

Package: luxR (via r-universe)

September 27, 2026

Type Package

Title Underwater Light Analysis and Visual Ecology

Version 0.1.1

Date 2026-07-29

Description Tools for quantifying and modelling underwater light environments. Propagates spectral irradiance through the water column via Beer-Lambert attenuation, computes photosynthetically active radiation (PAR), and converts between energy and photon-flux units. Provides the 'lux_spectrum' S3 class for unit-safe spectral arithmetic and format readers for TriOS RAMSES and Ocean Optics instruments. The optional 'lightr' package parses further instrument formats into strict native records; explicit measurement, scale, calibration, and resampling policies govern conversion via read_spectrum(). Spectra convert to and from the generic spectral classes of the optional 'photobiology' package, which also serves as the reference implementation luxR's integration and photometry are validated against. Includes visual ecology utilities: Govardovskii (2000) visual-pigment templates, photoreceptor quantum catch, Vorobyev-Osorio colour discrimination (JND), Secchi depth, and explicitly unvalidated contrast-threshold distance scenarios.

License GPL (>= 3)

URL <https://johnkirwan.github.io/luxR/>,
<https://github.com/JohnKirwan/luxR>

BugReports <https://github.com/JohnKirwan/luxR/issues>

Encoding UTF-8

LazyData true

RoxygenNote 8.0.0

Depends R (>= 3.5.0)

Imports graphics, stats, utils, Rdpack

RdMacros Rdpack

Suggests testthat (>= 3.0.0), shiny (>= 1.7.0), lightr (>= 2.0.0),
pavo (>= 2.9.0), photobiology (>= 0.11.0), colourvision (>= 2.0.0), knitr, rmarkdown, pkgdown

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Repository <https://johnkirwan.r-universe.dev>

Date/Publication 2026-07-29 21:25:56 UTC

RemoteUrl <https://github.com/JohnKirwan/luxR>

RemoteRef HEAD

RemoteSha 75ae5c201a47cf658f3ea43bb9ef75689be01158

Contents

luxR-package	4
absorbance	5
angle_of_polarization	6
as_colourvision	6
as_lux_spectrum	8
as_reflector_spct	10
as_rspect	11
as_source_spct	12
attenuate_depth	13
attenuate_spectrum	14
band_irradiance	15
beam_attenuation	17
broadband2spectrum	18
CIE_scotopic	19
CIE1931	20
colour_jnd	21
contrast_at_distance	22
convert_unit	24
degree_of_polarization	24
detectability	25
detection_range	27
fit_Kd	30
fit_sensitivity	31
fresnel_reflectance	32
from_naples	33
from_ocean_optics	34
from_solar	35
from_trios	35
ft2m	36
govardovskii_template	37

inherent_contrast	38
irradiance2lux	39
irradiance2radiance	41
jerlov_Kd	42
jerlov_types	44
light_at_depth	44
lux_spectrum	46
lux2irradiance	47
m2ft	48
micelson_contrast	49
n2W_spec_irradiance	49
Naples	51
par_fraction	51
par_irradiance	52
photic_depth	54
photon2W	54
plot_spectra	55
polarization_contrast	55
propagate_depth	56
propagate_spectrum	57
quantum_catch	59
radiance2irradiance	60
read_instrument_spectrum	61
read_ocean_optics	62
read_spectrum	62
read_trios	64
receptor_absorptance	65
reflectance_to_radiance	66
refracted_solar_angle	67
resample_spectrum	68
run_app	69
scattering_angle	70
scotopic_lux	71
secchi_depth	73
single_scattering_albedo	74
snells_window	74
solar_irradiance	75
solar_spectra	76
species_brightness	77
species_channel_support	78
species_channels	79
species_LEF	80
species_sensitivities	80
species_sensitivity_matrix	82
surface_transmittance	83
transmittance	84
underwater_polarization	85
unit_expression	87

unit_label	87
visual_range	88
W2mol_spec_irradiance	89
W2photon	90
wavelength_in_medium	91
weber_contrast	91
Index	93

luxR-package	<i>luxR: Underwater Light Analysis and Visual Ecology</i>
--------------	---

Description

Tools for quantifying and modelling underwater light environments. Propagates spectral irradiance through the water column via Beer-Lambert attenuation, computes PAR, and converts between energy and photon-flux units. Provides the [lux_spectrum](#) S3 class for unit-safe spectral arithmetic and native format readers for TriOS RAMSES and Ocean Optics instruments. Optional **lightr** interoperability preserves native instrument records until their measurement, scale, calibration, and resampling policies are explicitly declared. Includes visual ecology utilities: Govardovskii (2000) visual-pigment templates, photoreceptor quantum catch, Vorobyev-Osorio colour discrimination (JND), Secchi depth, and heuristic horizontal visual-range and object-detection scenarios.

Author(s)

Maintainer: John D Kirwan <jkirwan+luxR@pm.me>

Authors:

- John D Kirwan <jkirwan+luxR@pm.me>

See Also

Useful links:

- <https://johnkirwan.github.io/luxR/>
- <https://github.com/JohnKirwan/luxR>
- Report bugs at <https://github.com/JohnKirwan/luxR/issues>

absorbance	<i>Absorbance (optical density) from transmittance.</i>
------------	---

Description

The negative logarithm of transmittance. Natural-log absorbance (base = "e") is used by vision scientists and oceanographers; base-10 absorbance (base = "10") is "optical density" (OD), used by chemists and filter makers. They differ by a constant: $OD = A \log_{10} e \approx 0.4343 A$ (Johnsen 2012, ch. 4).

Usage

```
absorbance(transmittance, base = c("e", "10"))
```

Arguments

transmittance	Transmittance (0-1]. Vectorised.
base	"e" (natural-log absorbance, default) or "10" (optical density).

Value

Absorbance / optical density (dimensionless, ≥ 0).

References

Johnsen S (2012) The Optics of Life. Princeton University Press, ch. 4.

See Also

[transmittance](#)

Examples

```
absorbance(0.01, base = "10") # OD = 2 (1% transmitted)
absorbance(exp(-1.5))          # natural-log absorbance = 1.5
```

angle_of_polarization	<i>Angle of polarization (e-vector orientation) from Stokes parameters</i>
-----------------------	--

Description

$$AoP = \frac{1}{2} \operatorname{atan2}(U, Q)$$

. Polarization orientation is axial (period 180 degrees); the value is returned in degrees in the half-open range $(-90, 90]$.

Usage

angle_of_polarization(Q, U)

Arguments

Q, U Linear polarization Stokes parameters.

Value

Angle of polarization in degrees, in $(-90, 90]$. The result is NA where both Q and U are zero because orientation is undefined. Scalar inputs are broadcast; incompatible lengths are rejected.

See Also

[degree_of_polarization](#)

Examples

```
angle_of_polarization(Q = 1, U = 0) # 0
angle_of_polarization(Q = 0, U = 1) # 45
```

as_colourvision	<i>Convert luxR spectra to a colourvision input data frame</i>
-----------------	--

Description

Reshapes luxR data into the plain data frame **colourvision** expects for an illuminant (I) or reflectance (R1, R2, Rb): the first column is wavelength and each remaining column is one spectrum. luxR handles the water (depth propagation, the in-water light field); colourvision handles the colour-space / visual model (RNLmodel(), CTTKmodel(), colour spaces). Build the receptor matrix C with [species_sensitivity_matrix](#).

Usage

```
as_colourvision(x, ...)

## S3 method for class 'lux_spectrum'
as_colourvision(x, name = NULL, ...)

## S3 method for class 'list'
as_colourvision(x, ...)
```

Arguments

x	A lux_spectrum , or a (named) list of them. Every list element must already share the first spectrum's wavelength grid; mismatched grids are an error, so resample explicitly before conversion rather than relying on silent extrapolation.
...	Currently unused; present for method consistency.
name	Single-spectrum method only: the value column name. Defaults to the spectrum's meta\$label, then meta\$source, then "spec".

Details

Unlike [as_rspec](#), this performs no interpolation: colourvision resamples inputs itself via its nm/interpolate arguments at model time, so the converter is a pure reshape with no hidden resampling. **colourvision** is an optional (Suggests) dependency.

Value

A data.frame with a wl column plus one column per spectrum, ready to pass as colourvision's I, R1, R2, or Rb.

See Also

[species_sensitivity_matrix](#) for the C matrix; [as_rspec](#) for the pavo equivalent; `colourvision::RNLmodel`

Examples

```
## Not run:
I <- as_colourvision(from_solar("clear_noon"), name = "I")

## End(Not run)
```

as_lux_spectrum	<i>Coerce an object to lux_spectrum</i>
-----------------	---

Description

Converts a numeric vector or data frame to a [lux_spectrum](#) object. The data-frame method auto-detects the wavelength column (lambda > wavelength > wv) and the spectral-value column (E > irradiance > radiance > value). Passing an existing lux_spectrum returns it unchanged.

Usage

```
as_lux_spectrum(x, ...)
```

```
## S3 method for class 'numeric'
as_lux_spectrum(
  x,
  lambda,
  quantity = "irradiance",
  unit = "W/m2/nm",
  binwidth = NULL,
  meta = list(),
  ...
)
```

```
## S3 method for class 'data.frame'
as_lux_spectrum(
  x,
  lambda_col = NULL,
  E_col = NULL,
  quantity = "irradiance",
  unit = "W/m2/nm",
  binwidth = NULL,
  meta = list(),
  ...
)
```

```
## S3 method for class 'lux_spectrum'
as_lux_spectrum(x, ...)
```

```
## S3 method for class 'lux_instrument_spectrum'
as_lux_spectrum(
  x,
  quantity = NULL,
  unit = NULL,
  calibration = NULL,
  negative_policy = "error",
  ...
)
```

```

)

## S3 method for class 'rspec'
as_lux_spectrum(
  x,
  column = 2,
  quantity = "irradiance",
  unit = "W/m2/nm",
  meta = list(),
  ...
)

## S3 method for class 'source_spct'
as_lux_spectrum(
  x,
  measurement = NULL,
  negative_policy = "error",
  interpolate = FALSE,
  ...
)

## S3 method for class 'reflector_spct'
as_lux_spectrum(x, negative_policy = "error", interpolate = FALSE, ...)

```

Arguments

x	Object to coerce.
...	Additional arguments passed to methods.
lambda	Wavelength vector in nm (required for numeric method).
quantity	Physical quantity string. Default "irradiance".
unit	Unit string from the controlled vocabulary. Default "W/m2/nm".
binwidth	Bin width in nm; inferred if NULL.
meta	Named list of metadata.
lambda_col	Column name for wavelengths. Auto-detected from <code>lambda > wavelength > vv</code> if NULL.
E_col	Column name for spectral values. Auto-detected from <code>E > irradiance > radiance > value</code> if NULL.
calibration	For an instrument spectrum, a non-empty calibration identifier or description.
negative_policy	For an instrument spectrum, either error on negative processed values or explicitly replace them with zero and record the affected indices.
column	For the <code>rspec</code> method (pavo), which spectrum column to convert: a column name or index. Default the first spectrum (column 2, after <code>wl</code>).
measurement	For a photobiology <code>source_spct</code> carrying both an energy and a photon column, which to import: "energy" or "photon". Required only when the choice is ambiguous.

`interpolate` For a photobiology spectrum, whether an irregular wavelength grid may be re-sampled onto integer nanometres. Defaults to FALSE, so irregular grids fail with guidance to opt in.

Value

A `lux_spectrum` object.

See Also

[lux_spectrum](#), [from_naples](#), [from_solar](#)

Examples

```
# from a numeric vector
lam <- seq(400, 700, by = 10)
as_lux_spectrum(rep(1, length(lam)), lambda = lam)

# from a data frame
df <- data.frame(lambda = lam, E = rep(1, length(lam)))
as_lux_spectrum(df)
```

as_reflector_spct	<i>Convert luxR reflectance to a photobiology reflector_spct</i>
-------------------	--

Description

Bridges a reflectance [lux_spectrum](#) into **photobiology** as a `reflector_spct`. `luxR` already constrains reflectance to $[0, 1]$, so values map onto `Rfr` directly.

Usage

```
as_reflector_spct(x, ...)

## S3 method for class 'lux_spectrum'
as_reflector_spct(x, Rfr.type = NULL, strict.range = TRUE, ...)

## S3 method for class 'list'
as_reflector_spct(x, ...)
```

Arguments

<code>x</code>	A reflectance lux_spectrum , or a (named) list of them.
<code>...</code>	Passed to <code>photobiology::reflector_spct()</code> .
<code>Rfr.type</code>	Either "total" or "specular". Defaults to <code>x\$meta\$Rfr_type</code> ; an error is raised if neither is supplied.
<code>strict.range</code>	Passed to <code>photobiology::reflector_spct()</code> . Defaults to TRUE.

Details

`Rfr.type` is required. Whether a reflectance is total or specular is a fact about how it was measured, which `luxR` does not otherwise carry; defaulting it would silently mislabel the measurement inside another package's ecosystem. Record it once in the spectrum's `meta$Rfr_type` to have it picked up automatically.

Value

A **photobiology** `reflector_spct`.

See Also

[as_source_spct](#), [as_lux_spectrum](#)

Examples

```
## Not run:
r <- as_reflector_spct(my_reflectance, Rfr.type = "total")

## End(Not run)
```

as_rspec

Convert luxR spectra to a pavo rspec object

Description

Bridges `luxR` data into the **pavo** ecosystem so `pavo`'s colour-space and visual-modelling tools (`vismodel()`, `coldist()`, `colspace()`, `plotting`) can be applied to `luxR`-derived spectra. The division of labour: `luxR` handles the water (depth propagation, inherent optical properties, the in-water light field); `pavo` handles the colour space.

Usage

```
as_rspec(x, ...)

## S3 method for class 'lux_spectrum'
as_rspec(x, name = NULL, ...)

## S3 method for class 'list'
as_rspec(x, ...)
```

Arguments

<code>x</code>	A lux_spectrum , or a (named) list of them — for example the output of from_trios . List elements are interpolated onto the first spectrum's wavelength grid before merging.
<code>...</code>	Passed to <code>pavo::as.rspec()</code> (e.g. <code>lim</code> , <code>interp</code>).
<code>name</code>	Single-spectrum method only: the column name to give the spectrum in the <code>rspec</code> . Defaults to the spectrum's <code>meta\$label</code> , then <code>meta\$source</code> , then <code>"spec"</code> .

Details

Delegates to `pavo::as_rspec()`, which validates the input and, by default, interpolates onto a 1 nm grid, so the result is a genuine `rspec` object. **pavo** is an optional (Suggests) dependency.

Value

A **pavo** `rspec` data frame: a `wl` column plus one column per spectrum.

See Also

[as_lux_spectrum](#) for the reverse conversion; `pavo::as_rspec`, `pavo::vismodel`

Examples

```
## Not run:
# A bundled solar spectrum as an rspec, ready for the pavo toolkit:
r <- as_rspec(from_solar("clear_noon"))

# Several spectra at once (e.g. TriOS radiance scans):
r2 <- as_rspec(from_trios(system.file("extdata/trios.dat", package = "luxR")))

## End(Not run)
```

as_source_spct

Convert luxR spectra to photobiology spectral classes

Description

Bridges `luxR` data into the **photobiology** ecosystem so its generic spectral classes, integration, and photometry can be applied to `luxR`-derived spectra. The division of labour: `luxR` handles the water (depth propagation, inherent optical properties, the in-water light field); `photobiology` handles generic spectral computation.

Usage

```
as_source_spct(x, ...)

## S3 method for class 'lux_spectrum'
as_source_spct(x, strict.range = TRUE, ...)

## S3 method for class 'list'
as_source_spct(x, ...)
```

Arguments

<code>x</code>	A lux_spectrum , or a (named) list of them.
<code>...</code>	Passed to the <code>photobiology</code> constructor.
<code>strict.range</code>	Passed to the <code>photobiology</code> constructor. Defaults to <code>TRUE</code> so <code>photobiology</code> validates the values independently of <code>luxR</code> .

Details

`as_source_spct()` requires spectral irradiance and delegates to `photobiology::source_spct()`. Energy units become `s.e.irrad` and photon units become `s.q.irrad`, rescaled to photobiology's `mol/m2/s/nm`. The spectrum is **not** converted between energy and photon bases: it arrives in the basis it already had, and `photobiology::e2q()` or `q2e()` does any conversion, so only one implementation of that conversion is ever in play.

Radiance is refused, because `source_spct` has no per-steradian term. **photobiology** is an optional (Suggests) dependency.

Value

A **photobiology** `source_spct`.

See Also

[as_reflector_spct](#) for reflectance, [as_lux_spectrum](#) for the reverse conversion, [as_rspec](#) for the **pavo** bridge

Examples

```
## Not run:
# The downwelling light field at 15 m, as a photobiology source_spct:
# Jerlov Kd data cover 350-700 nm, so trim before propagating.
surface <- from_solar("clear_noon")[350, 700]
at_depth <- propagate_spectrum(surface, jerlov_Kd("II", surface$lambda),
                               from = 0, to = 15)
s <- as_source_spct(at_depth[["15"]])
photobiology::e_irrad(s)

## End(Not run)
```

attenuate_depth

Beer-Lambert attenuation at a single wavelength.

Description

Computes $E(z) = E_0 * \exp(-K_d * z)$. Vectorised over depth.

Usage

```
attenuate_depth(E0, Kd, depth)
```

Arguments

<code>E0</code>	Surface irradiance. Any unit; result is in the same unit.
<code>Kd</code>	Diffuse attenuation coefficient in 1/m. Must be ≥ 0 .
<code>depth</code>	Depth(s) in metres. Must be ≥ 0 . Scalar or vector.

Details

This is the package's principal lightweight water-column model. It assumes that K_d is constant over the modeled path and propagates each wavelength independently. It does not solve the angular radiance field, multiple scattering, or wavelength redistribution. The scalar API has no inherent wavelength or depth range; the caller is responsible for supplying a K_d that is valid for the medium and path.

Value

Numeric vector the same length as depth.

