRUE).
rotate	logical indicating if the data should be rotated (defaults to TRUE).
rotcenter	should the vectors for rotation be centered in the achromatic center ("achro") or the data centroid ("mean", the default)?
ref1	the cone to be used as a the first reference. May be NULL (for no first rotation in the 3-dimensional case) or must match name in the original data that was used for coldist(). Defaults to 'l'.
ref2	the cone to be used as a the second reference. May be NULL (for no first rotation in the 3-dimensional case) or must match name in the original data that was used for coldist(). Defaults to 'u'. (only used if data has 3 dimensions).
axis1	A vector of length 3 composed of 0's and 1's, with 1's representing the axes (x, y, z) to rotate around. Defaults to c(1, 1, 0), such that the rotation aligns with the xy plane (only used if data has 2 or 3 dimensions). Ignored if ref1 is NULL (in 3-dimensional case only)
axis2	A vector of length 3 composed of 0's and 1's, with 1's representing the axes (x, y, z) to rotate around. Defaults to c(0, 0, 1), such that the rotation aligns with the z axis (only used if data has 3 dimensions). Ignored if ref2 is NULL (in 3-dimensional case only)

Author(s)

Rafael Maia <rm72@zips.uakron.edu>

References

Pike, T.W. (2012). Preserving perceptual distances in chromaticity diagrams. Behavioral Ecology, 23, 723-728.

Maia, R., White, T. E., (2018) Comparing colors using visual models. Behavioral Ecology, ary017 doi:10.1093/beheco/ary017

Examples

```
# Load floral reflectance spectra
data(flowers)

# Estimate quantum catches visual phenotype of a Blue Tit
vis.flowers <- vismodel(flowers, visual = 'bluetit')

# Estimate noise-weighted colour distances between all flowers
cd.flowers <- coldist(vis.flowers)

# Convert points to Cartesian coordinates in which Euclidean distances are
# noise-weighted.
jnd2xyz(cd.flowers)
```

`legendtetra` *Add legend to a static tetrahedral colourspace*

Description

Adds a legend to a static tetrahedral colourspace plot.

Usage

```
legendtetra(x = 0.8, y = 1.2, ...)
```

Arguments

x, y	position of the legend relative to plot limits (usually a value between 0 and 1, but because of the perspective distortion, values greater than 1 are possible)
...	additional arguments passed to `legend()`.

Value

`legendtetra()` adds a legend to a static tetrahedral colourspace plot. for additional information on which arguments are necessary and how they are used, see `legend()`.

Author(s)

Rafael Maia <rm72@zips.uakron.edu>

Examples

```
data(sicalis)

vis_sicalis <- vismodel(sicalis)
tcs_sicalis <- colspace(vis_sicalis)

cols <- c("#1B9E77", "#D95F02", "#7570B3")
plot(tcs_sicalis, col = cols)
legendtetra(
  legend = c("Crown", "Throat", "Breast"),
  col = cols, pch = 16
)
```

`merge.rspec` *Merge two rspec objects*

Description

Merges two `rspec` objects into a single `rspec` object.

Usage

```
## S3 method for class 'rspec'
merge(x, y, ...)
```

Arguments

`x, y`	(required) `rspec` objects to merge.
`...`	additional class arguments.

Value

an object of class `rspec` for use with `pavo` functions. Will use `by = "wl"` if unspecified, or automatically append `wl` to the `by` argument if one is specified.

Author(s)

Chad Eliason <cme16@zips.uakron.edu>

See Also

`as.rspec()`, `aggspec()`

Examples

```
# Load angle-resolved reflectance data for a green-winged teal, and
# split it in two
data(teal)
teal1 <- teal[, c(1, 2:5)]
teal2 <- teal[, c(1, 6:13)]

# Merge the two split datasets back into one, with a shared 'wl' column
teal.mer <- merge(teal1, teal2, by = "wl")

# Examine the results, and compare the original to the (identical)
# reconstructed version
plot(teal.mer)
plot(teal)
identical(teal.mer, teal)

# Or an equivalent method, which also allows for the merging of more than one rspec
# object at a time (simply add further objects to the list())
```

```
teal.mer2 <- do.call(merge, list(teal1, teal2))

# Check equivalence
identical(teal.mer2, teal)
```

---

peakshape *Peak shape descriptors*

Description

Calculates height, location and width of peak at the reflectance midpoint (FWHM). Note: bounds should be set wide enough to incorporate all minima in spectra. Smoothing spectra using `procspec()` is also recommended.

Usage

```
peakshape(
  rspecdata,
  select = NULL,
  lim = NULL,
  plot = TRUE,
  ask = FALSE,
  absolute.min = FALSE,
  ...
)
```

Arguments

`rspecdata`	(required) a data frame, possibly of class `rspec`, which contains a column containing a wavelength range, named 'wl', and spectra data in remaining columns.
`select`	specification of which spectra to plot. Can be a numeric vector or factor (e.g., `sex == "male"`)
`lim`	a vector specifying the wavelength range to analyze.
`plot`	logical. Should plots indicating calculated parameters be returned? (Defaults to `TRUE`).
`ask`	logical, specifies whether user input needed to plot multiple plots when number of spectra to analyze is greater than 1 (defaults to `FALSE`).
`absolute.min`	logical. If `TRUE`, full width at half maximum will be calculated using the absolute minimum reflectance of the spectrum, even if that value falls outside the range specified by `lim`. (defaults to `FALSE`)
`...`	additional arguments to be passed to plot.

Value

a data frame containing column names (id); peak height (max value, B3), location (hue, H1) and full width at half maximum (FWHM), as well as half widths on left (HWHM.l) and right side of peak (HWHM.r). Incl.min column indicates whether user-defined bounds incorporate the actual minima of the spectra. Function will return a warning if not.

Author(s)

Chad Eliason <cme16@zips.uakron.edu>

Rafael Maia <rm72@zips.uakron.edu>

Hugo Gruson <hugo.gruson+R@normalesup.org>

See Also

procspec()

Examples

```
data(teal)

peakshape(teal, select = 3)
peakshape(teal, select = 10)

# Use wavelength bounds to narrow in on peak of interest
peakshape(teal, select = 10, lim = c(400, 550))
```

---

plot.colspace *Plot spectra in a colourspace*

Description

Plots reflectance spectra in the appropriate colourspace.

Usage

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

Arguments

x	(required) an object of class colspace.
...	additional graphical options, which vary by modeled space. Refer to their individual documentation: • diplot(): dichromat space • triplot(): trichromat space • tetraplot(): tetrahedral space

- `catplot()`: categorical space
- `hexplot()`: colour hexagon
- `cocplot()`: colour-opponent-coding space
- `cieplot()`: cie spaces
- `segplot()`: segment analysis space
- `jndplot()`: perceptual, 'noise corrected' space (for the results of `jnd2xyz()`)

Also see `par()`.

Value

A colourspace plot appropriate to the input data.

Author(s)

Rafael Maia <rm72@zips.uakron.edu>

Thomas White <thomas.white026@gmail.com>

Chad Eliason <cme16@zips.uakron.edu>

References

Smith T, Guild J. (1932) The CIE colorimetric standards and their use. Transactions of the Optical Society, 33(3), 73-134.

Westland S, Ripamonti C, Cheung V. (2012). Computational colour science using MATLAB. John Wiley & Sons.

Chittka L. (1992). The colour hexagon: a chromaticity diagram based on photoreceptor excitations as a generalized representation of colour opponency. Journal of Comparative Physiology A, 170(5), 533-543.

Chittka L, Shmida A, Troje N, Menzel R. (1994). Ultraviolet as a component of flower reflections, and the colour perception of Hymenoptera. Vision research, 34(11), 1489-1508.

Troje N. (1993). Spectral categories in the learning behaviour of blowflies. Zeitschrift fur Naturforschung C, 48, 96-96.

Stoddard, M. C., & Prum, R. O. (2008). Evolution of avian plumage color in a tetrahedral color space: A phylogenetic analysis of new world buntings. The American Naturalist, 171(6), 755-776.

Endler, J. A., & Mielke, P. (2005). Comparing entire colour patterns as birds see them. Biological Journal Of The Linnean Society, 86(4), 405-431.

Kelber A, Vorobyev M, Osorio D. (2003). Animal colour vision - behavioural tests and physiological concepts. Biological Reviews, 78, 81 - 118.

Backhaus W. (1991). Color opponent coding in the visual system of the honeybee. Vision Research, 31, 1381-1397.

Endler, J. A. (1990) On the measurement and classification of color in studies of animal color patterns. Biological Journal of the Linnean Society, 41, 315-352.

See Also

`plot()`, `points.colspace()`

Examples

```
data(flowers)
data(sicalis)

# Dichromat
vis.flowers <- vismodel(flowers, visual = "canis")
di.flowers <- colspace(vis.flowers, space = "di")
plot(di.flowers)

# Colour hexagon
vis.flowers <- vismodel(flowers,
  visual = "apis", qcatch = "Ei", relative = FALSE,
  vonkries = TRUE, achromatic = "l", bkg = "green"
)
hex.flowers <- colspace(vis.flowers, space = "hexagon")
plot(hex.flowers, sectors = "coarse")

# Tetrahedron (static)
vis.sicalis <- vismodel(sicalis, visual = "avg.uv")
tcs.sicalis <- colspace(vis.sicalis, space = "tcs")
plot(tcs.sicalis)


# Tetrahedron (interactive)
vis.sicalis <- vismodel(sicalis, visual = "avg.uv")
tcs.sicalis <- colspace(vis.sicalis, space = "tcs")
tcsplot(tcs.sicalis, size = 0.005)

## Add points to interactive tetrahedron
patch <- rep(c("C", "T", "B"), 7)
tcs.crown <- subset(tcs.sicalis, "C")
tcs.breast <- subset(tcs.sicalis, "B")
tcsplot(tcs.crown, col = "blue")
tcspoints(tcs.breast, col = "red")

## Plot convex hull in interactive tetrahedron
tcsplot(tcs.sicalis, col = "blue", size = 0.005)
tcsvol(tcs.sicalis)
```

---

plot.rimg *Plot unprocessed or colour-classified images*

Description

Plot unprocessed or colour-classified image data. If the images are in a list, they will be stepped through one by one.