Examples

```
attenuate_depth(100, Kd = 0.06, depth = c(0, 10, 25, 50))
```

attenuate_spectrum	<i>Propagate a surface spectrum through the water column.</i>
--------------------	---

Description

Applies per-wavelength K_d across a depth grid using the Beer-Lambert law. Returns a tidy data frame, a numeric matrix, or (for lux_spectrum input) a named list of lux_spectrum objects.

Usage

```
attenuate_spectrum(E0_lambda, ...)

## S3 method for class 'numeric'
attenuate_spectrum(
  E0_lambda,
  Kd_lambda,
  depths,
  format = c("long", "matrix"),
  lambda = NULL,
  ...
)

## S3 method for class 'lux_spectrum'
attenuate_spectrum(E0_lambda, Kd_lambda, depths, ...)
```

Arguments

E0_lambda	Surface spectral irradiance. Length-N numeric vector or a lux_spectrum object.
...	Ignored.
Kd_lambda	Finite, non-negative diffuse attenuation coefficients (1/m) at the same wavelength bins as E0_lambda. Length-N numeric vector.

depths	Finite, non-negative depths to evaluate in metres. Length-M numeric vector.
format	"long" (default): tidy data frame with columns depth, lambda, E. "matrix": N x M numeric matrix. Ignored for lux_spectrum input.
lambda	Optional wavelength values (nm) for the lambda column of the long-format output. Defaults to bin indices 1:N.

Details

This is a wavelength-resolved Beer-Lambert calculation, not a radiative-transfer solution. It assumes a homogeneous $K_d(\lambda)$ over the modeled path, treats wavelengths independently, and accepts only irradiance when a lux_spectrum is supplied. The generic numeric API cannot verify physical quantity or wavelength alignment; callers must supply aligned irradiance and K_d vectors. Bundled Jerlov coefficients are supported only from 350–700 nm; see [jerlov_Kd](#).

Value

A data frame, matrix, or named list of lux_spectrum.

References

Kirk JTO (1994) Light and Photosynthesis in Aquatic Ecosystems, 2nd edn. Cambridge University Press.

See Also

[propagate_spectrum](#), [band_irradiance](#), [jerlov_Kd](#)

Examples

```
E0 <- c(10, 20, 30)
Kd  <- c(0.01, 0.05, 0.20)
lam <- c(450, 550, 650)
attenuate_spectrum(E0, Kd, depths = c(0, 10, 25), lambda = lam)
```

band_irradiance	<i>Band-integrated irradiance at one or more depths</i>
-----------------	---

Description

Propagates a spectrum from a known depth to one or more target depths via Beer-Lambert attenuation and integrates the result over a wavelength band. Works in both directions: set from > depths to recover shallower values (including the surface) from a deeper measurement.

Usage

```
band_irradiance(E_lambda, ...)

## S3 method for class 'numeric'
band_irradiance(
  E_lambda,
  Kd_lambda,
  lambda,
  depths,
  from = 0,
  lambda_min = NULL,
  lambda_max = NULL,
  photonic = FALSE,
  molar_unit = "umol",
  binwidth = NULL,
  allow_above_surface = FALSE,
  ...
)

## S3 method for class 'lux_spectrum'
band_irradiance(E_lambda, Kd_lambda, depths, lambda = NULL, ...)
```

Arguments

E_lambda	Spectral irradiance ($W\ m^{-2}\ nm^{-1}$) at depth from. Length-N vector, or a lux_spectrum object.
...	Ignored.
Kd_lambda	Diffuse attenuation coefficients (1/m) at the same wavelengths. Length-N vector.
lambda	Wavelengths in nm. Length-N vector (not needed when E_lambda is a lux_spectrum).
depths	Target depth(s) in metres. Scalar or vector.
from	Depth of the known spectrum in metres. Default 0 (surface). Set to a positive value to propagate from a subsurface measurement.
lambda_min	Lower wavelength bound in nm (inclusive). Defaults to min(lambda).
lambda_max	Upper wavelength bound in nm (inclusive). Defaults to max(lambda).
photonic	Logical; if TRUE the result is returned in $\mu\text{mol photons m}^{-2}\ \text{s}^{-1}$ rather than $W\ m^{-2}$. Default FALSE.
molar_unit	Molar unit for photonic output. Default "umol".
binwidth	Wavelength bin width in nm. Inferred from lambda if not supplied.
allow_above_surface	Logical; permits target depths above the surface (negative values). Default FALSE.

Details

Two operations are applied in sequence. First, Beer-Lambert propagation at each wavelength:

$$E(z, \lambda) = E_0(\lambda) \exp[-K_d(\lambda)(z - z_0)]$$

where z_0 is the depth of the known spectrum (from) and $K_d(\lambda)$ is the diffuse attenuation coefficient. Second, the propagated spectrum is integrated over the specified wavelength window $[\lambda_{\min}, \lambda_{\max}]$:

$$E_{\text{band}}(z) = \int_{\lambda_{\min}}^{\lambda_{\max}} E(z, \lambda) d\lambda$$

approximated as $\sum_i E(z, \lambda_i) \Delta\lambda$ over bins within the window. See [propagate_spectrum](#) for the full Beer-Lambert documentation.

Value

A data.frame with columns:

depth Target depth in metres.

E Band-integrated irradiance in $W\ m^{-2}$ (or $\mu\text{mol}\ m^{-2}\ s^{-1}$ when photonic = TRUE).

See Also

[propagate_spectrum](#), [par_irradiance](#)

Examples

```
sp <- solar_irradiance("clear_noon")
keep <- sp$wavelength >= 350 & sp$wavelength <= 700
lam <- sp$wavelength[keep]
E0 <- sp$irradiance[keep]
Kd <- jerlov_Kd("II", lam)

# surface to depth profile, 500-600 nm band
band_irradiance(E0, Kd, lam, depths = c(0, 5, 10, 20, 50),
               lambda_min = 500, lambda_max = 600)

# recover surface from a 10 m measurement
E_10m <- E0 * exp(-Kd * 10)
band_irradiance(E_10m, Kd, lam, depths = 0, from = 10)
```

beam_attenuation	<i>Beam attenuation coefficient from absorption and scattering.</i>
------------------	---

Description

The beam (total) attenuation coefficient is the sum of the absorption and scattering coefficients, $c = a + b$ (Johnsen 2012, ch. 5). It governs how a collimated beam — and the contrast of a sighted object — fades with distance, and is always larger than the diffuse attenuation K_d .

Usage

```
beam_attenuation(absorption, scattering)
```

Arguments

absorption	Absorption coefficient a (1/m). Vectorised.
scattering	Scattering coefficient b (1/m). Vectorised.

Value

Beam attenuation coefficient c (1/m).

References

Johnsen S (2012) The Optics of Life. Princeton University Press, ch. 5.

See Also

[single_scattering_albedo](#), [transmittance](#), [visual_range](#)

Examples

```
beam_attenuation(absorption = 0.1, scattering = 0.05) # 0.15
```

broadband2spectrum	<i>Convert a broadband scalar reading into a plausible spectrum.</i>
--------------------	--

Description

Given a single measured value and a reference spectral shape, returns a rescaled spectrum whose forward operator reproduces the input. The shape assumption must be provided by the caller (e.g. from `solar_irradiance()`).

Usage

```
broadband2spectrum(
  value,
  unit = c("lx", "W", "kW", "umol", "radiance_W", "cd"),
  spectrum,
  water_type = NULL,
  depth = 0,
  wavelength_policy = c("error", "trim", "constant")
)
```

Arguments

value	The measured scalar (e.g. 18200 lux, 480 W, 1.2 kW).
unit	One of "lx", "W", "kW", "umol", "radiance_W", "cd".
spectrum	Reference spectral shape as a data frame with columns wavelength and irradiance.
water_type	Optional Jerlov water type string (e.g. "IA"). If supplied, the reference shape is pre-attenuated to depth before rescaling — use when the measurement was taken at depth.
depth	Depth in metres at which the measurement was taken. Only used when water_type is not NULL. Default 0.
wavelength_policy	Policy for wavelengths outside the bundled Jerlov domain when attenuation is requested: "error" (default), explicit restriction to the supported intersection with "trim", or explicit constant endpoint extension with "constant".

Details

Supported units and their forward operators:

"lx" photopic integral via `irradiance2lux()`

"W" broadband integral $\sum E \cdot \Delta\lambda$

"kW" same as "W" divided by 1000

"umol" PAR-band photon flux (400–700 nm) in $\mu\text{mol m}^{-2} \text{s}^{-1}$

"radiance_W" Lambertian radiance: $E = \pi L$, then integrate

"cd" photometric radiance: $L = \text{lux}/\pi$

Value

A data frame with columns wavelength and irradiance, with a `luxR.assumption` attribute recording the provenance.

Examples

```
lam <- seq(400, 700, by = 10)
spec <- data.frame(wavelength = lam, irradiance = rep(1, length(lam)))
broadband2spectrum(480, unit = "W", spectrum = spec)
```

CIE_scotopic

CIE Scotopic Luminous Efficiency Function

Description

The CIE scotopic luminous efficiency function $V'(\lambda)$ for night vision, from CIE 018:2019 Table 2. Tabulated at 1 nm intervals from 380 nm to 780 nm, with its peak at 507 nm. Used by [scotopic_lux](#).

Usage

CIE_scotopic

Format

A data frame with 401 rows and 2 columns:

lambda Wavelength in standard air, in nm (380-780, step 1).

W $V'(\lambda)$, dimensionless sensitivity from 0 to 1.

Details

Generated by data-raw/build_cie_efficiency_data.R from the checksum-pinned CIE CSV and companion metadata. Source URLs, DOI, checksums, retrieval date, standard, and license are stored as data-frame attributes under "luxR.provenance".

Source

International Commission on Illumination (CIE), *CIE spectral luminous efficiency for scotopic vision*, DOI: [doi:10.25039/CIE.DS.gr6w4b5g](https://doi.org/10.25039/CIE.DS.gr6w4b5g), licensed CC BY-SA 4.0.

CIE1931

CIE Photopic Luminous Efficiency Function

Description

The CIE 1931 photopic luminous efficiency function $V(\lambda)$, also called the standard observer luminosity curve, from CIE 018:2019 Table 1. Tabulated at 1 nm intervals from 360 nm to 830 nm, with its peak at 555 nm.

Usage

CIE1931

Format

A data frame with 471 rows and 2 columns:

lambda Wavelength in standard air, in nm (360-830, step 1).

W $V(\lambda)$, dimensionless sensitivity from 0 to 1.

Details

Generated by data-raw/build_cie_efficiency_data.R from the checksum-pinned CIE CSV and companion metadata. Source URLs, DOI, checksums, retrieval date, standard, and license are stored as data-frame attributes under "luxR.provenance".

Source

International Commission on Illumination (CIE), *CIE spectral luminous efficiency for photopic vision*, DOI: [doi:10.25039/CIE.DS.dktna2s3](https://doi.org/10.25039/CIE.DS.dktna2s3), licensed CC BY-SA 4.0.

colour_jnd	<i>Chromatic distance between two illuminants (Vorobyev-Osorio model).</i>
------------	--

Description

Computes the just-noticeable difference (JND) between two spectral stimuli as seen by a named species, using the receptor-noise-limited colour discrimination model of Vorobyev & Osorio (1998). A $\Delta S < 1$ indicates the pair is not reliably discriminable; $\Delta S \geq 1$ indicates discrimination is possible.

Usage

```
colour_jnd(  
  stim1,  
  stim2,  
  lambda,  
  species,  
  receptor = NULL,  
  noise = 0.05,  
  binwidth = NULL  
)
```

Arguments

stim1, stim2	Numeric vectors of spectral irradiance ($\text{W m}^{-2} \text{nm}^{-1}$) on the same lambda grid.
lambda	Wavelength vector in nm.
species	Species name matching species_sensitivities\$species.
receptor	Character vector of receptor class(es) to include (e.g. c("L-cone", "M-cone", "S-cone")). Default NULL uses the species' validated default chromatic channel from species_channels . Explicit receptors must belong to that channel.
noise	Weber fraction(s). Either a single value applied to all receptors, or a named numeric vector with one entry per receptor class. Default 0.05 (5% noise), typical for cones in well-lit conditions.
binwidth	Wavelength bin width in nm. Inferred from lambda spacing if NULL (default).

Details

For each receptor type i , the log-ratio of quantum catches $\Delta f_i = \ln(Q_{2i}/Q_{1i})$ is computed, then the chromatic distance is:

$$\Delta S^2 = \Delta \mathbf{f}^\top \Sigma^{-1} \Delta \mathbf{f} - \frac{(\mathbf{1}^\top \Sigma^{-1} \Delta \mathbf{f})^2}{\mathbf{1}^\top \Sigma^{-1} \mathbf{1}}$$

where $\Sigma = \text{diag}(e_i^2)$ and e_i is the noise (Weber fraction) of receptor i . The formula is general for any number of receptor types ≥ 2 .

Value

Scalar: chromatic distance ΔS in just-noticeable differences (JNDs).

References

- Vorobyev M, Osorio D (1998) Receptor noise as a determinant of colour thresholds. Proc. R. Soc. B 265:351–358.
- Vorobyev M et al. (2001) Colour thresholds and receptor noise: behaviour and physiology compared. Vision Res. 41:639–653.

See Also

[quantum_catch](#), [species_brightness](#), [species_LEF](#)

Examples

```
sp1 <- solar_irradiance("clear_noon")
sp2 <- solar_irradiance("underwater_10m")

# Human trichromat
colour_jnd(sp1$irradiance, sp2$irradiance,
           lambda = sp1$wavelength, species = "Homo sapiens",
           receptor = c("L-cone", "M-cone", "S-cone"))

# Zebrafish tetrachromat (all cones)
colour_jnd(sp1$irradiance, sp2$irradiance,
           lambda = sp1$wavelength, species = "Danio rerio")
```

contrast_at_distance	<i>Heuristic scalar contrast remaining after a sighting path through water.</i>
----------------------	---

Description

Attenuates an inherent contrast over a horizontal (or vertical) viewing path as a teaching/scenario heuristic: $C(r) = C_0 e^{-c_{\text{eff}} r}$ (Johnsen 2012, ch. 5). The beam attenuation c is approximated as $K_d * kd_to_c$; looking up uses $c - K_d$ and looking down $c + K_d$.

Usage

```
contrast_at_distance(
  C0,
  distance,
  Kd,
  kd_to_c = 1.5,
  direction = c("horizontal", "up", "down"),
  beam_c = NULL
)
```

Arguments

<code>C0</code>	Inherent (zero-distance) contrast (achromatic Weber contrast or chromatic ΔS). Scalar or vector.
<code>distance</code>	Finite, non-negative viewing distance(s) in metres. Scalar or vector.
<code>Kd</code>	Diffuse attenuation coefficient at the sighting wavelength (1/m).
<code>kd_to_c</code>	Ratio c/K_d . Default 1.5. See visual_range . Ignored when <code>beam_c</code> is supplied.
<code>direction</code>	Viewing direction: "horizontal" (default), "up", or "down".
<code>beam_c</code>	Optional measured beam attenuation c (1/m), e.g. from <code>beam_attenuation()</code> , used instead of $K_d * kd_to_c$. K_d is still used for the "up"/"down" corrections.

Details

This scalar calculation does not propagate spectra or recompute receptor catches. In particular, applying it to a chromatic JND is not a validated chromatic-vision model; use `detection_range(model = "spectral_path")` for wavelength-resolved propagation.

Value

Heuristic contrast remaining at each distance.

References

Johnsen S (2012) The Optics of Life. Princeton University Press, ch. 5.

See Also

[inherent_contrast](#), [detection_range](#)

Examples

```
contrast_at_distance(C0 = 1, distance = c(0, 5, 10), Kd = 0.06)
```

convert_unit	<i>Convert a lux_spectrum to a different unit</i>
--------------	---

Description

Converts the spectral values in a lux_spectrum between energy and photon-flux units, or between mW and W for radiance spectra. Supported conversions: "W/m2/nm" ↔ "umol/m2/s/nm" / "mmol/m2/s/nm" / "mol/m2/s/nm", and the corresponding "/sr" photonic units. Energy radiance units "W/m2/sr/nm" and "mW/m2/sr/nm" are also interchangeable.

Usage

```
convert_unit(x, to, ...)
```

Arguments

- x A lux_spectrum object.
- to Target unit string from the controlled vocabulary.
- ... Ignored.

Value

A new lux_spectrum with updated unit and E.

Examples

```
lam  <- seq(400, 700, by = 10)
x_W  <- lux_spectrum(rep(1, length(lam)), lam, "irradiance", "W/m2/nm")
x_mol <- convert_unit(x_W, "umol/m2/s/nm")
x_mol$unit
```

degree_of_polarization	<i>Degree of polarization from Stokes parameters</i>
------------------------	--

Description

$$DoP = \sqrt{Q^2 + U^2 + V^2} / I$$

. With V = 0 (the default, and the usual case for the underwater light field, which is essentially linearly polarized) this is the degree of *linear* polarization.

Usage

```
degree_of_polarization(I, Q, U, V = 0)
```

Arguments

I	Total radiance (Stokes I). Must be > 0.
Q, U	Linear polarization Stokes parameters.
V	Circular polarization Stokes parameter. Default 0.

Details

All inputs must be finite numeric vectors. A Stokes vector is physical only when $I > 0$ and $\sqrt{Q^2 + U^2 + V^2} \leq I$. Values exceeding this bound by more than 64 machine epsilons are rejected with a `luxR_polarization_physicality_error`. Values within that round-off tolerance are returned as exactly one.

Value

Degree of polarization in $[0, 1]$. Vectorised. Scalar inputs are broadcast; incompatible non-scalar lengths are rejected.

See Also

[angle_of_polarization](#), [underwater_polarization](#)

Examples

```
degree_of_polarization(I = 1, Q = 0.4, U = 0) # 0.4
```

detectability	<i>Object-vs-background detection summary.</i>
---------------	--

Description

One-stop wrapper that bundles [inherent_contrast](#), [detection_range](#), and a contrast-vs-distance curve into a single `lux_detection` object with `print` and `plot` methods.