Usage

```
## S3 method for class 'rimg'
plot(x, axes = TRUE, col = NULL, ...)
```

Arguments

x	(required) an image of class rimg, or list thereof.
axes	should axes be drawn? (defaults to TRUE)
col	optional vector of colours when plotting colour-classified images. Defaults to the mean RGB values of the k-means centres (i.e. the average 'original' colours).
...	additional graphical parameters. Also see par().

Value

an image plot.

Author(s)

Thomas E. White <thomas.white026@gmail.com>

Examples

```
papilio <- getimg(system.file("testdata/images/butterflies/papilio.png", package = "pavo"))
plot(papilio)

papilio_class <- classify(papilio, kcols = 4)
plot(papilio_class)


# Multiple images
snakes <- getimg(system.file("testdata/images/snakes", package = "pavo"))
plot(snakes)

snakes_class <- classify(snakes, kcols = 3)
plot(snakes_class)
```

---

plot.rspec *Plot spectra*

Description

Plots reflectance spectra in different arrangements.

Usage

```
## S3 method for class 'rspec'
plot(
  x,
  select = NULL,
  type = c("overlay", "stack", "heatmap"),
  varying = NULL,
  n = 100,
  labels = FALSE,
  labels.stack = NULL,
  labels.cex = 1,
  wl.guide = TRUE,
  ...
)
```

Arguments

x	(required) a data frame, possibly an object of class rspec, with a column with wavelength data, named 'wl', and the remaining column containing spectra to plot.
select	specification of which spectra to plot. Can be a numeric vector or factor (e.g., sex == "male")
type	what type of plot should be drawn. Possibilities are: • overlay (default) for plotting multiple spectra in a single panel with a common y-axis. • stack for plotting multiple spectra in a vertical arrangement. • heatmap for plotting reflectance values by wavelength and a third variable (varying).
varying	a numeric vector giving values for y-axis in type = "heatmap".
n	number of bins with which to interpolate colors and varying for the heatplot.
labels	logical. Add labels identifying each spectrum to the outer plot margin? Defaults to FALSE. Ignored when type = 'heatmap'.
labels.stack	a vector of labels for spectra when labels = TRUE. Defaults to the column names from spectral data. Note you will likely want to adjust the plot margins to accommodate the text labels. See ?par() for guidance on setting margins.
labels.cex	size of the text labels when labels = TRUE.
wl.guide	logical determining whether visible light spectrum should be added to the x-axis.
...	additional arguments passed to plot() (or image() for "heatmap").

Author(s)

Thomas White <thomas.white026@gmail.com>

Hugo Gruson <hugo.gruson+R@normalesup.org>

Chad Eliason <cme16@zips.uakron.edu>

See Also

spec2rgb(), image(), plot()

Examples

```
# Load angle-resolved reflectance data for a green-winged teal
data(teal)

# Create an overlay plot (default)
plot(teal)

# Create an stacked spectral plot
plot(teal, type = "stack")

# Create a reflectance heatmap
plot(teal, type = "heatmap")
```

---

plot.sensmod *Plot absorbance spectra from* sensmodel()

Description

Plot absorbance spectra from sensmodel()

Usage

```
## S3 method for class 'sensmod'
plot(
  x,
  select = NULL,
  type = c("overlay", "stack", "heatmap"),
  varying = NULL,
  n = 100,
  labels = FALSE,
  labels.stack = NULL,
  labels.cex = 1,
  wl.guide = TRUE,
  ...
)
```

Arguments

x	(required) a data frame, possibly an object of class rspec, with a column with wavelength data, named 'wl', and the remaining column containing spectra to plot.
select	specification of which spectra to plot. Can be a numeric vector or factor (e.g., sex == "male")

type	what type of plot should be drawn. Possibilities are: • overlay (default) for plotting multiple spectra in a single panel with a common y-axis. • stack for plotting multiple spectra in a vertical arrangement. • heatmap for plotting reflectance values by wavelength and a third variable (varying).
varying	a numeric vector giving values for y-axis in type = "heatmap".
n	number of bins with which to interpolate colors and varying for the heatplot.
labels	logical. Add labels identifying each spectrum to the outer plot margin? Defaults to FALSE. Ignored when type = 'heatmap'.
labels.stack	a vector of labels for spectra when labels = TRUE. Defaults to the column names from spectral data. Note you will likely want to adjust the plot margins to accommodate the text labels. See ?par() for guidance on setting margins.
labels.cex	size of the text labels when labels = TRUE.
wl.guide	logical determining whether visible light spectrum should be added to the x-axis.
...	additional arguments passed to plot() (or image() for "heatmap").

See Also

plot.rspec(), sensmodel()

Examples

```
# Blue tit visual system based on Hart et al (2000)
bluesens <- sensmodel(c(371, 448, 502, 563),
  beta = FALSE,
  lambdacut = c(330, 413, 507, 572),
  oiltype = c("T", "C", "Y", "R"), om = TRUE
)
plot(bluesens)

# Alternatively, you can specify your own ylab
plot(bluesens, ylab = "absor.")
```

---

plotsmooth *Plot loess smoothed curves*

Description

Plots spectral curves with various levels of loess smoothing to help decide which loess parameters are best for subsequently smoothing the data (e.g. via procspec()).

Usage

```
plotsmooth(
  rspecdata,
  minsmooth = 0.05,
  maxsmooth = 0.2,
  curves = 5,
  specnum = "ALL",
  ask = TRUE
)
```

Arguments

rspecdata	(required) a data frame, possibly of class rspec, which contains a column containing a wavelength range, named 'wl', and spectra data in remaining columns.
minsmooth	the minimum f value of the loess function to visualize (defaults to 0.05).
maxsmooth	the maximum f value of the loess function to visualize (defaults to 0.20).
curves	the number of curves to display on the same plot (defaults to 5).
specnum	the number of spectral curves, from the data frame, to visualize (defaults to ALL).
ask	logical. if TRUE, asks for user input before changing plot pages

Value

Series of plot with curves processed with varying level of loess smoothing

Author(s)

Pierre-Paul Bitton <bittonp@uwindsor.ca>

See Also

procspec()

Examples

```
# Load reflectance spectra
data(sicalis)

# Visualise the spectral reflectance curves across a range of smoothing levels
plotsmooth(sicalis, minsmooth = 0.05, maxsmooth = 0.1, curves = 7, specnum = 6)
```

points.colspace *Plot points in a colourspace*

Description

Add points to a colourspace plot

Usage

```
## S3 method for class 'colspace'
points(x, ...)
```

Arguments

x	(required) an object of class colspace.
...	additional graphical options. See par().

Value

points.colspace adds points to a colourspace plot.

Author(s)

Rafael Maia <rm72@zips.uakron.edu>

Thomas White <thomas.white026@gmail.com>

See Also

plot.colspace()

procimg *Process images*

Description

Specify scales, resize, and/or define focal objects within images.

Usage

```
procimg(
  image,
  resize = NULL,
  rotate = NULL,
  scaledist = NULL,
  outline = FALSE,
  reclass = NULL,
  smooth = FALSE,
  iterations = 1L,
  col = "red",
  obj_dist = NULL,
  obj_width = NULL,
  eye_res = NULL,
  plotnew = FALSE,
  ...
)
```

Arguments

image	(required) image data. Either a single image array, or a number of images stored in a list. Preferably the result of getimg().
resize	an integer specifying a percentage for resizing images, if so desired. E.g. 50 to half the size of an image, 200 to double it.
rotate	an integer specifying the angle of image rotation, in degrees. Images are rotated around the centre, and linearly interpolated.
scaledist	an integer, or numeric vector equal in length to the number of images, specifying the length of the scale in the image(s). Image(s) will then be presented, and the user asked to select either end of the scale corresponding to the input value.
outline	interactively specify the focal object in an image by clicking around its outline. The xy-coordinates of the resulting closed polygon are saved as an attribute, for use in generating a masking layer & separating animals/plants from backgrounds in further analyses. This is particularly useful when backgrounds are complex, such as in natural settings.
reclass	interactively specify an area on a colour-classified image that is to be reclassified as the numeric value provided. e.g. when reclass = 1, the user will be asked to select a polygon on the image, within which all colour-category values will be changes to 1.
smooth	should the polygon specified when outline = TRUE be smoothed using Chaikin's corner-cuting algorithm? Defaults to FALSE.
iterations	the number of smoothing iterations, when smooth = TRUE. Defaults to 1.
col	the color of the marker points and/or line, when using interactive options.
obj_dist, obj_width, eye_res	blur the image to model the visual acuity of non-human animals as per Caves & Johnsen (2018)'s AcuityView 2.0 algorithm. The procedure requires three arguments; obj_dist is the real-world distance between the viewer and the focal

	object in the image in the image, obj_width is the real-world width of the entire image; eye_res is the minimum resolvable angle of the viewer in degrees. All three arguments are numeric, and any units of measurement are suitable for obj_dist and obj_width, but they must match. Note that this is the more flexible v2.0 implementation meaning that any rectangular image is suitable; it need not be square with dimensions a power of 2. If using this capability, please cite Caves & Johnsen (2018), as per the included reference, and see note below.
`plotnew`	should plots be opened in a new window? Defaults to `FALSE`.
`...`	additional graphical parameters. Also see `par()`.