Usage

```
detectability(
  object,
  background,
  illuminant,
  lambda,
  Kd,
  kd_to_c = 1.5,
  species = NULL,
  receptor = NULL,
  channel = c("both", "achromatic", "chromatic"),
  contrast_threshold = 0.02,
  jnd_threshold = 1,
```

```

direction = c("horizontal", "up", "down"),
noise = 0.05,
distances = NULL,
binwidth = NULL,
beam_c = NULL,
model = c("scalar_heuristic", "spectral_path"),
veiling = NULL,
max_distance = 1000,
search_points = 1001L,
geometry = c("lambertian", "scalar", "collimated", "custom"),
solid_angle = NULL
)

```

Arguments

object, background	Reflectance spectra (numeric vectors on the lambda grid, or dimensionless reflectance lux_spectrum objects). Values must be finite and in [0, 1].
illuminant	The in-water light field illuminating both (numeric vector or irradiance lux_spectrum in W/m2/nm); e.g. a depth-propagated solar spectrum. Values must be finite and non-negative.
lambda	Finite, positive, strictly increasing, regularly spaced wavelength grid in nm.
Kd	Positive diffuse attenuation coefficient (1/m). The scalar heuristic requires one value at the sighting wavelength, e.g. jerlov_Kd("IA", lambda = 490). The spectral-path model accepts one value or one value per wavelength.
kd_to_c	Ratio c/K_d . Default 1.5. Ignored when beam_c is supplied.
species	Species name in species_sensitivities. Required for the chromatic channel. For the achromatic channel, a species must have a cited default in species_channels . If NULL, achromatic brightness is explicitly treated as total photon radiance without a species visual model.
receptor	Receptor class(es) to include. Default NULL uses the complete validated default for each requested species channel. An explicit selection must be valid for every requested channel.
channel	"both" (default), "achromatic", or "chromatic".
contrast_threshold	Achromatic Weber-contrast threshold. Default 0.02.
jnd_threshold	Chromatic threshold in JNDs. Default 1.
direction	Viewing direction: "horizontal" (default), "up", or "down".
noise	Receptor Weber fraction(s) for the chromatic channel; passed to colour_jnd . Default 0.05.
distances	Optional distance grid (m) for the contrast-vs-distance curve. Default spans 0 to 1.2x the largest detection range.
binwidth	Bin width in nm; inferred from lambda if NULL.
beam_c	Optional measured beam attenuation c (1/m), e.g. from beam_attenuation(), used instead of $K_d * kd_to_c$. For the spectral-path model this is required and must have one value per wavelength.

model	"scalar_heuristic" (default) or "spectral_path". The former exponentially attenuates a scalar contrast/JND. The latter propagates wavelength-resolved radiances and recomputes receptor catches.
veiling	Veiling/path radiance spectrum on the lambda grid. Required for the spectral-path model, in the radiance scale produced from illuminant by geometry (normally W m ⁻² sr ⁻¹ nm ⁻¹).
max_distance	Finite positive maximum search distance in metres. A criterion still detectable there is reported as censored, not infinite.
search_points	Integer number of distances used to bracket the furthest downward threshold crossing in the spectral-path model. Default 1001.
geometry	Irradiance-to-radiance geometry used by the spectral-path model: "lambertian" (default), "scalar", "collimated", or "custom". See irradiance2radiance .
solid_angle	Solid angle in steradians when geometry = "custom".

Value

An object of class `lux_detection`: a list with inherent (zero-distance contrast), range (scenario estimate per channel), curve (data frame of contrast vs distance), status metadata, and the call settings.

References

Johnsen S (2012) The Optics of Life. Princeton University Press, ch. 5.

See Also

[inherent_contrast](#), [detection_range](#), [contrast_at_distance](#)

Examples

```
sp <- solar_irradiance("clear_noon")
lam <- sp$wavelength
grey <- rep(0.3, length(lam))
red <- 0.2 + 0.25 * exp(-((lam - 625) / 35)^2)
d <- detectability(red, grey, sp$irradiance, lam,
                  Kd = jerlov_Kd("IA", lambda = 490), species = "Danio rerio")
d
## Not run: plot(d)
```

detection_range	<i>Scenario estimate of object-background threshold-crossing distance.</i>
-----------------	--

Description

Generalises [visual_range](#) by using the estimated inherent contrast of an object against its background (from [inherent_contrast](#)) rather than assuming a contrast of 1. The default scalar heuristic uses $r = \ln(|C_0|/\text{threshold})/c_{\text{eff}}$. The spectral-path model instead propagates object, background, and veiling radiances at every wavelength and recomputes receptor catches at each evaluated distance.

Usage

```
detection_range(
  object,
  background,
  illuminant,
  lambda,
  Kd,
  kd_to_c = 1.5,
  species = NULL,
  receptor = NULL,
  channel = c("both", "achromatic", "chromatic"),
  contrast_threshold = 0.02,
  jnd_threshold = 1,
  direction = c("horizontal", "up", "down"),
  noise = 0.05,
  binwidth = NULL,
  beam_c = NULL,
  model = c("scalar_heuristic", "spectral_path"),
  veiling = NULL,
  max_distance = 1000,
  search_points = 1001L,
  geometry = c("lambertian", "scalar", "collimated", "custom"),
  solid_angle = NULL
)
```

Arguments

object, background	Reflectance spectra (numeric vectors on the lambda grid, or dimensionless reflectance lux_spectrum objects). Values must be finite and in [0, 1].
illuminant	The in-water light field illuminating both (numeric vector or irradiance lux_spectrum in W/m ² /nm); e.g. a depth-propagated solar spectrum. Values must be finite and non-negative.
lambda	Finite, positive, strictly increasing, regularly spaced wavelength grid in nm.
Kd	Positive diffuse attenuation coefficient (1/m). The scalar heuristic requires one value at the sighting wavelength, e.g. <code>jerlov_Kd("IA", lambda = 490)</code> . The spectral-path model accepts one value or one value per wavelength.
kd_to_c	Ratio c/K_d . Default 1.5. Ignored when beam_c is supplied.
species	Species name in <code>species_sensitivities</code> . Required for the chromatic channel. For the achromatic channel, a species must have a cited default in species_channels . If NULL, achromatic brightness is explicitly treated as total photon radiance without a species visual model.
receptor	Receptor class(es) to include. Default NULL uses the complete validated default for each requested species channel. An explicit selection must be valid for every requested channel.
channel	"both" (default), "achromatic", or "chromatic".

contrast_threshold	Achromatic Weber-contrast threshold. Default 0.02.
jnd_threshold	Chromatic threshold in JNDs. Default 1.
direction	Viewing direction: "horizontal" (default), "up", or "down".
noise	Receptor Weber fraction(s) for the chromatic channel; passed to colour_jnd . Default 0.05.
binwidth	Bin width in nm; inferred from lambda if NULL.
beam_c	Optional measured beam attenuation c (1/m), e.g. from <code>beam_attenuation()</code> , used instead of $Kd * kd_to_c$. For the spectral-path model this is required and must have one value per wavelength.
model	"scalar_heuristic" (default) or "spectral_path". The former exponentially attenuates a scalar contrast/JND. The latter propagates wavelength-resolved radiances and recomputes receptor catches.
veiling	Veiling/path radiance spectrum on the lambda grid. Required for the spectral-path model, in the radiance scale produced from illuminant by geometry (normally $W\ m^{-2}\ sr^{-1}\ nm^{-1}$).
max_distance	Finite positive maximum search distance in metres. A criterion still detectable there is reported as censored, not infinite.
search_points	Integer number of distances used to bracket the furthest downward threshold crossing in the spectral-path model. Default 1001.
geometry	Irradiance-to-radiance geometry used by the spectral-path model: "lambertian" (default), "scalar", "collimated", or "custom". See irradiance2radiance .
solid_angle	Solid angle in steradians when geometry = "custom".

Details

Neither model is an empirically validated prediction of actual organismal detection distance. They omit target angular size, acuity, photon limitation, adaptation changes, behaviour, and observation probability.

The spectral-path scenario treats object and background as coplanar stimuli sharing one water path and one asymptotic veiling radiance. For each stimulus it evaluates

$$L(r, \lambda) = L_v(\lambda) + [L_0(\lambda) - L_v(\lambda)]e^{-c_{\text{eff}}(\lambda)r}.$$

Both stimuli therefore converge toward the supplied veiling spectrum. This is a lightweight single-path model, not a multiple-scattering radiative-transfer solution. Because wavelength-dependent signals can be non-monotonic, the reported spectral distance is the furthest downward threshold crossing bracketed by `search_points`; callers should check resolution sensitivity.

Value

A list with elements achromatic and chromatic giving scenario estimates in metres. A value is 0 when no evaluated distance is detectable, NA when the channel was not requested or remained detectable through `max_distance`. Attributes record per-channel status, model, validation state, and maximum distance.

References

Johnsen S (2012) The Optics of Life. Princeton University Press, ch. 5.

See Also

[inherent_contrast](#), [contrast_at_distance](#), [visual_range](#)

Examples

```
sp <- solar_irradiance("clear_noon")
lam <- sp$wavelength
grey <- rep(0.3, length(lam))
red <- 0.2 + 0.25 * exp(-((lam - 625) / 35)^2)
Kd <- jerlov_Kd("IA", lambda = 490)
detection_range(red, grey, sp$irradiance, lam, Kd = Kd, species = "Danio rerio")
```

fit_Kd	<i>Estimate the diffuse attenuation coefficient from two irradiance readings.</i>
--------	---

Description

Closed-form Beer-Lambert inversion: $K_d = \log(E_1/E_2) / (z_2 - z_1)$. Works for scalar or spectral (vector) inputs, or lux_spectrum objects.

Usage

```
fit_Kd(E1, ...)

## S3 method for class 'numeric'
fit_Kd(E1, z1, E2, z2, ...)

## S3 method for class 'lux_spectrum'
fit_Kd(E1, z1, E2, z2, ...)
```

Arguments

- E1 Irradiance at the shallower depth z1. Numeric vector or lux_spectrum. When E1 is a lux_spectrum, S3 dispatch handles both arguments; passing a numeric E1 with a lux_spectrum E2 is not supported.
- ... Ignored.
- z1 Depth of E1 in metres. Must be < z2.
- E2 Irradiance at the deeper depth z2. Numeric vector or lux_spectrum (only when E1 is also a lux_spectrum).
- z2 Depth of E2 in metres. Must be > z1.

Value

Kd in 1/m. Same length as E1/E2.

References

Kirk JTO (1994) Light and Photosynthesis in Aquatic Ecosystems, 2nd edn. Cambridge University Press.

See Also

[propagate_spectrum](#), [attenuate_spectrum](#), [jerlov_Kd](#)

Examples

```
data(Naples)
fit_Kd(sum(Naples$depth_0m), 0, sum(Naples$depth_10m), 10)
```

fit_sensitivity	<i>Fit a Govardovskii visual-pigment template to measured spectral sensitivity.</i>
-----------------	---

Description

Finds the peak wavelength ($\lambda_{\max}$) that minimises the sum of squared residuals between the Govardovskii (2000) template and the supplied sensitivity measurements. Uses [optimize](#) for an exact single-parameter search, so the result is a continuous estimate rather than a grid point.

Usage

```
fit_sensitivity(
  lambda,
  sensitivity,
  chromophore = c("A1", "A2"),
  lambda_max_range = c(300, 700)
)
```

Arguments

lambda	Wavelengths at which sensitivity was measured (nm).
sensitivity	Measured spectral sensitivity. Normalised to [0, 1] internally; need not be pre-normalised. NA values are silently dropped.
chromophore	"A1" (11-cis retinal, default) or "A2" (11-cis 3,4-didehydroretinal).
lambda_max_range	Length-2 numeric vector giving the search interval for $\lambda_{\max}$ in nm. Default <code>c(300, 700)</code> .

Value

A list with components:

`lambda_max` Best-fit peak wavelength (nm).

`chromophore` Chromophore used.

`SS` Residual sum of squares at the optimum.

`fitted` Data frame with columns `lambda` and `S` giving the template evaluated on the input wavelength grid.

References

Govardovskii VI, Fyhrquist N, Reuter T, Kuzmin DG, Donner K (2000) In search of the visual pigment template. *Visual Neuroscience* 17:509–528.

See Also

[govardovskii_template](#)

Examples

```
lam <- seq(400, 700, by = 10)
true <- govardovskii_template(lam, lambda_max = 530)$S
noisy <- pmax(0, true + rnorm(length(true), 0, 0.02))
fit <- fit_sensitivity(lam, noisy)
fit$lambda_max # should be close to 530
```

`fresnel_reflectance` *Fresnel reflectance at the air-water interface.*

Description

Unpolarised Fresnel reflectance for light crossing a flat air-water boundary, as a function of incidence angle (Johnsen 2012, ch. 5). At normal incidence about 2% of downwelling light is reflected ($R = ((n - 1)/(n + 1))^2$). Going from water to air, total internal reflection ($R = 1$) occurs beyond the critical angle (the edge of [snells_window](#)).

Usage

```
fresnel_reflectance(
  angle = 0,
  n = 1.333,
  from = c("air", "water"),
  component = c("unpolarized", "s", "p")
)
```

Arguments

angle	Angle of incidence from the surface normal, in degrees (0-90). Vectorised.
n	Refractive index of water relative to air. Default 1.333.
from	Incident medium: "air" (default) or "water".
component	Which polarization component to return: "unpolarized" (default; the mean of the two), "s" (perpendicular / TE), or "p" (parallel / TM). The difference between the s- and p-components is why reflection off water partially polarizes light (vanishing for the p-component at Brewster's angle).

Value

Reflectance (0-1); the transmitted fraction is $1 - R$.

References

Johnsen S (2012) The Optics of Life. Princeton University Press, ch. 5.

See Also

[snells_window](#), [underwater_polarization](#)

Examples

```
fresnel_reflectance(0)           # ~0.02 at normal incidence
fresnel_reflectance(60, from = "water") # total internal reflection -> 1
fresnel_reflectance(53.1, component = "p") # ~0 near Brewster's angle
```

from_naples	<i>Load a Naples depth spectrum as a lux_spectrum</i>
-------------	---

Description

Load a Naples depth spectrum as a lux_spectrum

Usage

```
from_naples(depth = c("0m", "5m", "10m"))
```

Arguments

depth	One of "0m", "5m", "10m".
-------	---------------------------

Value

A lux_spectrum with unit "umol/m2/s/nm".

See Also

[Naples](#), [as_lux_spectrum](#)

Examples

```
x <- from_naples("0m")
print(x)
```

from_ocean_optics	<i>Read a calibrated Ocean Optics spectrum as a lux_spectrum</i>
-------------------	--

Description

Ocean Optics text files do not contain a standardized machine-readable unit or calibration state. All physical declarations are therefore required. Use [read_ocean_optics](#) to inspect raw or relative values.

Usage

```
from_ocean_optics(path, quantity = NULL, unit = NULL, calibration = NULL)
```

Arguments

path	Path to the Ocean Optics text file.
quantity	Explicit physical quantity represented by the values.
unit	Explicit unit compatible with quantity.
calibration	Non-empty calibration identifier or description.

Value

A validated lux_spectrum with import provenance in metadata.

See Also

[read_ocean_optics](#), [from_trios](#), [read_spectrum](#)

from_solar	<i>Load a reference solar spectrum as a lux_spectrum</i>
------------	--

Description

Load a reference solar spectrum as a lux_spectrum

Usage

```
from_solar(condition = "clear_noon")
```

Arguments

condition One of "clear_noon", "clear_dawn", "overcast", "underwater_1m", or "underwater_10m".

Value

A lux_spectrum with unit "W/m2/nm".

See Also

[solar_irradiance](#), [as_lux_spectrum](#)

Examples

```
x <- from_solar("clear_noon")
print(x)
```

from_trios	<i>Read a TriOS RAMSES file as calibrated radiance spectra</i>
------------	--

Description

Wraps [read_trios](#). The low-level reader preserves negative instrument values. Construction of non-negative physical radiance spectra therefore either fails (the default) or requires an explicit, recorded zero-floor preprocessing policy.

Usage

```
from_trios(path, negative_policy = c("error", "zero"))
```

Arguments

path Path to a TriOS RAMSES .dat file.
negative_policy Either "error" (default) or "zero".

Value

A named list of radiance lux_spectrum objects.

See Also

[read_trios](#), [read_spectrum](#)

ft2m

Convert feet to metres.

Description

Vectorised; passes through NA.

Usage

ft2m(z)

Arguments

z Depth or distance in feet.

Value

Numeric vector in metres.

See Also

[m2ft](#)

Examples

```
propagate_depth(8260, Kd = 0.062, from = ft2m(72), to = 0)
```

govardovskii_template Govardovskii et al. (2000) visual-pigment template.

Description

Reconstructs a photoreceptor's normalised absorbance spectrum from the peak wavelength alone, using the parametric template of Govardovskii et al. (2000) for A1 and A2 chromophores. This is the standard workhorse of comparative visual physiology.

Usage

```
govardovskii_template(lambda, lambda_max, chromophore = c("A1", "A2"))
```

Arguments

lambda	Wavelength grid in nm (numeric vector).
lambda_max	Peak absorbance wavelength in nm.
chromophore	Chromophore type: "A1" (11-cis retinal, default) or "A2" (11-cis 3,4-didehydroretinal). A2 shifts the peak toward longer wavelengths and broadens the spectrum.

Details

The template models absorbance as the sum of an α -band (main peak) and a β -band (short-wavelength shoulder). The α -band follows the four-parameter expression:

$$S_{\alpha}(\lambda) = \left[e^{A(a-x)} + e^{B(0.922-x)} + e^{C(1.104-x)} + D \right]^{-1}, \quad x = \frac{\lambda_{\max}}{\lambda}$$

where A , B , C , D , and a are chromophore-specific constants (A1: $A = 69.7$, $B = 28$, $C = -14.9$, $D = 0.674$, a wavelength-dependent; A2: $A = 62.7$, $B = 26.5$, $C = -15.6$, $D = 0.530$). The β -band is a Gaussian:

$$S_{\beta}(\lambda) = 0.26 \exp \left[- \left(\frac{\lambda - \lambda_{\beta}}{b} \right)^2 \right]$$

with $\lambda_{\beta} = 189 + 0.315 \lambda_{\max}$ and $b = -40.5 + 0.195 \lambda_{\max}$. The full template is $S(\lambda) = \max(0, S_{\alpha} + S_{\beta})$, normalised to a peak of 1.

Value

A data frame with columns lambda and S (normalised absorbance, 0–1).

References

Govardovskii VI, Fyhrquist N, Reuter T, Kuzmin DG, Donner K (2000) In search of the visual pigment template. Visual Neuroscience 17:509–528.

Examples

```
lam <- seq(300, 750, by = 1)
govardovskii_template(lam, lambda_max = 560)
```

inherent_contrast	<i>Inherent (zero-distance) contrast of an object against its background.</i>
-------------------	---

Description

Computes the contrast a viewer perceives between an object and its background at zero viewing distance, from their reflectance spectra and the in-water illuminant. Two channels are available: an **achromatic** (brightness) Weber contrast and a **chromatic** distance (ΔS , in JNDs) from the Vorobyev-Osorio model. Object and background radiances are formed as reflectance times illuminant.

Usage

```
inherent_contrast(
  object,
  background,
  illuminant,
  lambda,
  species = NULL,
  receptor = NULL,
  noise = 0.05,
  channel = c("both", "achromatic", "chromatic"),
  binwidth = NULL
)
```

Arguments

object, background	Reflectance spectra (numeric vectors on the lambda grid, or dimensionless reflectance lux_spectrum objects). Values must be finite and in [0, 1].
illuminant	The in-water light field illuminating both (numeric vector or irradiance lux_spectrum in W/m2/nm); e.g. a depth-propagated solar spectrum. Values must be finite and non-negative.
lambda	Finite, positive, strictly increasing, regularly spaced wavelength grid in nm.
species	Species name in species_sensitivities. Required for the chromatic channel. For the achromatic channel, a species must have a cited default in species_channels . If NULL, achromatic brightness is explicitly treated as total photon radiance without a species visual model.
receptor	Receptor class(es) to include. Default NULL uses the complete validated default for each requested species channel. An explicit selection must be valid for every requested channel.
noise	Receptor Weber fraction(s) for the chromatic channel; passed to colour_jnd . Default 0.05.
channel	"both" (default), "achromatic", or "chromatic".
binwidth	Bin width in nm; inferred from lambda if NULL.

Value

A list with elements achromatic (Weber contrast) and chromatic (ΔS in JNDs); unrequested channels are NA.

References

Johnsen S (2012) The Optics of Life: A Biologist's Guide to Light in Nature. Princeton University Press. (Ch. 5: contrast and sighting distance.)

See Also

[contrast_at_distance](#), [detection_range](#), [colour_jnd](#), [weber_contrast](#)

Examples

```
sp <- solar_irradiance("clear_noon")
lam <- sp$wavelength
grey <- rep(0.3, length(lam))
red <- 0.2 + 0.25 * exp(-((lam - 625) / 35)^2)
inherent_contrast(red, grey, sp$irradiance, lam, species = "Danio rerio")
```

irradiance2lux

Provide lux values for measures of spectral irradiance.

Description

Computes photopic illuminance (lux) by weighting spectral irradiance with the CIE 1931 $V(\lambda)$ luminous efficiency function. The default LEF argument can be swapped for CIE_scotopic to obtain scotopic lux, or for any user-supplied spectral sensitivity curve.

Usage

```
irradiance2lux(irradiance, ...)

## Default S3 method:
irradiance2lux(irradiance, ...)

## S3 method for class 'numeric'
irradiance2lux(
  irradiance,
  lambda,
  photonic = FALSE,
  molar_unit = "photons",
  total = TRUE,
  LEF = CIE1931,
  binwidth = NULL,
  integration = c("rectangle", "trapezoid"),
```

```

    verbose = FALSE,
    ...
)

## S3 method for class 'lux_spectrum'
irradiance2lux(
  irradiance,
  total = TRUE,
  LEF = CIE1931,
  integration = c("rectangle", "trapezoid"),
  verbose = FALSE,
  ...
)

```

Arguments

irradiance	Spectral irradiance ($W\ m^{-2}\ nm^{-1}$ or $photons\ m^{-2}\ s^{-1}\ nm^{-1}$) at each wavelength bin, or a lux_spectrum object whose quantity is "irradiance". Photonic lux_spectrum inputs are converted to energy units before photometric integration; radiance and reflectance spectra are rejected.
...	Ignored.
lambda	Wavelength in nm (not needed for lux_spectrum).
photonic	Logical; if TRUE, input is in photon or molar units — specify which via molar_unit. Note: photonic = TRUE means raw photon counts only in par_irradiance , and molar units only in n2W_spec_irradiance .
molar_unit	If photonic, unit string such as "umol". Defaults to "photons".
total	Logical; sum per-bin values into a scalar. Defaults to TRUE.
LEF	Luminous efficiency function data frame. Defaults to CIE1931.
binwidth	Wavelength bin width in nm for rectangular integration. For numeric input, inferred from a regularly spaced wavelength grid when NULL. A single-bin input requires an explicit width. Must be NULL for trapezoidal integration because the wavelength intervals determine the integration weights.
integration	Integration method: "rectangle" (default) retains the existing bin-centre sum and requires a regular grid; "trapezoid" supports strictly increasing regular or irregular numeric grids with at least two wavelengths.
verbose	Logical; emit diagnostic messages. Defaults to FALSE.

Details

Photopic illuminance is a spectrally weighted integral of irradiance using the CIE 1931 photopic luminous efficiency function $V(\lambda)$. The maximum luminous efficacy $K_m = 683\ lm\ W^{-1}$ occurs at 555 nm:

$$E_v = 683 \int_{360}^{830} E(\lambda) V(\lambda) d\lambda$$

With the default rectangular method, the integral is approximated as $E_v \approx 683 \sum_i E(\lambda_i) V(\lambda_i) \Delta\lambda$. For irregular numeric wavelength grids, callers can explicitly select composite trapezoidal integration. This treats the weighted spectrum as piecewise linear between measured wavelengths and

integrates only over $[\min(\lambda), \max(\lambda)]$; it does not infer outer bin edges. $V(\lambda)$ values are taken from the LEF data frame by linear interpolation at each measured wavelength. Wavelengths outside the range of the supplied LEF receive zero weight.

Value

A scalar lux value (total = TRUE) or numeric vector.

References

CIE Publication 15: Colorimetry, 3rd edn (2004). Commission Internationale de l'Eclairage, Vienna.

See Also

[lux2irradiance](#), [scotopic_lux](#), [par_irradiance](#)

Examples

```
utils::data(Naples)
irradiance2lux(Naples$depth_0m, Naples$wv, photonic = TRUE, molar_unit = "umol")
irradiance2lux(c(0.2, 0.8, 0.4), c(500, 515, 540),
               integration = "trapezoid")
```

irradiance2radiance	<i>Convert irradiance to radiance under a geometry assumption.</i>
---------------------	--

Description

The relationship between irradiance E and radiance L depends on the angular distribution of the light field. The caller must supply the geometry; the function refuses to silently apply the wrong identity.

Usage

```
irradiance2radiance(
  E,
  geometry = c("lambertian", "scalar", "collimated", "custom"),
  solid_angle = NULL
)
```

Arguments

E	Irradiance value(s) in $W\ m^{-2}$ (or any energy unit). Vectorised.
geometry	One of: "lambertian" Planar (vector) irradiance from a Lambertian surface, or equivalently from an isotropic hemisphere of radiance — the cosine-weighted integral $\int L \cos \theta\ d\Omega = \pi L$. This is what a cosine collector (e.g. a downwelling-irradiance sensor) measures: $L = E/\pi$. Default.

"scalar" Scalar irradiance (fluence rate) over a 2π sr hemisphere — the *un*-weighted integral $\int L d\Omega = 2\pi L$, as measured by a spherical collector. Scalar and vector irradiance are different physical quantities, not two flavours of the same measurement (Johnsen 2012, ch. 2): $L = E/(2\pi)$.

"collimated" Normal-incidence collimated beam: $L = E$ (solid angle $\rightarrow 0$).

"custom" User-specified solid angle Ω : $L = E/\Omega$. Requires `solid_angle`.

`solid_angle` Finite solid angle in $(0, 4\pi]$ steradians. Required when `geometry = "custom"` and rejected for other geometries.

Value

Radiance in $W m^{-2} sr^{-1}$ (same numeric unit as `E`).

References

Johnsen S (2012) The Optics of Life: A Biologist's Guide to Light in Nature. Princeton University Press. (Ch. 2: vector vs scalar irradiance.)

See Also

[radiance2irradiance](#)

Examples

```
irradiance2radiance(pi)           # lambertian: L = 1
irradiance2radiance(2 * pi, "scalar") # scalar: L = 1
irradiance2radiance(10, "collimated") # L = E
irradiance2radiance(10, "custom", solid_angle = 2) # L = 5
```

jerlov_Kd

Diffuse attenuation coefficient for Jerlov optical water types.

Description

Returns `Kd(lambda)` for the specified Jerlov water type from the bundled `jerlov_types` dataset (Jerlov 1976; Solonenko & Mobley 2015). Optionally interpolates onto a user-supplied wavelength grid.

Usage

```
jerlov_Kd(
  type = "IA",
  lambda = NULL,
  interp = c("linear", "spline", "nearest"),
  extrapolation = c("error", "constant")
)
```

Arguments

type	Jerlov water type. One of "I", "IA", "IB", "II", "III", "C1", "C2", "C3". Case-insensitive.
lambda	Wavelengths in nm to interpolate Kd onto. If NULL (default), returns the full tabulated rows as a data frame.
interp	Interpolation method when lambda is supplied. One of "linear" (default), "spline", or "nearest". Interpolation is applied only within the supported wavelength range.
extrapolation	Out-of-range policy. The default "error" raises a luxR_jerlov_range_error. Set to "constant" to explicitly extend the nearest endpoint value; the choice is recorded in the returned vector's "luxR.jerlov" attribute.

Details

The bundled Jerlov table supports 350–700 nm. Values outside that domain are not measured by this dataset and therefore fail by default. Constant endpoint extension is available only as an explicit modelling assumption; linear or spline extrapolation is not supported.

Value

If lambda is NULL: a data frame with columns type, lambda, Kd and dataset metadata attributes.
If lambda is supplied: a numeric vector of Kd values (1/m) with a "luxR.jerlov" provenance attribute recording the water type, interpolation and extrapolation policies, supported and requested ranges, source checksum, build commit, and model version.

References

Jerlov NG (1976) Marine Optics. Elsevier, Amsterdam.
Solonenko MG, Mobley CD (2015) Inherent and apparent optical properties of Jerlov water types. Applied Optics 54(17):5392-5401.

Examples

```
jerlov_Kd("IA")
Kd_IA <- jerlov_Kd("IA", lambda = c(400, 450, 500, 550, 600))
as.numeric(Kd_IA)
attr(Kd_IA, "luxR.jerlov")[c("type", "interp", "extrapolation",
                             "supported_wavelength_range_nm")]

lam <- seq(400, 700, by = 10)
Kd <- jerlov_Kd("II", lambda = lam)
attenuate_spectrum(rep(1, length(lam)), Kd, depths = c(10, 25, 50),
                   lambda = lam)
```

jerlov_types

Diffuse attenuation coefficients for Jerlov optical water types.

Description

Long-format table of $K_d(\lambda)$ values for five oceanic and three coastal water types from Jerlov (1976) and Solonenko & Mobley (2015). Attributes record the supported wavelength range and spacing, a table checksum, source artifact path, build commit, model version, and package version.

Usage

```
jerlov_types
```

Format

A data frame with 120 rows and 3 columns:

type Jerlov water type (character): "I", "IA", "IB", "II", "III" (oceanic, clearest to most turbid) and "C1", "C2", "C3" (coastal).

lambda Wavelength in nm (350 to 700, 25 nm steps).

Kd Diffuse attenuation coefficient in 1/m.

References

Jerlov NG (1976) Marine Optics. Elsevier, Amsterdam.

Solonenko MG, Mobley CD (2015) Inherent and apparent optical properties of Jerlov water types. Applied Optics 54(17):5392-5401.

Examples

```
data(jerlov_types)
subset(jerlov_types, type == "IA")
```

light_at_depth

In-water downwelling spectral irradiance at a single depth.

Description

Convenience wrapper that takes one of the bundled reference solar spectra, converts it to $W m^{-2} nm^{-1}$, and propagates it from its recorded reference depth to a single absolute depth in a Jerlov water type via [jerlov_Kd](#) and [propagate_spectrum](#). This is the "solar condition + water type + depth -> spectrum at depth" operation the Shiny app's colour-discrimination and detection tabs share.

Usage

```
light_at_depth(
  condition = "clear_noon",
  water_type = "IA",
  depth = 0,
  surface_source = c("direct", "diffuse"),
  surface_angle = 30,
  refractive_index = 1.333,
  wavelength_policy = c("error", "trim", "constant")
)
```

Arguments

condition	Solar condition passed to from_solar (e.g. "clear_noon"). Default "clear_noon".
water_type	Jerlov water type passed to jerlov_Kd (e.g. "IA"). Default "IA".
depth	Absolute target depth in metres below the water surface. Must be length-1, non-negative, and not shallower than an in-water source's recorded reference depth. Default 0 (surface).
surface_source	Surface illumination model used when condition is referenced in air: "direct" (default) or "diffuse".
surface_angle	Incidence angle in degrees from the surface normal for surface_source = "direct". Default 30, matching the worked vignette.
refractive_index	Refractive index of water relative to air used by the Fresnel surface model. Default 1.333.
wavelength_policy	Policy for source wavelengths outside the bundled Jerlov domain (350–700 nm): "error" (default) fails, "trim" explicitly restricts the calculation to the supported intersection, and "constant" explicitly extends endpoint Kd values.

Details

An above-surface source is multiplied by [surface_transmittance](#) before water-column attenuation. The default is a flat interface and direct illumination incident 30 degrees from vertical. An in-water source is unchanged when depth equals its reference depth. For deeper targets, only the additional water-column distance is applied. Shallower targets are rejected because reconstructing them would require an explicit inverse-propagation model. Water-column propagation is the lightweight spectral Beer-Lambert model: it assumes a depth-invariant Jerlov $K_d(\lambda)$, propagates wavelength bins independently, and is not a multiple-scattering or angular radiative-transfer calculation. No universal maximum depth is implied; the selected water type must remain representative of the whole path.

Value

A data frame with columns lambda (nm) and irradiance (downwelling spectral irradiance at depth, in $W m^{-2} nm^{-1}$). The "luxR.jerlov" attribute records the wavelength policy and data provenance.

See Also

[from_solar](#), [jerlov_Kd](#), [propagate_spectrum](#)

Examples

```
head(light_at_depth("clear_noon", "IA", depth = 5,
                    wavelength_policy = "trim"))
```

lux_spectrum	<i>Create a lux_spectrum object</i>
--------------	-------------------------------------

Description

Constructs a validated lux_spectrum — the core spectral container used throughout luxR. The object carries spectral values, wavelength grid, physical quantity, unit, bin width, and free-form metadata together, so unit conversions and arithmetic operations stay consistent.

Usage

```
lux_spectrum(
  E,
  lambda,
  quantity = "irradiance",
  unit = "W/m2/nm",
  binwidth = NULL,
  meta = list()
)
```

Arguments

E	Numeric vector of spectral values, one per wavelength bin.
lambda	Numeric vector of bin-centre wavelengths in nm.
quantity	Physical quantity: "irradiance", "radiance", or "reflectance".
unit	Unit string from the controlled vocabulary. See Details.
binwidth	Bin width in nm. Inferred from the wavelength spacing when NULL; defaults to 1 for a single-bin spectrum. An explicit value must be finite, positive, scalar, and match the regular wavelength spacing.
meta	Named list of free-form metadata (depth, instrument, etc.).

Details

Valid unit strings: "W/m2/nm", "W/m2/sr/nm", "mW/m2/sr/nm", "umol/m2/s/nm", "mmol/m2/s/nm", "mol/m2/s/nm", "umol/m2/s/sr/nm", "mmol/m2/s/sr/nm", "mol/m2/s/sr/nm", "dimensionless".

Both spectral vectors must be non-empty, numeric, finite, and equal in length. Wavelengths must be strictly increasing, unique, and regularly spaced. Irradiance and radiance values must be non-negative; reflectance values must lie in $[0, 1]$. Units without "/sr" describe irradiance, units with "/sr" describe radiance, and "dimensionless" is valid only for reflectance.

S3 methods available on lux_spectrum objects: print(), summary(), plot(), as.data.frame(), [(wavelength subsetting), and the arithmetic operators +, -, *, /. A list of lux_spectrum objects can be plotted with [plot_spectra](#).

Value

An S3 object of class "lux_spectrum".

See Also

[as_lux_spectrum](#), [convert_unit](#), [plot_spectra](#)

Examples

```
lam <- seq(400, 700, by = 10)
x <- lux_spectrum(rep(1, length(lam)), lam, "irradiance", "W/m2/nm")
print(x)
```

lux2irradiance

Distribute integrated lux across a reference spectral shape.

Description

lux is a single scalar; a spectrum has many degrees of freedom. This function takes a reference *shape* and rescales it so its photopic integral equals lx. The shape assumption must be supplied by the caller.

Usage

```
lux2irradiance(
  lx,
  spectrum,
  lambda = NULL,
  LEF = CIE1931,
  binwidth = NULL,
  integration = c("rectangle", "trapezoid")
)
```

Arguments

lx	Target illuminance in lux.
spectrum	Reference spectral shape. Either a data frame with columns lambda and irradiance, or a bare numeric vector (in which case lambda must be supplied separately).
lambda	Wavelength grid in nm. Required when spectrum is a bare vector; ignored when spectrum is a data frame.
LEF	Luminous efficiency function. Defaults to CIE1931.
binwidth	Wavelength bin width in nm. Inferred from a regular multi-bin wavelength grid when NULL; required for a single bin under rectangular integration. Must be NULL for trapezoidal integration.
integration	Integration method passed to irradiance2lux : "rectangle" (default) or "trapezoid".