Value

an image, or list of images, for use in further `pavo` functions.

Note

There are several caveats that should be considered when using the AcuityView algorithm. First and foremost, the converted image is not what the animal actually sees. For example, it does not account for edge enhancement and other processing by the retina and brain that may alter an image. It does, however, show what spatial information can be detected and then processed by the visual system. Second, the converted image is static, which does not allow one to assess how movement may reveal the presence of an otherwise indiscernible object. Third, AcuityView makes several assumptions about the Modulation Transfer Function (MTF), which describes how the optical system affects image contrast as a function of the level of detail. These assumptions include that the MTF is constant over the region of the retina that views the scene, is circularly symmetrical, and is wavelength independent. For a full discussion and details, please do read Caves & Johnsen (2018).

Author(s)

Thomas E. White <thomas.white026@gmail.com>

References

Caves, E. M., & Johnsen, S. (2018). AcuityView: An r package for portraying the effects of visual acuity on scenes observed by an animal. Methods in Ecology and Evolution, 9(3), 793-797 doi:10.1111/2041210X.12911.

Chaikin, G. 1974. An algorithm for high speed curve generation. Computer Graphics and Image Processing 3, 346-349.

Examples

```
if (interactive()) {
  # Interactively add a scale to a single image
 papilio <- getimg(system.file("testdata/images/butterflies/papilio.png", package = "pavo"))
  papilio <- procimg(papilio, scaledist = 10)

  # Interactively assign individual scales to each image,
  # after slightly reducing their size (to 90% of original).
  snakes <- getimg(system.file("testdata/images/snakes", package = "pavo"))
```

```
  snakes <- procimg(snakes, scaledist = c(10, 14), resize = 90)

  # Model the appearance of a butterfly given the reduced visual acuity of another
  # animal viewer as per the AcuityView algorithm. Here our butterfly is 60 cm away,
 # the image width is 10 cm, and the minimum resolvable angle of the viewer is 0.2-degrees.
  tiger <- getimg(system.file("testdata/images/tiger.png", package = "pavo"))
  tiger_acuity <- procimg(tiger, obj_dist = 60, obj_width = 10, eye_res = 0.2)
}
```

`procspec` *Process spectra*

Description

Applies normalization and/or smoothing to spectra for further analysis or plotting.

Usage

```
procspec(
  rspecdata,
  opt = c("none", "smooth", "maximum", "minimum", "bin", "sum", "center"),
  fixneg = c("none", "addmin", "zero"),
  span = 0.25,
  bins = 20
)
```

Arguments

`rspecdata`	(required) a data frame, possibly of class `rspec`, which contains a column containing a wavelength range, named 'wl', and spectra data in remaining columns.
`opt`	what type of processing options to apply. User can select multiple options by providing a vector. Possibilities are: • `"none"` does not perform any processing (default). • `"smooth"` applies LOESS smoothing to each spectrum using `loess.smooth()`. Optimal smoothing parameter can be assessed by using `plotsmooth()`. • `"minimum"` subtracts the minimum from each individual spectra. • `"maximum"` divides each spectrum by its maximum value. • `"sum"` divides each spectrum by summed values. • `"bin"` bins each spectrum into the specified number of bins. `bins` argument must be set. • `"center"` centers individual spectra by subtracting mean reflectance from all values.
`fixneg`	how to handle negative values. Possibilities are: • `"none"` does not perform negative value correction (default). • `"zero"` sets all negative values to zero.

- "addmin" adds the absolute value of the maximally negative values of each spectra to the reflectance at all other wavelengths (setting the minimum value to zero, but scaling other values accordingly).

span — sets the smoothing parameter used by loess.smooth().

bins — sets the number of equally sized wavelength bins for opt = "bin".

Value

A data frame of class rspec with the processed data.

Author(s)

Chad Eliason <cme16@zips.uakron.edu>

References

Cuthill, I., Bennett, A. T. D., Partridge, J. & Maier, E. 1999. Plumage reflectance and the objective assessment of avian sexual dichromatism. The American Naturalist, 153, 183-200.

Montgomerie R. 2006. Analyzing colors. In Hill, G.E, and McGraw, K.J., eds. Bird Coloration. Volume 1 Mechanisms and measurements. Harvard University Press, Cambridge, Massachusetts.

White, T. E., Dalrymple, R. L., Noble D. W. A., O'Hanlon, J. C., Zurek, D. B., Umbers, K. D. L. 2015. Reproducible research in the study of biological coloration. Animal Behaviour, 106, 51-57.

See Also

loess.smooth(), plotsmooth()

Examples

```
data(teal)
plot(teal, select = 10)

# Smooth data to remove noise
teal.sm <- procspec(teal, opt = "smooth", span = 0.25)
plot(teal.sm, select = 10)

# Normalize to max of unity
teal.max <- procspec(teal, opt = c("max"))
plot(teal.max, select = 10)
```

projplot *2D projection of a tetrahedral colourspace*

Description

Produces a 2D projection plot of points in a tetrahedral colour space

Adds points to a tetrahedral colorspace projection

Usage

```
projplot(tcsdata, ...)

projpoints(tcsdata, ...)
```

Arguments

tcsdata	(required) tetrahedral color space coordinates, possibly a result from colspace(), containing values for the 'h.theta' and 'h.phi' coordinates as columns (labeled as such).
...	additional parameters to be passed to the plotting of data points.

Value

projplot() creates a 2D plot of color points projected from the tetrahedron to its encapsulating sphere, and is ideal to visualize differences in hue.

projpoints() creates points in a projection color space plot produced by projplot().

Note

projplot() uses the Mollweide projection, and not the Robinson projection, which has been used in the past. Among other advantages, the Mollweide projection preserves area relationships within latitudes without distortion.

Author(s)

Rafael Maia <rm72@zips.uakron.edu>

References

Stoddard, M. C., & Prum, R. O. (2008). Evolution of avian plumage color in a tetrahedral color space: A phylogenetic analysis of new world buntings. The American Naturalist, 171(6), 755-776.

Endler, J. A., & Mielke, P. (2005). Comparing entire colour patterns as birds see them. Biological Journal Of The Linnean Society, 86(4), 405-431.

Examples

```
data(sicalis)
vis.sicalis <- vismodel(sicalis, visual = "avg.uv")
tcs.sicalis <- colspace(vis.sicalis, space = "tcs")
projplot(tcs.sicalis, pch = 16, col = setNames(rep(seq_len(3), 7), rep(c("C", "T", "B"), 7)))
```

`sensdata` *Retrieve or plot in-built spectral sensitivity data*

Description

Retrieve (as an rspec object) or plot pavo's in-built spectral sensitivity data.

Usage

```
sensdata(
 visual = c("none", "all", "avg.uv", "avg.v", "bluetit", "ctenophorus", "star", "pfowl",
    "apis", "canis", "cie2", "cie10", "musca", "drosophila", "habronattus",
    "rhinecanthus"),
 achromatic = c("none", "all", "bt.dc", "ch.dc", "st.dc", "md.r1", "dm.r1", "ra.dc",
    "cf.r"),
  illum = c("none", "all", "bluesky", "D65", "forestshade"),
  trans = c("none", "all", "bluetit", "blackbird"),
  bkg = c("none", "all", "green"),
  plot = FALSE,
  ...
)
```

Arguments

`visual` visual systems. Options are:

- `"none"`: no visual sensitivity data.
- `"all"`: all visual sensitivity data.
- `"apis"`: Honeybee *Apis mellifera* visual system.
- `"avg.uv"`: average avian UV system.
- `"avg.v"`: average avian V system.
- `"bluetit"`: Blue tit *Cyanistes caeruleus* visual system.
- `"canis"`: Canid *Canis familiaris* visual system.
- `"cie2"`: 2-degree colour matching functions for CIE models of human colour vision. Functions are linear transformations of the 2-degree cone fundamentals of Stockman & Sharpe (2000), as ratified by the CIE (2006).
- `"cie10"`: 10-degree colour matching functions for CIE models of human colour vision. Functions are linear transformations of the 10-degree cone fundamentals of Stockman & Sharpe (2000), as ratified by the CIE (2006).
- `"ctenophorus"`: Ornate dragon lizard *Ctenophorus ornatus*.

- "musca": Housefly *Musca domestica* visual system.
- 'drosophila': Vinegar fly *Drosophila melanogaster* (Sharkey et al. 2020).
- "pfowl": Peafowl *Pavo cristatus* visual system.
- "star": Starling *Sturnus vulgaris* visual system.
- "habronattus": Jumping spider *Habronattus pyrrithrix*.
- "rhinecanthus": Triggerfish *Rhinecanthus aculeatus*.

achromatic — the sensitivity data used to calculate luminance (achromatic) receptor stimulation. Options are:

- "none": no achromatic sensitivity data.
- "all": all achromatic sensitivity data.
- "bt.dc": Blue tit *Cyanistes caeruleus* double cone.
- "ch.dc": Chicken *Gallus gallus* double cone.
- "st.dc": Starling *Sturnus vulgaris* double cone.
- "cf.r": Canid *Canis familiaris* rod
- "md.r1": Housefly *Musca domestica* R1-6 photoreceptor.
- 'dm.r1': Vinegar fly *Drosophila melanogaster* R1-6 photoreceptor.
- "ra.dc": Triggerfish *Rhinecanthus aculeatus* double cone.

illum — illuminants. Options are:

- "none": no illuminant data.
- "all": all background spectral data.
- "bluesky" open blue sky.
- "D65": standard daylight.
- "forestshade" forest shade.

trans — Ocular transmission data. Options are:

- "none": no transmission data.
- "all": all transmission data.
- "bluetit": blue tit *Cyanistes caeruleus* ocular transmission (from Hart et al. 2000).
- "blackbird": blackbird *Turdus merula* ocular transmission (from Hart et al. 2000).

bkg — background spectra. Options are:

- "none": no background spectral data.
- "all": all background spectral data.
- "green": green foliage.

plot — should the spectral data be plotted, or returned instead (defaults to FALSE)?

... — additional graphical options passed to plot.rspec() when plot = TRUE.

Value

An object of class rspec (when plot = FALSE), containing a wavelength column "wl" and spectral data binned at 1 nm intervals from 300-700 nm.

Author(s)

Thomas E. White <thomas.white026@gmail.com>

Rafael Maia <rm72@zips.uakron.edu>

References

Sharkey, C. R., Blanco, J., Leibowitz, M. M., Pinto-Benito, D., & Wardill, T. J. (2020). The spectral sensitivity of Drosophila photoreceptors. Scientific reports, 10(1), 1-13.

Examples

```
# Plot the honeybee's receptors
sensdata(visual = "apis", ylab = "Absorbance", plot = TRUE)

# Plot the average UV vs V avian receptors
sensdata(visual = c("avg.v", "avg.uv"), ylab = "Absorbance", plot = TRUE)

# Retrieve the CIE colour matching functions as an rspec object
ciedat <- sensdata(visual = c("cie2", "cie10"))
```

---

sensmodel *Modeling spectral sensitivity*

Description

Models spectral sensitivity (with oil droplets; optional) based on peak cone sensitivity according to the models of Govardovskii et al. (2000) and Hart & Vorobyev (2005).

Usage

```
sensmodel(
  peaksens,
  range = c(300, 700),
  lambdacut = NULL,
  Bmid = NULL,
  oiltype = NULL,
  beta = TRUE,
  om = NULL,
  integrate = TRUE,
  sensnames = NULL
)
```

Arguments

peaksens	(required) a vector with peak sensitivities for the cones to model.
range	a vector of length 2 for the range over which to calculate the spectral sensitivities (defaults to 300nm to 700nm).
lambdacut	a vector of same length as peaksens that lists the cut-off wavelength value for oil droplets. Needs either Bmid or oiltype to also be entered. See Hart and Vorobyev (2005).
Bmid	a vector of same length as peaksens that lists the gradient of line tangent to the absorbance spectrum of the oil droplets. See Hart and Vorobyev (2005).
oiltype	a list of same length as peaksens that lists the oil droplet types (currently accepts only "T", C", "Y", "R", "P") when Bmid is not known. Calculates Bmid based on the regression equations found in Hart ad Vorobyev (2005).
beta	logical. If TRUE the sensitivities will include the beta peak See Govardovskii et al.(2000) (defaults to TRUE).
om	a vector of same length as range1-range2 that contains ocular media transmission data. If included, cone sensitivity will be corrected for ocular media transmission. Currently accepts "bird" using values from Hart et al. (2005), or user-defined values.
integrate	logical. If TRUE, each curve is transformed to have a total area under the curve of 1 (best for visual models; defaults to TRUE). NOTE: integration is applied before any effects of ocular media are considered, for compatibility with visual model procedures.
sensnames	A vector equal in length to peaksens, specifying custom names for the resulting sensitivity curves (e.g. c('s', 'm', 'l') for short-, medium- and long-wavelength sensitive receptors.)

Value

A data frame of class rspec containing each cone model as a column.

Author(s)

Pierre-Paul Bitton <bittonp@uwindsor.ca>

Chad Eliason <cme16@zips.uakron.edu>

References

Govardovskii VI, Fyhrquist N, Reuter T, Kuzmin DG and Donner K. 2000. In search of the visual pigment template. Visual Neuroscience 17:509-528, doi:10.1017/S0952523800174036

Hart NS, and Vorobyev M. 2005. Modeling oil droplet absorption spectra and spectral sensitivities of bird cone photoreceptors. Journal of Comparative Physiology A. 191: 381-392, doi:10.1007/s0035900405953

Hart NS, Partridge JC, Cuthill IC, Bennett AT (2000) Visual pigments, oil droplets, ocular media and cone photoreceptor distribution in two species of passerine bird: the blue tit (*Parus caeruleus* L.) and the blackbird (*Turdus merula* L.). J Comp Physiol A 186:375-387, doi:10.1007/s003590050437

Examples

```
# Blue tit visual system based on Hart et al (2000)
bluesens <- sensmodel(c(371, 448, 502, 563),
  beta = FALSE,
  lambdacut = c(330, 413, 507, 572),
  oiltype = c("T", "C", "Y", "R"), om = TRUE
)

# Danio aequipinnatus based on Govardovskii et al. (2000)
daniosens <- sensmodel(c(357, 411, 477, 569))
```

---

sicalis *Spectral curves from three body regions of stripe-tailed yellow finch (*Sicalis citrina*) males*

Description

Dataset containing reflectance measurements from 3 body parts ("C": crown, "B": breast, "T": throat) from seven male stripe-tailed yellow finches (*Sicalis citrina*)

Usage

```
data(sicalis)
```

Format

An object of class `rspec` (inherits from `data.frame`) with 401 rows and 22 columns.

Author(s)

Rafael Maia <rm72@zips.uakron.edu>

---

simulate_spec *Simulate a spectrum*

Description

Simulate a naturalistic reflectance, radiance, irradiance, or transmission spectrum. Curves may have sigmoidal (s-shaped) and/or Gaussian (bell-shaped) features. Multiple Gaussian and sigmoidal curves can be combined in a single spectrum, to simulate more complex spectral functions.

Usage

```
simulate_spec(
  wl_inflect = NULL,
  wl_peak = NULL,
  width_sig = 20,
  width_gauss = 70,
  skew_gauss = 0,
  xlim = c(300, 700),
  ylim = c(0, 100)
)
```

Arguments

wl_inflect	A numeric vector specifying the wavelength location (in nm) for one or more inflection point(s) for a 'sigmoid' shaped curve, if desired.
wl_peak	A numeric vector specifying the wavelength location (in nm) for one or more inflection point(s) for a 'Gaussian' (or 'bell') shaped curve, if desired.
width_sig	A numeric value or vector (if multiple wl_inflect values are specified) specifying the steepness of the change, for any sigmoidal curve(s). Required when wl_peak is specified. Defaults to 20 nm.
width_gauss	A numeric value or vector specifying the the full-width at half-maximum of any Gaussian curve(s). Required when wl_peak is specified. Defaults to 70 nm.
skew_gauss	Skewness parameter for controlling the direction and magnitude of skew, when simulating for Gaussian curves (ignored when simulating only sigmoidal curves). Curves will have no skew when skew_gauss = 0 (default), right skew when skew_gauss > 0, and left skew when skew_gauss < 0. The parameter corresponds to 'alpha' in the the skew-normal distribution (Azzalini 1985).
xlim	A vector specifying the wavelength range of the simulated spectra. Defaults to 300-700nm (c(300, 700)).
ylim	A vector specifying the minimum and maximum reflectance values of the resulting curve. Defaults to 0 - 100 % (c(0, 100)). Note: sigmoidal curves, by default, move from low to high reflectance. But if high-to-low sigmoidal curves are desired this can be controlled by the ordering of the values given to ylim(). E.g. c(0, 100) will generate a low-to-high sigmoidal curve, whereas c(100, 0) will generate a high- to-low curve. The ordering of values has no effect on the Gaussian portions of the final curve.

Value

An object of class rspec.

Author(s)

Thomas White <thomas.white026@gmail.com>

Hugo Gruson <hugo.gruson+R@normalesup.org>

References

Azzalini A (1985). A class of distributions which includes the normal ones. Scan. J. Stat. 171-178.

Examples

```
# Single ideal 'grey' reflectance spectrum, with 50% reflectance across 300 - 700 nm.
reflect0 <- simulate_spec(ylim = c(0, 50))

# Single sigmoidal spectrum, with a low-to-high inflection at 550 nm.
reflect1 <- simulate_spec(wl_inflect = 550)

# Single Gaussian spectrum, with a peak at 400 nm
reflect2 <- simulate_spec(wl_peak = 400)

# Combination of both Gaussian (with peak at 340 nm) and sigmoidal (with inflection
# at 560 nm)
reflect3 <- simulate_spec(wl_inflect = 560, wl_peak = 340)

# Double-Gaussian peaks of differing widths
reflect4 <- simulate_spec(wl_peak = c(340, 560), width_gauss = c(12, 40))

# Complex spectrum with single sigmoidal peak and multi-Gaussian peaks
reflect5 <- simulate_spec(wl_inflect = 575, wl_peak = c(340, 430), width_gauss = c(20, 60))

# Simulate a set of Gaussian reflectance curves with peaks varying between 400-600nm
# in increments of 10, then combine into a single rspec object, and plot the result
peaks <- seq(400, 600, 10)  # Peak locations
reflectances <- lapply(seq_along(peaks), function(x) simulate_spec(wl_peak = peaks[x])) # Simulate
reflectances <- Reduce(merge, reflectances)  # Combine
plot(reflectances)  # Plot

# Simulate a set of Gaussian reflectance curves with a single peak at 500 nm, but
# with maximum reflectance varying from 30 to 90% in 10% increments, then combine
# into a single rspec object, and plot the result
ymax <- lapply(seq(30, 90, 10), function(x) c(0, x))  # Varying reflectance maxima
reflectances2 <- lapply(ymax, function(x) simulate_spec(wl_peak = 500, ylim = x)) # Simulate
reflectances2 <- Reduce(merge, reflectances2)  # Combine
plot(reflectances2)  # Plot

# To simulate non-reflectance spectra (like irradiances or radiances), it's often useful
# to explore more 'extreme' parameters. Here's a simple example which simulates
# natural daylight, as represented by the D65 standard daylight spectrum.
D65_real <- procspec(sensdata(illum = 'D65'), opt = 'smooth')  # Official D65 daylight spectrum
D65_sim <- simulate_spec(wl_peak = 400,
                         width_gauss = 1300,
                         skew_gauss = 10,
                         ylim = c(0, 1))  # Simulated D65
cor.test(D65_real$D65, D65_sim$spec_p400)  # >0.99 correlation
plot(merge(D65_real, D65_sim), lty = 1:2, ylab = 'Irradiance (%)') # Merge and plot the two spectra
```

spec2rgb *Spectrum to rgb colour conversion*

Description

Calculates rgb values from spectra based on human colour matching functions.