Value

Numeric vector of rescaled spectral irradiance (same length as the wavelength grid).

Examples

```
spec <- data.frame(lambda = CIE1931$lambda, irradiance = CIE1931$W)
irr <- lux2irradiance(5000, spec)
```

m2ft	<i>Convert metres to feet.</i>
------	--------------------------------

Description

Vectorised; passes through NA.

Usage

```
m2ft(z)
```

Arguments

z	Depth or distance in metres.
---	------------------------------

Value

Numeric vector in feet.

See Also

[ft2m](#)

Examples

```
m2ft(30)
```

michelson_contrast	<i>Michelson contrast</i>
--------------------	---------------------------

Description

Computes the Michelson (sinusoidal) contrast: $(I_{\text{bright}} - I_{\text{dark}})/(I_{\text{bright}} + I_{\text{dark}})$. Values range from 0 (no contrast) to 1 (maximum contrast). Commonly used for periodic patterns such as gratings.

Usage

```
michelson_contrast(I_bright, I_dark)
```

Arguments

I_bright	Luminance or radiance of the brighter region (scalar or vector). Must be $\geq I_{\text{dark}}$.
I_dark	Luminance or radiance of the darker region (scalar or vector).

Value

Contrast value(s) in $[0, 1]$.

See Also

[weber_contrast()]

Examples

```
michelson_contrast(I_bright = 8, I_dark = 2)
```

n2W_spec_irradiance	<i>Convert photon-count spectral irradiance to $\text{W m}^{-2} \text{nm}^{-1}$.</i>
---------------------	---

Description

Converts per-bin photon flux (photons $\text{m}^{-2} \text{s}^{-1} \text{nm}^{-1}$, or molar equivalents) to energy-based spectral irradiance in $\text{W m}^{-2} \text{nm}^{-1}$. The inverse operation is [W2mol_spec_irradiance](#).

Usage

```
n2W_spec_irradiance(value, ...)

## S3 method for class 'numeric'
n2W_spec_irradiance(
  value,
  lambda,
  photonic = FALSE,
  molar_unit = "photons",
  ...
)

## S3 method for class 'lux_spectrum'
n2W_spec_irradiance(value, ...)
```

Arguments

value	Spectral irradiance in $\text{photons m}^{-2} \text{s}^{-1} \text{nm}^{-1}$ (or molar units), or a <code>lux_spectrum</code> with a photonic irradiance or radiance unit.
...	Ignored.
lambda	Wavelength in nm for each bin.
photonic	Logical; if TRUE, value is in molar units (μmol , mmol, or $\text{mol m}^{-2} \text{s}^{-1} \text{nm}^{-1}$) — specify which via <code>molar_unit</code> . Defaults to FALSE. Note: <code>photonic = TRUE</code> means raw photon counts in par_irradiance , and photon-or-molar units in irradiance2lux .
molar_unit	Molar unit string (e.g. "umol"). Defaults to "photons".

Details

Each photon at wavelength λ carries energy $E = hc/\lambda$. Spectral irradiance in $\text{W m}^{-2} \text{nm}^{-1}$ is therefore the photon flux N ($\text{photons m}^{-2} \text{s}^{-1} \text{nm}^{-1}$) multiplied by the per-photon energy:

$$E(\lambda) = N(\lambda) \cdot \frac{hc}{\lambda}$$

where $h = 6.626 \times 10^{-34} \text{ J s}$ (Planck constant) and $c = 2.998 \times 10^8 \text{ m s}^{-1}$ (speed of light). The constant hc in nm-compatible units is $1.98645 \times 10^{-16} \text{ J nm}$, the value used internally.

Value

Numeric vector of spectral irradiance in $\text{W m}^{-2} \text{nm}^{-1}$, or a `lux_spectrum` with `unit = "W/m2/nm"` or `"W/m2/sr/nm"` according to the input dimensionality.

Examples

```
utils::data(Naples)
n2W_spec_irradiance(Naples$depth_0m, Naples$wv, photonic = TRUE, molar_unit = "umol")
```

 Naples

Example downwelling irradiance from Mare Chiaro, Gulf of Naples

Description

A small set of downwelling spectral irradiance measurements made at three depths at Mare Chiaro in the Gulf of Naples, Italy. This is **example field data** — a single site on particular days — provided to illustrate working with measured spectra; it is not a general reference spectrum (cf. the ASTM [solar_spectra](#)). Values are in micromol photons per square metre per second per nm ($\mu\text{mol m}^{-2} \text{s}^{-1} \text{nm}^{-1}$); wavelength bins are approximately 5 nm wide.

Usage

Naples

Format

A data frame with 103 rows and 4 columns:

wv Bin-centre wavelength in nm.

depth_0m Irradiance at the surface (0 m depth).

depth_5m Irradiance at 5 m depth.

depth_10m Irradiance at 10 m depth.

Source

Field measurements of downwelling spectral irradiance at Mare Chiaro, Gulf of Naples, Italy, attributed to M. J. Bok and J. D. Kirwan (2021, unpublished data). The bundled object is regenerated from an explicitly labelled legacy canonical snapshot; original acquisition metadata and redistribution terms are not documented.

Examples

```
data(Naples)
irradiance2lux(Naples$depth_0m, Naples$wv, photonic = TRUE, molar_unit = "umol")
```

 par_fraction

Fraction of spectral irradiance in the PAR window.

Description

Returns the ratio of band-integrated irradiance over 400–700 nm to the total integrated irradiance across the full wavelength range supplied. Useful for characterising how spectrally narrowed a light field has become (e.g. at depth, where red wavelengths are preferentially removed).

Usage

```
par_fraction(irradiance, ...)

## S3 method for class 'numeric'
par_fraction(irradiance, lambda, binwidth = NULL, ...)

## S3 method for class 'lux_spectrum'
par_fraction(irradiance, ...)
```

Arguments

irradiance	Spectral irradiance (any consistent energy unit per nm), or a lux_spectrum object.
...	Ignored.
lambda	Wavelength vector in nm.
binwidth	Bin width in nm. Inferred from lambda if NULL.

Value

Numeric scalar in (0, 1].

Examples

```
sp <- solar_irradiance("clear_noon")
par_fraction(sp$irradiance, sp$wavelength)
```

par_irradiance	<i>PAR irradiance from a spectral irradiance measurement.</i>
----------------	---

Description

Integrates spectral irradiance over the photosynthetically active range 400–700 nm and returns the result in $\mu\text{mol photons m}^{-2} \text{s}^{-1}$. Accepts energy units ($\text{W m}^{-2} \text{nm}^{-1}$), raw photon counts, or a lux_spectrum object in any supported unit.

Usage

```
par_irradiance(irradiance, ...)

## S3 method for class 'numeric'
par_irradiance(irradiance, lambda, photonic = FALSE, binwidth = NULL, ...)

## S3 method for class 'lux_spectrum'
par_irradiance(irradiance, ...)
```

Arguments

irradiance	Spectral irradiance ($\text{W m}^{-2} \text{ nm}^{-1}$ or $\text{photons s}^{-1} \text{ m}^{-2} \text{ nm}^{-1}$) at each bin, or a <code>lux_spectrum</code> object.
...	Ignored.
lambda	Wavelength vector in nm (not needed for <code>lux_spectrum</code>).
photonic	Logical. If FALSE (default), input is in $\text{W m}^{-2} \text{ nm}^{-1}$. If TRUE, input is in <i>raw</i> photon counts ($\text{photons m}^{-2} \text{ s}^{-1} \text{ nm}^{-1}$), not molar units. For molar-unit ($\mu\text{mol/m}^2\text{/s/nm}$) input, either wrap in a <code>lux_spectrum</code> or convert to $\text{W/m}^2\text{/nm}$ first with <code>n2W_spec_irradiance</code> . Note: <code>photonic = TRUE</code> means molar units in <code>n2W_spec_irradiance</code> and photon-or-molar units in <code>irradiance2lux</code> .
binwidth	Bin width in nm. Inferred from <code>lambda</code> if NULL.

Details

PAR is the photon flux integrated over the photosynthetically active range 400–700 nm, expressed in $\mu\text{mol photons m}^{-2} \text{ s}^{-1}$. When input is in energy units ($\text{W m}^{-2} \text{ nm}^{-1}$), each bin is first converted to photon flux via $N = E\lambda/(hc)$, then the total is divided by Avogadro's constant scaled to μmol :

$$Q_{\text{PAR}} = \frac{1}{N_A \times 10^{-6}} \int_{400}^{700} \frac{E(\lambda) \lambda}{hc} d\lambda$$

where $N_A = 6.022 \times 10^{23} \text{ mol}^{-1}$ and $hc = 1.98645 \times 10^{-16} \text{ J nm}$. The result is in $\mu\text{mol photons m}^{-2} \text{ s}^{-1}$.

Value

Scalar PAR irradiance in $\mu\text{mol photons m}^{-2} \text{ s}^{-1}$.

References

Thimijan RW, Heins RD (1983) HortScience 18:818-822.

See Also

`par_fraction`, `irradiance2lux`, `band_irradiance`

Examples

```
sp <- solar_irradiance("clear_noon")
par_irradiance(sp$irradiance, sp$wavelength)
```

photic_depth	<i>Depth at which a given fraction of surface light remains.</i>
--------------	--

Description

Solves $\exp(-K_d * z) = \text{fraction}$ for z , giving $-\log(\text{fraction}) / K_d$. The default 1

Usage

```
photic_depth(Kd, fraction = 0.01)
```

Arguments

Kd	Diffuse attenuation coefficient(s) in 1/m.
fraction	Fraction of surface irradiance. Must be in (0, 1). Default 0.01 (1%).

Value

Depth(s) in metres at which fraction of surface light remains.

Examples

```
photic_depth(0.06)
photic_depth(0.06, fraction = 0.1)
```

photon2W	<i>Convert photon counts to watts at a given wavelength.</i>
----------	--

Description

Uses $E_{\text{photon}} = hc / \lambda$, with $hc = 1.98644585714893e-16$ J·nm. Vectorised over both n and λ .

Usage

```
photon2W(n, lambda)
```

Arguments

n	Photon count(s) in $\text{photons } m^{-2} s^{-1} nm^{-1}$.
lambda	Wavelength(s) in nm.

Value

Numeric vector of spectral irradiance in $W m^{-2} nm^{-1}$.

See Also[W2photon](#)**Examples**

```
photon2W(1e15, 555)
```

plot_spectra	<i>Plot a list of lux_spectrum objects as overlaid lines.</i>
--------------	---

Description

Draws all spectra in `x` on a single plot with a shared wavelength axis. Line colours cycle through the default palette. Names of the list (or `meta$label` if present) are used for the legend. Additional arguments in `...` are passed to the initial `plot()` call.

Usage

```
plot_spectra(x, ...)
```

Arguments

<code>x</code>	A named or unnamed list of <code>lux_spectrum</code> objects.
<code>...</code>	Additional arguments passed to <code>plot</code> and <code>lines</code> .

Value

`invisible(x)` — called for its side effect (plot).

polarization_contrast	<i>Polarization contrast between a target and its background</i>
-----------------------	--

Description

How distinct a target's polarization is from its background, for a polarization-sensitive viewer. Each polarization state is mapped to its normalised linear-polarization (Stokes) coordinates

$$q = DoP \cos 2\psi, \quad u = DoP \sin 2\psi,$$

where ψ is the angle of polarization; the **polarization distance** is the Euclidean separation in this plane,

$$\Delta P = \sqrt{(q_t - q_b)^2 + (u_t - u_b)^2}.$$

The factor of two in the angle correctly handles the 180-degree periodicity of polarization orientation, so orthogonal e-vectors (90 degrees apart) at equal degree of polarization give the maximal angular contrast. This is the geometric core of How & Marshall's (2014) polarization-distance model.

Usage

```
polarization_contrast(target_dop, target_aop, background_dop, background_aop)
```

Arguments

target_dop, target_aop
Degree of polarization (0-1) and angle of polarization (degrees) of the target. Vectorised.

background_dop, background_aop
Degree and angle of polarization of the background. Vectorised. Scalars are broadcast; implicit partial recycling is rejected.

Value

A data frame with columns delta_dop (absolute difference in degree of polarization), delta_aop (axial angular difference in degrees, $[0, 90]$), and distance (the polarization distance ΔP).

References

How MJ, Marshall NJ (2014) Polarization distance: a framework for modelling object detection by polarization vision systems. Proceedings of the Royal Society B 281:20131632.

See Also

[underwater_polarization](#), [degree_of_polarization](#), [weber_contrast](#)

Examples

```
# A horizontally polarized target against a weakly polarized background:
polarization_contrast(target_dop = 0.5, target_aop = 0,
                      background_dop = 0.1, background_aop = 45)
```

propagate_depth	<i>Propagate a known irradiance to any other depth (bidirectional).</i>
-----------------	---

Description

Computes $E(\text{to}) = E_{\text{known}} * \exp(-K_d * (\text{to} - \text{from}))$. Works forward ($\text{to} > \text{from}$), inverse ($\text{to} < \text{from}$, recovering a shallower value), and between any two depths.

Usage

```
propagate_depth(E_known, Kd, from = 0, to, allow_above_surface = FALSE)
```

Arguments

E_known	Irradiance at depth from. Scalar or vector.
Kd	Diffuse attenuation coefficient in 1/m. Scalar.
from	Depth of the known measurement in metres. Default 0 (surface).
to	Target depth(s) in metres. Scalar or vector.
allow_above_surface	Logical; permits negative absolute depths (above the surface). Default FALSE.

Value

Irradiance at to, in the same units as E_known.

Note

Un-attenuating with a negative depth delta assumes the water column is homogeneous above the measurement point. For stratified water bodies (thermocline, chlorophyll layer) the result will be approximate. Inverse propagation also amplifies measurement noise and can overflow for long or strongly attenuating paths. This model does not cross the air-water interface; use [surface_transmittance](#) explicitly or a high-level workflow such as [light_at_depth](#).

Examples

```
propagate_depth(112400, Kd = 0.062, from = 0, to = 30)
propagate_depth(8260, Kd = 0.062, from = 22, to = 0)
```

propagate_spectrum	<i>Propagate a known spectrum to any other depth(s) (bidirectional).</i>
--------------------	--

Description

Applies Beer-Lambert attenuation at each wavelength independently, propagating a known spectral irradiance from depth from to one or more target depths to. Works in both directions: set to < from to recover a shallower (or surface) spectrum from a deeper measurement, assuming a homogeneous water column.

Usage

```
propagate_spectrum(E_lambda, ...)

## S3 method for class 'numeric'
propagate_spectrum(
  E_lambda,
  Kd_lambda,
  from = 0,
  to,
  format = c("long", "matrix"),
```

```

    lambda = NULL,
    allow_above_surface = FALSE,
    ...
)

## S3 method for class 'lux_spectrum'
propagate_spectrum(
  E_lambda,
  Kd_lambda,
  from = 0,
  to,
  allow_above_surface = FALSE,
  ...
)

```

Arguments

<code>E_lambda</code>	Spectral irradiance at depth from. Length-N vector or a <code>lux_spectrum</code> object.
<code>...</code>	Ignored.
<code>Kd_lambda</code>	Finite, non-negative diffuse attenuation coefficients (1/m). Length-N vector.
<code>from</code>	Finite scalar depth of the known spectrum in metres. Default 0.
<code>to</code>	Finite target depth(s) in metres. Scalar or vector.
<code>format</code>	"long" (default) or "matrix". Ignored for <code>lux_spectrum</code> input.
<code>lambda</code>	Optional wavelength values (nm) for long-format output.
<code>allow_above_surface</code>	Logical; permits negative absolute depths. Default FALSE.

Details

This is the package's principal lightweight propagation tier. It applies $E(z, \lambda) = E(z_0, \lambda) \exp[-K_d(\lambda)(z - z_0)]$, assuming K_d is constant along the path and wavelengths propagate independently. It does not model an angular radiance field, multiple scattering, or spectral redistribution. There is no universal supported depth range: validity depends on whether the supplied K_d represents the entire path. Bundled Jerlov data are limited to 350–700 nm. Inverse propagation is an assumption-heavy reconstruction that amplifies noise.

Value

A data frame, matrix, or named list of `lux_spectrum`.

References

Kirk JTO (1994) *Light and Photosynthesis in Aquatic Ecosystems*, 2nd edn. Cambridge University Press.

Examples

```
E_10m <- c(8, 15, 20)
Kd     <- c(0.02, 0.06, 0.25)
propagate_spectrum(E_10m, Kd, from = 10, to = c(0, 25, 50))
```

quantum_catch	<i>Sensitivity-weighted photon irradiance.</i>
---------------	--

Description

Computes $Q_w = \sum E_p(\lambda)S(\lambda)\Delta\lambda$, where E_p is spectral photon irradiance and S is a dimensionless sensitivity or absorptance curve. The result is a photon irradiance weighted by the supplied curve, not an absolute photon rate for one receptor.

Usage

```
quantum_catch(
  irradiance,
  lambda,
  S,
  input_unit = NULL,
  total = TRUE,
  binwidth = NULL
)
```

Arguments

irradiance	Non-negative spectral irradiance vector, one value per wavelength bin, in the exact unit declared by input_unit.
lambda	Wavelength bin centres in nm.
S	Spectral sensitivity: a data frame with columns lambda and S (normalised 0–1). Accepts output of species_LEF or govardovskii_template directly.
input_unit	Exact unit of irradiance. One of "W/m2/nm", "photon/m2/s/nm", "mol/m2/s/nm", "mmol/m2/s/nm", or "umol/m2/s/nm". Molar inputs are converted using the Avogadro constant.
total	Logical; sum across wavelengths (default TRUE) or return per-bin contributions.
binwidth	Wavelength bin width in nm. Inferred from lambda spacing if NULL (default).