Usage

```
spec2rgb(rspecdata, alpha = 1)
```

Arguments

rspecdata	(required) a data frame, possibly of class rspec, which contains a column containing a wavelength range, named 'wl', and spectra data in remaining columns.
alpha	alpha value to use for colours (defaults to 1, opaque).

Value

A character vector consisting of hexadecimal colour values for passing to further plotting functions.

Author(s)

Hugo Gruson <hugo.gruson+R@normalesup.org>

Chad Eliason <cme16@zips.uakron.edu>

References

CIE(1932). Commission Internationale de l'Eclairage Proceedings, 1931. Cambridge: Cambridge University Press.

Examples

```
data(teal)
spec2rgb(teal)

# Plot data using estimated perceived colour
plot(teal, col = spec2rgb(teal), type = "overlay")
```

subset.rspec *Subset rspec, vismodel, and colspace objects*

Description

Subsets various object types based on a given vector or grep partial matching of data names.

Usage

```
## S3 method for class 'rspec'
subset(x, subset, ...)

## S3 method for class 'colspace'
subset(x, subset, ...)

## S3 method for class 'vismodel'
subset(x, subset, ...)
```

Arguments

x	(required) an object of class rspec, vismodel, or colspace, containing spectra, visual model output or colourspace data to subset.
subset	a string used for partial matching of observations.
...	additional attributes passed to grep. Ignored if subset is logical.

Value

a subsetted object of the same class as the input object.

Note

if more than one value is given to subset, any spectra that matches *either* condition will be included. It's a union, not an intersect.

Author(s)

Chad Eliason <cme16@zips.uakron.edu>

Examples

```
data(sicalis)
vis.sicalis <- vismodel(sicalis)
tcs.sicalis <- colspace(vis.sicalis, space = "tcs")

# Subset all 'crown' patches (C in file names)
head(subset(sicalis, "C"))
head(subset(sicalis, c("B", "C")))
head(subset(sicalis, "T", invert = TRUE))
```

```
subset(vis.sicalis, "C")
subset(tcs.sicalis, "C")[, seq_len(5)]
```

`summary.colspace` *Colourspace data summary*

Description

Returns the attributes of `colspace` objects.

Usage

```
## S3 method for class 'colspace'
summary(object, by = NULL, ...)
```

Arguments

`object`	(required) a `colspace` object.
`by`	when the input is in `tcs` colourspace, `by` is either a single value specifying the range of colour points for which summary tetrahedral-colourspace variables should be calculated (for example, `by = 3` indicates summary will be calculated for groups of 3 consecutive colour points (rows) in the quantum catch colour data frame) or a vector containing identifications for the rows in the quantum catch colour data frame (in which case summaries will be calculated for each group of points sharing the same identification). If `by` is left blank, the summary statistics are calculated across all colour points in the data.
`...`	class consistency (ignored).

Value

returns all attributes of the data as mapped to the selected colourspace, including options specified when calculating the visual model. Also return the default `data.frame` summary, except when the object is the result of `tcspace()`, in which case the following variables are output instead:

- `centroid.u, .s, .m, .l` the centroids of `usml` coordinates of points.
- `c.vol` the total volume occupied by the points, computed with a convex hull.
- `rel.c.vol` volume occupied by the points (convex hull volume) relative to the tetrahedron volume.
- `colspan.m` the mean hue span.
- `colspan.v` the variance in hue span.
- `huedisp.m` the mean hue disparity.
- `huedisp.v` the variance in hue disparity.
- `mean.ra` mean saturation.
- `max.ra` maximum saturation achieved by the group of points.
- `a.vol` colour volume computed with α-shapes.

Author(s)

Rafael Maia <rm72@zips.uakron.edu>

References

Stoddard, M. C., & Prum, R. O. (2008). Evolution of avian plumage color in a tetrahedral color space: A phylogenetic analysis of new world buntings. The American Naturalist, 171(6), 755-776.

Endler, J. A., & Mielke, P. (2005). Comparing entire colour patterns as birds see them. Biological Journal Of The Linnean Society, 86(4), 405-431.

Gruson H. (2020). Estimation of colour volumes as concave hypervolumes using α-shapes. Methods in Ecology and Evolution, 11(8), 955-963 doi:10.1111/2041210X.13398

Examples

```
# Colour hexagon
data(flowers)
vis.flowers <- vismodel(flowers,
  visual = "apis", qcatch = "Ei", relative = FALSE,
  vonkries = TRUE, bkg = "green"
)
flowers.hex <- hexagon(vis.flowers)
summary(flowers.hex)

# Tetrahedral model
data(sicalis)
vis.sicalis <- vismodel(sicalis, visual = "avg.uv")
csp.sicalis <- colspace(vis.sicalis)
summary(csp.sicalis, by = rep(c("C", "T", "B"), 7))
```

`summary.rimg` *Image summary*

Description

Returns the attributes of, and optionally plots, an image.

Usage

```
## S3 method for class 'rimg'
summary(object, plot = FALSE, axes = TRUE, col = NULL, ...)
```

Arguments

`object`	(required) an image of class `rimg`, or list thereof.
`plot`	logical; plot the image and, if the image is color-classified, the colours corresponding to colour class categories side-by-side? Defaults to `FALSE`.
`axes`	should axes be drawn when `plot = TRUE`? (defaults to `TRUE`).

col	optional vector of colours when plotting colour-classified images with plot = TRUE. Defaults to the mean RGB values of the k-means centres (i.e. the 'original' colours).
...	additional graphical options when plot = TRUE. Also see par().

Value

Either the RGB values of the k-means centres from the colour-classified image, or a plot of both the image and specified colours (when plot = TRUE).

Author(s)

Thomas E. White <thomas.white026@gmail.com>

Examples

```
papilio <- getimg(system.file("testdata/images/butterflies/papilio.png", package = "pavo"))
papilio_class <- classify(papilio, kcols = 4)
summary(papilio_class)

# Plot the colour-classified image alongside the colour class palette
summary(papilio_class, plot = TRUE)

# Multiple images
snakes <- getimg(system.file("testdata/images/snakes", package = "pavo"))
snakes_class <- classify(snakes, kcols = 3)
summary(snakes_class, plot = TRUE)
```

---

summary.rspec *Colourimetric variables*

Description

Calculates all 23 colourimetric variables reviewed in Montgomerie (2006).

Usage

```
## S3 method for class 'rspec'
summary(object, subset = FALSE, lim = NULL, wlmin = NULL, wlmax = NULL, ...)
```

Arguments

object	(required) a data frame, possibly an object of class rspec, with a column with wavelength data, named 'wl', and the remaining column containing spectra to process.

subset	Either FALSE (the default), TRUE, or a character vector. If FALSE, all variables calculated are returned. If TRUE, only a subset of the complete output (composed of B2, S8 and H1; the variables described in Andersson and Prager 2006) are returned. Finally, a user-specified string of variable names can be used in order to filter and show only those variables.
lim	The range of wavelengths used in calculations. The default is to use the entire range in the rspec object (typically equivalent to lim = c(300, 700)).
wlmin, wlmax	Deprecated. Use the lim argument instead.
...	class consistency (ignored)

Value

A data frame containing either 23 or 5 (subset = TRUE) variables described in Montgomerie (2006) with spectra name as row names. The colorimetric variables calculated by this function are described in Montgomerie (2006) with corrections included in the README CLR file from the May 2008 distribution of the CLR software. Authors should reference both this package,Montgomerie (2006), and the original reference(s). Description and notes on the measures:

B1 (Total brightness): Sum of the relative reflectance over the entire spectral range (area under the curve). Frequently used but should be discouraged because values are difficult to compare across studies (B2 is preferred). REF 1-3, 7, 9-11, 13

B2 (Mean brightness): Mean relative reflectance over the entire spectral range. This is preferred to B1 since values are easier to compare across studies. REF 4, 12

B3 (Intensity): Maximum relative reflectance (Reflectance at wavelength of maximum reflectance). Note that may be sensitive to noise near the peak. REF 1, 5, 6

S1 (Chroma): Relative contribution of a spectral range to the total brightness (B1) S1 is arbitrarily divided in 6 measures of chroma based on the wavelength ranges normally associated with specific hues. The values are calculated using the following ranges: S1U (UV, if applicable): lambda min-400nm; S1V (Violet) lambda min-415nm; S1B (Blue) 400nm-510nm; S1G (Green) 510nm-605nm; S1Y (Yellow) 550nm-625nm; S1R (Red) 605nm-lambda max. REF 2, 7, 8, 11-13

S2 (Spectral saturation): Rmax/Rmin This measure is sensitive to spectral noise. Proper interpretation of this value may be difficult for spectra with multiple peaks in the range of interest. REF 1

S3 (Chroma): Reflectance over the Rmax +- 50nm range divided by B1. Values for peaks within 50nm of either the minimum or maximum range of the data will not be comparable since the area under the curve for the area of interest will not always be based on the same wavelength range. Therefore, S3 should be interpreted with caution for peaks in the UV or Red range. REF 11

S4 (Spectral purity): |bmaxneg| , calculated by approximating the derivative of the spectral curve. As such, it is very sensitive to noise and should only be considered when data is adequately smoothed. NAs are returned for curves which do not, at any range of wavelength, decrease in intensity. Therefore, reflectance curves for brown and red surfaces, for example, should not generate a values. REF 1

S5 (Chroma): Similar in design to segment classification measures (see Montgomerie 2006 for details). REF 10

S6 (Contrast): Rmax - Rmin. Because it uses both Rmin and Rmax, this measure may be sensitive to spectral noise. REF 5, 6

S7 (Spectral saturation): Difference between the relative reflectance before and after the wavelength at which reflectance is halfway between its minimum (Rmin) and its maximum (Rmax). Somewhat sensitive to noise and can be misleading when more than one maxima and/or minima are present. REF 3, 9

S8 (Chroma): (Rmax - Rmin)/B2. Because it uses both Rmin and Rmax, this measure may be sensitive to spectral noise. REF 3, 13

S9 (Carotenoid chroma): (R700 - R450)/R700. Should only be used when the colour of the surface is clearly due to carotenoid pigmentation and R450 is lower than R700. Could be sensitive to noise. REF 8

S10 (Peaky chroma): (Rmax - Rmin)/B2 x |bmaxneg|. Should be used with properly smoothed curves. REF 7

H1 (Peak wavelength, hue): Wavelength of maximum reflectance. May be sensitive to noise and may be variable if there is more than one maxima. REF 1, 2, 4, 6, 7, 10-13

H2 (Hue): Wavelength at bmaxneg. Should be calculated using smoothed data. REF 2, 13

H3 (Hue): Wavelength at Rmid. Sensitive to noisy spectra and may be variable if there are more than one maxima and minima. REF 3, 9, 13

H4 (Hue): Similar in design to segment classification measures see Montgomerie (2006) for details. REF 10

H5 (Hue): Wavelength at bmax. Sensitive to noise and may be variable if there is more than one maxima and minima. REF 5

Note

If minimum wavelength is over 400, UV chroma is not computed.

Variables which compute bmax and bmaxneg should be used with caution, for they rely on smoothed curves to remove noise, which would otherwise result in spurious results. Make sure chosen smoothing parameters are adequate.

Smoothing affects only B3, S2, S4, S6, S10, H2, and H5 calculation. All other variables can be reliably extracted using non-smoothed data.

Author(s)

Thomas E. White <thomas.white026@gmail.com>

Pierre-Paul Bitton <bittonp@windsor.ca>

Rafael Maia <rm72@zips.uakron.edu>

References

Montgomerie R. 2006. Analyzing colors. In Hill, G.E, and McGraw, K.J., eds. Bird Coloration. Volume 1 Mechanisms and measurements. Harvard University Press, Cambridge, Massachusetts.

References describing variables:

1- Andersson, S. 1999. Morphology of uv reflectance in a whistling-thrush: Implications for the study of structural colour signalling in birds. Journal of Avian Biology 30:193-204.

2- Andersson, S., J. Ornborg, and M. Andersson. 1998. Ultraviolet sexual dimorphism and assortative mating in blue tits. Proceedings of the Royal Society B 265:445-450.

3- Andersson, S., S. Pryke, J. Ornborg, M. Lawes, and M. Andersson. 2002. Multiple receivers, multiple ornaments, and a trade-off between agonistic and epigamic signaling in a widowbird. American Naturalist 160:683-691.

4- Delhey, K., A. Johnsen, A. Peters, S. Andersson, and B. Kempenaers. 2003. Paternity analysis reveals opposing selection pressures on crown coloration in the blue tit (Parus caeruleus). Proceedings of the Royal Society B 270:2057-2063.

5- Keyser, A. and G. Hill. 1999. Condition-dependent variation in the blue-ultraviolet coloration of a structurally based plumage ornament. Proceedings of the Royal Society B 266:771-777.

6- Keyser, A.J. and G. Hill. 2000. Structurally based plumage coloration is an honest signal of quality in male blue grosbeaks. Behavioural Ecology 11:202-209.

7- Ornborg, J., S. Andersson, S. Griffith, and B. Sheldon. 2002. Seasonal changes in a ultraviolet structural colour signal in blue tits, Parus caeruleus. Biological Journal of the Linnean Society 76:237-245.

8- Peters, A., A. Denk, K. Delhey, and B. Kempenaers. 2004. Carotenoid-based bill colour as an indicator of immunocompetence and sperm performance in male mallards. Journal of Evolutionary Biology 17:1111-1120.

9- Pryke, S., M. Lawes, and S. Andersson. 2001. Agonistic carotenoid signalling in male red-collared widowbirds: Aggression related to the colour signal of both the territory owner and model intruder. Animal Behaviour 62:695-704.

10- Saks, L., K. Mcgraw, and P. Horak. 2003. How feather colour reflects its carotenoid content. Functional Ecology 17:555-561.

11- Shawkey, M., A. Estes, L. Siefferman, and G. Hill. 2003. Nanostructure predicts intraspecific variation in ultraviolet-blue plumage colour. Proceedings of the Royal Society B 270:1455-1460.

12- Siefferman, L. and G. Hill. 2005. UV-blue structural coloration and competition for nestboxes in male eastern bluebirds. Animal Behaviour 69:67-72.

13- Smiseth, P., J. Ornborg, S. Andersson, and T. Amundsen. 2001. Is male plumage reflectance correlated with paternal care in bluethroats? Behavioural Ecology 12:164-170.

Examples

```
# Load data
data(sicalis)

# Calculate and display all spectral summary variables
summary(sicalis)

# Calculate only subset of B2, S8 and H1 as per Andersson (1999)
summary(sicalis, subset = TRUE)

# Calculate user-specified subset of B1 and H4
summary(sicalis, subset = c("B1", "H4"))
```

`summary.vismodel` *Visual model summary*

Description

Returns the attributes used when calculating a visual model using `vismodel()`

Usage

```
## S3 method for class 'vismodel'
summary(object, ...)
```

Arguments

`object`	(required) Results of `vismodel()`
`...`	class consistency (ignored)

Value

Returns all attributes chosen when calculating the visual model, as well as the default `data.frame` summary

Author(s)

Rafael Maia <rm72@zips.uakron.edu>

References

Vorobyev, M., Osorio, D., Bennett, A., Marshall, N., & Cuthill, I. (1998). Tetrachromacy, oil droplets and bird plumage colours. Journal Of Comparative Physiology A-Neuroethology Sensory Neural And Behavioral Physiology, 183(5), 621-633.

Hart, N. S. (2001). The visual ecology of avian photoreceptors. Progress In Retinal And Eye Research, 20(5), 675-703.

Stoddard, M. C., & Prum, R. O. (2008). Evolution of avian plumage color in a tetrahedral color space: A phylogenetic analysis of new world buntings. The American Naturalist, 171(6), 755-776.

Endler, J. A., & Mielke, P. (2005). Comparing entire colour patterns as birds see them. Biological Journal Of The Linnean Society, 86(4), 405-431.

Examples

```
data(sicalis)
vis.sicalis <- vismodel(sicalis, visual = "avg.uv")
summary(vis.sicalis)
```

`tcsplot` *Interactive plot of a tetrahedral colourspace*

Description

Produces an interactive 3D plot of a tetrahedral colourspace using OpenGL capabilities.

Plots points in a tetrahedral colour space

Usage

```
tcsplot(
  tcsdata,
  size = 0.02,
  alpha = 1,
  col = "black",
  vertexsize = 0.02,
  achro = TRUE,
  achrosize = 0.01,
  achrocol = "grey",
  lwd = 1,
  lcol = "lightgrey",
  new = FALSE,
  hspin = FALSE,
  vspin = FALSE,
  floor = TRUE,
  gamut = FALSE
)

tcspoints(tcsdata, size = 0.02, col = "black", alpha = 1)

tcsvol(
  tcsdata,
  type = c("convex", "alpha"),
  avalue = "auto",
  col = "black",
  alpha = 0.2,
  grid.alpha = 1,
  grid = TRUE,
  fill = TRUE,
  lwd = 1
)
```

Arguments

`tcsdata`	(required) a data frame, possibly a result from the `colspace()` or `tcspace()` function, containing values for the 'x', 'y' and 'z' coordinates as columns (labeled as such).

size	size of the points in the plot (defaults to 0.02)
alpha	transparency of points (or volume fill in tcsvol())
col	colour of the points in the plot (defaults to black)
vertexsize	size of the points at the vertices
achro	should a point be plotted at the origin (defaults to TRUE)?
achrosize	size of the point at the origin when achro = TRUE (defaults to 0.8).
achrocol	color of the point at the origin achro = TRUE (defaults to 'grey').
lwd, lcol	graphical parameters for the edges of the tetrahedron.
new	should a new 3D plot be called (defaults to FALSE)?
hspin	if TRUE, the graphic will spin horizontally (around the 'z' axis)(defaults to FALSE)
vspin	if TRUE, the graphic will spin vertically (around the 'x' axis)(defaults to FALSE)
floor	if TRUE, a reference xy plane is plotted under the tetrahedron (defaults to TRUE)
gamut	logical. Should the polygon showing the possible colours given visual system and illuminant used in the analysis (defaults to FALSE). This option currently only works when qcatch = Qi.
type	accepts a vector of length 1 or 2 with 'p' for points and/or 'l' for lines from the point to the base of the tetrahedron.
avalue	if type = "alpha", which alpha parameter value should be used to compute the alphashape. avalue = "auto" (default) finds and use the α^* value as defined in Gruson (2020).
grid.alpha	transparency of the volume polygon grid lines
grid	if TRUE, connects the polygon outlining the volume occupied by points (defaults to TRUE)
fill	if TRUE, fills the volume occupied by points (WARNING: transparency is not saved properly if exported using rgl.postscript)(defaults to TRUE).