Details

All inputs are converted to raw photon counts before integration, so the returned unit is photons $\text{m}^{-2} \text{s}^{-1}$. With a normalised pigment template, Q_w is a relative sensitivity-weighted photon irradiance. With a calibrated absorptance curve, it is absorbed photon irradiance per unit collection area at the plane where irradiance was measured.

Calculating photons $\text{s}^{-1} \text{receptor}^{-1}$ additionally requires receptor collection area, the solid angle subtended by the source, ocular transmission, and quantum efficiency. This function does not accept

or assume those quantities and therefore does not return an absolute receptor rate. Sensitivity is assigned zero outside the wavelength range supplied in S; endpoint values are not extrapolated.

Value

A scalar (when total = TRUE) or numeric vector of integrated per-bin contributions, in photons $\text{m}^{-2} \text{s}^{-1}$.

References

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

See Also

[species_LEF](#), [govardovskii_template](#), [species_brightness](#), [colour_jnd](#)

Examples

```
lam <- Naples$wv
S <- species_LEF("Danio rerio", receptor = "L-cone", lambda = lam)
quantum_catch(Naples$depth_0m, lam, S,
              input_unit = "umol/m2/s/nm")
```

`radiance2irradiance` *Convert radiance to irradiance under a geometry assumption.*

Description

Inverse of [irradiance2radiance](#). The geometry argument must match the one used during collection.

Usage

```
radiance2irradiance(
  L,
  geometry = c("lambertian", "scalar", "collimated", "custom"),
  solid_angle = NULL
)
```

Arguments

L	Radiance value(s) in $\text{W m}^{-2} \text{sr}^{-1}$. Vectorised.
geometry	One of "lambertian" (default), "scalar", "collimated", or "custom". See irradiance2radiance for details.
solid_angle	Finite solid angle in $(0, 4 \pi]$ steradians. Required for geometry = "custom" and rejected for other geometries.

Value

Irradiance in $W\ m^{-2}$.

Examples

```
radiance2irradiance(1)                # lambertian: E = pi
radiance2irradiance(1, "scalar")      # scalar: E = 2*pi
radiance2irradiance(5, "custom", solid_angle = 2) # E = 10
```

```
read_instrument_spectrum
```

Parse one instrument spectrum with `lightr`

Description

Uses **lightr** for file-format dispatch while retaining a strict, single-spectrum record that can represent irregular wavelength grids and signed processed values. This function does not claim that processed values are a physically valid [lux_spectrum](#); use [as_lux_spectrum](#) or [read_spectrum](#) for that step.

Usage

```
read_instrument_spectrum(
  path,
  measurement = NULL,
  value_scale = NULL,
  range = NULL,
  interpolate = FALSE,
  parser_args = list(),
  warning_policy = "error"
)
```

Arguments

<code>path</code>	Path to one spectral file.
<code>measurement</code>	Declared measurement represented by the processed values: one of "reflectance", "irradiance", "radiance", "transmittance", "absorbance", or "raw".
<code>value_scale</code>	Declared scale of the processed values: "fraction", "percent", "absolute", or "raw".
<code>range</code>	Two finite increasing wavelength limits in nm.
<code>interpolate</code>	Whether <code>lightr</code> should interpolate to integer nanometres. The default preserves native sampling in the instrument record.
<code>parser_args</code>	Named list of simple scalar arguments passed to <code>lightr</code> 's individual parser. Arguments controlling file discovery, wavelength range, or interpolation cannot be overridden.
<code>warning_policy</code>	Whether parser and metadata warnings should cause an error (the default) or be explicitly retained in the import record.

Value

A lux_instrument_spectrum containing wavelengths, processed values, declarations, instrument metadata, notices, and reproducibility context.

See Also

[read_spectrum](#), [as_lux_spectrum](#)

read_ocean_optics	<i>Strictly parse an Ocean Optics / Ocean Insight spectrum file</i>
-------------------	---

Description

Parses the plain-text format without assigning a physical quantity or unit. Every row must contain exactly two tab-separated numeric fields. Instrument values, including negatives, are preserved without clipping.

Usage

`read_ocean_optics(path)`

Arguments

path Path to the Ocean Optics .txt file.

Value

A data frame containing lambda and signal. Parsed header and import provenance are attached as "luxR.header" and "luxR.import" attributes.

read_spectrum	<i>Read a declared, calibrated spectrum with lightr</i>
---------------	---

Description

Parses one instrument file with [read_instrument_spectrum](#), then applies explicit scale and negative-value policies before constructing a [lux_spectrum](#). Typical native spectrometer grids are irregular, so interpolate = TRUE is normally required. Use read_instrument_spectrum(interpolate = FALSE) to inspect or retain native sampling without claiming it satisfies the lux_spectrum grid contract.

Usage

```
read_spectrum(  
  path,  
  measurement = NULL,  
  value_scale = NULL,  
  quantity = NULL,  
  unit = NULL,  
  calibration = NULL,  
  range = NULL,  
  interpolate = FALSE,  
  negative_policy = "error",  
  parser_args = list(),  
  warning_policy = "error"  
)
```

Arguments

path	Path to one spectral file.
measurement	Declared measurement represented by lightr's processed values. See read_instrument_spectrum .
value_scale	Declared input scale: "percent" or "fraction" for reflectance, and "absolute" for irradiance or radiance.
quantity	Explicit luxR physical quantity. It must match measurement.
unit	Explicit unit compatible with quantity.
calibration	Non-empty calibration identifier or description.
range	Required finite increasing wavelength limits in nm.
interpolate	Whether lightr should interpolate to integer nanometres. Defaults to FALSE; irregular native grids will fail construction with guidance to opt in or retain the instrument record.
negative_policy	Either "error" (default) or an explicit, recorded "zero" floor.
parser_args	Named list of simple scalar arguments passed to the lightr parser.
warning_policy	Whether lightr warnings should fail the import or be explicitly retained. Defaults to fail-fast behavior.

Value

A validated [lux_spectrum](#) with calibration, instrument metadata, parser provenance, notices, scaling, and resampling policy.

See Also

[read_instrument_spectrum](#), [from_ocean_optics](#), [read_ocean_optics](#), [from_trios](#)

Examples

```
## Not run:
x <- read_spectrum(
  "path/to/calibrated-irradiance.IRR8",
  measurement = "irradiance", value_scale = "absolute",
  quantity = "irradiance", unit = "W/m2/nm",
  calibration = "certificate CAL-2026-04",
  range = c(300, 700), interpolate = TRUE
)

## End(Not run)
```

read_trios

Read a TriOS RAMSES spectroradiometer data file

Description

Parses one or more spectra from a TriOS RAMSES ‘.dat’ text file. Each spectrum is preceded by a ‘Comment’ header line and terminated by a ‘[END]’ marker. Data rows contain three whitespace-separated fields: index, wavelength (nm), and spectral radiance ($\text{mW m}^{-2} \text{nm}^{-1} \text{sr}^{-1}$). Radiance values are converted to $\text{W m}^{-2} \text{nm}^{-1} \text{sr}^{-1}$. Negative instrument values are preserved; no clipping or other preprocessing is performed by this low-level reader.

Usage

```
read_trios(path)
```

Arguments

path Character. Path to the ‘.dat’ file.

Value

A data.frame with columns:

spectrum Character. Spectrum label from the Comment header.

lambda Numeric. Wavelength in nm.

radiance Numeric. Spectral radiance in $\text{W m}^{-2} \text{nm}^{-1} \text{sr}^{-1}$.

receptor_absorptance	<i>Photoreceptor absorptance from a normalised absorbance template (self-screening).</i>
----------------------	--

Description

Converts a normalised visual-pigment absorbance spectrum (e.g. from [govardovskii_template](#)) into the receptor's *absorptance* — the fraction of incident photons actually captured at each wavelength — given the axial optical density of the photoreceptor. At low optical density the absorptance is nearly proportional to the template; at high optical density the peak saturates toward 1 and the curve broadens ("self-screening"), so the realised spectral sensitivity is wider than the bare pigment absorbance. This is the sensitivity that should be fed to [quantum_catch](#) for long or dense photoreceptors (e.g. deep-sea rods).

Usage

```
receptor_absorptance(
  S,
  optical_density = NULL,
  alpha = NULL,
  path_length = NULL
)
```

Arguments

<code>S</code>	A data frame with columns <code>lambda</code> and <code>S</code> (a normalised absorbance template, e.g. from govardovskii_template), or a bare numeric vector of normalised absorbance values. Re-normalised to a peak of 1 internally, so the optical density refers to the peak.
<code>optical_density</code>	Peak axial optical density (base-10 absorbance), dimensionless. Typically ~0.3–0.5 for vertebrate cones, up to ~1 or more for long deep-sea rods. Supply this <i>or</i> (<code>alpha</code> and <code>path_length</code>), not both.
<code>alpha</code>	Peak absorption coefficient per unit length (e.g. per μm).
<code>path_length</code>	Outer-segment (path) length, in the same length unit as <code>alpha</code> .

Details

For a peak-normalised absorbance $S(\lambda)$ and axial optical density D (base-10 absorbance along the outer segment at the peak), the absorptance is

$$A(\lambda) = 1 - 10^{-D S(\lambda)} = 1 - e^{-\kappa S(\lambda)},$$

with $\kappa = D \ln 10$. Equivalently, supply the peak absorption coefficient a_{max} (alpha, per unit length) and the outer-segment path_length ℓ , giving $\kappa = a_{\text{max}} \ell$ (Johnsen 2012, ch. 4, eq. 4.8).

Value

If S is a data frame, a data frame with columns lambda and S holding the absorptance (so it drops straight into [quantum_catch](#)); if S is numeric, a numeric vector of absorptance values in $[0, 1]$.

References

Johnsen S (2012) The Optics of Life: A Biologist's Guide to Light in Nature. Princeton University Press. (Ch. 4: absorptance and self-screening.)

See Also

[govardovskii_template](#), [quantum_catch](#)

Examples

```
lam <- seq(300, 750, by = 1)
tmpl <- govardovskii_template(lam, lambda_max = 500)

# Realised sensitivity of a moderately dense cone (axial OD 0.4):
A <- receptor_absorptance(tmpl, optical_density = 0.4)

# A long deep-sea rod via Johnsen's absorption coefficient (0.064 / um):
A75 <- receptor_absorptance(tmpl, alpha = 0.064, path_length = 75)
```

reflectance_to_radiance

Convert spectral reflectance to spectral radiance

Description

Multiplies a reflectance spectrum by an illuminant spectrum bin-by-bin. For a lux_spectrum irradiance illuminant, a Lambertian reflector is assumed: radiance is $L = \rho E / \pi$, and the output unit gains the steradian term. For a radiance illuminant, the values are multiplied directly and its radiance unit is retained.

Usage

```
reflectance_to_radiance(reflectance, ...)

## S3 method for class 'numeric'
reflectance_to_radiance(reflectance, illuminant, lambda, ...)

## S3 method for class 'lux_spectrum'
reflectance_to_radiance(reflectance, illuminant, ...)
```

Arguments

reflectance	Numeric vector of reflectance values in [0, 1], or a lux_spectrum with quantity = "reflectance".
...	Ignored.
illuminant	Numeric vector of spectral irradiance/radiance, or a lux_spectrum with quantity "irradiance" or "radiance".
lambda	Numeric vector of wavelengths in nm (not needed when both inputs are lux_spectrum).

Value

Named numeric vector, or a lux_spectrum with quantity = "radiance" and a dimensionally compatible radiance unit. The numeric method performs only element-wise multiplication because numeric vectors do not carry irradiance/radiance dimensional metadata.

Examples

```
sp <- solar_irradiance("clear_noon")
lam <- sp$wavelength
refl <- rep(0.5, length(lam))
reflectance_to_radiance(refl, sp$irradiance, lam)
```

refracted_solar_angle *Refracted zenith angle of the sun beneath the surface*

Description

Snell's-law refraction of the solar beam entering water: $\theta_w = \arcsin(\sin \theta_{air}/n)$. As the in-air zenith approaches 90 degrees the refracted angle approaches the edge of Snell's window (about 48.6 degrees for $n = 1.333$); see [snells_window](#).

Usage

```
refracted_solar_angle(sun_zenith, n = 1.333)
```

Arguments

sun_zenith	In-air solar zenith angle in degrees, [0, 90].
n	Refractive index of water. Default 1.333.

Value

Refracted (in-water) solar zenith angle in degrees. Vectorised.

See Also

[snells_window](#), [underwater_polarization](#)

Examples

```
refracted_solar_angle(0)    # 0
refracted_solar_angle(90)  # ~48.6 (edge of Snell's window)
```

resample_spectrum	<i>Resample a spectrum to a new wavelength grid</i>
-------------------	---

Description

Interpolates spectral values onto a target wavelength grid using [approx](#) (linear, default) or [spline](#) (cubic). The most common use case is harmonising spectra from two instruments before arithmetic or comparison (e.g. aligning a TriOS RAMSES spectrum at 3.3 nm resolution with a Jerlov-based Kd vector at 25 nm steps).

Usage

```
resample_spectrum(
  x,
  lambda,
  from = NULL,
  method = c("linear", "cubic"),
  rule = 1
)

## S3 method for class 'lux_spectrum'
resample_spectrum(
  x,
  lambda,
  from = NULL,
  method = c("linear", "cubic"),
  rule = 1
)

## S3 method for class 'numeric'
resample_spectrum(
  x,
  lambda,
  from = NULL,
  method = c("linear", "cubic"),
  rule = 1
)
```

Arguments

x	A lux_spectrum object or numeric spectral-value vector.
lambda	Target wavelength grid in nm (numeric vector). When x is a lux_spectrum this argument is required.

from	Source wavelength grid in nm. Required when x is a numeric vector; ignored when x is a lux_spectrum (wavelengths are taken from x\$lambda).
method	Interpolation method: "linear" (default) or "cubic".
rule	Integer passed to approx controlling out-of-range behaviour. Default 1 (return NA); use 2 to extend with boundary values.

Details

Values are not extrapolated beyond the source wavelength range. By default (rule = 1) wavelengths outside that range receive NA; set rule = 2 to extend using the nearest boundary value instead. Because a lux_spectrum cannot contain non-finite values, its method raises a validation error when rule = 1 produces NA. The target grid for a lux_spectrum must also be strictly increasing and regular. The numeric method can be used to resample raw values from an irregular source grid before constructing a lux_spectrum.

Value

A lux_spectrum (when x is a lux_spectrum) or a named numeric vector, resampled onto lambda.

See Also

[lux_spectrum](#), [as_lux_spectrum](#)

Examples

```
# Resample a solar spectrum from 10 nm to 5 nm bins
sp <- solar_irradiance("clear_noon")
x <- as_lux_spectrum(sp)
x5 <- resample_spectrum(x, lambda = seq(300, 800, by = 5))
x5

# Align a Jerlov Kd vector to the Naples wavelength grid
lam_naples <- from_naples("0m")$lambda
lam_naples <- lam_naples[lam_naples >= 350 & lam_naples <= 700]
lam_jerlov <- seq(350, 700, by = 25)
Kd_coarse <- jerlov_Kd("II", lam_jerlov)
Kd_fine <- resample_spectrum(Kd_coarse, lambda = lam_naples,
                             from = lam_jerlov, rule = 1)
```

run_app

Launch the luxR Shiny explorer.

Description

Opens an interactive Shiny application for analysing underwater light environments. It has five guided tabs:

- **Depth Propagation** – attenuate a built-in or uploaded irradiance spectrum through a Jerlov water type and download the per-depth lux and photon-flux table.

- **Species Perception** – photoreceptor quantum catch for a chosen species under a selected bundled light-source condition at that source’s stated reference depth.
- **Colour discrimination** – Vorobyev-Osorio just-noticeable difference (JND) between two reflectances under the in-water light field.
- **Visibility** – Secchi depth, photic depth, and horizontal visual-range scenario estimate.
- **Detection** – explicitly unvalidated object-background contrast-threshold distance scenarios.

Requires the shiny package.

Usage

```
run_app(...)
```

Arguments

... Arguments passed to [runApp](#).

Value

Invisible NULL; called for its side effect of starting the app.

Examples

```
## Not run:
run_app()

## End(Not run)
```

scattering_angle	<i>Angle between a line of sight and the solar beam</i>
------------------	---

Description

The angle Θ between two directions given as (zenith, azimuth):

$$\cos \Theta = \cos \theta_1 \cos \theta_2 + \sin \theta_1 \sin \theta_2 \cos(\phi_1 - \phi_2).$$

For the single-scattering polarization model this is the relevant scattering angle (the degree of polarization peaks near $\Theta = 90^\circ$).

Usage

```
scattering_angle(view_zenith, view_azimuth, sun_zenith, sun_azimuth)
```

Arguments

view_zenith, view_azimuth
Viewing direction in degrees.

sun_zenith, sun_azimuth
Solar-beam direction in degrees (use [refracted_solar_angle](#) for the in-water zenith).

Value

Angle in degrees, $[0, 180]$. Vectorised over its arguments.

See Also

[underwater_polarization](#)

Examples

```
scattering_angle(view_zenith = 90, view_azimuth = 0,
                 sun_zenith = 0, sun_azimuth = 0) # 90
```

scotopic_lux	<i>Compute scotopic illuminance ($V'(\lambda)$ weighting).</i>
--------------	---

Description

Drop-in replacement for [irradiance2lux](#) using the CIE 1951 scotopic luminous efficiency function $V'(\lambda)$. The scotopic luminous efficacy $K_m' = 1700$ lm/W replaces the photopic $K_m = 683$ lm/W.