Value

tcsplot() creates a 3D plot using functions of the package **rgl**, based on openGL capabilities. Plot is interactive and can be manipulated with the mouse (left button: rotate along 'z' axis; right button: rotate along 'x' axis; third button: zoom).

tcspoints() adds points to the plot. Points are currently plotted only as spheres to maintain export capabilities.

tcsvol() creates a 3D colour volume within a tcsplot object.

Author(s)

Rafael Maia <rm72@zips.uakron.edu>

References

Stoddard, M. C., & Prum, R. O. (2008). Evolution of avian plumage color in a tetrahedral color space: A phylogenetic analysis of new world buntings. The American Naturalist, 171(6), 755-776.

Endler, J. A., & Mielke, P. (2005). Comparing entire colour patterns as birds see them. Biological Journal Of The Linnean Society, 86(4), 405-431.

Examples

```
# For plotting
data(sicalis)
vis.sicalis <- vismodel(sicalis, visual = "avg.uv")
tcs.sicalis <- colspace(vis.sicalis, space = "tcs")
tcsplot(tcs.sicalis, size = 0.005)
rgl::rgl.postscript("testplot.pdf", fmt = "pdf")
rgl::rgl.snapshot("testplot.png")

# For adding points
patch <- rep(c("C", "T", "B"), 7)
tcs.crown <- subset(tcs.sicalis, "C")
tcs.breast <- subset(tcs.sicalis, "B")
tcsplot(tcs.crown, col = "blue")
tcspoints(tcs.breast, col = "red")

# For plotting convex hull
tcsplot(tcs.sicalis, col = "blue", size = 0.005)
tcsvol(tcs.sicalis)
```

---

teal *Angle-resolved reflectance data for the iridescent wing patch of a male green-winged teal (*Anas carolinensis*)*

Description

Dataset containing reflectance measurements from the wing patch of a single male at different incident angles (15-75 degrees in 5-degree increments).

Usage

```
data(teal)
```

Format

An object of class `rspec` (inherits from `data.frame`) with 401 rows and 13 columns.

Author(s)

Chad Eliason <cme16@zips.uakron.edu>

vismodel *Visual models*

Description

Calculates quantum catches at each photoreceptor. Both raw and relative values can be returned, for use in a suite of colourspace and non-colourspace models.

Usage

```
vismodel(
  rspecdata,
  visual = c("avg.uv", "avg.v", "bluetit", "ctenophorus", "star", "pfowl", "apis",
    "canis", "cie2", "cie10", "musca", "drosophila", "segment", "habronattus",
    "rhinecanthus"),
  achromatic = c("none", "bt.dc", "ch.dc", "st.dc", "md.r1", "dm.r1", "ra.dc", "cf.r",
    "ml", "l", "all"),
  illum = c("ideal", "bluesky", "D65", "forestshade"),
  trans = c("ideal", "bluetit", "blackbird"),
  qcatch = c("Qi", "fi", "Ei"),
  bkg = c("ideal", "green"),
  vonkries = FALSE,
  scale = 1,
  relative = TRUE
)
```

Arguments

rspecdata	(required) a data frame, possibly of class rspec, which contains a column containing a wavelength range, named 'wl', and spectra data in remaining columns.
visual	the visual system to be used. Options are:

- a data frame such as one produced containing by sensmodel(), containing user-defined sensitivity data for the receptors involved in colour vision. The data frame must contain a 'wl' column with the range of wavelengths included, and the sensitivity for each other cone as a column.
- 'apis': Honeybee *Apis mellifera*.
- 'avg.uv': average avian UV system (default).
- 'avg.v': average avian V system.
- 'bluetit': Blue tit *Cyanistes caeruleus*.
- 'canis': Canid *Canis familiaris*.
- 'cie2': 2-degree colour matching functions for CIE models of human colour vision. Functions are linear transformations of the 2-degree cone fundamentals of Stockman & Sharpe (2000), as ratified by the CIE (2006).
- 'cie10': 10-degree colour matching functions for CIE models of human colour vision. Functions are linear transformations of the 10-degree cone fundamentals of Stockman & Sharpe (2000), as ratified by the CIE (2006).

- 'ctenophorus': Ornate dragon lizard *Ctenophorus ornatus*.
- 'musca': Housefly *Musca domestica*.
- 'drosophila': Vinegar fly *Drosophila melanogaster* (Sharkey et al. 2020).
- 'pfowl': Peafowl *Pavo cristatus*.
- 'segment': Generic tetrachromat 'viewer' for use in the segment analysis of Endler (1990).
- 'star': Starling *Sturnus vulgaris*.
- 'habronattus': Jumping spider *Habronattus pyrrithrix*.
- 'rhinecanthus': Triggerfish *Rhinecanthus aculeatus*.

achromatic — the sensitivity data to be used to calculate luminance (achromatic) receptor stimulation. Either a vector containing the sensitivity for a single receptor, or one of the options:

- 'none': no achromatic stimulation calculated (default).
- 'bt.dc': Blue tit *Cyanistes caeruleus* double cone.
- 'ch.dc': Chicken *Gallus gallus* double cone.
- 'st.dc': Starling *Sturnus vulgaris* double cone.
- 'md.r1': Housefly *Musca domestica* R1-6 photoreceptor.
- 'dm.r1': Vinegar fly *Drosophila melanogaster* R1-6 photoreceptor.
- 'ra.dc': Triggerfish *Rhinecanthus aculeatus* double cone.
- 'cf.r': Canid *Canis familiaris* cone.
- 'ml': the summed response of the two longest-wavelength photoreceptors.
- 'l': the longest-wavelength photoreceptor.
- 'all': the summed response of all photoreceptors.

illum — either a vector containing the illuminant, or one of the options:

- 'ideal': homogeneous illuminance of 1 across wavelengths (default)
- 'bluesky' open blue sky.
- 'D65': standard daylight.
- 'forestshade' forest shade.

trans — either a vector containing the ocular or environmental transmission spectra, or one of the options:

- 'ideal': homogeneous transmission of 1 across all wavelengths (default)
- 'bluetit': blue tit *Cyanistes caeruleus* ocular transmission (from Hart et al. 2000).
- 'blackbird': blackbird *Turdus merula* ocular transmission (from Hart et al. 2000).

qcatch — Which quantal catch metric to return. Options are:

- 'Qi': Quantum catch for each photoreceptor (default)
- 'fi': Quantum catch according to Fechner's law (the signal of the receptor channel is proportional to the logarithm of the quantum catch)
- 'Ei': Hyperbolic-transformed quantum catch, where Ei = Qi / (Qi + 1).

bkg — background spectrum. Note that this will have no effect when vonkries = FALSE. Either a vector containing the spectral data, or one of the options:

- 'ideal': homogeneous illuminance of 1 across all wavelengths (default).
- 'green': green foliage.

vonkries — logical. Should the von Kries colour correction transformation be applied? (defaults to FALSE).

scale — a value by which the illuminant will be multiplied. Useful for when the illuminant is a relative value (i.e. transformed to a maximum of 1 or to a percentage), and does not correspond to quantum flux units ($\mu mol.s^{-1}.m^{-2}$). Useful values are, for example, 500 (for dim light) and 10000 (for bright illumination). Note that if vonkries = TRUE this transformation has no effect.

relative — should relative quantum catches be returned (i.e. is it a colour space model? Defaults to TRUE).

Value

An object of class vismodel containing the photon catches for each of the photoreceptors considered. Information on the parameters used in the calculation are also stored and can be called using the summary.vismodel() function.

Note

Built-in visual, achromatic, illum, bkg and trans are only defined on the 300 to 700nm wavelength range. If you wish to work outside this range, you will need to provide your own data.

Author(s)

Thomas E. White <thomas.white026@gmail.com>

Rafael Maia <rm72@zips.uakron.edu>

References

Vorobyev, M., Osorio, D., Bennett, A., Marshall, N., & Cuthill, I. (1998). Tetrachromacy, oil droplets and bird plumage colours. Journal Of Comparative Physiology A-Neuroethology Sensory Neural And Behavioral Physiology, 183(5), 621-633.

Hart, N. S., Partridge, J. C., Cuthill, I. C., Bennett, A. T. D. (2000). Visual pigments, oil droplets, ocular media and cone photoreceptor distribution in two species of passerine bird: the blue tit (Parus caeruleus L.) and the blackbird (Turdus merula L.). Journal of Comparative Physiology A, 186, 375-387.

Hart, N. S. (2001). The visual ecology of avian photoreceptors. Progress In Retinal And Eye Research, 20(5), 675-703.

Barbour H. R., Archer, M. A., Hart, N. S., Thomas, N., Dunlop, S. A., Beazley, L. D, Shand, J. (2002). Retinal characteristics of the Ornate Dragon Lizard, Ctenophorus ornatus.

Stoddard, M. C., & Prum, R. O. (2008). Evolution of avian plumage color in a tetrahedral color space: A phylogenetic analysis of new world buntings. The American Naturalist, 171(6), 755-776.