Usage

```
scotopic_lux(irradiance, ...)

## Default S3 method:
scotopic_lux(irradiance, ...)

## S3 method for class 'numeric'
scotopic_lux(
  irradiance,
  lambda,
  photonic = FALSE,
  molar_unit = "photons",
  total = TRUE,
  binwidth = NULL,
  integration = c("rectangle", "trapezoid"),
  verbose = FALSE,
  ...)
```

```

)

## S3 method for class 'lux_spectrum'
scotopic_lux(
  irradiance,
  total = TRUE,
  integration = c("rectangle", "trapezoid"),
  verbose = FALSE,
  ...
)

```

Arguments

irradiance	Spectral irradiance ($W\ m^{-2}\ nm^{-1}$ or $photons\ m^{-2}\ s^{-1}\ nm^{-1}$) at each wavelength bin, or a lux_spectrum object whose quantity is "irradiance". Photonic lux_spectrum inputs are converted to energy units before photometric integration; radiance and reflectance spectra are rejected.
...	Ignored.
lambda	Wavelength in nm (not needed for lux_spectrum).
photonic	Logical; if TRUE, irradiance is in photon or molar units. Default FALSE.
molar_unit	Molar unit string when photonic = TRUE (e.g. "umol"). Default "photons".
total	Logical; sum per-bin values into a scalar. Default TRUE.
binwidth	Wavelength bin width in nm. Inferred from a regular multi-bin wavelength grid when NULL; required for a single bin under rectangular integration. Must be NULL for trapezoidal integration.
integration	Integration method passed to irradiance2lux : "rectangle" (default) or "trapezoid".
verbose	Logical; emit diagnostic messages. Default FALSE.

Details

Scotopic illuminance uses the CIE 1951 scotopic luminous efficiency function $V'(\lambda)$, which peaks at 507 nm. The maximum scotopic luminous efficacy is $K'_m = 1700\ lm\ W^{-1}$:

$$E'_v = 1700 \int_{380}^{780} E(\lambda) V'(\lambda) d\lambda$$

Computed internally by calling [irradiance2lux](#) with LEF = CIE_scotopic and applying the efficacy ratio 1700/683.

Value

A scalar scotopic lux value (when total = TRUE) or numeric vector.

Note

The scotopic/photopic crossover occurs near 10 lx. Below that threshold — twilight, moonlight, deep water — scotopic readings better reflect perceived brightness.

Examples

```
data(Naples)
scotopic_lux(Naples$depth_10m, Naples$swv,
             photonic = TRUE, molar_unit = "umol")
```

secchi_depth	<i>Secchi depth from diffuse attenuation coefficient(s).</i>
--------------	--

Description

Estimates the Secchi disk visibility depth using the Tyler (1968) approximation $z_S \approx 1.7/K_d$. When a spectral irradiance is supplied the function first computes an irradiance-weighted mean K_d across the visible range before applying the approximation.

Usage

```
secchi_depth(Kd, lambda = NULL, irradiance = NULL, spectrum = NULL)
```

Arguments

Kd	Diffuse attenuation coefficient(s) in 1/m.
lambda	Optional wavelength vector (nm) matching Kd.
irradiance	Optional spectral irradiance for PAR-weighted mean Kd.
spectrum	Optional lux_spectrum; when supplied its E and lambda slots replace the irradiance and lambda arguments.

Value

Secchi depth in metres.

References

Tyler JE (1968) Limnol. Oceanogr. 13:1-6. Preisendorfer RW (1986) Limnol. Oceanogr. 31:909-926.

See Also

[visual_range](#), [jerlov_Kd](#), [photic_depth](#)

Examples

```
secchi_depth(jerlov_Kd("IA", lambda = 490))
```

single_scattering_albedo

Single-scattering albedo.

Description

The fraction of beam attenuation due to scattering rather than absorption, $\omega_0 = b/(a + b) = b/c$ (Johnsen 2012, ch. 5). Values near 1 indicate a strongly scattering, weakly absorbing medium (e.g. fog or milk); values near 0 a strongly absorbing one.

Usage

```
single_scattering_albedo(absorption, scattering)
```

Arguments

absorption	Absorption coefficient a (1/m). Vectorised.
scattering	Scattering coefficient b (1/m). Vectorised.

Value

Single-scattering albedo ω_0 (dimensionless, 0-1).

References

Johnsen S (2012) The Optics of Life. Princeton University Press, ch. 5.

See Also

[beam_attenuation](#)

Examples

```
single_scattering_albedo(absorption = 0.1, scattering = 0.05) # 0.333
```

snells_window

Snell's window half-angle.

Description

Underwater, the entire above-water hemisphere is refracted into a cone — the "Snell's window" or "optical manhole" — of half-angle $\theta_c = \arcsin(1/n)$ (Johnsen 2012, ch. 5). For seawater ($n \approx 1.333$) this is about 48.6 degrees, so the whole sky is compressed into a ~97 degree cone overhead.

Usage

```
snells_window(n = 1.333)
```

Arguments

n Refractive index of the water relative to air. Default 1.333. Vectorised.

Value

Half-angle of Snell's window, in degrees.

References

Johnsen S (2012) The Optics of Life. Princeton University Press, ch. 5.

See Also

[fresnel_reflectance](#), [wavelength_in_medium](#)

Examples

```
snells_window()            # ~48.6 degrees
```

solar_irradiance	<i>Reference solar spectral irradiance under standard conditions.</i>
------------------	---

Description

Returns a reference spectrum from the bundled `solar_spectra` dataset, based on ASTM G173-03 global-tilt values scaled to several representative illumination conditions. Intended for use as the reference shape in [broadband2spectrum](#) and [lux2irradiance](#).

Usage

```
solar_irradiance(condition = "clear_noon")
```

Arguments

condition Character string naming the illumination condition. One of the five conditions listed in Details.

Details

Available conditions:

"clear_noon" Full-sun midday irradiance (ASTM G173-03 GTI).

"clear_dawn" Clear sky near sunrise/sunset (~35% of noon).

"overcast" Overcast diffuse sky (~22% of noon).

"underwater_1m" ASTM GTI attenuated through 1 m of Jerlov IA water.

"underwater_10m" ASTM GTI attenuated through 10 m of Jerlov IA water.

The historical underwater sources span 300–800 nm and were generated with constant endpoint extension beyond the bundled Jerlov dataset's supported 350–700 nm domain. Their attributes explicitly record this assumption and the Jerlov table checksum so it is not a silent extrapolation path.

Value

A data frame with columns wavelength (nm) and irradiance ($W m^{-2} nm^{-1}$). Attributes record condition, reference_depth_m (metres relative to the water surface), and reference_medium ("air" or "water").

Examples

```
solar_irradiance("clear_noon")
broadband2spectrum(480, unit = "W", spectrum = solar_irradiance("overcast"))
```

solar_spectra	<i>ASTM G173-03 reference solar spectral irradiance.</i>
---------------	--

Description

A named list of reference spectral irradiance data frames representing five standard illumination conditions, derived from the ASTM G173-03 global-tilt spectrum. Used by [solar_irradiance](#).

Usage

```
solar_spectra
```

Format

A named list of five data frames, each with columns:

wavelength Wavelength in nm (300 to 800, step 10).

irradiance Spectral irradiance in $W m^{-2} nm^{-1}$.

Names: clear_noon, clear_dawn, overcast, underwater_1m, underwater_10m. Each data frame has attributes condition, reference_depth_m, and reference_medium. Reference depth is measured in metres from the water surface; the three atmospheric spectra use depth 0 in "air", and the underwater spectra use depths 1 and 10 in "water", respectively.

Source

ASTM G173-03(2012) Standard Tables for Reference Solar Spectral Irradiances. American Society for Testing and Materials, West Conshohocken, PA.

Examples

```
solar_irradiance("clear_noon")
```

species_brightness	<i>Per-species sensitivity-weighted photon irradiance.</i>
--------------------	--

Description

Convenience wrapper around [quantum_catch](#) and [species_LEF](#) that returns quantum catches for all (or one) receptor class of a named species.

Usage

```
species_brightness(
  irradiance,
  lambda,
  species,
  input_unit = NULL,
  channel = "by_receptor",
  binwidth = NULL
)
```

Arguments

irradiance	Non-negative spectral irradiance vector, one value per wavelength bin, in the exact unit declared by input_unit.
lambda	Wavelength bin centres in nm.
species	Species name. Must appear in species_sensitivities.
input_unit	Exact unit of irradiance; see quantum_catch . No photon or molar unit is inferred from a bare numeric vector.
channel	One of: • "by_receptor" — named numeric vector, one value per receptor class; • "all" — scalar sum across all receptor classes; • a specific receptor name (e.g. "L-cone") — single scalar.
binwidth	Wavelength bin width in nm. Inferred from lambda spacing if NULL (default).

Value

Named numeric vector, scalar, or single numeric depending on channel.

See Also

[quantum_catch](#), [species_LEF](#), [colour_jnd](#)

Examples

```
idx <- Naples$wv >= 400 & Naples$wv <= 700
lam <- Naples$wv[idx]
irr <- Naples$depth_0m[idx]
species_brightness(irr, lam, "Homo sapiens",
                  input_unit = "umol/m2/s/nm",
                  channel = "by_receptor")
```

species_channel_support

Species visual-channel support decisions.

Description

Records whether every bundled species has a supported default chromatic and achromatic model. Unsupported combinations remain explicit rather than being inferred from receptor-role labels.

Usage

```
species_channel_support
```

Format

A data frame with 16 rows and 5 columns:

species Scientific name matching [species_sensitivities](#).

channel_role Either "chromatic" or "achromatic".

status Either "supported" or "unavailable".

reason Evidence summary or reason no default is available.

source Literature citation or explicit package review decision.

Details

Every supported row must correspond to exactly one default in [species_channels](#). Every unavailable row must have no default. Dataset generation and runtime validation enforce this invariant.

Examples

```
data(species_channel_support)
subset(species_channel_support, species == "Homo sapiens")
```

species_channels	<i>Validated visual-channel memberships for selected species.</i>
------------------	---

Description

Defines cited receptor combinations that package models may use by default. Receptors marked with a role in [species_sensitivities](#) are not model-eligible unless they occur in this table.

Usage

```
species_channels
```

Format

A data frame with 13 rows and 7 columns:

species Scientific name matching `species_sensitivities$species`.

channel Stable channel identifier within a species.

channel_role Biological role from the controlled vocabulary.

receptor Receptor member matching a species/receptor pair in `species_sensitivities`.

weight Finite positive channel membership weight. Chromatic JND currently uses receptor-noise weighting rather than this field.

is_default Whether this is the default channel for its species and role. A species can have at most one default channel per role.

source Literature citation supporting the receptor set.

Details

The curated set contains default chromatic channels for *Homo sapiens*, *Danio rerio*, and *Apis mellifera*, plus behavioural achromatic channels for adult light-adapted *Danio rerio*, *Apis mellifera*, and *Drosophila melanogaster*. Species with incomplete or ambiguous receptor sets are deliberately absent and therefore fail rather than falling back to every known receptor.

References

Krauss A, Neumeyer C (2003) Wavelength dependence of the optomotor response in zebrafish (*Danio rerio*). *Vision Research* 43:1275-1284. doi:[10.1016/S00426989\(03\)000907](#)

Niggebrugge C, Hempel de Ibarra N (2003) Colour-dependent target detection by bees. *Journal of Comparative Physiology A* 189:915-918. doi:[10.1007/s0035900304663](#)

Yamaguchi S, Wolf R, Desplan C, Heisenberg M (2008) Motion vision is independent of color in *Drosophila*. *PNAS* 105:4910-4915. doi:[10.1073/pnas.0711484105](#)

Examples

```
data(species_channels)
subset(species_channels, species == "Homo sapiens")
```

species_LEF	<i>Spectral sensitivity for a named species and photoreceptor class.</i>
-------------	--

Description

Looks up peak wavelength and chromophore from the bundled `species_sensitivities` table, then applies the Govardovskii (2000) template to return a normalised sensitivity curve.

Usage

```
species_LEF(species, receptor = NULL, lambda = seq(300, 700, by = 1))
```

Arguments

<code>species</code>	Species name (character). Must match a row in <code>species_sensitivities\$species</code> . Case-sensitive.
<code>receptor</code>	Photoreceptor class (e.g. "L-cone", "rod"). If NULL (default), returns all receptor classes as a named list.
<code>lambda</code>	Wavelength grid in nm. Default <code>seq(300, 700, by = 1)</code> .

Value

A data frame (single receptor) or named list of data frames (all receptors). Each data frame has columns `lambda` and `S`.

Examples

```
species_LEF("Homo sapiens", receptor = "L-cone")
species_LEF("Danio rerio")
```

<code>species_sensitivities</code>	<i>Photoreceptor spectral sensitivities for selected species.</i>
------------------------------------	---

Description

Peak wavelengths and chromophore types for visual photoreceptors of representative species from vertebrate and invertebrate taxa. Used by [species_LEF](#) and [species_brightness](#).

Usage

```
species_sensitivities
```

Format

A data frame with 27 rows and 6 columns:

species Scientific name.

receptor Receptor class (e.g. "L-cone", "rod").

lambda_max Peak absorbance wavelength in nm.

chromophore "A1" (11-cis retinal) or "A2" (11-cis 3,4-didehydroretinal).

channel_role Primary biological role from the controlled vocabulary "chromatic", "achromatic", "irradiance", "polarization", or "fluorescence".

source Literature citation for the lambda_max value.

Details

Each row's lambda_max is taken from the primary-literature citation recorded in its source column; luxR reconstructs full sensitivity curves from these peaks using the Govardovskii et al. (2000) visual-pigment template (see [govardovskii_template](#)).

Per-receptor lambda_max sources: Bowmaker & Dartnall (1980) and Bailes & Lucas (2013) (*Homo sapiens*); Robinson et al. (1993) (*Danio rerio*); Peitsch et al. (1992) (*Apis mellifera*); Salcedo et al. (1999) (*Drosophila melanogaster*); Hart et al. (2004) (*Callorhinchus milii*); Yokoyama et al. (1999) (*Latimeria chalumnae*); Yoshiura et al. (2011) (*Xenopus laevis*); Hawryshyn et al. (2003) (*Oncorhynchus mykiss*).

A role label does not by itself validate a receptor for a model. Complete, cited channel memberships are recorded separately in [species_channels](#). The generic mantis-shrimp record and *Aequorea victoria* GFP previously included in this table were removed because they are not supported visual-pigment channel records.

References

Govardovskii VI, Fyhrquist N, Reuter T, Kuzmin DG, Donner K (2000) In search of the visual pigment template. *Visual Neuroscience* 17:509-528.

Full citation keys and their verification status are recorded in `data-raw/species_source_map.csv`. The legacy Hart et al. (2004) and Yoshiura et al. (2011) attributions remain explicitly unverified pending primary-source confirmation; the values are not silently relabelled.

Robinson J, Schmitt EA, Hárosi FI, Reece RJ, Dowling JE (1993). "Zebrafish Ultraviolet Visual Pigment: Absorption Spectrum, Sequence, and Localization." *Proceedings of the National Academy of Sciences*, **90**(13), 6009–6012. doi:10.1073/pnas.90.13.6009. 2026-06-04.

Hawryshyn CW, Martens G, Allison WT, Anholt BR (2003). "Regeneration of Ultraviolet-Sensitive Cones in the Retinal Cone Mosaic of Thyroxin-Challenged Post-Juvenile Rainbow Trout (*Oncorhynchus Mykiss*)." *Journal of Experimental Biology*, **206**(15), 2665–2673. ISSN 0022-0949. doi:10.1242/jeb.00470. 2026-06-04.

Examples

```
data(species_sensitivities)
subset(species_sensitivities, species == "Danio rerio")
```

species_sensitivity_matrix

Photoreceptor sensitivity matrix for colourvision

Description

Assembles a species' validated chromatic receptors into the sensitivity matrix C that **colourvision** models consume (a data frame whose first column is wavelength and whose remaining columns are one photoreceptor each). The receptors and their Govardovskii templates are exactly those [colour_jnd](#) uses, so a colourvision model built on this matrix reproduces luxR's own quantum catches.

Usage

```
species_sensitivity_matrix(
  species,
  lambda = seq(300, 700, 1),
  receptor = NULL,
  weight = c("photon", "energy")
)
```

Arguments

species	Character scalar naming a species in <code>species_sensitivities</code> with at least one validated chromatic receptor.
lambda	Numeric wavelength grid (nm); must be finite, strictly increasing, and unique. Defaults to <code>seq(300, 700, 1)</code> , colourvision's default nm.
receptor	Optional character vector selecting a subset of the species' validated chromatic receptors; passed through to the same channel resolution colour_jnd uses.
weight	One of "photon" (default, columns $S\lambda$) or "energy" (columns S).

Details

luxR integrates **photon** catch (proportional to $E\lambda$) while colourvision integrates energy ($\int IRC$). With `weight = "photon"` (the default) the columns are the sensitivities multiplied by wavelength ($S\lambda$), so colourvision's energy integral reproduces luxR's photon catch; the proportionality constant cancels in the log RNL model. Use `weight = "energy"` for the raw sensitivities.

Value

A data.frame with a `wl` column plus one column per receptor, named and ordered as luxR's channel definitions.

See Also

[as_colourvision](#), [colour_jnd](#), `colourvision::RNLmodel`

Examples

```
## Not run:
C <- species_sensitivity_matrix("Apis mellifera")

## End(Not run)
```

surface_transmittance *Fraction of downwelling irradiance transmitted across the water surface.*

Description

The fraction of above-surface downwelling irradiance that enters the water after Fresnel reflection at a flat air-water interface — the surface term of an above-vs-below-water light budget: $E_{\text{below}} \approx \tau E_{\text{above}}$ (Johnsen 2012, ch. 5).

Usage

```
surface_transmittance(angle = NULL, n = 1.333, source = c("direct", "diffuse"))
```

Arguments

angle	Solar/incidence angle from the surface normal, in degrees (0-90). Required for source = "direct"; ignored for "diffuse".
n	Refractive index of water relative to air. Default 1.333.
source	"direct" (collimated beam at angle) or "diffuse" (isotropic sky).

Details

For a "direct" (collimated, e.g. solar-beam) source at a given incidence angle, $\tau = 1 - R(\theta)$. For a "diffuse" (isotropic sky) source it is the cosine-weighted hemispherical average of $1 - R(\theta)$, about 0.93 for seawater (i.e. ~7% is reflected). This is the planar-irradiance transmittance; in-water *radiance* is additionally boosted by n^2 because the refracted beam is compressed into [snells_window](#).