Endler, J. A., & Mielke, P. (2005). Comparing entire colour patterns as birds see them. Biological Journal Of The Linnean Society, 86(4), 405-431.

Chittka L. (1992). The colour hexagon: a chromaticity diagram based on photoreceptor excitations as a generalized representation of colour opponency. Journal of Comparative Physiology A, 170(5), 533-543.

Stockman, A., & Sharpe, L. T. (2000). Spectral sensitivities of the middle- and long-wavelength sensitive cones derived from measurements in observers of known genotype. Vision Research, 40, 1711-1737.

CIE (2006). Fundamental chromaticity diagram with physiological axes. Parts 1 and 2. Technical Report 170-1. Vienna: Central Bureau of the Commission Internationale de l' Eclairage.

Neitz, J., Geist, T., Jacobs, G.H. (1989) Color vision in the dog. Visual Neuroscience, 3, 119-125.

Sharkey, C. R., Blanco, J., Leibowitz, M. M., Pinto-Benito, D., & Wardill, T. J. (2020). The spectral sensitivity of Drosophila photoreceptors. Scientific reports, 10(1), 1-13.

See Also

`sensdata()` to retrieve or plot in-built spectral sensitivity data used in `vismodel()`

Examples

```
# Dichromat (dingo)
data(flowers)
vis.dingo <- vismodel(flowers, visual = "canis")
di.dingo <- colspace(vis.dingo, space = "di")

# Trichromat (honeybee)
data(flowers)
vis.bee <- vismodel(flowers, visual = "apis")
tri.bee <- colspace(vis.bee, space = "tri")

# Tetrachromat (blue tit)
data(sicalis)
vis.bluetit <- vismodel(sicalis, visual = "bluetit")
tcs.bluetit <- colspace(vis.bluetit, space = "tcs")

# Tetrachromat (starling), receptor-noise model
data(sicalis)
vis.star <- vismodel(sicalis, visual = "star", achromatic = "bt.dc", relative = FALSE)
dist.star <- coldist(vis.star, achromatic = TRUE)

# Estimate quantum catches using a custom trichromatic visual phenotype
custom <- sensmodel(c(330, 440, 550))
names(custom) <- c("wl", "s", "m", "l")
vis.custom <- vismodel(flowers, visual = custom)
tri.custom <- colspace(vis.custom, space = "tri")
```

vol *Plot a tetrahedral colour space*

Description

Produces a 3D colour volume in tetrahedral colour space when plotting a non-interactive tetrahedral plot.

Usage

```
vol(
  tcsdata,
  type = c("convex", "alpha"),
  avalue = "auto",
  alpha = 0.2,
  grid = TRUE,
  fill = TRUE,
  new = FALSE,
  ...
)
```

Arguments

tcsdata	(required) a data frame, possibly a result from the colspace() or tcspace() function, containing values for the 'x', 'y' and 'z' coordinates as columns (labeled as such).
type	if "convex", the colour volume is plotted using a convex hull and if "alpha", it is plotted using alphashapes.
avalue	if type = "alpha", which alpha parameter value should be used to compute the alphashape. avalue = "auto" (default) finds and use the α^* value as defined in Gruson (2020).
alpha	transparency of volume (if fill = TRUE).
grid	logical. if TRUE (default), draws the polygon outline defined by the points.
fill	logical. if TRUE (default), fills the volume defined by the points.
new	logical. Should a new plot be started or draw over an open plot? (defaults to FALSE)
...	additional graphical options. See polygon() and tetraplot().

Value

vol() creates a 3D colour volume within a static tetrahedral plot.

Author(s)

Rafael Maia <rm72@zips.uakron.edu>

Hugo Gruson

References

Stoddard, M. C., & Prum, R. O. (2008). Evolution of avian plumage color in a tetrahedral color space: A phylogenetic analysis of new world buntings. The American Naturalist, 171(6), 755-776.

Stoddard, M. C., & Stevens, M. (2011). Avian vision and the evolution of egg color mimicry in the common cuckoo. Evolution, 65(7), 2004-2013.

Maia, R., White, T. E., (2018) Comparing colors using visual models. Behavioral Ecology, ary017 doi:10.1093/beheco/ary017

Gruson H. (2020). Estimation of colour volumes as concave hypervolumes using α-shapes. Methods in Ecology and Evolution, 11(8), 955-963 doi:10.1111/2041210X.13398

Examples

```
# For plotting
data(sicalis)
vis.sicalis <- vismodel(sicalis, visual = "avg.uv")
tcs.sicalis <- colspace(vis.sicalis, space = "tcs")
plot(tcs.sicalis)

# Convex hull
vol(tcs.sicalis, type = "convex")

# Alpha-shape
if (require("alphashape3d")) {
  vol(tcs.sicalis, type = "alpha", avalue = 1)
}
```

`voloverlap` *Colour volume overlap*

Description

Calculates the overlap between the volumes defined by two sets of points in cartesian space.

Usage

```
voloverlap(
  colsp1,
  colsp2,
  type = c("convex", "alpha"),
  avalue = "auto",
  plot = FALSE,
  interactive = FALSE,
  col = c("blue", "red", "darkgrey"),
  fill = FALSE,
  new = TRUE,
  nsamp = 1000,
  psize = 0.001,
```

```
  lwd = 1,
  ...
)
```

Arguments

colsp1, colsp2	(required) data frame, possibly a result from the colspace() function, containing values for the 'x', 'y' (and possibly 'z') coordinates as columns (labeled as such)
type	if "convex", the colour volume is plotted using a convex hull and if "alpha", it is plotted using alphashapes.
avalue	if type = alpha, the alpha parameter values for colsp1 and colsp2 respectively to compute the alphashapes. Can be a numeric of length one if the same value is used in both cases. avalue = "auto" (default) finds and use the α^* value as defined in Gruson (2020).
plot	logical. Should the volumes and points be plotted? (defaults to FALSE). This only works for tetrahedral colourspaces at the moment.
interactive	logical. If TRUE, uses the rgl engine for interactive plotting; if FALSE then a static plot is generated.
col	a vector of length 3 with the colours for (in order) the first volume, the second volume, and the overlap.
fill	logical. should the two volumes be filled in the plot? (defaults to FALSE)
new	logical. Should a new plot window be called? If FALSE, volumes and their overlap are plotted over the current plot (defaults to TRUE).
nsamp	if type = "alpha", the number of points to be sampled for the Monte Carlo computation. Stoddard & Stevens(2011) use around 750,000 points, but more or fewer might be required depending on the degree of overlap.
psize	if type = "alpha" and plot = TRUE, sets the size to plot the points used in the Monte Carlo computation.
lwd	if plot = TRUE, sets the line width for volume grids.
...	additional arguments passed to the plot. See vol()

Value

Calculates the overlap between the volumes defined by two set of points in colourspace. The volume from the overlap is then given relative to:

- vsmallest the volume of the overlap divided by the smallest of that defined by the the two input sets of colour points. Thus, if one of the volumes is entirely contained within the other, this overlap will be vsmallest = 1.
- vboth the volume of the overlap divided by the combined volume of both input sets of colour points. If type = "alpha", If used, the output will be different:
- s_in1, s_in2 the number of sampled points that fall within each of the volumes individually.
- s_inboth the number of sampled points that fall within both volumes.
- s_ineither the number of points that fall within either of the volumes.

- psmallest the proportion of points that fall within both volumes divided by the number of points that fall within the smallest volume.
- pboth the proportion of points that fall within both volumes divided by the total number of points that fall within both volumes.

Note

Stoddard & Stevens (2011) originally obtained the volume overlap through Monte Carlo simulations of points within the range of the volumes, and obtaining the frequency of simulated values that fall inside the volumes defined by both sets of colour points.

Stoddard & Stevens (2011) also return the value of the overlap relative to one of the volumes (in that case, the host species). However, for other applications this value may not be what one expects to obtain if (1) the two volumes differ considerably in size, or (2) one of the volumes is entirely contained within the other. For this reason, we also report the volume relative to the union of the two input volumes, which may be more adequate in most cases.

Author(s)

Rafael Maia <rm72@zips.uakron.edu>

Hugo Gruson <hugo.gruson+R@normalesup.org>

References

Stoddard, M. C., & Prum, R. O. (2008). Evolution of avian plumage color in a tetrahedral color space: A phylogenetic analysis of new world buntings. The American Naturalist, 171(6), 755-776.

Stoddard, M. C., & Stevens, M. (2011). Avian vision and the evolution of egg color mimicry in the common cuckoo. Evolution, 65(7), 2004-2013.

Maia, R., White, T. E., (2018) Comparing colors using visual models. Behavioral Ecology, ary017 doi:10.1093/beheco/ary017

Gruson H. (2020). Estimation of colour volumes as concave hypervolumes using α-shapes. Methods in Ecology and Evolution, 11(8), 955-963 doi:10.1111/2041210X.13398

Examples

```
data(sicalis)
tcs.sicalis.C <- subset(colspace(vismodel(sicalis)), "C")
tcs.sicalis.T <- subset(colspace(vismodel(sicalis)), "T")
tcs.sicalis.B <- subset(colspace(vismodel(sicalis)), "B")

# Convex hull volume
voloverlap(tcs.sicalis.T, tcs.sicalis.B, type = "convex")
voloverlap(tcs.sicalis.T, tcs.sicalis.C, type = "convex", plot = TRUE)
voloverlap(tcs.sicalis.T, tcs.sicalis.C, type = "convex", plot = TRUE, col = seq_len(3))

# Alpha-shape volume
if (require("alphashape3d")) {
  voloverlap(tcs.sicalis.T, tcs.sicalis.B, type = "alpha", avalue = 1)
}
```

Index