Value

Transmitted fraction of downwelling irradiance (0-1).

References

Johnsen S (2012) The Optics of Life. Princeton University Press, ch. 5.

See Also

[fresnel_reflectance](#), [snells_window](#)

Examples

```

surface_transmittance(angle = 0)           # overhead sun: ~0.98
surface_transmittance(source = "diffuse")  # isotropic sky: ~0.93

```

transmittance	<i>Beer-Lambert transmittance over a path.</i>
---------------	--

Description

Fraction of collimated light transmitted through a path of length `path_length` in a medium with attenuation coefficient: $T = e^{-c\ell}$ (Johnsen 2012, ch. 4). The fraction *absorbed* (or attenuated) is $1 - T$; note that absorbances add over stacked paths while transmittances multiply.

Usage

```
transmittance(coefficient, path_length)
```

Arguments

<code>coefficient</code>	Attenuation (or absorption) coefficient (1/length). Vectorised.
<code>path_length</code>	Path length in the same length unit. Vectorised.

Value

Transmittance (0-1).

References

Johnsen S (2012) The Optics of Life. Princeton University Press, ch. 4.

See Also

[absorbance](#), [beam_attenuation](#)

Examples

```

transmittance(coefficient = 0.15, path_length = 10)
1 - transmittance(0.064, 10)  # hatchetfish rod: ~47% absorbed

```

underwater_polarization

Tunable Rayleigh-like approximation to underwater polarization

Description

A lightweight, analytic single-scattering model of linear polarization in the water column. The degree of polarization follows the Rayleigh-like form

$$DoP(\Theta) = DoP_{max} \frac{\sin^2 \Theta}{1 + \cos^2 \Theta},$$

where Θ is the scattering angle between the line of sight and the refracted solar beam ([scattering_angle](#)); polarization is maximal at $\Theta = 90^\circ$ and vanishes toward the sun and anti-sun. With depth, multiple scattering randomizes the field, so the model relaxes toward an asymptotic floor:

$$DoP(z) = DoP_{asym} + (DoP(\Theta) - DoP_{asym}) e^{-z/z_p}.$$

The e-vector (angle of polarization) is perpendicular to the scattering plane.

Usage

```
underwater_polarization(
    view_zenith,
    view_azimuth = 0,
    sun_zenith,
    sun_azimuth = 0,
    depth = 0,
    dop_max = 0.4,
    dop_asym = 0.03,
    z_p = 20,
    n = 1.333
)
```

Arguments

view_zenith, view_azimuth	Viewing direction in degrees (zenith: 0 = up, 90 = horizontal, 180 = down). Vectorised. Scalars are broadcast; implicit partial recycling is rejected.
sun_zenith	In-air solar zenith angle in degrees. Refracted internally via refracted_solar_angle .
sun_azimuth	Solar azimuth in degrees. Default 0.
depth	Depth in metres (≥ 0). Default 0.
dop_max	Maximum (surface, 90-degree-scattering) degree of polarization. Default 0.4 is an illustrative value at the upper end of the approximately 0.25–0.40 maxima reported at 15 m in clear tropical marine water by Cronin and Shashar (2001). Because this model defines dop_max at depth zero, that observation does not constitute a calibration of the default. The value is not transferable to other water types.

dop_asym	Asymptotic deep-field degree of polarization. Default 0.03 is a heuristic illustrative floor, not an empirically calibrated constant.
z_p	Depth scale (m) over which DoP relaxes toward dop_asym. Default 20 is a heuristic illustrative scale, not an empirical attenuation length. Site-specific use requires calibration.
n	Refractive index of water. Default 1.333 is a conventional visible- light approximation. Refractive index varies with wavelength, temperature, and salinity.

Details

This is an explicitly uncalibrated, first-order approximation for teaching, geometry exploration, and field intuition. It is not a vector radiative-transfer solution and must not be interpreted as a quantitative prediction without calibration to measurements from the relevant site, wavelength, water type, surface state, and viewing geometry. It omits the wavelength dependence of scattering and refraction, particulate and dissolved constituents, surface waves and polarized skylight, bottom reflection, and a physical multiple-scattering calculation.

Value

A data frame with one row per viewing direction and columns `scattering_angle` (deg), `dop` (0–1), and `aop` (deg, $(-90, 90]$; NA when looking along the solar beam). The data frame has a `"luxR.polarization"` attribute recording model status, model version, parameters, and the validity statement.

References

- Cronin TW, Shashar N (2001) The linearly polarized light field in clear, tropical marine waters. *Journal of Experimental Biology* 204:2461-2467.
- Johnsen S (2012) *The Optics of Life*. Princeton University Press, ch. 8.

See Also

[degree_of_polarization](#), [scattering_angle](#), [refracted_solar_angle](#), [snells_window](#)

Examples

```
# DoP across viewing zenith with the sun overhead, near the surface:
underwater_polarization(view_zenith = seq(0, 90, 15), view_azimuth = 0,
                        sun_zenith = 0, depth = 1)
```

unit_expression	<i>Render a unit string as a plotmath expression</i>
-----------------	--

Description

Formats a unit from the controlled vocabulary as a plotmath language object with SI negative exponents, suitable for passing directly as the xlab or ylab of a base graphics plot.

Usage

```
unit_expression(unit)
```

Arguments

unit	A single unit string from the controlled vocabulary. See lux_spectrum for the valid values.
------	---

Value

A language object suitable for use as a plot label.

See Also

[unit_label](#) for the plain-text equivalent.

Examples

```
plot(400:700, rep(1, 301), type = "l",  
     xlab = "Wavelength (nm)", ylab = unit_expression("W/m2/nm"))
```

unit_label	<i>Render a unit string for display</i>
------------	---

Description

Formats a unit from the controlled vocabulary as a human-readable label using SI negative exponents, for use in table headers, user interfaces, and prose. The canonical unit strings stored in `lux_spectrum$unit` are deliberately ASCII ("umol/m2/s/nm") so they remain stable identifiers; this function is the display counterpart.

Usage

```
unit_label(unit)
```

Arguments

unit A single unit string from the controlled vocabulary. See [lux_spectrum](#) for the valid values.

Value

A length-1 character vector holding the rendered label, e.g. $\mu\text{mol m}^{-2} \text{s}^{-1} \text{nm}^{-1}$ for "umol/m2/s/nm".

See Also

[unit_expression](#) for the plotmath equivalent used in plot axis labels.

Examples

```
unit_label("umol/m2/s/nm")
unit_label("W/m2/sr/nm")
```

visual_range	<i>Heuristic horizontal visual-range estimate in water.</i>
--------------	---

Description

Applies the Koschmieder–Berek law to estimate the maximum horizontal sighting distance in water: $V = -\ln(C_t)/c$ where C_t is the inherent contrast threshold and c is the beam attenuation coefficient. Because c is rarely measured directly, a conversion factor `kd_to_c` maps diffuse attenuation K_d to c .

Usage

```
visual_range(Kd, contrast_threshold = 0.02, kd_to_c = 1.5, beam_c = NULL)
```

Arguments

Kd Diffuse attenuation coefficient in 1/m (scalar or vector).

contrast_threshold Inherent contrast threshold of the visual system or camera (dimensionless, typically 0.01–0.05). Default 0.02.

kd_to_c Ratio c/K_d . Typical values: 1.2–1.5 for clear oceanic water, 2–4 for coastal/turbid water. Default 1.5. Ignored when `beam_c` is supplied.

beam_c Optional measured beam attenuation coefficient c (1/m), e.g. from `beam_attenuation()`. When supplied it is used directly instead of the $K_d * kd_to_c$ approximation.

Details

This is a contrast-threshold scenario estimate, not an empirically validated prediction of detection by a particular observer or camera.

The coefficient that governs horizontal sighting distance is the beam attenuation coefficient c , not the diffuse attenuation K_d (Johnsen 2012, ch. 5: as you move away from an object, veiling light replaces it at a rate equal to c). If beam_c is not given, this function approximates $c = K_d \times kd_to_c$. Note that $c > K_d$ always, so treating K_d as c ($kd_to_c = 1$) is an upper bound that overestimates range. The model assumes **horizontal** viewing; for an upward-looking observer the effective coefficient is $c - K_L$ (smaller, so ranges are longer), which is not modelled here.

Value

Heuristic visual-range estimate in metres. Same length as Kd.

References

Duntley SQ (1963) Light in the sea. J. Opt. Soc. Am. 53:214–233.
 Zaneveld JRV, Pegau WS (2003) Robust underwater visibility parameter. Opt. Express 11:2997–3009.
 Johnsen S (2012) The Optics of Life: A Biologist's Guide to Light in Nature. Princeton University Press. (Ch. 5: sighting distance and c .)

Examples

```
# Visual range for Jerlov IA water at 490 nm
visual_range(jerlov_Kd("IA", lambda = 490))

# Compare water types
types <- c("I", "IA", "II", "C2")
sapply(types, function(t) visual_range(jerlov_Kd(t, lambda = 490)))
```

W2mol_spec_irradiance *Convert energy spectral irradiance to molar photon flux.*

Description

Converts $\text{W m}^{-2} \text{ nm}^{-1}$ spectral irradiance to a photon-count flux expressed in molar units (mol, mmol, or μmol photons $\text{m}^{-2} \text{ s}^{-1} \text{ nm}^{-1}$). The inverse operation is [n2W_spec_irradiance](#). The same per-bin conversion factor applies to spectral radiance ($\text{W m}^{-2} \text{ sr}^{-1} \text{ nm}^{-1}$).

Usage

```
W2mol_spec_irradiance(W, ...)

## S3 method for class 'numeric'
W2mol_spec_irradiance(W, lambda, molar_unit = "umol", ...)

## S3 method for class 'lux_spectrum'
W2mol_spec_irradiance(W, molar_unit = "umol", ...)
```

Arguments

W	Spectral irradiance in $W\ m^{-2}\ nm^{-1}$, or a lux_spectrum with an energy irradiance or radiance unit.
...	Ignored.
lambda	Wavelength in nm for each bin.
molar_unit	Target molar unit: "umol", "mmol", or "mol". Default "umol".

Details

The inverse of [n2W_spec_irradiance](#): photon flux is recovered via $N = E\lambda/(hc)$, then divided by Avogadro's constant scaled to the requested molar unit:

$$\Phi(\lambda) = \frac{E(\lambda)\lambda}{hc N_A} \times s$$

where $N_A = 6.022 \times 10^{23}\ mol^{-1}$ and $s = 10^6$ for "umol", 10^3 for "mmol", or 1 for "mol".

Value

Numeric vector of photon flux in the requested molar unit per nm, or a lux_spectrum with the updated unit. Radiance outputs retain the "/sr" term.

Examples

```
sp <- solar_irradiance("clear-noon")
W2mol_spec_irradiance(sp$irradiance, sp$wavelength, molar_unit = "umol")
```

W2photon

Convert watts to photon counts at a given wavelength.

Description

Inverse of [photon2W](#).

Usage

```
W2photon(W, lambda)
```

Arguments

W	Spectral irradiance in $W\ m^{-2}\ nm^{-1}$.
lambda	Wavelength(s) in nm.

Value

Numeric vector of photon counts in $photons\ m^{-2}\ s^{-1}\ nm^{-1}$.

Examples

```
W2photon(1, 555)
```

wavelength_in_medium	<i>Wavelength of light inside an optical medium.</i>
----------------------	--

Description

Returns the in-medium wavelength after accounting for the refractive index. Frequency (and photon energy) are unchanged; only the spatial period shortens.

Usage

```
wavelength_in_medium(lambda, n)
```

Arguments

lambda	Wavelength in vacuum or air (nm, or any consistent unit). Vectorised.
n	Refractive index of the medium (dimensionless). Must be ≥ 1 .

Value

Wavelength in the medium, in the same unit as lambda.

Examples

```
wavelength_in_medium(500, n = 1.336) # seawater: ~374 nm in medium
wavelength_in_medium(c(400, 500, 600), n = 1.5)
```

weber_contrast	<i>Weber contrast</i>
----------------	-----------------------

Description

Computes the Weber contrast: $(I_{target} - I_{background}) / I_{background}$. Positive values indicate a target brighter than its background; negative values indicate a darker target. Commonly used for isolated objects on a uniform background.

Usage

```
weber_contrast(I_target, I_background)
```

Arguments

I_target	Luminance or radiance of the target (scalar or vector).
I_background	Luminance or radiance of the background (scalar or vector). Must be non-zero.

Value

Contrast value(s); unbounded but typically in $[-1, \infty)$.

See Also

[michelson_contrast()]

Examples

```
weber_contrast(I_target = 8, I_background = 4)
```

Index

* datasets

- CIE1931, [20](#)
- CIE_scotopic, [19](#)
- jerlov_types, [44](#)
- Naples, [51](#)
- solar_spectra, [76](#)
- species_channel_support, [78](#)
- species_channels, [79](#)
- species_sensitivities, [80](#)
- *.lux_spectrum(lux_spectrum), [46](#)
- +.lux_spectrum(lux_spectrum), [46](#)
- .lux_spectrum(lux_spectrum), [46](#)
- /.lux_spectrum(lux_spectrum), [46](#)
- [.lux_spectrum(lux_spectrum), [46](#)

absorbance, [5](#), [84](#)

angle_of_polarization, [6](#), [25](#)

approx, [68](#), [69](#)

as.data.frame.lux_spectrum
(lux_spectrum), [46](#)

as_colourvision, [6](#), [82](#)

as_lux_spectrum, [8](#), [11–13](#), [34](#), [35](#), [47](#), [61](#),
[62](#), [69](#)

as_reflector_spct, [10](#), [13](#)

as_rspec, [7](#), [11](#), [13](#)

as_source_spct, [11](#), [12](#)

attenuate_depth, [13](#)

attenuate_spectrum, [14](#), [31](#)

band_irradiance, [15](#), [15](#), [53](#)

beam_attenuation, [17](#), [74](#), [84](#)

broadband2spectrum, [18](#), [75](#)

CIE1931, [20](#)

CIE_scotopic, [19](#)

colour_jnd, [21](#), [26](#), [29](#), [38](#), [39](#), [60](#), [78](#), [82](#)

contrast_at_distance, [22](#), [27](#), [30](#), [39](#)

convert_unit, [24](#), [47](#)

degree_of_polarization, [6](#), [24](#), [56](#), [86](#)

detectability, [25](#)

detection_range, [23](#), [25](#), [27](#), [27](#), [39](#)

fit_Kd, [30](#)

fit_sensitivity, [31](#)

fresnel_reflectance, [32](#), [75](#), [83](#)

from_naples, [10](#), [33](#)

from_ocean_optics, [34](#), [63](#)

from_solar, [10](#), [35](#), [45](#), [46](#)

from_trios, [11](#), [34](#), [35](#), [63](#)

ft2m, [36](#), [48](#)

govardovskii_template, [32](#), [37](#), [59](#), [60](#), [65](#),
[66](#), [81](#)

inherent_contrast, [23](#), [25](#), [27](#), [30](#), [38](#)

irradiance2lux, [39](#), [48](#), [50](#), [53](#), [71](#), [72](#)

irradiance2radiance, [27](#), [29](#), [41](#), [60](#)

jerlov_Kd, [15](#), [31](#), [42](#), [44–46](#), [73](#)

jerlov_types, [44](#)

light_at_depth, [44](#), [57](#)

lux2irradiance, [41](#), [47](#), [75](#)

lux_spectrum, [4](#), [7](#), [8](#), [10–12](#), [46](#), [53](#), [61–63](#),
[69](#), [87](#), [88](#)

luxR (luxR-package), [4](#)

luxR-package, [4](#)

m2ft, [36](#), [48](#)

michelson_contrast, [49](#)

n2W_spec_irradiance, [40](#), [49](#), [53](#), [89](#), [90](#)

Naples, [34](#), [51](#)

optimize, [31](#)

par_fraction, [51](#), [53](#)

par_irradiance, [17](#), [40](#), [41](#), [50](#), [52](#)

photoc_depth, [54](#), [73](#)

photon2W, [54](#), [90](#)

`plot.list (lux_spectrum)`, 46
`plot.lux_spectrum (lux_spectrum)`, 46
`plot_spectra`, 47, 55
`polarization_contrast`, 55
`print.lux_spectrum (lux_spectrum)`, 46
`propagate_depth`, 56
`propagate_spectrum`, 15, 17, 31, 44, 46, 57

`quantum_catch`, 22, 59, 65, 66, 77, 78

`radiance2irradiance`, 42, 60
`read_instrument_spectrum`, 61, 62, 63
`read_ocean_optics`, 34, 62, 63
`read_spectrum`, 34, 36, 61, 62, 62
`read_trios`, 35, 36, 64
`receptor_absorptance`, 65
`reflectance_to_radiance`, 66
`refracted_solar_angle`, 67, 71, 85, 86
`resample_spectrum`, 68
`run_app`, 69
`runApp`, 70

`scattering_angle`, 70, 85, 86
`scotopic_lux`, 19, 41, 71
`secchi_depth`, 73
`single_scattering_albedo`, 18, 74
`snells_window`, 32, 33, 67, 74, 83, 86
`solar_irradiance`, 35, 75, 76
`solar_spectra`, 51, 76
`species_brightness`, 22, 60, 77, 80
`species_channel_support`, 78
`species_channels`, 21, 26, 28, 38, 78, 79, 81
`species_LEF`, 22, 59, 60, 77, 78, 80, 80
`species_sensitivities`, 79, 80
`species_sensitivity_matrix`, 6, 7, 82
`spline`, 68
`summary.lux_spectrum (lux_spectrum)`, 46
`surface_transmittance`, 45, 57, 83

`transmittance`, 5, 18, 84

`underwater_polarization`, 25, 33, 56, 67, 71, 85
`unit_expression`, 87, 88
`unit_label`, 87, 87

`visual_range`, 18, 23, 27, 30, 73, 88

`W2mol_spec_irradiance`, 49, 89
`W2photon`, 55, 90

`wavelength_in_medium`, 75, 91
`weber_contrast`, 39, 56, 91